



## עבודת גמר 5 יח"ל

נושא העבודה : sniffer המשלב תחקור טופולוגית רשת וזיהוי חדירות

שם תלמיד :

ת.ז תלמיד :

שם בית ספר ועיר : קריית החינוך ע"ש עמוס דה-שליט, רחובות

שם המנחה : ערן בינט

מועד הגשה : 1.6.2022

## תוכן עניינים

3	1. מבוא
4	2. תיאוריה
10	3. תוצר סופי
23	4. תהליך כתיבת הפרויקט
28	5. מרכיבי פתרון
29	6. תסריטי בדיקה
36	7. רפלקציה
37	8. הוראות התקנה ותפעול
38	9. ביבליוגרפיה
39	10. נספחים

# 1. מבוא

## 1.1. נושא העבודה

הפרויקט הינו מערכת לניתוח תעבורה רשתית (Network Packet Analyzer), בעלת ממשק משתמש גרפי (Graphic User Interface – GUI). ככל ה Network Packet Analyzers, מטרתה העיקרית של המערכת היא לקלוט את התעבורה ברשת, ולהציג למשתמש ניתוח שלה. בנוסף לכך, המערכת משלבת פונקציונאליות של מערכת לזיהוי חדירות (IDS - Intrusion Detection System), ויודעת לבצע גילוי פאסיבי ואקטיבי של רכיבי הרשת (Active\Passive Network Discovery) ולפענח את המבנה שלה.

המערכת מתבססת על ספריית הקוד הפתוח PcapDotNet, המהווה מעטפת (wrapper) ל-pcap API ב-C#.

## 1.2. מטרת מרכזיות

המטרות המרכזיות שלי בפרויקט מתחלקות למטרות התוצר – המערכת, ולמטרותיי האישיות.

### מטרות התוצר:

- האזנה לתעבורה והצגתה, תוך ניתוח הודעות על פי הפרוטוקולים שלהן
- ריצה בזמן אמת או טעינת קובץ מהקלטה
- זיהוי מתקפות והתרעה עליהן בזמן אמת
- פענוח מבנה הרשת
- שילוב היכולות הללו לכלי יחיד, המאפשר ללמוד על הרשת בצורה נוחה יותר

### מטרות אישיות:

- העמקת את הידע שלי בשפת C# ובתחום התקשורת, בפרט בנושאים שלא נלמדו במגמה
- התנסות בכתיבת פרויקט גדול
- שיפור יכולות התכנות שלי ובממשק משתמש מורכב

## 1.3. רציונל

המוטיבציה שלי לפיתוח הרעיון הינה הרצון ליצור ארגז כלים בתחום התקשורת, המאגד מספר יכולות ומאפשר למידה אינטגרטיבית והתמקצעות בתחום הרשתות – צפייה ויזואלית בתעבורה, פענוח מבנה רשת וזיהוי מתקפות. קיימים בשוק כלים רבים עם יכולות דומות, אך לא קיים כלי יחיד שמאפשר ניהול משותף שלהן. באמצעות הפרויקט אני מקווה להרחיב את הידע שלי בתחום התקשורת, בעיקר בתתי-התחומים שלו שלא נלמדו במגמה. אני מקווה ללמוד אלגוריתמים ושיטות המשמשים ל-passive network mapping, ולהיחשף לדרכים חדשות לדלות מידע מתקשורת. בנוסף, אני מקווה לשפר את כישורי התכנות שלי, ולרכוש ניסון בתחום.

## 1.4. קישור לחומר הנלמד

העבודה מתקשרת לחומר הנלמד בצורות רבות. המערכת כולה מתבססת על רשתות, פרוטוקולי תקשורת, ניתוח תעבורה ברשת ומתקפות, נושאים שנלמדו במהלך לימודי הסייבר בצורה מקיפה.

גילוי הרשת הפאסיבי יתבצע גם הוא על בסיס החומר הנלמד. המערכת עצמה נכתבה ב-C#, ומימושה מכיל מגוון רכיבים שנלמדו במגמה – גישת OOP, שימוש ב-events, שימוש בתכנות אסינכרוני ועוד. בנוסף, הממשק הגרפי של המערכת הוא ממשק WinForms, שנלמד במגמה.

## 2. תיאוריה

### 2.1. תיאוריה

התעבורה ברשת עוברת באמצעות הודעות, המיושמות מעל מספר רב של פרוטוקולים. ניתן לחלק את הפרוטוקולים לשתי קבוצות עיקריות:

1. פרוטוקולים המעבירים מידע, דוגמת TCP ו-UDP. פרוטוקולים האלה משתמשים להעברת מידע בין תהליכים שרצים על גבי מחשבים שונים, במודל שרת-לקוח (client-server model) לדוגמה.
2. פרוטוקולים בקרה, דוגמת ICMP ו-ARP. מטרתם העיקרית של הפרוטוקולים הללו הוא לאפשר את העברת המידע על ידי הפרוטוקולים מהקבוצה השנייה.

מנתח תעבורת רשתית משמש לפענוח המידע שמועבר באמצעות הפרוטוקולים השונים בצורה שתיתן למשתמשים להבין את המתרחש ברשת. לצורך תפקודו, נדרשים לו יכולות בתחום:

- הסנפה של תעבורה רשתית ברשת מקומית, ובכלל זאת להאזין לתעבורה ברשת, ולהיות מסוגל לסנן את התעבורה על פי בקשת המשתמש, כדי לספק לו תמונת מצב ממוקדת.
- ניתוח תעבורה, ובכלל זאת גילוי רשת, ניתוח חסר הקשר ותלוי מצב
- ניהול קבצים, ובכלל אלו, היכולת לשמור הודעות (Dump File) ולקרוא הודעות מקובץ.
- ניהול מקבילי של משימות על מנת לא לאבד מידע

#### 1) הסנפה של תעבורה רשתית

המערכת עושה שימוש בספריית PcapDotNet (API הכתוב ב-C, המאפשר האזנה להודעות ושליחתן). Drivers שונים שמטרתם לאפשר לתוכנה להאזין להודעות ולשלוח הודעות מממשים את pcap API. אחד ה-Drivers הללו הוא Npcap, שבו משתמשת המערכת שלי. מאחר וה-API כתוב בשפת C, השתמשתי בספרייה PcapDotNet שמאפשרת להשתמש ב-Driver מתוך קוד בשפת C#.

כלים המבצעים ניתוח ברשת, משתמשים בשתי דרכים עיקריות לסנן תעבורה. הדרך הראשונה נקראת capture filter ומתבססת על קלט של הודעות שעונות לקריטריונים מסוימים. בדרך זו, ההודעות מסוננות ברמת ה-Driver. שיטה זו יעילה יותר, אך מאפשרת לסנן הודעות רק על בסיס שדות בודדים של פרוטוקולים בשכבות הקו (Link Layer), בשבת הרשת (Network Layer) ובשכבת התעבורה (Transport Layer). הדרך השנייה נקראת display filter ומתמקדת בקלט מלא ותצוגה חלקית של ההודעות, על פי מענה לקריטריונים מסוימים. שיטה זו יעילה פחות, אך מאפשרת סינון על בסיס מספר גדול מאוד של קריטריונים בכל השכבות.

## 2) ניתוח תעבורה - Network Analysis

גילוי (או חקירת) רשת הוא קשת רחבה של טכניקות, בהן נעשה שימוש בתעבורה כדי ללמוד על הרשת, דוגמת מיפוי הטופולוגיה שלה. שתי טכניקות נפוצות הן זיהוי גרסת מערכת ההפעלה של מחשב מסוים (fingerprinting) וסריקת פורטים (port scanning). גילוי רשת מתחלק לשני סוגים, גילוי רשת אקטיבי וגילוי רשת פאסיבי. בניגוד לגילוי רשת אקטיבי, גילוי רשת פאסיבי נעשה ללא שליחת הודעות. חסרון של גילוי רשת פאסיבי הוא שהוא מאפשר לדלות פחות מידע, אך מנגד קשה יותר לזיהוי, ואינו מפריע למחשבי הרשת. Fingerprinting יכול להתבצע על ידי מגוון אלגוריתמים, ולהתבסס על פרוטוקולים שונים, ביניהם TCP ו-ICMP. כמו מיפוי רשת, fingerprinting יכול להתבצע באופן אקטיבי ובאופן פאסיבי. דוגמה לכלי המבצע passive fingerprinting הוא p0f, בעוד ש-Nmap מבצע אותו באופן אקטיבי. העקרון המנחה של טכניקה זו זיהוי התנהגות הקשורה לתעבורה המייחדת מערכת הפעלה ספציפית, או לעיתים אפילו גרסה מסוימת שלה.

מיפוי טופולוגי של הרשת, שאליו אתייחס בתור פענוח מבנה הרשת, הוא זיהוי המחשבים ברשת (על פי כתובות IP או MAC) והחיבור ביניהם. מיפוי הרשת הפשוט ביותר הוא מיפוי כתובות ה-MAC של המחשבים ב-LAN. לכך ניתן להוסיף שיוך של כתובות IP לכתובות ה-MAC, זיהוי כתובת ה-MAC של הנתב (router) וזיהוי שרתי DHCP. מיפוי מסובך יותר של ה-LAN כולל זיהוי של גשרים (bridges) מתגים (switches) והמחשבים המחוברים אליהם. מיפוי הרשת מחוץ ל-LAN הוא מסובך יותר, והוא כולל זיהוי של כתובות IP של מחשבים ושל הנתבים שדרכם הם מחוברים. גילוי של כתובות MAC מחוץ ל-LAN אינו אפשרי, מפני שמחשבים ב-LANs שונים מתקשרים ביניהם מרמת הרשת ומעלה.

גילוי רשת פאסיבי מאפשר רק מיפוי של ה-LAN, וזיהוי מחשבים שנמצאים מחוצה לו. לעומת זאת, מיפוי רשת אקטיבי מאפשר להשיג מידע נוסף. באמצעות הודעות DHCP-DISCOVER בפרוטוקול DHCP (הנשלחות כהודעות broadcast) ניתן לגלות שרתי DHCP. באמצעות פרוטוקול IRDP (ICMP Router Discovery Protocol) ניתן לגלות נתבים. באמצעות SNMP (Simple Net Management Protocol) ניתן לגלות מתגים ולהשיג את טבלאות ה-CAM (Content Addressable Memory Table) שלהם, ולהשיג את טבלאות הניתוב (Routing Tables).

המערכת שלי מאפשרת גילוי רשת פאסיבי וגילוי רשת אקטיבי בסיסיים. כפי שיפורט בהמשך, המערכת מזהה מחשבים ושרתים ב-LAN ומחוצה לו באמצעות גילוי רשת פאסיבי. בנוסף, היא יכולה למפות את המחשבים בדרך למחשב מסוים באמצעות גילוי רשת אקטיבי.

נהוג להתייחס למספר סוגי ניתוח מצב, הבולטים ביניהם ניתוח חסר הקשר ותלוי מצב.

ניתוח חסר הקשר (Stateless Analysis) הוא שיטת ניתוח שמנתחת כל הודעה כעומדת בפני עצמה. שיטה זו פשוטה למימוש, מפני שהיא מפשטת את העיבוד של כל הודעה. למרות זאת, לשיטה זו מגבלות רבות מאוד, מפני שלעיתים קרובות ניתן ללמוד רבות על הרשת על ידי שילוב המידע של מספר הודעות.

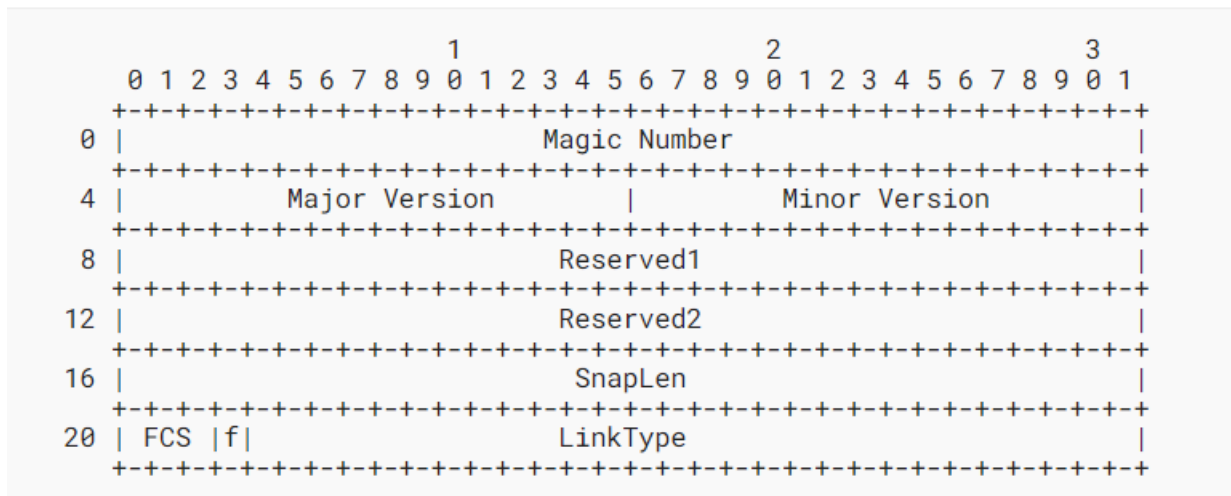
ניתוח תלוי מצב (Stateful Analysis) הוא שיטת ניתוח שמנתחת הודעה מסוימת תוך התחשבות במידע מהודעות קודמות. לצורך כך, המערכת שמבצעת Stateful Analysis צריכה לשמור מידע רלוונטי לגבי הודעות, ולהשתמש בו מאוחר יותר בעת ניתוח ההודעות הבאות. שיטה זו מסובכת יותר למימוש והניתוח בה ארוך יותר, אך היא מאפשרת לדלות הרבה יותר מידע משיטת הניתוח חסר המצב. שיטה זו יעילה משמעותית בחסימת וזיהוי מתקפות, מפני שחלק גדול מאוד מהן מורכב מיותר מהודעה אחת, או ניתן לזיהוי אך ורק בהשוואה לתקשורת תקינה.

המערכת שלי עושה שימוש בניתוח תלוי מצב עבור פרוטוקולים מסוימים לצורך הפקת מידע, זיהוי נתבים וזיהוי מתקפות.

