

עבודת גמר 5 יח"ל

נושא העבודה: אנטי וירוס

שם התלמיד: אלעד ישעיהו

ת.ז התלמיד: 328488432

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

שם המנחה: ערן בינט

מועד הגשה: 2022/12/20

תוכן עניינים

[1. מבוא 3](#_Toc121383790)

[2. תיאוריה 4](#_Toc121383795)

[3. תוצר סופי 8](#_Toc121383798)

[4. תהליך כתיבת הפרויקט 17](#_Toc121383806)

[5. מרכיבי פתרון 18](#_Toc121383809)

[6. תסריטי בדיקה 20](#_Toc121383817)

[7. רפלקציה 20](#_Toc121383820)

[8. הוראות התקנה ותפעול 21](#_Toc121383824)

[9. ביבלוגרפיה 21](#_Toc121383827)

[10. נספחים 21](#_Toc121383828)

# 1. מבוא

## 1.1 נושא העבודה

נושא העבודה אותו בחרתי ליישום הוא Anti-Virus , אותו אני מכנה "**YeshScanner**" על שמי Elad Yesh. המערכת מאפשרת סריקה חכמה, מהירה ויעילה בסביבה בטוחה של התקנות חדשות במחשב או תיקיות וכוננים שונים הנמצאים במחשב. המערכת מנטרת ומזהה התנהגויות חשודות וקבצים אשר מנסים לגשת למקומות המוגדרים כחשודים במחשב, תוך ניטור פרמטרים שונים וסמויים בקובץ הריצה, וחקירת התנהגות מחשידה בעת ריצתו.

הפרויקט משתמש במספר שיטות עיקריות, ביניהם:

* WinApi Hooking לחקירה ההתנהגותית של הקובץ (Blackbox testing).
* PE Parsing וניתוח ה IAT על מנת לבצע בדיקה סטטית עמוקה על קובץ ההרצה (Whitebox testing)
* התממשקות עםVirusTotal כדי להיעזרבמנועי הסריקה שלו למטרת זיהוי ואבחון וירוסים ונוזקות חבויות בתוך קובץ תמים.
* Fuzzy Hashing ו Hash Comparison על מנת למצוא דמיון בין קובץ ההרצה החשוד לקובץ malware מוכר.

## 1.2 מטרות מרכזיות

המטרות המרכזיות של הפרויקט הן:

* דיווח על מספר רב של פרמטרים ופונקציות המוגדרים כחשודים, למטרת ניתוח מעמיק של קובץ ההרצה.
* ניתוח של וירוסים בשפות שונות ומשפחות שונות של נוזקות.
* ממשק נוח למשתמש הדומה לVirus Total .

המטרות האישיות המרכזיות של הפרויקט הן:

* למידה של מבנה הPE של קובץ וWinApi ברמה גבוהה, בצירוף הבנה עמוקה של DLL והסוגים השונים והמגוונים של נוזקות. מטרה זו תעמיק את החקירה והניתוח של קובץ ההרצה, ותאפשר תוצר משמעותי ואיכותי.
* התמקצעות בגרפיקה כדי ליצור עיצוב גרפי יסודי ויפה, וגיבוש שליטה בפונקציונליות של הכלי.
* הבנה עמוקה בתחום הסייבר ההתקפי המבוצע באמצעות נוזקות, והבנת ההיגיון העומד מאחוריהן לעומק.
* יישום הפרויקט ברמה גבוהה, על בסיס התכנון המשתקף במסמך זה, באופן הממצה את יכולותיי כתלמיד במגמת סייבר.

## 1.3 רציונל

באופן כללי, נוזקות תמיד עוררו אצלי עניין. מאז שהייתי צעיר ידעתי שלווירוסים יש השפעה גדולה והרסנית וביכולתם לפגוע בתפקודו של מחשב עד למצב בלתי הפיך. העניין הרב שיש לי בתחום הווירוסים, והרצון שיש לי להבין את ההיגיון העומד מאחורי Executables ואיך הם מורכבים גרמה לי להתאהב בנושא הזה. לא למדתי בשיעורי סייבר עד כה את הנושא הזה, ואני מאמין שהחקירה העצמית תעזור לי לגדול ולהתפתח בתחום הווירוסים והנוזקות.

בנוסף, נושא הפרויקט מאפשר לי הזדמנות להעמיק ולהרחיב את הידע שלי בתחום הLow level Programming, וללמוד שפות תכנות חדשות כמו Assembly, C, C++ על מנת לחקור בצורה יותר מעמיקה ולהגיע למקומות היותר חבויים הנמצאים בקובץ.

## 1.4 קישור לחומר הנלמד

העבודה מתקשרת לחומר הנלמד בשיעורים במספר תחומים שונים.

ראשית, בתחום מערכות הפעלה. הפרויקט עוסק בתחומים הרבים הקשורים למערכות הפעלה וחומר הנלמד בכיתה כגון: DLL, WinApi, Portable Executable . אני משתמש ומתבסס על נושאים אלה באופן נרחב בפרויקט שלי, וחוקר אותם לעומק על מנת להגיע לניתוח מקסימלי של נוזקה.

בנוסף, בתחום התקשורת, הפרויקט מממש את העברת הנוזקה בין המחשב למכונה הווירטואלית באמצעות תקשורת Client-Server מעל Sockets אשר ייכתבו בשפת Python כפי שנלמד בכיתה. הפרויקט שלי ממומש חלקית בשפת Python ונעזר בלא מעט קונספטים שלמדנו כמו: regular expression, packaging, inheritance, encapsulation decorations, classes ועוד. במהלך כתיבת הפרויקט השתמשתי ב-github בתור source control, כפי שנלמד.

