רשתות תקשורת – פרויקט גמר

מגישים :
רון אמסלם : 326029600

רואי מעודד:207302340

גל אשכנזי: 212032957

עומרי סבג:322925041

תוכן עניינים

רשתות תקשורת – פרויקט גמר 1

חלק ראשון – שאלות פתוחות 3-7

חלק שני – ניתוח המאמר הראשון 8-11

חלק שני – ניתוח המאמר השני 12-15

חלק שני- ניתוח מאמר השלישי 16-19

חלק שלישי – ניתוח הגרפים, שיעורים מסקנות ותוצאות :

Wikipedia 20-43

Microsoft Edge 44-61

Spotify 62-76

YouTube 77-97

Zoom 98-110

השוואת הגרפים בין האפליקציות השונות 111-124

חלק 1 – שאלות פתוחות

שאלה 1 :

משתמש מדווח שהעברת הקבצים שלו איטית, ועליך לנתח את שכבת התעבורה כדי לזהות את הגורמים האפשריים. אילו גומרים עשויים לגרום להעברה איטית, וכיצד תוכל לפתור את הבעיה?

פתרון:

כאשר משתמש מדווח כי העברת הקובץ איטית, עלינו לנתח את שכבת התעבורה , ולבדוק אילו גורמים עשויים לגרום לבעיה.

יכולות להיות לכך כמה סיבות:

עומס ברשת - אם הרשת עמוסה ביותר מידי חבילות נתונים, ייתכן עיכוב משמעותי בהעברת הקובץ.

הדרך לאבחן את הבעיה תהיה דרך מנגנון הQos (Quality of service), אשר זהו מנגנון לניהול ובקרת רשת, אשר מאפשר תיעדוף של סוגי התעבורה השונים ברשת← כדי למנוע עומסים.

איבוד חבילות- חבילות עשויות לא להגיע ליעדן, מה שגורם להאטה בגלל שליחה חוזרת (Retransmissions).

הדרך לפתור זאת היא לבדוק סטטיסטיקות TCP , לזהות חבילות שנשלחו מחדש, ולוודא שהנתב אינו עמוס יתר על המידה.

השהייה גבוהה- אם זמן ההגעה(Round trip time) גבוה, זה ישפיע לרעה על מהירות ההעברה- האישור מגיע לאט יותר ,ומגביל את קצב העברת הנתונים. הדרך לפתור זאת היא דרך שימוש בפרוטוקולים מהירים, כמו HTTP3 /QUIC , וזה גורם להפחתת השפעת ה-RTT באמצעות שימוש בחיבורים מתמשכים.

חוסר התאמה ברוחב פס- אם המחשב השולח יכול לשלוח מהר יותר ממה שהמחשב המקבל יכול לעבד, ייווצר "צוואר בקבוק" , אשר זהו מצב בו רכיב מסוים במערכת מגביל את הביצועים של כל המערכת. נזכיר כי רוחב פס הינו כמות הנתונים שיכולה לעבור דרך חיבור רשת בפרק זמן מסוים.

הפתרון לכך הינו שימוש בפרוטוקול MPTCP אשר נסביר עליו בשאלה 4, והוא יכול לעזור בחלוקת העומס בין מספר חיבורים.

שאלה 2:

נתח את ההשפעות של מנגנון בקרת הזרימה (Flow Control) של ה-TCP על העברת הנתונים .

כיצד הוא ישפיע על הביצועים כאשר לשולח יש כוח עיבוד גבוה משמעותית בהשוואה למקבל ?

פתרון:

ראשית, מנגנון בקרת הזרימה זוהי טכניקה ברשתות תקשורת, אשר מטרתה למנוע עומס יתר על המקבל, כאשר השולח שולח נתונים מהר מדי.

למנגנון בקרת הזרימה יש לו מספר השפעות על העברת הנתונים:

מניעת הצפה של המקבל- מנגנון בקרת הזרימה מונע מצב שבו השולח שולח יותר נתונים ממה שהמקבל יכול להתמודד איתו.
בכך, הוא מגן על המערכת של המקבל מקריסה, או איבוד נתונים.

התאמת קצב ההעברה -בקרת הזרימה מאלצת את השולח להתאים את קצב שליחת הנתונים ליכולת העיבוד של המקבל.

במצב בו המקבל איטי יותר← השולח ייאלץ להאט את קצב ההעברה.

שמירה על יציבות התקשורת- זאת על ידי ויסות תעבורת הנתונים, המנגנון מונע מצבים של עומס יתר, ומבטיח שהנתונים יגיעו בצורה מסודרת, וללא איבודים.

ייעול שימוש בזיכרון- המנגנון מאפשר שימוש יעיל בזיכרון של המקבל, ומונע ממנו להגיע למצב שבו הוא מקבל יותר מידע ממה שהוא יכול לאחסן ולעבד.

אז איך זה ישפיע על הביצועים כאשר יש לשולח כוח עיבוד גבוהה משמעותית מהמקבל?

המהירות של השולח תוגבל-גם אם השולח יכול לשלוח מהר, הוא יהיה חייב להאט ולהמתין עד שהמקבל יוכל לעבד את הנתונים.

השולח יחכה בהרבה- אם המקבל לא מצליח לעבד מהר, השולח יעצור כל הזמן כדי לחכות לאישור מהמקבל, כאשר הוא מוכן לקבל עוד נתונים.

פגיעה בביצועים-העברת הנתונים תהיה איטית יותר ממה שהשולח מסוגל ל ספק בפועל.

עומס מיותר ברשת- אם המערכת לא מוגדרת כמו שצריך, הנתונים עלולים להיאבד, מה שיוביל לשליחה מחדש, ויגרום לעומס מיותר על הרשת.

שאלה 3:

נתח את תפקיד הניתוב(Routing) ברשת, שבה קיימים מסלולים מרובים בין המקור ליעד.

כיצד בחירת הנתיב משפיעה על ביצועי הרשת, ואילו גורמים יש לקחת בחשבון בהחלטות ניתוב?

פתרון:

נזכיר כי ניתוב הינו תהליך שבו חבילות נתונים עוברות דרך הרשת מנקודת המוצא ליעד.

במערכות תקשורת מודרניות, בפרט ברשתות IP , רשתות אלחוטיות, ורשתות מבוססות ,קיימים מסלולים מרובים בין נקודת המוצא(source) לבין היעד(destination) .

מטרת הניתוב:

1. איזון עומסים, מניעת עומס ים על מסלול מסוים ע"י שימוש במסלולים חלופיים.
2. הפחתת זמן השהיה, בחירת מסלול קצר או מהיר יותר.
3. הימנעות מפקקים, הפחתת הצורך בשליחת מנות דרך קישורים עמוסים.
4. הגברת שרידות הרשת- שימוש במסלול חלופי במקרה של כשל בקישור .

איך בחירת המסלול משפיעה? :

1. בחירת מסלול ישיר וקצר מקטינה את זמן ההשהיה .
2. אם הנתיב כולל מספר קפיצות(hops) בין נתבים, ההשהיה עשויה לגדול.
3. השפעה על קצב העברת ה נתונים, כלומר מסלולים עם רוחב פס מאפשרים העברת נתונים מהירה יותר.
4. השפעה על אמינות הרשת ← בחירת מסלול דינאמי מאפשרת להימנע מתלות מסוימת, כלומר אם מסלול אחד כושל ,הנתונים מנותבים דרך מסלול חלופי.
5. ניתוב נכון מפחית עומסים ברשת, ומונע מצב שבו חלק מהקישורים עמוסים, בעוד שאחרים אינם מנוצלים.
6. השפעה על אבטחת הרשת, כלומר בחירת מסלול דרך קישורים לא מאובטחים← עלולה לחשוף את הנתונים לתקיפות.

איזה גורמים יש לקחת בחשבון בהחלטת הניתוב:

בחירת המסלול ברשת תלויה במגוון פרמטרים, המשפיעים על ביצועים, אמינות, אבטחה ,וניצול משאבים.

ההחלטות מתקבלות יל ידי פרוטוקולי ניתוב דינאמיים, אך גם על ידי מדיניות הרשת של הארגון ,או ספק השירות.

1. גורמים טכניים- זמן השהייה)הזמן שלוקח לחבילה לעבור ממקור ליעד(, רוחב הפס ,כמות הקפיצות בין הנתבים.

2. גורמים הקשורים לאמינות ושרידות- זמינות המסלול, כלומר מסלול חייב להיות פעיל וללא תקלות, כדי שניתן יהיה להשתמש בו.
   בנוסף, מסלולים חלופיים חשובים, כדי להבטיח שהרשת תוכל להמשיך לפעול במקרה של תקלה בקישור, או נפילת נתב.
   כמו כן, נשקול גם באיכות הנתיב, כלומר נתיבים מסוימים עשויים להיות פחות יציבים, ולגרום לאובדן מנות.

3. גורמים הקשורים לאבטחת מידע-רמת אבטחה של המסלול, שימוש ב-VPN או ב- MPLS.

שאלה 4:

כיצד פרוטוקול MPTCP משפר את ביצועי הרשת?

פתרון:

פרוטוקול MPTCP הינו שדרוג לפרוטוקול TCP, המאפשר לשימוש במספר נתיבים בו-זמנית, וזאת כדי לשפר אמינות וביצועי רשת.

כלומר← במקום שהמידע יעבור דרך מסלול אחד, הוא מתפצל לכמה מסלולים, ואז זה מאפשר לקבל ביצועים טובים יותר.

היתרונות:

אם יש כמה חיבורים כמו למשל wi-fi , רשת סלולרית, אזי התעבורה תעבור בשניהם ,וזה משפר את המהירות.

במקרה שחיבור אחד נופל, מכל סיבה שהיא, אז החיבור השני עדיין עובד← מה שמאפשר תקשורת מתמשכת.

במקרה של עומס על מסלול אחד, אזי הפרוטוקול יודע להפנות יותר מידע למסלול הפנוי יותר, מה משצייר איזון בין התעבורה.

לא מתקיים מצב בו מסלול אחד מלא, והשני ריק.

שאלה 5:

אתה מבצע ניטור של תעבורת רשת, ושם לב לאובדן חבילות (Packet Loss) גבוה בין 2 נתבים.

נתח את הגורמים האפשריים לאובדן החבילות בשכבת הרשת(Network layer) , ובשכבת התעבורה (Transport layer), והצע צעדים לפתרון הבעיה.

פתרון:

הגורמים האפשריים בשכבת הרשת:

1. אם הנתבים עמוסים מידי, וזורקים חבילות כדי לא לקרוס.
2. בעיות ניתוב- אם הנתיב לא מוגדר כמו שצריך← החבילות ילכו לאיבוד.
3. במידה והרשת סובלת מהפרעות מסוימות, זה יכול לגרום לאובדן מידע.
4. תקלה בחומרה← יכול להיות שהכבל תקול, או שהנתב תקול.

הגורמים האפשריים בשכבת התעבורה:

1. TCP ינסה לווסת את השליחה, ואם יש לנו עומס← חלק מהחבילות לא יגיעו.
2. .NAT אינו מתפקד כמו שצריך← זה יכול לגרום לחלק מהחבילות לא להיות מנותבות כמו שמצופה.
3. .UDP לא מבצע שליחה חוזרת← אם החבילה נאבדה, אז היא נאבדה.

דרכי פתרון:

בשכבת הרשת: ראשית, אפשר לבדוק איפה חבילות נעלמו דרך ה-Wireshark , אפשרי לשדרג נתבים אם הם לא עומדים בעומס, ולבדוק כבלים וחיבורים בעייתיים.

ברמת החומרה: לבדוק אם יש עומס, אז לנסות לאזן תעבורה במסלולים שונים.

בשכבת התעבורה: אפשר לבצע שימוש ב-TCP, ולא ב-UDP. אפשרי גם להשתמש ב- MPTCP, וזאת כדי לאפשר לניתוב תעבורה, ובנוסף← לבדוק אם הנתב שלנו חוסם חבילות.

קריאת המאמרים וניתוחם:

לכל מאמר, כתוב:

מהי התרומה המרכזית של המאמר?

אילו מאפייני תעבורה (Traffic Features) המאמר משתמש בהם, ואילו מהם הם חדשניים?

מהם הממצאים העיקריים של המחקר? (אפשר להעתיק תמונות ותרשימים מהמאמרים)

מה התובנות שניתן להסיק מהתוצאות?

מאמר 1:

FlowPic Encrypted Internet Traffic Classification is as Easy as Image Recognition

בעבר, ניתוח וסיווג תעבורת רשת היה מבוסס על מספרי פורטים, זיהוי תוכן החבילות ,וניתוח סטטיסטי של זרימות.

הבעיה הינה ההצפנה(כגון VPN,TLS) , אשר מסתירה את תוכן החבילות, והופכת את השיטות הישנות לבלתי יעילות.

הצורך הינו בשיטה, אשר לא דורשת גישה לתוכן החבילה, אלא מבוססת אך ורק על המאפיינים של הזרימה.

מהי התרומה המרכזית של המאמר?

לכן, במאמר זה מוצג שיטה חדשה לסיווג תעבורת אינטרנט מוצפנת בשם FlowPic , היא עושה זאת באמצעות המרת זרמי התעבורה לתמונות, ושימוש ברשתות נוירונים קונבולציונית (CNN) , זאת לשימוש זיהוי דפוסים. ה-CNN גורם לסיווג סוגי התעבורה והאפליקציות, ובנוסף משמש לסיווג התמונה בדיוק גבוה.

השיטה הזו עובדת גם כאשר התעבורה מוצפנת (VPN), היא גורמת לביצועים מצוינים בדיוק של 99.7%, גורמת לתפעול מצוין בזמן אמת, ויכולת הכללה של אפליקציות חדשות.

אמנם FlowPic נועדה לסיווג תעבורה, אך ניתן לציין שגם חוקרי אבטחת מידע יכולים להשתמש בה← לזיהוי אנומליות(חריגות או סטיות מהתנהגות צפויה) התעבורה מוצפנת .

במקום להשתמש בגישות מסורתיות לסיווג תעבורה, כמו ניתוח חבילות או מאפיינים סטטיסטיים, המאמר מראה שניתן לנתח תעבורה מוצפנת באותה הדרך שבה מזהים תמונות.

אילו מאפייני תעבורה (Traffic Features) המאמר משתמש בהם, ואילו מהם הם חדשניים?

במאמר זה , נעשה שימוש במספר מאפייני תעבורה (Traffic Features) כדי לסווג את זרמי הנתונים המוצפנים.

יש לנו במאמר מאפייני תעבורה, אשר מוכרים לנו:

גודל החבילות (Packet size distribution) -המאמר מתחשב בהתפלגות גדלי

החבילות (Bytes) בתעבורה הנכנסת והיוצאת, גודל החבילות משמש ליצירת ציר ה-X של ה- FlowPic.

)- מדידת הפרשי הזמן בין חבילות, זה מאפשר **Inter-Packet arrival time זמני הגעה בין החבילות(**
. FlowPic של ה-Yלנו לזהות דפוסים ייחודיים לשירותים שונים. זה משמש לנו ליצירת ציר ה-

סיווג פרוטוקולים (Transport layer protocols)- יש לנו שימוש בפרוטוקולים כמו QUIC
.HTTP2,TCP,UDP,

כיוון התעבורה(Direction of traffic)- האם מדובר בחבילות נכנסות או יוצאות.

נפח הנתונים בזרם(Flow value)-מדידת כמות הנתונים המועברת בפרק זמן מסוים.

מופיעים לנו במאמר מאפייני תעבורה חדשים שהמאמר מציג:

המרת נתוני התעבורה לתמונות(FlowPic representation)-השיטה החדשנית במאמר משתמשת בגישה שבה נתוני הרשת מומרים לדימויים חזותיים(תמונות), במקום לטבלאות מספריות.

שימוש ב-CNN לזיהוי דפוסים- המאמר מציע דרך לזהות מאפייני תעבורה באמצעות רשתות נוירונים קונבולציוניות, שמסוגלות לזהות דפוסים מורכבים בתמונת התעבורה.

יכולת סיווג ללא גישה לתוכן-בניגוד לשיטות קודמות שהסתמכו על בדיקת תוכן החבילות ,השיטה החדשה עובדת גם על תעבורה מוצפנת, מה שמעלה את רמת הדיוק והיעילות.

מהם הממצאים העיקריים של המחקר?

דיוק גבוה בזיהוי סוגי התעבורה-FlowPic הוכיח את יכולתו לרמות דיוק גבוהות, במיוחד בסיווג של סוגי תעבורה שונים:

, 67.8% בזיהוי VPN Traffic בזיהוי 98.4% , Non-VPN Traffic בזיהוי 97% בזיהוי אפליקציות, 99.7%
(דיוק נמוך יחסית עקב אנונימיות מוגברת).Tor Traffic

FlowPic מציג שיפור משמעותי לעומת שיטות קודמות, בעיקר בסיווג תעבורה מוצפנת. למשל, בתמונה למטה אפשר לראות כי כיצד ה-FlowPic משיג דיוק גבוה יותר בהשוואה לשיטות אחרות, במיוחד בסיווג תעבורה מוצפנת:

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

שימוש ב-FlowPic להמרת תעבורה לתמונות וזיהוי אנומליות- FlowPic ממיר זרמים של תעבורה לרצפי תמונות כדי לאפשר לרשתות CNN לבצע סיווג.

השיטה מסוגלת להבחין בין סוגי תעבורה שונים כמו VoIP, Browsing, Streaming, צ'אט וכולי.., גם כאשר נעשה שימוש ב-VPN או Tor.

המאמר מציג מטריצת בלבול(Confusion Matrix) , שמראה כיצד FlowPic מסווג סוגי תעבורה עם שגיאות מינימאליות.

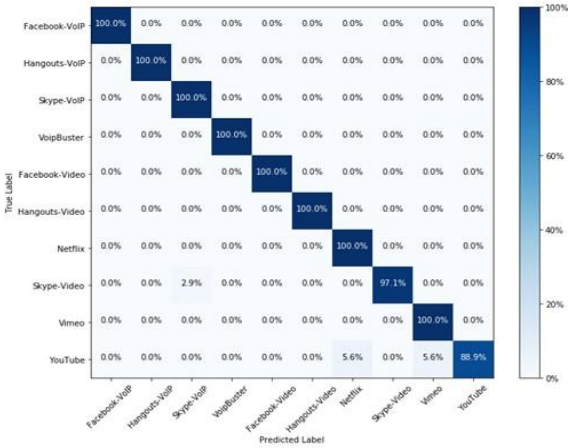
בתמונה למטה, אפשר לראות **מטריצת בלבול(Confusion Matrix)** , אשר מציגה את אחוזי הזיהוי עבור כל קטגוריית תעבורה, כולל ,Facebook, Skype, YouTube וכולי..

FlowPic מצליח להבדיל היטב בין האפליקציות עם מינימום שגיאות.

השורות מייצגות את התוויות האמיתיות של הדגימות בפועל, העמודות הן התחזיות של המודל, כלומר מה שהמודל חזה.

האלכסון הראשי מייצג תחזיות נכונות, כלומר דגימות שסווגו כראוי.

כל תא אשר מחוץ לאלכסון הראשי מייצג שגיאות סיווג, כלומר המודל ניבא לא נכון, וסיווג את הדגימה תחת קטגוריה אחרת.



אדריכלות הרשת הנוירונית(CNN Architecture for FlowPic)-FlowPic משתמש ברשת CNN מהסוג LeNet-5 לניתוח התמונות, שנוצרו מהתעבורה.

זה מאפשר לזהות תבניות חבילות שונות, כולל דפוסי הצפנה, חבילות UDP/TCP , ואופן התנהגות של זרמים מוצפנים.

למשל, בתמונה למטה אפשר לראות את המבנה של CNN , שבו משתמשים לזיהוי דפוסי התעבורה בתמונות שנוצרו.

ובכך, אפשר להבין כי רשת ה-CNN מאפשרת לסווג זרמים לפי המבנה הוויזואלי שלהם ,ללא צורך בגישה לתוכן החבילה.

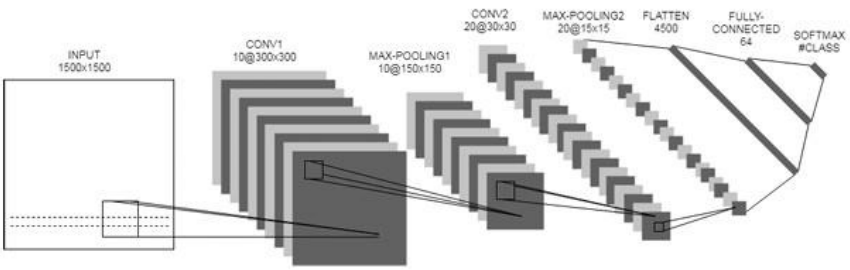


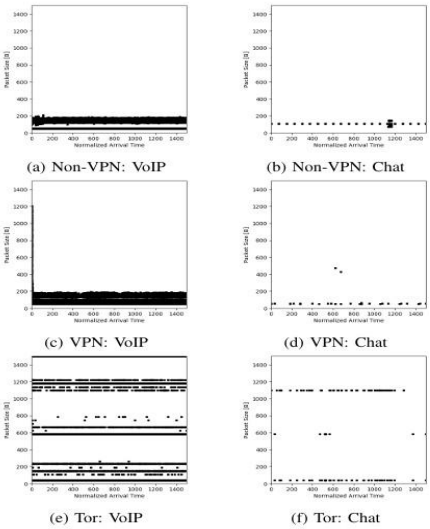
Figure 3: An illustration of the architecture of our LeNet-5 style CNN.

ניתוח זרמי תעבורה עם דוגמאות-FlowPic מציג תמונות שונות של File Transfer, Chat **VoIP, ו-Streaming , כאשר Non-VPN ו-Tor מטשטשים חלק מהמאפיינים.**

ההבדלים בתמונות מראים רמות שונות של אחידות בדפוסי התעבורה, כאשר Tor ו- VPN מטשטשים חלק מהמאפיינים, אך FlowPic עדיין מצליח לזהות אותם.

למשל בתמונה למטה אפשר לראות איך נראית תעבורה מסוגים שונים, כולל VoIP וצ'אט, תחת VPN ו Tor .

המשמעות לכך הינה ש-FlowPic מצליח להציג דפוסים ברורים← גם כאשר ההצפנה מסתירה את תוכן החבילות.



תוצאות השוואה בין Non-VPN, VPN ו-Tor- המאמר מציג טבלת ביצועים בין דיוק הזיהוי של סוגי תעבורה שונים, כולל Non-VPN,VPN ו-Tor.

FlowPic מצליח לשמור על רמות דיוק גבוהות גם בתעבורה מוצפנת.

Tor נותר האתגר הגדול ביותר, אך FlowPic מצליח להבחין בין סוגי זרמים גם שם.

בתמונה למטה אפשר לראות את טבלת ההשוואה בין Non-VPN,VPN ו-Tor :

	Non-VPN	VPN	Tor
VoIP	3304	2872	2978
Video	1553	302	754
File Transfer	1174	242	1126
Chat	635	1061	422
Browsing	3191	-	2026

התמונה מציגה כיצד FlowPic מזהה זרמי תעבורה עם רמות דיוק שונות, בהתאם לרמת האנונימיות של החיבור.

ככל שהתעבורה מוצפנת יותר(למשל ב-Tor), כך הדיוק מעט יורד, אך FlowPic עדיין מזהה דפוסים ייחודיים.

מאמר 2 :

Early Traffic Classification with Encrypted ClientHello a Multi- Country Study

מושגים המופיעים במאמר:

Traffic Classification (סיווג תעבורה)

סיווג תעבורה הוא קריטי לניהול רשתות מודרניות, במיוחד בהקשרים של איכות שירות, אבטחת מידע, והנדסת תעבורה. משמש כדי לזהות במהירות את סוג השירותים, שנעשה בהם שימוש בתעבורה באינטרנט. המטרה היא לשפר את איכות השירות ברשתות ,זאת על ידי התאמה אוטומטית של משאבים, כמו למשל רוחב פס, סדר עדיפויות, ושימוש נכון בתשתיות.

TLS-פרוטוקול לאבטחת מידע ברשת

הפרוטוקול פועל כך:

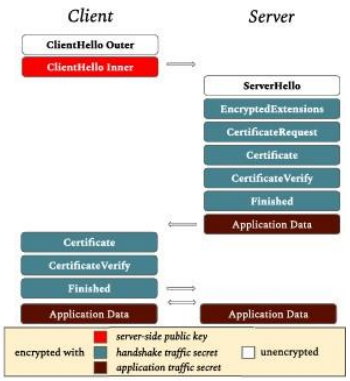
1. Handshake ראשוני: הלקוח שולח (ClientHello), והשרת מחזיר (ServerHello). ב-ClientHello יש מידע כמו SNI (Server Name Indication),שמזהה את הדומיין שאליו הלקוח מנסה להתחבר .
   TLS קובע איזה סוג הצפנה ישמש את החיבור.
2. הצפנת הנתונים:
   לאחר השלמת ה -Handshake ← כל התקשורת בין הלקוח לשרת מוצפנת, כך שרק הם יכולים לקרוא אותה .

לפי עמודים 3-5:

המאמר עוסק בסיווג מוקדם של תעבורת רשת בסביבה, אשר בה הלקוח משתמש ב- Encrypted ClientHello (ECH) בפרוטוקול TLS , זאת מכיוון ש-ECH מסתיר מידע קריטי כמו SNI , ובכך סיווג התעבורה הופך להיות למאתגר יותר.

ECH זהו מנגנון הצפנה מתקדם בפרוטוקול TLSv1.3 , שנועד להסתיר את שם השרת (SNI), וזאת כדי לשפר את פרטיות המשתמשים ברשת.

הרחבת ECH נועדה להסתיר את שם השרת(SNI), מה שמונע סיווג תעבורה מסורתי, המבוסס על SNI.



התרומות המרכזיות של המאמר:

1. יצירת מאגר נתונים רחב ומגוון- כולל יותר מ-600,000 זרמי TLS מחולקים ל-19 קטגוריות שונות, אשר נאספו ממדינות בצפון אפריקה, אירופה , ואסיה.
2. פיתוח אלגוריתם חדש בשם hRFTC.

זהו אלגוריתם חדשני לסיווג מוקדם של התעבורה, אשר משתמש בתכונות לא מוצפנות של TLS , יחד עם תכונות המבוססות זרימה.

מדובר במודל Random Forest היברידי, אשר משלב מידע מוצפן חלקית מה-ClientHello , יחד עם מאפיינים סטטיסטיים של התעבורה← לפני קבלת נתונים מהשרת .

3. השוואת ביצועים עם אלגוריתמים קיימים-הראו שהאלגוריתם החדש hRFTC משפר משמעותית את דיוק הסיווג, לעומת האלגוריתמים המובילים הקיימים.
4. מחקר על שונות גיאוגרפית, והשפעתה על הסיווג- נמצא כי אלגוריתמים, המודרים במדינה מסוימת← אינם בהכרח מתאימים למדינות אחרות, מה שמדגיש צורך באימון מחדש, בהתאם למיקום הגיאוגרפי.

לפי עמודים 9-10:

זאת תחת הכותרת "Feature Extraction" , המאמר משלב שני סוגי מאפיינים:

1. Packet-based Features (מאפיינים מבוססי חבילות)

פרטים מהודעות ClientHello ו-ServerHello , אשר אינם מוצפנים.

גודל ותוכן הודעות TLS בתוך החבילות.

ותבניות בהודעות ההתחלה של TLS.

2. מאפיינים מבוססי זרימה(Flow-based Features)

סטטיסטיקות של גדלי החבילות(Packet Size-PS) בכל זרימה.

סטטיסטיקות של זמני ההגעה בין החבילות(Inter-Packet Time-IPT) .

ופילוח בין תעבורת העלאה (uplink)להורדה (downlink).

Uplink הינו תעבורת נתונים מהמשתמש לרשת (העלאה), ואילו Downlink זוהי תעבורת נתונים מהרשת למשתמש (להורדה) .

מאפיינים חדשים שהמאמר מציע:

- שימוש משולב במאפייני חבילות ומאפייני זרימה← שילוב זה מוכיח שהוא מניב דיוק סיווג גבוה יותר.
- הארכה של RF-RB (מודל קודם), כך שיתמוך גם בתעבורה המבוססת על QUIC , תוך התאמות ייחודיות לפרוטוקול QUIC.
- הצעה לקריטריון חדש לבחירת החבילות הרלוונטיות לצורך סיווג- במקום לקחת מספר קבוע של חבילות(כמו במחקרים קודמים), הם בוחרים את כל החבילות עד החבילה הראשונה המכילה נתוני אפליקציה.

לפי עמודים 11-12 :

זאת תחת הכותרת "Hybrid RFTC" , המאמר מציע אלגוריתם חדש hRFTC .

המאמר מציג את hRFTC כאלגוריתם חדש לסיווג מוקדם של תעבורה מוצפנת, אשר תוכנן במיוחד כדי להתמודד עם הצפנת ECH בפרוטוקול TLS.

הבעיה היא שהשיטות הקודמות של סיווג תעבורה הסתמכו במידה רבה על מידע ,אשר הופיע ב-ClientHello , כמו SNI, פרטי הצפנה.

ECH מצפין את כל המידע הזה, מה שמונע מסווגים קודמים לזהות נכון את סוג התעבורה.

למשל- שימוש רק בתכונות TLS מביא לדיוק נמוך של כ-38.4%.

hRFTC משלב שני סוגי מאפיינים:

1. מאפיינים מתוך ה-TLS שלא הוצפנו(Packet-based) .
2. מאפיינים סטטיסטיים של זרימה(Flow-based).

האלגוריתם משתמש ב -Random Forest – כדי לשלב את המאפיינים, ולבצע סיווג מדויק של סוג התעבורה, גם כאשר המידע מוצפן.

Random Forest הוא אלגוריתם למידת מכונה, המבוסס דרך עצי החלטה← בנוי מקבוצת עצים עצמאיים , כאשר כל אחד מהם מסווג זרימת תעבורה בצורה נפרדת.

השיטה הזו משפרת דיוק, ומונעת למידת יתר.

ממצאים עיקריים:

1. דיוק הסיווג עם Packet-based בלבד (ללא נתוני זרימה)יורד משמעותית, זאת כאשר משתמשים ב-ECH.

   אלגוריתמים המבוססים רק על נתוני ה-TLS) ללא המאפיינים הסטטיסטיים של הזרימה(מגיעים לדיוק של 38.4% בלבד.

2. השיפור המשמעותי עם hRFTC , האלגוריתם החדש שהמאמר מציע מגיע לדיוק של 94.6%, זאת לעומת 60-80% באלגוריתמים קיימים.
3. שונות גבוהה בין אזורים גיאוגרפיים- נמצא שהאלגוריתמים מאומנים היטב במדינה אחת, אף אינה מתפקדים בצורה טובה במדינה אחרת, ככל הנראה בשל ההבדלים בתשתיות הרשת .
4. למרות ההצפנה של ECH , עדיין ניתן לבצע סיווג אפקטיבי- אך נדרש שימוש במידע סטטיסטי נוסף, מעבר לנתוני ה-TLS בלבד .

Country	Cities	Number of Flows
Germany	Munich	121,936
Kazakhstan	Aktau, Aktobe	17,228
Russia	Moscow, Dolgoprudny, Zelenograd	234,335
Spain	Girona	95,154
Turkey	Antalya, Kemer	160,131
USA	Miami	43,850

תובנות מהמחקר:

- שימוש ב-ECH מוריד משמעותית את היכולת של שיטות סיווג מסורתיות, אשר התבססו על מידע מתוך ה-ClientHello , ולכן חייבים להשתמש בגישות היברידיות.
- מאפייני זרימה חשובים לא פחות ממאפייני TLS , בפרט נתונים כמו גודל חבילות ,זמן ביניהן , והבדלים בין uplink ל-downlink , אשר מסייעים לסיווג.
- שימוש ב-QUIC , זה מוסיף מורכבות , אך לא מונע סיווג מדויק← כל עוד נלקחים בחשבון מאפיינים סטטיסטיים.
- יש צורך לבצע אימון מחדש של המודל, בהתאם לאזור גיאוגרפי-בגלל ההבדלים בתשתיות הרשת, והגדרות שרתים.

מאמר 3 :

Analyzing HTTPS Encrypted Traffic to Identify User's Operating System, Browser, and Application

מבוא:

כיום, רוב תעבורת האינטרנט מוצפנת באמצעות פרוטוקול HTTPS , זאת כדי להגן על פרטיות המשתמשים, ולמנוע מגורמים חיצוניים לגשת למידע המועבר.

עם זאת, המחקר מראה שאפשר לזהות את מערכת ההפעלה, הדפדפן, והאפליקציה שהמשתמשים משתמשים בהם, וזאת רק בעזרת ניתוח מאפייני התעבורה המוצפנת(HTTPS) , ללא צורך בפיענוח המידע עצמו.

למרות שהמידע מוצפן, במחקר החוקרים גילו כי קיימים דפוסים קבועים בתעבורה← כמו גודל החבילות, זמני השליחה, מספר הבקשות, ותגובות שיכולים לגלות איזה מכשיר, דפדפן , ואפליקציה המשתמש מפעיל.

מה היא התרומה המרכזית של המחקר :

התרומה המרכזית של המאמר היא היכולת לזהות את מערכת ההפעלה, הדפדפן , והאפליקציה של המשתמש ← זאת על סמך תעבורת HTTP מוצפנת, ללא צורך בפיענוח הנתונים עצמם.

מדוע זה מועיל :

- המחקר מוכיח שגם בלי לפענח את התוכן, אפשר לזהות את מערכת ההפעלה ,הדפדפן , והאפליקציה← זאת לפי דפוסי התעבורה(למשל גודל החבילות, וזמני שליחה וקבלה) .
- המחקר מועיל בכך שעולות חששות לפרטיות, זאת מכיוון שאפילו שמשתמשים בהצפנה חזקה, עדיין יהיה אפשר לזהות מידע על המשתמש .
- אפשרויות חדשות לאבטחת מידע, ניתן להשתמש בשיטה לזיהוי תוכנות זדוניות ,ולשיפור הגנה על רשתות.

<u>אילו מאפייני תעבורה המאמר משתמש בהם, ואילו מתוכם הם חידושיים ?</u>

אלו מאפיינים שהיו בשימוש במחקרים קודמים לניתוח תעבורת רשת:

1. מאפייני גודל חבילות(Packet Size Features) :

- גודל מינימלי של חבילה(Minimum Packet Size)
- גודל מקסימלי של חבילה(Maximum Packet Size)
- גודל ממוצע של חבילות(Mean Packet Size)
- שונות בגודל החבילות(Packet Size Variance)
- סך כל הבייטים שנשלחו בכל כיוון(Forward/Backward Total Bytes)

2. מאפייני זמן(Timing Features) :

- זמן הגעה בין חבילות – (Inter-arrival Time) מינימום, מקסימום, ממוצע ושונות
- זמן תגובה בין שליחת ובקבלת חבילות

3. מאפייני זרימה(Flow-Based Features) :

- מספר חבילות שהתקבלו ונשלחו(Forward/Backward Packets)

- קצב שליחת חבילות(Packet Rate)
- כיוון התעבורה(Packet Direction)
- מספר הבקשות והתגובות בתוך החיבור(Request/Response Count)

ואלו המאפיינים החידושיים שבהם השתמשו במאמר:

1. מאפייני SSL/TLS חדשים(Encryption-Based Features) :

- מספר הרחבות SSL בשימוש(SSL Extension Count)
- מספר שיטות הצפנה(SSL Cipher Methods)
- אורך מזהה ה Session-של ה(SSL Session ID Length) SSL-
- גרסת(Forward SSL Version) SSL Forward

2. מאפייני TCP מתקדמים(TCP Features)

- גודל חלון TCP ראשוני(TCP Initial Window Size)
- גורם קנה מידה של חלון(TCP Window Scaling Factor) TCP
- מספר חבילותKeep-Alive
- ערך (Maximum Segment Size) MSS בTCP

3. ניתוח התפרצויות נתונים(Bursty Behavior Features)

- מספר התפרצויות נתונים (Bursts) בזרימה קדימה ואחורה
- קצב השיא של throughput (נתוני רוחב פס) בזרימה קדימה ואחורה
- ממוצע זמן בין שיאי throughput קדימה ואחורה
- מקסימום ומינימום זמן בין שיאי throughput

מהם הממצאים העיקריים? ומה התובנות שניתן להפיק מהם?

1. דיוק גבוה בסיווג מערכת ההפעלה, הדפדפן והאפליקציה.

המודל שסיווג את מערכת ההפעלה, הדפדפן והאפליקציה הצליח להגיע לדיוק של 96.06%כאשר נעשה שימוש במאפיינים החדשים שהוצגו במאמר.

- ללא המאפיינים החדשים ,השיטה הצליחה להגיע לדיוק של .93.52%
- השיטה ה"נאיבית "(ללא למידת מכונה, רק לפי התפלגות הנתונים) הצליחה להגיע לדיוק של 32.34%בלבד.
- השיפור הגדול בדיוק הגיע מהמאפיינים החדשים של SSL/TLS ושל דפוסי ההתפרצויות בתעבורה.(Bursty Behavior)

2. השפעת המאפיינים החדשים על שיפור הדיוק.

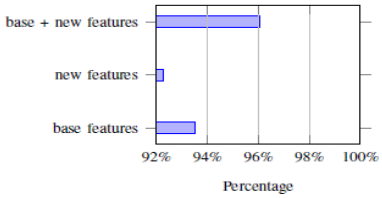
- מאפייני SSL/TLS סייעו לשיפור דיוק הסיווג של מערכת ההפעלה והדפדפן.
- מאפייני Bursty Behavior שיפרו את הדיוק בזיהוי אפליקציות כמו YouTube, Skypeו- TeamViewer.

- מאפייני TCP מתקדמים שיפרו את היכולת לזהות פרוטוקולים ומכשירים על פי צורת ניהול התקשורת שלהם.

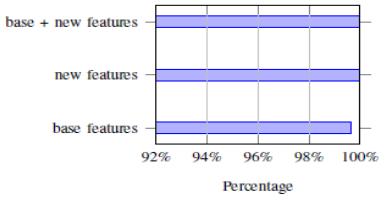
3. דיוק הסיווג עבור כל קטגוריה.

- מערכת ההפעלה –סווגה כמעט ללא טעויות (דיוק של 99%+).
- הדפדפן –היה מעט יותר קשה לזיהוי, אך עדיין 97%דיוק.
- האפליקציה –הייתה הקטגוריה הכי קשה לזיהוי, עם מספר שגיאות גבוה יותר (בעיקר עבור Facebook, שזוהתה לא נכון בכ-29% מהמקרים).

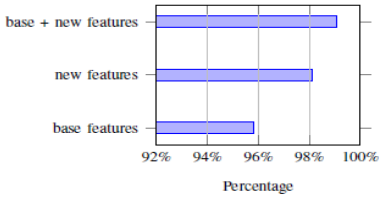
בתמונה הזאת נראה את רמות הדיוק לפי מאפיינים שונים.



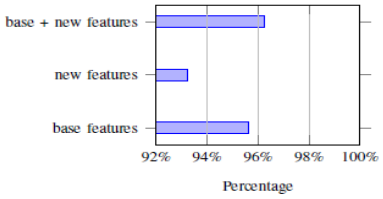
(a) Tuple Accuracy Results



(b) OS Accuracy Results

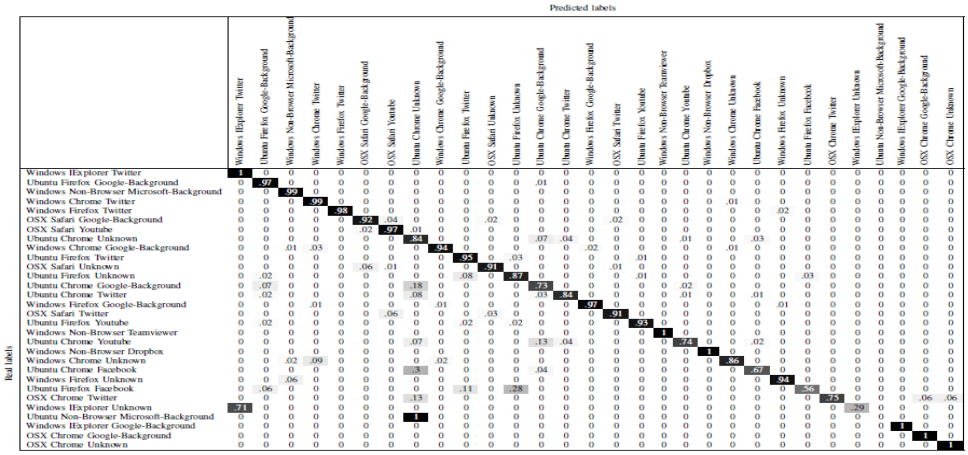


(c) Browser Accuracy Results



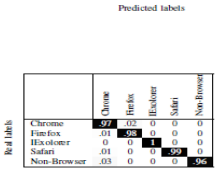
(d) Application Accuracy Results

ובתמונה בזאת נראה את מטריצות הבלבול שמראות כי לרוב הקטגוריות יש סיווג כמעט מושלם.

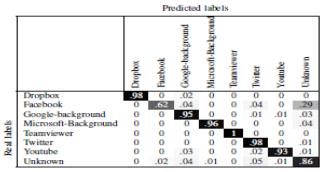


(a) Tuple Confusion Matrix

(b) OS Confusion Matrix



(c) Browser Confusion Matrix



(d) Application Confusion Matrix

התובנות שאפשר להסיק מהמאמר:

- גם כאשר התעבורה מוצפנת, ניתן להסיק מידע על המשתמש .
  ההנחה שההצפנה מגנה באופן מלא על הפרטיות אינה נכונה← ניתן להסיק מסקנות מתזמון, וגודל החבילות בלבד.
- SSL/TLS מגן על התוכן, אך אינו מסתיר את דפוסי התעבורה, מה שמאפשר לזהות את פעילות המשתמשים.
- משמעות מעשית← צריך לשפר הצפנת META נתונים, למשל על ידי הוספת רעש אקראי לתקשורת
- השיטה יכולה לשמש גם לזיהוי אנומליות, ואבטחה.
  ניתן להשתמש בטכניקה לזיהוי תעבורה חריגה ברשת, למשל זיהוי ניסיונות גישה של תכונות זדוניות.

אפשר לשלב שיטה זו במערכות אבטחה לזיהוי משתמשים, המנסים להסתיר את זהותם.

- משמעות מעשית נוספת← גופים ארגוניים יכולים לפתח מערכות הגנה חכמות, אשר מזהות דפוסי תעבורה לא רגילים.

הוספת רעש אקראי לזמני שליחת חבילות, זאת כדי למנוע זיהוי דפוסי תעבורה.

שינוי דינאמי של גודל החבילות כדי למנוע יצירת חותמות רשת ייחודיות.

שימוש בproxy VPN , או TOR כדי להקשות על ניתוח דפוסים.

פיתוח פרוטוקולי הצפנה מתקדמים , אשר מסתירים גם את ה- META נתונים של התעבורה.

חלק שלישי :

גלישה באתר אינטרנט דרך כרום :

חלק שלישי – הקלטה של אתר מאובטח בדפדפן Chrome

בחרנו לעשות על אתר מאובטח של ויקיפדיה .

ראשית נכנסנו לWireshark ומשם ניגשנו לאינטרפייס של Wi-Fi
התחלנו את ההקלטה ולאחר מכן נכנסו דרך דפדפן כרום ל Wikipedia.com יצאנו מהאתר ועצרנו את ההקלטה .

ביצענו את הסינון על מנת שרק הפאקטות הרלוונטיות לויקיפדיה יהיו בסטטיסטיקות שלנו .
נסנן את ההקלטה (הסינון אסביר בהמשך) ונשמור את ההקלטה כקובץ csv שעליו ננתח את כל הסטטיסטיות מWireshark.

את ההקלטה נשמור בשם Wikipedia.csv.

סיננו את ההקלטות כך :

```
(ip.addr == 185.15.59.224) and (tcp.port == 443 or quic)
```

הסבר הסינון :

ip.addr == 185.15.59.224

- מסנן רק חבילות רשת שבהן כתובת ה IP 185.15.59.224-מופיעה כמקור (Source) או כיעד (Destination) .
- כתובת זו שייכת לוויקיפדיה, ולכן החבילות שיתקבלו הן רק אלו שמתקשרות עם האתר.

(tcp.port == 443 or quic)

- מציג רק חבילות שמשתמשות באחד משני הפרוטוקולים:

tcp.port == 443 מסנן רק תעבורה על גבי פרוטוקול TCPבפורט ,443כלומר תקשורת HTTPS (TLS).

quic מסנן גם חבילות שמשתמשות בפרוטוקול ,QUIC שהוא האלטרנטיבה של HTTP/3שאינו מבוסס TCP

הסינון מוודא שהתוצאות יכילו רק תעבורה הקשורה לויקיפדיה.

לאחר שימוש בפילטר הזה ב ,Wireshark-תוצג רק חבילות תקשורת בין המחשב שלי לוויקיפדיה ,כגון:

- חבילות הקמת חיבור.
- שליחת וקבלת בקשות למשל טעינת דף.
- חבילות QUIC אם הדפדפן תומך בHTTP/3

לסיכום הסינון משמש לזיהוי תעבורה לאתר ויקיפדיה בלבד.
הפילטר מסנן חבילות שבהן מופיעה כתובת ה IP-של ויקיפדיה (185.15.59.224) ומציג רק תעבורה מוצפנת (TLS) HTTPS – בפורט 443 או HTTP/3מעל.QUIC
כך ניתן לנתח רק את החבילות הרלוונטיות לתקשורת מול האתר, ללא רעשי רקע מפרוטוקולים אחרים.

כעת ניגש לקובץ פיתון שכתבתי , על מנת להפיק גרפים שיציגו את הסטטיסטיות
נדרשנו להתעסק בנושאים הבאים :
א. שדות כותרות IP

ב. שדות כותרת TCP

ג. שדות כותרת TLS

ד גודלי מנות.

ה. זמן הגעה בין מנות (הזמן בין כל שתי מנות).

ו. גודל זרימה (כלומר, המספר הכולל של מנות).

ז. נפח זרימה (כלומר, המספר הכולל של בתים ששודרו).

תחילת הקוד הוא טעינת הקובץ שמכיל את המידע הרלוונטי מההקלטה השורות הבאות טוענות את הנתונים מקובץ הCSV ממירות את עמודת הזמן לפורמט מספרי על מנת למנוע בעיות עם חישובי זמן. ומסדרות את הנתונים לפי זמן כך שניתוחי ההפרשים יהיו נכונים. כמו כן אנחנו עושים ייבוא לספריות שונות.

```
import pandas as pd
import matplotlib.pyplot as plt

# 1. Load the CSV
df = pd.read_csv('wikipedia.csv')

# 2. Convert Time to numeric if needed
df['Time'] = pd.to_numeric(df['Time'], errors='coerce')
df = df.sort_values('Time').reset_index(drop=True)

```

- גרף ראשון : שדות כותרות IP

```python
##############################################################################
# 1) IP Address Distribution (Source vs. Destination) - Improved Readability
##############################################################################
source_counts = df['Source'].value_counts()
dest_counts = df['Destination'].value_counts()

plt.figure(figsize=(10, 5))
width = 0.4  # Width for bars
x = range(len(source_counts.index))
plt.bar(x, source_counts.values, width=width, label='Source IPs', color='steelblue')
plt.bar([i + width for i in x], dest_counts.values, width=width, label='Destination IPs', color='darkorange')
plt.xticks([i + width / 2 for i in x], source_counts.index, rotation=45)
plt.title('IP Address Distribution (Source vs. Destination)')
plt.legend()
plt.tight_layout()
plt.show()
```

הגרף מציג התפלגות של כתובות IP בתור מקור (Source) ויעד.(Destination)
זה עוזר להבין איזו כתובת IP שלחה הכי הרבה חבילות ואיזו קיבלה הכי הרבה חבילות.

צירי הגרף:

- ציר הX-האופקי:
מציג את כתובות ה IP-שהתגלו בקובץ הנתונים, כאשר:

  185.15.59.226היא השרת (ויקיפדיה).