### (3) ניהול קבצים

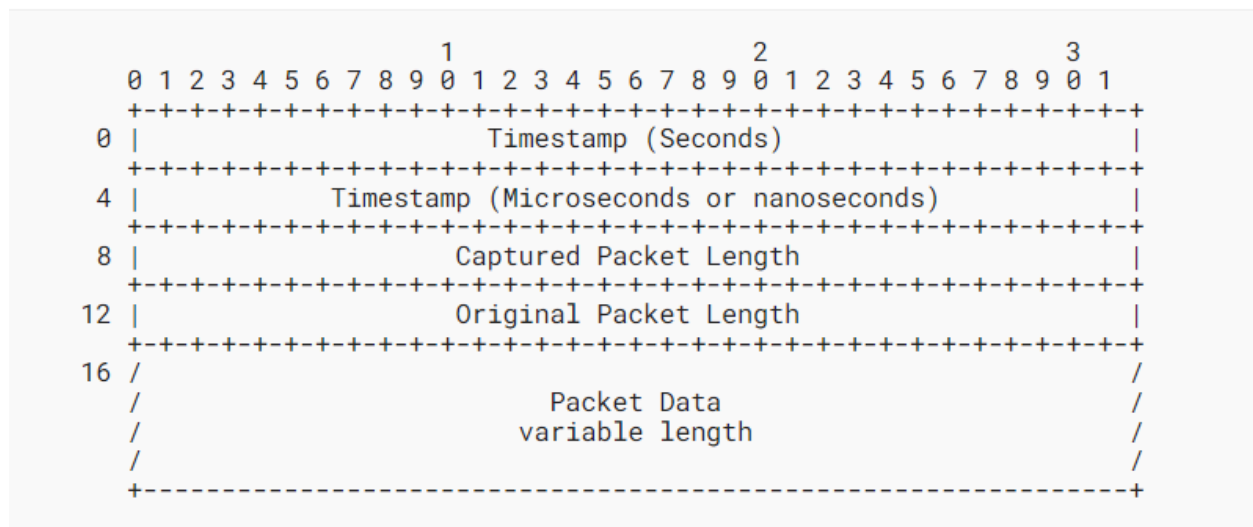
שני פורמטים חשובים ונפוצים מאוד לשמירת קובצי הקלטה הם פורמט pcap, ופורמט pcapng. פורמט pcap פשוט מאוד ביחס לפורמט pcapng, אך גמיש פחות. בנוסף, ניתן להוסיף לקבצים בפורמט pcapng מידע חשוב שלא ניתן לשמור בקבצי pcap.

קובץ בפורמט pcap מורכב מכתרת (header) ומרשומות (records) של הודעות. תרשים 1 מציג את השדות שמכילה הכותרת. קובץ pcap חייב להתחיל באחד במספרי הקסם (magic number) 0xA1B2C3D4 או 0xA1B23C4D. הכותרת מכילה את גרסת פורמט ה-pcap, את סוג פרוטוקול שכבת הקו, אורך ההודעה המקסימלי בקובץ, ואת אורך רצף בדיקת הפריים (Frame Check Sequence; FCS) ב-Bytes. לאחר הכותרת מופיעות הרשומות של ההודעות.



תרשים 1: כותרת (Header) של פורמט pcap.

תרשים 2 מציג את השדות של כל רשומה. כל רשומה מכילה את הזמן שבו נלכדה ההודעה, את אורך ההודעה המקורי, את אורך ההודעה שנלכדה ולאחר מכן את ההודעה עצמה. חשוב לציין שקובץ pcap יכול להכיל רק הודעות מאותו ה-interface. פורמט זה מאפשר עבודה פשוטה עם קבצי pcap, אך אינו גמיש כלל וכלל.



**תרשים 2:** רשומה (Record) של פורמט pcap. שני התרשימים נלקחו מתוך:

<https://tools.ietf.org/id/draft-gharris-opsawg-pcap-00.html>

בניגוד לקובץ pcap המכיל אך ורק header ו-records, קובץ בפורמט pcapng בנוי מבלוקים. כל בלוק מכיל את סוג הבלוק, אורך הבלוק ואת גוף הבלוק. סוג הבלוק הוא אחד מסוגי הבלוקים הבסיסיים, דוגמת בלוק הודעה פשוט (Simple Packet Block; SPB) או בלוק הודעה משופר (Enhanced Packet Block; EPB) או בלוק מותאם (Custom Block; CB). קובץ חייב להתחיל בבלוק כותרת קטע (Section Header Block; SHB). בנוסף, קובץ יכול להכיל יותר מקטע אחד, כשכל קטע מתחיל ב-SHB. סוג חשוב נוסף של בלוק הוא בלוק תיאור ממשק (Interface Description Block; IDB). בעזרתם, pcapng מאפשר לשמור הודעות ממשקים שונים – בלוקים מסוג EPB מכילים שדה המפנה לממשק המתואר על ידי בלוק מסוג IDB.

כפי שניתן לראות, פורמט pcapng מורכב יותר מפורמט pcap, ומאפשר לשמור יותר מידע. המאפיינים הללו מאפשרים ל-pcapng לענות על מספר מגבלות של פורמט pcap, בהן שמירת הודעות שהתקבלו מ-interfaces שונים ותמיכה בהערות. אולם, היתרון הבולט של pcapng הוא הגמישות ויכולת התרחבות שלו – הפורמט בנוי כך שחברות צד שלישי יכולות לשמור בקבצים שלו מידע נוסף, בלי לפגוע בקריאות שלהם על ידי תוכנות שלא מסוגלות לפענח מידע זה. למרות זאת, פורמט pcapng עדיין מאפשר שמירת הודעות בלבד בצורה דומה ל-pcap, באמצעות אוסף של SPBs. יתרון נוסף של pcapng הוא שבניקוד לפורמט pcap, concatenation של שני קבצי pcapng תקינים נותן קובץ pcapng תקין. מהיתרונות של הפורמט נובע גם החיסרון שלו עבור פרויקט בקנה מידה קטן, שהוא הקושי לפתח כלים לעבודה איתו. מסיבה זו, אין למנתח תעבורה רשתית פשוט, דוגמת המערכת שלי, דרישה להשתמש בפורמט pcapng, חוץ מתאימות עם לתוכנות כגון Wireshark.

(4) ניהול מקבילי של משימות

מקביליות היא היכולת של תוכנה לבצע מספר פעולות במקביל. יכולת זו חשובה מאוד, מפני שלעיתים קרובות על תוכנה לבצע מספר משימות ארוכות, דבר שאם יתבצע באופן טורי יגרום לתוכנה לאבד

מידע על אירועים שקורים במהלך ביצוע המשימות או יהפוך אותה לבלתי ניתנת לשימוש. שכך, הקצב המהיר שבו הודעות מגיעות, ויתר על כן, הפיענוח שלהן שיכול להיות לעיתים איטי יחסית, מחייב מקביליות משימות. בנוסף, מקביליות חשובה מאוד גם בתוכנות המשלבות GUI, שנדרשות להגיב למשתמש גם בעת ביצוע מטלות אחרות.

הדרכים העיקריות להשגת מקביליות הן:

- שימוש במספר תהליכים (Multiprocessing). דרך זו ככל הנראה מאפשרת את להשיג את המקביליות הגבוהה ביותר, אך קיים קושי לא מבוטל בשיתוף והעברת מידע בין תהליכים שונים (IPC – Inter Process Communication).
- שימוש במספר threads (Multithreading). יתרון מהותי של דרך זו הוא שה-threads השונים יכולים לגשת לאותם עצמים השמורים ב-heap.

חיסרון של מקביליות הוא שנוצר צורך לסנכרן בין תהליכים וה-threads שונים. לדוגמה, לא סנכרון - שני threads שמשנים מבנה נתונים כלשהו במקביל יכולים לפגוע בתקינות המידע שמוכל בו, וליצור התנהגות בלתי צפויה. חלק מהדרכים לסנכרון threads הינם מנעולים (locks), mutexes ו- semaphores. הדרכים הללו מאפשרות רק ל-thread יחיד (או למספר מוגבל של threads) לבצע קטע קוד מסוים, וכך למנוע התנהגות בלתי צפויה. סנכרון מציב אתגרים בפני עצמו (מניעת deadlocks למשל), אך במרבית המקרים לא ניתן לכתוב תוכנה תקינה בלעדיו.

שימוש ב-tasks: בניגוד ל-threads שמנוהלים על ידי מערכת ההפעלה ורצים באופן עצמאי יחסית, Tasks הן פעולות קצרות המנוהלות על ידי .Net. הן מבוצעות ב-threads של ה-thread pool. tasks קלים יותר, ויש APIs רבים של .Net. שמקילים מאוד על השימוש בהם. ניתן לסנכרן tasks באותם מנגנונים כמו threads, ואין הבדלים רבים בהתנהגות שלהם. יתרון של threads על tasks הוא שהשליטה ב-threads שנוצרו על ידי התהליך גבוהה יותר.

ה-thread pool הוא אוסף של background threads שיש לתהליך. קיים תור (Queue) של ה-threads הפנויים, והתהליך יכול לבצע פעולה כלשהי באחד מהם. לאחר שה-thread סיים את הפעולה, הוא חוזר לתור, והתהליך יכול להשתמש בו שוב. Tasks יעילים יותר מפני שהשימוש בהם לא דורש יצירת threads חדשים.

ה-syntax של C# מפשט את השימוש ב-tasks באמצעות המילים השמורות (keywords) async ו-await. await באמצעות await ניתן לחכות לתוצאה של פעולה שמחזירה task בצורה אסינכרונית. כלומר, למרות שה-thread מחכה לתוצאות הפעולה, הוא עדיין יוכל להגיב ל-events חיצוניים. async חייבת להופיע בכותרת (signature) של כל הפעולות המשתמשות ב-await, ורק בהן.

## [2.2 מוצרים קיימים](#)

מוצרים דומים בשוק הם:

**Wireshark** <https://www.wireshark.org>, הוא רחרחן (sniffer) ומנתח פרוטוקולים (protocol analyzer) שנמצא בשימוש נרחב. ל-Wireshark יכולות רבות – האזנה לתעבורה,



שמירת קבצי הקלטה ופתיחתם, ניתוח מפורט מאוד של פרוטוקולים רבים ויצירת סטטיסטיקה. אחד היתרונות העיקריים של Wireshark הוא הגנריות שלו. Wireshark תומך בשמירת ופתיחת קבצי הקלטה בפורמטי pcapng, pcap, Catapult DCT2000, Cisco Secure IDS iplog, ורבים נוספים. הוא יכול לעבוד בסביבת Windows, Linux, macOS ומערכות Unix-like אחרות. הוא יכול להאזין לתעבורה מעל Ethernet, IEEE 802.11, PPP/HDLC, ATM, Bluetooth, USB, Token Ring ופרוטוקולים נוספים. זאת ועוד, ל-Wireshark יש GUI מפותח וידידותי למשתמש.

**Metasploit** <https://www.metasploit.com>, הוא כלי לבדיקת חולשות (vulnerabilities) ונוזקות (exploits). מטרת Metasploit היא לגלות חולשות ברשימת מטרות, באמצעות השוואה למאגר. בנוסף, Metasploit מאפשר שיגור של נוזקות, שמסוגלים לבדוק את הנזק הפוטנציאלי מהחולשות שהתגלו. MetaModules הם כלים לאוטומטיזציה של משימות אבטחה. אחד מהם הוא <sup>1</sup>Discovery Passive Network, המאפשר למפות רשת בצורה פאסיבית על פי הקלטה, ללא שליחת הודעות. הוא עושה זאת באמצעות הפקת מידע מהודעות ARP ו-DHCP.

**Snort** <https://www.snort.org> הוא מערכת לגילוי חדירות (Intrusion Prevention System; IPS) שיכול לפעול גם בתור רחרחן ורשם הודעות (packet logger) - כלי המאפשר לשמור את התעבורה ברשת, ללא פענוח שלה. בתור sniffer, Snort מציג את ההודעות אותן הוא מקבל בזמן אמת. בתור packet logger, Snort שומר את ההודעות על הדיסק כקובץ טקסט בפורמט ASCII או כקובץ בינארי בפורמט pcap. בתור IPS, Snort פועל פי חוקים שהוגדרו לו על ידי המשתמש. המרכיבים העיקריים של החוקים של Snort הם כתובות מקור וכתובות היעד, פרוטוקול ופעולה. שתיים מהפעולות האפשריות הן חסימת ההודעה או אישור ההודעה. בגרסתו הנוכחית, Snort תומך ב-TCP, UDP, ICMP ו-IP בלבד. החוקים של Snort יכולים לכלול תנאים גם על מטען (payload) ההודעה באחד מהפרוטוקולים הללו.

הייחוד של המערכת שלי, הוא שהיא משלבת את היכולות של הכלים הללו (מלבד חסימת המתקפות ש-Snort מאפשר) בכלי יחיד, שקל ונוח יותר יהיה להשתמש בו. בנוסף, המערכת שלי תוכל לבנות את מבנה הרשת באופן פאסיבי בזמן אמת. היכולות הללו מוצגות בטבלה הבאה:

המערכת שלי	Metasploit *	Snort	Wireshark	
פרוטוקולי שכבת קו נתמכים	פרוטוקולים רבים	פרוטוקולים רבים	פרוטוקולים רבים	
רחרחן	כן	כן	כן	
רשם הודעות (logger packet)	כן	כן	כן	

<sup>1</sup> [https://www.rapid7.com/globalassets/external/docs/download/MS\\_pnd\\_qsg.pdf](https://www.rapid7.com/globalassets/external/docs/download/MS_pnd_qsg.pdf)

פרוטוקולים נתמכים	פרוטוקולים רבים מאוד (בתור מנתח פרוטוקולים)	IP, TCP, UCP ו-ICMP (בתור IDS)	ARP ו-DHCP בלבד (בתור מגלה רשת)	מספר פרוטוקולים (בכל התפקידים)
פענוח מבנה הרשת	לא מיועד	לא מיועד	לאחר ביצוע ההקלטה בלבד	online ומתוך הקלטה
זיהוי מתקפות	לא מיועד	על פי סט חוקים שניתן לשינוי	לא מיועד	סט חוקים קבוע בלבד
חסימת הודעות על פי אוסף חוקים	לא מיועד	על פי סט חוקים שניתן לשינוי	לא מיועד	לא מאפשר

\* Metasploit - Passive Network Discovery MetaModule

[טבלה 1: מוצרים דומים](#)

### 3. תוצר סופי

#### 3.1. תיאור הפרויקט

המערכת הינה sniffer המשולב עם כלי ל-passive network discovery ו-NIDS (Network based Intrusion Detection System). המערכת מציגה בפני הממשק את התעבורה ברשת, מאפשרת לו לשמור ולפתוח קבצי הקלטה, מפענחת את מבנה הרשת באופן פאסיבי, ומתריעה על מתקפות. בנוסף, המערכת מציגה את טבלת ה-ARP וטבלת הניתוב של המחשב עליו היא רצה בצורה גרפית.