# 2. תיאוריה

## 2.1 תיאוריה

הרעיון של המערכת הוא לחקור בצורה כמה שיותר עמוקה קובץ Executable ולהתריע על כמה שיותר דברים שנמצאו חשודים בתוכו. על מנת לעשות זאת, המערכת מבצעת שלושה בדיקות עיקריות: Dynamic Analysis, Static Analysis ו Hash Analysis . לפני שאסביר על הבדיקות האלה, מוטב לתת הקדמה על מושגים בסיסיים בעולם הנוזקות, שכן הבנתם הכרחית להבנת מרכיבי המערכת:

* VIRUS – נוזקה, או בשמה היותר מוכר, וירוס,

הפרויקט שלי מבצע סוגי בדיקות שונות על מנת להגיע לחקירה כמה שיותר עמוקה:

* בדיקת קופסה שחורה (Blackbox testing) היא בדיקה פונקציונלית המתבצעת ללא התייחסות למבנה הפנימי של הרכיב או המערכת. בבדיקה זו מתייחסים אך ורק לקלט ולפלט של המערכת; כלומר החקירה ההתנהגותית של המערכת. כמוכן, בבדיקה זו יש חשיבות אך ורק ל"מה", ולא ל"איך". חשובה התוצאה, ולא איך המערכת הגיעה אליה.
* בדיקת קופסה לבנה (Whitebox testing) היא בדיקות תוכנה המתייחסת למבנה הפנימי של הרכיב או המערכת. בניגוד לבדיקת קופסה שחורה, שמטרתה לבדוק את קיום הדרישות של הרכיב, בדיקה זו בודקות את המטרה והלוגיקה העומדות מאחורי החלקים הפנימיים של הרכיב. במסגרת זו מתבצעת בדיקה על הקוד עצמו, ולכן זוהי בדיקה סטטית.
* המערכת בודקת את קיומם של רכיבים המוגדרים כחשודים וזדוניים למשל Packers ("אורזי קוד" בתרגום לעברית) . המושג Packers או Protectors מוגדר כרכיב זדוני חיצוני כלומר סוג של "outer shell" שתפקידו להצפין, ליצור כאוס בקובץ, ולהקשות על אנטי-וירוסים ותוכנות בדיקה שונות לזהות את הנוזקה על ידי החבאת הPayload שלהן. על מנת להפוך את הזיהוי והניתוח למורכב עוד יותר, הPackers משתמשים בשיטות Anti-Debugging, Anti-VM ו Code Obfuscation.
* הפרויקט משתמש באלגוריתם "Shannon Entropy" המחשב את האי סדר שיש בקובץ מסוים –מידת הRandomness של הקובץ. כאשר הכותב של הקובץ מבצע דחיסה למידע ומזריק או משתמש בקוד זדוני (שיטות כגון Payload Encryption או Function Call Obfuscation), הוא מעלה את רמת ה- unpredictability של קובץ ההרצה ולכן האנטרופיה עולה. לכן, ניתן לומר שככל שרמת האנטרופיה עולה, גם הסבירות שהקובץ החשוד הוא נוזקה – עולה.
* המערכת בודקת תאימות בין הLinker Versions של ה Optional Header ו Rich Header בתוך הPE . הגרסה של ה Linker שעוזרת לבנות ולהרכיב את הקובץ נשמרת בתוך ה Optional Header והRich Header של הקובץ. כאשר נוזקה מוחדרת לתוך קובץ תמים, ייתכן מצב ובו הDos Stub והRich Header ישתנו ויוחלפו בכאלה של וירוסים אחרים. אם גרסת הLinker שונתה בRich Header של הPE אבל לא בOptional Header , הקובץ שונה וייתכן ומוחבא בתוכו ווירוס.

בינתיים מספק ויש להקדים כמובן על נושא ה VIRUS, PE ועוד

## 2.2 מוצרים קיימים

יש בשוק כיום מגוון מוצרי Anti-Virus. כל מערכת מתמקדת בהיבטים שונים המגדירים נוזקה.

**"Virus Total"** – הוא אתר אינטרנט בבעלות חברת גוגל, המספק שרותי בדיקה של קבצים חשודים להימצאות וירוס מחשב, תולעת מחשב, סוס טרויאני או תוכן מזיק או לא רצוי אחר, כגון רוגלה או תוכנת פרסום. האתר עושה שימוש ביותר מ 55 תוכנות אנטי וירוס שונות וביותר מ 70 מנועי סריקה מגוונים כדי לסרוק ולאתר קבצים חשודים. השימוש הנרחב במגוון תוכנות ומנועים בעת ובעונה אחת מהווה לכאורה פתרון מספק לאיתור והגנה נגד נוזקות שלא זוהו בתוכנת האנטי-וירוס המותקנת במחשב, או בכדי לאשר סטטיסטית את הימצאותה של תוכנה זדונית.

ההסתמכות על מרחב כה גדול של כלים עשוי להגדיל את טעות ה beta, כלומר ייתכנו מקרים שבהם בסופו של דבר, לא יהיה מדובר בווירוס או תוכנה מזיקה.

תוכנת **"Avast Free Antivirus"** מזהה, חוסמת ומסירה את מרבית סוגי התוכנה הזדונית: וירוסים, תוכנת פרסומת, תוכנת ריגול, סוסים טרויאניים ועוד. היא כוללת אבטחה לרשת ה‑Wi‑Fi והגנה בזמן אמת נגד התקפות phishing, אתרים לא בטוחים ואיומים נוספים~~.~~

התוכנה מאפשרת מספר אמצעי התגוננות ומאפיינים ייחודיים:

* הגנה מפני תוכנות ריגול ו Rootkit.
* אפשרות לשחזור קבצים נגועים שנפגעו מווירוסים (Virus Recovery Database).
* Self-Protection (הגנה עצמית) –מקשה על ווירוסים לנטרל את האנטי-וירוס.
* עדכונים אוטומטיים – כאמצעי ברירת מחדל, לתוכנה ולמאגר המידע שלה. כאשר נשלח עדכון לתוכנה - תישמע ההודעה הקולית: "Virus database has been updated" ("מאגר הווירוסים התעדכן"). בנוסף, כאשר חודרת תוכנה זדונית למחשב, נשמעת האזהרה הקולית: "!Caution, a virus has been detected".
* Boot-time scan – אפשרות לקביעת תאריך לסריקת המחשב והסרת קבצים חשודים בעת האתחול הבא של מערכת ההפעלה.
* Virus Chest – כאשר קובץ זדוני מתגלה, יש אפשרות "לבודד" אותו משאר הקבצים במחשב, ובכך למנוע ממנו לפגוע באותם קבצים, עד שאותו קובץ ינוטרל.

