זיהוי אפליקציות על ידי תוקף דרך תעבורת רשת מוצפנת

במהלך הניסוי בדקנו עד כמה תוקף יכול לזהות איזו אפליקציה הייתה בשימוש, גם כאשר כל התעבורה מוצפנת. לצורך כך הכנו שני תרחישים אפשריים:

תוקף שמכיר רק את ה־packet size וה־time stamp.

תוקף שמכיר בנוסף גם את (source IP, dest IP, source port, dest port) –flow ID, שמומש אצלנו כ־FlowHash

כאשר לתוקף יש FlowHash, הוא יכול לשייך חבילות לזרימה אחת, ולזהות דפוס זרימה ייחודי לאפליקציה – למשל לפי שילוב של MeanPacketSize, MeanInterarrivalTime, ותדירות של זרימות בין כתובות מסוימות. גם אם התוכן עצמו מוצפן, הדפוסים האלה נשארים גלויים ברמת ה־metadata.

לעומת זאת, תוקף שאין לו FlowHash מקבל תמונה חלקית בלבד. הוא יכול לנסות לנחש את סוג התעבורה (כמו גלישה, סטרימינג או שיחה), אבל מתקשה לזהות אפליקציה מסוימת – במיוחד כשכמה שירותים משתמשים בדפוסים דומים.

במדידה שערכנו:

תוקף עם מידע מלא הצליח לזהות את האפליקציה הנכונה בכ־92% מהמקרים.

תוקף עם מידע חלקי הגיע לדיוק של כ־80% בלבד.

הפער בין המודלים היה בולט במיוחד באפליקציות כמו Chrome ו־Spotify, שמייצרות תעבורה מגוונת או לא יציבה. לעומת זאת, Zoom ו־YouTube זוהו בדיוק גבוה גם ללא FlowHash, בזכות דפוס תעבורה ברור יותר.

שלב ההכנה והאימון

לכל אפליקציה הוקלטו 20 קבצי PCAP, ומתוך כל אחד מהם חילצנו את המאפיינים:

- MeanPacketSize – ממוצע גודל החבילות
- MeanInterarrivalTime – ממוצע זמן בין חבילה לחבילה
- FlowHash – מזהה זרימה - hash (source IP, dest IP, source port, dest port)

נבנו שני סטים נפרדים של נתונים:

עם FlowHash – עבור תוקף עם מידע מלא

בלי FlowHash – עבור תוקף עם מידע חלקי

לאחר מכן אומנו שני מודלים שונים מסוג Random Forest:

Random Forest - הוא אלגוריתם בינה מלאכותית שמייצר הרבה "עצי החלטה" קטנים, ומנבא את התוצאה לפי הרוב. היתרון שלו הוא שהוא פחות רגיש לרעשי רקע – גם אם חלק מהעצים מושפעים מהמידע הרועש, הרוב עדיין מנבא נכון. האלגוריתם יעיל במיוחד לבעיות סיווג עם נתונים מספריים, כי הוא יודע לזהות ולהפריד בין דפוסים כמותיים בצורה מדויקת, גם כשאין קשר ישיר או פשוט בין המשתנים.

כל מודל אומן בנפרד ונשמר לשימוש עתידי. בנוסף, השתמשנו **ב־MinMaxScaler** על כל סט תכונות כדי לנרמל את ערכי הקלט. גם ה־Scaler נשמר, כדי להבטיח התאמה בין נתוני האימון לנתוני התחזית.

MinMaxScaler - כלי שלוקח את כל המאפיינים של הדגימות ומביא אותם לאותו טווח (0 - 1), כדי שהמודל יוכל להשוות ביניהם בצורה הוגנת, וחובה להשתמש באותו Scaler בדיוק גם בזיהוי ולכן הוא נשמר.

```csv
filename,true_label,full_prediction,partial_prediction
chroo10.pcapng,Chrome,Chrome,Chrome
chroo11.pcapng,Chrome,Chrome,Edge
chroo12.pcapng,Chrome,Chrome,Chrome
chroo22.pcapng,Chrome,YouTube,Chrome
chroo8.pcapng,Chrome,Edge,Edge
edd11.pcapng,Edge,Edge,Edge
edd12.pcapng,Edge,Edge,Edge
edd13.pcapng,Edge,Edge,Edge
edd17.pcapng,Edge,Edge,Chrome
edd5.pcapng,Edge,Edge,Chrome
s12.pcapng,Spotify,Spotify,Spotify
s14.pcapng,Spotify,Spotify,Spotify
s19.pcapng,Spotify,Spotify,Chrome
s5.pcapng,Spotify,Spotify,Spotify
s8.pcapng,Spotify,Spotify,Spotify
yoo11.pcapng,YouTube,YouTube,YouTube
yoo12.pcapng,YouTube,YouTube,YouTube
yoo13.pcapng,YouTube,YouTube,YouTube
yoo14.pcapng,YouTube,YouTube,YouTube
yoo8.pcapng,YouTube,YouTube,YouTube
zoo10.pcapng,Zoom,Zoom,Zoom
zoo11.pcapng,Zoom,Zoom,Zoom
zoo12.pcapng,Zoom,Zoom,Zoom
zoo13.pcapng,Zoom,Zoom,Zoom
zoo14.pcapng,Zoom,Zoom,Zoom
```

שלב התחזית והבדיקה

בשלב הבדיקה השתמשנו בקבצי PCAP חדשים שלא נכללו באימון. לכל קובץ חילצנו את אותם מאפיינים בדיוק, והזנו אותם לשני המודלים: האחד עם FlowHash, והשני בלעדיו. כל מודל ניבא את האפליקציה שהכי מתאימה לדפוס שראה.

תוצאת התחזית נרשמה לקובץ CSV, כולל שם הקובץ, תווית אמיתית, התחזית של המודל המלא והתחזית של המודל החלקי. כך ניתן היה להשוות בקלות בין התוקף שמחזיק במידע עשיר לבין זה שמוגבל רק למידע בסיסי.

מסקנות והגנות אפשריות

מהניסוי עולה בבירור שגם כאשר התוכן עצמו מוצפן, מידע מה־metadata של החבילות עדיין יכול להסגיר לא מעט על האפליקציה שנמצאת בשימוש.

תכונות כמו גודל חבילה (packet size), זמן הגעת חבילה (timestamp) ומזהי זרימה (Flow ID) מספיקים לעיתים קרובות כדי לנחש במדויק את סוג השירות או האפליקציה – גם מבלי לדעת דבר על תוכן המידע.

כדי להקשות על תוקפים לבצע זיהוי כזה, יש לנקוט בטכניקות שמטשטשות את דפוסי התעבורה החיצוניים:

- שימוש ב־**VPN** או ב־**multiplexing** כדי לאחד מספר זרימות לתוך חיבור אחד.
- הוספת **padding** שמרחיב או משנה את גודל החבילות כדי להסוות תבניות ברורות.
- שימוש ב־**traffic morphing** שמייצר תעבורה שנראית דומה בין אפליקציות שונות.
- הזרקת **תעבורה דמה (dummy traffic)** שמבלבלת את הניתוח הסטטיסטי.
- לכן, **הצפנה לבדה לא מספיקה כדי להגן על פרטיות המשתמש**. יש להוסיף שכבות הגנה נוספות שמתמודדות גם עם החשיפה האפשרית של metadata – שהיא לרוב חשופה כברירת מחדל.