אחד הדגשים העיקריים בפרויקט הוא נוחות המשתמש – המערכת מאפשרת ניהול משולב של מגוון פונקציונליות הקשורות ברשת באמצעות ממשק משתמש גרפי.

המערכת מאפשרת מספר פעולות:

- האזנה לתעבורה ברשת וניתוח פרוטוקולים – המערכת מציגה את המידע המועבר בפרוטוקולי ניהול (דוגמת ARP ו-DHCP), ובשדות הבקרה (דוגמת IP ו-TCP)
- שמירת קבצי הקלטה בפורמט pcap ופתיחתם
- פיענוח מבנה הרשת על פי התעבורה בה בזמן אמת או מתוך הקלטה
- זיהוי מתקפות בזמן אמת או מתוך הקלטה והתרעה עליהן באמצעות הודעה מתאימה וסימון ההודעה או ההודעות התוקפות בצבע בולט. המערכת מזהה את מחשב הקורבן, ומתריעה בצורה מיוחדת אם מדובר במחשב עליו היא רצה.
- הצגת ואפשרות עריכה של טבלת ה-ARP של המחשב עליו המערכת רצה
- הצגה ואפשרות עריכה של טבלת הניתוב (Routing Table) של המחשב עליו המערכת רצה.

תצוגת המערכת מיושמת מעל WinForms, וניתן לתצוגה דגש חשוב. בעת שימוש במערכת, המשתמש יוכל לפתוח קובץ קיים, או להתחיל הקלטה חדשה. בשני המצבים, המשתמש יוכל לבחור האם להשתמש ב-capture filter או לא. עבור הקלטה חדשה, המשתמש יוכל לציין את מספר ההודעות המקסימלי שהוא רוצה לקלוט, והאם את כל התעבורה ברשת (Promiscuous Mode) או רק את ההודעות שנשלחו על ידי המחשב שלו או אליו (זה המצב הרגיל בו פועלת מערכת ההפעלה).

המערכת תציג את ההודעות שנקלטו בצורה טבלאית, ובכלל זאת את מספר ההודעה, זמן הקבלה שלה, את היעד והמקור שלה, את הפרוטוקול שלה (מבין הפרוטוקולים הנתמכים על ידי המערכת) ותיאור קצר של תכולת ההודעה. בנוסף, המשתמש יוכל לצפות בתוכן הבינארי של ההודעה ולציין display filter, כך שמערכת תציג רק את ההודעות שמתאימות לו. במהלך ההקלטה, המשתמש יכול לגשת למידע סטטיסטי לגבי התעבורה. אפשרות זו פתוחה גם עבור קריאת הודעות מקובץ.

בשני מצבי הפעולה, המערכת מזהה את המחשבים ב-LAN ומחוצה לו. המערכת מנסה לזהות אילו מבין שמחבים ב-LAN הם נתבים או שרתי DHCP. בנוסף, המערכת תנסה לזהות גם שרתי DNS מחוץ ל-LAN. במהלך הקלטה, המערכת יכולה

- למפות את המחשבים בדרך למחשב מסוים מחוץ ל-LAN.
- להציג את המידע לגבי חיבורי ה-TCP (TCP streams) של המחשבים שזיהתה.
- להציג את כל המחשבים שמחשב מסוים מחובר אליהם, את הפורטים שזרכם מתבצע החיבור, ואת כמות המידע שכל מחשב שלח. אפשרות זו פתוחה עבור גם עבור קריאת הודעות מקובץ

עבור שני מצבי הריצה (טעינת הקלטה או האזנה בזמן אמת), המערכת תזהה מתקפות. המערכת תסמן את ההודעות המעורבות במתקפה בצבע בולט, ותתריע על המתקפה למשתמש. לאחר שהמשתמש יבחר בהודעה החשודה כמתקפה, הוא יוכל לצפות במידע עליה – סוג המתקפה, מספרי ההודעות התוקפות, וכתובות התוקף והקורבנות. חלון נפרד יציג log המכיל את המידע הזה לגבי כל המתקפות ברשת.

המשתמש יוכל להתחיל ההקלטה חדשה, להתחיל את ההקלטה מחדש, או לעצור אותה. לאחר עצירת ההקלטה, המשתמש יוכל לשמור אותה בקובץ pcap. המשתמש יוכל לבחור display filter, כך שרק הודעות שעונות עליו ישמרו בקובץ.

בנוסף, המערכת מכיל מספר utilities נוספים. היא מאפשרת להציג ולערוך את טבלת ה-CAM של המחשב עליו היא רצה, להציג את טבלת הניתוב שלו, ואת המידע לגבי ה-interfaces שלו. היא עושה זאת באמצעות פקודות route, arp ו- ipconfig בהתאמה.

## 3.2. אלגוריתמים עיקריים

### 1. Sniffing

ההאזנה תתבצע ב-thread נפרד, ותמומש בתור task שיריוץ ב-thread pool של התהליך. עיבוד ההודעה, שיכלול ניתוח שלה על פי פרוטוקולים, זיהוי מתקפות ופענוח מבנה הרשת יבוצע בצורה אסינכרונית, גם בתור tasks ב-thread pool. ה-thread pool מנוהל על ידי Net, ולכן נוח מאוד וגם בטוח להשתמש בו. כמות ה-threads ב-thread pool תלויה בכמות הזיכרון הווירטואלי

ובפרמטרים נוספים, והיא יכולה להגיע ל threads רבים. לאור זאת, ה-delay בין קבלת ההודעה והפענוח שלה והצגתה יהיה נמוך מאוד. בנוסף, לא יהיה delay בין פעולות המשתמש לביצוע שלהם על ידי האפליקציה.

ההאזנה בתוך ה-thread מתבצעת באמצעות הפעולה ReceivePackets של עצם מהמחלקה PacketCommunicator מתוך PcapDotNet.Core. הפעולה מקבלת את מספר ההודעות שיש ללכוד, ו-delegate מסוג HandlePacket, שמקבל הודעה ולא מחזיר כלום (void), שבו אמור להימצא עיבוד ההודעה. הפעולה מבצעת את ה-delegate עבור כל הודעה שהיא קיבלה. אם הפעולה מקבלת 0 כמספר ההודעות שיש ללכוד, היא תמשיך לרוץ ללא הפסקה. בגלל שהפעולה חוסמת (blocking), היא מורצת בתור task ב-thread pool של התהליך. מאחר ועיבוד ההודעה עלול לקחת זמן, וכדי שלא יהיה אובדן הודעות, ה-delegate שמועבר לפעולה מעביר את ההודעה לעיבוד ב-thread אחר ב-thread pool.

## 2. Packet Display

המערכת מיושמת מעל WinForms. המערכת מורכבת מ-UI thread, ומבצעת את ההאזנה לתעבורה ואת עיבודה ב-threads נפרדים ב-thread pool של התהליך בו היא רצה. שיטה זו תשאיר את ה-UI thread פנוי כדי להגיב לפעולות המשתמש. המערכת תשתמש ב-Timer כדי לעדכן את תצוגת הסטטיסטיקה והתצוגות המתעדכנות הנוספות, וב-events כדי לעדכן את מבנה הרשת ואת המתקפות המזוהות. הבחירה ב-Timer לעדכון הסטטיסטיקה והתצוגות הנוספות נעשתה בגלל קצת העדכון המהיר של הנתונים.

כל ההודעות שמורות ב-list ארוך, אך ל-UI אין גישה ישירה אליו. לכן, המערכת מציגה את ההודעות באמצעות ListView. המערכת יוצרת ListViewItem עבור כל הודעה, ומוסיפה אותו לאוסף ListView.Items על פי מספר ההודעה שמוצגת בו. לאחר שניתוח ההודעה מסתיים, ה-ListViewItem מתעדכן בהתאם. פלטפורמת WinForms היא זאת שמציגה את ה-ListViewItems הרלוונטיים מתוך כלל ה-ListViewItems בתוך ListView.Items.

## 3. Intrusion Detection

כדי לזהות מתקפה, המערכת תבדוק אם הודעה או אוסף הודעות עומדות בתנאים מסוימים. המערכת תשתמש בניתוח תלוי מצב (stateful analysis). לצורך כך, המערכת תצטרך לדעת להפיק מהודעות בפרוטוקולים הללו מידע ולשמור אותו. המערכת יודעת לזהות מתקפת ARP man-in-the-middle ומתקפת TCP Syn flood.

כדי לזהות את מתקפת ARP MitM, המערכת בונה טבלת CAM משוערת עבור כל מחשב, על פי תשובות ה-ARP שהוא קיבל. המערכת תדע שמחשב מסוים נמצא תחת מתקפה זו אם היא מזהה הודעה לפיה מחשב עם כתובת MAC מסוימת טוען בפני מחשב א' שהוא מחשב ב', כאשר מחשב ב' חושב שכתובת ה-MAC הזו משויכת לכתובת ה-IP של מחשב א'. המערכת יודעת

לאחזר את ההודעה שגרמה למחשב ב' לרשום זאת בטבלת ה-CAM שלו, ומזהה את שני ההודעות כמתקפה.

כדי לזהות מתקפת Syn Flood, המערכת בודקת אם למחשב מסוים יש כמות מסוימת של חיבורי TCP שמי שיזם אותם לא הגיב עליו יותר אחרי שלב ה-Syn. המערכת מזהה את ההודעות שמכילות את ה-Syn כמתקפה אם עבר זמן סביר והם עדיין לא נענו, ואם הן נשלחו בתוך פרק זמן קצר מסוים.

השוני בין הפרמטרים לזיהוי המתקפות נובעים מכך שמתקפת Syn Flood היא מתקפת DoS/DDoS, בניגוד ל-MitM.

#### 4. Export & Import

המערכת יודעת לשמור את ההקלטות שהיא מבצעת בפורמט pcap ולפתוח קבצים בפורמט זה, על ידי שימוש בפונקציית ייעודיות של הספרייה PcapDotNet. שמירת הקבצים מסווגת למשתמש שיצר אותם.

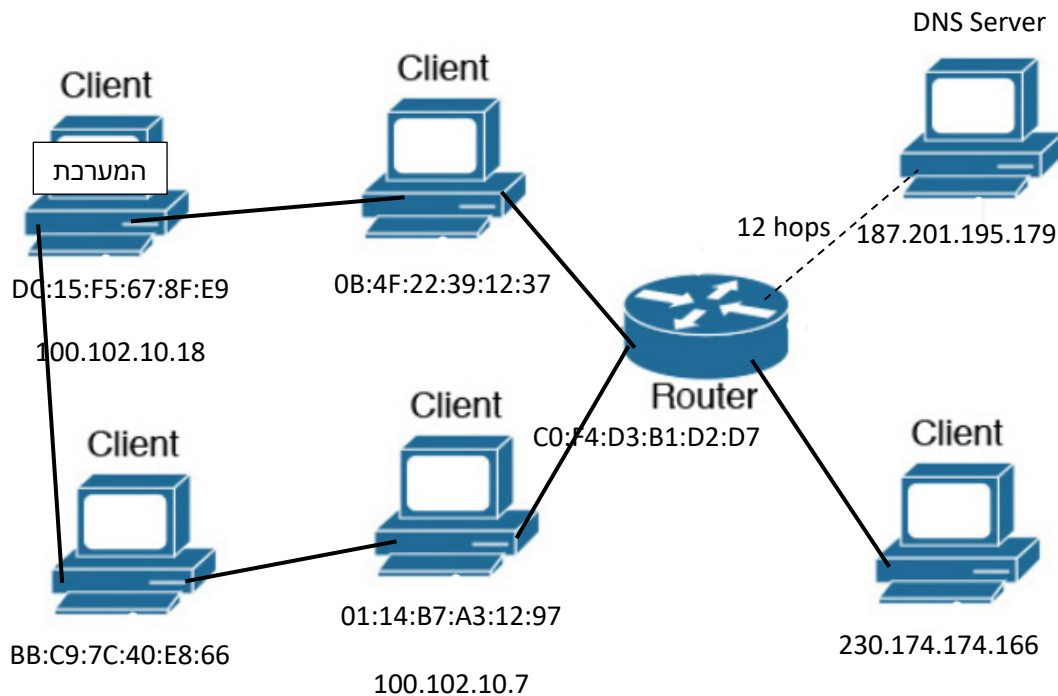
המערכת מאפשרת להגדיר משתמשים, ונותנת לכל משתמש את האפשרות לשמור קבצים שיהיו שייכים רק לו. פתיחת הקבצים הללו תתאפשר רק עם שם משתמש וסיסמה. שמירת הקבצים הללו תתבצע באמצעות System.IO.IsolatedStorage, כך שגם לתוכנות חיצוניות לא תהיה לקבצים הללו.

Isolated Storage הוא מנגנון של Windows שנועד לאפשר לתוכנות לשמור מידע בצורה סטנדרטית, פרטית ומאובטחת. Isolated Storage מאפשר לתוכנה לשמור קבצים ללא חשש שתוכנות אחרות יפגעו בו או ישנו אותו. ה-Isolated Storage מאפשר הפרדה במספר רמות. ראשית, רק ה-User (של Windows) שהריץ את התוכנה ששמרה את המידע ב-Isolated Storage יוכל לגשת אליו. בנוסף, ניתן להגביל את הגישה למידע על פי Assembly ו-Application Domain. הפרדה לפי Application Domain מאפשרת רק לתוכנה ששמרה את המידע לגשת אליו, והפרדה על פי Assembly מאפשר רק לקובץ ה-.exe או ל-dll. שיצר את הקבצים לגשת אליהם. ניתן ליצור הפרדה על פי User ו-Assembly, ועל פי User, Application Domain ו-Assembly. תוכנה צריכה הרשאה מתאימה על מנת להשתמש במנגנון ה-Isolated Storage, שניתנת בברירת מחדל לכל תוכנה שמורצת ב-.Net.