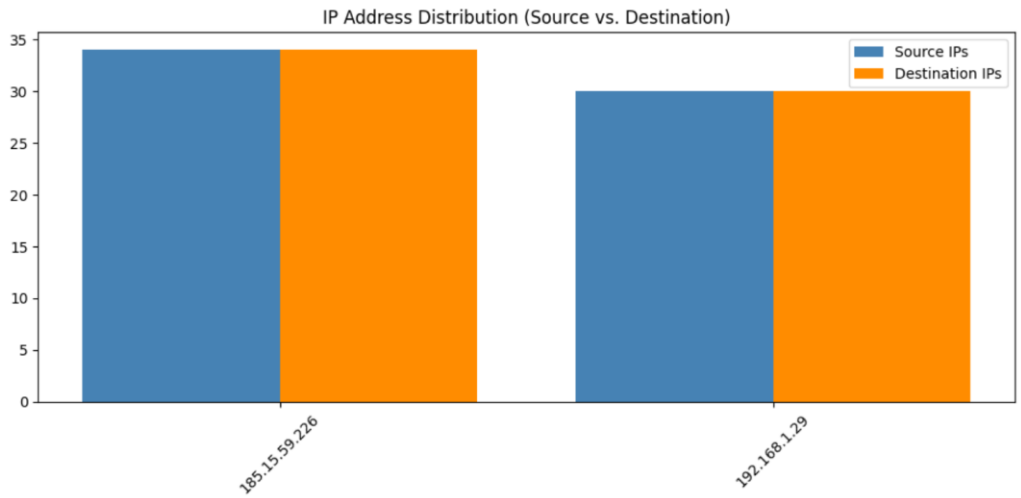
  192.168.1.29 היא הכתובת של המחשב שלי.

- ציר ה Y-האנכי:
מייצג את מספר החבילות שכל כתובת הייתה מעורבת בהן (כמה פעמים היא הופיעה בתור מקור או יעד).

- כחול :(Source IPs) מייצג את מספר הפעמים שכל כתובת הופיעה כמקור של חבילה (כלומר, מי שלח את החבילה).

- כתום :(Destination IPs) מייצג את מספר הפעמים שכל כתובת הופיעה כיעד של חבילה (כלומר, מי קיבל את החבילה).

להלן הגרף :



שתי הכתובות מופיעות בכחול ובכתום כמעט שווה ,כלומר:

- נראה שיש כמות שווה של חבילות שנשלחו מהכתובת 192.168.1.29 (המחשב שלי) ל-185.15.59.226 (ויקיפדיה) ולהיפך. זה מצביע על תקשורת דו-כיוונית יציבה ,מה שמאוד אופייני לחיבור HTTPS .
- הגובה של העמודות מציין את כמות החבילות: שתי הכתובות מופיעות באותו גובה ,מה שמראה שהתעבורה הייתה מאוזנת (בערך אותו מספר חבילות נשלחו והתקבלו).

- גרף שני : שדות כותרת TCP

```
##############################################################################
# 2) TCP Flags Analysis
##############################################################################
flag_counts = df['Flags'].value_counts()
plt.figure(figsize=(8, 5))
plt.bar(flag_counts.index, flag_counts.values, color='green')
plt.title('TCP Flags Distribution')
plt.xlabel('Flags')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

הגרף מציג התפלגות של דגלי TCP (Flags) בחבילות הרשת שנקלטו בWireshark.

פירוט הצירים:

- ציר ה X-האופקי
מציג את ערכי דגלי ה TCP-בפורמט הקסדצימלי. כל ערך מייצג סוג מסוים של חבילה בפרוטוקולTCP .
- ציר ה Y-האנכי:
מציג כמה פעמים כל דגל הופיע בתעבורה שנלכדה.

בפרוטוקול TCP (Transmission Control Protocol) כל חבילה (Packet) מכילה דגלים (Flags)שמגדירים את סוג הפעולה שהיא מבצעת.
הדגלים הם ביטים מיוחדים בכותרת ה (TCP Header)-TCP-שמסמנים איך החיבור מתקדם , האם יש בקשה לפתיחת חיבור ,אישור נתונים ,סגירת חיבור ,או שידור מיוחד של מידע.

הדגלים שמופיעים בגרף :

1. ACK (Acknowledgment) - 0x010

זהו הדגל הכי נפוץ בתקשורת תקינה.

חבילה עם דגל ACK משמשת להודיע לצד השני שהמחשב קיבל את הנתונים בהצלחה כל חבילה שנשלחת דורשת אישור קבלה ,אז יש הרבה חבילות ACK בתעבורה רגילה.

2. PSH + ACK (Push + Acknowledgment) - 0x018

זה שילוב של שני דגלים:

– PSH (Push Flag) מבקש מהמחשב המקבל להעביר את הנתונים מהר לזיכרון ולא לחכות לצבירה של חבילות נוספות.

– ACK (Acknowledgment) מאשר שקיבלנו את החבילה.

3. FIN + PSH + ACK - 0x011

כאן יש שלושה דגלים יחד:

– FIN (Finish Flag) מציין שהשולח רוצה לסגור את החיבור.

– PSH (Push Flag)מבקש להעביר את הנתונים האחרונים ללא עיכוב.

– ACK (Acknowledgment)מאשר את הקבלה של הנתונים.

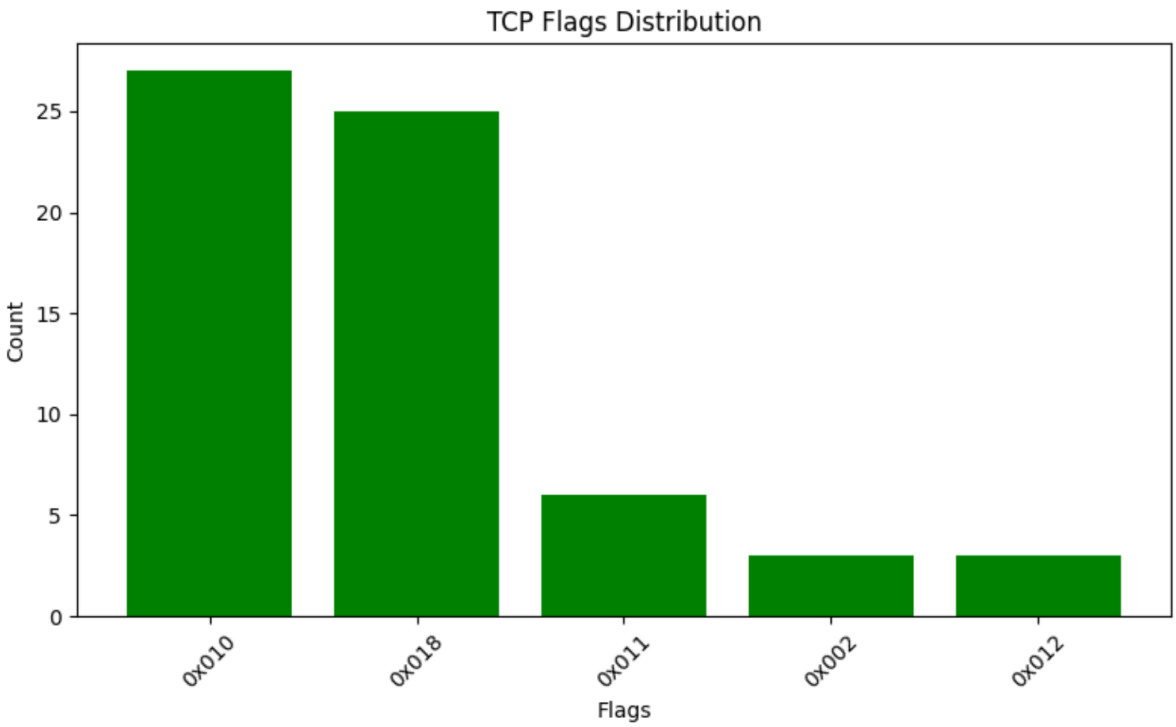
4. SYN (Synchronize) - 0x002

דגל זה נשלח בתחילת חיבור כדי להתחיל שיחה בין שני מחשבים.

5. SYN + ACK - 0x012

זהו שלב האישור בתהליך יצירת חיבור TCP .

נשלח מהשרת כתגובה לבקשת חיבור (SYN) .



דרך הגרף אפשר לראות:

1. הרבה חבילות עם(0x010) ACK
לכן נסיק כי התקשורת יציבה, ונתונים נשלחים ומאושרים ללא בעיה.

2. הרבה חבילות עם(0x018) PSH + ACK
המחשב מקבל נתונים מהר, וזה סימן לתעבורה פעילה ודינאמית למשל גלישה באינטרנט .

3. מעט חבילות עם (0x002) SYN ו(0x012) SYN + ACK-
כלומר אין ניסיונות רבים לפתוח חיבורים חדשים, מה שמצביע על תקשורת רגילה ולא התקפה.

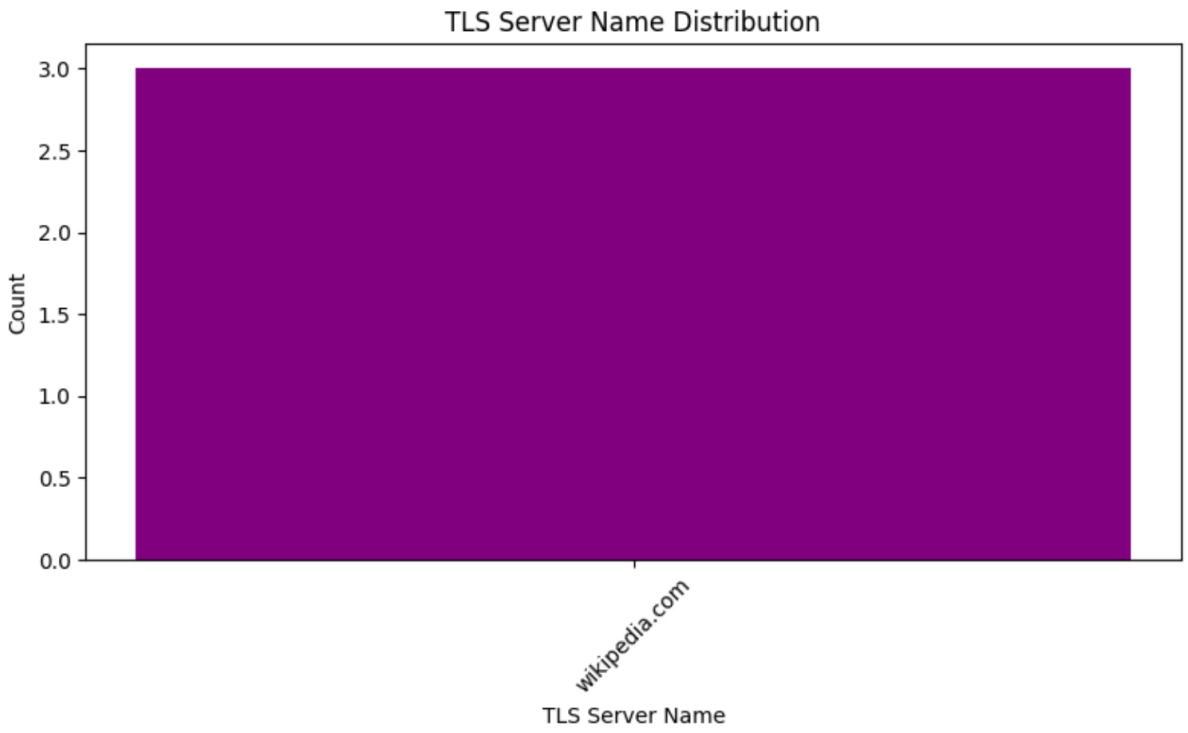
4. מעט חבילות עם(0x011) FIN + PSH + ACK
חלק מהחיבורים נסגרים, אבל זה לא חריג.

- גרף שלישי: שדות כותרות TLS

```
##############################################################################
# 3) TLS Header Fields Analysis
##############################################################################
if 'TLS Server Name' in df.columns and df['TLS Server Name'].notna().any():
    tls_counts = df['TLS Server Name'].value_counts()
    plt.figure(figsize=(8, 5))
    plt.bar(tls_counts.index, tls_counts.values, color='purple')
    plt.title('TLS Server Name Distribution')
    plt.xlabel('TLS Server Name')
    plt.ylabel('Count')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
else:
    print("No TLS Server Name data available or column missing.")
```

הגרף מציג התפלגות שמות השרתים (TLS Server Name) שנמצאו בכותרות.TLS
המשמעות היא שבחבילות שנלכדו, ניתן היה לזהות לאיזה שרת HTTPS התחברו.

- ציר הX-האופקי:
מציג את שם השרת שנמצא בהתקשרות המוצפנת –wikipedia.com .
זהו ה (Server Name Indication) SNI-שמאפשר ללקוח לדווח לשרת לאיזה דומיין הוא רוצה להתחבר.
- ציר הY-האנכי:
מציג כמה פעמים שם השרת הופיע בתעבורה ,כלומר כמה חבילות TLS כללו בקשה לשרת זה.

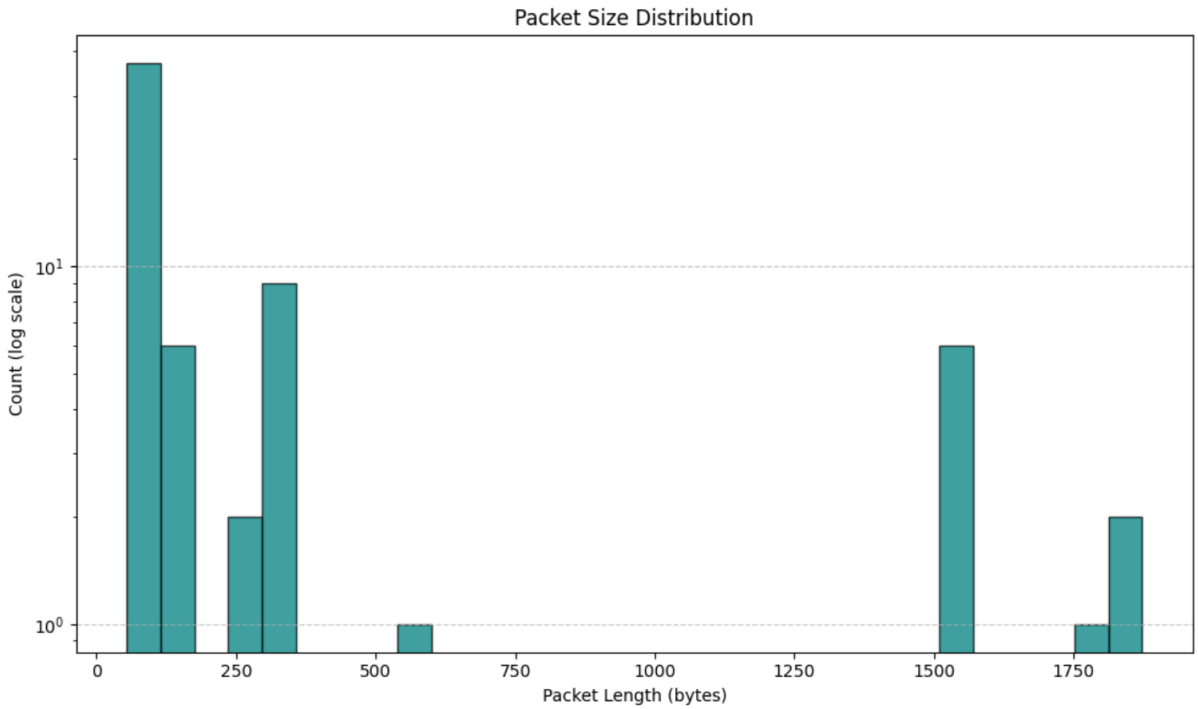


גרף רביעי : גודלי המנות

```python

##############################################################################
# 4) Packet Size Distribution - Improved Readability
##############################################################################
plt.figure(figsize=(10, 6))
plt.hist(df['Length'], bins=30, color='teal', edgecolor='black', alpha=0.75)
plt.yscale('log')  # Log scale for better visibility of smaller counts
plt.title('Packet Size Distribution')
plt.xlabel('Packet Length (bytes)')
plt.ylabel('Count (log scale)')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

הגרף מציג התפלגות של גדלי חבילות (Packet Size Distribution) שנמצאו בתעבורת הרשת.
המטרה היא להבין מהם הגדלים הנפוצים של חבילות הרשת שנלכדו במהלך הניתוח.

- ציר הX-האופקי:
מייצג את אורך החבילה בבתים ,(Bytes) כלומר גודל הנתונים שכל חבילה מכילה.
הערכים נעים בין 0ל-1800 בתים ,כאשר חבילות קטנות הן בקשות/אישורים, וחבילות גדולות מכילות מידע אמיתי.
- ציר ה Y-האנכי :
מייצג את מספר החבילות בגודל מסוים



ניתן לראות מהגרף כי
רוב החבילות קטנות מאוד (0-250 בתים)
חבילות אלו הן לרוב:

- בקשות/תשובות קצרות שלTCP (SYN, ACK) .
- חבילות שליטה בפרוטוקולים כמו TLS.
- בקשות קטנות ב HTTP (GET, POST) .

קיימות גם חבילות גדולות (1500 בתים ומעלה)

- גודל זה מצביע על חבילות נתונים מקסימליות.
- חבילות כאלה מכילות תוכן אמיתי במקרה שלנו של אתר אינטרנט ויקיפדיה.

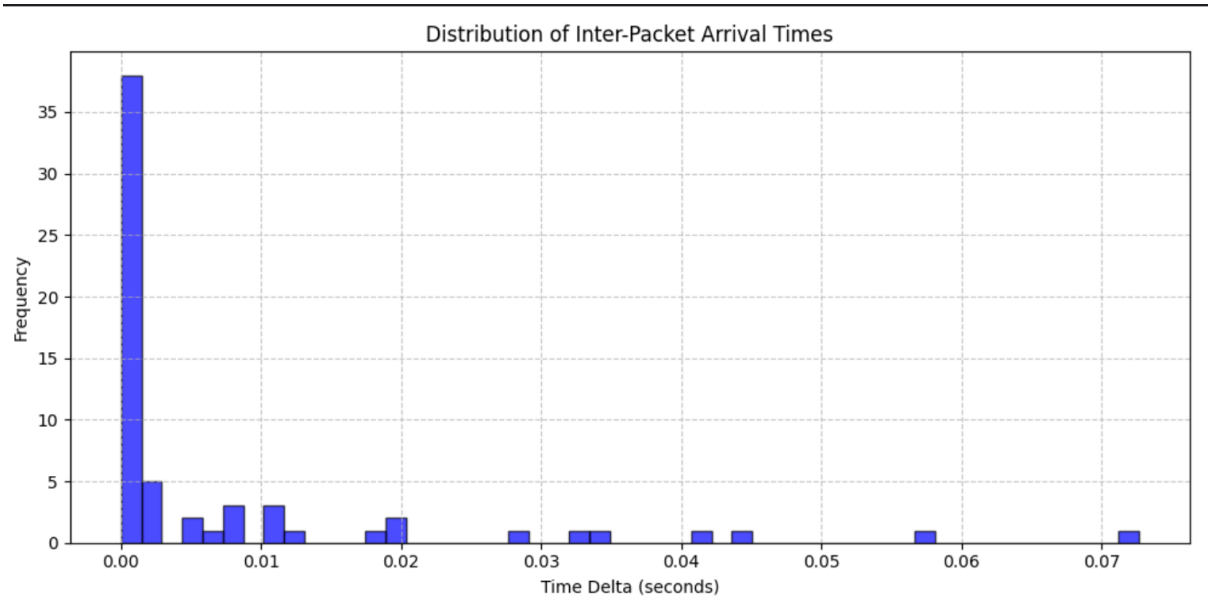
- גרף חמישי : זמן הגעה בין מנות (הזמן בין כל שתי מנות)

```
##############################################################################
# 5) Inter-Packet Time Differences
##############################################################################

df['time_delta'] = df['Time'].diff().fillna(0)

df_filtered = df[df['time_delta'] < df['time_delta'].quantile(0.99)].copy()
plt.figure(figsize=(10, 5))
plt.hist(df_filtered['time_delta'], bins=50, color='blue', edgecolor='black', alpha=0.7)

plt.title('Distribution of Inter-Packet Arrival Times')
plt.xlabel('Time Delta (seconds)')
plt.ylabel('Frequency')
plt.grid( visible: True, linestyle='--', alpha=0.7)
plt.tight_layout()

plt.show()
```

הגרף מציג התפלגות זמני ההגעה בין חבילות.(Inter-Packet Arrival Times Distribution)
זהו ניתוח של כמה זמן עבר בין קבלת חבילה אחת לחבילה הבאה.

- ציר ה X-האופקי:
  מציג את הפרשי הזמנים (Time Delta) בשניות בין חבילה אחת לשנייה.
  כלומר, כל ערך על הציר מייצג כמה שניות עברו בין קבלת חבילה אחת לחבילה שאחריה.

- ציר ה Y-האנכי:
  מייצג את תדירות – (Frequency) כלומר כמה פעמים התרחש פרק זמן מסוים בין חבילות.



ניתן להסיק מהגרף :

רוב החבילות מגיעות בהפרש זמן מאוד קטן (קרוב ל-0 שניות) כלומר המשמעות :התעבורה מאוד אינטנסיבית ורציפה. כאן רואים שהכל מתרכז קרוב ל-0 שניות ,כלומר התקשורת זורמת מהר וללא בעיות גדולות.

- גרף שישי :גודל זרימה (המספר הכולל של המנות)

```python

##############################################################################
# 6) Flow Size (Number of Packets per Flow) - Improved Readability
##############################################################################
flow_cols = ['Source', 'Destination', 'Protocol']
if set(flow_cols).issubset(df.columns):
    flow_group = df.groupby(flow_cols)
    flow_size = flow_group.size().reset_index(name='packet_count')
    flow_size_sorted = flow_size.sort_values('packet_count', ascending=False)
    flows = [f"{row['Source']}→{row['Destination']}({row['Protocol']})" for _, row in flow_size_sorted.iterrows()]

    plt.figure(figsize=(10, 6))
    plt.barh(flows, flow_size_sorted['packet_count'], color='darkcyan')
    plt.title('Flow Size Distribution (Number of Packets per Flow)')
    plt.xlabel('Packet Count')
    plt.ylabel('Flow (Src→Dst)')
    plt.gca().invert_yaxis()
    plt.tight_layout()
    plt.show()
else:
    print("Flow columns (Source, Destination, Protocol) not available.")
```

הגרף מציג התפלגות של זרמי תעבורה (Flow Size Distribution) לפי מספר החבילות בכל זרם.
"זרם (Flow) "מוגדר כהעברת מידע בין מקור (Source) ליעד (Destination) בפרוטוקול מסוים.
המטרה היא להבין איזה חיבורים היו פעילים ביותר וכמה חבילות נשלחו בכל אחד מהם.

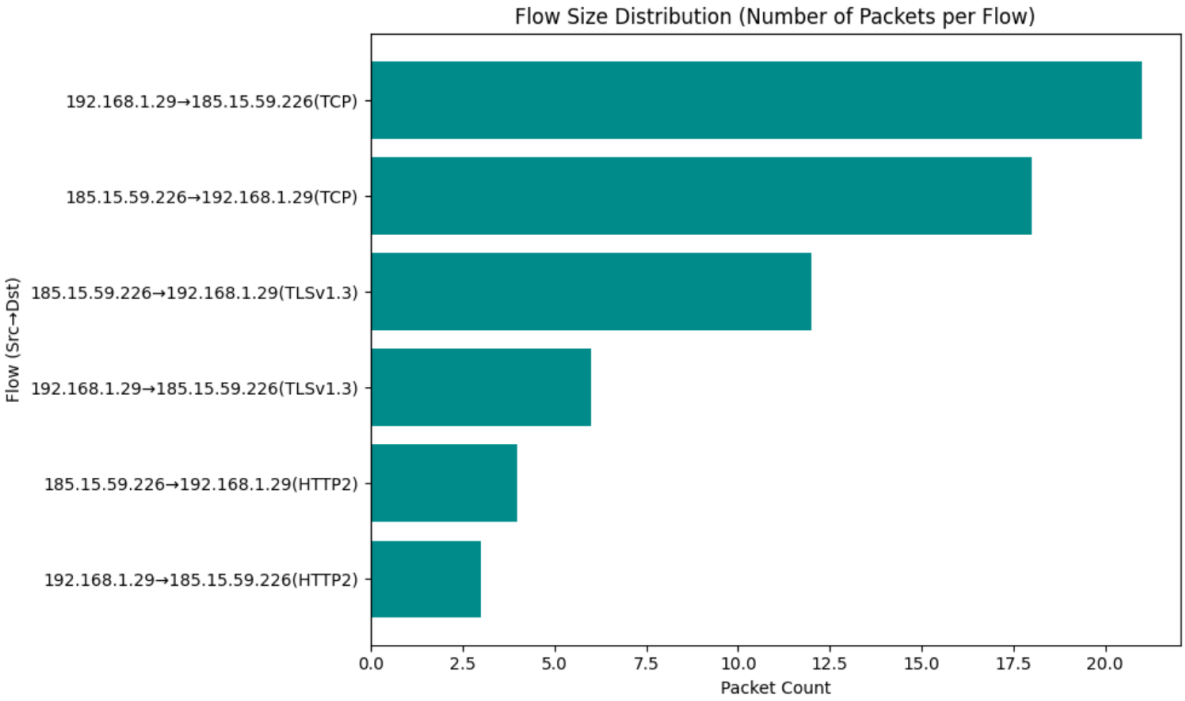
- ציר הX-האופקי: מציג את מספר החבילות (Packet Count) בכל זרם.
  כלומר, ככל שהעמודה ארוכה יותר הזרם כלל יותר חבילות.
- ציר ה Y-האנכי:
  מציג את הזרמים (Flows) .

לדוגמה:

(TCP) 185.15.59.226 → 192.168.1.29זרם TCP מהמחשב המקומי לוויקיפדיה.

(TLSv1.3) 192.168.1.29 → 185.15.59.226 זרם TLS מאתר ויקיפדיה למחשב שלי.

(HTTP2) 185.15.59.226 → 192.168.1.29 זרם HTTP/2 בין המחשב לשרת.



מה ניתן ללמוד מהגרף :

זרמי ה TCP-(בקצה העליון של הגרף) הם הפעילים ביותר

- הזרם (TCP) 185.15.59.226 → 192.168.1.29מכיל הכי הרבה חבילות.
- המשמעות :הרבה תעבורת TCP נשלחה מהמחשב המקומי לוויקיפדיה.

TLSv1.3 קיבל מספר חבילות משמעותי

- זרם (TLSv1.3) 192.168.1.29 → 185.15.59.226מופיע בכמות גדולה.
- המשמעות :השרת של ויקיפדיה שלח הרבה חבילות TLS חזרה.

HTTP/2 נמצא בתחתית עם מספר חבילות קטן יחסית

- זרמי HTTP/2מופיעים עם פחות חבילות מאשר TCP או TLS .
- המשמעות :מעט בקשות HTTP בוצעו ,או שהן היו קצרות.

- הגרף השביעי -נפח הזרימה (המספר הכולל של הבייתים ששודרו)

```python

##############################################################################
# 7) Flow Volume (Total Bytes per Flow)
##############################################################################
if set(flow_cols).issubset(df.columns) and 'Length' in df.columns:
    flow_volume = flow_group['Length'].sum().reset_index(name='total_bytes')
    flow_volume_sorted = flow_volume.sort_values('total_bytes', ascending=False)
    flows_vol = [f"{row['Source']}→{row['Destination']}({row['Protocol']})" for _, row in flow_volume_sorted.iterrows()]

    plt.figure(figsize=(10, 6))
    plt.barh(flows_vol, flow_volume_sorted['total_bytes'], color='sienna')
    plt.title('Flow Volume Distribution (Total Bytes per Flow)')
    plt.xlabel('Total Bytes')
    plt.ylabel('Flow (Src→Dst)')
    plt.gca().invert_yaxis()
    plt.tight_layout()
    plt.show()
else:
    print("Flow volume columns or 'Length' not available.")

```

הגרף מציג התפלגות נפח הנתונים בכל זרם (Flow Volume Distribution) לפי סך הבתים (Bytes)שהועברו בכל זרם.
כל זרם (Flow) מוגדר לפי:

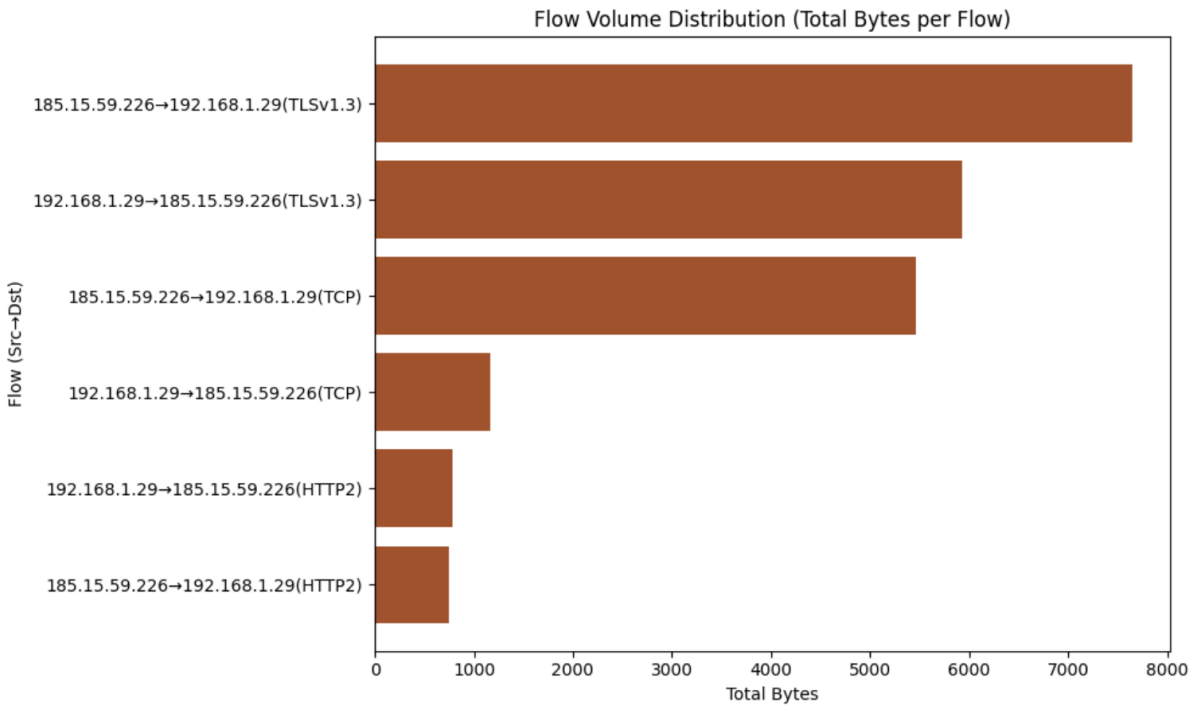
1. כתובת מקור(Source)
2. כתובת יעד(Destination)
3. הפרוטוקול שבו נעשה שימוש (TCP, TLS, HTTP/2)

- ציר ה X-האופקי:
  מציג את סך הנתונים (Total Bytes) שהועברו בכל זרם.
  כלומר, ככל שהעמודה ארוכה יותר יותר נתונים עברו באותו הזרם.
- ציר הY-האנכי:
  מציג את הזרמים (Flows)

לדוגמה:

– 185.15.59.226 → 192.168.1.29 (TLSv1.3) נתונים שנשלחו מהשרת (ויקיפדיה) למחשב שלי .

– 192.168.1.29 → 185.15.59.226 (TCP) נתונים שנשלחו מהמחשב שלי לשרת.



ניתן ללמוד מהגרף :

TLSv1.3 מכיל את נפח הנתונים הגדול ביותר

- 185.15.59.226 → 192.168.1.29 (TLSv1.3)הוא הזרם עם הכי הרבה נתונים.
- המשמעות :השרת של ויקיפדיה שלח הכי הרבה נתונים של דף אינטרנט
- זרמים של TCP מכילים גם כמות משמעותית של נתונים.
- המשמעות :הרבה נתונים נשלחו גם בשכבת TCP לפני שהוצפנו בTLS

HTTP/2

- זרמים של HTTP/2 כוללים פחות נתונים ביחס ל TLS-וTCP
- המשמעות :בקשות קצרות ולא העברת קבצים גדולים.

חלק 4 :

נדרש מאיתנו להקליט שוב את תעבורת רשת שלנו , בהנחה שאנחנו התוקף ואנחנו מנסים ללמוד אילו אפליקציות המשתמש ניגש אליהם .

מטרת הבדיקה היא לנתח עד כמה ניתן לזהות אתרים ואפליקציות שבהם השתמש המשתמש, גם כאשר התעבורה מוצפנת. נבדוק שני תרחישים:

1. התקפה עם – Flow ID כאשר התוקף יודע את גודל החבילות, חותמות הזמן, ואת מזהה הזרם (Flow ID) שמבוסס על כתובות ה IP-והפורט.
2. התקפה ללא – Flow ID כאשר התוקף יודע רק את גודל החבילות וחותמות הזמן.

- נתחיל במקרה הראשון : התקפה עם Flow ID

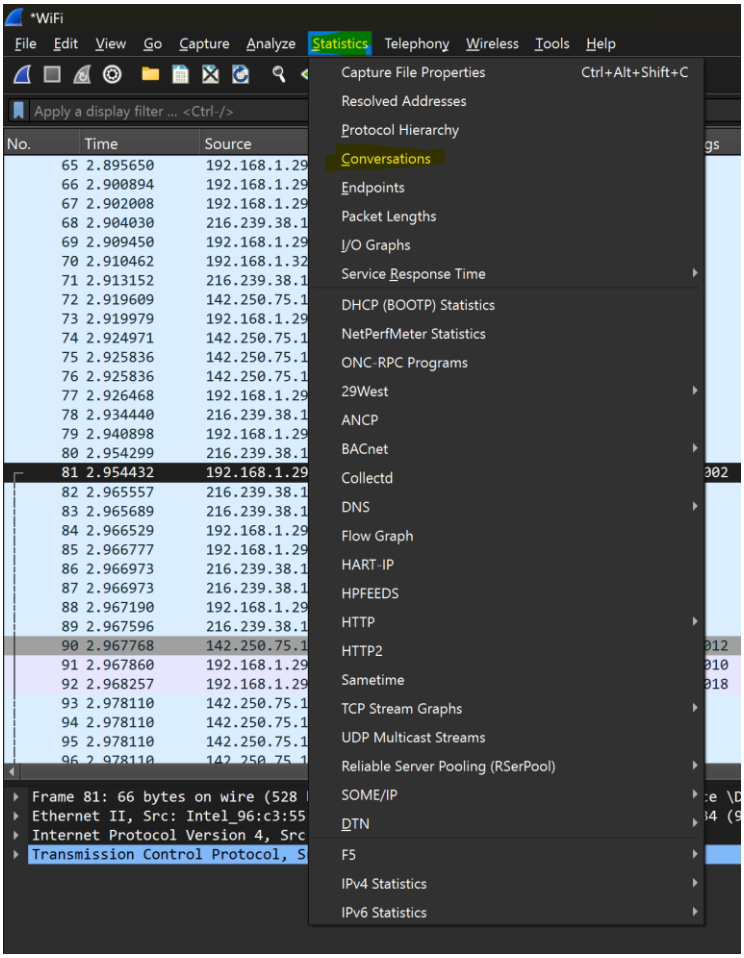
מטרתי היא להראות שהתוקף יכול לזהות עם אילו שרתים המשתמש תקשר ,איזה פרוטוקולים היו בשימוש ,וכיצד זרמים שונים של תעבורה עוזרים בזיהוי אתרים ואפליקציות בהינתן התרחיש הראשון.

הקלטתי את ההקלטה מחדש באותו אופן נכנסתי לאתר Wikipedia.com
נכנסתי שם לערך כלשהו שיהיה מעניין ויצאתי מהאתר והתחלתי את ההקלטה.

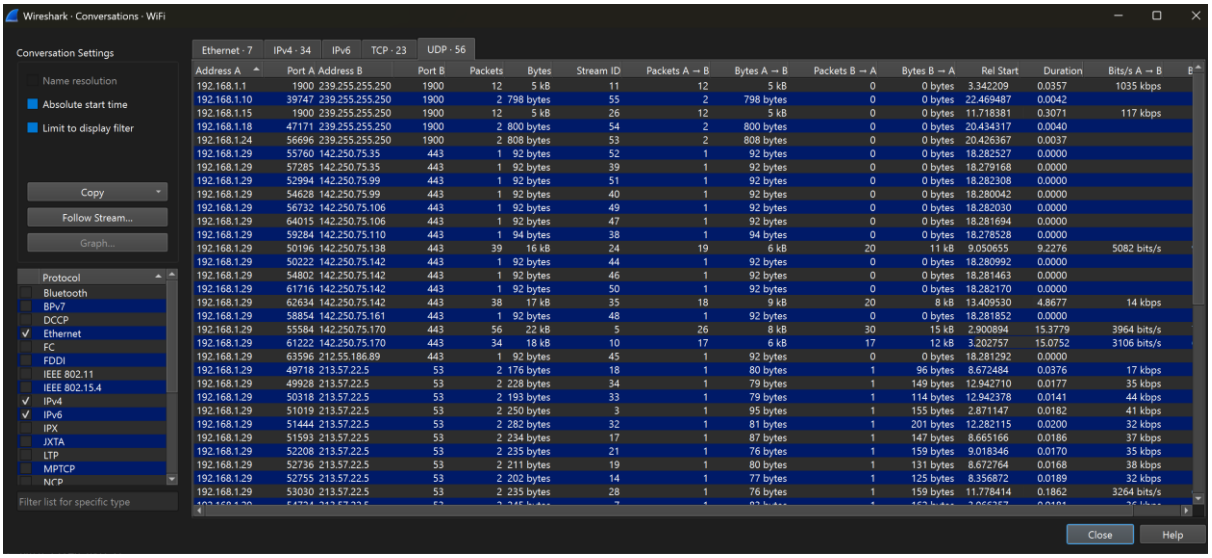
שם ההקלטה נשמר : part4_wikipedia.pcapng

1.שלב ראשון נזהה את כל הזרמים שהתבצעו כולל כתובות IP ופרוטוקולים

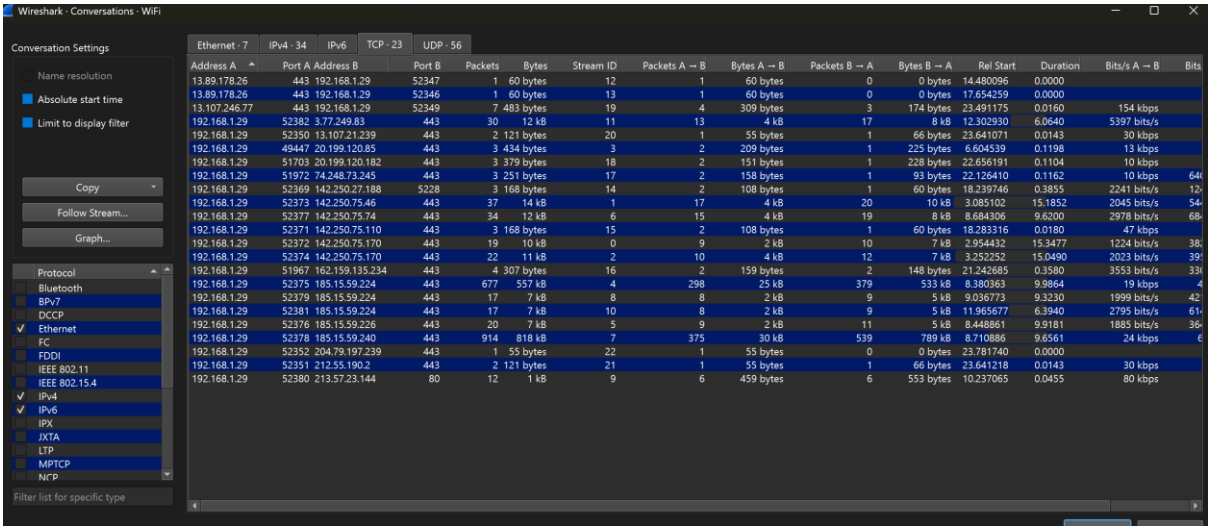
ניגש למידע שלנו כך



תוצאות עבור חיבורי UDP



תוצאות עבור חיבורי TCP



Address A	Port A	Address B	Port B	Packets	Bytes	Stream ID	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits
13.89.178.26	443	192.168.1.29	52347	1	60 bytes	12	1	60 bytes	0	0 bytes	14.480096	0.0000		
13.89.178.26	443	192.168.1.29	52346	1	60 bytes	13	1	60 bytes	0	0 bytes	17.654259	0.0000		
13.107.246.77	443	192.168.1.29	52349	7	483 bytes	19	4	309 bytes	3	174 bytes	23.491175	0.0160	154 kbps	
192.168.1.29	52382	3.77.249.83	443	30	12 kB	11	13	4 kB	17	8 kB	12.302930	6.0640	5397 bits/s	
192.168.1.29	52350	13.107.21.239	443	2	121 bytes	20	1	55 bytes	1	66 bytes	23.641071	0.0143	30 kbps	
192.168.1.29	49447	20.199.120.85	443	3	434 bytes	3	2	209 bytes	1	225 bytes	6.604539	0.1198	13 kbps	
192.168.1.29	51703	20.199.120.182	443	3	379 bytes	18	2	151 bytes	1	228 bytes	22.656191	0.1104	10 kbps	
192.168.1.29	51972	74.248.73.245	443	3	251 bytes	17	2	158 bytes	1	93 bytes	22.126410	0.1162	10 kbps	64
192.168.1.29	52369	142.250.27.188	5228	3	168 bytes	14	2	108 bytes	1	60 bytes	18.339746	0.3855	2241 bits/s	12
192.168.1.29	52373	142.250.75.46	443	37	14 kB	1	17	4 kB	20	10 kB	3.085102	15.1852	2045 bits/s	54
192.168.1.29	52377	142.250.75.74	443	34	12 kB	6	15	4 kB	19	8 kB	8.684306	9.6200	2978 bits/s	68
192.168.1.29	52371	142.250.75.110	443	3	168 bytes	15	2	108 bytes	1	60 bytes	18.283316	0.0180	47 kbps	
192.168.1.29	52372	142.250.75.170	443	19	10 kB	0	9	2 kB	10	7 kB	2.954432	15.3477	1224 bits/s	38
192.168.1.29	52374	142.250.75.170	443	22	11 kB	2	10	4 kB	12	7 kB	3.252252	15.0490	2023 bits/s	39
192.168.1.29	51967	162.159.135.234	443	4	307 bytes	16	2	159 bytes	2	148 bytes	21.242685	0.3580	3553 bits/s	33
192.168.1.29	52375	185.15.59.224	443	677	557 kB	4	298	25 kB	379	533 kB	8.380363	9.9864	19 kbps	4
192.168.1.29	52379	185.15.59.224	443	17	7 kB	8	8	2 kB	9	5 kB	9.036773	9.3230	1999 bits/s	42
192.168.1.29	52381	185.15.59.224	443	17	7 kB	10	8	2 kB	9	5 kB	11.965677	6.3940	2795 bits/s	61
192.168.1.29	52376	185.15.59.226	443	20	7 kB	5	9	2 kB	11	5 kB	8.448861	9.9181	1885 bits/s	36
192.168.1.29	52378	185.15.59.240	443	914	818 kB	7	375	30 kB	539	789 kB	8.710886	9.6561	24 kbps	6
192.168.1.29	52352	204.79.197.239	443	1	55 bytes	22	1	55 bytes	0	0 bytes	23.781740	0.0000		
192.168.1.29	52351	212.55.190.2	443	2	121 bytes	21	1	55 bytes	1	66 bytes	23.641218	0.0143	30 kbps	
192.168.1.29	52380	213.57.23.144	80	12	1 kB	9	6	459 bytes	6	553 bytes	10.237065	0.0455	80 kbps	

ניתן לראות שיש חשיפה מלאה לפרוטוקולים שיוצרים איתם חיבור במהלך היקשרות לאתר מאובטח

כלומר ניתן לראות זרמים נפרדים עם כתובות IP שונות זה מוכיח שהתוקף יכול לזהות לאילו שרתים המשתמש התחבר למצוא דרכם נקודות חולשה ולנסות לתקוף משם .

לסיכום תוקף יכול לראות את כל הזרמים בתעבורה דרך Wireshark. ניתן לזהות אילו כתובות IPהתקשרו ביניהן, באיזה פורטים, וכמה נתונים הועברו .זיהוי זה מאפשר לתוקף להסיק איזה אתרים או שירותים ביקר המשתמש ,גם ללא גישה ישירה לתוכן החבילות.

2.שלב שני סינון לפי Flow ID :

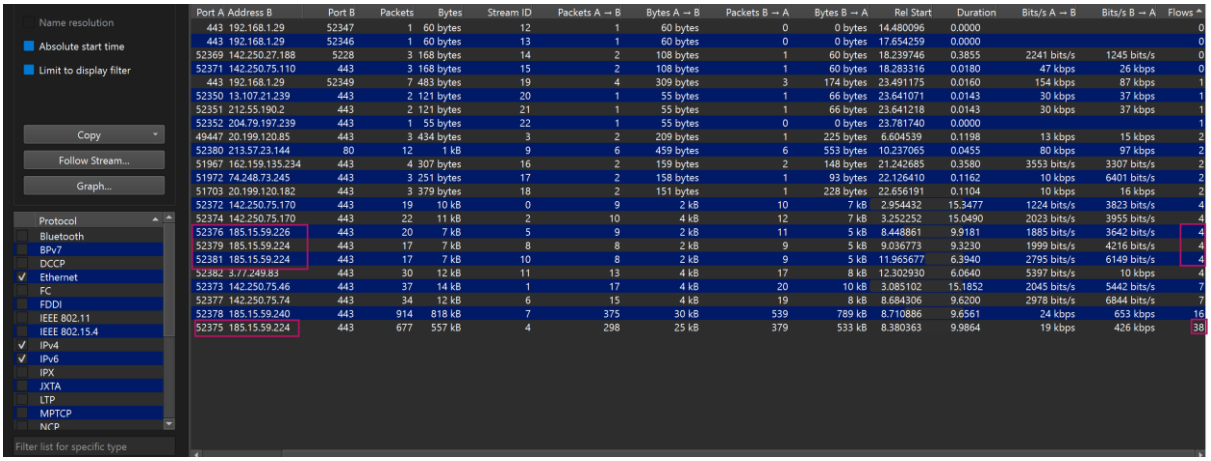
כאן אנו נוכיח שהכתובות 185.15.59.224 185.15.59.226שייכות לויקיפדיה על ידי הרצת הפקודה שצילום מסך בשורת הפקודה

```
C:\Users\ronam>nslookup 185.15.59.226
Server:  ns2-cache.hotnet.net.il
Address:  213.57.22.5

Name:    ncredir-lb.esams.wikimedia.org
Address:  185.15.59.226


C:\Users\ronam>
```

```
C:\Users\ronam>nslookup 185.15.59.224
Server:  ns2-cache.hotnet.net.il
Address:  213.57.22.5

Name:    text-lb.esams.wikimedia.org
Address:  185.15.59.224


C:\Users\ronam>
```

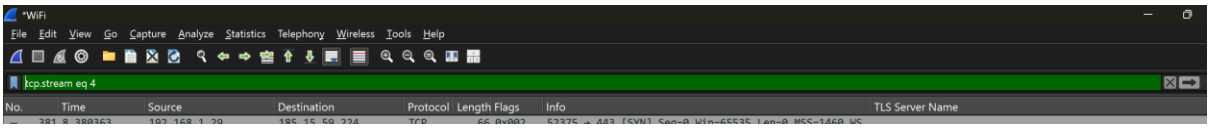
כעת ניקח את המידע שיש לנו על חיבורים וננצל אותו לגלות מידע

ניקח את הכתובות שרלוונטיות האלו וניקח את מספר הזרם ונבצע סינון לפי זה



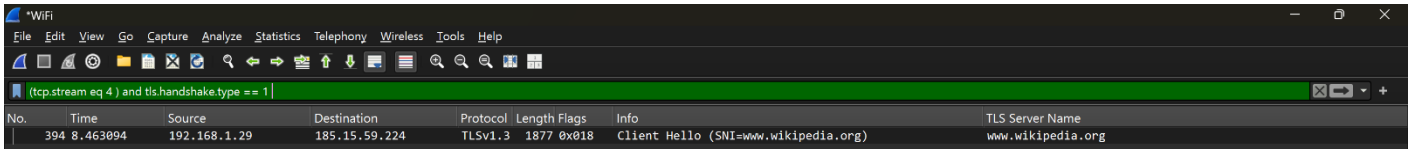
Port A	Address B	Port B	Packets	Bytes	Stream ID	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A	Flows
443	192.168.1.29	52347	1	60 bytes	12	1	60 bytes	0	0 bytes	14.480096	0.0000			0
443	192.168.1.29	52346	1	60 bytes	13	1	60 bytes	0	0 bytes	17.654259	0.0000			0
52369	142.250.27.188	5228	3	168 bytes	14	2	108 bytes	1	60 bytes	18.239746	0.3855	2241 bits/s	1245 bits/s	0
52371	142.250.75.110	443	3	168 bytes	15	2	108 bytes	1	60 bytes	18.283316	0.0180	47 kbps	26 kbps	0
443	192.168.1.29	52349	7	483 bytes	19	4	309 bytes	3	174 bytes	23.491175	0.0160	154 kbps	87 kbps	1
52350	13.107.21.239	443	2	121 bytes	20	1	55 bytes	1	66 bytes	23.641071	0.0143	30 kbps	37 kbps	1
52351	212.55.190.2	443	2	121 bytes	21	1	55 bytes	1	66 bytes	23.641218	0.0143	30 kbps	37 kbps	1
52352	204.79.197.239	443	1	55 bytes	22	1	55 bytes	0	0 bytes	23.781740	0.0000			1
49447	20.199.120.85	443	3	434 bytes	3	2	209 bytes	1	225 bytes	6.604539	0.1198	13 kbps	15 kbps	2
52380	213.57.23.144	80	12	1 kB	9	6	459 bytes	6	553 bytes	10.237065	0.0455	80 kbps	97 kbps	2
51967	162.159.135.234	443	4	307 bytes	16	2	159 bytes	2	148 bytes	21.242685	0.3580	3553 bits/s	3307 bits/s	2
51972	74.248.73.245	443	3	251 bytes	17	2	158 bytes	1	93 bytes	22.126410	0.1162	10 kbps	6401 bits/s	2
51703	20.199.120.182	443	3	379 bytes	18	2	151 bytes	1	228 bytes	22.656191	0.1104	10 kbps	16 kbps	2
52372	142.250.75.170	443	19	10 kB	0	9	2 kB	10	7 kB	2.954432	15.3477	1224 bits/s	3823 bits/s	4
52374	142.250.75.170	443	22	11 kB	2	10	4 kB	12	7 kB	3.252252	15.0490	2023 bits/s	3955 bits/s	4
52376	185.15.59.226	443	20	7 kB	5	9	2 kB	11	5 kB	8.448861	9.9181	1885 bits/s	3642 bits/s	4
52379	185.15.59.224	443	17	7 kB	8	8	2 kB	9	5 kB	9.036773	9.3230	1999 bits/s	4216 bits/s	4
52381	185.15.59.224	443	17	7 kB	10	8	2 kB	9	5 kB	11.965677	6.3940	2795 bits/s	6149 bits/s	4
52382	3.77.249.83	443	30	12 kB	11	13	4 kB	17	8 kB	12.302930	6.0640	5397 bits/s	10 kbps	4
52373	142.250.75.46	443	37	14 kB	1	17	4 kB	20	10 kB	3.085102	15.1852	2045 bits/s	5442 bits/s	7
52377	142.250.75.74	443	34	12 kB	6	15	4 kB	19	8 kB	8.684306	9.6200	2978 bits/s	6844 bits/s	7
52378	185.15.59.240	443	914	818 kB	7	375	30 kB	539	789 kB	8.710886	9.6561	24 kbps	653 kbps	16
52375	185.15.59.224	443	677	557 kB	4	298	25 kB	379	533 kB	8.380363	9.9864	19 kbps	426 kbps	38

נבצע סינון לפי זרם 4 כי אנחנו רואים תוצאות גבוהות מאוד



הסינון הזה מסנן רק את החבילות ששייכות לאותו חיבור TCP ספציפי בין כתובת מקור לכתובת יעד. כל חבילה שמופיעה במסנן הזה היא חלק מאותו הזרם –כלומר, הן נשלחו בין אותו מקור ויעד, באותו פורט.