הייחוד בפרויקט שלי ובשונה מVirus Total והאחרים מתבטא בשני תחומים עיקריים:

**ברמה ההתנהגותית (דינמית):** "YeshScanner" מבצע בראשונה בדיקה שחורה כלומר חקירה התנהגותית בכך שהוא מדווח על ניסיונות גישה לרכיבים, תיקיות, וpaths ספציפיים במערכת (למשל בתוך הRegistry או ניסיון גישה לקבצים חשודים במחשב). בנוסף, המערכת מבצעת בדיקה לבנה על קובץ ההרצה החשוד בכך שהוא מזהה תבניות ספציפיות המגדירות וירוס או תוכנה זדונית בריצה דינמית למשל port scanning או החדרת קוד אל תוך process מסוים. התועלת מכך היא ש YeshScanner משער בצורה חדה יותר את מטרת קובץ ההרצה, ובכך עוזר להבין טוב יותר את ההיגיון שעומד מאחוריו.

**ברמה הסטטית:** YeshScanner מזהה רכיבים חשודים בתוך מבנה הPE ש Virus Total וAvast לא מדווחים עליהם. YeshScanner נכנס יותר לעומק ה Executable ומזהה רכיבים חשודים ומקרים חשודים בתוך הPE Header . למשל, offsets שונים בהעלאות של dll's ו חישוב אנטרופיה של sections שונים בתוך הExe .

בנוסף, YeshScanner מדווח על קובץ בצורה יותר פשוטה, מובנת וידידותית למשתמש מאשר Virus Total , כך גם שמי שלא מבין ומתעסק בניתוח נוזקות יוכל להבין למה הקובץ שלו זדוני, ולהבין יותר טוב את מטרת הקובץ. לדוגמא, YeshScanner לעומת Avast ו Virus Total לא רק יצהיר שהוא נגוע, אלא יגיע למטרה האמיתית של הקובץ וינסה להבין את ההיגיון העומד מאחוריו.

ניתן לייצג את ההבדלים בטבלה:

|  |  |  |  |
| --- | --- | --- | --- |
| תבחינים \ פרויקט | YeshScanner | Virus Total | Avast |
| שימוש במנועי סריקה מגוונים | לא משתמש במנועים שונים | Virus Total משתמשת ביותר מ 70 מנועי סריקה על מנת למצוא תוכנות חשודות | משתמש במגוון סורקים וחבילת אמצעי אבטחה ומודלים. לכל סורק ישנו תפקיד אבטחתי שונה, הגדרה ורמת אבטחה בדרגה שונה. |
| יכולת לזהות התנהגויות ספציפיות של נוזקות | ניתן לשער (בצורה סטטיסטית) על התנהגות חשודה של נוזקה | Virus Total מציגה תכונות בקובץ המצביעות על התנהגויות חשודות אך לא בצורה חד משמעית | מסוגל לדווח על סוג הווירוס החשוד כלומר מאיזה משפחת וירוסים הוא נמצא (למשל Win32 Malware) |
| טיפול בשפות תכנות שונות | מטפל רק בשפות c#, c, c++ | מטפל בכל שפות התכנות | מטפל בכל שפות התכנות |
| פירוט על מבנה הPE | מפרט על כל הPE sections והheader והאם משהו בהם נמצא חשוד | מפרט על כל הPE sections והheader אך ללא בדיקה מעבר, קרי הדיווח לא יראה על זדוניות הקובץ מבחינת הPE | אין פירוט |
| פירוט על פונקציות הWinApi והimports | פירוט ודיווח מלא על פונקציות הWinApi והimports | פירוט חלקי ודיווח חלקי על פונקציות הWinApi וה imports של הExe . | אין פירוט |
| אופציה לבדוק URL חשוד / IP חשוד | האופציה קיימת | האופציה קיימת | ישנה אפשרות לסרוק את הרשת הביתית ולראות האם בוצעה כניסה לאתר חשוד |
| נדרש תשלום | לא נדרש לשלם על האפליקציה | יש אפשרות לניתוח יותר מפורט בתשלום - אפשרות Premium אשר כוללת העלאת קובץ בגודל בלתי מוגבל, API המשתמש ביותר אפשרויות חקירה (similarity search, clustering, behavioral information) ושימוש ב YARA בשביל למצוא malwares דומים לקובץ החשוד | יש אפשרות לניתוח יותר מפורט בתשלום - אפשרות Premium הכוללת Sandbox של Avast, אפשרות לבצע פעולות מקוונות למשל כניסה לחשבון הבנק בצורה מאובטחת, הגנה מהתקפות Phishing |
| אפשרות לראות גרפים המציגים קישור בין רכיבים שונים של הExe | אין כזאת אפשרות במערכת | ישנה אפשרות לראות גרפים המציגים קשר חשוד בין קבצים, URL שונים, Domains שונים, כתובות IP שונות הנמצאים בExecutable | אין אפשרות כזאת במערכת |
| אפשרות Community | אין כזאת אפשרות במערכת | ישנה אפשרות שmalware researchers אחרים יחוו דעתם על הקובץ החשוד | אין אפשרות כזאת במערכת |
| סריקת Directory | קיימת אפשרות כזאת במערכת | לא קיימת אפשרות כזאת במערכת | קיימת אפשרות כזאת במערכת |

# 3. תוצר סופי

## 3.1 תיאור הפרויקט

המערכת בנויה על שלושה רבדים – חקירה התנהגותית (הרצת הקובץ והתבוננות במעשיו כלומר ,Blackbox testing חקירה סטטית (בחינת פרטים ורכיבים חשודים הנמצאים בתוך קובץ ההרצה, כלומר Whitebox testing), ותקשורת Client-Server על מנת להעביר קבצים בין הVirtual Machine לHost Machine (המחשב)