#### 5. Passive network discovery

התפקיד העיקרי של המערכת בתור מנתח טופולוגית רשת הוא לזהות את המחשבים ב-LAN על פי כתובות MAC וכתובות IP. זיהוי כתובת ה-MAC של מחשב ב-LAN פשוטה ביותר מאחר והיא מופיעה ב-Ethernet header, אך השיוך שלה לכתובות IP מורכב יותר. המערכת תבצע שיוך זה ופעולות נוספות על בסיס מספר פרוטוקולים – ARP ו-DHCP. בנוסף, היא תנסה לדלות מידע מכתובות ה-MAC, מכתובות ה-IP ומשדות ה-TTL בפרוטוקול IP. הודעות ARP והודעות DHCP מכילות שיוך ישיר בין כתובות MAC לכתובות IP. שיוך זה יכול להיות מושג גם על ידי זיהוי הודעת IP עם ערך TTL שלא החסירו ממנו אף פעם.

בנוסף, המערכת תזהה מחשבים בעלי חשיבות ב-LAN ומחוצה לו. ב-LAN, המערכת תנסה לזהות נתבים (Routers) ושרתי DHCP. מחוץ ל-LAN, המערכת תנסה לזהות שרתי HTTP/S ושרתי DNS. המערכת תזהה את מספר הקפיצות אליהם באמצעות ההפרש בשדה ה-TTL.



### תרשים 3: דוגמה למידע שתוכל המערכת להשיג

בתרשים 3 מוצגים 4 מחשבים ב-LAN, ביניהם המחשב שעליו מופעלת המערכת, שני מחשבים שמערכת זיהתה על פי כתובות MAC, ומחשב שהיא זיהתה גם על פי כתובת IP. ייתכן שיש ב-LAN מחשבים נוספים, אך למערכת אין דרך לדעת עליהם. המערכת זיהתה את כתובת ה-MAC של הנתב, כתובת IP של המחשב המחובר ישירות לנתב ממשק דרך ממשק אחר, וכתובות IP של שרת DNS הנמצא במרחק 12 קפיצות.

### Active network Mapping

המערכת יכולה לבצע tracert כדי לגלות את ה-router ב-LAN, ואת המחשבים בדרך למחשב מסוים. tracert מתבצע על ידי שליחת הודעות ICMP echo למחשב היעד, עם TTL שמשתנה בקפיצות של 1, עד לקבלת תשובה. מאחר וכל נתב מוריד את ערך שדה ה-TTL ב-1, כל מחשב בדרך יקבל פקטת ICMP שה-TTL שלה יהיה 1. הוא ישלח למחשב שעליו רצה המערכת הודעת שגיאה עם כתובות ה-IP שלו. ניתן להניח בביטחון ששני מחשבים ששלחו הודעות שגיאה עבור בקשות echo עם TTL בהפרש של 1 מקושרים זה לזה.

### 3.3 אילוצים ודרישות

#### אילוצים

- המערכת פועלת על בסיס פלטפורמת Net 5.0, וזקוקה לה כדי לעבוד.

- מאחר והמערכת תכתב באמצעות WinForms בפלטפורמת .Net, היא תוכל להיות מופעלת על מחשבי Windows 10 בלבד.
- המערכת דורשת את הספריות PcapDotNet ו-Npcap על מנת לעבוד. ספריית Npcap חיונית לעבודת המערכת מפני שמערכת ההפעלה Windows לא מממשת את pcap API בעצמה (בניגוד ל-Linux ול-macOS).
- המערכת תוכל לעבוד על ממשקי רשת מסוג Ethernet בלבד.
- המערכת תעבוד ב-Promiscuous Mode, כפי ש-Npcap ו-PcapDotNet מאפשרים. מסיבה זו, היא תקלוט את כל התעבורה שמגיעה לממשק הרשת שלו היא מאזינה במחשב עליו היא מופעלת. היקף תעבורה זאת תלוי בסוג מכשירי החומרה ב-LAN – רכזת (Hub) או מתג (Switch). רכזת מעבירה את כל התעבורה שהיא מקבלת לפורט מסוים לכל שאר הפורטים – broadcast, לעומת זאת, מתג מעביר אותה רק לפורט שמוביל למחשב לו היא מיועדת (על פי כתובת MAC). שימוש במתג לעומת רכזת מקטין משמעותית את כמות התעבורה שמחשב מקבל בלי שתהיה מיועדת אליו, ויגביל משמעותית את פעולת המערכת.

### דרישות

- הדרישה העיקרית של המערכת היא ללכוד את כל ההודעות בממשק הרשת אליו היא מאזינה. בנוסף, עליה לדעת להציג נכון את ההודעות בפרוטוקולים שהיא יודעת לנתח.
- על המערכת לתמוך בשמירת ההקלטה בפורמט pcap ולפתוח קבצים בפורמט זה.
- על המערכת לדעת להתריע על המתקפות שהיא מתוכננת לזהות (true positive) בדיוק סביר, ולא לזהות הודעות שאינן מתקפה בתור כאלה (false positive) בדיוק סביר.
- בנוסף, על המערכת אמורה לגלות לפחות את כתובות ה-MAC של המחשבים ב-LAN של המחשב עליו היא מופעלת.

### 3.4 ממשקים למערכות חיצוניות

המערכת משתמשת ב-API של ipdata <https://ipdata.co> כדי למצוא את המדינה בה רשומה כתובת IP מסוימת. לאחר ששרת ה-HTTP של ipdata מקבל HTTP Request עם כתובת IP, הוא מחזיר HTTP Response שמכיל טקסט בפורמט JSON עם נתונים גאוגרפיים לגבי כתובת ה-IP.

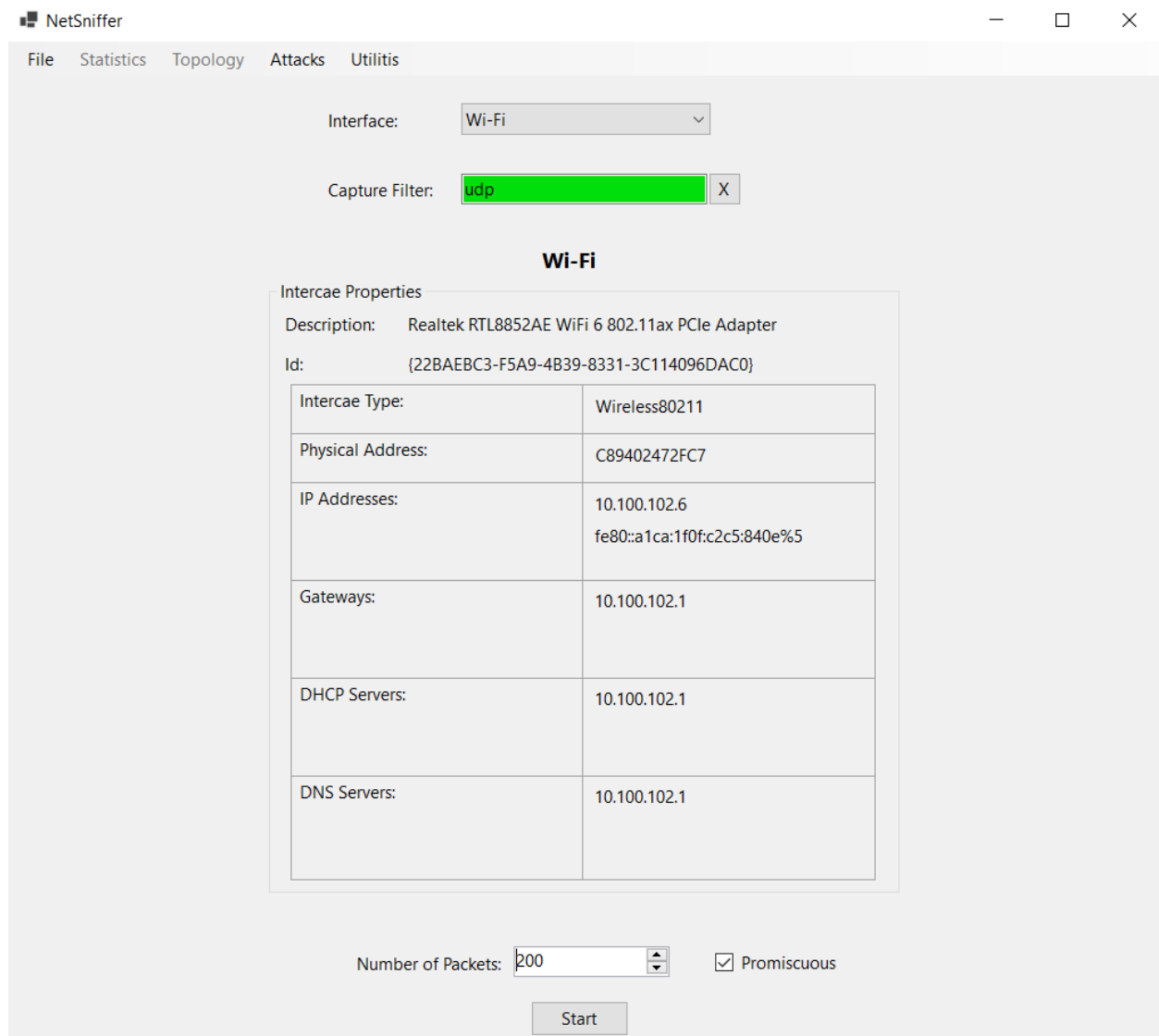
למרות שניתן לערוך את תקשורת ה-HTTP באופן עצמאי, בחרתי להשתמש ב-API של ipdata באמצעות החבילה של ipdata עבור .Net, בגלל הקושי הטמון בפענוח פורמט ה-JSON של ipdata.

### 3.5 התייחסות לנושא אבטחה

המערכת יכולה לנתח רק את המידע שעובר דרכה בצורה לא מוצפנת – היא לא מיועדת לפענח הצפנות של SSL/TLS או של IPsec. המערכת שולחת באופן עצמאי רק הודעות ICMP echo שנשלחות ללא כל הצפנה.

המערכת מאפשרת לשמור את ההקלטות בשתי דרכים, בשתיהן בפורמט pcap. בדרך הראשונה, המערכת שומרת את הקבצים ללא הצפנה. שמירה זו יושמה כדי שתוכנות אחרות, דוגמת Wireshark יוכלו לפתוח את הקבצים ששמרה המערכת. בדרך השנייה, המערכת שומרת את הקבצים כך שרק המשתמש ששמר אותם יוכל לפתוח אותם בהמשך. כדי לפתוח קובץ כזה, המשתמש צריך לספק שם משתמש וסיסמה. שם המשתמש וה-hash של הסיסמה נשמרים במאגר מידע (Database) ייעודי. פונקציית ה-hash שנבחרה היא פונקציית hash קריפטוגרפית, כלומר פונקציה שלא מאפשרת שיחזור את הסיסמה מתוך ה-hash. צורת שמירה זו מבטיחה את הישארות הסיסמה בידי המשתמש בלבד, גם עבור מאגר מידע לא מוצפן.

### 3.6. ממשק משתמש



#### חלון 1: חלון הפתיחה.

חלון הפתיחה מאפשר למשתמש לבחור ממשק רשת, capture filter ומספר הודעות מקסימלי לקליטה. בנוסף, המשתמש יכול לבחור אם המערכת תקלוט רק את ההודעות שהמחשב שלו שלח או את כל ההודעות ברשת (Promiscuous Mode). עבור ממשק הרשת הנבחר, המערכת מציגה את



שמו, הכתובת הפיזית וכתובת ה-IP שלו, את ה-Gateways שלו, ואת כתובות שרתי ה-DHCP ו-DNS שלו.

NetSniffer

File Statistics Topology Attacks Utilitis

Selected Interface: Wi-Fi

Selected Capture Filter: udp

Display Filter: ip.src == 10.100.102.006

Index	Time	Protocol	Source	Destination	Length	Info
5	5/21/2022 11:03:47 PM	UDP	10.100.102.6	216.58.198.74	75	60753 → 443 Len=33
9	5/21/2022 11:03:53 PM	UDP	10.100.102.6	216.58.198.74	75	60753 → 443 Len=33
10	5/21/2022 11:03:53 PM	UDP	10.100.102.6	216.58.198.74	75	60753 → 443 Len=33
12	5/21/2022 11:03:53 PM	UDP	10.100.102.6	216.58.198.74	75	60753 → 443 Len=33
14	5/21/2022 11:03:53 PM	UDP	10.100.102.6	216.58.198.74	75	60753 → 443 Len=33
16	5/21/2022 11:03:54 PM	UDP	10.100.102.6	216.58.198.74	75	60753 → 443 Len=33
18	5/21/2022 11:03:54 PM	UDP	10.100.102.6	216.58.198.74	75	60753 → 443 Len=33
21	5/21/2022 11:03:54 PM	UDP	10.100.102.6	216.58.198.74	75	60753 → 443 Len=33
25	5/21/2022 11:03:55 PM	UDP	10.100.102.6	216.58.198.74	75	60753 → 443 Len=33
27	5/21/2022 11:03:56 PM	UDP	10.100.102.6	216.58.198.74	75	60753 → 443 Len=33
30	5/21/2022 11:03:57 PM	UDP	10.100.102.6	216.58.198.74	75	60753 → 443 Len=33
32	5/21/2022 11:04:01 PM	UDP	10.100.102.6	216.58.198.74	75	60753 → 443 Len=33
33	5/21/2022 11:04:01 PM	UDP	10.100.102.6	216.58.198.74	75	60753 → 443 Len=33
39	5/21/2022 11:04:08 PM	UDP	10.100.102.6	216.58.198.74	75	60753 → 443 Len=33
57	5/21/2022 11:04:13 PM	UDP	10.100.102.6	216.58.198.74	244	60753 → 443 Len=202
56	5/21/2022 11:04:13 PM	UDP	10.100.102.6	216.58.198.74	75	60753 → 443 Len=33
60	5/21/2022 11:04:13 PM	UDP	10.100.102.6	216.58.198.74	75	60753 → 443 Len=33