כעת נוסיף סינון על מנת לבדוק האם ניתן לראות לאיזה אתר התבצעה הבקשה



הזרם הזה הוא חיבור HTTPS בין המחשב (192.168.1.29) לשרת היעד (185.15.59.224 - ויקיפדיה).

החלק tls.handshake.type ==1

מסנן רק את חבילות ה TLS-מסוגClientHello חבילה זו היא השלב הראשון בתקשורת TLS, שבו הדפדפן שולח בקשה להתחבר לשרת.

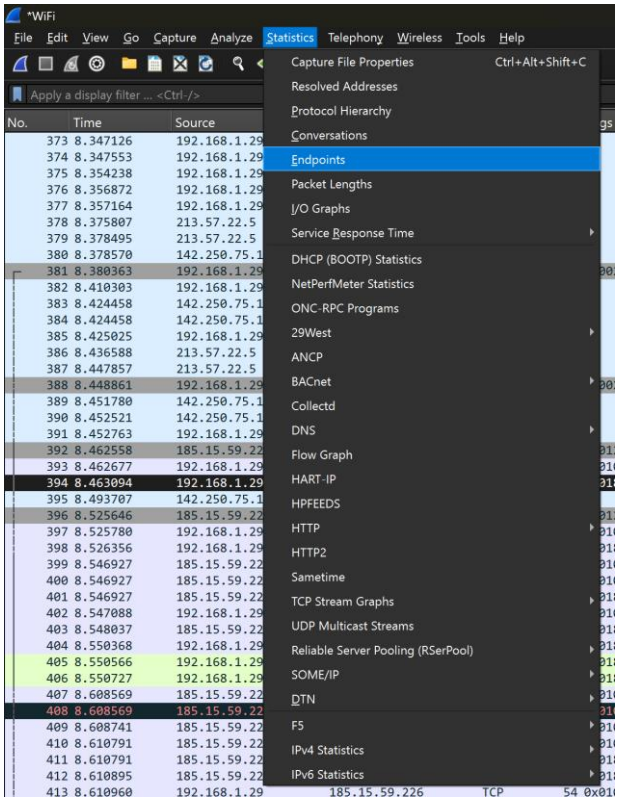
מופיע בעמודה "TLS Server Name"בתור www.wikipedia.org.

משמעות הדבר :למרות שהתוכן מוצפן, ניתן לראות לאיזה אתר התבצעה הבקשה.

צילום מסך מראה כי למרות שהתוכן מוצפן, ניתן לזהות את האתר Wikipedia על ידי ה SNI-המופיע בחבילת ה TLS ClientHello-עובדה זו מאפשרת לתוקף לדעת לאילו אתרים המשתמש נכנס, גם מבלי לראות את תוכן החבילות.

לסיכום באמצעות המסנן tcp.stream eq X תוקף יכול להתמקד בזרם מסוים ולראות את כל החבילות ששייכות לחיבור ספציפי ואם הזרם מכיל תעבורת TLS, ניתן לזהות את שם השרת (SNI) למרות שהתוכן מוצפן .במקרה זה, זוהה www.wikipedia.org, שמוכיח שתוקף יכול לדעת שהמשתמש התחבר לויקיפדיה.

3. ניתוח נפח הנתונים שנשלח לכל שרת: נרצה להראות שהתוקף יכול להעריך איזה אתר השתמש ביותר נתונים ,מה שעוזר לנחש סוג השירות. ניגש לפה



כתובת היעד 185.15.59.226ו185.15.59.224-

- אלו שרתים של Wikipedia .
- הנתונים מראים 711חבילות ו-572 KB של תעבורה –כלומר, כמות משמעותית של נתונים הועברה.
- מסקנה :תוקף יכול להסיק שהמשתמש ביקר בוויקיפדיה.

הכתובת 213.57.22.5

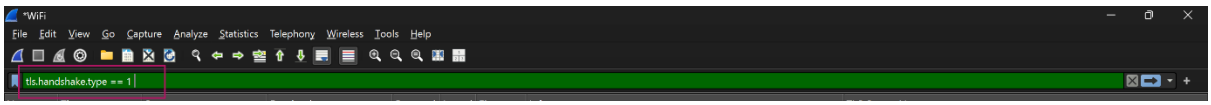
- כתובת זו מופיעה גם בתוצאות DNS, והיא ככל הנראה שרת Proxy/Cache שמטפל בתעבורתWikipedia .
- מסקנה :התוקף יכול להבין שהתעבורה עברה דרך שירות תיווך CDN או שרתProxy .

כתובת ה192.168.1.29

- זו כתובת ה IP-של המחשב שלי ברשת המקומית.
- הנתונים מראים 2,267חבילות שנשלחו, ו-2 MBשל נתונים סה"כ.
- מסקנה :תוקף ברשת המקומית יכול לזהות כמה נתונים שלחת וקיבלת, ולשער את סוג הפעילות של המשתמש.

לסיכום תוקף יכול להשתמש בזה כדי לבדוק כמה נתונים נשלחו לכל כתובת IP. ניתן לראות שהשרת 185.15.59.226קיבל כמות משמעותית של נתונים, דבר שמסגיר שימוש באתר ויקיפדיה . זיהוי נפח הנתונים יכול לאפשר לתוקף לנחש אם מדובר בגלישה רגילה, סטרימינג או הורדה.

4. זיהוי חיבורי TLS והצגת שם השרת(SNI)



השתמשנו בסינון הזה על מנת להראות שהתוקף יכול לזהות את שם השרת (SNI) בתקשורת מוצפנת, מה שעוזר לו לדעת איזה אתר ביקר המשתמש.

כלומר אם ניתן לראות שמות שרתים ב SNI-, זה מוכיח שתוקף יכול לזהות במדויק איזה אתרים ביקר המשתמש למרות שהתוכן מוצפן.

לסיכום , תוקף יכול להפעיל את המסנן tls.handshake.type == 1 ולזהות את שם האתר דרך SNIבמקרה זה, זוהה www.wikipedia.org, למרות שהתוכן עצמו מוצפן עובדה זו מאפשרת לתוקף לדעת בדיוק לאילו אתרים המשתמש נכנס.

סיכום תרחיש ראשון :

נסיק מכל הממצאים שמצאנו כי מסקנה התוקף יכול לזהות את האתרלמרות שהתוכן מוצפן.

- זיהוי כתובות ה IP-של השרתים מאפשר לתוקף לשייך אותם לשירותים ידועים.
- השימוש ב TLS Handshake-חושף את שם האתר ,(SNI) אלא אם כן נעשה שימוש ב- Encrypted SNI (ESNI).
- ניתוח נפח הנתונים להסיק את סוג השירות (גלישה, סטרימינג, צ'אט)

אם תוקף יודע את ,Flow ID הוא יכול לזהות אתרים בקלות יחסית .אם יש לו רק גודל חבילות וזמן, הוא עדיין יכול לזהות דפוסים של שימוש . ניתוח זה מוכיח שאפילו כאשר התעבורה מוצפנת, עדיין קיימת אפשרות לזהות פעילות משתמשים – ולכן יש לנקוט באמצעי אבטחה מתאימים.

כיצד ניתן למנוע מתקפה כזו?

1. שימוש ב VPNאו Tor מונע מתוקף לזהות את כתובת הIP ולהצליב את הנתונים עם שרתים ידועים.
2. Encrypted SNI (ESNI) או ECH (Encrypted Client Hello) מסתיר את שם השרת בשלב הTLS Handshake
3. DNS-over-HTTPS (DoH)או DNS-over-TLS (DoT) מונעים מתוקפים לראות בקשות DNS ולזהות אתרים לפי שאילתות DNS גלויות.
4. Traffic Obfuscationערפול תעבורה – שימוש בכלים כמו Obfsproxy, Shadowsocksאו Masquerading שמקשים על זיהוי התעבורה
5. הוספת רעש לתעבורה – (Traffic Padding) טכניקות הוספת חבילות מזויפות מסייעות להסוות נפח וזמן הגעת חבילות.
6. שימוש בפרוטוקולי HTTP/3 ו – QUIC-מאחדים חיבורים ומשתמשים בהצפנה חזקה יותר שמקשה על תוקפים לזהות זרמים בודדים.
7. שימוש ברשתות Mix Networks כגון I2P או – Garlic Routing טכניקות שמערבבות תעבורה בין משתמשים שונים ומונעות מעקב מבוסס זמן ונפח.

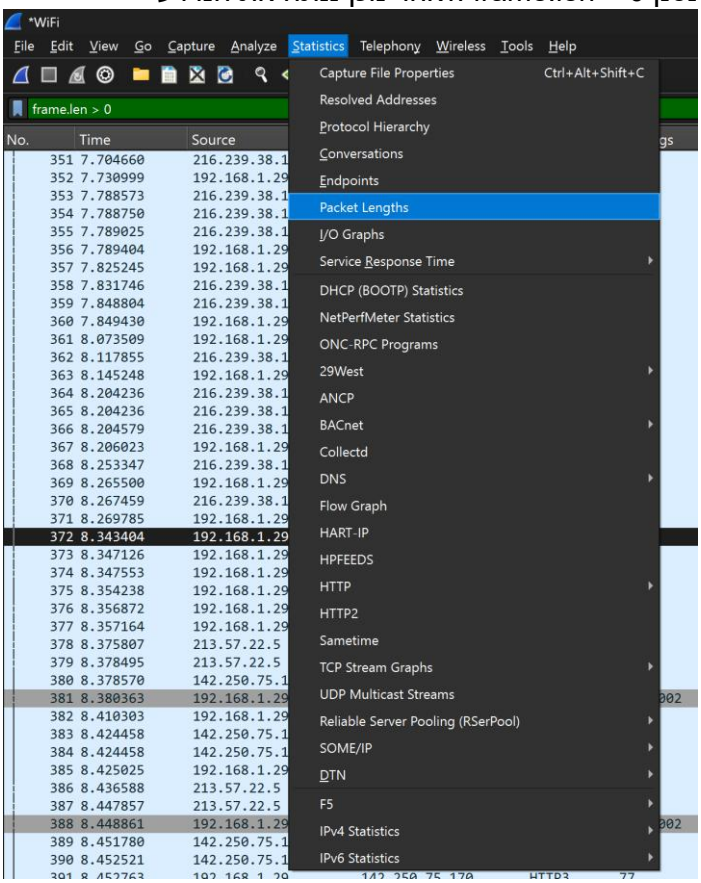
תרחיש שני - כאשר התוקף יודע רק את גודל החבילות וחותמות הזמן.

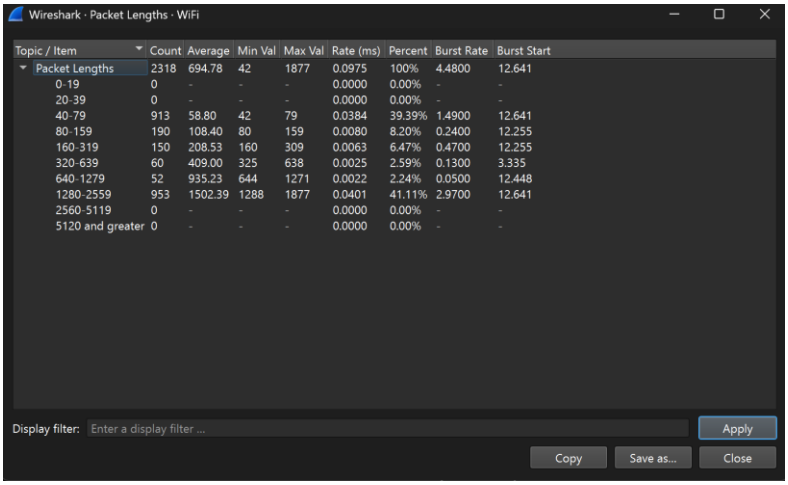
הארה: נשתמש באותה הקלטה מהתרחיש הראשון ,אבל ננתח אותה בצורה שונה כדי לדמות מצב שבו לתוקף אין מידע על ,Flow ID אלא רק גודל חבילות וחותמות זמן.

- שלב ראשון: ניתוח גודל החבילות

נרצה לראות כיצד גודל החבילות משתנה בתעבורה ולהסיק אם מדובר בתעבורת גלישה, חיפוש או סטרימינג.

נסנן frame.len > 0 ולאחר מכן ננתח את המידע





Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
Packet Lengths	2318	694.78	42	1877	0.0975	100%	4.4800	12.641
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	913	58.80	42	79	0.0384	39.39%	1.4900	12.641
80-159	190	108.40	80	159	0.0080	8.20%	0.2400	12.255
160-319	150	208.53	160	309	0.0063	6.47%	0.4700	12.255
320-639	60	409.00	325	638	0.0025	2.59%	0.1300	3.335
640-1279	52	935.23	644	1271	0.0022	2.24%	0.0500	12.448
1280-2559	953	1502.39	1288	1877	0.0401	41.11%	2.9700	12.641
2560-5119	0	-	-	-	0.0000	0.00%	-	-
5120 and greater	0	-	-	-	0.0000	0.00%	-	-

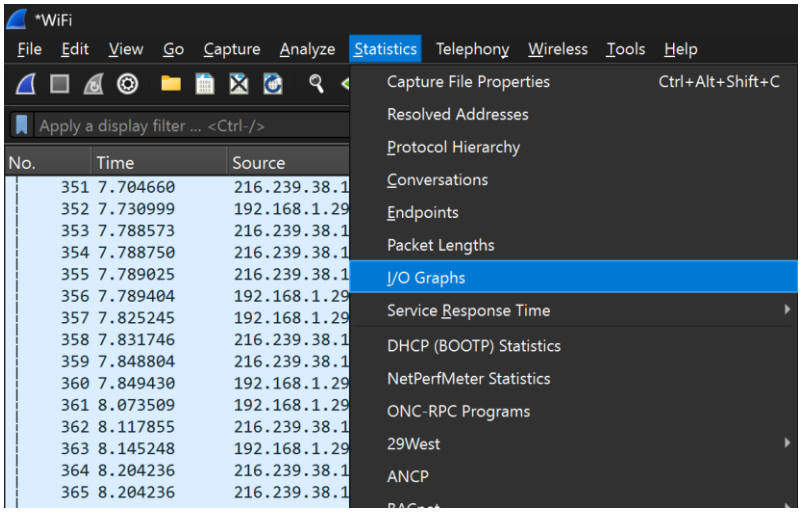
אם יש הרבה חבילות קטנות (50-200 בייתים), מדובר בתקשורת קלה כמו חיפוש בגוגל או שליחת הודעות .

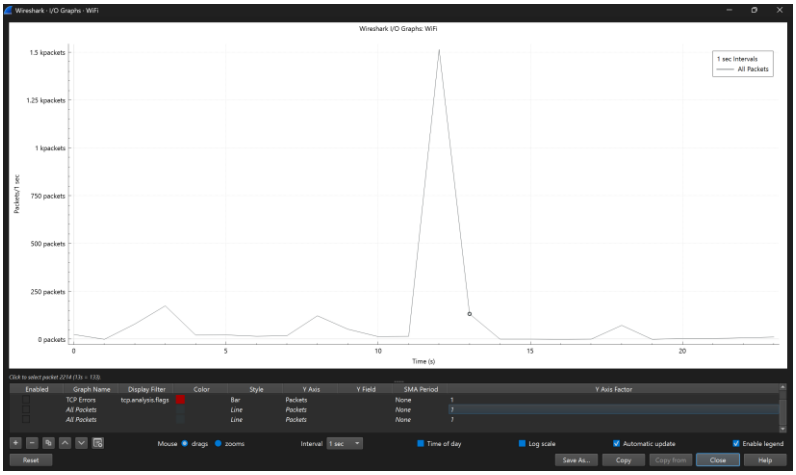
אם יש הרבה חבילות גדולות (1500 בייתים), מדובר כנראה בתעבורת וידאו או הורדת קבצים.

לסיכום , תוקף יכול לראות התפלגות גודל החבילות.
חבילות קטנות מעידות על בקשות DNS, HTTP, או חיפושים בגוגל.
חבילות גדולות ועקביות מצביעות על הורדות קבצים או סטרימינג.
אם קיימות חבילות רבות בגודל 1500 בייתים, ניתן לשער שהמשתמש הוריד קובץ או צפה בסרטון.
אם רוב החבילות קטנות, זה עשוי להצביע על גלישה באתרים עם תוכן טקסטואלי כמו ויקיפדיה.

- שלב 2: זיהוי דפוסי תעבורה לפי IO Graphs

נרצה לראות את התפלגות התעבורה לאורך הזמן ולהבין אם המשתמש ביצע טעינת אתר, סטרימינג, או גלישה רגילה.





1. בזמן X הייתה עלייה חדה בכמות החבילות דבר המעיד על טעינת עמוד או משאב כבד.
2. לאחר מכן הייתה ירידה חדה מה שמראה שהנתונים נשלחו בפרץ אחד ולא באופן רציף.
3. בהשוואה לאתרים אחרים, התנהגות זו מתאימה לאתרי מידע כמו Wikipedia, שבהם דף נטען ואז אין הרבה תנועה לאחר מכן.

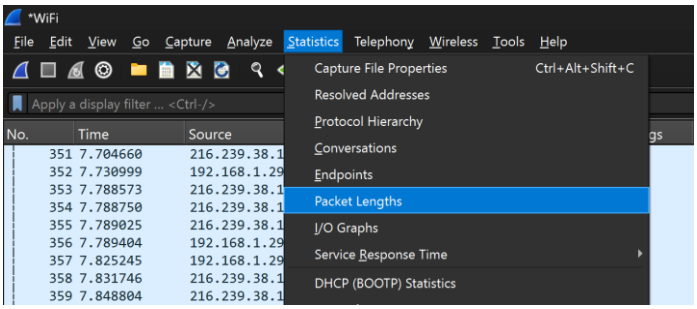
מסקנה :גם אם אין Flow ID, ניתן להסיק שמדובר בגלישה לאתר שמבצע טעינה ראשונית כבדה ואינו ממשיך לזרום כמו וידאו.

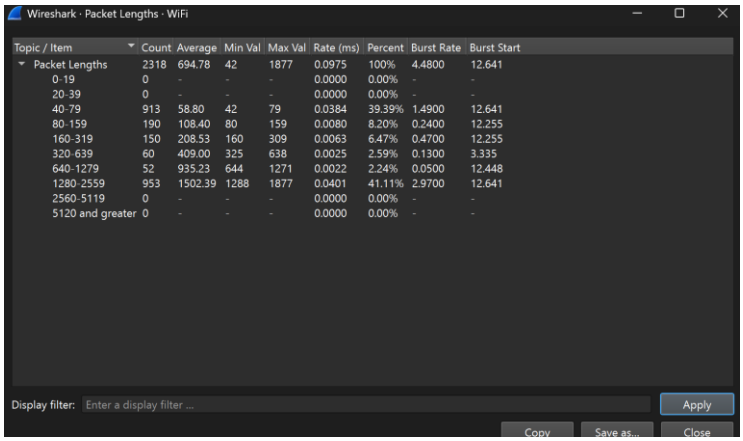
לסיכום , ניתן לראות את התנהגות זרימת הנתונים.
שינויים חדים יכולים להצביע על טעינה של דפים גדולים.
תעבורה עקבית עם מעט מרווחים מצביעה על וידאו או סטרימינג.

זה עוזר לתוקף בכך שאם יש קפיצות גדולות בתעבורה ,ייתכן שהמשתמש נכנס לאתר עם תוכן כבד.

- שלב 3: זיהוי תבניות חזרתיות

נבדוק אם ניתן לזהות דפוסי שימוש חוזרים בתעבורה של המשתמש, גם אם היא מוצפנת.





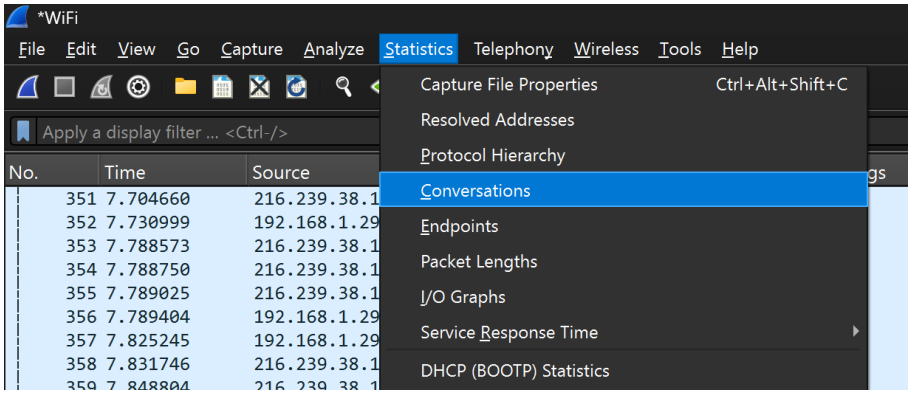
גם ללא כתובות IP ניתן לזהות חזרות של פעילות בתעבורה לפי נפח וזמן החבילות.
התוקף יכול להסיק שהמשתמש חזר לאותו אתר מספר פעמים על בסיס קפיצות בנפח הנתונים.

לסיכום , תוקף יכול לזהות דפוסי שימוש חוזרים אצל המשתמש.
אם משתמש פותח אתר עם עמוד כבד ,זה יכול להופיע כקפיצה ברורה בנפח התעבורה.

זה עוזר לתוקף בכך שאם התוקף מזהה חזרות של דפוסי תעבורה דומים ,ניתן להסיק שהמשתמש חוזר שוב ושוב לאותו שירות.

- שלב 4: השוואת התעבורה למאגרים קיימים

תוקפים יכולים להשוות דפוסי תעבורה למאגרי מידע ידועים של אתרי אינטרנט ולנחש באיזה אתר המשתמש ביקר.



לא נתעסק בכתובות IP, אלא רק בנפח התעבורה והחבילות.

אם יש שורות עם מספר גבוה של חבילות (נפח נתונים משמעותי), זה מצביע על פעילות מול שירות אינטרנט.
בהתאם לגודל החבילות ולמשך הזמן של הזרם, ניתן לשער איזה סוג שירות זה (למשל, ויקיפדיה – נפח קטן, יוטיוב – נפח גדול).

לסיכום תוקף יכול להשוות את התבנית של זמני ההגעה והגודל למאגרים ידועים של דפוסי תעבורה(Website Fingerprinting) .
ניתן להשתמש בשיטות Machine Learning כדי לזהות אילו אתרים נצפו בהתבסס על גודל ומבנה התעבורה בלבד.

זה עוזר לתוקף בכף שדפוסים מסוימים בתעבורה תואמים לאתרים ידועים. לדוגמה:

- ויקיפדיה זה מעט חבילות קטנות עם קפיצה ראשונית גדולה.
- YouTube זה זרם קבוע של חבילות גדולות.
  כלומר תוקף יכול להצליב את דפוסי החבילות מול מאגרי מידע ידועים של תעבורת אתרים ולנחש באיזה אתר המשתמש ביקר.

מהממצאים שמצאנו ניתן לראות כי התוקף לא יכול לדעת בוודאות באיזה אתר המשתמש היה, כלומר אי אפשר לדעת במאה אחוז, אבל הוא יכול לנחש ברמת דיוק גבוהה על סמך דפוסי תעבורה. אם דפוס תעבורה מסוים תואם למאגרים של אתרים ידועים, התוקף יכול לשער שהמשתמש ביקר באתר מסוים. אם אין מידע השוואתי כלומר, אין מאגרי, יהיה לתוקף קשה לזהות את האתר בדיוק, אבל הוא עדיין יכול לזהות סוג פעילות (סטרימינג, קריאה, חיפוש וכו)

איך אפשר למנוע מתקפה כזו?

כדי להקשות על תוקף לזהות אתרים על בסיס ניתוח גודל וזמן חבילות, ניתן להשתמש בפתרונות הבאים:

VPN(רשת פרטית וירטואלית):גורמת לכך שכל התעבורה יוצאת דרך שרת אחד, מה שמקשה על זיהוי הדפוסים האישיים של המשתמש.

Tor (The Onion Router) מנתב את התעבורה דרך מספר שרתים ומוסיף עיכובים רנדומליים, מה שמקשה על זיהוי דפוסי תעבורה.

Padding (ריפוד חבילות) :חלק מהאתרים ושירותי VPN מוסיפים "ריפוד" לחבילות כדי ליצור גודל אחיד ולמנוע זיהוי מבוסס גודל .

Traffic Obfuscation (ערפול תעבורה) ישנם כלים שמבצעים שינוי רנדומלי בגודל ובתזמון של חבילות כדי להקשות על ניתוח סטטיסטי.

מתרחיש זה קיבלנו כי אפשר לזהות אתרים מוצפנים באופן חלקי אפשרי גם ללא כתובות IP באמצעות ניתוח גודל החבילות, זמני ההגעה ודפוסי תעבורה. כלומר מדובר בניחוש מושכל ולא בזיהוי ודאי התוקף לא רואה את כתובת האתר, אך יכול להסיק באיזה סוג אתר המשתמש ביקר על בסיס דפוסי שימוש ידועים.

1. ניתוח דפוסים מאפשר לזהות שירותים על ידי ניחוש מושכל כמו ויקיפדיה, יוטיוב, חדשות או סטרימינג גם אם התוכן עצמו מוצפן לחלוטין.
2. השוואת הנתונים למאגרים של Fingerprinting משפרת את הדיוק –אך אם אין מידע השוואתי, קשה יותר לזהות את האתר המדויק.
3. הגנה על פרטיות דורשת טכניקות מיוחדות –כמו VPN, Tor, ריפוד חבילות (Padding) או Traffic Obfuscationכדי לטשטש דפוסים.

מקרה שני : גלישה באתר אינטרנט באמצעות דפדפן Microsoft Edge

בחרנו לעשות על אתר מאובטח של GitHub

ראשית

בכדי לסנן את תעבורת הרשת לאתר github נכנסתי לטרמינל והכנסתי את הפקודה

dig github.com (**Domain Information Groper**)

פקודה זו מחפשת את כל הכתובות ה-DNS של האתר github ומספק את כל כתובות הip

של האתר.

```
Last login: Sun Mar  9 12:53:38 on ttys000
omri@Omri-M2-MacBook ~ % dig github.com


; <<>> DiG 9.10.6 <<>> github.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10910
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL:

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;github.com.                    IN      A

;; ANSWER SECTION:
github.com.             60      IN      A       20.217.135.5

;; Query time: 69 msec
;; SERVER: 172.16.0.232#53(172.16.0.232)
;; WHEN: Sun Mar 09 12:55:42 IST 2025
;; MSG SIZE  rcvd: 55

omri@Omri-M2-MacBook ~ %
```

קיבלנו כתובת ip אחת של האתר github

הכתובת - 20.217.135.5

פקודה זו אפשרה לנו לסנן ב-Wireshark את הפקאטות הרלוונטיות של github

ושרק פקטות אלו יהיו בסטטיסטיקה שלנו

נסנן את ההקלטה ונשמור את ההקלטה כקובץ csv שעליו ננתח את כל הסטטיסטיקות

מה-Wireshark

את ההקלטה נשמור בשם -GitHub.csv

(ip.addr == 20.217.135.5) and (tcp.port == 443)

No.	Time	Source	Destination

ip.addr == 20.217.135.5

מסנן את החבילות רשת שבהן כתובת ה- ip מופיע כמקור או כיעד

כמו שאמרנו לעיל כתובת זו שייכת לאתר github.com ולכן אלו החבילות שיתקבלו על ידי הסינון הזה

tcp.port==443 סינון זה מציג רק חבילות TCP שעוברות דרך פורט 443 שזה הפורט שמשמש לחיבורי HTTPS (SSL/TLS)

כמו כן הסינון מוודא שהתוצאות יכילו רק תעבורה בין המחשב שלי לאתר github.com ללא רעשי רקע מפרוטוקולים אחרים וכתובות ip שונות

הסינון יכיל חבילות כמו הקמת חיבור , שליחה וקבלת בקשות כמו טעינת דף ועוד..

כעת נריץ את הקובץ פייתון על GitHub.csv אשר מכיל את המידע הרלוונטי מההקלטה

```
import pandas as pd
import matplotlib.pyplot as plt

# 1. Load the CSV
df = pd.read_csv('GitHub.csv')

# 2. Convert Time to numeric if needed
df['Time'] = pd.to_numeric(df['Time'], errors='coerce')
df = df.sort_values('Time').reset_index(drop=True)
```

גרף ראשון: שדות כותרת IP

בגרף זה אני משווים בין מספר הפעמים שכל כתובת IP מופיע ככתובת מקור לעומת כתובת יעד

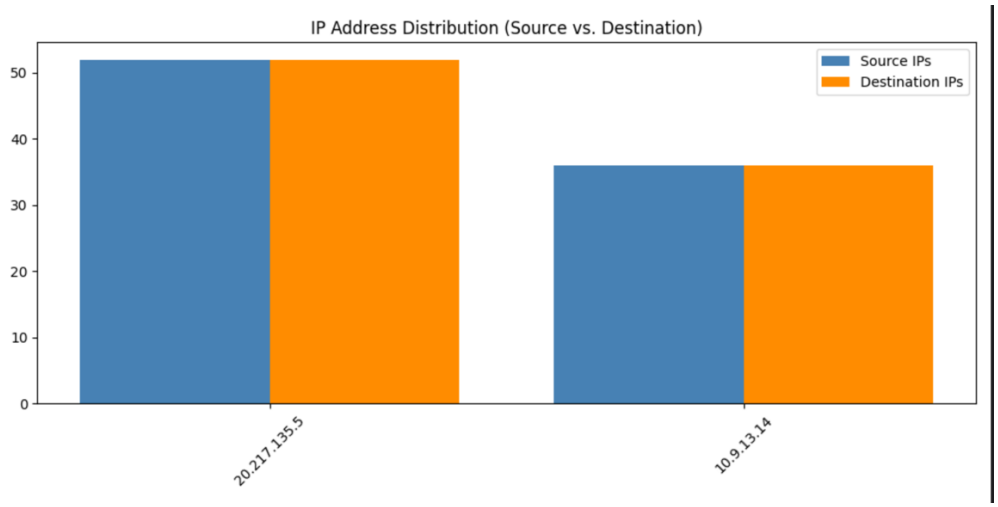
כתובת מקור - 10.9.13.14 (המחשב שלי)

כתובת יעד - 20.217.135.5 (github.com)

עבור כל כתובת מוצגות שתי עמודות אחת

עמודה כחולה - מייצגת את מספר הפעמים שהיא מופיעה כמקור

ועמודה כתומה - מייצגת את מספר הפעמים שהיא מופיעה כיעד



התפלגות כמעט שווה מצביעה על איזון בפעילות הנמדדת, כלומר אין חריגות גדולות שמצביעות על תופעות מיוחדות או בעיות ברשת.

גרף שני: שדות כותרת TCP

הגרף מציג את סוגי הדגלים בפרוטוקול TCP כלומר את סוגי החבילות שנשלחות

כל דגל מיוצג כעמודה עם גובהה המציין את מספר ההופעות שלו (בהקלטה)

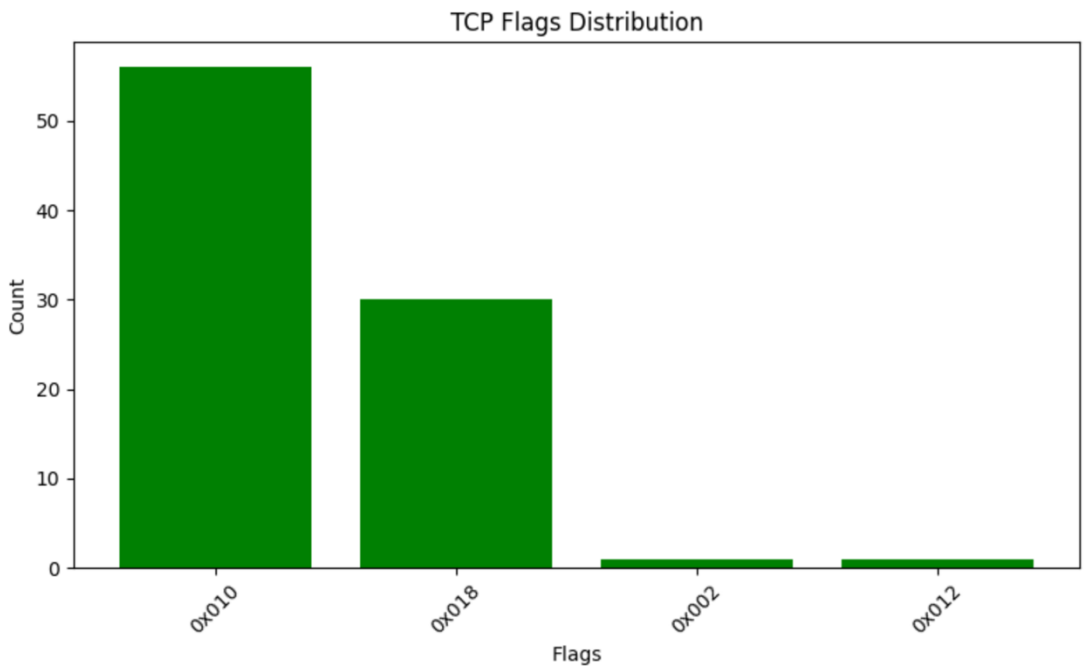
0x002 (SYN) – מסמן תחילת חיבור

0x012 (SYN+ACK) - מסמן תשובה לבקשת חיבור

0x010 (ACK) - משמש לאישור קבלת נתונים

0x018 (PSH+ACK) - שולח נתונים (push) יחד עם (ACK)

*היעדר דגל FIN – סגרתי את ההקלטה לפני שסגרתי את האתר



הסבר על הגרף:

אטרקציית חיבור:

ערכים כמו 0x002 ו- 0x012 מצביע על שלבי תחילת חיבור (handshake)

מעיד על כך שיש תעבורה של פתיחת חיבורים

אישור והעברת נתונים:

ערך הACK 0x010 מופיע הרבה מה שמצביע על כך שברוב המקרים מתקיימת העברת נתונים שבה יש אישור על קבלת חבילות

0x018 מראה שיש חבילות שבהן מועברים נתונים בפועל עם אישור (ACK+push

גרף שלישי: שדות כותרות TLS

גרף זה מתמקד בשדות TLS Server Name שזה בעצם השדה שמכיל את שם השרת שאליו מתבצעת ההתחברות המוצפנת

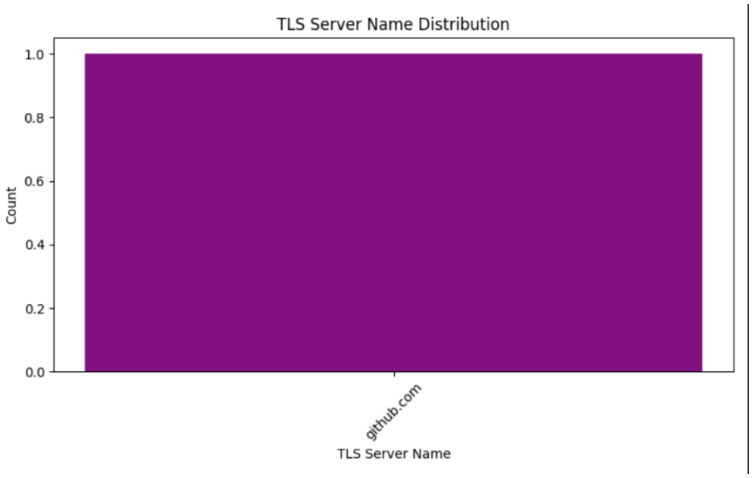
הגרף מציג עמודות כאשר כל עמודה מייצגת שם שרת שצויין בשלב ה-SNI

(Server Name Indication)

ציר ה-x מציג את שם השרת שנמצא בתקשורת מוצפנת – github.com

זה ה- SNI (Server Name Indication)

ציר ה-y מציג את מספר הפעמים ש-שם השרת הופיע בתעבורת הרשת (פעם אחת לפי גרף זה)



גרף רביעי: גודלי מנות

הגרף המציג את התפלגות גדלי החבילות בתעבורת הרשת שנלכדה בהקלטה,

ציר ה-X - מייצג את הגודל החבילה בבייטים (Packet Length) כלומר גודל של כל חבילה שנשלחה או התקבלה

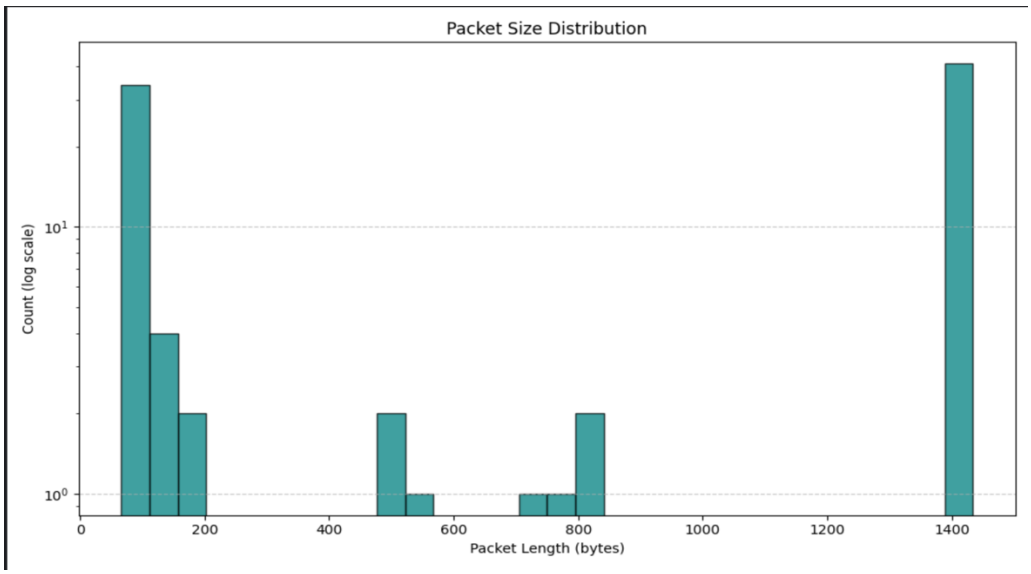
ציר ה-Y – מייצג את מספר החבילות שנפלו בטווחי הגודל השונים

באמצעות הגרף הזה ניתן להבין כיצד מתפלגות החבילות בתקשורת

לדוגמה אם רובן

קטנות - אלו פקטות בקרה

גדולות - נתונים בקובץ או תמונות (או שילוב של השניים)



הסבר על הגרף:

מהגרף ניתן להבחין בכמה תופעות מרכזיות לפי גודלי החבילות

1. חבילות קטנות מאוד (0-100 בייטים) – כוללת בעיקר את חבילות בקרה של TCP כמו למשל
ACK – אישור קבלת נתונים
SYN/SYN+ACK – הקמת חיבור TCP
FIN – סיום חיבור

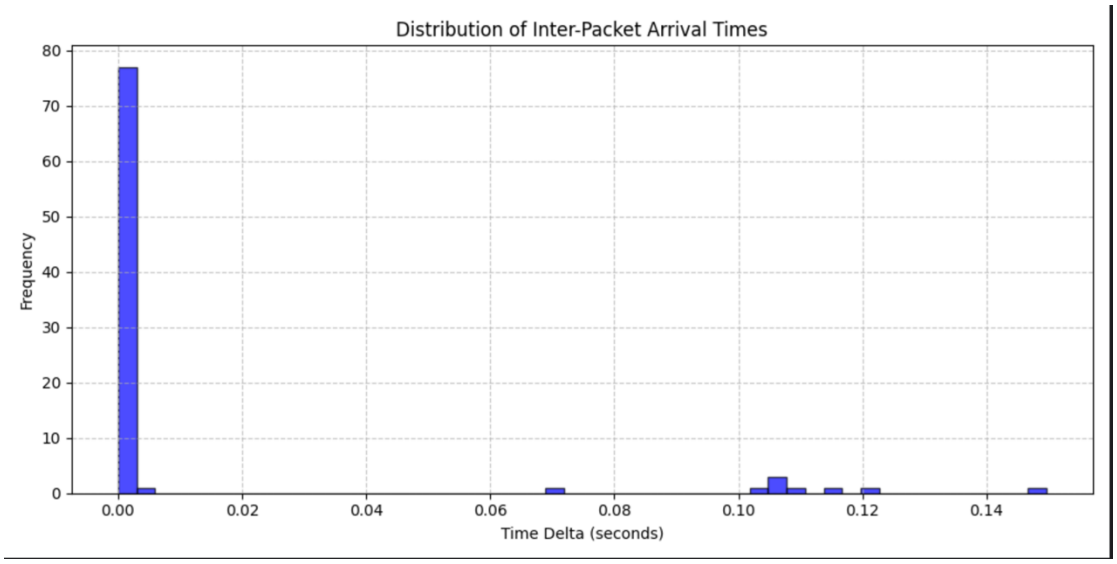
2. חבילות גדולות (1400-1500 בייטים) – הן החבילות שמכילות נתונים בפועל בפרוטוקולי רשת כמו Ethernet גודל החבילה המקסימלי הוא 1500 בייטים (MTU) ולכן רוב הנתונים מחולקים לחבלות בגודל זה לניצול מקסימלי של הרשת (חלוקה זו מבטיחה שהנתונים יועברו ביעילות)

בשרתים כמו GitHub מרבית הנתונים המועברים הם קוד, וטקסטים , וקבצים מה שמסביר את החבילות הגדולות

3. מעט חבילות בינוניות (500-1000 בייטים) – הסיבה לכך שיש מעט חבילות בינוניות היא ש- HTTP\2 מפצל את הנתונים מחבילות גדולות לחבילות קטנות יותר (חלוקת עומס בין מספר חיבורים)
בנוסף מבנה ב- HTTP\2 הכותרות של הבקשות והתשובות נדחסות דוג׳
אם בקשה או תשובה מסויימת כוללת כמות בינונית של נתונים היא תשלח את החבילה בגודל הבינוני (500-1000)
גרף חמישי: זמן הגעה בין מנות

בגרף זה מוצגת התפלגות של זמני ההגעה בין חבילות

ציר ה-X – מייצג את הפרש הזמן בין חבילה אחת לשניה (במילי שניות או שניות)
ציר ה-Y – מייצג את מספר הפעמים שהתקבל פרק זמן מסויים בין חבילה לחבילה (כלומר כמה חבילות חוו השהייה מסויימת)



הסבר על הגרף:

רוב החבילות מתקבלות בהפרשי זמן קטנים מאוד קרובים ל (0ms) – המשמעות היא שרוב החבילות מתקבלות ברצף מהיר כמעט ללא השהייה דבר זה מצביע על **תקשורת יציבה**, שבה השרת שולח חבילות באופן רציף והלקוח מקבל אותם ללא המתנה.

בגרף ישנה קבוצה קטנה של חבילות עם **הפרשים גדולים** – המשמעות היא שישנם רגעים שבהם היו עיכובים בין שליחת החבילה אחת לשנייה ישנם מספר סיבות כגון

1. השהיה בשרת (Server Delay) – אם השרת מעבד בקשה זמן רב יותר מהרגיל זה יגרום להשהיה עד לשליחת החבילה הבאה
2. עומס רשת (Network Congestion) – אם הרשת עמוס חלק מהחבילות יתעכבו
3. וויסות TCP (Flow Control) - אם מקבל החבילות מבצע ויסות ומאט את קצב קבלת החבילות או שהשרת או הלקוח מבצע שליחה חוזרת (Restransmission)
4. בקשות חדשות או פתיחת חיבורים הם גם כן גורם להשהיות גבוהות יותר (בקשות HTTP חדשות להקמת חיבור)

גרף שישי: גודל זרימה

בניתוח תעבורת רשת, זרימה (Flow) מוגדרת כסדרה של חבילות שנשלחות **בין מקור ויעד מסוימים** תוך שימוש בפרוטוקול תקשורת מסוים (TCP , TLS)

גרף זה מציג **כמה חבילות שייכות לכל זרימה** כלומר, כמה חבילות עברו בין מכשירים מסוימים לאורך החיבור

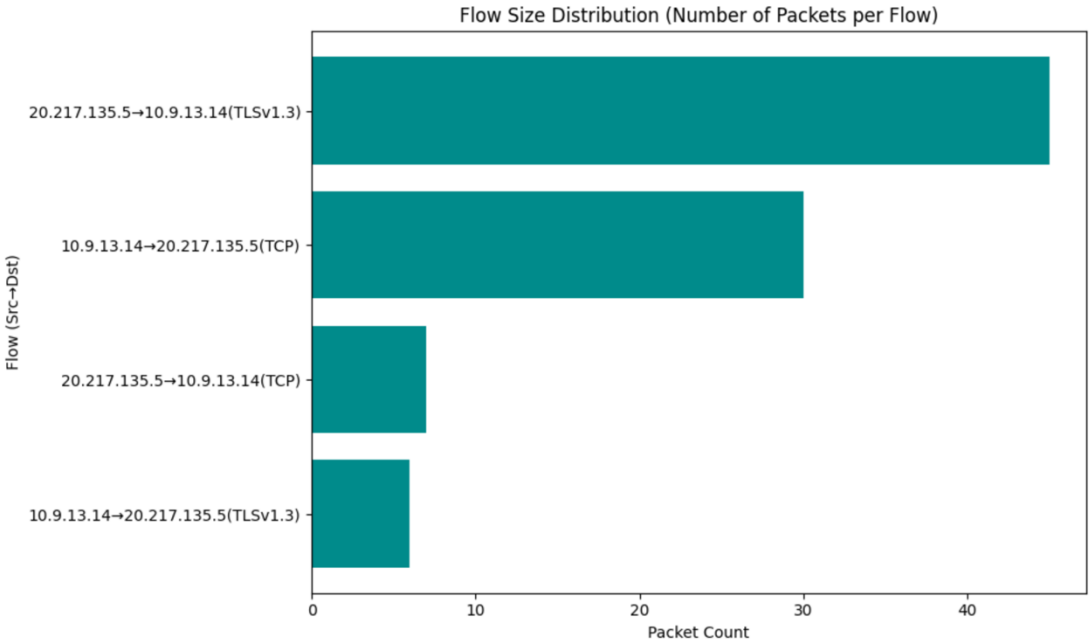
גרף זה עוזר לנו להבין את התפלגות הזרימות ומה שמסייע לנו להבין או לזהות

דפוסי תקשורת כמו שיחות ארוכות או אינטרקציות קצרות.