רכיבי המערכת הינם:

* מכונה וירטואלית (Virtual Machine) – מכונה המותקנת ב ,Host Machine אשר מיושם בה שרת (server). המכונה מהווה סביבה בטוחה אשר מריצה את הווירוס ומחזירה דיווח ל Host Machine .
* המחשב (Host Machine) – בו מיושם הלקוח, מכיל את פונקציונליות החקירה הסטטית על הווירוס, את ההצגה של הדיווח מהVirtual Machine ואת ניתוח הקובץ החשוד על פי הHash שלו.
* sender – תוכנה הנמצאת על ה Host Machine ומתפקדת כלקוח, השולחת את קובץ ההרצה החשוד לVirtual-Machine .
* server – תוכנה המתפקדת כשרת בתוך הVirtual-Machine , אשר מקבלת את קובץ ההרצה החשוד מה Sender , מריצה אותו ומחזירה את הדיווח המפורט ללקוח.
* Analyzer – תוכנה המותקנת ב Host המבצעת את הניתוח הסטטי על הווירוס.

בהעלאת מערכת האנטי-וירוס, המשתמש יוכל לטעון קובץ הרצה לבחירתו ולהעבירו ניתוח דינאמי ב server וסטטי ב analyzer. במהלך הניתוח יוצג Loading Circle שיסתיים ברגע שיתקבל הדיווח המלא. בסיום הניתוח, יקבל המשתמש מפרט תצוגתי בשלוש רמות:

* בחלון ה Dynamic Analysis יוצגו כל פונקציות הWinApi שקובץ הExe השתמש בהן. הפירוט יכלול את כל הפרמטרים המועברים לפונקציות, time-stamp לכל פונקציה - מספר שניות שעברו מאז תחילת הריצה עד לקריאה לפונקציה.
* בחלון הStatic Analysis יוצג פירוט של מבנה הPE של הקובץ, כולל הimports של הקובץ וכתובות ה sections שלו, פירוט לגבי נתונים חשודים שנמצאו במבנה הPE היכולים להצביע על קיום נוזקה.
* בחלון הHash Analysis , הקובץ יועלה לVirus Total במסגרת רכיב ה Analyzer לצורך ניתוח מעמיק יותר, ויוחזר דיווח מפורט הכולל את כל מנועי הסריקה שמצאו את הExe כקובץ חשוד, והפרטים המלאים עליהם.

ממשק המשתמש יהיה ברור ויאפשר התנהלות פשוטה, כך שהשימוש במערכת יהיה אינטואיטיבי למשתמש.

## 3.2 אלגוריתמים עיקריים

1. שליחת הקובץ למכונה הווירטואלית (שרת) וקבלת דיווח בתקשורת

אחרי העלאת הקובץ למערכת , מתבצעות מספר פעולות:

* הHost Machine מתחבר למחשב הווירטואלי באמצעות TCP Socket אשר ממתיi להתחברות המשתמש וטעינת הקובץ החשוד.
* הקובץ מתקבל במכונה הווירטואלית, והשרת מזריק את הDLL אל תוך הExe כמפורט באלגוריתם הבא.
* ה DLL יוצר קובץ LOG.txt המכיל דיווח מפורט על פונקציות הWinApi של הExe
* הקובץ נשלח בחזרה למחשב הHost Machine להמשך הניתוח סטטי ברכיב ה Analyzer על פי ה Hash

1. החדרת ה DLL המכיל את הHooks אל תוך הקובץ החשוד

המערכת בונה DLL אשרמכיל WinApi Hooks המיועדים לזהות הימצאות של נוזקה לפי 5 פרמטרים עיקריים:

* **Files** (File Damage) –המערכת סורקת האם בוצעה בקשת פתיחה של קובץ חשוד, אשר הגישה אליו היא לא שגורה (למשל system32) או מצב בו קובץ מנסה לכתוב קובץ חדש בתוך אחד מהDirectories הרגישים במחשב (למשל C:\Program Files). הגדרת הקבצים והDirectories מתבצעת במנהל המערכת ונשמרת (ונקראת) מתוך קובץ TXT.
* **Registry** (Registry Damage) – ה DLL בודק האם בוצעה פתיחה או גישה ל Registry Key חשוד - הכוונה כאלה שיכולים לשנות הגדרות מהותיות של המחשב. למשל, אם הקובץ מנסה לגשת אל המפתח HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Run תעלה הסבירות שהוא מנסה להעלות את עצמו בכל פעם שהמחשב נפתח. הDLL בודק גם יצירה של מפתחות חדשים וQuery של מפתחות שקובץ רגיל לא אמור לגשת אליהם. הגדרת המפתחות מתבצעת במנהל המערכת ונשמרת (ונקראת) מתוך קובץ TXT.
* **CPU** ((CPU Damage ווירוסים רבים מדגימים צריכה גבוהה יחסית של CPU. הDLL בודק עבור כל קריאה לפונקציית WinApi בקובץ את הCurrent CPU של המחשב, ומתריע על חציית גבול חשוד של צריכת CPU. הגדרת גבול זה מתבצעת במנהל המערכת ונשמרת (ונקראת) מתוך קובץ TXT.
* **Port Scanning** – ה DLL בודק רצף של פתיחת Socket ואחריו Connect לפחות 3 פעמים באמצעות ה. WinSock רצף המצביע על העלאת הסבירות להימצאותה של נוזקה.
* **Process Hollowing and Code Injection** – ה DLL בודק האם בוצעה החדרת קוד אל תוך אחד מה Processes הרצים כעת במחשב. הבדיקה מבוצעת על הProcess שנפתח באמצעות פונקציית הOpenProcess ושומרת את הHandle . אחר כך, מתבצעת בדיקה של רצף פונקציות המגדירותInjection - OpenProcess 🡪 VirtualAllocEx 🡪CreateRemoteThread כדלקמן: אם נמצא כי המשתמש פתח Process עם Handle מסוים (השתמש בHandle והקצה זיכרון), ולאחר מכן הפעיל Routine בכתובת הזיכרון שהקצה בה מקום, הרי שבוצע Code Injection.