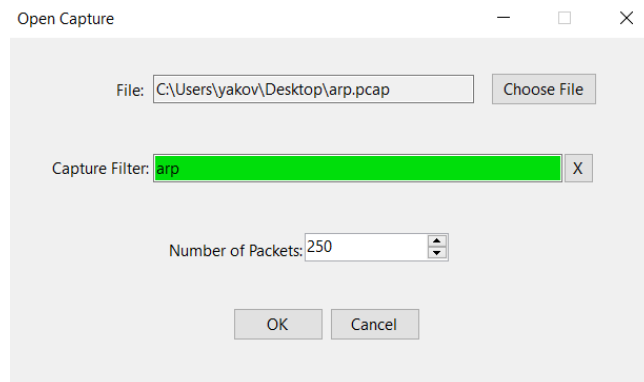
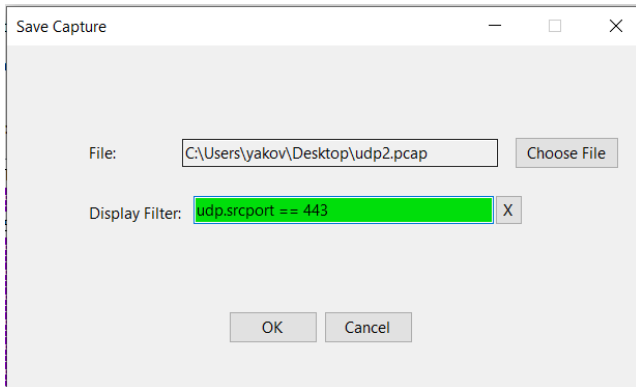
Packet Attacks


Binary Data:

```
8C59 C3E9 8249 C894 0247 2FC7 0800 4500 003D 02E5 4000 4011 28DC 0A64 6606 D83A C64A ED51 01BB
0029 E08C 4AEF BB58 2EAF 31B3 14D4 9176 2F6D C3D8 BFCA 592B EE85 649E 0C21 475F A9BF 0E7D
```

## חלון 2: חלון ההקלטה.

חלון ההקלטה מציג את ההודעות שהתקבלו. ההודעות מוצגות בצורה טבלאית, כשכל שורה מורכבת מהזמן שבו התקבלה ההודעה, הפרוטוקול הגבוהה ביותר של ההודעה, כתובות היעד והמקור, אורך ההודעה, ותיאור מילולי קצר של התוכן שלה. המשתמש יכול לראות את התוכן של הבינארי של ההודעה באזור התחתון.




ArpTableSimple

Add

Delete

Interface: 10.100.102.6 --- 0x5

Internet Address	Physical Address	Type
10.100.102.1	8c-59-c3-e9-82-49	dynamic
10.100.102.111	00-0d-de-ea-ad-db	static
10.100.102.255	ff-ff-ff-ff-ff-ff	static
10.102.202.111	00-0d-de-ea-ad-db	static
224.0.0.22	01-00-5e-00-00-16	static
224.0.0.251	01-00-5e-00-00-fb	static
224.0.0.252	01-00-5e-00-00-fc	static
239.255.255.250	01-00-5e-7f-ff-fa	static
239.255.255.251	01-00-5e-7f-ff-fb	static
255.255.255.255	ff-ff-ff-ff-ff-ff	static

Interface: 192.168.201.1 --- 0x16

Internet Address	Physical Address	Type
192.168.201.255	ff-ff-ff-ff-ff-ff	static
224.0.0.22	01-00-5e-00-00-16	static
224.0.0.251	01-00-5e-00-00-fb	static
224.0.0.252	01-00-5e-00-00-fc	static
239.255.255.250	01-00-5e-7f-ff-fa	static
239.255.255.251	01-00-5e-7f-ff-fb	static
255.255.255.255	ff-ff-ff-ff-ff-ff	static

Interface: 192.168.80.1 --- 0x17

Internet Address	Physical Address	Type
192.168.80.255	ff-ff-ff-ff-ff-ff	static
224.0.0.22	01-00-5e-00-00-16	static
224.0.0.251	01-00-5e-00-00-fb	static
224.0.0.252	01-00-5e-00-00-fc	static
239.255.255.250	01-00-5e-7f-ff-fa	static
239.255.255.251	01-00-5e-7f-ff-fb	static
255.255.255.255	ff-ff-ff-ff-ff-ff	static

### חלון 3: חלון שמירת ההקלטה באופן ציבורי.

המשתמש בוחר את הקובץ בו ההקלטה תישמר, ובוחר `display filter`, כך שרק הודעות שעונות לתנאים בו ישמרו. בדוגמה המוצגת כאן, רק הודעות UDP שנשלחו מ-port 443.

### חלון 4: חלון פתיחת ההקלטה ציבורית.

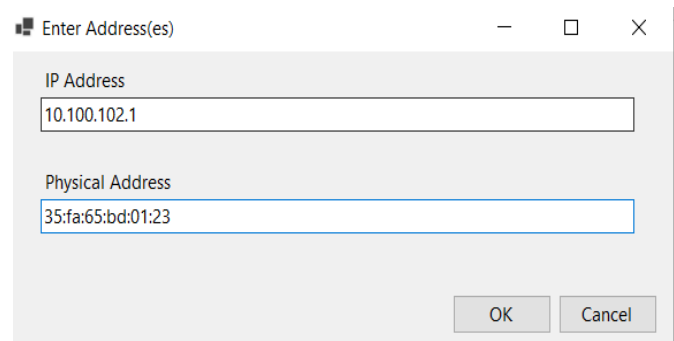
המשתמש בוחר קובץ הקלטה בפורט `pcap`, ויכול לבחור גם `filter capture` ומספר מקסימלי של הודעות. המערכת תקרא לכל היותר את המספר הזה של הודעות שעונות ל-`capture filter` מתוך הקובץ. בדוגמה, המערכת תקרא עד 250 הודעות בפרוטוקול ARP.

### חלון 5: טבלת ה-CAM.

הטבלה הושגה על ידי שימוש בפקודת `arp -a`, והיא מציגה את המיפוי בין כתובות לוגיות ופיזיות עבור כרטיסי הרשת השונים. בעזרת הכפתורים Add ו-Delete המשתמש יכול להוסיף ולהוריד רשומות מהטבלה.

### חלון 6: חלון הוספת רשומה לטבלת ה-CAM.

לחיצה על OK תוסיף לטבלת ה-CAM של המחשב רשומה שבה מופיעה כתובת ה-IP וכתובת ה-MAC שהמשתמש הזין.



Enter Address(es)

IP Address

10.100.102.1

OK Cancel

[חלון 7](#): חלון הסרת רשומה מטבלת ה-CAM. הרשומה של הכתובת ה-IP שהמשתמש הזין תוסר מהטבלה.

[חלון 8](#): טבלת הניתוב.

הטבלה הושגה על ידי הפקודה router PRINT. החלון מציג את ממשקי הרשת השונים, ולאחר מכן את טבלת הניתוב. עבור כל כתובת ידע, הטבלה מכילה netmask, gateway (הכתובת אליה צריך להעביר את ההודעה), ממשק רשת, ומטריקה.

RouteTableSimple

---

Interface List

```

15...50 81 40 93 08 2c .....Realtek Gaming GbE Family Controller
8...00 ff 47 70 2b 86 .....ExpressVPN TAP Adapter
9.....ExpressVPN Wintun Driver
3...00 ff 0b 0a d7 17 .....TAP-Windows Adapter V9
42...00 15 5d 34 f5 b6 .....Hyper-V Virtual Ethernet Adapter
18...ca 94 02 47 2f c7 .....Microsoft Wi-Fi Direct Virtual Adapter
7...ea 94 02 47 2f c7 .....Microsoft Wi-Fi Direct Virtual Adapter #2
23...00 50 56 c0 00 01 .....VMware Virtual Ethernet Adapter for VMnet1
22...00 50 56 c0 00 08 .....VMware Virtual Ethernet Adapter for VMnet8
5...c8 94 02 47 2f c7 .....Realtek RTL8852AE WiFi 6 802.11ax PCIe Adapter
1.....Software Loopback Interface 1

```

---

IPv4 Route Table

---

Active Routes:

Network Destination	Netmask	Gateway	Interface	Metric
0.0.0.0	0.0.0.0	10.100.102.1	10.100.102.6	35
10.100.102.0	255.255.255.0	On-link	10.100.102.6	291
10.100.102.6	255.255.255.255	On-link	10.100.102.6	291
10.100.102.255	255.255.255.255	On-link	10.100.102.6	291
127.0.0.0	255.0.0.0	On-link	127.0.0.1	331
127.0.0.1	255.255.255.255	On-link	127.0.0.1	331
127.255.255.255	255.255.255.255	On-link	127.0.0.1	331
172.22.0.0	255.255.240.0	On-link	172.22.0.1	271
172.22.0.1	255.255.255.255	On-link	172.22.0.1	271
172.22.15.255	255.255.255.255	On-link	172.22.0.1	271
192.168.80.0	255.255.255.0	On-link	192.168.80.1	291
192.168.80.1	255.255.255.255	On-link	192.168.80.1	291
192.168.80.255	255.255.255.255	On-link	192.168.80.1	291
192.168.201.0	255.255.255.0	On-link	192.168.201.1	291

[חלון 9](#): מידע על ה-interfaces השונים.

המידע הושג על ידי הפקודה ipconfig. עבור כל ממשק, מוצגות כתובת ה-IPv4 וה-IPv6 של המחשב, ה-subnet mask של הכתובת, וה-default gateway של הממשק.

```
IpConfigSimple
Windows IP Configuration

Ethernet adapter ??Ethernet:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Ethernet adapter ??Ethernet 2:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Unknown adapter ??%~ --%:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Unknown adapter ??%~ --% 2:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Ethernet adapter vEthernet (Default Switch):

    Connection-specific DNS Suffix  . :
    Link-local IPv6 Address . . . . . : fe80::9088:2869:2542:2812%42
    IPv4 Address. . . . . : 172.22.0.1
    Subnet Mask . . . . . : 255.255.240.0
    Default Gateway . . . . . :

Wireless LAN adapter ??%~ --%* 1:
```

[חלון 10](#): חלון תחקור המתקפות.

החלון מציג את מספר המתקפות, ותיאור קצר של כל אחת מהן. התיאור מכיל את שם המתקפה, קישור לאתר המסביר עליה, מספרי ההודעות שיצרו את המתקפה ואת הכתובות של התוקף או התוקפים והקורבן או קורבנות. המשתמש יכול לשמור את ה-log באמצעות הכפתור Save Log.

#### Attack Log

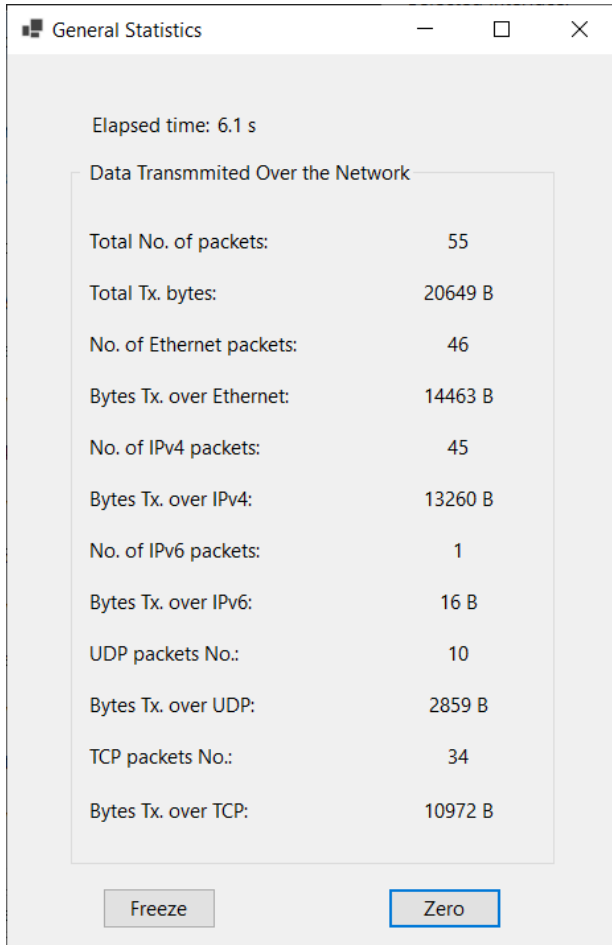
Attacks No.: 1

Attack Name: [Man-in-the-Middle \(ARP\)](#)  
Packets: 5890, 5891  
Attackers: 00DEADBFAF00  
Targets: 10.100.102.102, 10.100.102.101

Save Log

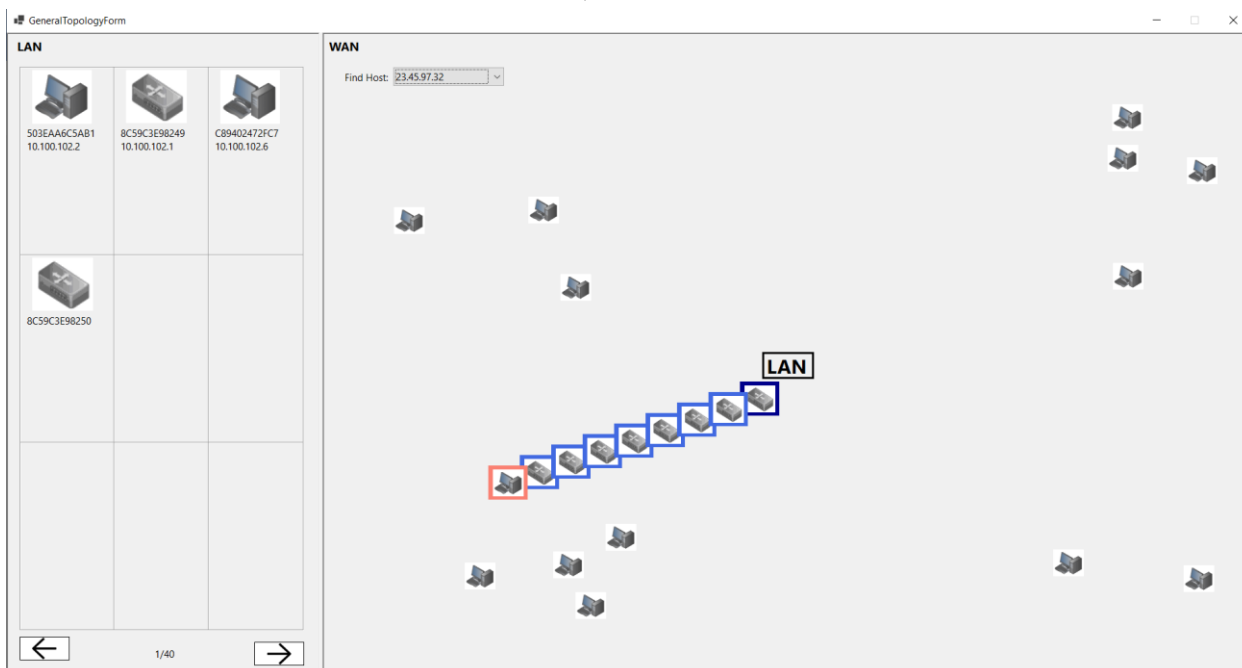
### חלון 11: חלון הסטטיסטיקה.