ציר ה-X – מייצג את מספר החבילות בכל זרימה

ציר ה-Y – הזרימות ברשת

ככל שהבר ארוך יותר, כך הזרימה מכילה יותר חבילות ומייצגת תקשורת משמעותית יותר בין המחשב שלי לגיthub (יעד ← מקור)



הזרימה הגדולה ביותר (מעל 40 פאקטות) היא תקשורת בין github.com לכתובת המחשב שלי בפרוטוקול TLSv1.3

המשמעות היא שמדובר בגלישה באתר שמאובטח ב- HTTPS , שבו יש הרבה חבילות עקב ההצפנה

TLSv1.3 כולל תהליך Handshake מוצפן ולכן הזרימה מכילה גם כן שלבי פתיחת חיבור, אישור הצפנה ועוד..

יתכן שגם זו בקשת HTTP גדולה כמו טעינת עמוד.

זרימות TCP נפרדות עם פחות חבילות

הזרימה בין המחשב שלי לאתר github מכילה 30 חבילות ואילו הזרימה בכיוון ההפוך מכיל מספר קטן משמעותית של חבילות

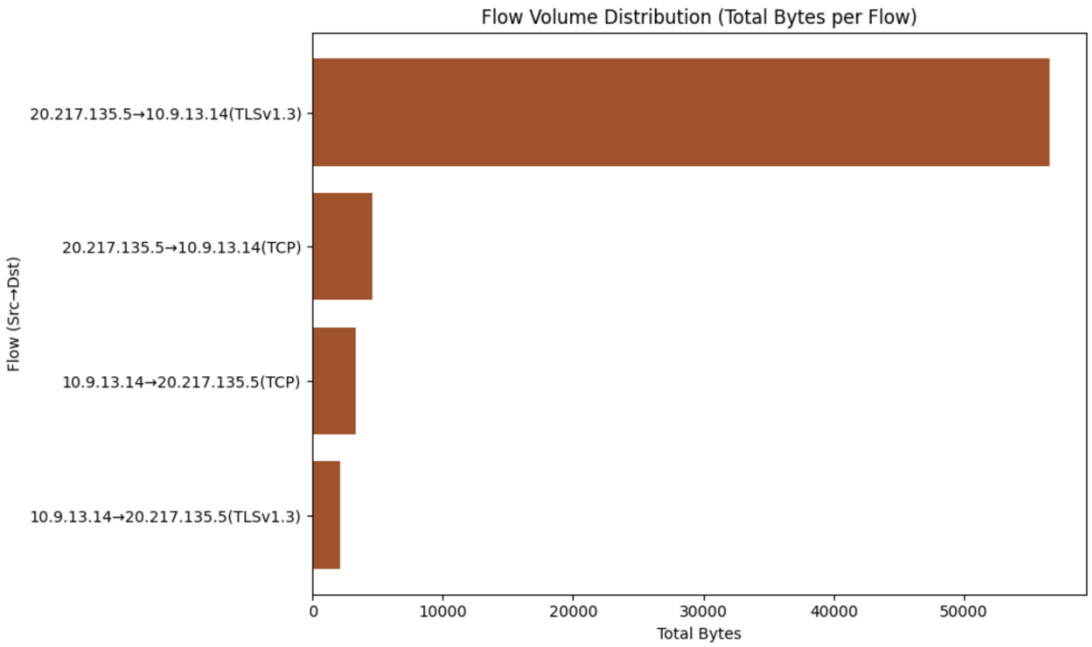
זרימה קצרה של TLSv1.3 מהמחשב שלי לגithub של פחות מ10 חבילות שיכולה להיות לדוגמה, מבדיקת סטטוס , סגירת חיבורים , בדיקות חיבור.

גרף שביעי: נפח זרימה

גרף זה מציג את כמות הנתונים הכוללת (Total Bytes) שהועברה בכל הזרימה בין המחשב שלי לבין github בחלוקה לפי הפרוטוקול שבו נעשה שימוש TLS,TCP

ציר ה-X – מספר הבייטים שהועברו בכל זרימה

ציר ה-Y – הזרימות , מסודרות לפי המקור, היעד והפרוטוקול.



הסבר על הגרף:

הזרימה שבה הועבר הכי הרבה מידע היא מהשרת github למחשב שלי באמצעות TLSv1.3

זה אומר ש- github שלח כמות גדולה של נתונים למחשב שלי כמו קבצי HTML וCSS ועוד..

זה שהנתונים הועברו דרך TLSv1.3 מצביע על כך שהכל הוצפן.

זו הזרימה הדומיננטית כי בדרך כלל השרת שולח יותר מידע ללקוח מאשר הלקוח לשרת

יש גם זרימה נוספת בטווח בינוני, שבה github שלח למחשב שלי נתונים ב-TCP ישיר ללא TLS זה אומר שזו היית תקשורת ראשונית (כמו פתיחת חיבור) , הודעת ניתוב או חבילת ניהול חיבור

כמות החבילות כאן קטנה יותר מה שמראה שהתקשרת המהותית היית מוצפנת ולא ישירה ב TCP בלבד

כמות הנתונים מהמחשב שלי ל github קטנה בהרבה זה אומר שאני (המחשב שלי) שלח מעט נתונים לgithub (דפוס תקשורת קלאסי בין לקוח לשרת)

חלק 4:

נדרש מאיתנו להקליט שוב את תעבורת הרשת שלנו, בהנחה שאנחנו התוקף ומנסים ללמוד אילו אפליקציות או אתרים המשתמש ניגש אליהם

המטרה היא לנתח עד כמה ניתן לזהות אתרים ואפליקציות שהמשתמש ניגש אליהם גם כאשר התעבורה מוצפנת

אם כן נבדוק את שני התרחישים הבאים:

1. התקפה עם Flow ID – כאשר התוקף יודע את גודלי החבילות, חותמות הזמן, ואת מזהה הזרם (Flow ID) כלומר כתובות המקור והיעד , פורטי המקור והיעד, והפרוטוקול
2. התקפה ללא Flow ID – כאשר התוקף אינו יודע את מזהה הזרם כלומר רק את גודלי החבילות וחותמות הזמן.

עשיתי הקלטה חדשה נכנסתי לgithub.com , ונכנסתי לאחד הקישורים הפנימיים באתר.

שם ההקלטה: part4_github.pcapng

הארה: הקלטה זו תשמש לשני המקרים אך הניתוחים יתבצעו באופן שונה לכל אחד מהמקרים

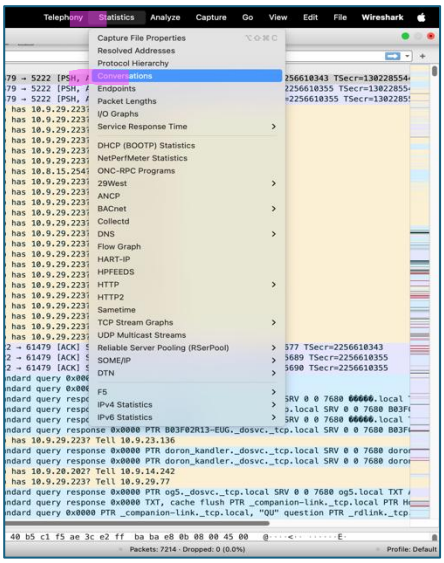
מקרה ראשון:

כאמור לעיל המטרה היא להראות שהתוקף יכול לזהות עם אילו שרתים היה למשתמש אינטרקציה (תקשורת) איתם , ואיזה פרוטוקולים היו בשימוש, **עם Flow ID**

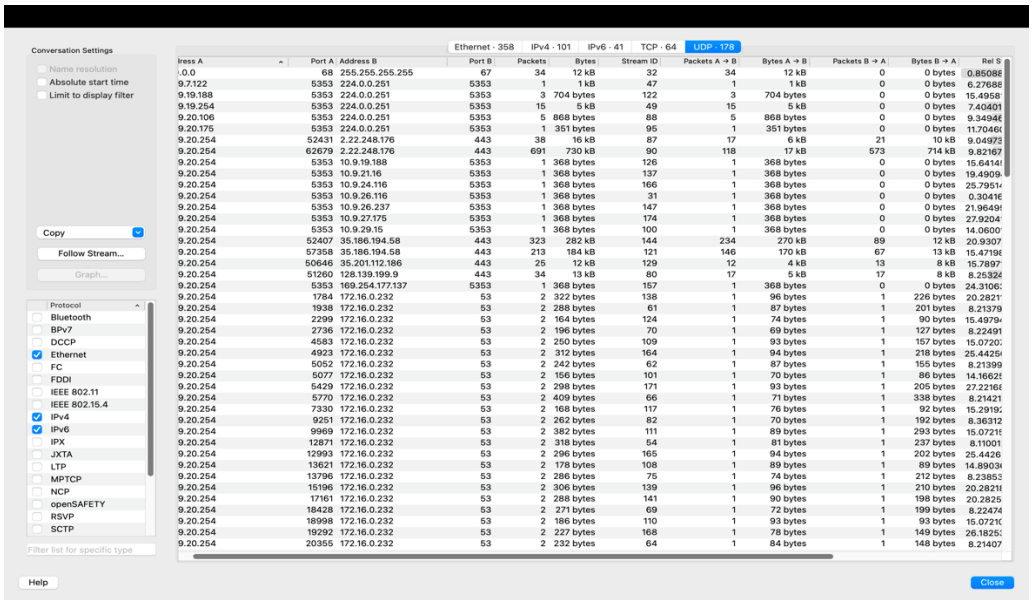
נראה זאת באמצעות ניתוח שיראה כיצד זרמים שונים של תעבורה עוזרים בזיהוי אתרים ואפליקציות

שלב ראשון: **נזהה את כל הזרמים** שהתבצעו כולל כתובות IP ופרוטוקולים

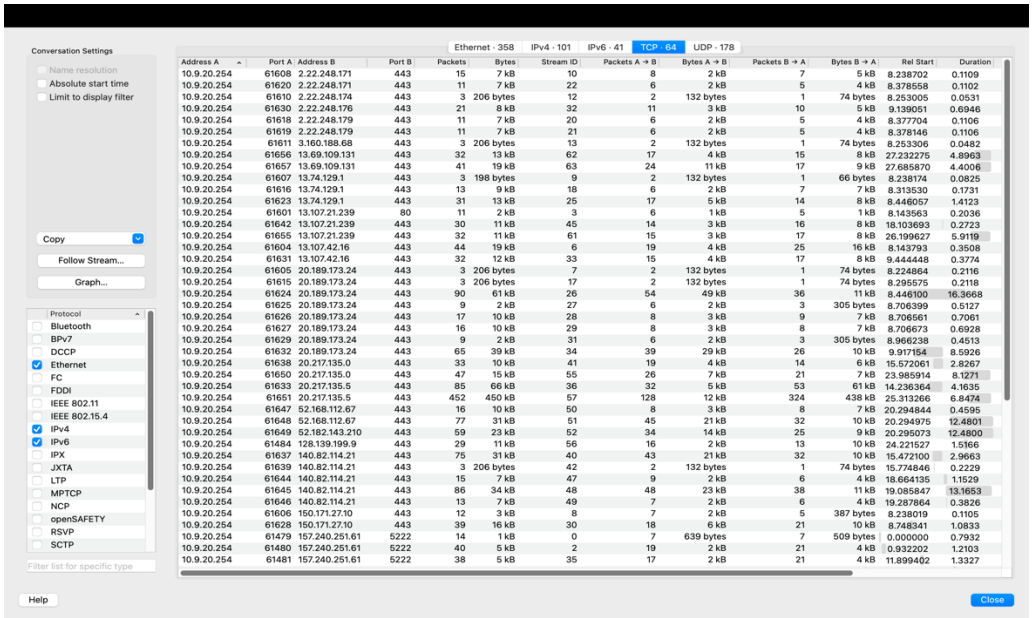
באופן הבא:



תוצאות עבור UDP:



תוצאות עבור TCP:



מהסינון החבילות לפי UDP וTCP בלבד ניתן לראות שגם כאשר התוכן מוצפן (HTTPS\TLS)

עדיין אפשר לראות את כתובות ה-IP,מספרי הפורטים והפרוטוקולים.

מידע זה מאפשר לזהות לאן המשתמש מתחבר גם אם לא ניתן לקרוא את התוכן בפועל.

סינון זה תחילה נותן אפשרות לנתח את התעבורת רשת בצורה יותר יעילה "כתוקף"

זה מאפשר לזהות במהירות לאן המשתמש מתחבר (כתובות יעד) באילו פורטים השתמש (443,80 ...) , וכמה תעבורה עברה בכל חיבור

שלב שני: סינון לפי Flow ID

ניתן לראות שעל ידי פקודת nslookup לאתר github.com

ניתן לקבל את הכתובת 20.217.135.5

```
omri@Omri-M2-MacBook ~ % nslookup github.com
Server:         172.16.0.232
Address:        172.16.0.232#53

Non-authoritative answer:
Name:   github.com
Address: 20.217.135.5
```

עכשיו נוכל לנצל את המידע הזה בשביל לגלות עוד מידע.

נרצה לאתר חיבורים לאתרים מאובטחים המתבצעים באמצעות פורט 443

(HTTPS)

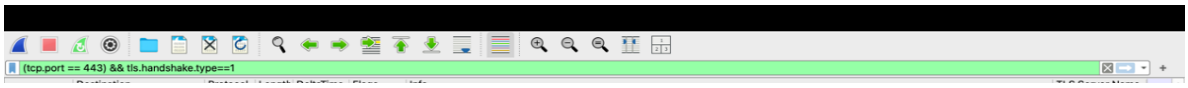
שכן github משתמש בתקשורת HTTPS שמבוססת על פרוטוקול TCP ושבדרך כלל התקשורת נעשית דרך פורט 443

10.9.20.254	61650	20.217.135.0	443	47	15 kB	55	26	7 kB	21	7 kB	23.985914	8.1271
10.9.20.254	61638	20.217.135.0	443	33	10 kB	41	19	4 kB	14	6 kB	15.572061	2.8267
10.9.20.254	61651	20.217.135.5	443	452	450 kB	57	128	12 kB	324	438 kB	25.313266	6.8474
10.9.20.254	61633	20.217.135.5	443	85	66 kB	36	32	5 kB	53	61 kB	14.236364	4.1635

אזי נסנן את התעבורה לפי פורט 443

ונוסיף סינון נוסף בשם tls.handshake.type == 1

שהוא מסנן רק את חבילות ה- (TLS) מסוג hello Client שכן למרות שהתוכן לאחר מכן נשלח מוצפן בזמן זה שדה ה SNI נשלח בגילוי כלומר אפשר לראות לאיפה המשתמש רוצה להתחבר.



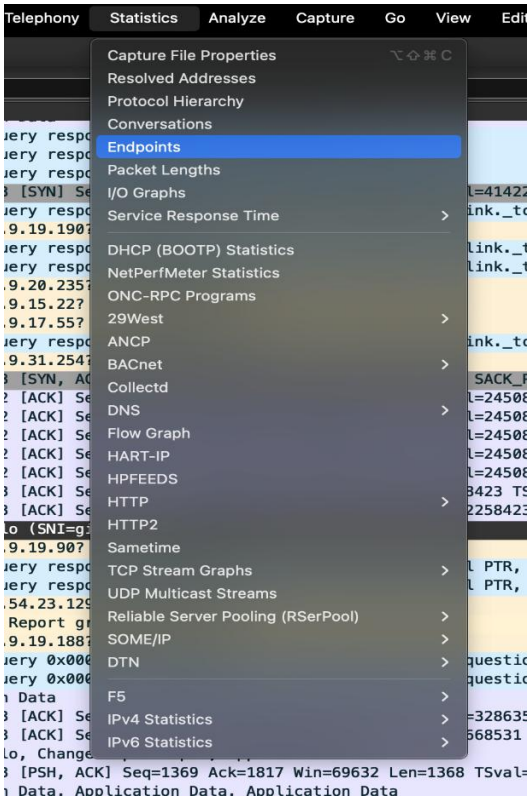
מסקנה: משני סינונים אלו נוכל להגיע למסקנה שהמשתמש יצר אינטרקציה עם האתר

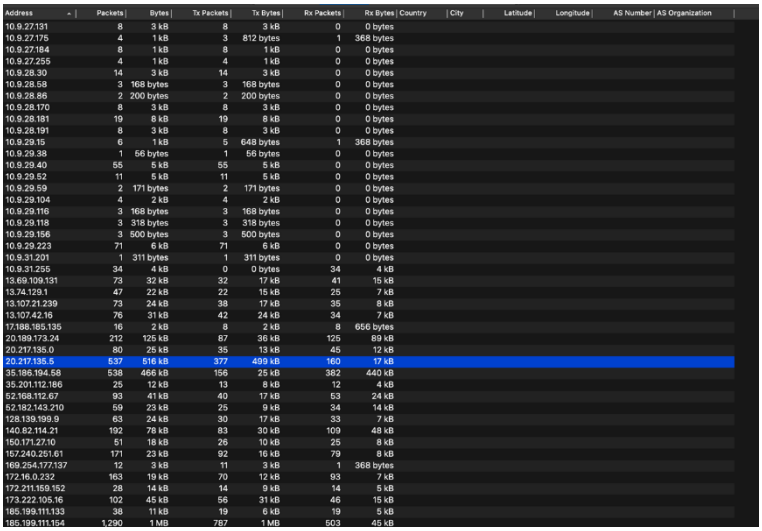
github.com

3. ניתוח נפח הנתונים שנשלח לכל שרת: נראה שהתוקף יכול להעריך איזה אפליקציות/אתרים המשתמש ביקר בהם ולפי נפח הנתונים נוכל להסיק מסקנות של פעולות מסיימות שעשה בהם

ניגש לendpoint

טבלה זו תביא לנו את כל כתובות ה- IP המעורבות בתעבורה עם כמות החבילות שנשלחו וגודל נתוני התעבורה ששודרו מכל endpoint





Address	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes	Country	City	Latitude	Longitude	AS Number	AS Organization
10.9.27.131	8	3 kB	8	3 kB	0	0 bytes						
10.9.27.175	4	1 kB	3	812 bytes	1	368 bytes						
10.9.27.184	8	1 kB	8	1 kB	0	0 bytes						
10.9.27.255	4	1 kB	4	1 kB	0	0 bytes						
10.9.28.30	14	3 kB	14	3 kB	0	0 bytes						
10.9.28.58	3	168 bytes	3	168 bytes	0	0 bytes						
10.9.28.86	2	200 bytes	2	200 bytes	0	0 bytes						
10.9.28.170	8	3 kB	8	3 kB	0	0 bytes						
10.9.28.181	19	8 kB	19	8 kB	0	0 bytes						
10.9.28.191	8	3 kB	8	3 kB	0	0 bytes						
10.9.29.15	6	1 kB	5	648 bytes	1	368 bytes						
10.9.29.38	1	56 bytes	1	56 bytes	0	0 bytes						
10.9.29.40	55	5 kB	55	5 kB	0	0 bytes						
10.9.29.52	11	5 kB	11	5 kB	0	0 bytes						
10.9.29.59	2	171 bytes	2	171 bytes	0	0 bytes						
10.9.29.104	4	2 kB	4	2 kB	0	0 bytes						
10.9.29.116	3	168 bytes	3	168 bytes	0	0 bytes						
10.9.29.118	3	318 bytes	3	318 bytes	0	0 bytes						
10.9.29.156	3	500 bytes	3	500 bytes	0	0 bytes						
10.9.29.223	71	6 kB	71	6 kB	0	0 bytes						
10.9.31.201	1	311 bytes	1	311 bytes	0	0 bytes						
10.9.31.255	34	4 kB	0	0 bytes	34	4 kB						
13.69.109.131	73	32 kB	32	17 kB	41	15 kB						
13.74.129.1	47	22 kB	22	15 kB	25	7 kB						
13.107.21.239	73	24 kB	38	17 kB	35	8 kB						
13.107.42.16	76	31 kB	42	24 kB	34	7 kB						
17.188.185.135	16	2 kB	8	2 kB	8	656 bytes						
20.189.173.24	212	125 kB	87	36 kB	125	89 kB						
20.217.135.0	80	25 kB	35	13 kB	45	12 kB						
20.217.135.5	537	516 kB	377	499 kB	160	17 kB						
35.186.194.58	538	466 kB	156	25 kB	382	440 kB						
35.201.112.186	25	12 kB	13	8 kB	12	4 kB						
52.168.112.67	93	41 kB	40	17 kB	53	24 kB						
52.182.143.210	59	23 kB	25	9 kB	34	14 kB						
128.139.199.9	63	24 kB	30	17 kB	33	7 kB						
140.82.114.21	192	78 kB	83	30 kB	109	48 kB						
150.171.27.10	51	18 kB	26	10 kB	25	8 kB						
167.240.251.61	171	23 kB	92	16 kB	79	8 kB						
169.254.177.137	12	3 kB	11	3 kB	1	368 bytes						
172.16.0.232	163	19 kB	70	12 kB	93	7 kB						
172.211.159.152	28	14 kB	14	9 kB	14	5 kB						
173.222.105.16	102	45 kB	56	31 kB	46	15 kB						
185.199.111.133	38	11 kB	19	6 kB	19	5 kB						
185.199.111.154	1,290	1 MB	787	1 MB	503	45 kB						

תזכורת כתובת 20.217.135.5 – github

הנתונים מראים שישנם 537 חבילות והגודל הכולל הוא 516kb של תעבורה כלומר ישנו נפח דיי גדול של נתונים מה שמעיד שלא רק נוצר חיבור אלא גם הועברו נתוני מידע ממשיים

מסקנה: התוקף יכול לדעת בוודאות שהמשתמש ביקר ב - github

כתובת הרשת המקומית של המחשב שלי -10.9.20.254

שכן ניתן לראות בנוסף שגודל התעבורה של המשתמש מה שנותן **אינדיקציה לתוקף** לדעת איזה סוג של פעילות המשתמש מבצע במחשב.

ניתן לראות שגודל התעבורה עומד על 4mb

10.9.20.254	5,025	4 MB	2,189	862 kB	2,836	3 MB

לסיכום התוקף יכול לראות את סוגי התעבורה של שהמשתמש מבצע , בנוסף הוא יכול לשער את פעולות שהמשתמש מבצע ברשת (סטרימינג,הורדה,גלישה רגילה) על פי גודלי החבילות והip-ים שהוא מתקשר.

סיכום מקרה ראשון עם Flow ID:

גם כאשר התעבורה מוצפנת, התוקף יכול עם ה-flow id לקבוע עם אילו שרתים נוצרה אינטרקציה, באמצעות סינון לפי פורט 443 וחיפוש הודעות tls- client hello המכילות את שדה ה – (server name indicetion)

ניתן לזהות במדוייק שהמשתמש התחבר לאתר כמו github ולהעריך את נפח הנתונים שהועברו. כלומר , גם ללא קריאת התוכן המוצפן אפשר להסיק מסקנות לגבי הפעילו

כיצד ניתן למנוע מתקפה כזו?

ניתן למנוע מתקפה כזו, שבה תוקף מנתח את מטא-נתוני התעבורה (כמו כתובות IP, פורטים, גודלי חבילות וזמני העברה), באמצעות מספר אמצעים:

שימוש ב-VPN

כאשר כל התעבורה עוברת דרך שרת ביניים, כתובת ה-IP של היעד והפרטים המקוריים מוסתרים מהתוקף.

שדרוג ל-TLS 1.3 עם הצפנת Client Hello Encrypted SNI/ECH

כך ניתן להסתיר גם את שם השרת (SNI) שנשלח בהתחלת ההתקשרות, מה שמקטין את המידע החשוף.

הוספת Padding והסוואת תעבורה:

שימוש בטכניקות שמוסיפות מידע אקראי (Padding) או משנה דינמית את גודל החבילות והתזמון שלהן, מקשה על ניתוח סטטיסטי של התעבורה.

שימוש ב-DNS מוצפן (DNS over TLS/HTTPS)

מונע חשיפת בקשות DNS, שיכולות לגלות לאילו אתרים המשתמש מנסה להתחבר.

באמצעים אלו, ניתן להקטין משמעותית את המידע החשוף למתקפה מבוססת ניתוח מטא-נתונים, וכך להגן על פרטיות המשתמש ועל דפוסי הגלישה שלו.

מקרה שני – כאשר התוקף יודע רק את גודל החבילות וחותמות הזמן.

שלב ראשון: ניתוח גודל החבילות.

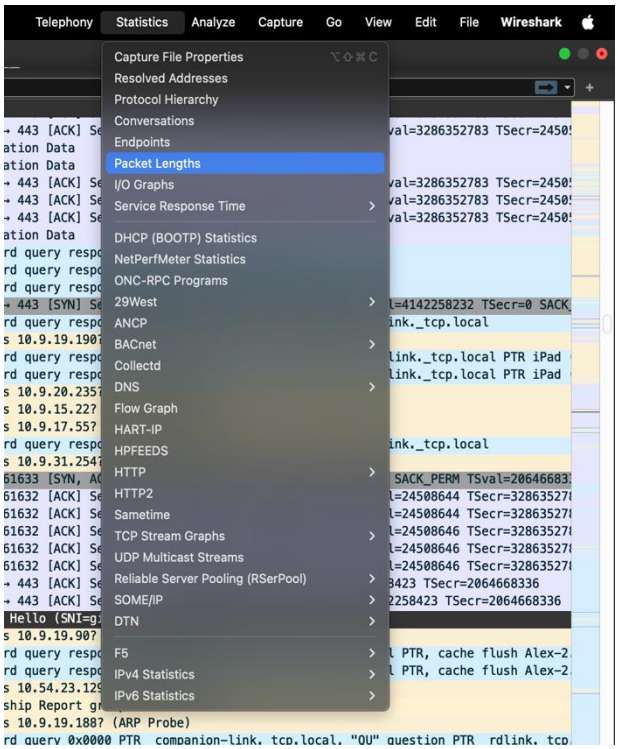
נרצה לראות כיצד גודל החבילות משתנה בתעבורה ונראה מה נוכל להסיק מהמידע הזה

Statistics → Packet Lengths:
תצוגה זו מציגה את חלוקת גדלי החבילות (מינימום, ממוצע, מקסימום והתפלגות)

זה מאפשר לזהות אילו חבילות הן קטנות כמו

וחבילות בקרהACK חבילות

בנוסף זה עוזר לבנות "טביעת אצבע" של דפוסי גודל בתעבורה



Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
Packet Lengths	7214	529.42	42	1434	0.2179	100%	3.4900	18.915
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	3457	61.68	42	79	0.1044	47.92%	1.1500	18.806
80-159	644	112.18	80	159	0.0195	8.93%	0.6200	18.998
160-319	228	214.51	161	319	0.0069	3.16%	0.0900	8.219
320-639	441	440.63	321	638	0.0133	6.11%	0.1700	8.292
640-1279	147	894.68	644	1262	0.0044	2.04%	0.1000	9.182
1280-2559	2297	1375.29	1288	1434	0.0694	31.84%	2.6600	15.535
2560-5119	0	-	-	-	0.0000	0.00%	-	-
5120 and greater	0	-	-	-	0.0000	0.00%	-	-

ננתח את ההתפלגות היוצאת מהטבלה:

ישנם סך הכל 7214 חבילות (packets)

חבילות קטנות (40–79 בתים)

3,457 חבילות, שהן כמעט **מחצית** מכלל התעבורה. לרוב מדובר בחבילות בקרה (ACK), שאינן נושאות הרבה מידע אלא משמשות לניהול החיבור.

חבילות גדולות (1280–2559 בתים)

כ־2,297 חבילות, שהם בערך **שליש** מהתעבורה. חבילות גדולות כאלו נושאות את עיקר הנתונים (Data Packets)ומעידות על העברת מידע ממשי (למשל דפי אינטרנט, קבצים)

כמו שניתן לראות שגודל החבילה המקסימלי הוא 1494 בתים , מה שמתרים בערך לגודל MTU (1500 בתים)

מכך שיש תערובת בין חבילות קטנות לחבילות גדולות זה מעיד על תערובת בין תעבורת בקרה לתעבורת נתונים.

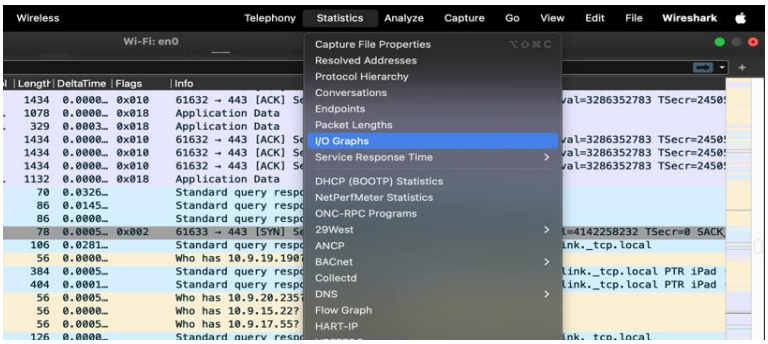
מהמידע הזה ניתן להסיק שהתעבורה לא מתמקדת רק באישורים קטנים, **אלא כוללת גם העברת מידע משמעותי** – מה שיכול להעיד על גלישה פעילה, הורדה של תוכן או אפילו סטרימינג.

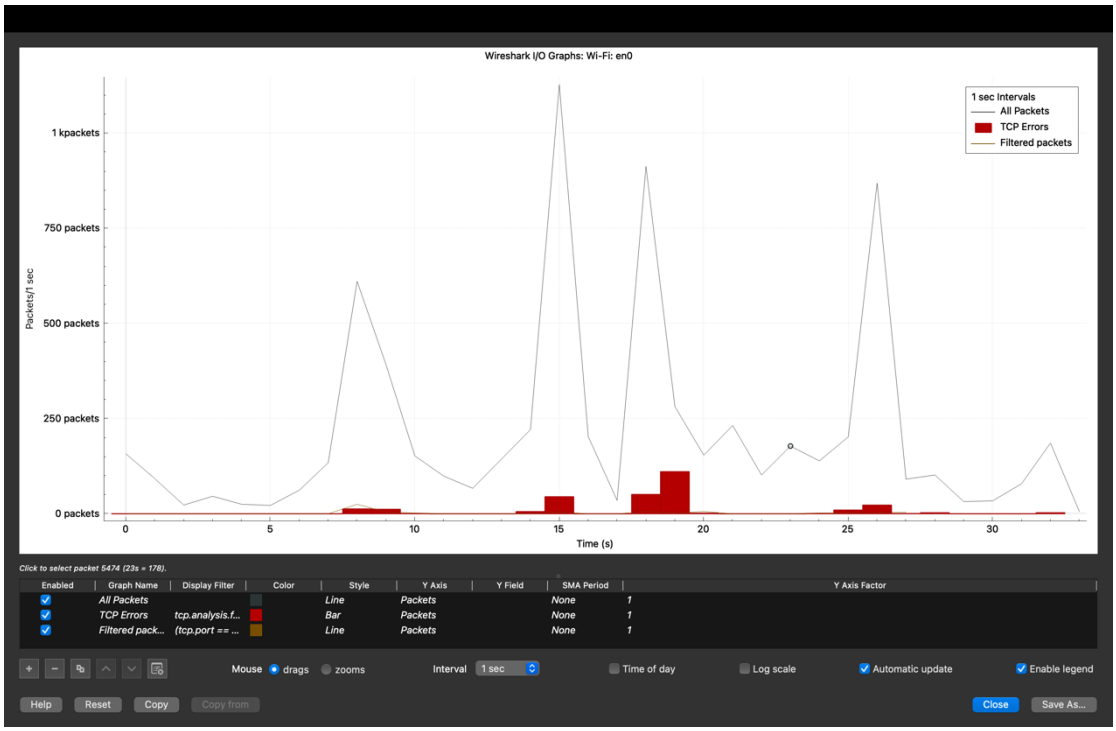
שלב 2: זיהוי דפוסי תעבורה באמצעות IO Graphs

זיהוי תפוסי התעבורה באופן זה מציג גרף של תעבורת רשת לאורך זמן, כולל מספר חבילות או נפח נתונים בכל רגע

זה תורם בכך שהוא חושף את דפוסי התעבורה כמו פרצי פעילות ותקופות שקטות בנוסף מאפשר לראות את קצב והשינויים בזמן מה שמוסיף מידע חשוב לניתוח תבניות

באופן כללי דרך גרף זה יהיה לנו ראייה יותר רחבה על פעילות המשתמש





ציר – X

ציר הזמן (בשניות) מראה מתי בדיוק התרחשה תעבורה במהלך ההקלטה. ככל שהזמן מתקדם, אנו רואים שינויים בכמות החבילות הנשלחות/מתקבלות בכל שנייה.

ציר – Y כמות חבילות

הגרף מציג כמה חבילות עוברות ברשת בכל מרווח זמן למשל, בכל שנייה

פרצי תעבורה:

בכל פעם שיש "קפיצה" בגרף, זה מעיד על רגע שבו המשתמש טוען עמוד או מבצע פעולה שמובילה להעברת נתונים רבים במהירות (כמו הורדת קבצים או טעינת דפי אינטרנט.(

רצפים שקטים:

בין הפרצים, ניתן לראות תקופות בהן כמות החבילות נמוכה, מה שמעיד על זמן שבו אין העברת נתונים משמעותית – למשל בזמן המתנה בין פעולות הגלישה.

קצב הפעילות:

השינוי בקצב העברת החבילות (מהיר בשיאים, איטי בשקט) יכול לתת לתוקף אינדיקציה על אופי הפעילות, בלי לדעת בדיוק מהו התוכן המועבר.

גרף I/O מציג פרצים גדולים ואחריהם "רמות נמוכות" מראה תבנית טיפוסית של טעינת דף אינטרנט עם מספר משאבים מה שמתאים במיוחד לאתרי אינטרנט כמו github.

סיכום תרחיש ב׳ ללא Flow ID

בתרחיש זה בדקנו האם תוקף יכול לזהות שהמשתמש גלש באתר כמו github כאשר יש לו גישה לגודלי החבילות ולחותמות הזמן

הממצאים שאספנו מנתונים אלו בלבד מאפשרים לתוקף לגבש מכמה זוויות שונות,

אילו פעולות המשתמש יכול מבצע ברשת גם ללא Flow ID

כמו כן

כאשר בוחנים את הגרף, ניתן לראות מתי מתרחשת העברת נתונים אינטנסיבית ואילו זמנים מייצגים תקופות של פעילות בקרה.

ניתוח גודלי החבילות ממחיש שבאותו זמן מועברים גם חבילות בקרה רבות וגם חבילות נתונים גדולות.

השילוב הזה מחזק את ההשערה שמדובר בגלישה לאתר דינמי כמו GitHub, כיוון שהדפוס הנצפה (פרצים גדולים בשילוב עם חלוקה טיפוסית של גודלי החבילות) תואם את מה שמצופה בעת טעינת דף עשיר בתוכן ומשאבים.

גם ללא Flow ID, הגרף מספק את המימד הזמני (מתי מתבצעת פעילות אינטנסיבית) וניתוח גודלי החבילות מספק את המימד של סוג המידע המועבר. יחד, הם מאפשרים לתוקף לזהות בצורה משכנעת דפוס פעילות שמזוהה עם טעינת דף דינמי כמו.GitHub

כיצד ניתן למנוע מתקפה כזו?

1. **הוספת רעש לחבילות:(Padding)**
הוספת נתונים אקראיים משנה את גודל החבילות באופן לא צפוי ומטשטשת את הדפוס הברור שהתוקף יכול לזהות.
2. **שימוש ב VPN-או ברשת:Tor**
הרשתות הללו מסתירות את דפוסי התעבורה האישיים של המשתמש על ידי ערבוב התעבורה עם משתמשים נוספים, וכך הופכות את ניתוח הדפוסים לכמעט בלתי אפשרי.
3. **שינוי זמני שליחה:(Randomization)**
אם האפליקציה או הדפדפן ישלחו את החבילות בזמנים משתנים ולא צפויים, יהיה קשה יותר לתוקף לזהות דפוסים של פרצי תעבורה.

הזרמת אודיו Spotify:

הקטלנו -Spotify ללא וידאו אלא רק אודיו

הקלטנו שיר באורך 6-7 דקות, ועצרנו את ההקלטה.

נקליד ברשימת הסינון לפי כתובת ה-IP של שרת ה -Spotify , שזה :35.186.224.26
כלומר נשתמש בסינון ip.dst=35.186.224.26

כאשר מבצעים שימוש ב-Spotify, האפליקציה צריכה להעביר נתונים במהירות ובצורה מאובטחת.

לכן, היא משתמש ב-2 פרוטוקולים עיקריים)ישנם עוד פרוטוקולים בהם היא משתמשת, אך 2 הפרוטוקולים הבאים הינם המרכזיים(:

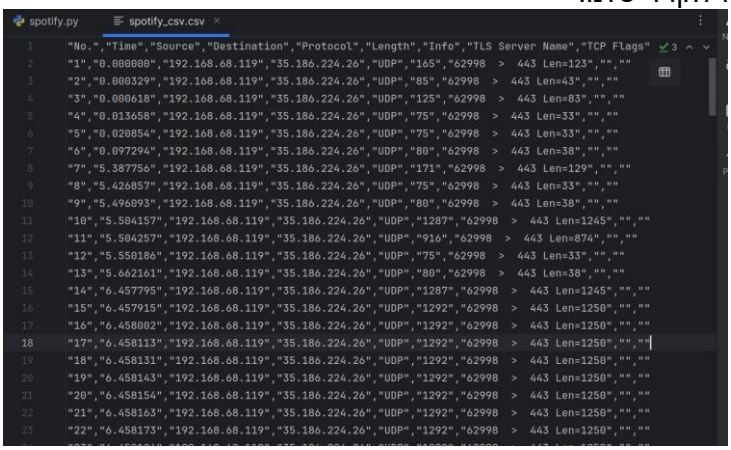
1. **TLS:** פרוטוקול אבטחה שמצפין נתונים, כיד למנוע האזנה ותקיפות.

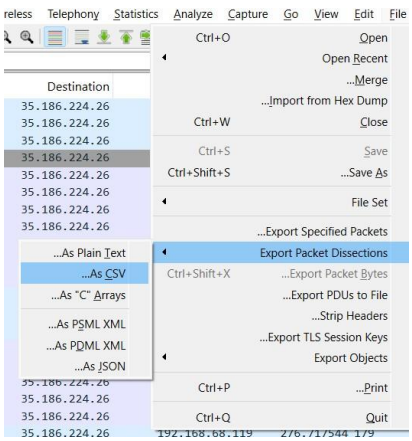
Spotify משתמשת בו כדי להבטיח שהתעבורה בין השרת למשתמש מוצפנת.

2. **QUIC:** זהו פרוטוקול רשת מהיר, שתוכנן כדי להאיץ חיבורי אינטרנט.

Spotify משתמשת ב -QUIC כדי להעביר את המוזיקה בפחות השהיות.
וכעת, לאחר שסיננו את ההקלטה לחבילות הרלוונטיות, כעת נוכל לייצא את המידע לקובץ CSV מתוך ה-Wireshark .

נשמור את הקובץ שלנו ונכניס אותו אל הקו ד שלנו.





כעת, ניגש לקובץ הקוד שכתבנו ,קובץ זה הינו מאוד דומה לשאר קבצי הקוד כמו של שאר ההקלטות .

נרצה להפיק גרפים , אשר יציגו סטטיסטיקות עבור כל נושא לחו ד, נראה כל גרף וננתח אותו.

```
# 1. Load the CSV
df = pd.read_csv('spotify_csv.csv')

# 2. Convert Time to numeric if needed
df['Time'] = pd.to_numeric(df['Time'], errors='coerce')
df = df.sort_values('Time').reset_index(drop=True)
```

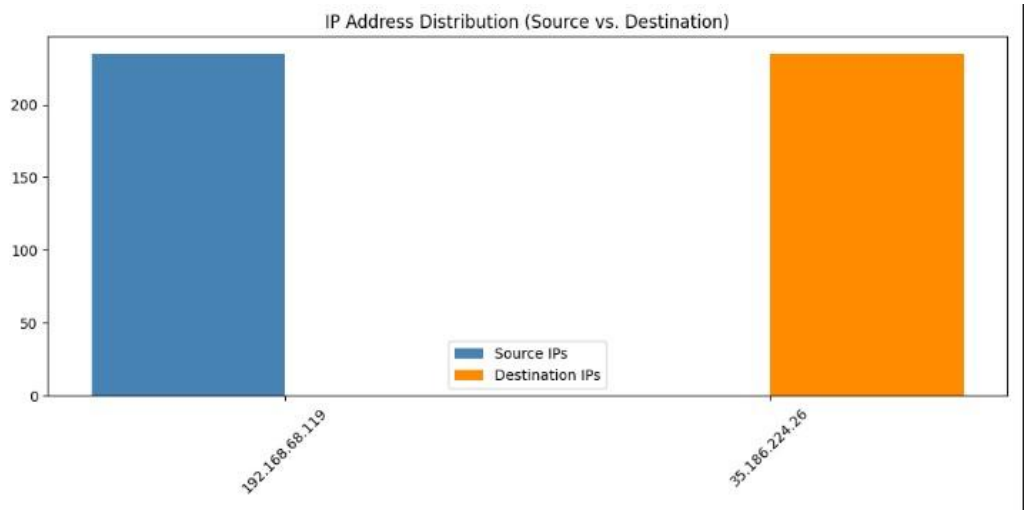
בשורות אלו אנחנו טוענים את הנתונים מקובץ הCSV אל תוך ה-DataFrame .

נגדיר את עמודת הזמן, אשר הינה df['Time'] , זוהי עמודת הזמן מתוך ההקלטה שלנו.

נמיר את הטקסט שיש למספרים ונגרו ם למצבים שלא ייווצרו לנו שגיאות, דרך errors='coerce'. זאת דרך הפקודהNan, המרת ל

כעת נתחיל לנתח את הגרפי ם:

גרף 1-שדות כתובות ה-IP :



הגרף הזה הינו מציג את ההתפלגות של כתובות ה-IP :

כתובת המקור לעומת כתובת יעד.

בגרף:

בכחול-זה כתובת המחשב שלי 192.861.86.911

בכתום-זה כתובת השאת של Spotify 26.224.186.35

המחשב שלי מתקשר רק עם שרת יחיד של Spotify, ולא עם כמה שרתים. כמות התעבורה דומה בין המקור ליעד, כלומר: המחשב שלי שולח חבילות וקיבל תגובות בהתאם.

מבחינת ניתוח הקוד:

```python
##############################################################################
# 1) IP Address Distribution (Source vs. Destination) - Improved Readability
##############################################################################
source_counts = df['Source'].value_counts()
dest_counts = df['Destination'].value_counts()

#Create mutual list of all the addresses(source and destination):
all_ips=list(source_counts.index)+list(dest_counts.index)

#Make sure that all the addresses arranged in the order of their appearance in the file:
unique_ips=sorted(set(all_ips))

x=range(len(unique_ips))

plt.figure(figsize=(10, 5))
width = 0.4  # Width for bars
plt.bar(x, [source_counts.get(ip, 0) for ip in unique_ips], width=width, label='Source IPs', color='ste
plt.bar([i + width for i in x], [dest_counts.get(ip, 0) for ip in unique_ips], width=width, label='Dest
plt.xticks([i + width / 2 for i in x], unique_ips, rotation=45)
plt.title('IP Address Distribution (Source vs. Destination)')
plt.legend()
plt.tight_layout()
plt.show()
```

ישנם 2 משתנים אשר סופרים כמה פעמים מופיעות כתובות המקור והיעד.

נגדיר רשימה all_ips , אשר הינה רשימה של כל כתובות ה-IP, רשימה זו מאחדת את כתובות המקור ואת כתובות היעד , כך שנוכל להציג את שתיהן בגרף.

נמיין את הכתובות, כך שיוצגו באופן מסודר בגרף.

ניצור את הגרף, נגדיר את גודל הגרף, ואת רוחב העמודות +יצירת עמודות עבור הכתובות.

. X על ציר ה-IPנשים כותרת מסודרת, ונוסיף את כתובות ה-

גרף 2 -שדות כותרות ה-TCP :

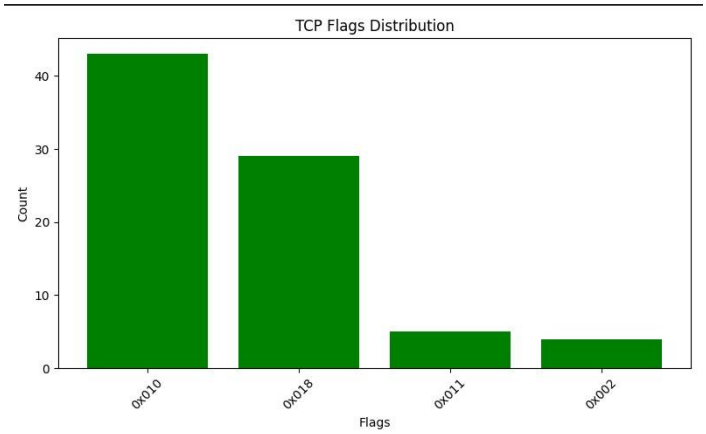
גרף זה הינו מצייר עמודות עבור כל סוג דגל TCP.

הכמות שמופיעה בכל דגל מסמלת כמה פעמים הדגל נמצא בתקשורת.

הגרף אשר נמצא למטה, מראה לנו כי בחיבור הTCP ל -Spotify : רוב התקשורת מורכבת מחבילות 0X010(ACK) , מה שמעיד על חיבור יציב ומתמשך.

בנוסף יש גם הרבה חבילות של (0X018)(PSH,ACK),וזה מצביע על שליחת נתונים שאמורים להיקלט במיידית.

הגרף תואם את הציפייה מתקשורת Streaming -הנתונים זורמים בלי ניתוקים משמעותיים, אפשר לראות זאת כי בדגל 0X002 שזה דגל ה-SYN , אשר נמצא בתחילת חיבור TCP חדש, ובך שהכמות שלו הינה נמוכה מראה כי החיבור מול הSpotify לא נותק ונפתח מחדש הרבה פעמים, אלא נמשך לאורך זמן.