1. ניתוח הPE מעמיק של קובץ ההרצה על מנת למצוא וירוס חבוי בתוך Exe תמים

כאשר המשתמש מגיע לחקירה הסטטית (Static Analysis):

* הבדיקה הראשונית שנעשית היא עבור כל section בתוך הPE ומתבצעת לאיתור flags דלוקים אחד כדלקמן: IMAGE\_SCN\_MEM\_EXECUTE, IMAGE\_SCN\_MEM\_READ, IMAGE\_SCN\_MEM\_WRITE.

וירוסים רבים משנים את התכונות של sections שונים על מנת לבצע שינוי בקובץ או להזריק לתוכו נוזקה. אם נבחין כי שלושת הדגלים האלה דלוקים, הרי שזהו לא צירוף מקרים וכנראה הקובץ שונה.

* הבדיקה השנייה היא עבור הImports של הקובץ. המערכת נכנסת אל תוך הIAT (Import Address Table) של הקובץ ובודקת לגבי כל DLL שהקובץ עושה לו import, האם ישנה אי תאימות בין הכתובת של DLL's השונים, כלומר אם הimports נמצאים בsections שונים בPE ולא כולם בsection אחד - דבר זה מכונה Fractioned imports , היכול להצביע על קובץ שמבנה ה PE הרגיל שלו שונה וכנראה הוחדרה לתוכו נוזקה. לדוגמה, אם ה DLL's של מערכת ההפעלה - adavpi32.dll ו kernel32.dll נטענים בכתובת 0x408 ב IAT, אך WinSock.dll  מועלה בכתובת 0x2408. האי סדר הנ"ל ב imports של קובץ ההרצה הוא תופעת לוואי של החדרת וירוס, וייבוא ה imports שלו משנה את מבנה הקובץ.
* הבדיקה השלישית היא לתאימות של הRich Header וOption Header. המערכת בודקת האם יש תאימות בין גרסאות הLinkers של שני הHeaders , ואם לא, הרי שעולה החשש שקובץ רגיל שונה.

1. בדיקה עבור Strings, Packers, Protectors של הקובץ וחישוב האנטרופיה של הSections השונים

בשלב הבא של החקירה הסטטית, תתבצע חקירה על "אורזי קוד" וStrings חשודים:

* תתבצע סקירה של הקובץ אל מול קובץ חתימות YARA של יותר מ 9000 חתימות שונות שאמורות לזהות Packers שונים האמורים להסוות את ההימצאות של הנוזקה ולהקשות על הניתוח של הExe . החקירה תהיה בעזרת ספרייה ב Python המובנית למצוא התאמה בין חתימות מקובץ YARA לקובץ הרצה.
* קובץ החתימות מורכב בצורה הבאה: כל חתימה תכיל כמה רצפי בתים מסוימים המגדירים Packer מסוים (למשל 0F 80 38). התנאי לחתימה היא שלפחות אחד מהרצפים האלה יופיע בקובץ ההרצה החשוד. אזהה את החתימות שלא כתובות בידי בכך שתהיה להן את הtag - "Peid" והחתימות הכתובות על ידי יהיו עם הtag - "Elad".
* סקירה של הקובץ אל מול קובץ YARA האמור לזהות Strings של פונקציות WinApi חשודות והימצאותם שלStrings הקשורים ל מפתחות Registry חשודים או Files. כל חתימה מכילה את הStrings הרלוונטיים למשפחת הווירוס .
* האנטרופיה של קובץ היא מדד האי הסדר בקובץ או כמה הוא שונה, ככל שהמדד יותר גבוה כך עולה הסבירות שהקובץ מכיל בתוכו נוזקה. במערכת מתבצע שימוש באלגוריתם "Shannon Entropy" לחישוב האנטרופיה של קובץ או הPE sections השונים של קובץ. בדיקת האנטרופיה תתבצע על הקובץ עצמו, ועל הPE Sections שלו ותתבצע בהשוואה לExe אחר תמים, שידוע שאין בו שום דבר חשוד.

1. בדיקת הHash של הקובץ אל מול VirusTotal בנוסף בדיקת הFuzzy Hash שלו בשימוש ssdeep.

בשלב הHash Analysis , יתבצעו מספר בדיקות לקובץ על פי הHash שלו:

* בהתחלה, תתבצע ההעלאה של ה md5 Hash של הקובץ אל תוך VirusTotal בשימוש במודל Requests המאפשר להעלאות קבצים ותכנים לאינטרנט עם Python. הדבר מבוצע רק כדי להעמיק את החקירה עם התקשרות והתממשקות לאתר חיצוני שיש לו גישה ל מגוון של מנועי סריקה. המטרה בכך היא לאשש ולהוסיף לרמת הסמך של הניתוח גם את ניתוח מנועי הסריקה הרבים והמוכרים שמספק .VirusTotal מעבר לכך, לא תערך השוואה של תוצאה זו, או חקירת הנתונים ש Virus Total מספק.
* אם ההעלאה תבוצע כראוי, המערכת תכתוב לקובץ את המידע שהתקבל מVirus Total ולאחר מכן תציג בחלון ה Hash Analysis כמה מנועי סריקה מצאו את הקובץ כחשוד, וכמה לא. המערכת תראה את הרשימה של המנועים שמצאו את הקובץ כחשוד כולל הפרטים המלאים שלהם: שם, קטגוריה, סוג הווירוס, שיטת הסריקה, ועדכון אחרון.
* תתבצע בדיקה עבור ssdeep hashes המרמז על מידת התאמה בין שני Hashes של קבצים החשודים. המערכת תיקח את הssdeep Hash של הקובץ החשוד, ותבדוק כמה הוא דומה לssdeep Hashes של קבצים חשודים אחרים – הלקוחים מתוך קובץ TXT שמכיל יותר מ 50,000,000 Hashes שונים (ככל שהמערכת תשווה בין קובץ ההרצה לבין מדגם רחב יותר של נוזקות אחרות, יעלה הסיכוי למציאת זדוניות בקובץ). אם יראה כי הקובץ דומה לאחד מהHashes ביותר מ 50 אחוז, תעלה הסבירות כי הקובץ הוא נוזקה. הדבר גם מאפשר בדיקה של וירוס חבוי בקובץ תמים, מכיוון שמטרת שיטת הFuzzy Hashing הוא למצוא דמיון בין קבצים. לגן, הFuzzy Hash של קובץ תמים אשר יחביא בתוכו וירוס יימצא דומה מספיק לFuzzy Hash של קובץ זדוני.