החלון מציג את הזמן שעבר מתחילת ההקלטה ונתונים מספריים על כמות ההודעות והמידע שהועברו בפרוטוקולים השונים. המשתמש יכול לעצור את עדכון הסטטיסטיקה באמצעות כפתור ה-Zero ולאפס את הנתונים באמצעות כפתור ה-Freeze.



### חלון 12: חלון מבנה הרשת.

החלון מציג את המחשבים ב-LAN, ואת המחשבים השונים ב-WAN. המערכת מסמנת באופן מיוחד נתבים ושרתים. המשתמש יכול לבצע `tracert` כדי לזהות את הנתבים למחשב מסוים. כפי שמוצג, במקרה הזה המערכת תסדר את הנתבים הללו על פי הסדר, ותסמן אותם בצורה מיוחדת.



### 3.7. תרחיש עיקריים

המערכת מאפשרת לנתח תעבורה בשני אופנים – בזמן אמת, ומתוך הקלטה.

#### 1. המשתמש פתח את המערכת

- המשתמש בחר להתחיל הקלטה חדשה:

תוצאה	תגובת המערכת	פעולת המשתמש
		המשתמש בחר ממשק רשת
	<u>ממשק פעיל:</u> <ul style="list-style-type: none"> <li>• <u>פילטר תפיסה תקין:</u> כפתור "התחל" מופעל. <u>ממשק לא פעיל:</u> כפתור "התחל" מבוטל. מופיע כיתוב רלוונטי.</li> </ul>	
		המשתמש הקליד capture filter
	<u>פילטר תקין:</u> נצבע בירוק. כפתור. <ul style="list-style-type: none"> <li>• <u>ממשק פעיל:</u> כפתור "התחל" מופעל. <u>פילטר לא תקין:</u> נבצע באדום.</li> </ul>	
		המשתמש בוחר מספר הודעות
		המשתמש בוחר האם לרוץ במצב Promiscuous או לא
		המשתמש לחץ על כפתור "התחל"
	המערכת עברה לפנל ההקלטה	
הקלטה החלה		
		השתמש הקליד פילטר תצוגה
	<u>פילטר תקין:</u> נצבע בירוק. רק הודעות שעונות לקריטריונים של הפילטר יוצגו מאתה והלאה. פילטר לא תקין: נצבע באדום.	
		המשתמש לחץ על כפתור עצור
	מופיעה כיתוב בראש החלון לפי ההקלטה נעצרה.	

	<u>חלון הסטטיסטיקה פתוח: עדכון הסטטיסטיקה נעצר.</u>	
ההקלטה נעצרת		
		המשתמש לחץ על כפתור היציאה
	<u>ההקלטה רצה: ההקלטה נעצרת.</u>	
	<u>ההקלטה לא נשמרה עדיין: המערכת שואלת האם המשתמש רוצה לשמור את ההקלטה.</u>	
		המשתמש בחר שהוא רוצה לשמור את ההקלטה
	חלון שמירת ההקלטה נפתח	
		המשתמש בחר שאינו רוצה לשמור את ההקלטה
התוכנה נסגרת		
		המשתמש רוצה לשמור את ההקלטה
	<u>ההקלטה רצה: הודעת שגיאה</u> <u>ההקלטה נעצרה: נפתח חלון שמירת ההקלטה</u>	

• המשתמש בחר לפתוח קובץ הקלטה פומבי:

תוצאה	תגובת המערכת	פעולת המשתמש
		המשתמש בחר קובץ הקלטה.
		המשתמש הקליד capture filter
	<u>פילטר תקין: נצבע בירוק.</u> • <u>ממשק פעיל:</u> כפתור "פתח" מופעל. <u>פילטר לא תקין: נבצע באדום.</u>	
		המשתמש לחץ על כפתור "פתח"
	המערכת עברה לפנל הקלטה	
קריאת הקובץ החלה.		

• **המשתמש בחר לפתוח קובץ הקלטה פרטי:**

תוצאה	תגובת המערכת	פעולת המשתמש
		המשתמש בחר קובץ הקלטה.
		המשתמש הקליד capture filter
	פילטר תקין: נצבע בירוק. • ממשק פעיל: כפתור "פתח" מופעל. פילטר לא תקין: נבצע באדום.	
		המשתמש לחץ על כפתור "פתח"
	המערכת מבקשת שם משתמש וסיסמה	
		המשתמש מקביל שם משתמש וסיסמה
	הפרטים תקינים: המערכת עברה לפנל הקלטה הפרטים לא תקינים: המערכת מבקשת מהמשתמש להכניס פרטים חדשים	
הפרטים תקינים: קריאת הקובץ החלה		

• **המשתמש פתח את חלון הסטטיסטיקה:**

תוצאה	תגובת המערכת	פעולת המשתמש
	<u>ריצה בזמן אמת</u> : המערכת מציגה את הזמן שעבר מתחילת ההקלטה ואת ומעדכנת אותם בזמן אמת.  <u>קריאה מתוך קובץ</u> : המערכת מציגה את פרק הזמן בין ההודעה הראשונה לאחרונה, ואת סך ההודעות שהתקבלו. כפתורי "הקפא" ו-"אפס" מבוטלים.	
		<u>בריצה בזמן אמת</u> : המשתמש לחץ על כפתור "הקפא".
	הסטטיסטיקה הפסיקה להתעדכן.	
		<u>בריצה בזמן אמת</u> : המשתמש לחץ על כפתור "אפס".



	ערכי הסטטיסטיקה שהתוכנה תציג יהיו ביחס לרגע ולכמות ההודעות בזמן שבו לחץ המשתמש על כפתור "אפס".	
		<u>בריצה בזמן אמת: המשתמש לחץ על כפתור "ביטול הקפאה".</u>
	הסטטיסטיקה התחילה שוב להתעדכן.	

• המשתמש פתח את חלון הטופולוגיה:

תוצאה	תגובת המערכת	פעולת המשתמש
	<u>ריצה בזמן אמת: המערכת מציגה את הטופולוגיה, מתעדכנת בזמן אמת.</u>  <u>קריאה מתוך קובץ: המערכת מציגה את סך הטופולוגיה מתוך הקובץ.</u>	
		<u>בריצה בזמן אמת בלבד:</u> המשתמש בחר לבצע tracert למחשב.
מתבצע tracert		
	המערכת מציגה את שרשרת המחשבים עד למחשב שאליו התבצע ה-tracert.	

• המשתמש פתח את חלון חקירת המתקפות:

תוצאה	תגובת המערכת	פעולת המשתמש
		המשתמש בחר לשמור את ה-Log
	המערכת ניגשת למשתנה סטטי (רשימה במחלקה PacketData) שמכיל את כלל המתקפות שזוהו	
ה-Log נשמר		

• המשתמש רוצה לשמור את ההקלטה:

תוצאה	תגובת המערכת	פעולת המשתמש
		המשתמש בחר את מיקום השמירה של הקובץ.
		המשתמש הקליד פילטר תצוגה

	<p><u>פילטר תקין</u>: נצבע בירוק. כפתור "שמירה" מופעל.</p> <p><u>פילטר לא תקין</u>: נבצע באדום. כפתור "שמירה" מבוטל.</p>	
		המשתמש בחר אם לשמור את הקובץ באופן פרטי
	<p><u>אופן פרטי</u>: המערכת מבקשת שם משתמש וסיסמה, או מציאה ליצור משתמש חדש</p>	
		<p><u>יש משתמש</u>: המשתמש מקליד את פרטיו</p> <p><u>אין משתמש</u>: המשתמש יוצר משתמש חדש</p>
	<p><u>יש משתמש</u>: המערכת בודקת את הפרטים. אם שגויים המערכת מתריעה על כך. הכפתור "שמירה" מבוטל עד להכנסת פרטים תקינים את ביטול השמירה הפרטית.</p> <p><u>אין משתמש</u>: המערכת בודקת אם שם המשתמש תפוס. אם הוא תפוס היא מתריעה על כך, ומבטלת את הכפתור "שמירה" עד להכנסת שם משתמש לא תפוס.</p>	
		המשתמש בלחץ על כפתור "שמירה"
	רק הרשומות המוצגות נשמרות	
הקובץ נשמר רק עם הרשומות שעונות על תנאי הפילטר.		

## 4. תהליך כתיבת הפרויקט

### 4.1. תהליך הפרויקט

כתיבת הפרויקט החלה במהלך אוקטובר, אחרי שלא הצלחתי לסיים את ה-POC של רעיון אחר לפרויקט, שמטרתו הייתה כתיבת Driver המאפשר תקשורת בין מחשבים שונים, שלא ניתן יהיה לעקוב אחריה. לאחר תקופת זמן בחוסר וודאות, החלטתי להכין sniffer לפרויקט שלי, ואת הגרסה

הראשונית של המערכת סיימתי בחודש נובמבר. מאחר והמערכת ביצעה את עיבוד התקשורת בצורה אסינכרונית, היא הצליחה לא לאבד הודעות.

בשלב הראשוני, המערכת תמכה בכמות מאוד מוגבלת של פרוטוקולים – IP, UDP, TCP, Ethernet ו-ARP. רציתי להרחיב את הפרוטוקולים שמערכת יודעת לעבוד איתם, ולקראת סוף חודש דצמבר, הוספתי תמיכה בפרוטוקולים נוספים – ICMP, DNS, DHCP ו-PcapDotNet. לא תומך בפרוטוקול DHCP, כך שנאלצתי למצוא דרך לפענח הודעות בפורמט זה בעצמי. ל-DHCP יש שדה מיוחד בעל ערך קבוע (Magic Cookie), שהקל מאוד על התהליך.

לאחר מכן, התחלתי לעבוד על הצד הטופולוגי של המערכת – בין חודשים ינואר ופברואר התחלתי לעבוד על זיהוי המחשבים ברשת, על פי כתובות MAC וכתובות IP. זיהוי המחשבים דרש הוספת מבנים רבים לקוד, והרחבה משמעותית את הפונקציונאליות שלו. בשלב זה הצגת מבנה הרשת הייתה ללא UI.

בפברואר, הוספתי למערכת את היכולת לבצע tracer, ולזהות מחשבים מקושרים. הדבר אומנם הוסיף למערכת צד אקטיבי, אך עדיין הותיר לה את האפשרות לעבוד באופן פסיבי במידת הצורך. בשלב זה, התחלתי לעבוד על זיהוי המתקפות של המערכת. הוספתי למערכת את האפשרות לנתר את טבלאות ה-ARP של המחשבים השונים, ולזהות בהן Man-In-the-Middle, כולל את ההודעות שגרמו לו.

בתחילת חודש אפריל התחלתי לעבוד על שיפור ממשק המשתמש של המערכת. בסוף חודש אפריל סיימתי את הוספת הרכיבים העיקריים למערכת, עם זיהוי מתקפות Syn Flood בפרוטוקול TCP. לאחר השלב הזה התחלתי לתקן את התקלות במערכת. במערכת התעוררו בעיות רבות בעקבות כמות המידע הגדולה שיש להשקיע (ראה חלק הבא), ונאלצתי להשקיע זמן רב בתיקון שלהן.

## 4.2. אתגרים ואופציות שונות למימוש

- פענוח התקשורת – למרות ש-PcapDotNet מאפשר גישה לשדות השונים של הפרוטוקולים הבסיסיים, השימוש הנכון במידע הזה היה לא קל. לדוגמה, הייתי צריך לשמור כתובות IP ו-MAC מתוך שכבות 2 ו-3 כדי לזהות נתבים ומחשבים שונים, לעקוב אחר הודעות ARP כדי לזהות מתקפות Man-In-the-Middle, ולעדכן את מצב ה-TCP Streams על פי הבית הדגלים שלהם. בנוסף, מימוש עיבוד ההודעות על מודל ה-OSI לא היה פשוט. בגלל ש-PcapDotNet לא תומך ב-DHCP, נאלצתי לבצע עיבוד של הודעות בפרוטוקול זה בעצמי, שהיווה משימה לא קלה בכלל.

- שימוש ב-WinForms: בתחילת הפרויקט ההיכרות שלי עם WinForms הייתה בסיסית למדי. ידעתי כיצד להשתמש ב-Controls הבסיסיים, כגון Buttons ו-Labels אך לא מעבר לכך. בהתחלה, השימוש ב-WinForms הווה אתגר, אך במהלך הפרויקט למדתי להשתמש ב-Controls מורכבים יותר, כגון ListView, וב-Containers כמו SplitPanel,

TableLayoutPanel ועוד. בנוסף, יצרתי אפילו Controls משלי באמצעות ירושה מהמחלקה UserControl. השימוש בהם הקל על הוספת פונקציונאליות חדשה לפרויקט ועל יצירת GUI אסתטי יותר ונוח יותר לשימוש.