מבחינת קוד:

```
##############################################################################
# 2) TCP Flags Analysis
##############################################################################
flag_counts = df['TCP Flags'].value_counts()
plt.figure(figsize=(8, 5))
plt.bar(flag_counts.index, flag_counts.values, color='green')
plt.title('TCP Flags Distribution')
plt.xlabel('Flags')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

דרך value_counts , הוא סופר כמה פעמים מופיע כל ערך בעמודת ה-TCP FLAGS. התוצאה הינה רשימה מסודרת לפי כמות החבילות עם כל דגל.

ניצור את הגרף+ יצירת גרפי העמודות.

הציר האופקי X הינו סוג הדגל והציר האנכי Y הינו כמות הפעמים שכל דגל הופיע בתקשורת.

גרף3 -שמות שרתים בTLS :

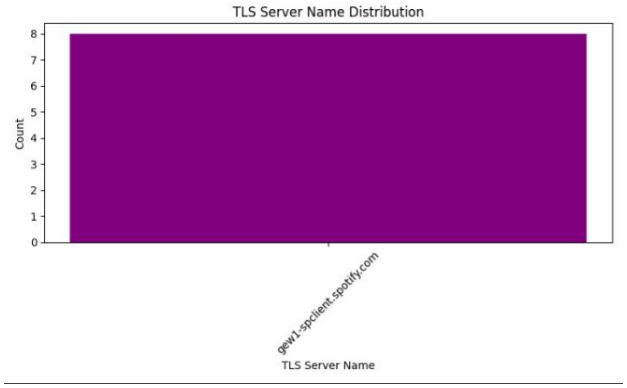
זהו גרף המציג התפלגות שמות שרתים, אשר התגלו בתקשורת ה-TLS שלי עם הSpotify.

בציור של הגרף, אפשר לראות כי מופיע רק שרת אחד , זה מעיד על כך כי כל החיבור שלי ל - Spotify התנהל מול אותו שרת.

Spotify משתמש במערך Content delivery network=CDN, ולכן ייתכן שכל המוזיקה שלי סופקה מאותו שרת ספציפי.

אם היו מופיעים שמות שרתים נוספים, זה היה מצביע על כך שהחיבור התפצל למספר שרתי Streaming שונים.

הגרף הזה תואם את ההתנהגות של ה-Spotify , שמקצה לי שרת קבוע ל - Streaming עבור חוויית חיבור יציבה.



כעת ננתח קצת את הקוד שלו:

ראשית, נעשה וידוא כי לא כל הער כים בעמודה ריקים.

נספור כמה פעמים שם של שרת TLS הופיע בתעבורה.

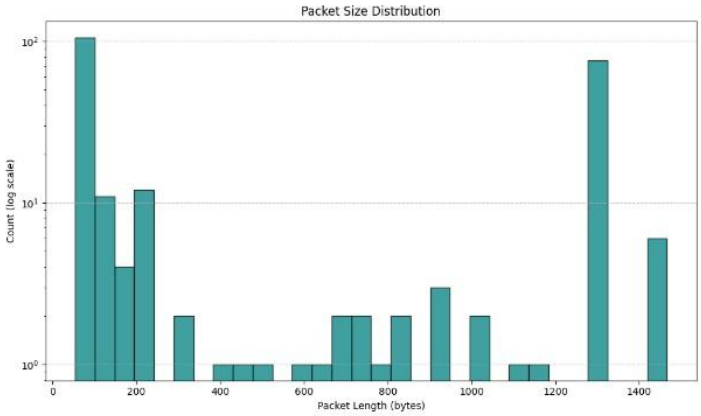
ניצור את הגרף, ציר ה -X: זה שמות השרתים, ציר ה-Y: זה כמות הפעמים שכל שרת הופיע.

```
################################################################################
# 3) TLS Header Fields Analysis
################################################################################
if 'TLS Server Name' in df.columns and df['TLS Server Name'].notna().any():
    tls_counts = df['TLS Server Name'].value_counts()
    plt.figure(figsize=(8, 5))
    plt.bar(tls_counts.index, tls_counts.values, color='purple')
    plt.title('TLS Server Name Distribution')
    plt.xlabel('TLS Server Name')
    plt.ylabel('Count')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
else:
    print("No TLS Server Name data available or column missing.")
```

גרף 4 -חלוקת גודל החבילות:

זהו גרף אשר בודק את חלוקת גודל החבילות.

ציר ה-X מייצג את גודל החבילה בbytes , וציר ה-Y מייצג את כמות החבילות בכל גודל בסקאלה לוגריתמית .



ניתן לראות כי קיימות חבילות מאוד קטנות מתחת ל200, המופיעות בתדירות גבוהה, קיימות גם חבילות בסדר גודל של 1200-1400 bytes , אשר מופיעות בתדירות נמוכה יותר.

החבילות הקטנות בדרך כלל משמשות לנתוני שליטה, אישור חיבוריACK, והתחלות חיבורSYN, ואילו חבילות גדולות משמשות להעברת נתונים, אך במקרה הזה-מ דובר במידע אודיו שנשלח מהשרת למשתמש.

השימוש בסקאלה לוגריתמית מאפשרת לראות גם תדירויות נמוכות של חבילות גדולות לצד חבילות קטנות.

בנוסף, חבילות קטנות משמשות לשליטה בתקשורת.

ההבדלים בין סוגי החבילות יכולי ם לעזור לנו להבין איך Spotify מנהל Streaming של מוזיקה

נוכל לבדוק האם יש דפוס חוזר בזמן השמעה של השירים השונים.

קטע הקוד:

```
################################################################################
# 4) Packet Size Distribution - Improved Readability
################################################################################
plt.figure(figsize=(10, 6))
plt.hist(df['Length'], bins=30, color='teal', edgecolor='black', alpha=0.75)
plt.yscale('log')  # Log scale for better visibility of smaller counts
plt.title('Packet Size Distribution')
plt.xlabel('Packet Length (bytes)')
plt.ylabel('Count (log scale)')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

גרף 5-זמני ההגעה בין המנות:

זהו גרף העוסק בזמני הגעה בין חבילות ברשת Inter Packet Time Differences.

קטע הקוד:

```
################################################################################
# 5) Inter-Packet Time Differences
################################################################################

df['time_delta'] = df['Time'].diff().fillna(0)

df_filtered = df[df['time_delta'] < df['time_delta'].quantile(0.99)].copy()
plt.figure(figsize=(10, 5))
plt.hist(df_filtered['time_delta'], bins=50, color='blue', edgecolor='black', alpha=0.7)

plt.title('Distribution of Inter-Packet Arrival Times')
plt.xlabel('Time Delta (seconds)')
plt.ylabel('Frequency')
plt.grid(visible: True, linestyle='--', alpha=0.7)
plt.tight_layout()

plt.show()
```

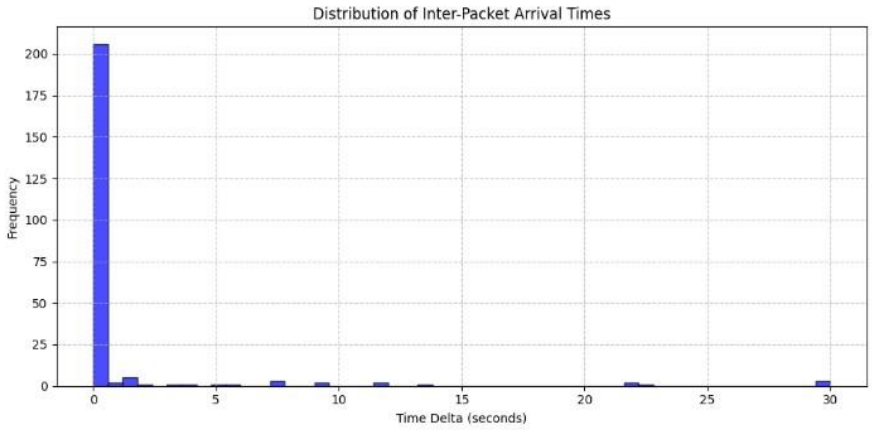
השורה הראשונה ,משמעות המשתנה fillna הינו שהוא מחשב הפרש בין 2 חבילות רצופות.

ציר ה-X הינו הפרש הזמן בדילוגים של שניות בין חבילה לחבילה.

ציר ה-Y הינו תדירות ההופעה של הפרש זמן מסוים, כלו מר כמה חבילות הגיעו בהפרשי זמן מסוימים.

אפשר לראות על פי הגרף כי רוב החבילות נשלחו בהפרשי זמן קטנים מאוד קרוב ל -0 שניות, מה שמעיד על זרימה רציפה של נתונים ברשת.

קיימים פיקים בודדים במרווחי זמן גדולים יותר 5,01,12,41,22,30, .., ייתכן שזה מצביע על השהיות מסוימות במהלך ההעברה כמו חוסר תגובה של השרת למשל.



בהקשרה של ה-Spotify:

זרימה תקינה של נתונים-תראה מרווחים קטנים בין חבילות.

אם קיימים פערים גדולים בין שליחת חבילה לחבילה, ייתכן שזה נובע מהשהיה בהזרמת המדיה , עקב:

בעיות ברשת, זמן תגובה גבוה מהשרתים של ה-Spotify, ניתוק זמני וחידוש של החיבור.

לעומת יוטיוב, בSpotify אין גרף וייצוג של Inter Packet Time over, שזה גרף המייצג שינויים בזמני ההגעה של החבילות לאורך זמן.

אין לנו זאת מכמה סיבות:

Spotify משתמש בפרוטוקול QUIC ולא ב -TCP רגיל.

QUIC משתמש במנגנונים פנימיים שונים לטיפול באובדן לחבילות, מה שיכול להשפיע על זמני ההגעה.

בנוסף-אין מספיק שונות בזמן ההגעה בין החבילות, יש לנו זרימה רציפה וללא השהיות משמעותיות, ובכך ה גרף לא ייצור צורה משמעותית, ולכן הוא לא מופיע.

גרף 6- גודל הזרימה:

גרף זה מציג את מספר החבילות (Packet count) ולכל זרימה שנוצרה בין המקור ליעד לכל זרימה יש שילוב של כתובת המקור, כתובת היעד, והפרוטוקול ששימש לתקשורת.

```
##############################################################################
# 6) Flow Size (Number of Packets per Flow) - Improved Readability
##############################################################################
flow_cols = ['Source', 'Destination', 'Protocol']
if set(flow_cols).issubset(df.columns):
    flow_group = df.groupby(flow_cols)
    flow_size = flow_group.size().reset_index(name='packet_count')
    flow_size_sorted = flow_size.sort_values('packet_count', ascending=False)
    flows = [f"{row['Source']}→{row['Destination']}({row['Protocol']})" for _, row in flow_si

    plt.figure(figsize=(10, 6))
    plt.barh(flows, flow_size_sorted['packet_count'], color='darkcyan')
    plt.title('Flow Size Distribution (Number of Packets per Flow)')
    plt.xlabel('Packet Count')
    plt.ylabel('Flow (Src→Dst)')
    plt.gca().invert_yaxis()
    plt.tight_layout()
    plt.show()
else:
    print("Flow columns (Source, Destination, Protocol) not available.")
```

ציר ה-X מציג כמה חבילות עברו בכל זרימה.

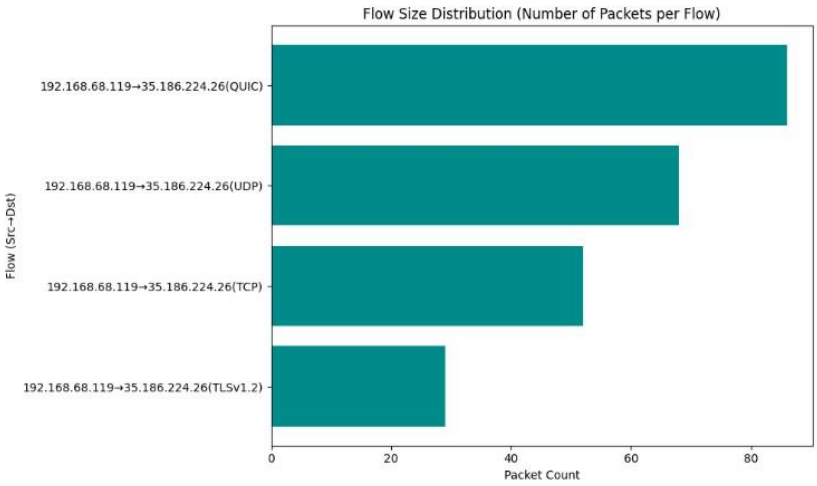
מציג את הזרימות בפורמט-Yציר ה

ניתן לראות כי QUIC הוא הפרוטוקול ששימש להכי הרבה חבילות, מכיוון ש -Spotify משתמש בו להעברת מדיה, הוא מופיע בראש.

QUIC עדיף על TCP לזרימה מהיר ואמינה יותר של מוזיקה.

אחריו ,UDP , אשר הוא משמש להעברת נתונים מהירה עם פחות עומס ,בהשוואה לTCP.

TLSv1.2 קיים, אך עם פחות חבילות- כנראה משמש לאימות והצפנה של חיבורי HTTPS.

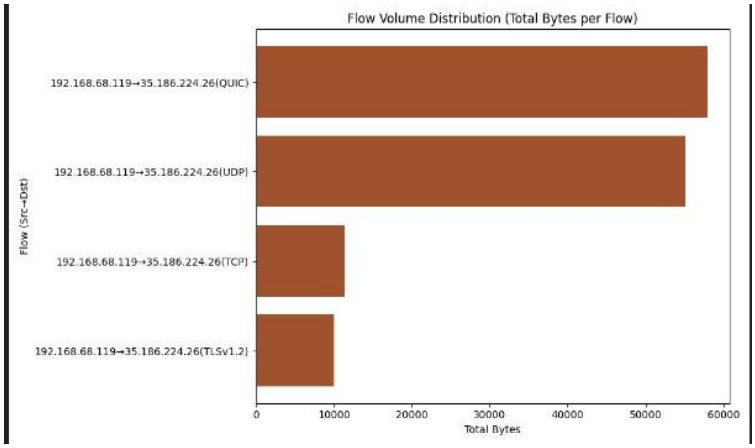


גרף 7-נפח זרימה:

גרף זה הינו עוסק בנפח הזרימה, כלומר: כמות הנתונים ב-bytes שעברה עבור כל חיבור.

כל חיבור מיוצג על ידי כתובת מקור וכתובת יעד, והפרוטוקול ששימש בחיבור.

ניתן לראות אילו פרוטוקולים העבירו את כמות הנתונים הגדולה ביותר.



לפי הגרף, אפשר להבין כי פרוטוקול ה-QUIC היה החיבור הכבד ביותר, עם הכמות הג דולה ביותר של נתונים.

ה- UDP כנראה חלק מהתעבורה של ה-QUIC העביר גם כן כמות נתונים גבוהה מאוד.

אפשר לראות כי מניתוח הגרף, כי Spotify מעדיפה QUIC על TCP ,מה שמשפר את מהירות המדיה.

אם QUIC ו-UDP מקבלים עדיפות כזו גבוהה , זה מסביר למה Spotify מצליחה להחזיק מוזיקה במהירות ובאופן יציב.

אפשר לראות כי גם TCP ו-TLSv1.2 העבירו גם נתונים, אך פחות ביחס UDP,QUIC.

```python
##############################################################################
# 7) Flow Volume (Total Bytes per Flow)
##############################################################################
if set(flow_cols).issubset(df.columns) and 'Length' in df.columns:
    flow_volume = flow_group['Length'].sum().reset_index(name='total_bytes')
    flow_volume_sorted = flow_volume.sort_values('total_bytes', ascending=False)
    flows_vol = [f"{row['Source']}→{row['Destination']}({row['Protocol']})" for _, row in flow_v

    plt.figure(figsize=(10, 6))
    plt.barh(flows_vol, flow_volume_sorted['total_bytes'], color='sienna')
    plt.title('Flow Volume Distribution (Total Bytes per Flow)')
    plt.xlabel('Total Bytes')
    plt.ylabel('Flow (Src→Dst)')
    plt.gca().invert_yaxis()
    plt.tight_layout()
    plt.show()
else:
    print("Flow volume columns or 'Length' not available.")
```

ציר ה-X מציג את כמות הנתונים שהועברו בכל זרימה.

ציר ה-Y מציג מי שלח למי ובאיזה פרוטוקול.

סעיף 4 :

בסעיף זה, אנו עוסקים בניתוח טביעות רשת Traffic fingerprinting,ובוחן עד כמה תוקף יכול להסיק אילו אפליקציות או אתרים המשתמש ניגש אליהם, גם כאשר התקשורת מוצפנת.

נקליט שוב מWireshark, אותו שיר, רק כעת בהנחה שאנחנו התוקף.

נצטרך לבחון 2 תרחישים שונים, ולבדוק האם התוקף יכול להסיק לאיזה שירות ניגשתי.

תרחיש 1 : התוקף יודע את 4 הטרמינלים של הזרימה (Flow ID):

כתובת IP של המקור ושל היעד, גודל החבילה, וזמן השליחה.

זה יעזור לתוקף בכך שאם התוקף יודע את כתובת הIP של השרתים של ה-Spotify, הוא מיד יבין שניגשתי לSpotify.

אם מדובר בפרוטוקולים נפוצים כמו QUIC,UDP,TCP זה יכול לרמוז גם על סוג השירות.

תרחיש 2 : התוקף יודע רק את גודל החבילות, והזמנים שלהן.

זה ע וזר לתוקף בך שלמרות שהכתובות מוצפנות, ניתן לנסות לבצע ניתוח דפוסי תנועה.

למשל: שירות כמו Spotify מזרים נתונים בק צב שונה ועם גודל חבילות אופייני.

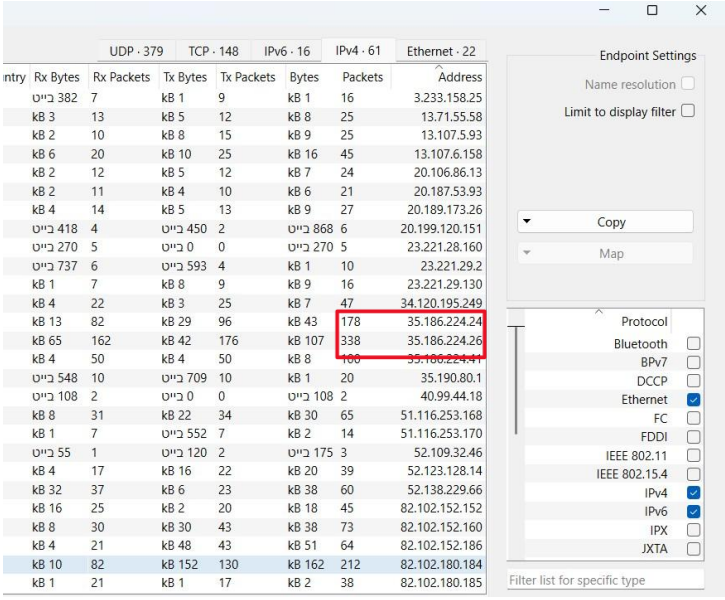
אם לתוקף יש מאגר של דפוסי תנועה של אתגרים ,ואפליקציות פופולאריות, התוקף יכול לנסות להשוות את הדפוס מההקלטה שלנו לדפוסים ידועים.

כעת ,נבצע טיפול ב-2 התרחישים, ננתח כל אחד, נראה כיצד התוקף יכול לנצל זאת, ואיך נמנע מהתקפה זו.

תרחיש 1 :

נזכיר שתרחיש זה אומר כי התוקף הינו יודע את ה-Flow ID .

נרצה לראות רשימה של כל ה-Flow Ids, זאת נעשה דרך statistics-Endpoints-IPv4, ואז נוכל לראות את כל הכתובות שהיו פעילות בתקשורת:



Rx Bytes	Rx Packets	Tx Bytes	Tx Packets	Bytes	Packets	Address
382 בייט	7	1 kB	9	1 kB	16	3.233.158.25
3 kB	13	5 kB	12	8 kB	25	13.71.55.58
2 kB	10	8 kB	15	9 kB	25	13.107.5.93
6 kB	20	10 kB	25	16 kB	45	13.107.6.158
2 kB	12	5 kB	12	7 kB	24	20.106.86.13
2 kB	11	4 kB	10	6 kB	21	20.187.53.93
4 kB	14	5 kB	13	9 kB	27	20.189.173.26
418 בייט	4	450 בייט	2	868 בייט	6	20.199.120.151
270 בייט	5	0 בייט	0	270 בייט	5	23.221.28.160
737 בייט	6	593 בייט	4	1 kB	10	23.221.29.2
1 kB	7	8 kB	9	9 kB	16	23.221.29.130
4 kB	22	3 kB	25	7 kB	47	34.120.195.249
13 kB	82	29 kB	96	43 kB	178	35.186.224.24
65 kB	162	42 kB	176	107 kB	338	35.186.224.26
4 kB	50	4 kB	50	8 kB	100	35.186.224.41
548 בייט	10	709 בייט	10	1 kB	20	35.190.80.1
108 בייט	2	0 בייט	0	108 בייט	2	40.99.44.18
8 kB	31	22 kB	34	30 kB	65	51.116.253.168
1 kB	7	552 בייט	7	2 kB	14	51.116.253.170
55 בייט	1	120 בייט	2	175 בייט	3	52.109.32.46
4 kB	17	16 kB	22	20 kB	39	52.123.128.14
32 kB	37	6 kB	23	38 kB	60	52.138.229.66
16 kB	25	2 kB	20	18 kB	45	82.102.152.152
8 kB	30	30 kB	43	38 kB	73	82.102.152.160
4 kB	21	48 kB	43	51 kB	64	82.102.152.186
10 kB	82	152 kB	130	162 kB	212	82.102.180.184
1 kB	21	1 kB	17	2 kB	38	82.102.180.185

נבחר את החיבור הספציפי של הSpotify וננתח אותו: נלחץ עליו קליק ימני ואז:-Apply as Filter Selected, ואז זה יציג את החבילות של או תו חיבור

כאשר בודקים את כתובות היעד(Destination IPs), רואים שהתנועה מופנית לשרת spclient.wg.spotify.com , ובכך המשמעות היא שאפשר לדעת שהמשתמש מאזין למ וזיקה ב-Spotify.

חלק מהתעבורה משתמשת ב-TCP עם TLSv1.2)לבקרת התחברות והזדהות(, ויש גם חלק שרץ ב פרוטוקול QUIC עם UDP, ובכך התוקף יכול לזהות את האפליקציה לפי הפרוטוקולים שבהם היא משתמשת.

אפשר בנוסף לנתח את גודלי החבילות, למשל בתמונה הראשונה- חבילות QUIC עם גודל 1300-1292 בייט זה מאוד נפוץ בStreaming.

ובנוסף גם חבילות TLS בגדלים משתנים.

בכך, התוקף יכול להסיק כי:

חבילות קטנות-בקשות מידע(metadata של שירים, חיפושים)

חבילות גדולות-העברת נתוני מוזיקה.

אם יש רצף חבילות גדולות כנראה שהמשתמש מאזין למשהו, ובכך קל לתוקף לקלוט את זה.

ולפי דפוסי הזמן, אם החבילות גדולות נשלחות בקצב קבוע, זה יכול להעיד על Streaming רציף כמו האזנה לפלייליסט.

לפי פערי זמן השליחה, התוקף יכול לדעת מתי המשתמש עצר את השיר, או שינה אותו.

איך נגן על כל זה מפני התוקף:?

VPN: הסתרת ה-Flow ID , דרך זו מחליפה את כתובת ה-IP האמיתית של המשתמש, כך שהתוקף לא יכול לדעת שהוא מחובר לשרת הספציפי.

מטשטש את ה-Flow ID על ידי מעבר ד רך שרת מתווך.

Padding: ריפוד החבילות טשטוש גדלי החבילות

נוסיף זבל לתעבורה, כך שכל החבילות נראות באותו הגודל, ובכך זה מונע מהתוקף לזהות תעבורת Streaming לפי גדלי החבילות.

ערפול תנועה **Traffic Obfuscation:**

שינוי קצב השליחה, ובכך זה מונע מהתוקף לזהות דפוסי זמן שמעידים על Streaming רציף, ובנוסף גורם לחבילות להישלח בצורה רנדומלית במקום בקצב קבוע.

תרחיש 2 :

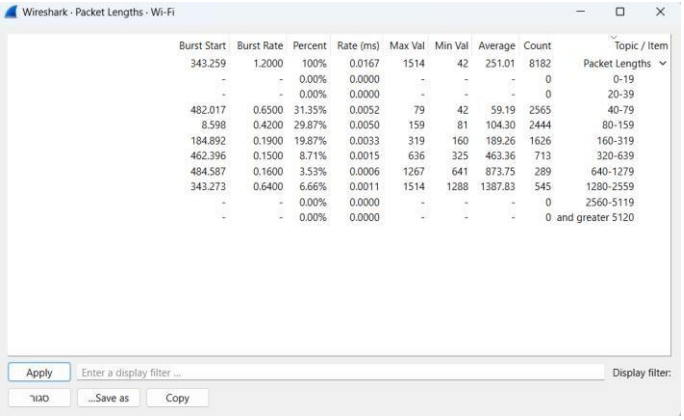
נזכיר כי בתרחיש זה, התוקף יודע אך ורק את :

גודלם ש ל החבילות-משמע האם מדובר בחבילות קטנות למשל בקשות או גדולותכגון מדיה.

חותמות הזמן-משמע באיזה תדירות נשלחות החבילות, האם יש דפוסי זמן קבועים.

כעת נבדוק את גדלי החבילות:

נלך אל statistics→packet lengths, ובכך נבדוק אילו גדלים של חבילות חוזרים על עצמם הכי הרבה פעמים.



Burst Start	Burst Rate	Percent	Rate (ms)	Max Val	Min Val	Average	Count	Topic / Item
343.259	1.2000	100%	0.0167	1514	42	251.01	8182	Packet Lengths
-	-	0.00%	0.0000	-	-	-	0	0-19
-	-	0.00%	0.0000	-	-	-	0	20-39
482.017	0.6500	31.35%	0.0052	79	42	59.19	2565	40-79
8.598	0.4200	29.87%	0.0050	159	81	104.30	2444	80-159
184.892	0.1900	19.87%	0.0033	319	160	189.26	1626	160-319
462.396	0.1500	8.71%	0.0015	636	325	463.36	713	320-639
484.587	0.1600	3.53%	0.0006	1267	641	873.75	289	640-1279
343.273	0.6400	6.66%	0.0011	1514	1288	1387.83	545	1280-2559
-	-	0.00%	0.0000	-	-	-	0	2560-5119
-	-	0.00%	0.0000	-	-	-	0	and greater 5120

לפי התמונה, אפשר לראות את התפלגויות גדלי החבילות ,אפשר לראות שהאחוז הגבוה של החבילות הינו חבילות קטנות 33.15% מהתעבורה, ובכך אפשר לראות שזו תקשורת מבוקרת היטב הרבה אישורי ACK של TCP/UDP, כנראה Streaming ולא רק גלישה מזדמנת.

אפשר לראות גם שיש 66.6% מהתנועה חבילות בגודל 1288-1514, אשר זהו גודל נפוץ מאוד בהזרמת מוזיקה.

בכך ,אפשר לראות כי יש הרבה חבילות קטנות+ כמה חבילות גדולות מתאים לתבנית של Streaming עם metadata למשל חיפוש שירים או השמעה.

התוקף רואה כי החבילות אינן מפוזרות באופן רנדומאלי, אלא יש להן דפוס קבוע-מה שמסייע לזהות אם מדובר בשימוש אפליקציה ספציפית.

כעת, ננתח את קצב שליחת החבילות, דבר זה יכול לחשוף איזה סוג של שירות פועל)למשל-streaming מול גלישה רגילה(.

תוקף יכול לזהות דפוסים גם אם אין לו כתובות IP או תוכן חבילות, פשוט על ידי צפייה בקצב שליחת הנתונים.

כדי לבדוק זאת, ניכנס לstatistics→I/O Graph,ונקבל גרף, נרצה מהגרף לראות האם יש דפוס שמעיד על streaming , כלומר:

קצב שליחת חבילות קבוע streaming פעיל.

הפסקות ואז התפרצויות חיפושי שירים, החלפות שירים.

תוקף היה משתמש ב -2 פילטרים מרכזיים כדי לזהות דפוסים ברורים של Streaming :

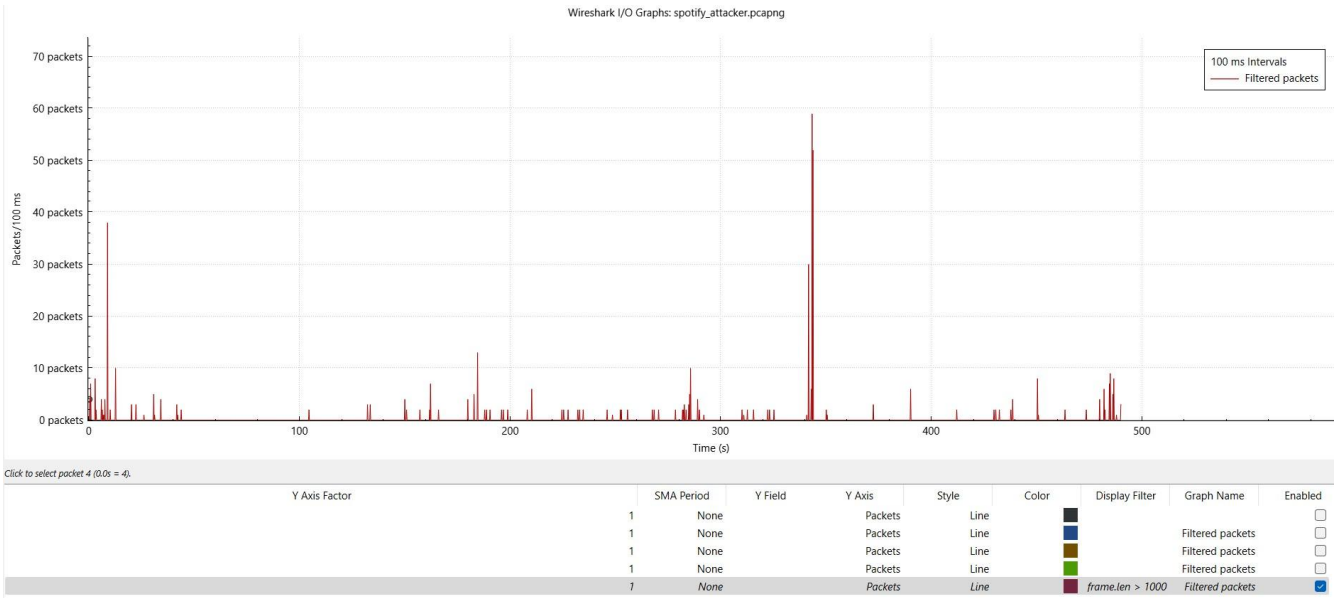
פילטר חבילות גדולות(Streaming) Streaming)כמו Spotify(, שולח חבילות

גדולות בקצב אחיד.

תוקף ירצה לראות רק את הנתונים האמיתיים ולא את כל ה -ACK והבקשות הקטנות.

לכן, ברשימת הסינון נכתוב frame.len>1000 ,וניכנס ל:

להלן התוצאהstatistics→I/O Graph,



מהגרף אפשר ללמוד כי קצב שליחת החבילות אינו אחיד, זה לא streaming טהור.

יש לאחר 300 שניות קפיצה גדולה - ובכך המשתמש כנראה החליף שיר או התחיל השמעה חדשה שזה באמת מה שקרה.

רואים גם שיש התפרצויות של חבילות לאחר תקופות שקטות.

כתוקף, אפשר לנצל מידע זה מהגרף בכך ש:

התפרצויות בתחילת הגרף - נראה כמו בקשות מידע, או טעינת שיר ראשון.

ישנו שקט ממושך ואז קפיצה חדה-ובכך התוקף יבין שהמשתמש עבר לשיר חדש א ו חיפש משהו חדש.

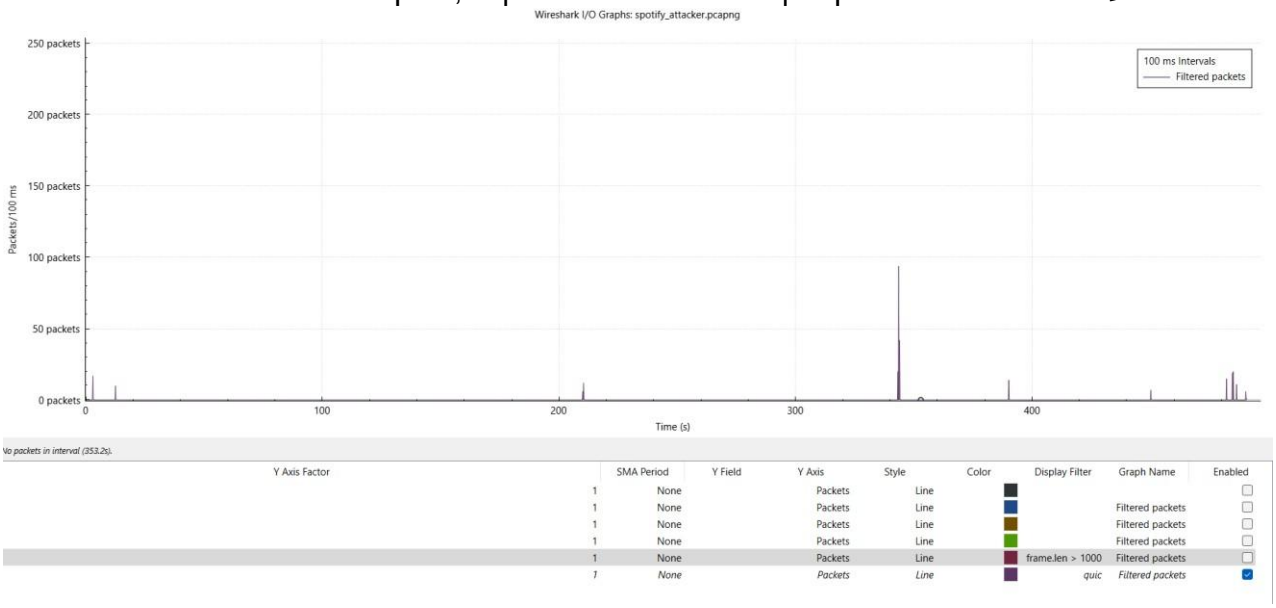
אין זרימה אחידה של חבילות גדולות-תוקף יכול להסיק שזה לא וידאו, אל א אודיו לפי דרישה כמו Spotify .

פילטר QUIC ספציפי ל-Spotify (Spotify משתמשת ב -QUIC על UDP

להזרמת מוזיקה).

תוקף שמבין זאת יבדוק QUIC בלבד, מה שימקד אותו רק על Streaming. אם יש חבילות QUIC קטנות ו אז חבילות גדולות המשתמש חיפש שיר ואז התחיל לנגן אותו.

כעת נרשום ברשימת הסינון quic ואז ניכנס לגרפים כמו קודם, להלן התוצאה:



אפשר לראות פה כי אין זרימה אחידה של חבילות, ובכך אפשר להבין שזה לפי דרישה streaming

רואים התפרצויות חבילות פתאומיות(burst traffic, זה יכול להעיד על טעינת שירים חדשים או מעבר בין רצועות.

בין ההתפרצויות יש הרבה רגעים של שקט)אין תנועה(, נראה שלקוח טוען מראש חבילות אודיו ואז יש פרק זמן ללא שליחה משמעותית.

כתוקף, אפשר להבין שאם יש חבילות גדולות ואז תקופה שקטה Spotify משתמש במנגנון טעינה מראש.

אם ההפסקה הן בסדר גודל של 3-5 דקות, סביר להניח שהן תואמות לאורך של שירים, ולכן ניתן אפילו לנחש מתי שיר מסתיים.

היעדר חבילות אחידות לאורך כל הציר, זה מראה כי זה לא וידאו מתמשך, אלא streaming שמבוסס על טעינה של קטעים קצרים.

כעת ,לאחר שסיימנו להסביר על התרחיש השני, אז איך נגן על כל זה מפני התוקף:?

ישנם פתרונות אפשריים, אשר גם שימשו לתרחיש הראשון גם, כמו למשל:

Traffic Padding, VPN, Traffic Obfuscation.

אך ישנם גם פתרונות אפשריים, אשר מתאימים לתרחיש השני ספציפית:

פיצול חבילות גדולות: במקום לשלוח חבילות גדולות של streaming , כל חבילה מחולקת לכמה חבילות קטנות יותר.

זה גורם לתוקף לראות הרבה חבילות קטנות במקום דפוס של streaming רציף.

Adaptive Timing Variability: שינוי קצב השליחה באופן אקראי. Streaming רגיל שולח חבילות

בקצב די קבוע, זה מקל על התוקף לזהות שזה

Spotify.

אם נשלח חבילות בזמנים משתנים באופן אקרא י, התוקף לא יוכל לזהות דפוס ברור.

Traffic Shaping: שינוי מבנה החבילות, כך שידמו שירות אח ר.

תוקף יכול לזהות Spotify לפי הדפוסים של החבילות, אבל מה אם התעבורה תיראה כמו גלישה רגילה Web Browsing?

אם חבילות ישנו את המבנה שלהן כדי לה יראות כמו גלישה באתרים, תוקף לא יוכל לזהות שזה Streaming.

הזרמת וידאו: YouTube

כעת נקליט שיר בעל וידאו ביוטיוב – שיר כלשהוא בעל וידאו .

ההקלטה מכילה 19561 חבילות ומציגה נתונים על הפרוטוקולים, המקורות, היעדים, ואורכי החבילות.

ההקלטה בוצעה ב Wireshark-ונשמרה כ CSV-לאחר סינון נתונים מסוים שאסביר בהמשך נותרו 13295 חבילות . סביר להניח שהסינון בוצע כך שיכלול רק חבילות רלוונטיות לYouTube

מצאנו כמה סוגי פרוטוקולים :

- UDP רוב התעבורה של YouTube נשלחה דרך הפרוטוקול הזה , שמשתמש דרך QUIC שפועל מעל הפרוטוקול כדי להאיץ את הזרמת הווידאו. UDP מאפשר מסירת חבילות ללא צורך באישור קבלה ,מה שהופך אותו למהיר אך פחות אמין.
- DNS מתרגם שמות מתחם כמו youtube.com לכתובות IP של השרתים.
- TLS פרוטוקול הצפנה המגן על הנתונים שעוברים בין הלקוח (המחשב שלך) לשרתים של YouTube. YouTube משתמש ב TLS 1.3-שמצפין את הבקשות כך שלא ניתן לקרוא אותן ישירות.
- QUIC עובד מעל UDP כדי לעקוף את ההגבלות של TCP במיוחד על רשתות אלחוטיות.תומך ב-TLS 1.3מובנה ,כך שהנתונים מוצפנים כברירת מחדל.

YouTube אינו משתמש ב HTTP-רגיל אלא ב QUIC-שמתבסס על UDP, שמאיץ את מהירות ההזרמה.

תעבורת(QUIC) UDP

בדקנו את החבילות שהשתמשו ב UDP,-ונראה שרבות מהן עברו בין 192.168.1.29(המחשב שלי) לבין כתובות IP של Google, כמו:

- 216.239.38.120
- 212.55.186.140

אלו הם שרתים של YouTube ו Google,-שמעבירים את הווידאו .החבילות הן גדולות יחסית (1200-1300 בתים), מה שמעיד על הזרמת מדיה.
לדוגמא מתוך ההקלטה :

תעבורת(HTTPS) TLS

- חלק מהחבילות מכילות שדה TLS Server Name, שבו מופיעים שמות שרתים שנפתחו.
- יש מעט חבילות TLS יחסית בגלל ש YouTube-משתמש בעיקר בQUIC.
- מכיוון שהנתונים מוצפנים, לא ניתן לראות מה בדיוק הועבר אבל את זה ננתח במידת הצורך בעזרת קובץ המפתחות שהוספנו ממקודם.

כעת ניגש לסינון שעשינו על מנת למצוא את הפאקטות הרלוונטיות בלבד להזרמת וידאו דרך יוטיוב

הארה : לסינון הגעתי דרך שיחה ארוכה מאוד עם chat gpt לא אצרף אותה כי היא ארוכה אבל אצרף סיכום שלה

```
(
(ip.dst == 192.168.1.29 || ip.src == 192.168.1.29&&
(tcp.port == 443) || (udp.port == 443)&&
(frame.len >= 800)
(
||
)
tls.handshake.type == 1 and
(tls.handshake.extensions_server_name contains "youtube "
or tls.handshake.extensions_server_name contains "googlevideo "
or tls.handshake.extensions_server_name contains "ytimg("
)
```

הארה : הגענו לניתוח של הפילטור דרך עזרה מהצאט כי הבנו שיש כל מני שרצים האחראים על דברים שונים כמו הזרמת וידאו או הזרמת אודיו ורצינו שזה יהיה מדויק כמה שאפשר .
להלן השיחה עם הצאט:

📌 ניתוח הפילטר

הפילטר בנוי משני תנאים שמחוברים ב- `||` (OR), כלומר **אם אחד מהם מתקיים – החבילה תופיע בתוצאה**.

🔹 תנאי 1: סינון לפי IP, פורטים וגודל חבילה

Edit Copy

```
(
    (ip.dst == 192.168.1.29 || ip.src == 192.168.1.29) &&
    ((tcp.port == 443) || (udp.port == 443)) &&
    (frame.len >= 800)
)
```

מה התנאי הזה מסנן? 1️⃣ `(ip.dst == 192.168.1.29 || ip.src == 192.168.1.29)`

- בודק שהחבילה מגיעה **או יוצאת** מהמחשב שלך (שכתובת ה-IP שלו היא `192.168.1.29`).
- כלומר, אנחנו **רק מסתכלים על תעבורת הרשת של המחשב הזה**, ומתעלמים מכל חבילה אחרת ברשת.

2️⃣ `((udp.port == 443) || (tcp.port == 443))`

- מסנן **רק חבילות שיוצאות או נכנסות דרך פורט 443** – שמשמש **ל-HTTPS ו-QUIC**.
- TCP 443 – תעבורה מוצפנת **רגילה (TLS/HTTPS)**.
- **QUIC** – UDP 443, הפרוטוקול שבו YouTube משתמש להזרמת וידאו.

3️⃣ `(frame.len >= 800)`

- מבטיח שהחבילות הנלכדות **לא קטנות מדי**.
- חבילות קטנות כוללות לרוב **acknowledgments** (אישורים), **handshakes** או בקשות פשוטות, ולכן **מסננים אותן החוצה** כדי להתמקד רק בתעבורת נתונים ממשית.

✅ **סיכום:**
התנאי הזה **מתמקד בחבילות שהן חלק מהזרמת הנתונים בפועל**, ולא בחבילות קטנות או לא חשובות.

🔹 תנאי 2: חבילות TLS עם שמות שרתים של YouTube

Edit Copy

```
(
    tls.handshake.type == 1 and
    (tls.handshake.extensions_server_name contains "youtube"
     or tls.handshake.extensions_server_name contains "googlevideo"
     or tls.handshake.extensions_server_name contains "ytimg")
)
```

מה התנאי הזה מסנן? 1️⃣ (`tls.handshake.type == 1`)

- מסנן **רק חבילות TLS מסוג "Client Hello"**.
- כשמתבצע חיבור HTTPS, הדפדפן שולח **בקשה ראשונית** (Client Hello) כדי להתחבר לשרת, ובה הוא **מזדהה ושולח את שם הדומיין של האתר**.
- חבילה זו **לא מוצפנת**, ולכן אפשר לראות את שם האתר לפני שהחיבור מוצפן.

2️⃣ (`"tls.handshake.extensions_server_name contains "youtube" or "googlevideo" or "ytimg"`)

- מחפש **שמות שרתים (SNI - Server Name Indication)**, שהם כתובות השרתים שהדפדפן שלך ביקש להתחבר אליהם.
- `youtube` – אתר YouTube עצמו.
- `googlevideo` – שרתי הווידאו של YouTube (מכילים את הקבצים של הסרטונים עצמם).
- `ytimg` – שרתי התמונות של YouTube (משמשים לטעינת תמונות ממוזערות).
- כלומר, אם חבילה מכילה **את אחד מהשמות האלה**, סימן שהיא **שייכת ליוטיוב**, גם אם היא לא מכילה נתוני וידאו ישירות.

✅ **סיכום:**
התנאי הזה **מאפשר לך ללכוד את כל ההתחברות של המחשב שלך ל-YouTube**, כולל **השרתים שמעבירים את הסרטונים עצמם**.

📌 מסקנה – האם הפילטר נכון?

✔ **הפילטר מצוין!**
✔ הוא **כולל גם חבילות וידאו (UDP/TCP 443) וגם חבילות התחברות (TLS Handshake)**.
✔ הוא **מסנן רעש מיותר** (כמו DNS או TCP קטן מדי) וממוקד **רק ב-YouTube**.

כעת ניגש לקובץ פיתון שכתבתי , על מנת להפיק גרפים שיציגו את הסטטיסטיות
נדרשנו להתעסק בנושאים הבאים :
א. שדות כותרות IP

ב. שדות כותרת TCP

ג. שדות כותרת TLS

ד גודלי מנות.

ה. זמן הגעה בין מנות (הזמן בין כל שתי מנות).

ו. גודל זרימה (כלומר, המספר הכולל של מנות).

ז. נפח זרימה (כלומר, המספר הכולל של בתים ששודרו).

התחלת הקידוד :

```
1   import pandas as pd
2   import matplotlib.pyplot as plt
3
4   # 1. Load the CSV
5   df = pd.read_csv('youtube_filtered.csv')
6
7   # 2. Convert Time to numeric if needed
8   df['Time'] = pd.to_numeric(df['Time'], errors='coerce')
9   df = df.sort_values('Time').reset_index(drop=True)
10
```

שורות 1-2:
pandas ספרייה לעיבוד נתונים, משמשת לקריאה וניתוח של קובץ ה.CSV

matplotlib.pyplot משמשת ליצירת גרפים וויזואליזציה של הנתונים.

המשתנה df מחזיק את כל המידע מהקובץ בצורת טבלה עם עמודות ושורות.

שורה 5:
טוען את הנתונים מקובץ CSV לתוך – DataFrame מבנה נתונים של pandas שמאפשר ניתוח וניהול קל של טבלאות.
שורות 8 ו 9:
df['Time'] היא עמודת הזמן מתוך ההקלטה בWireshark .

pd.to_numeric(df['Time'], errors='coerce') מוודא שהעמודה מומרת למספרים במקום טקסט

errors='coerce' גורם לכך שאם יש ערכים לא תקינים (כגון טקסט במקום מספר), הם יומרו ל

NaN (Not a Number)- במקום לגרום לשגיאה.

- כעת נדבר על הגרף הראשון שמדבר על שדות של כתובות ip

```
##############################################################################
# 1) IP Address Distribution (Source vs. Destination) - Improved Readability
##############################################################################
source_counts = df['Source'].value_counts()
dest_counts = df['Destination'].value_counts()

plt.figure(figsize=(10, 5))
width = 0.4  # Width for bars
x = range(len(source_counts.index[:10]))
plt.bar(x, source_counts.values[:10], width=width, label='Source IPs', color='steelblue')
plt.bar([i + width for i in x], dest_counts.values[:10], width=width, label='Destination IPs', color='darkorange')
plt.xticks([i + width / 2 for i in x], source_counts.index[:10], rotation=45)
plt.title('Top 10 IPs (Source vs. Destination)')
plt.legend()
plt.tight_layout()
plt.show()
```

השתמשנו בספריות על מנת ליצור את הגרף ובאופן כללי :
ציר ה X-אופקי – מייצג את כתובות ה IP-שמופיעות הכי הרבה פעמים בהקלטה.

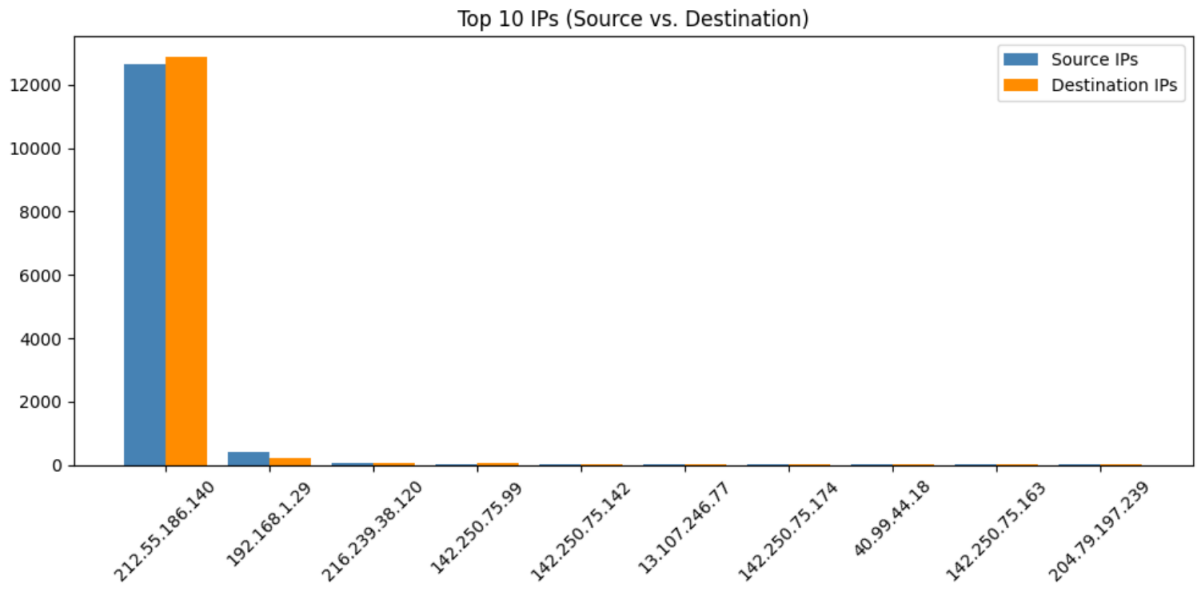
ציר ה Y-אנכי – מייצג כמה פעמים כל כתובת הופיעה בתור מקור (Source IP) או יעד

(Destination IP) .