1. סריקת Directory וניסיונות הגלישה ברשת באמצעות Virus Total

למשתמש תינתן האופציה לסרוק Directory שלם במחשב ואת הרשת שלו באופן הבא:

* אם המשתמש יבחר לסרוק Directory, המערכת תעלה כל קובץ ל Virus Total, ותכתוב לקובץ כמה מנועי סריקה מצאו את ה קובץ כקובץ חשוד, וכמה מצאו אותו כקובץ לא מזיק. לאחר מכן, הדבר יוצג בחלון Directory Analysis.
* אם המשתמש יבחר לסרוק את ניסיונות הגלישה ברשת שלו, המערכת תשלוף את האתרים האחרונים שהוא נכנס אליהם באמצעות כניסה אל ה DNS Cache.
* YeshScanner"" יכנס אל ה DNS Cache של המחשב באמצעות הפקודה "ipconfig /displaydns" ותהיה לו גישה אל ה IP's האחרונים שהמחשב ניגש אליהם.
* המערכת תעלה את ה IP's אל Virus Total, ותראה האם יש לפחות 5 מנועי סריקה שונים המצביעים על ה IP כחשוד.
* לאחר אישור משתמש יופעל Pydivert כדי לחסום את ה IP ולא לשלוח פקטות אינטרנט שמיועדות אליו. שכך לא יהיה ניתן לגשת אל ה IP יותר וכמו כן תוצג הודעה למשתמש המציגה את הIP החשוד.

1. העברה ל VAULT (CHEST)

* אם קובץ ההרצה נמצא כווירוס, משמע הוא נמצא כחשוד בלפחות שתי קטגוריות (מבין (Dynamic Behavior, Static Behavior, Hash Analysis הקובץ יוכנס לתיקייה נפרדת במחשב שנקראת "Found\_Virus" וישונו ה Attributes שלה כך שהיא תהיה Hidden – קרי המשתמש לא יוכל לראות אותה ובכך לא יוכל להשתמש בקובץ.
* כעת תינתן הבחירה למשתמש האם הוא רוצה להשאיר את הקובץ במחשב או שלא. אם לא, הקובץ ימחק לאלתר ולא יהיה ניתן להריץ אותו.
* הפתרון לא טוב, בייחוד כאשר מתייחס לסריקת תיקיה.

## 3.3 דרישות ואילוצי פתרון

לפרויקט מספר אילוצים:

* ה Virtual Machine נדרש להיות פתוח ידנית מהתחלת ריצת הפרויקט כדי לאפשר תקשורת והעברת הקובץ לסביבה בטוחה. לא ברור מדוע נדרש כאילוץ, למה לא לפתוח אותו מרחוק? לפחות לנסות?
* הפרויקט עובד רק על Executables מסוג C, C#, C++ כלומר המערכת תוכל לנתח במלואה רק Exe's הכתובים בשפות הללו. ישנה מגבלה על סוגי ה WinApi, האפשרות היחידה במערכת תהיה רק ניתוח פונקציות WinApi מסוג 32 bit יתרה מזאת, המערכת תעבוד בצורה חלקית על קבצי Python ותצליח לזהות פונקציות של קבצי PY והפרמטרים שלהם.
* המערכת תרוץ רק על מערכת הפעלה מסוג Windows 7 או Windows 10 מכיוון שאלו המערכות אשר ממצות את היכולת המקסימלית של הספריות ppdeep, pydivert, yara-python, pefile

למערכת מספר דרישות:.

* המערכת נדרשת לתמוך בכמות מספקת של פונקציות WinApi בתחומים שונים – Registry, Communication, Files על מנת להגדיר התנהגות חשודה.
* המערכת נדרשת לתמוך בביצועים מהירים ולבצע סריקה מלאה של הקובץ ללא המתנה לאורך זמן רב לצורך חווית משתמש נוחה ומהירה.
* המערכת נדרשת להכיל מאגר חתימות של Packers ומאגר מספיק גדול של Fuzzy Hashes על מנת לסרוק בצורה מספקת את הקובץ החשוד ולהשוות אותו למספיק חתימות כאשר המאגרים ישמרו כקבצים בתוך הפרויקט (Packers כקובץ yar וFuzzy Hashes כקובץ TXT).
* המערכת תציג סטטיסטיקה קרובה לזו של Virus Total ו Avast.

## 3.4 ממשקים למערכות חיצוניות

המערכת מתממשקת עם האתר Virus Total המתמחה בזיהוי וירוסים על פי תוכנות שונות ומנועי סריקה. המערכת עושה זאת באמצעות הVirusTotal API ומודל Requests בפייתון. המערכת מעלה את הקובץ או את הHash שלו מתוך התוכנית עם הHeaders הרלוונטיים – קישור לאתר ו API key, ומקבלת עליו דיווח מפורט בחזרה בפורמט JSON.

בנוסף לזאת, המערכת מתממשקת עם YARA - כלי לאיתור וזיהוי תוכנות זדוניות (או קבצים אחרים) המשתמש בסט כללים וחוקים המאתרים מאפיינים ספציפיים שיסייעו לזיהוי התוכנה כזדונית .המערכת יוצרת סט של חתימות על מנת לזהות Packers וחוקים האמורים לזהות Strings חשודים בקובץ.