#### • עבודה עם APIs של .Net.

מלבד הקושי בשימוש ב-WinForms, היה לי קשה ללמוד להשתמש ב-Tasks וב-API הקשור אליהם. ה-API מכיל פונקציונאליות רחבה, ולעיתים השימוש בה אינו אינטואיטיבי כלל וכלל. בחלק מהשלב, המערכת שלי התמודדה עם איטיות וחוסר יעילות, שהובילו לאובדן הודעות ולחווית משתמש ירודה. מקור הבעיה במקרים רבים היה חוסר שימוש או שימוש לא נכון ב-Tasks ובכלים אחרים להשגת מקביליות. אחרי תיקון הקוד רבות מהבעיות הללו נעלמו. שימוש נכון בתכנות מקבילי גם פתר את רוב בעיות הביצוע והקיפאון של התצוגה. בנוסף, השימוש ב-System.IO.IsolatedStorage לא היה קל, מפני שלא הכרתי אותו לפני כן. בסוף הצלחתי לשמור קבצים בצורה נפרדת עם הפונקציות המתאימות מתוך System.IO.IsolatedStorage ולפתוח אותם ממנו. במהלך הפרויקט למדתי להשתמש באוספים של שפת C#, וב-LINQ. תחילת השימוש בהם היה מאתגר, אבל לקראת סוף הפרויקט שלטתי בהם יחסית טוב.

## 5. מרכיבי פתרון

### 5.1. תיחום הפרויקט

- תקשורת – המערכת תאזין לתעבורה על המחשב בו היא מופעלת, ותנתח אותה
- תצוגה – המערכת תהיה בנויה בתצוגת WinForms
- מבנה נתונים – המערכת משתמשת רבות במרשימות, במילונים ובמערכים על מנת לשמור את ההודעות שהתקבלו, ואת המידע הקשור אליהן. המערכת שומרת את כל ההודעות שהתקבלו במילון, ושומרת גם את טבלת ה-ARP וחיבורי ה-TCP של כל מחשב במילונים. המערכת שומרת את המחשבים שזיהתה במספר רשימות.
- מערכות הפעלה – המערכת רצה על Windows 10, ותהליך הפיתוח והבדיקה יעשה עליה בלבד. נעשה שימוש מאסיבי ב-UI thread, וב threads מתוך ה-thread-pool נעשה שימוש בסנכרון של ה-threads עם שפת c#. ישנו שימוש עקיף רב ב-API של מערכת, Windows, ש WinForms מתבסס עליו.
- אבטחת נתונים – המערכת תעבוד במצב Promiscuous (Promiscuous Mode), ומאזינה לכל התעבורה שמגיעה לממשק הרשת במחשב עליו היא מופעלת. המערכת תשמור את הקלטות התעבורה בצורה לא מוצפנת, כדי שיוכלו להיפתח על ידי תוכנות אחרות, דוגמת Wireshark. המערכת מאפשרת ליצור משתמשים שונים, ונותנת למשתמש את האפשרות לשמור את ההקלטה כך שרק הוא יוכל לפתוח אותה מאוחר יותר, באמצעות שם המשתמש והסיסמה שלו.
- ארכיטקטורת קוד – הקוד מתבסס בצורה על ארכיטקטורת MVC, אך עם שינויים מסוימים.

## 5.2. סביבת העבודה (טכנולוגיה)

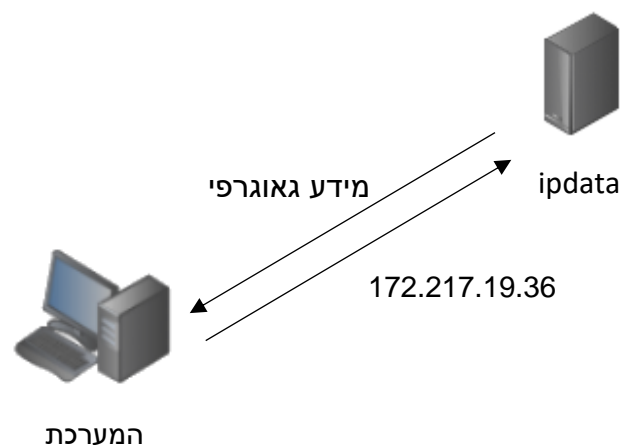
### ● שפת התכנות: C#

בחרתי לעבוד ב-C# ולא ב-Python. יתרון חד-משמעי של C# על פני Python הוא היכולת המובנת שלה ליצור ממשקים גרפיים מפותחים מאוד ולעצב אותם בצורה נוחה באמצעות Visual Studio. ספריית הקוד הפתוח scapy ב-Python מעניקה גם היא יכולות של האזנה וניתוח תעבורה, כך שספריית PcapDotNet, המתפקדת כמעטפת ל-pcap API לא היוותה יתרון לטובת C#. הסיבה העיקרית שלי לבחירת C# לכתיבת הפרויקט היא שאני מוצא אותה נוחה יותר לכתיבת פרויקט גדול. ראשית, C# היא שפה type safe. בנוסף, C# תומכת בתכנות מונחה עצמים (Object Oriented Programming; OOP) בצורה נרחבת יותר מ-Python – היא מאפשרת להשתמש בהרשאות גישה (באמצעות access modifiers), ליצור מחלקות אבסטרקטיות ומחלקות חתומות (sealed), להגדיר פונקציות וירטואליות ולא וירטואליות ועוד. היא תומכת ב-interfaces, ב-events וב-delegates, ובעלת ספרייה סטנדרטית (Base Class Library; BCL) עשירה יותר. אם זאת, יש לזכור שככל הנראה הייתי מצליח לכתוב את הפרויקט גם ב-Python.

### ● סביבות פיתוח: Visual Studio

## 5.3. מבט טופולוגי

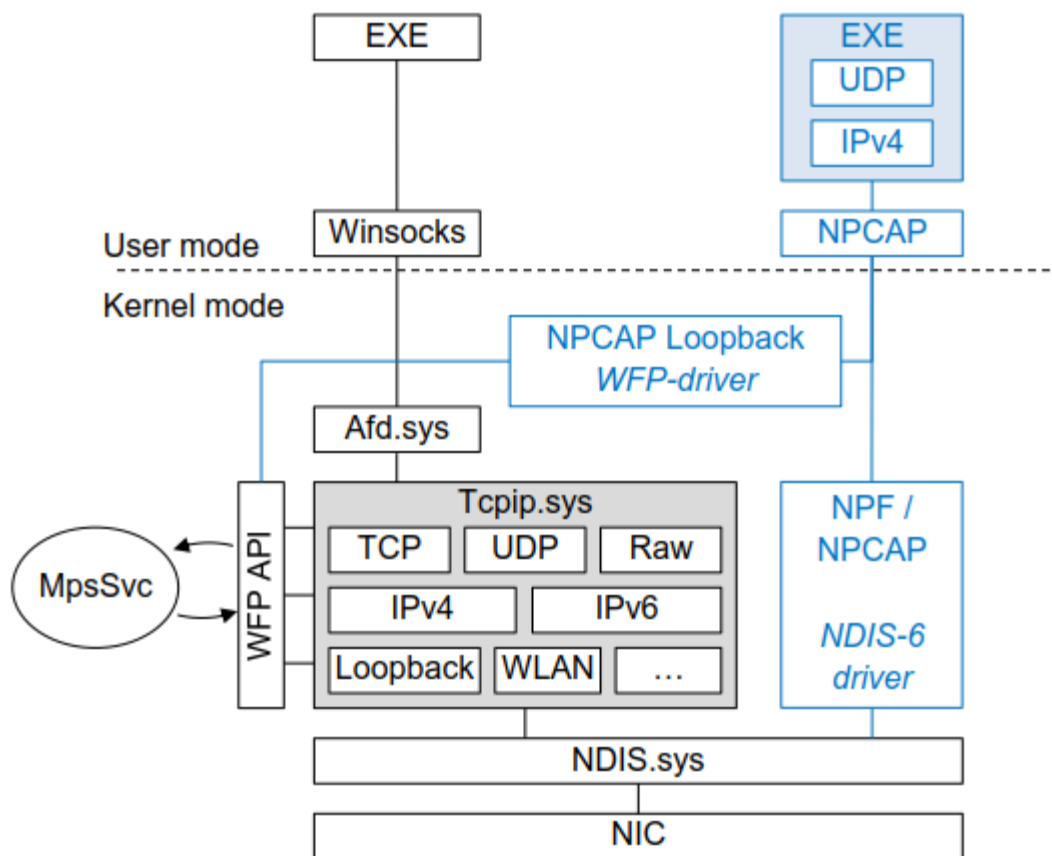
המערכת רצה על מחשב בודד ב-LAN, ומשתמשת בשירותים שמספק שרת ה-HTTP של ipdata. התקשורת עם השרת של ipdata מוצגת בתרשים 4. המערכת שולחת לשרת בקשה עם כתובת IP מסוימת, ומקבלת בתשובה מידע.



### תרשים 4: התקשורת של המערכת עם ipdata.

תרשים 5 מציג את ה-drivers והרכיבים השונים הקשורים בשליחת וקליטת הודעות על ידי תוכנות User Space ב-Windows. תקשורת רגילה מתנהלת באמצעות sockets, אותן ניתן ליצור באמצעות Winsocks API. ה-API משתמש ב-Afd.sys driver, שמשתמש בתורו ב-tcpip.sys. tcpip.sys

אחראי על ניהול התקשורת בפרוטוקולים השונים, בהם IPv4, IPv6, UDP ו-TCP. tcpip.sys פונה ישירות ל-ndis.sys, שאחראי על שליחת ההודעות באמצעות ה-NIC (Network Interface Controller). על מנת לחסום הודעות או לשנות אותן טרם קבלתן או שליחתן, יש להשתמש ב-Windows Filtering Platform API (WFP API). ה-driver של Npcap פונה ל-ndis.sys. באופן ישיר, ומשתמש WFP API רק לצורך קליטת תקשורת Loopback. זאת הסיבה מדוע Npcap מסוגל לקלוט ולשלוח הודעות באופן עצמאי, אך לא לחסום או לשנות הודעות מתקבלות.



[תרשים 5:](#) ה-drivers השונים הקשורים לשליחת לתקשורת ב-Windows

#### 5.4 מבנה נתונים

מבני נתונים חשובים בקוד כוללים:

- המילון `ConcurrentDictionary<int, Packet> packetsById` שמכיל את כל ההודעות, כאשר ה-key הוא id של ההודעה, שהוא שבעצם המספר הסידורי שלה.
- המחלקה `ArpTable` שמייצגת את טבלת ה-ARP של כל מחשב. מבנה נתונים חושב בה הוא המילון `Dictionary<IPAddress PhysicalAddress> arpTable` שמכיל את הנתונים של הטבלה עצמה.

- המחלקה LanHost שמייצגת מחשב ב-LAN. היא מכילה את PhysicalAddress, הכתובת הפיזית של המחשב ואת IPAddress כתובת ה-IP של המחשב.
- המחלקה LanMap שמאגדת את כל המחשבים ב-LAN. יש בה את הרשימות List<LanHost> Hosts, List<LanHost> Routers ו-List<LanHost> DhcpServers שמכילות את המחשבים, הנתבים ושרתי ה-DHCP ב-LAN.
- המחלקה WanHost שמייצגת מחשב מחוץ ל-LAN. מבני נתונים חשובים במחלקה הם List<WanHost> ConnectedHosts, רשימה שמכילה את המחשבים אליהם מחובר כל מחשב ו-List<TcpConnection> TcpConnections, רשימה שמכילה את חיבורי ה-TCP של כל מחשב.
- המחלקה WanMap שמאגדת את המחשבים מחוץ ל-LAN. היא מכילה את הרשימות List<WanHost> LanRouters, List<WanHost> WanRouters, List<WanHost> DnsServers שמכילות את המחשבים, הנתבים ושרתי ה-DNS מחוץ ל-LAN.

## 5.5 [מסד נתונים](#)

המערכת משתמשת במסד נתונים, שבו היא שומרת את שמות המשתמשים ואת ה-hashes של הסיסמאות שלהם. המסד מכיל טבלה אחת בשם Accounts, שמורכבת משתי עמודות, Username ו-PasswordHash. עמודת Username מכילה את שמות המשתמשים בתור text, ועמודת PasswordHash מכילה את ה-hashes של הסיסמות בתור מערכים של 32 בתים (32(binary)). המערכת משתמשת במסד כדי לדעת אילו משתמשים רשומים, והאם של המשתמש והסיסמה שהמשתמש מקליד נכונים. כאשר נוצר משתמש חדש, המערכת שומרת את פרטיו במסד הנתונים.

## 5.6 [מבט מודולרי](#)

הקוד מתבסס על ארכיטקטורת MVC,

- Controller: את תפקיד ה-Controller של הארכיטקטורה ממלאת המחלקה NetSniffer. לאחר קבלת הודעה, היא מודיעה על כך ל-View.
  - View: את תפקיד ה-View ממלאות המחלקות ImprovedMainForm, PacketViewer, LanViewer ו-WanViewer. המחלקה PacketViewer היא זאת שאומרת ל-Model לנתח את ההודעה. בניגוד לארכיטקטורת MVC רגילה, LanViewer ו-WanViewer לא מעודכנות על ידי ה-Model, אלא לוקחות ממנו את המידע בעצמן.
  - Model: את תפקיד ה-Model ממלאות המחלקות המנתחות את הנתונים לבקשת ה-View.
- להלן רשימה של המחלקות העיקריות בהן נעשה שימוש בכל אחד מהתחומים.

- ImprovedMainForm – החלון הראשי של המערכת. הוא מציג את תצוגת הפתיחה של המערכת (בחירת ממשק הרשת), ואת ההודעות שהתקבלו.
- UserControl – PacketViewer שמציג את ההודעות.
- GeneralStatisticsForm – חלון עזר שמציג את כמויות ההודעות שהתקבלו.
- GeneralTopologyForm – חלו המציג את המחשבים ב-LAN, כולל את הנתבים ושרתי ה-DHCP.
- AttackLogForm – מציג את כל המתקפות שזוהו על ידי המערכת
- TcpStreamsForm – מציג את חיבורי ה-TCP (streams) של מחשב מסוים

### הסנפת תקשורת

- NetSniffer – מחלקה אבסטרקטית שמייצגת עצמים שתומכים בפעולות של קבלת הודעה, ושמירת הודעות. לאחר שההקלטה מתחילה, המחלקה מוציאה event עבור כל הודעה שהתקבלה. ImprovedMainForm מקשיבה ל-event הזה, ואומרת למורה ל-PacketAnalyzer לנתח את התעבורה.
- LiveSniffer – מייצג NetSniffer המאזין לתעבורה בזמן אמת.
- OfflineSniffer – מייצג NetSniffer שמקור ההודעות שלו הוא קובץ.