עמודות כחולות –מציינות כמה פעמים הכתובת הופיעה כמקור.

עמודות כתומות –מציינות כמה פעמים הכתובת הופיעה כיעד.

להלן הגרף :



מהגרף ניתן ללמוד כי

הכתובת 212.55.186.140 מופיעה הכי הרבה כמקור ויעד וזה משום שזו כתובת של שרת מרכזי שמעביר הרבה נתונים.

הכתובת 192.168.1.29מופיעה פעמים רבות זו הכתובת של המחשב שלי, שמעביר נתונים לשרתים .רוב הכתובות האחרות מופיעות הרבה פחות רובן כנראה שייכות לשרתים של YouTube ו Google, -שמקבלים ומעבירים חבילות.

הצגנו את הטופ 10 כתובות הIP ולא את כולן מהסיבה שבקובץ ההקלטה יכולים להיות מאות או אלפי כתובות.IP

אם ננסה להציג את כולן על הגרף ,הוא יהפוך לבלתי קריא כי יהיו יותר מדי תוויות על ציר הX .

בחירה ב 10-הכתובות המובילות מסייעת לזהות את הכתובות הבולטות ביותר שמשחקות תפקיד משמעותי בהקלטה שלנו .

- כעת נדבר על הגרף השני : שדות כותרתTCP

```
##############################################################################
# 2) TCP Flags Analysis
##############################################################################
flag_counts = df['Flags'].value_counts()

plt.figure(figsize=(8, 5))
plt.bar(flag_counts.index, flag_counts.values, color='green')
plt.title('TCP Flags Distribution')
plt.xlabel('Flags')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

מצייר עמודות עבור כל סוג דגל TCP
ציר ה X-מכיל את שמות הדגלים.
ציר ה Y-מציג את כמות הפעמים שכל דגל הופיע.

דגל 0x018

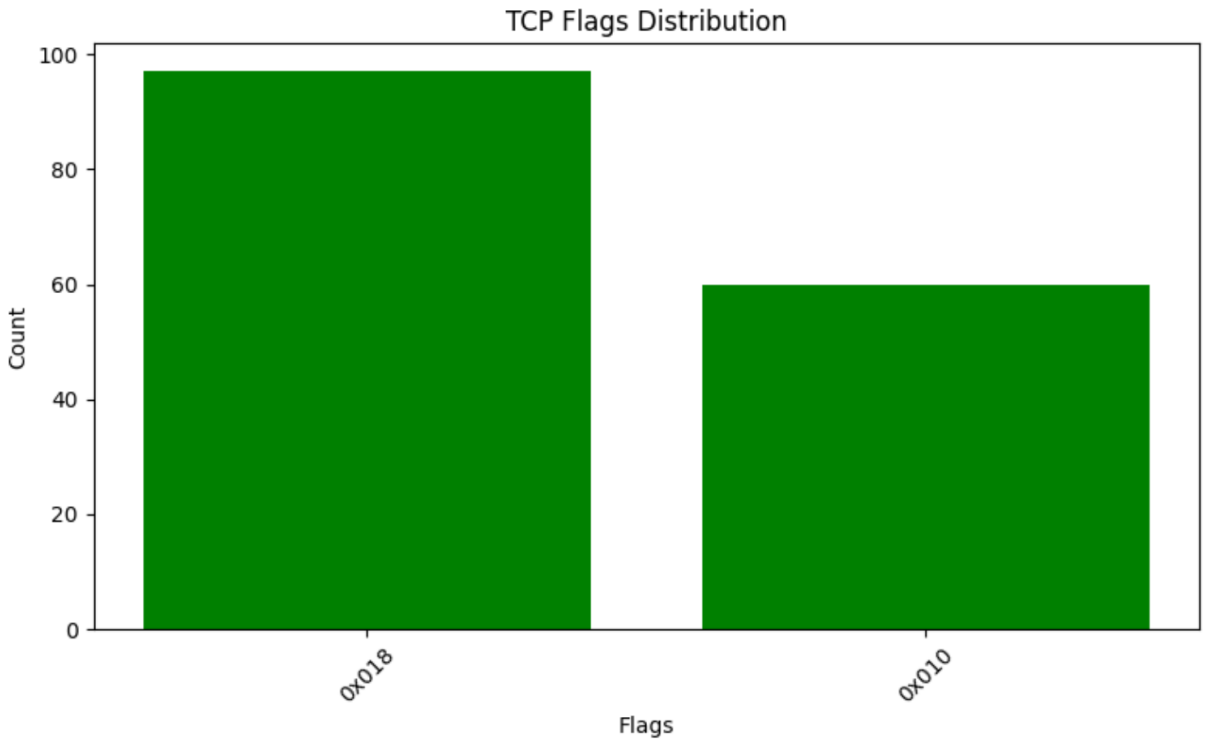
- זהו דגל :PSH + ACK
  - (0x010) ACKמאשר קבלה של נתונים.
  - (0x008) PSHמציין שהנתונים צריכים לעבור מיד (ללא המתנה לחבילות נוספות).
- דגל זה מופיע בכמות הגבוהה ביותר ,כלומר הרבה חבילות שולחות נתונים בזמן אמת.

דגל 0x010

- זהו דגל ACK בלבד.
- בדרך כלל זה אישור קבלת חבילה ,ללא שליחת נתונים נוספים.

רוב החבילות כוללות ,ACK + PSHמה שמעיד על תעבורה אינטראקטיבית)כגון סטרימינג או תקשורת מהירה.
מספר רב של ACK בלבד מראה שחבילות רבות מאושרות, גם אם הן לא מעבירות נתונים חדשות.

להלן הגרף :



- כעת נדון בגרף השלישי : שדות כותרתTLS

```
##############################################################################
# 3) TLS Header Fields Analysis
##############################################################################
if 'TLS Server Name' in df.columns and df['TLS Server Name'].notna().any():
    tls_counts = df['TLS Server Name'].value_counts()
    plt.figure(figsize=(8, 5))
    plt.bar(tls_counts.index[:10], tls_counts.values[:10], color='purple')
    plt.title('Top 10 TLS Server Names')
    plt.xlabel('TLS Server Name')
    plt.ylabel('Count')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
else:
    print("No TLS Server Name data available or column missing.")
```

מצייר גרף עמודות (bar chart) עם 10שרתי ה TLS-הנפוצים ביותר.
כל עמודה מייצגת שרת TLS מסוים ,והגובה שלה מראה כמה פעמים הוא הופיע בהקלטה.

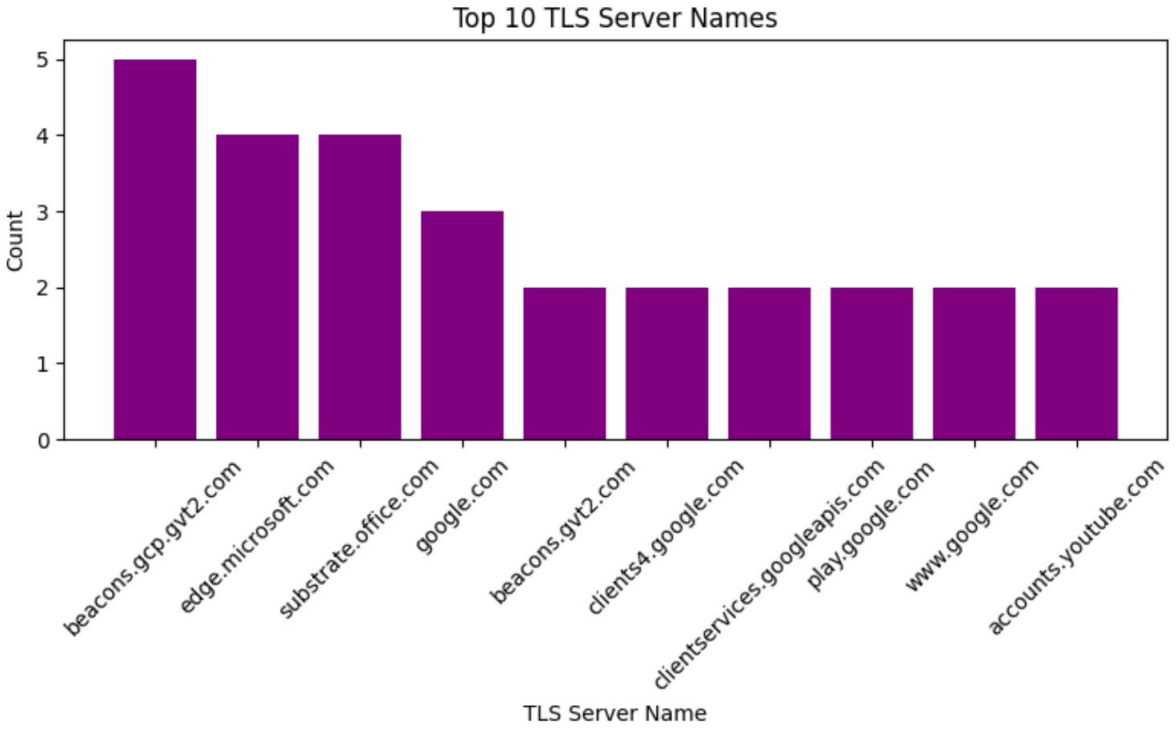
plt.xticks(rotation=45)
מסובב את שמות השרתים בציר ה X-ב-45° כדי שיהיו קריאים.

ציר הX-אופקי – מציג את שמות השרתים שנמצאו בהקלטה.

ציר ה Y-אנכי – מציג כמה פעמים כל שרת TLS הופיע.

כל עמודה מייצגת שרת TLS מסוים ,והגובה שלה מציין כמה פעמים התקיימה תקשורת עם השרת הזה

להלן הגרף :



השרת "beacons.gcp.gvt2.com" הופיע הכי הרבה פעמים

כנראה זהו שרת טלמטריה של Google Cloud Platform (GCP) שמנטר נתוני שימוש.

נוכחות שרתים של(Microsoft (edge.microsoft.com, substrate.office.com
מעידה על תקשורת עם שירותים של ,Microsoft אולי עדכונים או שירותי ענן.

נוכחות של שרתי Google וYouTube-
clients4.google.com, play.google.com, accounts.youtube.com

מצביעים על בקשות אימות, סטרימינג או שירותי גוגל.

- כעת נדון בגרף ה4 : גודלי מנות.

```
##############################################################################
# 4) Packet Size Distribution - Improved Readability
##############################################################################
plt.figure(figsize=(10, 6))
plt.hist(df['Length'], bins=30, color='teal', edgecolor='black', alpha=0.75)
plt.yscale('log')  # Log scale for better visibility of smaller counts
plt.title('Packet Size Distribution')
plt.xlabel('Packet Length (bytes)')
plt.ylabel('Count (log scale)')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

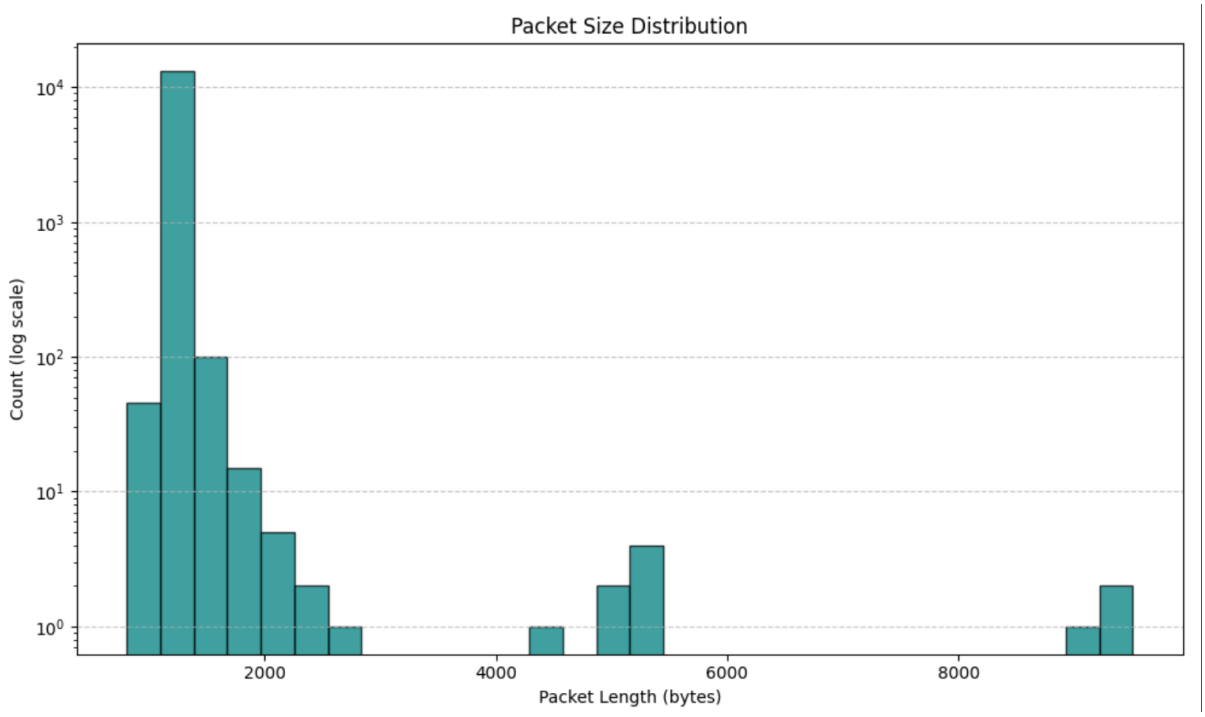
ציר ה X-אופקי – מייצג את אורך החבילות (Packet Length) בבתים.(bytes)

ציר ה Y-אנכי – (מייצג כמה חבילות יש בכל טווח אורך)בסקלה לוגריתמית

כל עמודה מייצגת כמה חבילות יש בגודל מסוים ,למשל:

- אם עמודה גבוהה מאוד הרבה חבילות בגודל זה.
- אם עמודה נמוכה מעט חבילות בגודל זה.

להלן הגרף :



רוב החבילות הן באורך של ~1200-1500 בתים
זהו גודל חבילה אופייני לפרוטוקול QUIC שמשמש את.(YouTube
רשתות מודרניות מתאימות את גודל החבילה ל (Maximum Transmission Unit) MTU-ברשת.

יש מעט חבילות גדולות מאוד (4000-9000 בתים)
חבילות גדולות יכולות להיות נתוני TLS, העברת קבצים, או נתוני סטרימינג מסוימים.
חלקן עשויות להיות חבילות מקוטעות (Fragmented Packets) בגלל מגבלות גודל.

השימוש בסקלה לוגריתמית (log scale) חושף חבילות נדירות
בגרף רגיל, היינו רואים רק את החבילות הנפוצות.

כעת ניתן לראות גם חבילות קטנות או גדולות מאוד ,שלא היינו מבחינים בהן קודם.

- כעת נדון בגרף החמישי : הגעה בין מנות (הזמן בין כל שתי מנות).

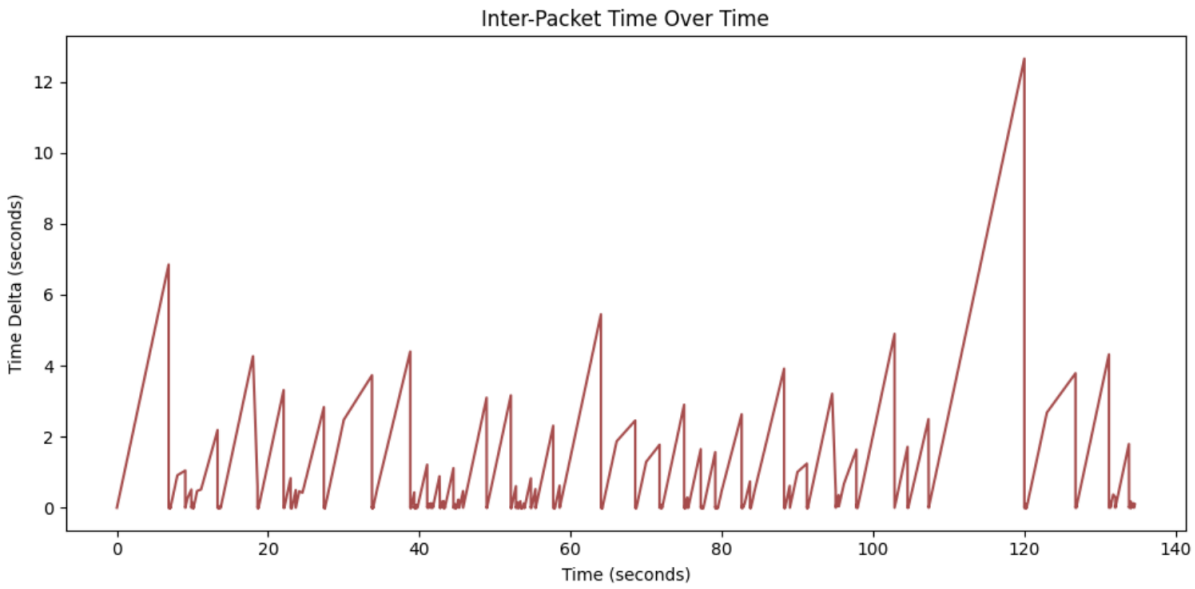
```
##############################################################################
# 5) Inter-Packet Time Differences (Line Plot)
##############################################################################
df['time_delta'] = df['Time'].diff().fillna(0)

plt.figure(figsize=(10, 5))
plt.plot( *args: df['Time'], df['time_delta'], color='maroon', alpha=0.7)
plt.title('Inter-Packet Time Over Time')
plt.xlabel('Time (seconds)')
plt.ylabel('Time Delta (seconds)')
plt.tight_layout()
plt.show()
```

ציר הX-אופקי – מציג את הזמן הכולל שבו התבצע לכידת הרשת (בשניות).

ציר הY- אנכי –מציג את הפרש הזמנים בין חבילה לחבילה הקודמת בכמה שניות.

נקודות גבוהות בגרף –מציינות פערים גדולים בין הגעת חבילה אחת לשנייה.

אזורים עם קווים צפופים ונמוכים –מצביעים על שליחת חבילות בתדירות גבוהה.



חלק גדול מהחבילות מגיעות ברצף עם הפרשי זמן קטנים (קרוב ל-0 שניות)

כאשר אין קפיצות חדות – התקשורת זורמת באופן יציב ורציף.

קפיצות חדות בהפרש הזמן (פיקים בגובה גדול)
מצביעות על הפסקות קצרות בתעבורה ,שיכולות להיגרם מ:

- טעינת וידאו מחדש.
- שיהוי ברשת.
- המתנה לתגובה מהשרת.

בסוף הגרף יש עלייה חדה בהפרש הזמנים
זה עשוי להעיד על סיום ההקלטה ,או שייתכן שהיה שיהוי משמעותי בתקשורת.

כעת נדון בגרף השישי - . גודל זרימה (כלומר, המספר הכולל של מנות).

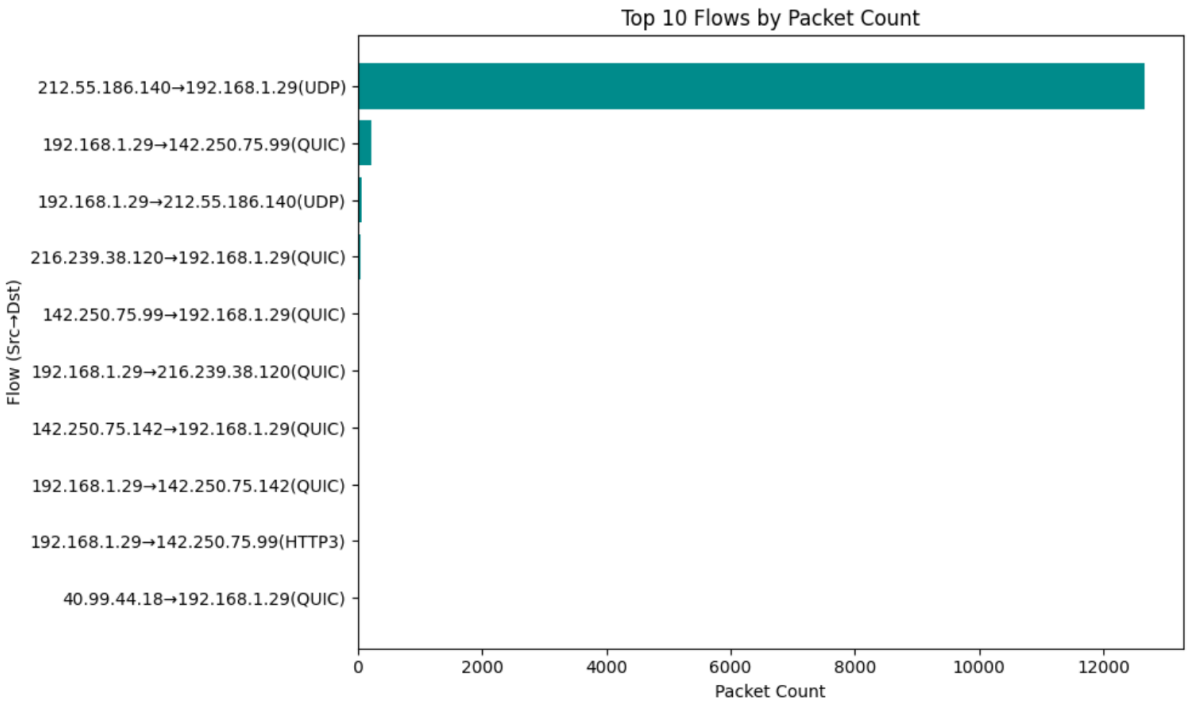
```
##############################################################################
# 6) Flow Size (Number of Packets per Flow) - Improved Readability
##############################################################################
flow_cols = ['Source', 'Destination', 'Protocol']
if set(flow_cols).issubset(df.columns):
    flow_group = df.groupby(flow_cols)
    flow_size = flow_group.size().reset_index(name='packet_count')
    flow_size_sorted = flow_size.sort_values('packet_count', ascending=False).head(10)

    flows = [f"{row['Source']}→{row['Destination']}({row['Protocol']})" for _, row in flow_size_sorted.iterrows()]

    plt.figure(figsize=(10, 6))
    plt.barh(flows, flow_size_sorted['packet_count'], color='darkcyan')
    plt.title('Top 10 Flows by Packet Count')
    plt.xlabel('Packet Count')
    plt.ylabel('Flow (Src→Dst)')
    plt.gca().invert_yaxis()
    plt.tight_layout()
    plt.show()
else:
    print("Flow columns (Source, Destination, Protocol) not available.")
```

ציר הX-אופקי – מציג כמה חבילות עברו בכל זרימה.

ציר ה Y-אנכי – מציג מי שלח למי, ובאיזה פרוטוקול.

עמודות ארוכות יותר זה אומר זרימה עם יותר חבילות.

להלן הגרף :



הזרימה (UDP) 192.168.1.29 → 212.55.186.140היא הדומיננטית
מדובר בשרת ששולח המון חבילות למחשב שלי.
ייתכן שמדובר בשרת סטרימינג או CDN של YouTube שמעביר וידאו דרך UDP.

הזרימות השנייה והשלישית הן (QUIC) 142.250.75.99 → 192.168.1.29
QUIC הוא הפרוטוקול ש YouTube-משתמש בו להזרמת וידאו.
זה אומר ש-המחשב שלי שולח חבילות בקשה לשירות של גוגל .(YouTube)

רוב הזרימות מבוססות על ,UDP/QUIC ורק זרימה אחת משתמשת בHTTP/3-
QUICהוא הפרוטוקול העיקרי, אבל יש גם שימוש מסוים ב.HTTP/3-
YouTube תומך ב HTTP/3-דרך ,QUIC ולכן זה מתאים לתבנית צפויה של סטרימינג וידאו

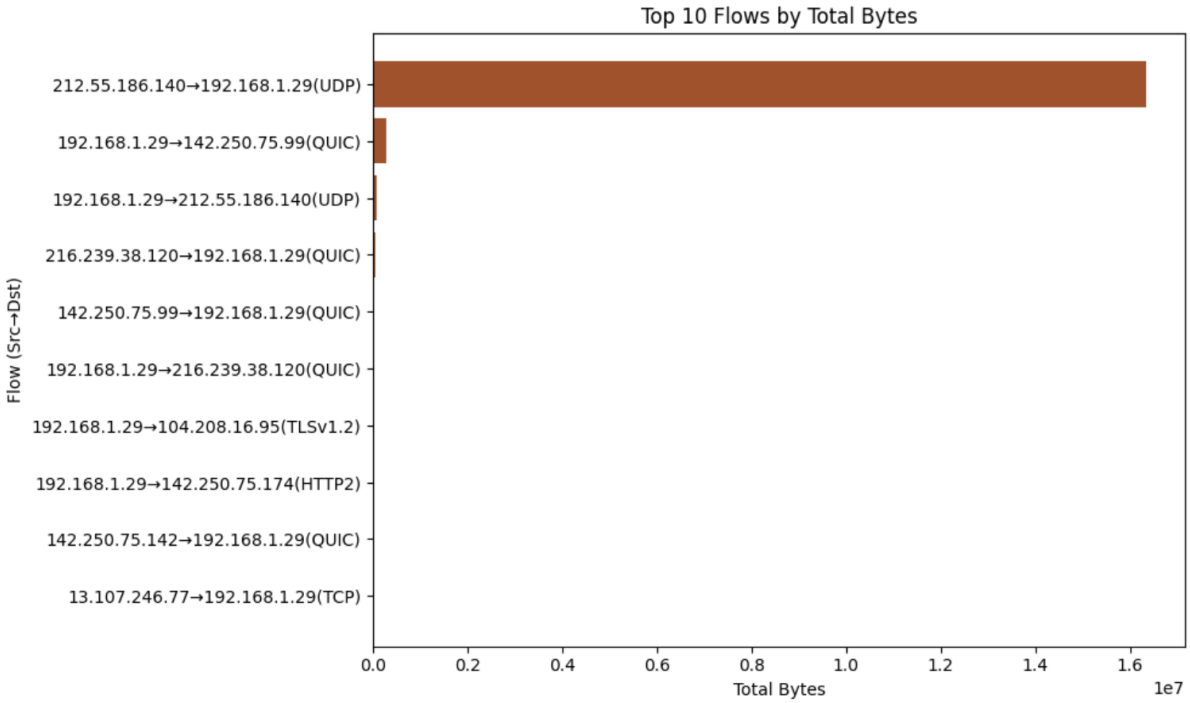
כעת נדון בגרף השביעי : . נפח זרימה (כלומר, המספר הכולל של בתים ששודרו).

```
##############################################################################
# 7) Flow Volume (Total Bytes per Flow)
##############################################################################
if set(flow_cols).issubset(df.columns) and 'Length' in df.columns:
    flow_volume = flow_group['Length'].sum().reset_index(name='total_bytes')
    flow_volume_sorted = flow_volume.sort_values('total_bytes', ascending=False).head(10)
    flows_vol = [f"{row['Source']}→{row['Destination']}({row['Protocol']})" for _, row in flow_volume_sorted.iterrows()]

    plt.figure(figsize=(10, 6))
    plt.barh(flows_vol, flow_volume_sorted['total_bytes'], color='sienna')
    plt.title('Top 10 Flows by Total Bytes')
    plt.xlabel('Total Bytes')
    plt.ylabel('Flow (Src→Dst)')
    plt.gca().invert_yaxis()
    plt.tight_layout()
    plt.show()
else:
    print("Flow volume columns or 'Length' not available.")
```

ציר הX-אופקי –מציג כמות הנתונים הכוללת שהועברה בכל זרימה.(Total Bytes)

ציר הY-אנכי – מציג מי שלח למי ובאיזה פרוטוקול.

עמודות ארוכות יותר זה אומר זרימה שבה עברו יותר נתונים.



הזרימה (UDP) 192.168.1.29 → 212.55.186.140מכילה את רוב הנתונים
מדובר בשרת ששולח כמות עצומה של נתונים למחשב שלי.
מדובר בשרת סטרימינג YouTube שמעביר וידאו דרך UDP.
זהו הזרם הראשי של הסרטון שנשלח אליך.

זרימות מבוססות QUIC מכילות פחות נתונים יחסית
QUIC פרוטוקול שמשתמש ב UDP-מופיע הרבה, אך העברת הנתונים פחותה בהרבה לעומת השרת הראשי. ייתכן שהן משמשות לשליטה וניהול סטרימינג, בעוד שהזרימה הראשית אחראית להעברת הווידאו עצמו.

TLS ו HTTP3-מופיעים, אך בכמויות קטנות יותר
TLSv1.2 מצביע על חיבור מאובטח כנראה לבדיקה ראשונית של הסרטון.
HTTP3גם מעיד על העברת תוכן דינמי למשל, טעינת עמודי YouTube עצמם

סעיף 4 :

נדרש מאיתנו להקליט שוב את תעבורת רשת שלנו , בהנחה שאנחנו התוקף ואנחנו מנסים ללמוד אילו אפליקציות המשתמש ניגש אליהם .

מטרת הבדיקה היא לנתח עד כמה ניתן לזהות אתרים ואפליקציות שבהם השתמש המשתמש, גם כאשר התעבורה מוצפנת. נבדוק שני תרחישים:

5. התקפה עם – Flow ID כאשר התוקף יודע את גודל החבילות, חותמות הזמן, ואת מזהה הזרם (Flow ID)שמבוסס על כתובות ה IP-והפורט.
6. התקפה ללא – Flow ID כאשר התוקף יודע רק את גודל החבילות וחותמות הזמן.

על מנת להשתיק רעשי רקע כיבנו את חומת האש הפעלנו את ההקלטה שוב על שיר מסוים בעל וידאו ביוטיוב וביצענו סינון .

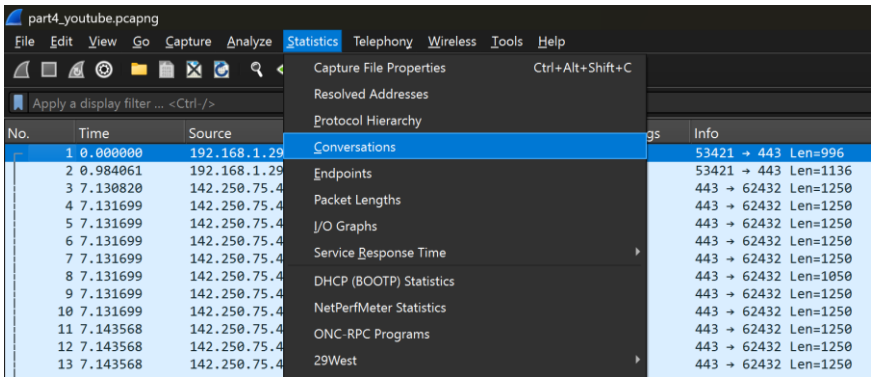
הארה : ההקלטה שנצטרף תיהיה הקטלה לאחר סינון כי זה מלא פאקטות ונקרא לה בשם part4_youtube

- כעת נתעסק בתרחיש הראשון כלומר התקפה עם– Flow ID .

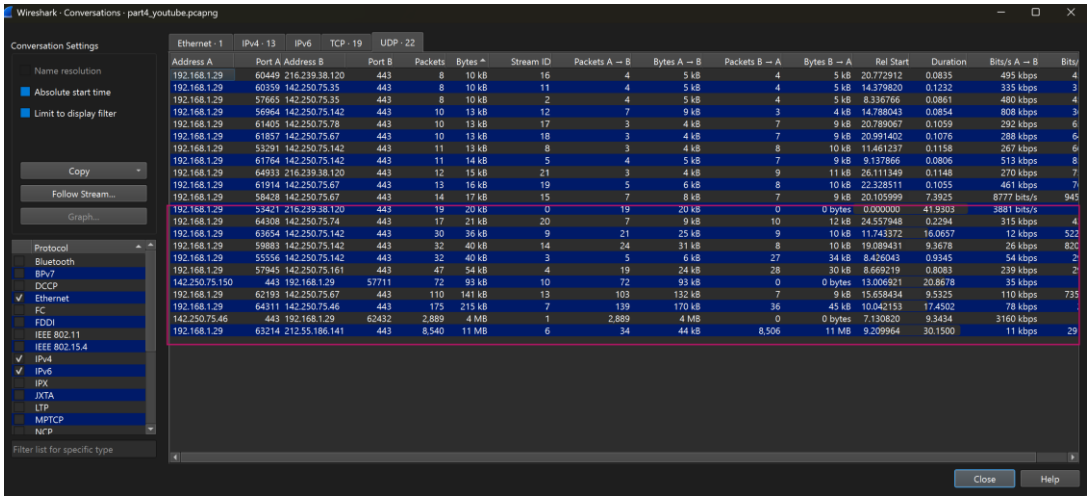
שלב 1: זיהוי חיבורים עם זרימה גבוהה(Flow ID)

נרצה לזהות חיבורים עם כמות נתונים גבוהה ,המעידה על סטרימינג ביוטיוב בנוסף נרצה לאתר חיבורים עם TCP Port 443 (HTTPS)או UDP (QUIC) וכמו כן לבדוק כתובות IP של Google המשויכות ל-YouTube

ניגש בWireshark :



ניגש לUDP ונמיין לפי Bytes כך שנוכל ראות את החיבורים שבהם הועברו כמויות גדולות של נתונים .אתרי חיבור עם מספר חבילות גדול מאוד (מאות או אלפים), המעיד על זרימת מדיה.



מכאן נסיק כי התעבורה עוברת בכמויות גדולות ומגיעה מכתובות ,Google זה סימן ברור לשירות סטרימינג כמו.YouTube

התוקף יכול להשתמש בזה בכך שהוא יכול לזהות חיבורי נתונים גדולים מאוד במיוחד דרכHTTPS/QUIC כלומר

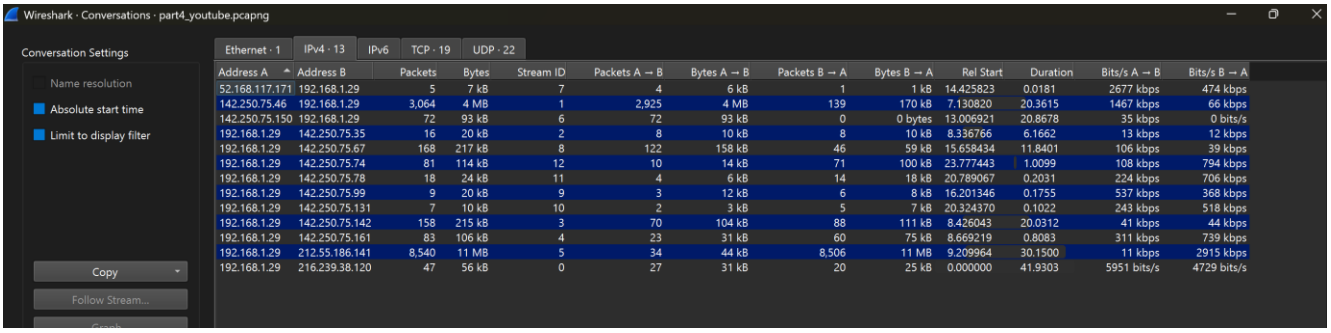
סטרימינג ביוטיוב דורש נפח נתונים גבוה -הרבה יותר מאתר חדשות רגיל או ויקיפדיה.
בתמונה ניתן לראות חיבורים שבהם עברו 4 MBאו 11 MBשל נתונים ,שהם סימן ברור לתוכן מדיה (כמו וידאו).

לכן אם התוקף מזהה מספר חיבורים גדולים בפרק זמן קצר, יש סבירות גבוהה שהמשתמש צפה בווידאו.
בנוסף אם תוקף רואה שרוב התעבורה הכבדה מגיעה מכתובות ,Google זה רמז חזק לסטרימינג ביוטיוב.

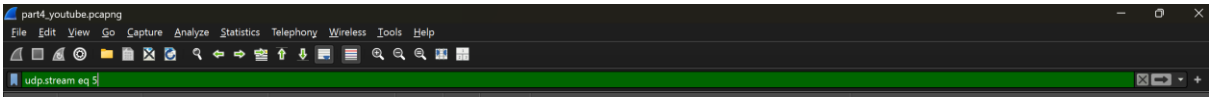
כמו כן סטרימינג יוצר חיבור רציף וארוך-טווח בין המשתמש לשרת.
בתמונה יש חיבורים שנמשכים למעלה מ-30 שניות ,שזה משך זמן טיפוסי לטעינת וידאו ביוטיוב.
כלומר אם תוקף מזהה חיבורי נתונים ארוכים עם גודל משמעותי – יש סיכוי גבוה שזה וידאו.

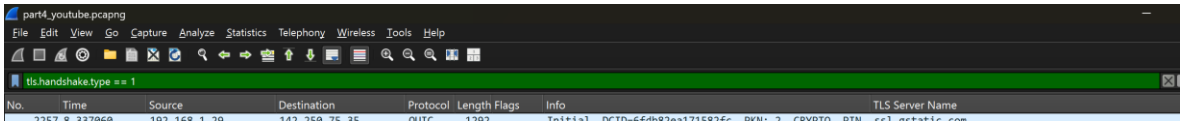
חשוב לציין כי ניתן לראות שיש חשיפה מלאה לפרוטוקולים שיוצרים איתם חיבור במהלך היקשרות לאתר מאובטח , כלומר ניתן לראות זרמים נפרדים עם כתובות IP שונות זה מוכיח שהתוקף יכול לזהות לאילו שרתים המשתמש התחבר למצוא דרכם נקודות חולשה ולנסות לתקוף משם .

התוקף יכול להיות חשוף גם לכל הכתובות הip ופשוט לבדוק ממאגרים ואתרים שונים לאן הם שייכים .

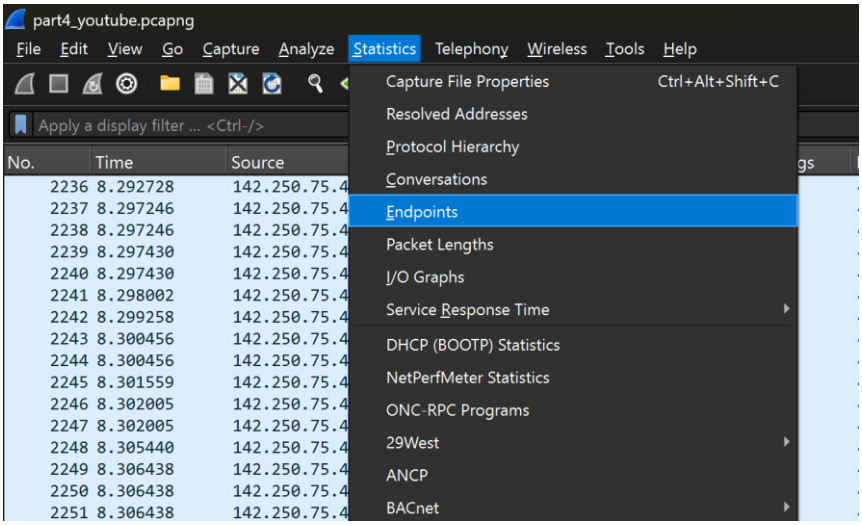


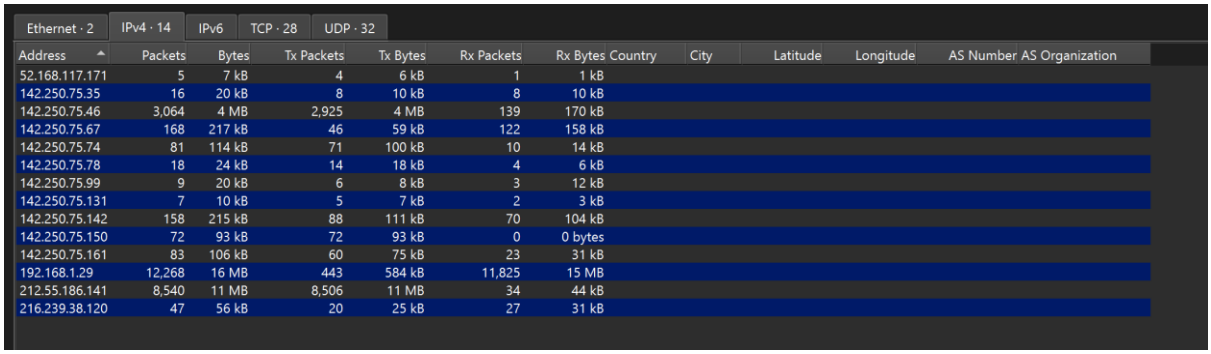
אפשר להשתמש בחיבורים שונים על מנת לראות את השרתים שהיה בהם שימוש לדוגמא





שלב שני : סינון לפי כתובות IP של שרתי YouTube .

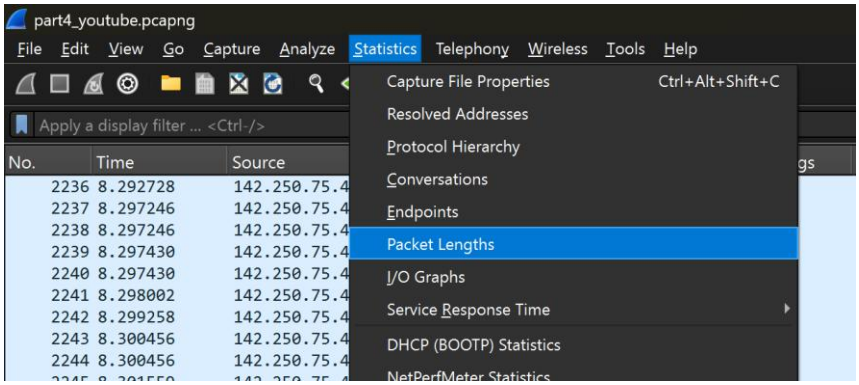


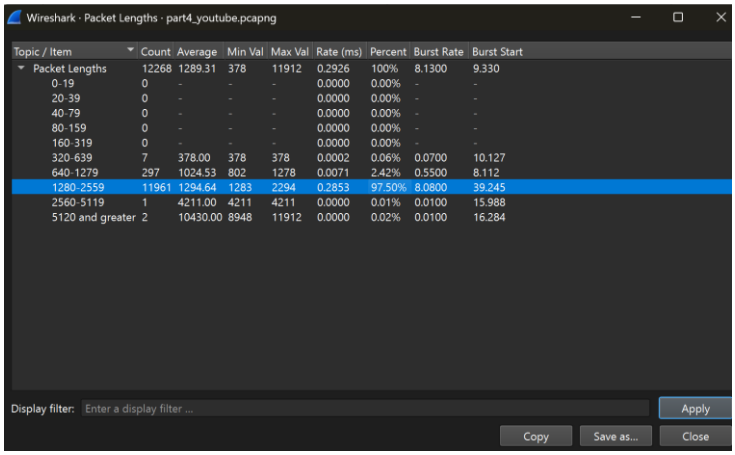


Address	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes	Country	City	Latitude	Longitude	AS Number	AS Organization
52.168.117.171	5	7 kB	4	6 kB	1	1 kB						
142.250.75.35	16	20 kB	8	10 kB	8	10 kB						
142.250.75.46	3,064	4 MB	2,925	4 MB	139	170 kB						
142.250.75.67	168	217 kB	46	59 kB	122	158 kB						
142.250.75.74	81	114 kB	71	100 kB	10	14 kB						
142.250.75.78	18	24 kB	14	18 kB	4	6 kB						
142.250.75.99	9	20 kB	6	8 kB	3	12 kB						
142.250.75.131	7	10 kB	5	7 kB	2	3 kB						
142.250.75.142	158	215 kB	88	111 kB	70	104 kB						
142.250.75.150	72	93 kB	72	93 kB	0	0 bytes						
142.250.75.161	83	106 kB	60	75 kB	23	31 kB						
192.168.1.29	12,268	16 MB	443	584 kB	11,825	15 MB						
212.55.186.141	8,540	11 MB	8,506	11 MB	34	44 kB						
216.239.38.120	47	56 kB	20	25 kB	27	31 kB						

תוקף שרואה כתובות IP השייכות לשרתי Google Video CDN יכול לדעת שהתוכן הגיע מיוטיוב.
אם החיבור נפח גבוה + כתובת היעד היא ,Google זה מחזק את ההשערה שזהו סרטון מיוטיוב.
כלומר זה מאפשר לתוקף לקבוע בסבירות גבוהה שהמשתמש השתמש ביוטיוב, גם בלי לראות את ה -URL של הסרטון

שלב שלישי: זיהוי גודל חבילות גדול





סרטוני YouTube נשלחים בחבילות גדולות במיוחד (כמעט תמיד 1500 בייטים).
כלומר אם תוקף מזהה זרימה רציפה של חבילות בגודל 1500 בייטים ,זה רמז חזק להזרמת וידאו .אתרים רגילים (למשל ויקיפדיה) משתמשים בחבילות קטנות יותר ,ולכן תוקף יכול להבחין בין סטרימינג לדפדוף רגיל.

בתרחיש זה בדקנו האם תוקף יכול לזהות שהמשתמש צפה בתוכן וידאו ביוטיוב, בהינתן שיש לו מידע על גודל החבילות, חותמות הזמן ו 4-Tupleשל כתובת מקור, כתובת יעד, פורט מקור ופורט יעד.

הממצאים העיקריים:
זוהתה תעבורה בהיקף גדול מאוד (MBs) רבים (בפרוטוקול HTTPS/TLS או ,QUIC המעידה על סטרימינג וידאו.
כתובות ה IP-של שרתי היעד שייכות ל ,Google Video CDN-המאחסנים תוכני YouTube
חבילות גדולות שנשלחות ברצף, תואמות להזרמת מדיה.
פרוטוקולי QUIC ו 443 TCP-שולטים בתעבורה, מאשרים שהשירות הוא יוטיוב.

תוקף יכול לזהות ברמת דיוק גבוהה שהמשתמש צפה בתוכן ביוטיוב, אך לא יכול לדעת איזה סרטון בדיוק.

עד כמה התוקף יכול לזהות את האפליקציות או האתרים שהמשתמש השתמש בהם?

התוקף יכול לזהות באיזה אתר או שירות המשתמש השתמש, גם אם התוכן מוצפן, על ידי ניתוח דפוסי תעבורה. התוקף השתמש בכלים שונים כלומר באמצעותFlow ID התוקף יכול לקבץ חבילות לאותו חיבור ולהבין כמה נתונים הועברו, באיזו תדירות ואיזה פרוטוקול שימש .כאשר תעבורה רבה מגיעה מכתובות IP השייכות ל ,Google-ניתן להסיק שהמשתמש השתמש בשירות כמו YouTube.
בנוסף אם החבילות הן בגודל 1500 בייתים לפחות ונשלחות ברצף, זה סימן חזק לכך שמדובר בהזרמת וידאו. למרות שהתוכן מוצפן, דפוסי השימוש מספיקים כדי לזהות את סוג הפעילות (לדוגמה, סטרימינג, גלישה באינטרנט, משחקים וכו').
לסיכום התוקף לא יכול לדעת בדיוק איזה סרטון המשתמש צפה בו, אך יכול לזהות שהיה שימוש ביוטיוב.

איך נמנע התקפה מסוג זה ?