## 3.5 התייחסות לנושא אבטחה

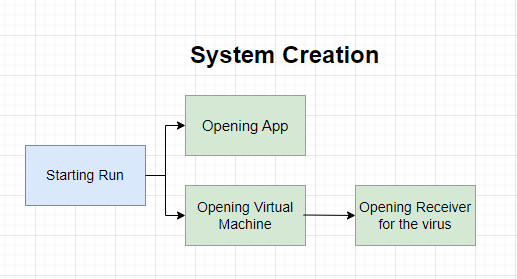
מכיוון שהווירוס או הקובץ החשוד רץ על המחשב לצורך חקירה דינמית, יש להריץ אותו בסביבה בטוחה על מנת שלא יעשה נזק מהותי למחשב שהוא בלתי הפיך – לכן המערכת תריץ אותו במכונה וירטואלית (Virtual Machine) על מנת שלא יהיו נזקים משמעותיים למערכת בעת הרצת הקובץ החשוד.

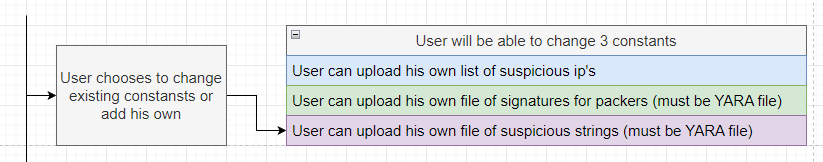
בכניסה למערכת יתבקש המשתמש להיכנס עם שם משתמש וסיסמא. כך, תגן המערכת מכניסת משתמשים לא מורשים (רשומים).

## 3.6 ממשק משתמש

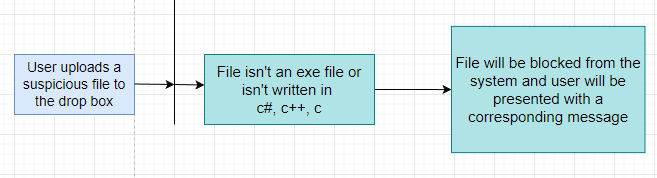
## 3.7 תרחישים עיקריים

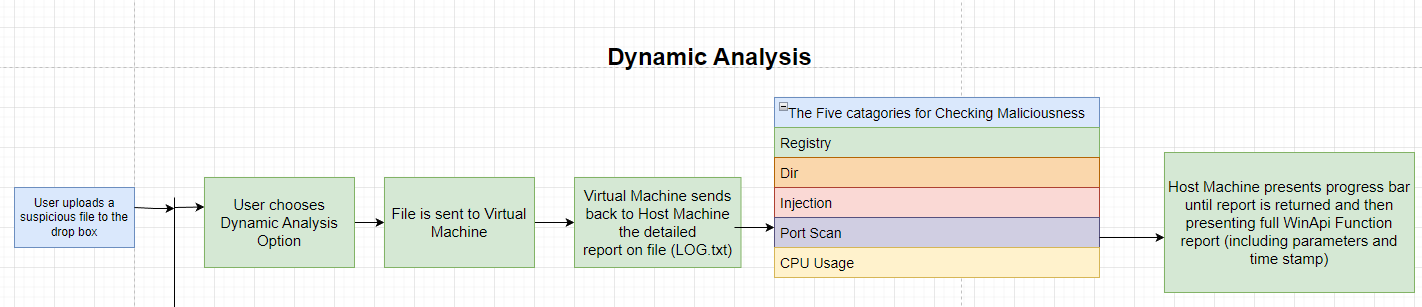
תרחיש 1 – יצירת המערכת (מתבצע באופן ידני)

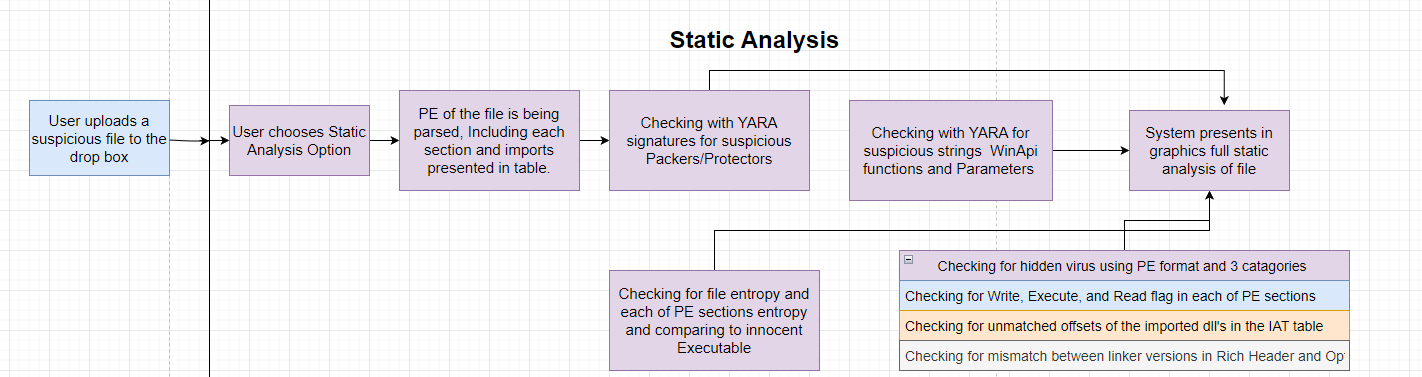


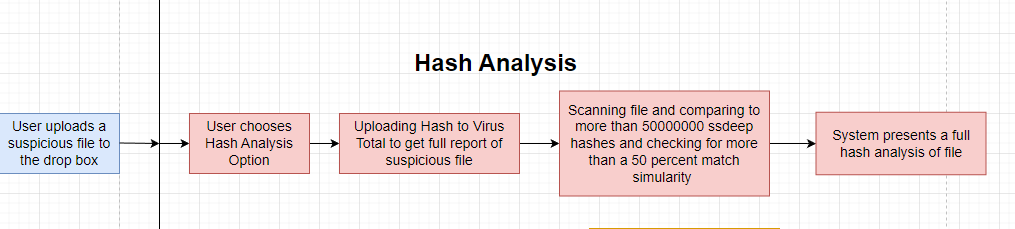
תרחיש 2 – הגדרת קבועים (מתבצע באופן ידני)