### פענוח תקשורת

- PacketAnalyzer – מחלקה המאגדת את כל הפעולות המבוצעות על התעבורה – היא מאפשרת את ניתוח ההודעה, ודרכה ניתן להגיש את הטופולוגיה של הרשת.
- PacketData – מחלקה המייצגת את המידע המתלווה להודעה, והמתקפות שמשויכות אליה. מוציא event אחרי סיום הניתוח של ההודעה, שנקרא על ידי PacketViewer.
- BaseAnalyzer – מחלקה אבסטרקטית שמכילה פונקציונאליות של ניתוח חלק מהודעה.
- ArpTable – מכיל את המידע של טבלת ARP
- TcpConnection – מייצג את המידע השמור לגבי חיבורי ה-TCP של מחשב

### 5.7 פירוט מודלים עיקריים

Class	Function	Input\Output	Description
NetSniffer	GetPacketDevice	Input: - Output: PacketDevice	יוצרת אובייקט מסוג PacketDevice, המתאר מקור שממנו יכולות להגיע הודעות. המקור יכול להיות קובץ הקלטה, או כרטיס הרשת.
	Start	Input: - Output: -	מתחילה אתה ההקלטה. קוראת לפעולה OnPacketReceived עבור על הודעה שהתקבלה
	StartAsync	Input: -	מתחילה את ההקלטה באופן אסינכרוני



		Output: -	
	OnPacketReceived	Input: - Output: -	מכניסה את ההודעה לתור
	OnPacketReceivedCore	Input: Packet packet Output: -	מופעלת בצורה סינכרונית עבור כל הודעה בתור. יוצרת אובייקט מסוג PacketData עבור ההודעה ומתחילה את הניתוח שחה בצורה אסינכרונית.
	Stop	Input: - Output: -	מפסיקה את ההקלטה
LiveSniffer	Ping	Input: IPAddress destination, byte ttl Output: PingReply	שולחת הודעת ICMP echo לכתובת מסוים, ועם TTL מסוים
PacketData	Create	Input: Packet packet Output: PacketData	מקבל מספר סידורי עבור ההודעה ויוצרת אובייקט חדש מסוג PacketData
	Analyze	Input: - Output: -	שולחת את ההודעה לניתוח
	AnalyzeAsync	Input: - Output: -	שולחת את ההודעה לניתוח בצורה אסינכרונית
IdManager	GetNewPacketId	Input: Packet packet Output: int	נותנת להודעה מספר סידורי ושומרת אותה במילון על פי המספר הזה.
Packet Analyzer	AnalyzePacket	Input: Packet packet, int packetId Output: PacketDescription	מנתחת את ההודעה ומחזירה תיאור שלה
	GetLanMap	Input: - Output: LanMap	מחזיר מפה שמייצגת את המחשבים ב-LAN
	GetWanMap	Input: - Output: WanMap	מחזיר מפה שמייצגת את המחשבים ב-LAN ואת החיבורים שלהם
	Tracert	Input: IPAddress destination Output: -	שולחות הודעות ICMP echo על מנת למצוא את הנתבים בדרך למחשב מסוים
Topology-Builder	MakeHostRouter	Input: PhysicalAddress physicalAddress Output: -	מסמנת שמחשב מסוים הוא נתב
	AddHostInLan	Input: PhysicalAddress physicalAddress, IPAddress ipAddress Output: -	מוסיפה מחשב למפת ה-LAN, ובודקת אם הוא נתב

	AddHostInWan	<b>Input:</b> IPAddress ipAddress <b>Output:</b> -	מוסיפה מחשב למפת ה-WAN
Base Analyzer	AnalyzeDatagram	<b>Input:</b> Datagram datagram, IContext context, <b>int</b> packetId <b>Output:</b> IAnalysis	ממירה datagram לסוג מתאים עבור הניתוח, השונה בין מחלקות שונות היורשות מ-BaseAnalyzer
	Analyze DatagramCore	<b>Input:</b> TDatagram datagram, TContext context, <b>int</b> packetId <b>Output:</b> IAnalysis	מנתחת את ההודעה ומחזירה את הניתוח שלה
ArpStateful Analyzer	AnalyzeRequest	<b>Input:</b> ArpDatagram request, <b>int</b> packetId <b>Output:</b> -	מוסיפה את בקשת ה-ARP לרשימה המכילה את כל הודעות ה-ARP
	AnalyzeReply	<b>Input:</b> ArpDatagram reply, <b>int</b> packetId <b>Output:</b> -	מוסיפה את תגובת ה-ARP לתור
	Analyze ReplyCore	<b>Input:</b> ArpDatagram reply, <b>int</b> packetId <b>Output:</b> -	לוקחת תגובת ARP מתוך התור, ובודקת האם בוצעה מתקפת Man-In-the-Middle
ArpTable	UpdateEntry	<b>Input:</b> IPAddress ipAddress, PhysicalAddress physicalAddress, DateTime receivedTime, <b>int</b> packetId <b>Output:</b> -	מעדכן רשומה בטבלת ה-ARP
	AddEntry	<b>Input:</b> IPAddress ipAddress, PhysicalAddress physicalAddress,	מוסיף רשומה לטבלת ה-ARP

		DateTime receivedTime, int packetId Output: -	
TcpStateful-Analyzer	Analyze Datagram	Input: TcpDatagram datagram, NetworkContext context, int packetId Output: -	מוצאת או יוצרת את האובייקט מסוג TcpConnection שמייצג את ה-stream שבו הועברה ההודעה, ומעביר אותה לניתוח שלו
	DetectSynFlood	Input: - Output: -	מנסה לזהות מתקפות מסוג Syn Flood
Tcp Connection	Analyze ConnectorPacket	Input: TcpControlBits flags, uint rawSequenceNumber, uint rawAcknowledgement- Number, uint payloadLength Output: -	מנתח את שדות הבקרה של פקטת TCP, ומעדכנת את כמות המידע שהועברה ב-stream
	Analyze ListenerPacket	Input: TcpControlBits flags, uint rawSequenceNumber, uint rawAcknowledgement- Number, uint payloadLength Output: -	מנתח את שדות הבקרה של פקטת TCP, ומעדכנת את כמות המידע שהועברה ב-stream
Improved MainForm	Sniffer_ PacketReceived	Input: object sender, Packet e Output: -	מגיבה ל-event שמוציא NetSniffer לאחר קבלת הודעה
Packet Viewer	AddPacket	Input: Packet packet Output: -	מציגה הודעה

## 6. תסריטי בדיקה

### 6.1. דגשים בבדיקה

- תמיכה מספר רב של הודעות, ללא השמטת הודעות, ותוך תצוגתם בסדר הגעתם.
- היכולת להציג מחשבים רבים ב-LAN ומחוצה לו.
- ביצוע כל הפעולות באופן תקין, ללא קריסות.
- נכונות תוצאות מיפוי הרשת של המערכת
- זיהוי מתקפות נכון
- נכונות המעקב אחרי חיבורי TCP שמבצעת המערכת

### 6.2. תסריטי בדיקה עיקריים

- פתיחת הקלטת חדשה, בדיקה שה-capture filter, כמות ההודעות המקסימלית, ובחירה האם המערכת קולטת את כל ההודעות (Promiscuous Mode) עובדים כמו שצריך.
- פתיחת קובץ הקלטת, בדיקה שה-capture filter עובד כמו שצריך.
- עצירת הקלטת רצה, והתחלתה מחדש.
- שמירת הקלטת, בדיקה שה-display filter עובד כמו שצריך.
- סימלון של מתקפה מכל סוג, בדיקה של הזיהוי שלה.
- ביצוע tracert, בדיקת נכונות התוצאות שלו.
- בדיקת הפעולה של ה-utilities של המערכת.

## 7. רפלקציה

### 7.1. לוח זמנים מוערך לניהול הפרויקט:

חודש	מטרה
נובמבר	<b>תכנון:</b> POC - האזנה לתעבורה לא אובדן הודעות, שמירת תעבורה בפורמט pcap ופתיחה של קבצים בפורמט זה <b>ביצוע:</b> תואם לתכנון
דצמבר	<b>תכנון:</b> הוספת תמיכה בפרוטוקולים נוספים (דוגמת DHCP) <b>ביצוע:</b> הוספת תמיכה בפרוטוקולים נוספים – DHCP, DNS ו-ICMP
ינואר	<b>תכנון:</b> תצוגה ראשונית פענוח ראשוני של מבנה הרשת על פי הודעות ARP וכתובות MAC

	<b>ביצוע:</b> זיהוי של מחשבי הרשת על פי כתובות IP ו-MAC, תצוגה על בסיס רשימה של מבנה הרשת.
פברואר	<b>תכנון:</b> זיהוי מתקפות פשוטות כגון ARP man-in-the-middle <b>ביצוע:</b> tracert
מרץ	<b>תכנון:</b> עדכון תצוגה, פענוח מתקדם של מבנה הרשת <b>ביצוע:</b> זיהוי מתקפת Man-In-the-Middle
אפריל	<b>תכנון:</b> שמירת מידע על מבנה הרשת והמתקפות שזוהו, הצגת מבנה הרשת בצורה גרפית מלאה <b>ביצוע:</b> זיהוי מתקפת Syn Flood, הוספת תצוגה גרפית של מבנה הרשת.
מאי	<b>תכנון:</b> שיפור תצוגה, הוספת מתקפות מזוהות, בדיקה מקיפה <b>ביצוע:</b> שיפור התצוגה ותיקון באגים

## 7.2. אתגרים ותרומה אישית

בתחילת כתיבת הפרויקט השליטה שלי ב-C# לא הייתה מלאה, והידע התאורטי שלי בתקשורת היה חלוד יחסית. בנוסף, לפני הפרויקט כמעט ולא יצא לי לעבוד ב-WinForms. לכן, בתחילת הפרויקט היו לי לא מעט קשיים, גם בתכנון הפרויקט וגם במימוש שלו. התרגלתי לעבודה עם WinForms יחסית מהר (למרות שנשארו בעיות עד השלבים האחרונים של הפרויקט), והרחבתי את הידע שלי ב-C# וב-OOP בצורה משמעותית. למדתי הרבה דברים שקשורים לתקשורת, גם ברמה התאורטית וגם ברמה המעשית. לסיכום, הפרויקט היה מעניין מאוד.

## 7.3. תובנות

למדתי שלכתוב פרויקט בקנה מידה גדול שונה משמעותית מכתבת תוכנות קטנות או פשוטות, ושיש לתכנן היטב את הארכיטקטורה של הקוד לפני שמתחילים לכתוב אותו (משהו שלא תמיד ביצעתי). אחת התובנות העיקריות שלי היא שאין דרך טובה יותר להשתפר בתחום מסוים חוץ מלעסוק בו, ואני חושב שהשתפרתי מאוד בתכנות ובתחום של עיבוד תקשורת.

# 8. הוראות התקנה ותפעול

## 8.1. תצורה ודרישות קדם

הדרישות החיוניות ביותר לצורך הרצת הפרויקט הן Npcap וספריית הקוד הפתוח PcapDotNet. בנוסף לכך, דרושות מספר חבילות ל-Visual Studio, ומאגר נתונים מתאים.

## 8.2. התקנה

את Npcap Driver ניתן להוריד באתר <https://nmap.org/npcap>. יש להוריד את SQL Server על המחשב בו מורצת התוכנה, וליצור בוא מאגר נתונים בשם NetSnifferDatabase. את המאגר ניתן ליצור בקלות באמצעות התוכנה Microsoft SQL Server Management Studio. במאגר יש ליצור טבלה בשם Accounts, ובה עמודה בשם Username מסוג text, ועמודה בשם PasswordHash מסוג מארך באורך קבוע של 32 בתים. יש לקבל את מחרוזת החיבור (Connection String) למאגר, ולהזין אותה בשדה connectionString (field) במחלקה AccountManager.

את PcapDotNet ניתן להוריד מ-GitHub, בכתובת <https://github.com/PcapDotNet/Pcap.Net>. בפרויקטים (Projects) NetSnifferApp ו-NetSnifferLib יש להוסיף ל-Dependencies את ה-DLLs הבאים:

- PcapDotNet.Base
- PcapDotNet.Core
- PcapDotNet.Core.Extensions
- PcapDotNet.Packets

בפרויקט NetSnifferApp יש להוריד את החבילה (Package) System.Data.SqlClient, בפרויקט NetSnifferLibWebInfo יש להוריד את החבילה IpData. את הורדת החבילות ניתן לבצע באמצעות NuGet.

## 9. ביבליוגרפיה

במהלך כתיבת הפרויקט הסתמכתי על מספר מקורות מידע:

ה-GitHub repository של PcapDotNet:

- <https://github.com/PcapDotNet/Pcap.Net>

האתרים הבאים:

- <https://nmap.org/npcap>, אתר הבית של Npcap
- <https://www.snort.org>, אתר הבית של Snort
- <https://www.wireshark.org>, אתר הבית של Wireshark
- [https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.rapid7.com%2Fglobalassets%2Fexternal%2Fdocs%2Fdownload%2FMS\\_pnd\\_qsg.pdf&psig=AOvVaw2VuROJqcl83Bm4pd77goeU&ust=1638022001544000&source=images&cd=vfe&ved=0CA0Q3YkBahcKEwjI3MzPmbb0AhUAAAA](https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.rapid7.com%2Fglobalassets%2Fexternal%2Fdocs%2Fdownload%2FMS_pnd_qsg.pdf&psig=AOvVaw2VuROJqcl83Bm4pd77goeU&ust=1638022001544000&source=images&cd=vfe&ved=0CA0Q3YkBahcKEwjI3MzPmbb0AhUAAAA), מסמך הסבר על השימוש ב-Passive Network Mapper MetaModule

ועל מקורות המידע הבאים:

- <https://en.wikipedia.org/wiki/Pcap>, ערך הוויקיפדיה על pcap API
- <https://www.forcepoint.com/cyber-edu/intrusion-prevention-system-ips>,  
הסבר על IDS ו-IPS
- <https://tools.ietf.org/id/draft-gharris-opsawg-pcap-00.html> טיוטה של IETF  
על פורמט קובץ pcap
- [https://www.ietf.org/staging/draft-tuexen-opsawg-pcapng-02.html#section\\_shb](https://www.ietf.org/staging/draft-tuexen-opsawg-pcapng-02.html#section_shb) טיוטה של IETF על פורמט קובץ pcapng

## 10. נספחים

---

הצעת הפרויקט מצורפת.