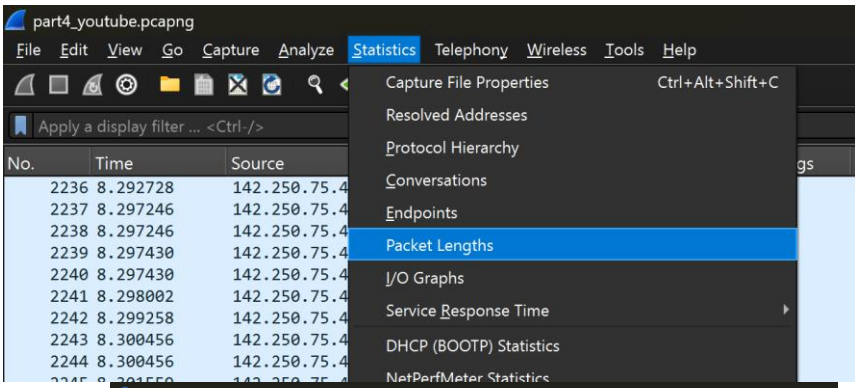
כדי להקשות על תוקף לזהות שימוש ביוטיוב או שירותים אחרים, ניתן להשתמש בטכניקות הבאות:

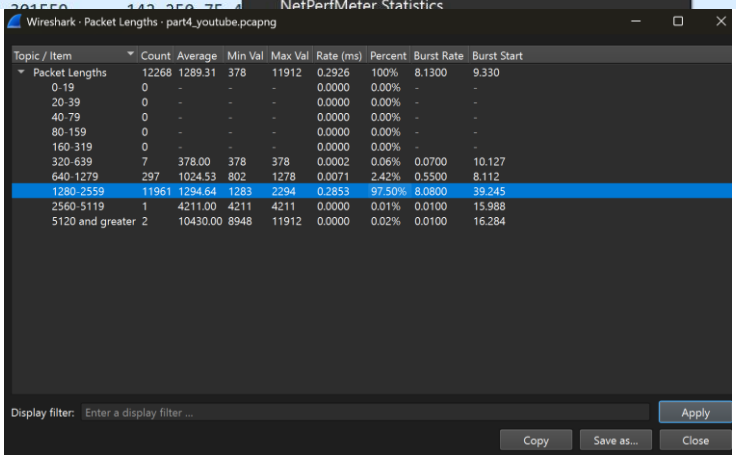
VPN או – Tor מרחיבים את האנונימיות על ידי הצפנת כל התעבורה והסתרת כתובות ה IP-של היעד.
Mix Networks מערבלים תעבורה עם חיבורים אחרים כדי לטשטש את דפוסי התעבורה של משתמש בודד.
Padding (רעש תעבורתי)– שליחת חבילות ריקות או באורך משתנה כדי לטשטש דפוסי גודל חבילות המאפיינים סטרימינג.
DNS over HTTPS (DoH) מסתיר את שאילתות ה DNS-מה שמונע זיהוי שמות האתרים הנצפים.
Traffic Obfuscation (ערפול תעבורה) - כלים כמו Obfsproxyמאפשרים להסתיר את מבנה התעבורה כך שלא יהיה ניתן להסיק ממנה מסקנות.
גלישה ברשתות ציבוריות –שימוש ברשתות Wi-Fi חינמיות יכול לטשטש את מקור התעבורה ולמנוע זיהוי של המשתמש עצמו.

תרחיש שני - כאשר התוקף יודע רק את גודל החבילות וחותמות הזמן.

הארה: נשתמש באותה הקלטה מהתרחיש הראשון ,אבל ננתח אותה בצורה שונה כדי לדמות מצב שבו לתוקף אין מידע על Flow ID, אלא רק גודל חבילות וחותמות זמן.

שלב ראשון : נראה שתעבורת וידאו מאופיינת בחבילות גדולות שנשלחות ברצף.





סרטוני YouTube מועברים בחבילות גדולות של 1500 בייטים.
אם התוקף רואה הרבה חבילות בגודל זה, הוא יכול לשער שמדובר בהזרמת תוכן מדיה.

שלב 2: ניתוח חותמות זמן(Inter-Packet Timing)

המטרה :להראות שסטרימינג מאופיין בדפוסי תזמון אחידים יחסית.

סטרימינג של וידאו מאופיין בגרף יציב עם מעט תנודות.
אם יש דפוס חזרתי של עליות פתאומיות , התוקף יכול לנחש שזהו וידאו.
לעומת זאת, גלישה רגילה מייצרת תעבורה לא יציבה, עם הפסקות ארוכות בין שליחת חבילות.

זה יעזור לתוקף בכך ש שירותי סטרימינג מאופיינים בקצב נתונים גבוה ויציב לאורך זמן.
כלומר אם רואים קצב נתונים של מאות Kbps או Mbps, התוקף יכול לשער שמדובר בוידאו.
לעומת זאת, דפדוף באתר רגיל יהיה עם קפיצות בודדות ולא זרימה אחידה.

בתרחיש זה בדקנו האם תוקף יכול לזהות שהמשתמש צפה בסרטון ביוטיוב, כאשר יש לו גישה רק ל:

- גודל החבילות
- חותמות הזמן של כל חבילה

הממצאים שמצאנו לאחר ניתוח ההקלטה :
נפח תעבורה גדול וממושך עם קצב אחיד, אופייני להזרמת וידאו.
ריבוי חבילות גדולות , המעידות על הורדת מדיה באיכות גבוהה.
גרף זרימת הנתונים (IO Graphs) מציג דפוס טעינה ראשונית ולאחר מכן קצב יציב, דבר אופייני לסטרימינג.
קצב הנתונים (Bits/sec) גבוה משמעותית לעומת גלישה רגילה באתרים טקסטואליים.

התוקף יכול לזהות שהמשתמש צפה בסטרימינג וידאו, אך לא יכול לדעת בוודאות שהשירות היהYouTube .
בהשוואה לתרחיש הראשון, ללא כתובות IP או Flow ID, קשה יותר לזהות את השירות הספציפי, אך הדפוסים עדיין מצביעים על שימוש בסטרימינג.

מכאן נסיק כי תוקף עם גישה רק לגודל חבילות וחותמות זמן עדיין יכול להסיק מידע משמעותי על פעילות המשתמש .בניתוח ההקלטה והבחנה של ניתוח גודל החבילות ודפוסי התעבורה, ניתן לזהות אם המשתמש גלש באתר מבוסס טקסט (כמו ויקיפדיה) או צפה בווידאו.
במקרה של סטרימינג, קל לזהות חבילות גדולות ודפוסי תעבורה אחידים, המעידים על הזרמת מדיה .עם זאת, ללא מידע על כתובות IP או Flow ID, קשה לקבוע בוודאות שהשירות היה יוטיוב ולא פלטפורמת וידאו אחרת. מפה ניתן להסיק שתוקף יכול לזהות את סוג הפעילות (גלישה, סטרימינג, הורדה), אך לא בהכרח את האתר או האפליקציה הספציפית.

נמנע מתקפה זו כלומר על מנת למנוע תוקפים מלזהות פעילות מבוססת דפוסי חבילות ותזמון, ניתן להשתמש בטכניקות הבאות:

1. Traffic Obfuscation שימוש בפרוטוקולים שמסתירים את דפוסי החבילות, כמו Obfsproxy או Pluggable Transports .
2. VPN או Tor מקשים על תוקף לנתח דפוסים בכך שהם מצפינים את כל החיבורים ומנתבים אותם דרך שרתים מתווכים.
3. Randomized Packet Padding שליחת חבילות עם אורכים משתנים כדי למנוע תבניות חזרתיות שניתן לנתח.
4. ריבוי משימות – (Multiplexing) שימוש בכמה אפליקציות במקביל (למשל סטרימינג ודפדוף באינטרנט) כדי לטשטש דפוסים מובהקים.
5. הזרמת תוכן באיכות משתנה – (Adaptive Bitrate Streaming) טכנולוגיות כמו DASH של יוטיוב משנות את איכות הווידאו בהתאם לרוחב הפס, מה שיכול להפוך דפוסי תעבורה לפחות צפויים.

הזרמת וידאו + אודיו – זום

הקדמה:

נבצע ניתוח תעבורת רשת שנאספה במהלך שימוש באפליקציית .Zoom מטרת העבודה היא לנתח את מאפייני התעבורה של ,Zoom להבין את התנהגות התעבורה, ולספק הסברים סטטיסטיים וגרפיים על זמני התגובה, גודל החבילות, ועוד. הניתוח בוצע בעזרת Wireshark וכלים נוספים לביצוע אנליזות על נתוני התעבורה

הסבר על המידע המוצפן והפילטרים שהשתמשתי בהם:

במהלך העבודה השתמשתי בנתונים מוצפנים של תעבורת ,Zoom המועברים באמצעות פרוטוקול TLS (Transport Layer Security)מכיוון ש Zoom-משתמשת בהצפנה זו, לא ניתן לראות את התוכן הגולמי של התעבורה בשלב הראשון של ההקלטה. כדי לפענח את המידע המוצפן ולחשוף את התעבורה, השתמשתי בכמה טכניקות וכלים על מערכת לינוקס.

סינון התעבורה

לאחר ש Wireshark-הצליח לפענח את המידע המוצפן, השתמשתי במספר פילטרים כדי לסנן את הנתונים ולמקד את התעבורה רק ל.Zoom-

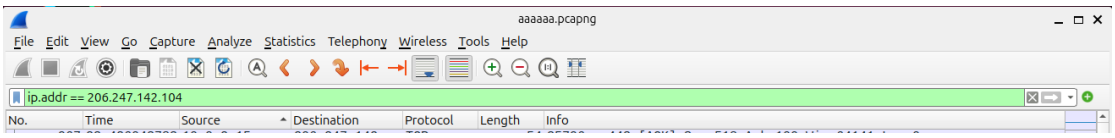
1. **סינון לפי כתובת :IP** כדי לסנן את התעבורה כך שתכיל רק את התעבורה של ,Zoom השתמשתי בכתובת ה IP-של השרתים של .Zoom תוכל להשתמש בסינון הבא ב-Wireshark

כאן תחילה מצאתי את הכתובת של זום על ידי הפקודה בטרמינל

```
galash@Gal:~$ nslookup 206.247.142.104
104.142.247.206.in-addr.arpa    name = 206-247-142.zoom.us.

Authoritative answers can be found from:
```

ולאחר מכן השתמשנו בפילטר :

```
ip.addr == 206.247.142.104
```

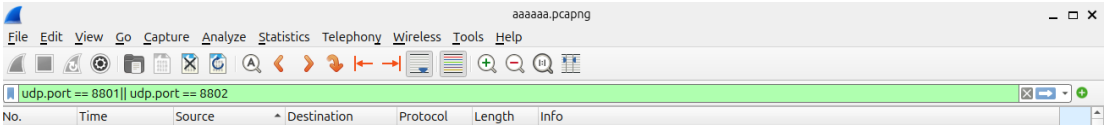


סינון לפי פרוטוקול :TLS מכיוון ש Zoom-משתמשת בהצפנה עם פרוטוקול ,TLS השתמשתי גם בסינון לפי שם השרת של Zoom

```
tls.handshake.extensions_server_name contains "zoom"
```

סינון לפי פורטים :בנוסף לכך, ביצעתי סינון לפי הפורטים בהם Zoom משתמשת להעברת נתונים. למשל:

```
udp.port == 8801 || udp.port == 8802
```

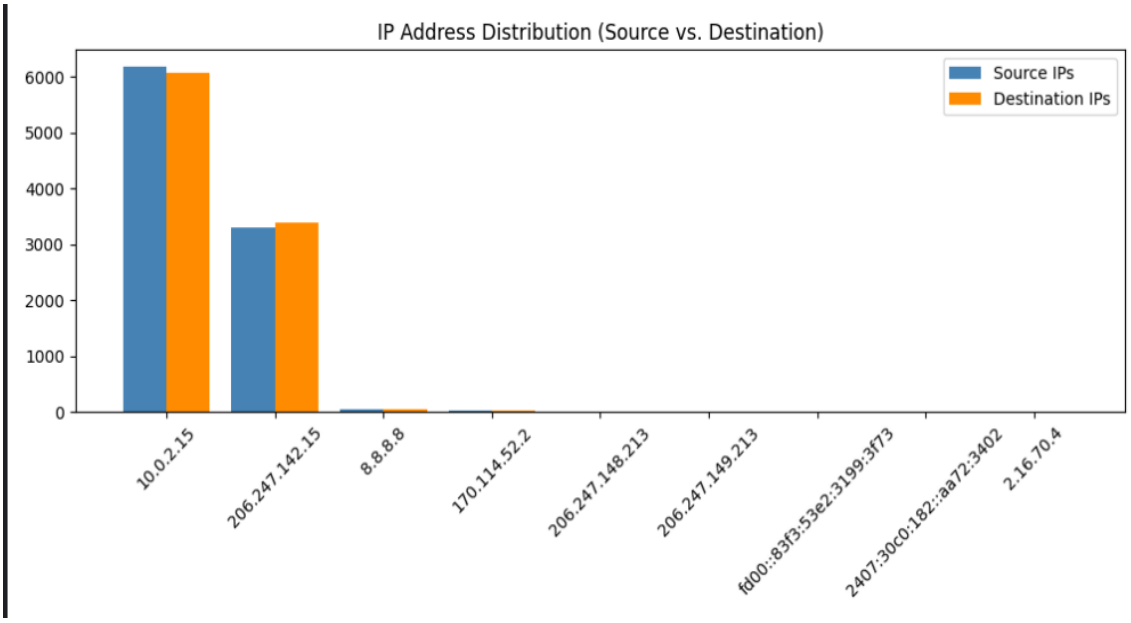


סיכום

באמצעות הכלים והפילטרים שהגדרתי, הצלחתי לפענח את המידע המוצפן בתעבורת Zoom ולסנן את הנתונים כך שיתמקדו רק בתעבורה המיועדת ל .Zoom-שימוש במפתחות TLS יחד עם סינונים מתאימים מאפשר לי לקבל תמונה ברורה על התנהגות התעבורה של האפליקציה בזמן אמת, ולנתח את הנתונים בצורה מדויקת.

IP Address Distribution (Source vs. Destination)

- מה הוא עושה :גרף זה מציג את ההתפלגות של כתובת ה IP-שמייצגת את המידע בתעבורה, ומבצע השוואה בין כתובת ה IP-ששימשה כמקור (Source IP) ובין כתובת ה IP-ששימשה כיעד .(Destination IP) הוא מציג את כמות הפעמים שכל כתובת IP פועלת כמקור או כיעד.
- איך הוא נבדק:
  - משתמשים בעמודות 'Source' ו'Destination'-ב DataFrame - עבור כל אחת מהעמודות, סופרים את כמות ההופעות של כתובת ה IP-השונה.
  - עבור כל כתובת ,IP הוא מציג את כמות הפעמים שהיא שימשה כמקור (Source) או כיעד ,(Destination) ויוצר גרף ברים עם צבעים שונים עבור כל אחד מהפרמטרים.
- מה הוא מציג:
  - ציר ה :X- כתובת ה .IP -
  - ציר ה :Y-מספר הפעמים שבהן כתובת ה IP-הופיעה כמקור או כיעד.
  - הברות הכחול והכתום מייצגות את כתובת ה IP-כמקור ויעד בהתאמה.



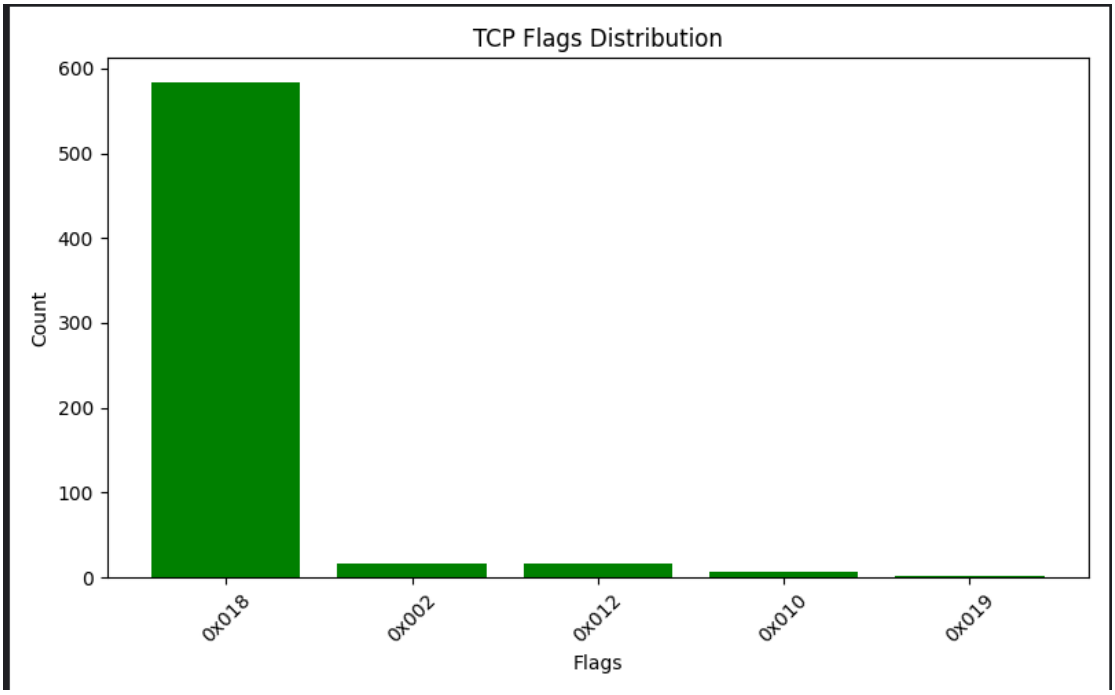
TCP Flags Distribution

מה גרף זה מציג :גרף זה מציג את ההתפלגות של דגלי ה TCP-בתעבורה. דגלי ה TCP-כגון SYN, (ACK, FIN)חשובים להערכת פעולות שנעשו בתעבורת ה TCP, כולל התחלת חיבור, הכרה, וסיום חיבור.

מה נבדק :הגרף סופר את כמות ההופעות של כל דגל בתעבורת ה TCP, תוך שימוש בעמודת 'Flags'ב DataFrame. כל דגל מציין פעולה מסוימת ב TCP, ונתוני ה TCP-מפולחים לפי ערכים שמייצגים דגלים שונים למשל (0x18, 0x02).

הסבר לגרף:

- ציר ה :X-ערכים המייצגים את הדגלים למשל : (0x18, 0x02)
- ציר ה :Y-מספר הפעמים שהדגל הופיע בתעבורה.
- כל בר מייצג את כמות ההופעות של דגל TCP מסוים.



TLS Server Name Distribution

מה גרף זה מציג :גרף זה מציג את ההתפלגות של שמות השרתים בתעבורת .TLS שמות השרתים בתעבורת TLS מציינים את השרת שאליו נעשה חיבור מוצפן, והשדות האלו יכולים להיות חשובים כשמנתחים את ההתנהגות של האפליקציה.

מה נבדק :הגרף סופר את ההופעות של שמות השרתים בתעבורת TLS בעמודת 'TLS Server Name' ב.DataFrame-כל שם שרת מייצג את השרת שאליו התחברו בחיבור מוצפן, ומניחים שהשמות השרתים שמוודאים את השרת שמטפל בחיבור (כגון .zoom.us)

הסבר לגרף:

- ציר ה :X-שמות השרתים בתעבורת.TLS
- ציר ה :Y-מספר הפעמים ששמות השרתים הללו הופיעו בתעבורה

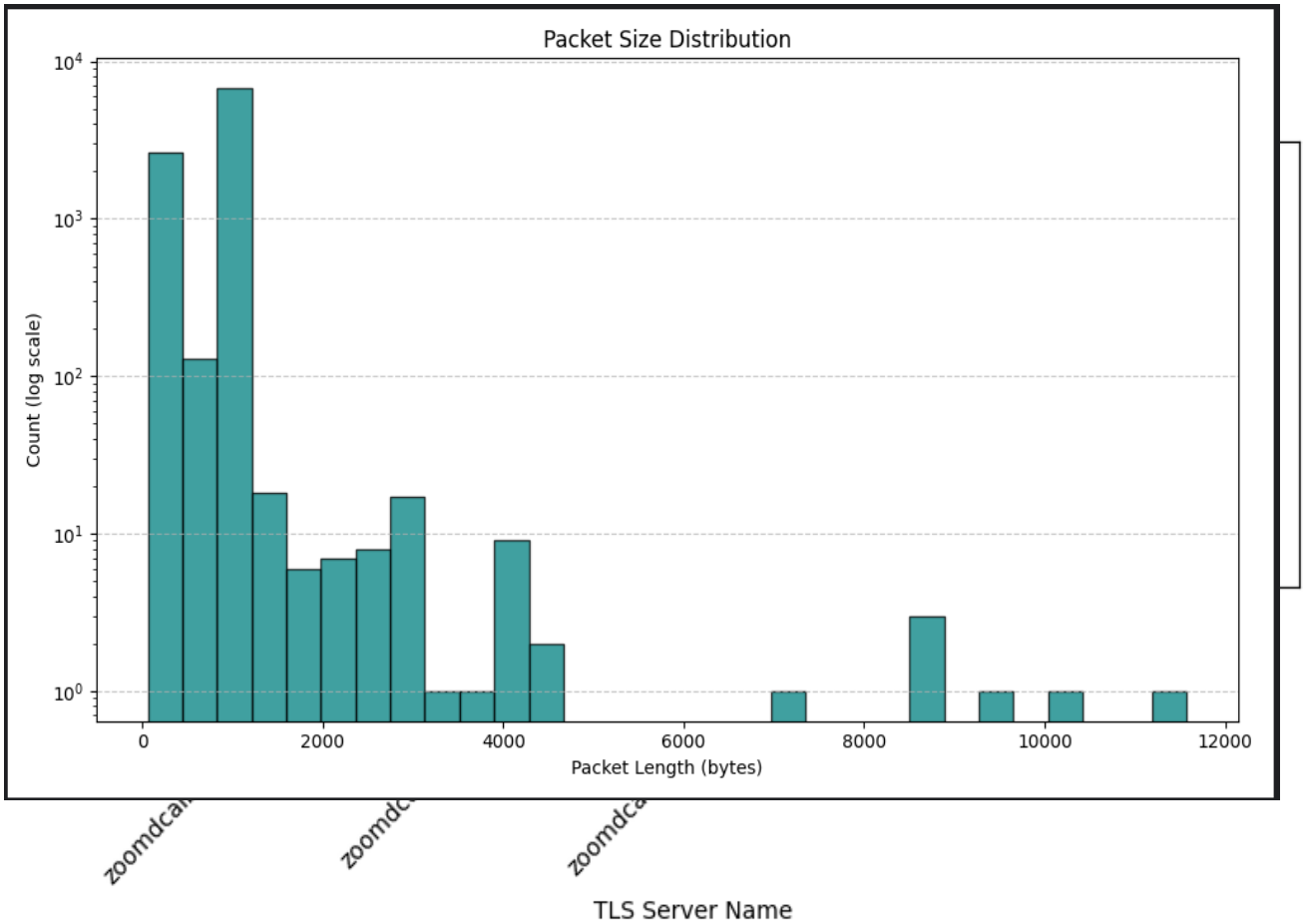
Packet Size Distribution

מה גרף זה מציג :גרף זה מציג את ההתפלגות של גודל החבילות בתעבורה. גודל החבילה הוא פרמטר חשוב כדי להבין את אופי התעבורה ברשת, והוא מראה את כמות המידע שנשלח בכל חיבור.

מה נבדק :בעמודת 'Length' ב DataFrame-נמצאים גדלי החבילות (בבייטים). הגרף מציג היסטוגרמה של גדלי החבילות, כאשר משתמשים בסקלת לוגריתם להבהרת התפלגות התדירויות הנמוכות יותר.

הסבר לגרף:

- ציר ה :X-גודל החבילה (בבייטים).
- ציר ה :Y-מספר הפעמים שהחבילה בגודל מסוים הופיעה בתעבורה.



- המטרה היא להציג את ההתפלגות של גודל החבילות בתעבורה ברשת.

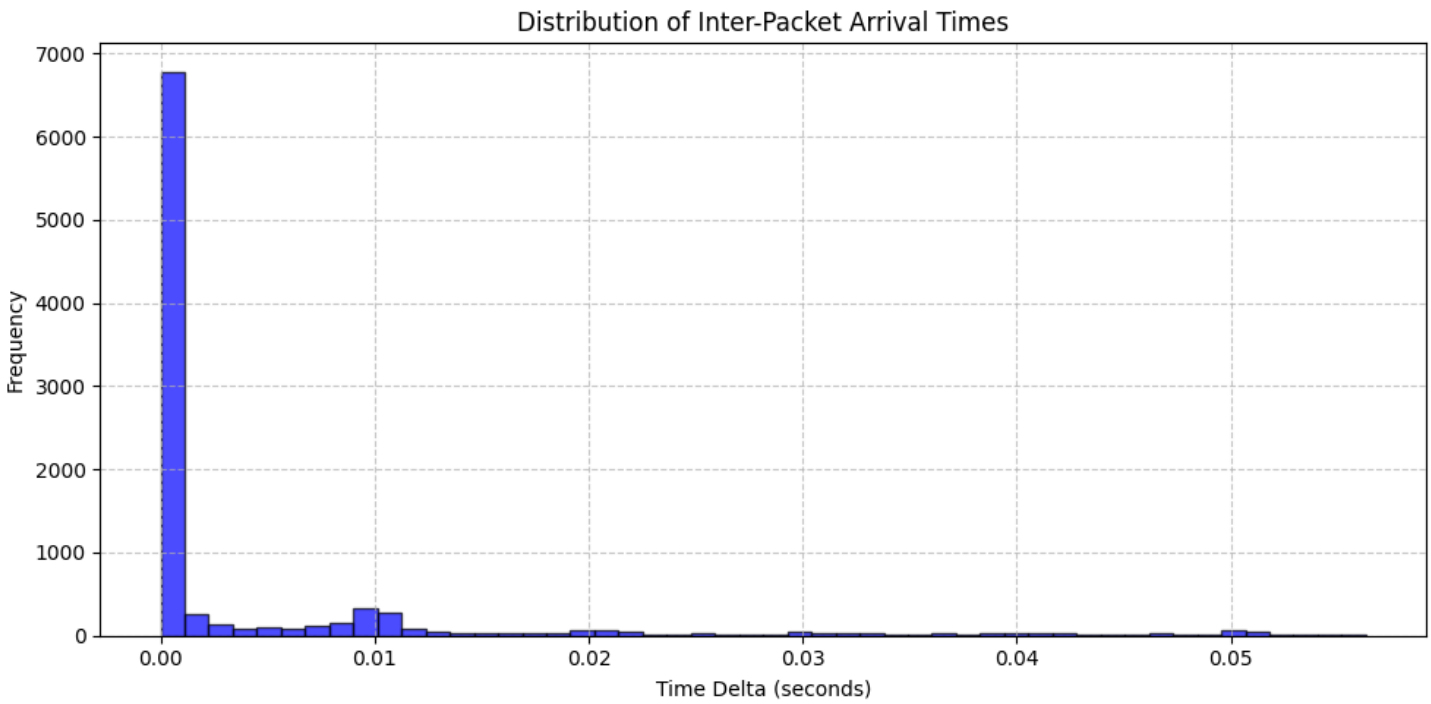
Inter-Packet Time Differences

מה גרף זה מציג :גרף זה מציג את ההתפלגות של זמני ההגעה בין חבילות עוקבות בתעבורה.
המטרה היא להבין כמה זמן עבר בין כל חבילה לחבילה, כלומר את הבדל הזמן בין כל שתי חבילות בתעבורה.

מה נבדק :חישוב ההבדל בזמן בין כל חבילה לחבילה מבוצע באמצעות df['Time'].diff().לאחר מכן, מסננים את הזמנים החזקים ביותר (כדי להתמקד בהבדלים הקטנים) ומבצעים היסטוגרמה.

הסבר לגרף:

- ציר ה :X-זמן ההגעה בין חבילות) ב.(seconds-
- ציר ה :Y-מספר הפעמים שבהם זמן זה הופיע בתעבורה.
- הגרף נותן תמונה ברורה של זמני הגעת החבילות.



Flow Size Distribution (Number of Packets per Flow)

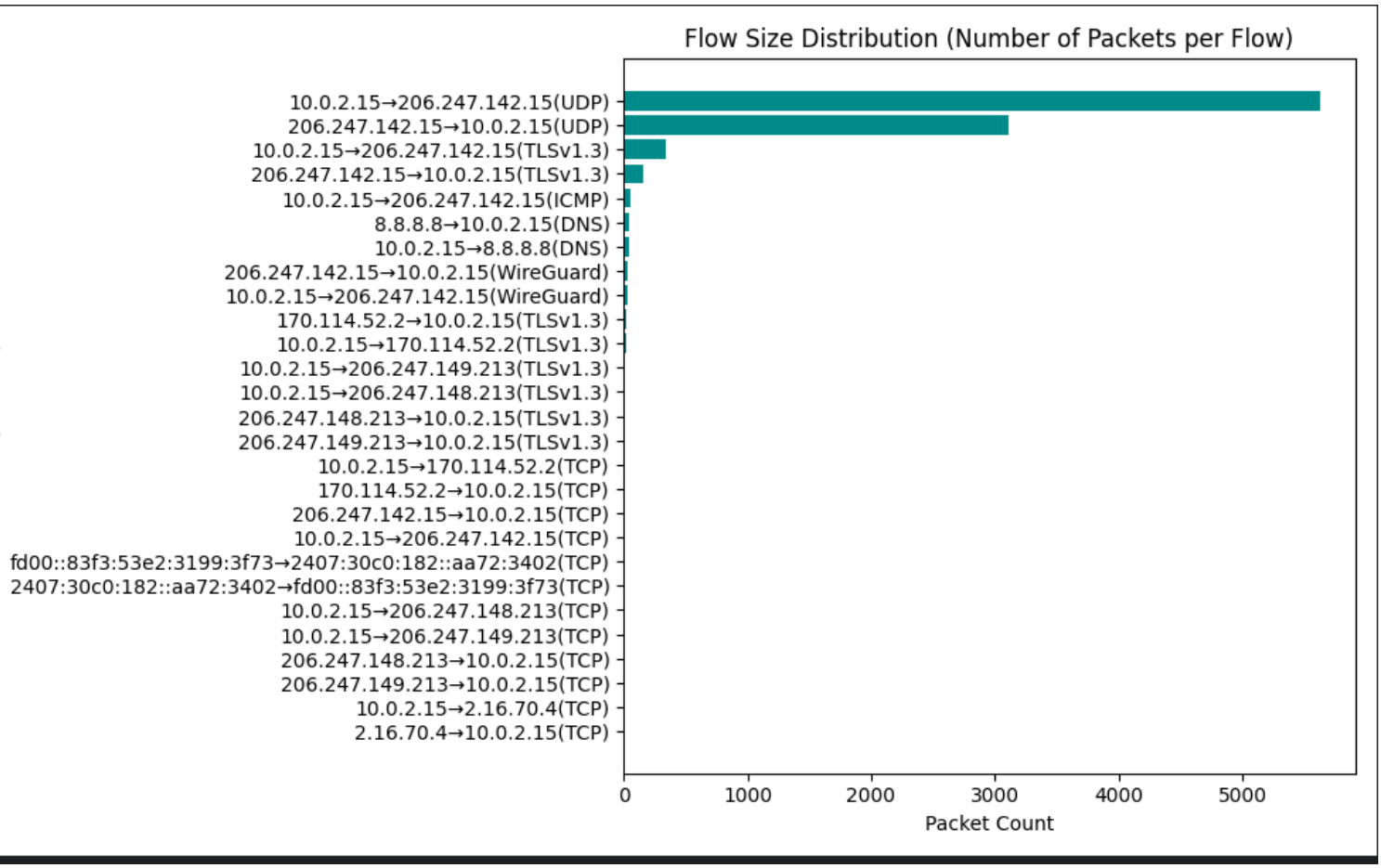
מה גרף זה מציג :גרף זה מציג את ההתפלגות של גודל הזרמים לפי מספר החבילות בכל זרם. זרם (Flow)מתאר את התקשורת בין מקור ליעד עם פרוטוקול מסוים כגון:

TCP || UDP

מה נבדק :הגרף סופר את כמות החבילות בכל זרם על פי העמודות 'Source', 'Destination', ו-'Protocol'.הוא מראה את מספר החבילות של כל זרם בתעבורה ומסדר את הזרמים לפי כמות החבילות.

הסבר לגרף:

- ציר ה :X-מספר החבילות בזרם.(Flow)
- ציר ה :Y-הזרמים השונים למשל.(UDP)10.0.2.15→206.247.142.15 ,
- כל בר מציין זרם בתעבורה עם כמות החבילות הנשלחות בו.



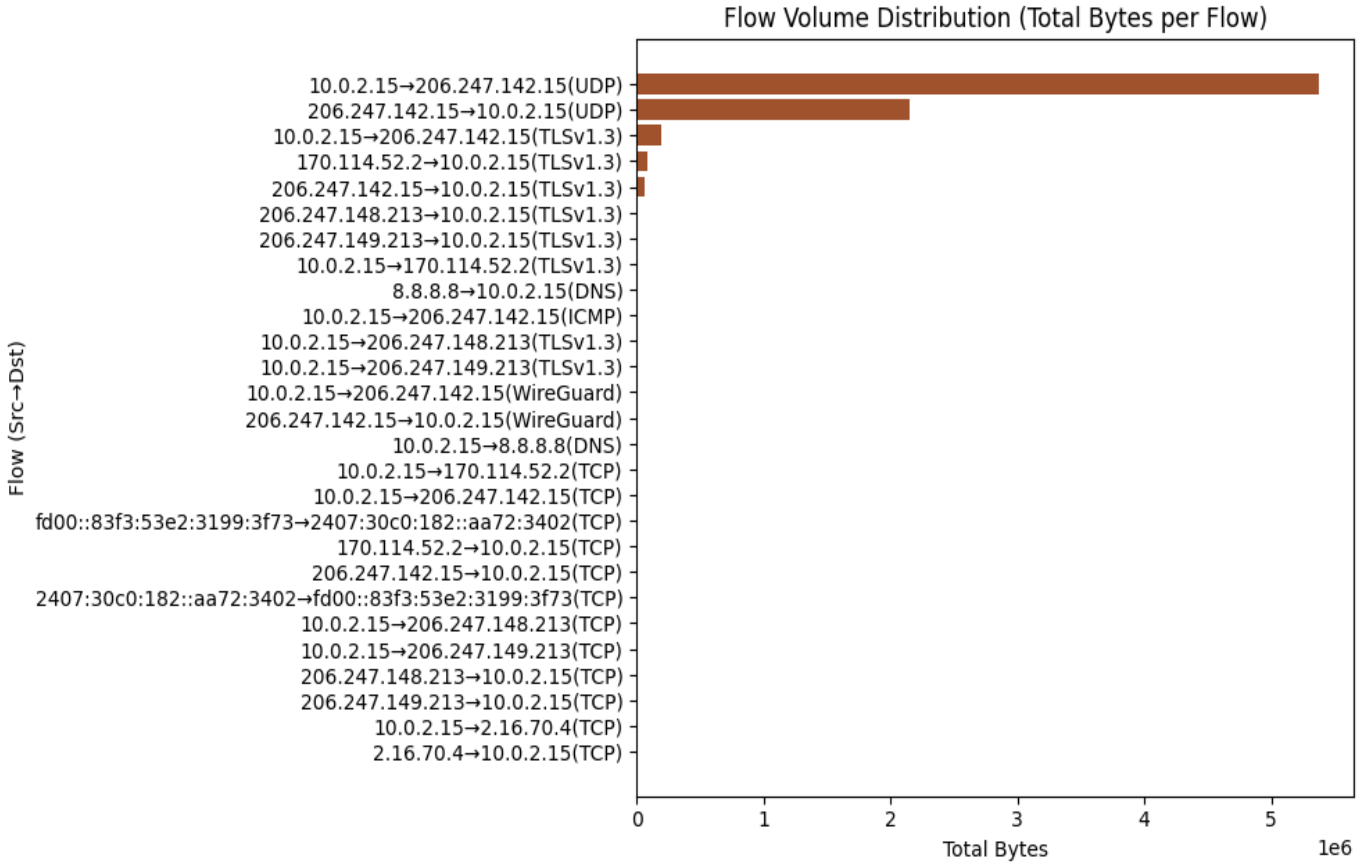
Flow Volume Distribution (Total Bytes per Flow)

מה גרף זה מציג :גרף זה מציג את ההתפלגות של נפח הזרמים לפי סך כל הבייטים שנשלחו בכל זרם. הזרם מתאר את התקשורת בין מקור ליעד, והגרף מציג את כמות הנתונים הכוללת שנשלחה בכל זרם.

מה נבדק :הגרף מסכם את סך כל הבייטים שנשלחו בכל זרם, תוך שימוש בעמודת 'Length'לצורך חישוב סך הנתונים המועברים.

הסבר לגרף:

- ציר ה :X-סך כל הבייטים שנשלחו בזרם.
- ציר ה :Y-הזרמים השונים למשל.(UDP)10.0.2.15→206.247.142.15 ,
- כל בר מייצג זרם אחד, והגובה של הבר מציין את סך הנתונים שהוזרמו בזרם הזה



סיכום הממצאים

במהלך העבודה הנוכחית, ביצעתי ניתוח תעבורת רשת שנאספה במהלך שימוש באפליקציית Zoom. העבודה התמקד בניתוח התעבורה כדי להבין את מאפייני השימוש, תוך מתן הסברים סטטיסטיים וגרפיים על משתנים שונים בתעבורה, כגון זמני ההגעה, גודל החבילות, דגלי ה TCP, ופרוטוקולי ההצפנה.

הנתונים נאספו באמצעות Wireshark, שנעשה בו שימוש לפענוח תעבורה שמועברת בהצפנה באמצעות TLS (Transport Layer Security). בנוסף, השתמשתי בכמה טכניקות על מנת לפענח את התעבורה המוצפנת, שכללו הגדרת משתנה סביבה המאפשר ל Wireshark לשמור את המפתחות הדרושים לפענוח המידע, כמו גם סינונים שונים שנעשו במטרה למקד את התעבורה ל Zoom.

ממצאים עיקריים:

1. **התפלגות כתובת ה IP:**
   - ניתוח כתובת ה IP הצביע על כך שהתעבורה ב Zoom נעשתה בעיקרה מכתובת IP אחת או ממספר כתובות עם יחס יציב בין מקורות ויעדים. ניתן לראות שתעבורה רבה התבצעה מכתובת IP אחת כגון 10.2.15 כשהיא פעלה הן כמקור והן כיעד.
2. **דגלי TCP:**
   - הגרפים שמציגים את דגלי ה TCP שציינו שהדגלים המשמעותיים ביותר בתעבורה היו SYN, ACK ו FIN, דבר המצביע על פעולות של התחלת חיבור, הכרת חבילה וסיום חיבור בין השרתים.
3. **התפלגות שמות שרתי TLS:**
   - רוב התעבורה נקשרה לשמות שרתים כמו zoom.us. הדבר מצביע על חיבורים בין השרתים של Zoom.
4. **התפלגות גודל החבילות:**
   - ניתוח גודל החבילות הצביע על כך שברוב המקרים גודל החבילות היה קטן (בין 100-500 בייטים), אך היו גם חבילות גדולות יותר (במאות אלפי בייטים) כאשר מדובר בשיתוף וידאו.
5. **הבדל זמן בין חבילות:**
   - נתונים מראים שהתעבורה הציגה זמני הגעה נמוכים מאוד בין החבילות, מה שמעיד על יציבות התעבורה ומומנטום גבוה של נתונים.
6. **גודל זרם ומספר החבילות בזרם:**
   - הזרמים המייצגים את החיבור בין מקורות ליעדים הראו יחס ביניהם, והנתונים הצביעו על זרמים גדולים יותר במקרים של שיחות וידאו.
7. **נפח זרם:**
   - במהלך השיחות, בעיקר בשיחות וידאו, נפח הזרם היה גבוה יותר, דבר המעיד על אופי התעבורה הכבדה.

סיכום אבטחה:

בהתאם לממצאים, מצאנו שזמני הגעת החבילות היו מהירים ושההצפנה באמצעות TLS מספקת רמה גבוהה של אבטחה. עם זאת, קיימת סיכון בתהליך פענוח המידע שנעשה על ידי ניתוח דפוסי התעבורה כגון גודל החבילות והפרוטוקול.

לכן, כדי להקשות על התקפות מסוג זה ולשפר את אבטחת המידע, מומלץ להשתמש בהצפנה ברמה גבוהה יותר כגון TLS 1.3, חיזוק אבטחת המפתחות באמצעות טכניקות פרקטיות כמו End-to-End Encryption, ומערכות גילוי חדירה (IDS) לזיהוי ומניעת תקיפות.

מסקנות:

- תעבורת Zoom יציבה אך דורשת שיפור בתחום האבטחה, במיוחד סביב החשש לפרטיות המשתמשים.
- יש צורך בהצפנה חזקה יותר והגברת ההגנה על מפתחות הצפנה.
- ניתן לשפר את פרטיות המשתמשים על ידי שימוש בטכניקות כמו הסוואת דפוסי התעבורה, הצפנה בקצה-לקצה (E2EE), ופרוטוקולים נוספים.

במקטע זה, נבצע ניתוח על תעבורת רשת של המשתמש כאשר התוקף מנסה ללמוד אילו אתרים ואפליקציות המשתמש גולש אליהם. באמצעות השוואה בין פרוטוקולים שונים כמו TCP, UDP, ו-TLS והבנת השפעת הצפנה על זיהוי אפליקציות, נוכל להציג את התמונה בצורה גרפית וברורה.

1. **כאשר התוקף יודע את כל המידע על כל חבילה**

כאשר התוקף יודע את המידע על כל חבילה, כולל גודל החבילה, זמן שליחה, וה-4-tuple (כתובת ה-IP של המקור והיעד, פורט המקור ופורט היעד), הוא יכול לנחש את האתר או האפליקציה שהמשתמש התחבר אליהם.

מה התוקף יכול לדעת?

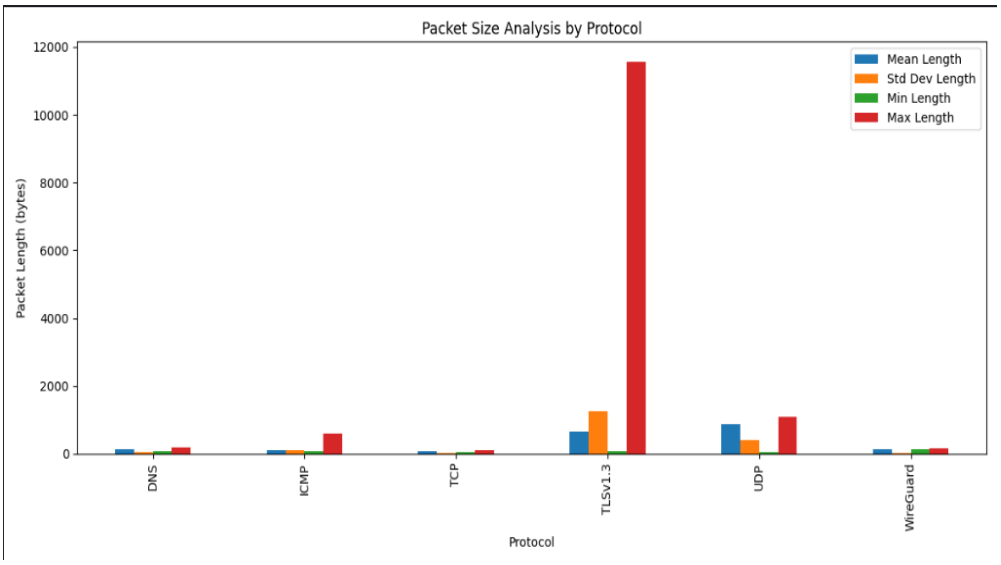
- גודל החבילה: כל אפליקציה או אתר דורשים חבילות בגודל שונה. אפליקציות כמו Zoom או Skype עשויות להשתמש בחבילות גדולות יותר, בעוד שדפדפנים או שירותי DNS עשויים להשתמש בחבילות קטנות יותר.
- הזמן: זמן שליחת החבילה יכול להעיד על פעולות בזמן אמת, כמו שיחות וידאו.
- ה-4-tuple: פורטים יכולים לעזור לתוקף לזהות את פרוטוקול התקשורת. לדוגמה, פורט 443 מציין שימוש ב- HTTPS.

גרף 1: ניתוח גודל החבילה לפי פרוטוקול

גרף זה מציג את **גודל החבילה הממוצע**, **סטיית התקן**, **האורך הקטן ביותר** ו**הגדול ביותר** עבור כל פרוטוקול בתעבורת הרשת. ניתן לראות ש:

- **TLSv1.3** -מציג חבילות גדולות במיוחד, מה שמעיד על הצפנה או תעבורה כבדה.
- **TCP**ו- **UDP** מציגים חבילות קטנות יותר ויעילות יותר.
- התוקף יכול להבחין בין סוגי התעבורה באמצעות השוואת גודל החבילות.

מסקנה:
הצפנה באמצעות **TLSv1.3** גורמת לתעבורה גדולה יותר שמצריכה יותר רוחב פס, בעוד ש **UDP** - מצביע על תעבורה קלה יותר.



2. כאשר התוקף יודע רק את גודל החבילה וזמן ההגעה שלה

במקרה הזה, התוקף יודע רק את הגודל של החבילה ואת הזמן שבו היא נשלחה. זיהוי האתר או האפליקציה קשה יותר, אך עדיין אפשרי:

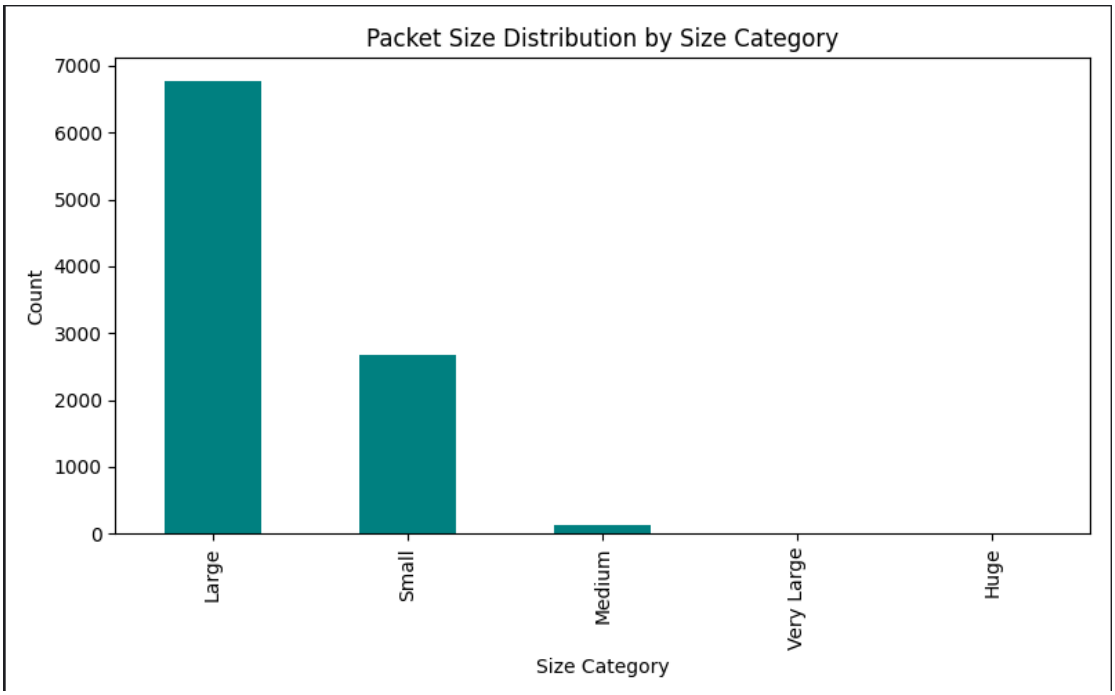
- **גודל החבילה**: חבילות גדולות יותר עשויות להעיד על אפליקציות עם תעבורה רבה כמו שיחות וידאו.
- **הזמן**: תוקף יכול לנחש שיחה בזמן אמת אם הוא רואה חבילות גדולות בזמנים קבועים.

גרף 2: התפלגות גודל החבילות לפי קטגוריות

גרף זה מציג את ההתפלגות של החבילות לפי **Large, Small, Medium, Very Large,** ו**Huge**-.

ניתן לראות שהקטגוריה **Large** מכילה את רוב החבילות, דבר שמעיד על תעבורה כבדה.

מסקנה:
המשתמש ככל הנראה עוסק במשימות כמו שיחות וידאו (עם חבילות גדולות), בעוד ששירותים אחרים כגון **DNS** מצריכים חבילות קטנות.



3.מה קורה כשיש הצפנה?

כאשר התעבורה מוצפנת כמו HTTPS או TLS התוקף לא יוכל לראות את התוכן של החבילה, אך הוא עדיין יכול לראות את **גודל החבילה** ו**הזמן** שבו היא נשלחה.

- לדוגמה, אם התוקף רואה חבילות גדולות שמגיעות בזמן קבוע, הוא יכול לשער שהמשתמש מבצע שיחה בווידאו ב ,Zoom-אך לא יכול לדעת את פרטי השיחה עצמה.
- אם התוקף יודע שהחיבור הוא HTTPS (פורט 443), הוא יוכל להניח שהמשתמש משתמש באפליקציה כמו ,Zoom אך לא ידע את תוכן התקשורת.

איך להקל על הגנה מפני התקפה כזו?

- **שימוש ב VPN-או Proxy**: אם המשתמש גולש דרך VPN או Proxy המידע אודות מקור ה ,IP-פורטים, וזמן החבילות יוסתר, מה שיקשה על התוקף לנחש את היעד.

- **הצפנה מקצה לקצה** :אם התעבורה מוצפנת מקצה לקצה, התוקף לא יוכל לדעת את היעד או את התוכן של החבילה.
- **הסוואת זמנים ושליחה אקראית של חבילות** :שינוי זמני שליחת החבילות יכול להקשות על התוקף לנחש מתי התרחשו פעולות מסוימות.

סיכום :במקרים בהם התוקף יודע את גודל החבילה, זמן שליחתה ואת פרטי ה-4 ,tuple-הוא יכול לנחש אילו אתרים או אפליקציות המשתמש גולש אליהם. הצפנה מוצפנת כמו TLS ו HTTPS- מפחיתה את היכולת של התוקף לראות את תוכן החבילות, אך הוא עדיין יכול לנחש את הפעולות שבוצעו על סמך גודל החבילה וזמן ההגעה שלה. שימוש ב ,VPN-הצפנה מקצה לקצה והסוואת זמני שליחת החבילות יכולים לסייע בהגנה מפני התקפות כאלו.

סיכום כללי:

במהלך הפרויקט, עסקת בניתוח תעבורת רשת שנאספה במהלך שימוש באפליקציית Zoom תוך שימוש במגוון כלים כגון **Wireshark** לפענוח התעבורה המוצפנת, שימוש במפתחות ,TLS וסינונים שונים. כל זאת על מנת להבין את דפוסי השימוש, גודל החבילות, דגלי ה ,TCP-שדות TLS ועוד. הצלחתה של עבודה זו הוכחה עם הגרפים שהסברת בהם בצורה ברורה את אופי התעבורה.

בנוסף, עבדת על הבנת אפשרויות ההתקפה האפשריות על תעבורת הרשת אם התוקף היה יכול לגשת למידע הרגיש, תוך הצגת פתרונות אפשריים לחיזוק אבטחת המידע כמו שימוש ב ,VPN- הצפנה מקצה לקצה, והסוואת זמני שליחת החבילות.

מסקנות עיקריות:

- התעבורה ב Zoom-מראה דפוסים מובהקים שיכולים להעיד על שימוש בשיחות וידאו או אמצעים אחרים, תוך שהמפתחות TLS מספקים פרטיות ברמה גבוהה יותר.
- באמצעות המידע שנמצא בתעבורה כמו גודל החבילה, זמן ההגעה, דגלי ה TCP-ועוד ,תוקף יכול לשער את האפליקציות והאתרים בהם המשתמש גולש.
- הצפנה מקצה לקצה יכולה למנוע גישה לתוכן התעבורה, אך לא מונעת את זיהוי האפליקציה או השרת שהמשתמש גולש אליו.

המלצות לאבטחה:

- שימוש ב VPN-או Proxy להסוואת המידע.
- הצפנה מקצה לקצה על מנת למנוע זיהוי של תוכן התעבורה.
- הסוואת זמני שליחת החבילות כדי להקשות על התקפות נוספות.

השוואת הגרפים בין האפליקציות השונות:

מטרתנו הינה לחקור כיצד ניתן לזהות אפליקציות ושירותים שונים על סמך מאפייני התעבורה שלהן, ולבחון עד כמה תוקף פוטנציאלי יכול להסיק מידע על פעילות המשתמש, גם ללא גישה ישירה לתוכן הנתונים.

במהלך העבודה שלנו, נרשמה תעבורה של אפליקציות שונות(גלישה באינטרנט, Streaming אודיו ו-וידאו, ושיחות וידאו), תוך שימוש ב-Wireshark.

לאחר מכן, נותחו מאפיינים מרכזיים כמו גודל חבילות, קצב שליחה, ושימוש בפרוטוקולים.

ניתוח זה מאפשר לנו להבחין בהבדלים בין שירותים שונים, ולתת לנו לזהות דפוסים מובנים בתעבורה שלהם.

לבסוף, כל אפליקציה מתייחסת לתרחיש שבו תוקף מנסה לזהות את השירותים בהם נעשה שימוש, תוך שימוש ב-2 מצבי תקיפה שונים:

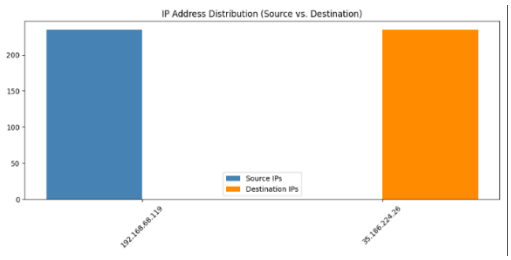
כאשר תוקף יודע את ה-Flow ID, וכאשר התוקף יודע רק את גודל החבילות וחותמות הזמן, ללא מידע מזהה אחר.

כחלק מהניתוח, נשווה בין האפליקציות השונות ←תוך בחינת ההבדלים בכל סוג גרף מרכזי, כמו למשל: גודל חבילות, קצב שליחה, ושימוש בפרוטוקולים. השוואה זו תאפשר לזהות דפוסים ייחודיים לכל שירות , וגם להבין כיצד ניתן להבדיל ביניהם על בסיס מאפייני התעבורה בלבד.

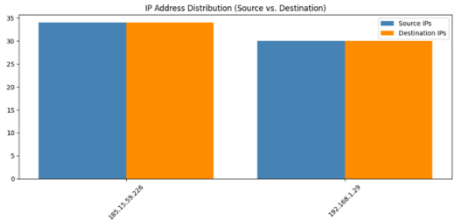
הגרף הראשון-שדות של כתובות ה-IP:

כעת נציג את הגרף הראשון של כל האפליקציות:

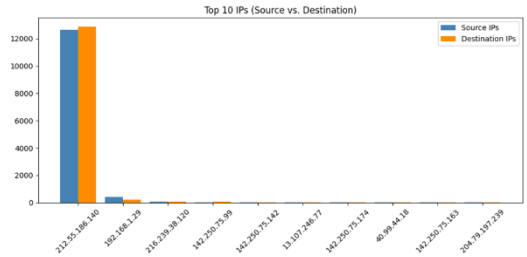
עבור Spotify:



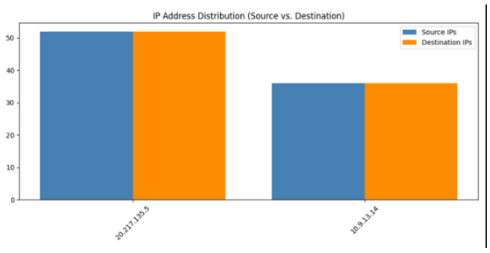
עבור ויקיפדיה(דפדפן chrome):



עבור YouTube:

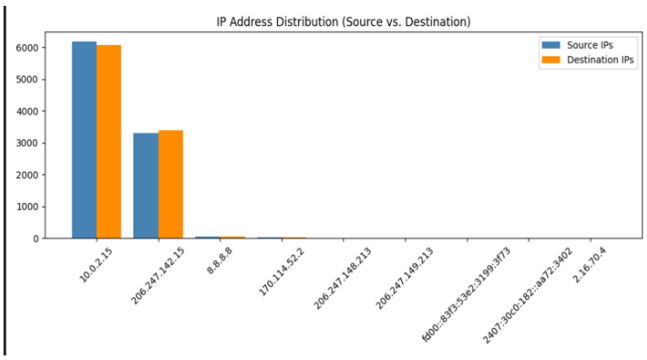


ב-



Microsoft Edge:

עבור Zoom:



נזכיר כי סוג גרף זה אומר כי יש לנו שני סוגים של עמודות:

עמודה כחולה, אשר משמעותה היא כתובת מקור(Source IPs), ועמודה כתומה, אשר משמעותה(Destination IPs).

עמודות אלו מראות את כמו החבילות הנשלחות לכל כתובת.

ב-**Spotify**, אפשר לראות כי יש חיבור יציב לשרת יחיד, וזאת בניגוד לYouTube-אשר מתחבר לכמה שרתים שונים.

ב-Spotify אין חיבורים נוספים, כל התעבורה זורמת דרך שרת אחד קבוע(ללא פיזור לשרתים אחרים), וזאת בניגוד ל-Chrome-אשר מחלק את התעבורה בין אתרים שונים.

בנוסף, בSpotify יש streaming אודיו מתמשך לשרת מרכזי לקבלת נתונים.

ב-**YouTube**, אפשר לראות כי יש שרת מרכזי עם פיזור לשרתי משנה(CDN),וזאת בניגוד לSpotify אשר מתחבר לשרת יחיד.

ישנו שרת יעד דומיננטי אחד, אך ישנם גם מספר חיבורים נוספים לשרתי CDN(=שרת להפצת תוכן , שתפקידו לאחסן ולהגיש נתונים (כגון וידאו, תמונות, וקבצים) בצורה מהירה ויעילה יותר).

יש כמות גדולה של חבילות לשרת הראשי, כי ידוע שstreaming וידאו מצריך זרימה רציפה של מידע, וזאת בניגוד לאודיו(Spotify).

דבר זה עומד בניגוד לChrome, אשר לא עובד עם שרת יעד ברור.

ב-**Chrome** , ישנם חיבורים מרובים ללא שרת דומיננטי.

בניגוד ל- YouTube ול-Spotify , הנתונים מתחלקים בין מספר כתובות יעד שונים.

בChrome, רואים שיש הרבה חיבורים קצרים לאתרים שונים, וזאת בניגוד לstreaming , אשר דורש חיבור רציף.

בניגוד לYouTube, אשר עובד עם שרת ראשי+ שרתי משנה של CDN, בChrome החיבורים מחולקים שווה בין היעדים(אפשר לראות זאת בגרף).

בנוסף, בכך שתעבורת דפדפן משתנה מאוד, אזי אין דפוס אחיד של שליחה וקבלה , וזה נובע בכך שזה תלוי בהתאם לאתרים שהמשתמש גולש בהם.

ב-**Microsoft Edge**, יש פה הצגה של שני שרתי יעד בלבד, מה שמצביע על תצורת שרתים יותר ריכוזית מאשר ב-Chrome, אך פחות ריכוזית מ-Spotify.

בנוסף, אפשר לראות כי יש מיעוט יחסי של כתובות היעד, מה שעמיד לרוב על שימוש ב-Microsoft Edge.

ניתן לזהות אם Edge מחובר לשירותים מסוימים על ידי זיהוי כתובת ה-IP של היעד.

ב-**Zoom**, אפשר לראות כי יש 2 כתובות עיקריות , השולטות בתעבורה, כאשר הכתובות 206.47.142.15, 10.2.1.5 מופיעות באופן משמעותי.

הכתובת 10.2.1.5 היא כתובת פנימית, מה שמרמז על תעבורה פנימית בתוך הרשת לפני יציאה לאינטרנט.

הכתובת 206.47.142.15 היא ככל הנראה אחת מכתובות השרתים של Zoom.

בניגוד ל-YouTube , אשר בו קיימים מספר שרתים מבוזרים של CDN, ב-Zoom נראה שהתעבורה מתרכזת במספר קטן של שרתים, שמנהלים את השיחות בווידאו ובאודיו.

Zoom דורש מספר חיבורים, זאת על מנת לאפשר שידור חי של ווידאו ואודיו באופן רציף, וללא השהיות.

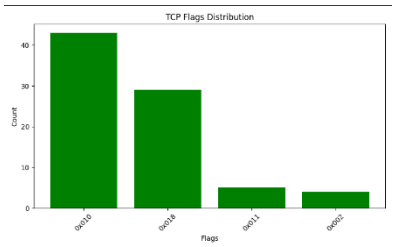
במונחים של ניהול עומסים← Zoom פועל כמו Microsoft edge, בכך שהוא שומר על חיבורים יציבים לשרתים מרכזיים, אך עם נטייה חזקה יותר לשימוש בשרתים ספציפיים, ולא פיזור רחב.

Zoom, בדומה ל-YouTube, צורך רוחב פס גבוה באופן יחסי, אך בשונה מ-YouTube, קצב הנתונים משתנה בהתאם לאיכות השידור בזמן אמת, מה שמאפשר התאמה דינאמית לתנאי הרשת.

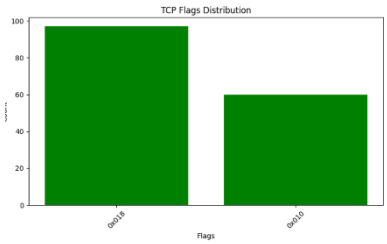
הגרף השני-שדות כותרות ה-TCP:

כעת נציג את הגרף השני של כל האפליקציות:

עבור Spotify:

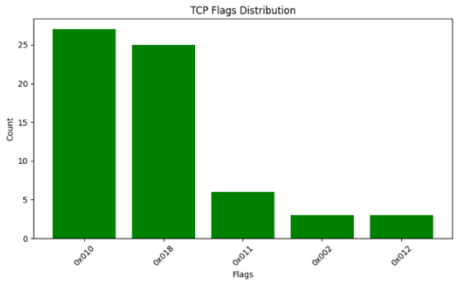


עבור YouTube:

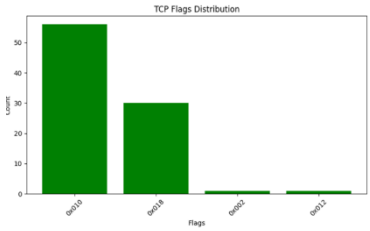


עבור ויקיפדיה(דפדפן

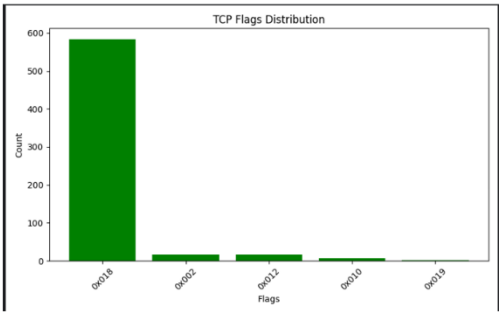
chrome



ב-Microsoft Edge:



עבור Zoom:



נזכיר כי בסוג זה של גרף מציג התפלגות של דגלי TCP בחבילות הרשת שנקלטו מה-Wireshark.

ב-**Spotify**, אפשר לראות כי ACK הינו הדגל הכי נפוץ, וזאת כי streaming אודיו מבוסס על חיבור יציב עם אישור קבוע של נתונים.

כמות גבוהה של Spotify , אפשר לראות זאת דרך הדגל PSH , אשר שולח נתונים בתבנית יציבה, אך אינה בכמות מאסיבית כמו וידאו(YouTube).

בנוסף, יש מעט מאוד FIN ו- SIN , כמעט ואין סגירות חיבורים או פתיחות חדשות, כי streaming אודיו דורש חיבור יציב.

ב-**YouTube**, אפשר לראות כי הדגל הכי נפוץ הינו PSH, וזאת כי streaming וידאו דורש שליחה רציפה ומיידית של נתונים, בקצב גבוה.

הדגל ACK קיים גם, אך פחות מ-PSH, השרת לא מחכה לכל אישור לפני שליחת הנתונים, המערכת סומכת על הזרימה המהירה של הנתונים.

כמו בSpotify, גם ב-YouTube יש מעט מאוד FIN וסגירת חיבורים- החיבור נשאר פתוח לאורך זמן.

ב-**Chrome**, אפשר לראות ש-ACK ו-PSH נמצאים ברמות גבוהות, וזאת כי טעינת דפים מצריכה אישור נתונים ושליחה מהירה.

יש לנו פה בניגוד ל-YouTube ול-Spotify כמות חבילות FIN גדולה יותר, וסגירת חיבורים← דפדפן סוגר הרבה חיבורים קצרים, כאשר הוא מסיים לטעון אתרים.

יש גם נוכחות של חבילות SIN ,וזאת כי דפדפן פותח הרבה חיבורים חדשים לאתרים שונים.

בניגוד ל-YouTube ול-Spotify, אין חיבור יציב לאורך זמן, וזאת כי דפדפן טוען דף, סוגר חיבור , ועובר לחיבור הבא.

ב-**Microsoft Edge**, אפשר לראות כי דגל ה-ACK הינו הדומיננטי ביותר, כמות גדולה של דגלים אלו מעידה על זרימה יציבה של תעבורה, עם אישורי קבלה תכופים.

בנוסף יש גם PSH+ACK, וזה מסמן שהנתונים צריכים להידחף מידי ליישום, ללא המתנה לאגירת חבילות נוספות.

ישנה כמות קטנה של דגלי SYN ו-SYN+ACK , וזה מעיד לנו כי הגרף נלקח באמצעות סשן גלישה, ולא בעת פתיחת חיבורי רשת חדשים בכמות גדולה.

Microsoft Edge מתנהג באופן דומה כמו ב-Chrome , עם דגש על זרימה עקבית של תעבורה, ואחוז גבוה של ACK ← ACK+PSH.

בניגוד ל-YouTube , ול-Spotify , הגלישה ב-Microsoft Edge דורשת פחות חבילת דחופות(PSH), מאחר שמדובר בדפדפן, ולא בשירות Streaming מתמשך.

ב-**Zoom**, אפשר לראות כי הדגל הנפוץ ביותר הינו 0X018, המייצג חבילות עם PSH+ACK, המעידות על זרימה רציפה של נתונים← וזאת כי Zoom מבצע שידור בזמן אמת, ולכן עליו להבטיח שהנתונים מועברים באופן רציף ומהיר.

השימוש הגבוה ב-PSH, מבציע על כך שאין צורך להמתין לאישור לפני שליחת מידע נוסף← דבר קריטי לווידאו שמע.

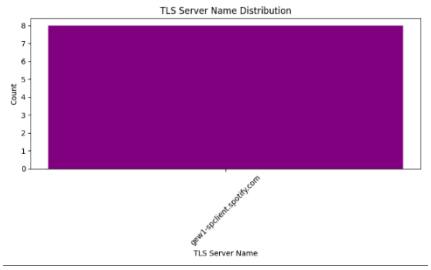
ניתן לראות כמות מינימאלית של דגלים כמו SIN, FIN , דבר המעיד על כך ש-Zoom שומר על חיבורים פתוחים לאורך זמן, ולא מבצע פתיחות וסגירות תכופות.

אפשר לראות כי השימוש בדגל ה-SIN (0X002)הינו נמוך← וזאת בניגוד לדפדפנים (שם יש הרבה SYN לפתיחת חיבורים קצרים), וזה מצביע על כך ש-Zoom לא יוזם חיבורים חדשים לעיתים קרובות, אלא פועל דרך חיבור קיים לאורך זמן, דבר זה תורם לחוויית שימוש חלקה.

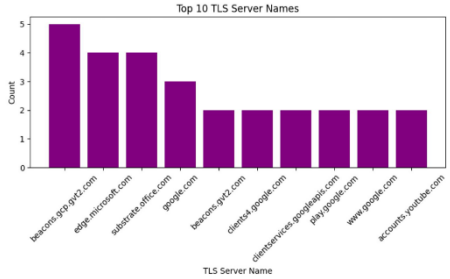
הגרף השלישי-שמות שרתים ב-TLS:

כעת נציג את הגרף השלישי של כל האפליקציות:

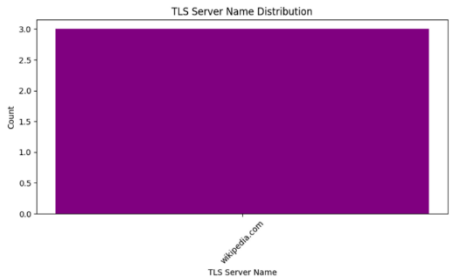
עבור Spotify:



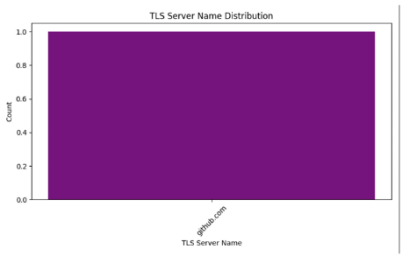
עבור YouTube:



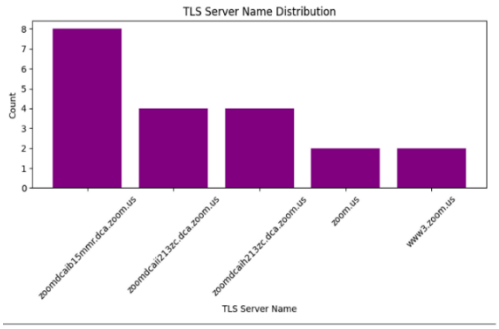
עבור ויקיפדיה(דפדפן chrome)



ב-Microsoft Edge:



עבור zoom



נזכיר כי סוג זה של גרף מציג את ההתפלגות של שמות של שרתים, אשר התגלו בתקשורת ה- TLS.

ב-**Spotify**, אפשר לראות כי יש שרת יחיד בלבד, כל התעבורה עוברת דרך שרת יחיד של Spotify.

לעומת YouTube, אין ב-Spotify חיבורים לשרתי צד שלישי.

אין לנו מגוון של שרתים, יש תעבורה מאוד צפויה וקבועה, זוהי תעבורה יציבה מול כתובת אחת בלבד.

ב-**YouTube**, יש לי פה ריבוי שמות שרתים, למשל
gvt2.com,play.google.com,google.com,accounts.youtube.com

יש פה שימוש ב-CDN(Content Delivery Network), סרטוני YouTube מגיעים ממקורות שונים כדי לאפשר זרימה יותר מהירה.

התקשורת ב-YouTube מתחלקת בין המון שרתים-אם זה שרתים כללים של גוגל, או קשור לחשבון המשתמש, או גם קשור לשרתי הCDN , ובכך← קשה יותר לזהות באופן ישיר שהתעבורה היא בהכרח של YouTube בלבד, וזאת כי היא מתפזרת בין כמה שירותים של גוגל.

וזאת בשוני מ-Spotify, אשר כל התעבורה הולכת לשרת אחד ספציפי.

ב-**Chrome**, אפשר לראות שזה תואם למה שמצפים באתר בדפדפן בזמן ההקלטה← יש שם שרת יחיד, שזה האתר של ויקיפדיה.

בניגוד ל-YouTube, אין שימוש בשרתי CDN(ובכך זה מאפיין מובהק לגלישה ישירה) , או שימוש בשירותי צד שלישי, ובכך זה מאפיין של גישה ישירה לאתר יחיד.

ב-Chrome, יש חיבורים ישרים וקצרים, משמע הוא ש-Chrome פותח חיבור TLS לכל אתר בנפרד.

ב-**Microsoft Edge**, כל התעבורה נותבה אל אתר ה-GitHub , מה שמעיד שהמשתמש היה מרוכז בעיקר באתר זה(פעילות ממוקדת).

גם כאן , בשוני משירותי Streaming ← אשר בהם יש חיבורים למספר שרתי CDN(YouTube), כאן נראה רק שרת אחד, אשר מרכז את כל התקשורת.

ב-**Zoom**, אפשר לראות כי הוא מתקשר למספר שרתים שונים תחת הדומיין zoom.us, וזאת בניגוד ל-Spotify ← שבו יש שרת אחד דומיננטי.

הדבר מעיד על מבנה מבוזר של התקשורת, שבו הלקוח אינו מתחבר לשרת אחד בלבד, אלא לשרתים שונים לפי הצורך.

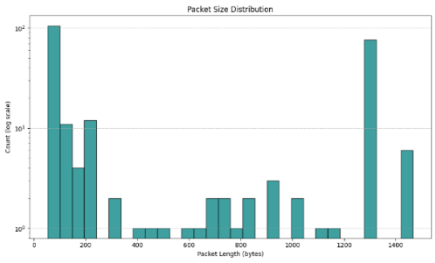
בנוסף, דבר זה מאוד אופייני לשירותי ווידאו , שבהם נדרשת תקשורת אמינה ורציפה עם שרתים קרובים למשתמש.

בלבד, מה שמעיד על אופי תקשורת Zoom כל החיבורים מתרכזים בפלטפורמת Zoomאפשר לראות כי ב-אשר מבצעים חיבורים לאתרים מגוונים אשר המשתמש גולש ←יותר סגור, וממוקד, זאת בניגוד לדפדפנים אליהם, כמו למשל אצלנו ב-GitHub ,Wikipedia.

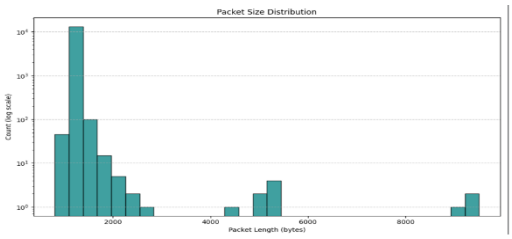
הגרף הרביעי-חלוקת גודל החבילות:

כעת נציג את הגרף הרביעי של כל האפליקציות:

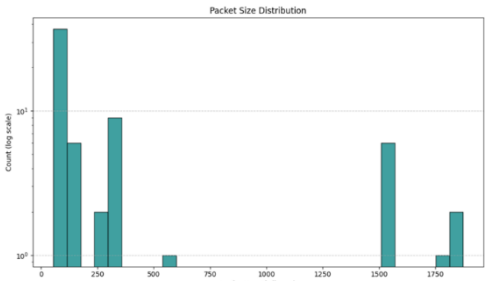
עבור Spotify:



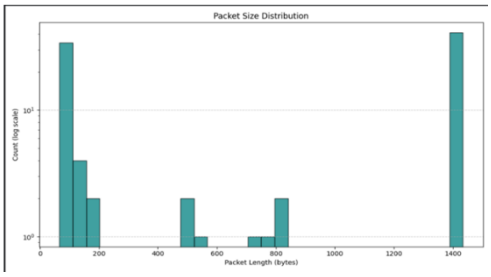
עבור YouTube:



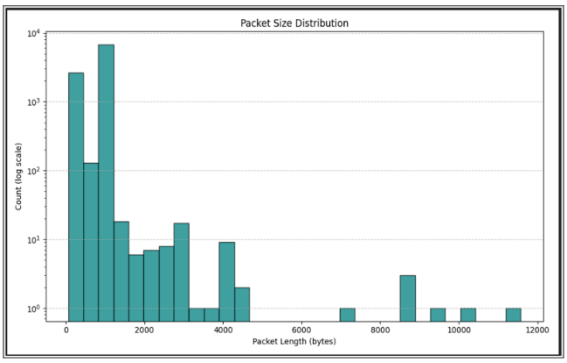
עבור ויקיפדיה(דפדפן chrome):



עבור Microsoft Edge-ב:



עבור Zoom:



נזכיר כי גרף זה הינו מציג התפלגות של גדלי חבילות שנמצאו בתעבורת הרשת, המטרה היא להבין מה הם הגדלים של חבילות הרשת שנלכדו במהלך הניתוח.

ב-**Spotify**, אפשר לראות שיש (bytes 100-200) חבילות קטנות, המשמשות לשליחת בקשות ופינגים← לבדוק באמת שהחיבור פעיל.

יש לנו גם חבילות גדולות (bytes1200-1500), המשמשות להעברת נתוני שמע בקצב אחיד.

אין לנו חבילות גדולות במיוחד(כמו למשל שיש ב-YouTube), ובכך אפשר להבין שזה לא מצביע בתוכן וידאו כבד, ובכך יש קצב זרימה מתון.

יש לנו חוסר עקביות בגודל החבילות← Spotify שולח מידע בגודל משתנה בהתאם לזרימה של השיר.

ב-**YouTube**, יש לנו חבילות גדולות מאוד(bytes10,000-90,000), וזאת כי streaming וידאו דורש קצב נתונים גבוה מאוד, ולכן החבילות גדולות יותר.

יש התפלגות אחידה סביב חבילות גדולות (בניגוד לדפדפן(Chrome)-שמציג חוסר אחידות), זאת כי YouTube שולח חבילות בגודל אחיד, כי וידאו מצריך הזרמה יציבה.

אנחנו רואים כי יש מעט מאוד חבילות קטנות(כמו ב-Chrome), מרבית הנתונים נשלחים במקטעים גדולים, למעט בקשות שליטה קטנות.

דבר זה מצביע על כך שלא מדובר בגלישה רגילה.

ב-**Chrome**, אפשר לראות כי יש הרבה חבילות קטנות מאוד(0-250 bytes), זאת כי דפדפן שולח המון בקשות HTTP קצרות.

יש לנו גם קצת חבילות גדולות(1500-1800bytes), הן משמשות להעברת משאבים כבדים(למשל תמונות גדולות).

אין לנו פה אחידות פה כמו בשירותי הstreaming , יש פה התפלגות ריבה מאוד, כי הגודל תלוי באתרים שנטענים← יש פה למעשה קפיצות בגרף עם פיקים בגדלים שונים.

ב-**Microsoft Edge**, אפשר לראות כמו ב-Chrome, כי גם כאן רוב החבילות הינן קטנות יחסית(בסביבות -0 200 bytes), זה נפוץ מאוד בדפדפני אינטרנט, כי דפדפן שולח הרבה בקשות HTTP קטנות.

קיימים גם אורכי חבילות ספציפיים, שיש להם מופעים רבים, כמו בגדלי 600 ו-800 bytes , ייתכן שאלו חבילות TCP ← הכוללות מידע חשוב כמו העברת נתונים חלקית.

ישנו גם פיק גדול באורך 1400 bytes, חבילות בגודל זה הן כנראה חבילות מלאות.

דפדפנים בדרך כלל שולחים נתונים בגודל המקסימאלי, זאת כדי למנוע חלוקה מיותרת של חבילות.

ב-**Zoom**, ניתן לראות כי יש התפלגות רחבה של גדלי החבילות, עם ריכוז גבוה של חבילות קטנות (0-200 bytes), אך גם עם מספר חבילות גדולות מאוד(מעל 10000 bytes).

יש דפוס מובהק של חבילות בינוניות (1400-5000 bytes), מה שמעיד על תעבורת נתונים עקבית.

השימוש בסקאלת ה-LOG מראה כי כמות החבילות הקטנות גדולה משמעותית מכמות החבילות הגדולות.

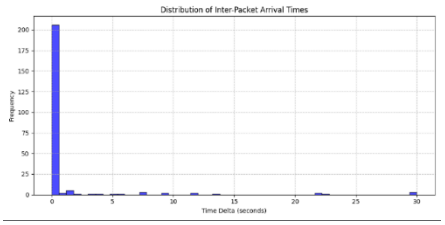
ההתפלגות של ה-Zoom מזכירה קצת את ההתפלגות של ה-Spotify, אך עם פחות חבילות גדולות במיוחד← וזאת שמאחר ואודיו דורש פחות נתונים מווידאו.

יש לנו ב-Zoom תמהיל מאוזן בין חבילות קטנות מאוד לבין חבילות גדולות, מה שמעיד על שילוב של שליחת פקודות, חבילות של אודיו-ווידאו, ושימוש בפרוטוקולים מבוססי UDP/TCP.

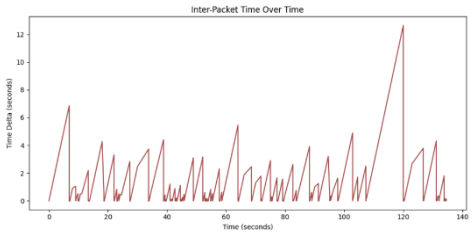
הגרף החמישי-זמני ההגעה בין המנות:

כעת נציג את הגרף החמישי של כל האפליקציות:

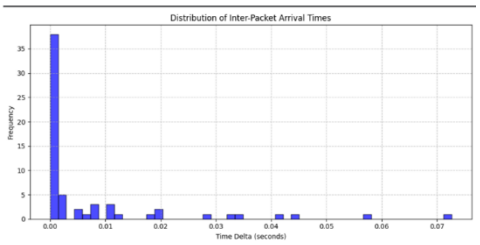
עבור Spotify:



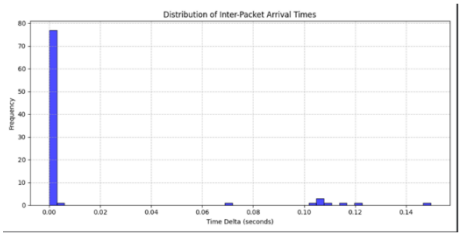
עבור YouTube:



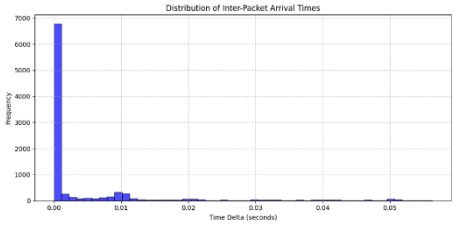
עבור ויקיפדיה(דפדפן chrome):



ב-Microsoft Edge:



עבור Zoom:



נזכיר כי גרף זה הינו מציג התפלגות זמני ההגעה בין החבילות.

ב-**Spotify**, אפשר לראות כי רוב החבילות מתקבלות במרווחים קצרים מאוד(קרוב מאוד ל-0 שניות), וזה מייצג זרימה יציבה של נתונים.

רואים גם כי יש מספר קטן של חבילות עם זמני הגעה גבוהים(פיקים), זה יכול להעיד על תיקוני רשת, או buffering, דבר זה מאוד אופייני לשירותי שמע דחוסים, שלא דורשים כמויות נתונים גדולות.

יש פה בנוסף קצב תעבורה קבוע← Spotify מחייב רצף של נתוני שמע, אך ללא נפח מידע גדול מאוד , כמו שיש ב-YouTube למשל.

ב-**YouTube**, הזמני הגעה משתנים מאוד, זה בין 0 ל-12 שניות, ובכך זוהי אינדיקציה להזרמת וידאו במקטעים גדולים.

יש פה בנוסף גם מעין דפוס מחזורי: חבילות רבות נשלחות יחד, ואז הפסקה, ובכך זה מאוד תואם לטעינת וידאו מראש(buffering).

בנוסף, יש פה שיאים חדים של עיכובים(בשונה מ-Spotify אשר מחייב רצף נתונים מתמיד), וזה מבציע על השהיות בין שליחת מקטעי הוידאו או שינוי באיכות הזרימה.

ב-**Chrome**, רוב החבילות מתקבלות כמעט מיד(קרוב ל-0 שניות), אך ישנם פיקים רבים בטווחי זמן קצרים(0.01-0.07 שניות).

יש פה חוסר יציבות מוחלט(בשוני מ-YouTube ו-Spotify) , וזאת כי זמני ההגעה תלויים באתרים שהמשתמש מבקר בהם.

בנוסף, ישנה תעבורה בלתי צפויה: טעינת דפים כבדים תגרום לעליות חדות, לעומת גלישה קלה ,שתגרום לחבילות קטנות במרווחים אחידים.

ב-**Microsoft Edge**, אפשר לראות כי יש רוב גדול של חבילות שמגיעות בהפרשי זמן קטנים, מה שמאפיין גלישה בדפדפן ללא עומסים.

ניתן להבחין בכמה חבילות שמתעכבות, וזאת ייתכן שזה קשור לטעינה של אלמנטים בדף אינטרנט, כמו תמונות או סקריפטים.

יש פה שוני משירותי Streaming כמו YouTube , ו-Spotify, אשר שם החבילות עשויות להגיע בקבוצות גדולות עם השהיות מסוימות.

בנוסף , בהשוואה ל-YouTube , לא רואים חבילות גדולות שמגיעות בהפרשים משתנים , אלא קצב אחיד יותר.

קפיצות קטנות בזמני ההגעה עשויות לנבוע מטעינה של תוכן אינטרנטי, אך לא משדרים חיים או Streaming.

ב-**Zoom**, ניתן לראות כי רוב החבילות מגיעות בהפרשי זמן קצרים מאוד(קרוב ל-0 שניות), וזה מעיד על זרימה רציפה של חבילות בין המשתמשים לשרתי ה-Zoom.

תופעה זו אופיינית לאפליקציות הדורשות תקשורת בזמן אמת, כמו שיחות ווידאו ואודיו.

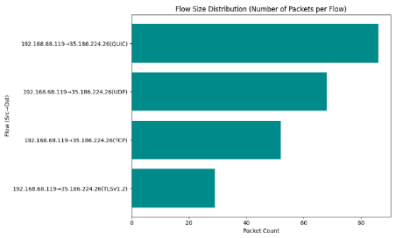
בניגוד ל-Spotify, ו-YouTube, שבהם קיימים זמני הגעה עם הצטברויות בזמנים גדולים יותר, ב-Zoom רוב התקשורת רציפה, ללא המתנות ממושכות, או Buffering משמעותי.

גם בדפדפנים ישנם זמני הגעה קצרים מאוד, אך ישנו פיזור מסוים, מה שמעיד על תקשורת מבוססת HTTP, ולא תקשורת בזמן אמת כמו ב-Zoom.

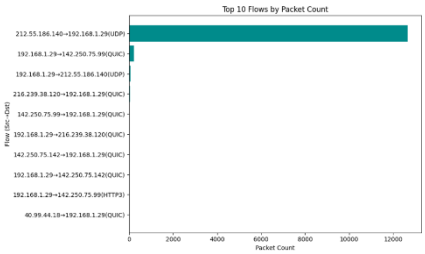
הגרף השישי-המספר הכולל של החבילות:

כעת נציג את הגרף השישי של כל האפליקציות:

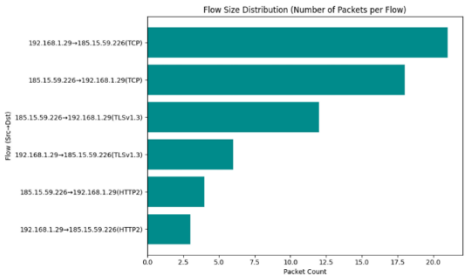
עבור Spotify:



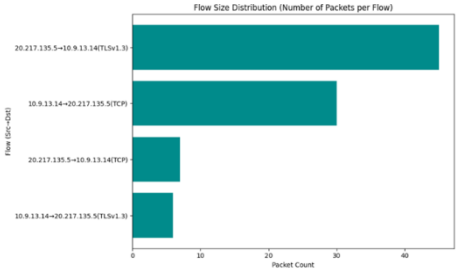
עבור YouTube:



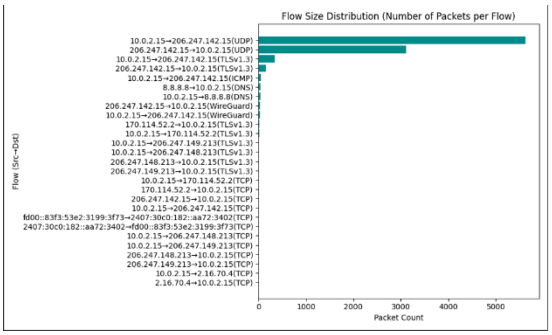
עבור ויקיפדיה(דפדפן chrome):



ב-Microsoft Edge:



עבור Zoom:



נזכיר כי גרף זה הינו מציג את מספר החבילות לכל זרימה שנוצרה בין המקור לבין היעד.

לכל זרימה יש שילוב של כתובת המקור, כתובת היעד, והפרוטוקול ששימש לתקשורת.

ב-**Spotify**, אפשר לראות כי יש כמות חבילות ממוצעת לכל זרימה(50-90 חבילות), זאת כי Streaming שמע מחייב קצב יציב של נתונים(ללא קפיצות קיצוניות), אך לא בנפח גדול.

יש כאן שימוש משמעותי ב-TCP,UDP,QUIC,TLS, Spotify מאזנת בין פרוטוקולים שונים להבטחת מהירות יציבה.

בניגוד לYouTube, אין זרימות עצומות← כל זרימה מכילה מספר חבילות סביר, שלא משתווה לתעבורה של Streaming וידאו.

אפשר להבין כי כמות החבילות בכל זרימה הינה בינונית ב-Spotify, היא לא קטנה כמו בדפדפן, איך היא אינה ענקית כמו ב-YouTube.

ב-**YouTube**, ישנה זרימה דומיננטית אחת ב-UDP עם מעל 12,000 חבילות!!

Streaming וידאו שולח כמות עצומה של נתונים דרך חיבור רציף.

ישנו גם שימוש בפרוטוקולים HTTP3,QUIC , אלו פרוטוקולים מתקדמים המשמשים הזרמת וידאו ללא השהיות.

זרימות קטנות אלו משמשות לבדיקת מצב הרשת, חלקן לטעינת קבצים נלווים כמו תמונות וכותרות.

ב-**Chrome**, ישנה כמות חבילות נמוכה לכל זרימה (5-22 חבילות), וזאת כי דפדפן שולח המון בקשות קטנות לכל דף אינטרנט.

ישנו שימוש בפרוטוקולים ב-HTTP2,TLS,TCP (לעומת YouTube אשר משתמש בUDP), וזה מראה כי מרבית התקשורת מאובטחת עם הצפנה + שימוש בפרוטוקולים אלו לטובת האצת טעינת הדפים.

אין לנו כמו ב-YouTube זרימות ענקיות, הדפדפן שולח המון זרימות קטנות במקום זרימה אחת מאסיבית.

ריבוי זרימות קטנות עם מספר חבילות נמוך← זוהי תבנית שמאפיינת אתרי אינטרנט, אשר טוענים משאבים רבים.

ב-**Microsoft Edge**, אפשר לראות שהוא מבצע שימוש בעיקר TLSv1.3, וזה מעיד על גישה לאתרים מאובטחים, למשל שירותים הדורשים הצפנה.

מדובר בתקשורת מוצפנת בין הדפדפן לבית השרתים המרוחקים.

ישנה זרימה TCP לפני TLS ← לפני שהתעבורה מוצפנת, קיים חיבור TCP רגיל.

תופעה זו נצפית גם בדפדפנים אחרים, כמו למשל ב-Chrome

אין לנו זרימות ענקיות כמו ב-YouTube, אשר מעבירה נפח גדול מאוד של נתונים, ולכן הזרימות מכילות אלפי חבילות.

כאן ב-Microsoft Edge, הזרימות בינוניות, וכוללות עשרות חבילות בלבד, בהתאם לאופי השימוש בדפדפן.

ב-**Zoom**, ניתן לראות כי ה-UDP הינו הפרוטוקול הדומיננטי ביותר(בדומה ל-YouTube), עם מספר חבילות גבוה במיוחד, מה שמעיד על השימוש ב-UDP לשידור נתונים בזמן אמת, כמו וידאו שמע.

TLSv1.3 מופיע כפרוטוקול תקשורת נלווה, המספק הצפנה וביטחון על גבי הזרימות.

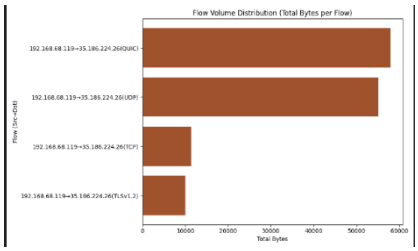
קיימים לנו גם זרימות TCP, אך הן במספרים נמוכים יחסית.

בנוסף, אפשר לראות כי יש שימוש קל ב-WireGuard , מה שמעיד על שימוש בפרוטוקולי VPN, זאת כנראה על מנת לאבטח את השיחות.

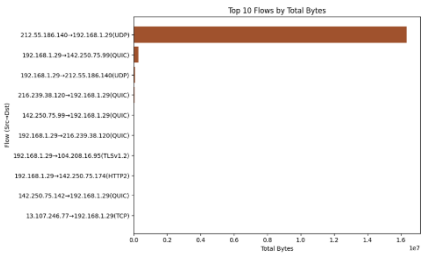
הגרף השביעי-נפח הזרימה:

כעת נציג את הגרף השביעי של כל האפליקציות:

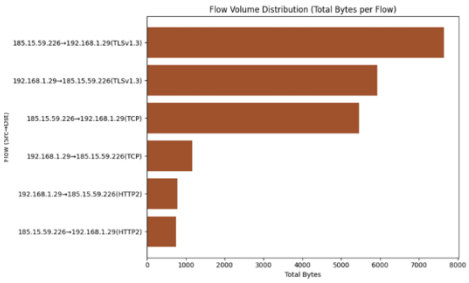
עבור Spotify:



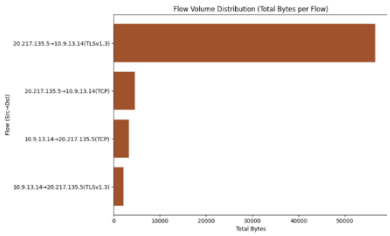
עבור YouTube:



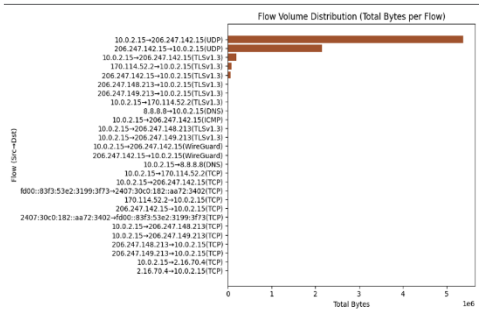
עבור ויקיפדיה(דפדפן chrome):



ב-Microsoft Edge:



עבור Zoom:



נזכיר כי גרף זה הינו מציג את התפלגות נפח הנתונים בכל זרם, לפי סך ה-Bytes שהועברו בכל זרם.

ב-**Spotify** , אפשר לראות כי נפח הנתונים לכל זרימה בינוני(עד bytes60,000), streaming שמע דורש נפח מתון-אך קבוע.

הנתונים מחולקים על פני כמה זרימות ב-QUIC/UDP/TCP , מראה שהאפליקציה מאזנת בין מספר חיבורים.

אין לנו זרימה אחת עצומה פה(כמו שיש ב-YouTube), אלא תעבורה עקבית ומתונה לאורך זמן.

יש פה ריבוי זרימות עם גודל אחיד יחסית-לא תואם גלישה סטנדרטית ב-Chrome.

ב-**YouTube** ישנה זרימה אחת דומיננטית, בגודל עצום של bytes 16,000,000 !, דבר זה אופייני מאוד לstreaming וידאו.

רוב הנתונים מועברים דרך UDP , ו-QUIC , דבר זה מצביע על ב-HTTP3 , ובפרוטוקולים מודרניים.

הזרימה העיקרית גדולה פי 10-100 מהזרימות של Chrom, Spotify , זהו סימן מובהק להזרמת וידאו.

ישנן מעט מאוד זרימות משניות, רובן קשורות לניהול הרשת, ולא לתוכן העיקרי.

ב-**Chrome**, נפח הנתונים לכל זרימה נמוך(עד bytes 8,000), גלישה באינטרנט מתבסס על בקשות קטנות(בניגוד לYouTube , אשר יש זרימות ענקיות).

ישנו שימוש דומיננטי בפרוטוקולים ב-TLS/TCP, וזאת כי גלישה באתרים דורשת הצפנה מאובטחת.

ישנו ריבוי של זרימות קטנות ← כל בקשה לדף חדש יוצרת חיבור נפרד עם כמות נתונים נמוכה, אך עם נתונים מגוונים← דבר זה מתאים לפרופיל של טעינת דפים רבים.

ב-**Microsoft Edge**, אפשר לראות כי הוא מעביר נפח נתונים גדול ביותר דרך TLSv1.3 , הדבר מעיד על חיבור מאובטח, רציף, וממושך לשרתים חיצוניים.

בהשוואה ל-Chrome, אפשר לראות כי שימוש גבוה יותר בהצפנה, עם פחות תקשורת לא מוצפנת.

יש לנו גם שימוש ב-TCP, והוא משמש בעיקר לשלבי ההתחברות← ניתן לראות שנפח הזרימה ב-TCP קטן משמעותית, דבר המרמז על שימוש בו רק בשלבי ההתחברות הראשוניים.

מרגע שהחיבור מתבצע, כל נפח הנתונים עובר דרך TLSv1.3.

ב-**Zoom**, אפשר לראות כי רוב נפח הנתונים מועבר באמצעות UDP , דבר המעיד על שימוש בפרוטוקול זה לצורך תקשורת קול ו-ווידאו בזמן אמת.

UDP מאפשר השהיה נמוכה ושידור מהיר, אך מספק גם תיקון שגיאות כמו ב-UDP, דבר זה אופייני לשירותים כמו שיחות ווידאו או הזרמה מהירה.

קיים נפח משמעותי של תעבורה בפרוטוקול TLSv1.3 , מה שמעיד על שימוש בהצפנה חזקה לצורך ניהול חיבורים מאובטחים, שימוש זה הינו נפוץ לצורך אותות שליטה , ולאימות משתמשים מול שרתי ה-Zoom.

יש לנו נפח נתונים נמוך שעוברים דרך ה-TCP , דבר זה מעיד ש-Zoom אינו מסתמך על TCP לתקשורת קול ו-ווידאו, אלא רק עבור משימות ניהול מסוימות.