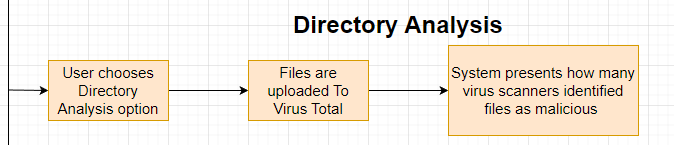
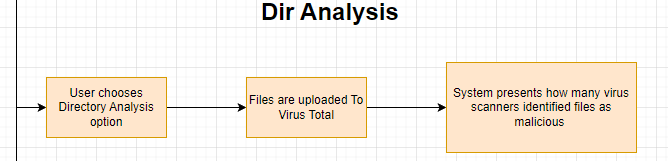
תרחיש 3 – טיפול בקובץ שגוי (מתבצע באופן אוטומטי)

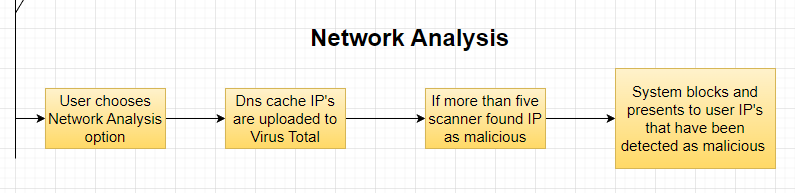


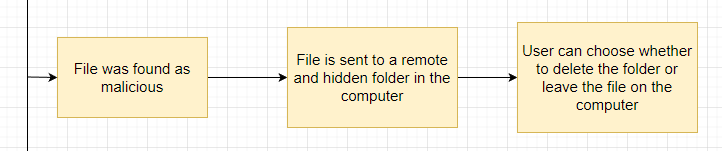
****תרחיש 4 - ניתוח התנהגותי (מתבצע באופן אוטומטי)

תרחיש 5 – ניתוח סטטי (מתבצע באופן אוטומטי)

תרחיש 6 – ניתוח Hash (מתבצע באופן אוטומטי)

תרחיש 7– חקירת ה Dir (מתבצע באופן אוטומטי)

תרחיש 8 – חקירת הרשת (מתבצע באופן אוטומטי)

תרחיש 9 – הכנסת הווירוס ל QUARNTINE – מתבצע באופן אוטומטי)

# 4. תהליך כתיבת הפרויקט

## 4.1 תהליך הפרויקט

התחלתי את הפרויקט עוד בתחילת הקיץ כשהתחלתי להתעניין בכל הנושא של וירוסים והנדסה לאחור (Reverse Engineering). התחלתי לכתוב קטעי קוד שמטרתם הייתה זדונית ובכך למדתי גם איך לזהות כאלה תוכניות, ואיך להגיע למטרתן גם כשאני לא רואה את הקוד בעצמי. למרות זאת, לא ידעתי בדיוק איך רציתי שהפרויקט שלי יהיה בנוי ואחרי זמן רב ומחשבה רבה, בנוסף להתייעצות עם המנחה, החלטתי להתחיל לכתוב את הפרויקט הסופי בתחילת השנה. בהתחלה כהוכחה שאני יודע לבצע את הפרויקט (POC), הראיתי שאני יודע לבצע Hooking לפונקציות WinApi מתחומים שונים. המטרה שלי הייתה לשלוח את הקובץ למכונה וירטואלית ולקבל קובץ מפורט על פונקציות ה WinApi שלו בחזרה (קובץ LOG.txt). צלחתי מסוכה זו.

לאחר מכן, יצרתי מערכת יותר מורכבת עם בדיקות אחרות שאמורות לחקור את ההתנהגות ומטרתו של הקובץ. התחלתי לחקור איך אני יכול לזהות התנהגות חשודה ומרכיבים חשודים באמצעות בדיקה סטטית. כתבתי קטעי קוד, הן בשפת C++ והן בשפת Python שכל אחד מהם מטרתו לזהות מרכיבים שונים בקובץ הרצה בצורה סטטית. קטעי הקוד האלה מטרתם לתת אינדיקציה יותר טובה על התנהגות הקובץ ומטרותיו גם בלי להסתכל על הקוד שלו למשל חקירת מבנה ה PE, Strings חשודים, וחתימות קבצים.

בשלב הבא בפרויקט עסקתי בתחום ה Hashing ואיך אפשר לזהות קבצים חשודים באמצעות הHash שלהם. וביצעתי התממשקות עם VirusTotal בנוסף לטכניקות מתקדמות כמו Fuzzy Hashing על מנת לדעת כמה דומה הקובץ החשוד לקבצים חשודים אחרים. לבסוף נותר לי לייצר ממשק משתמש נוח, ידידותי למשתמש ונוח לעין, עם סטנדרט גבוה, שגם היווה אתגר לא פשוט, לאור היכרותי המועטה עם PYQT.

בנוסף ללמידה מעמיקה ומשמעותית של החומר שהפרויקט עוסק בו, למדתי המון על כתיבת פרויקט. יש שוני רב בין כתיבת תוכנה קטנה, לבין כתיבת מערכת גדולה ומורכבת. אני חש כי כתיבת עבודה זו הקנתה לי ידע רב ומיומנות שימושית לעתידי.

## 4.2 אתגרים ואופציות שונות למימוש

במהלך הפרויקט נתקלתי בכמה אתגרים:

* בהתחלה הייתה לי התלבטות אם אני רוצה לעשות את הפרויקט רק בשפה אחת (C++) או לשלב גם את Python. כשהתחלתי לכתוב את הפרויקט, לא ידעתי איזה פונקציונליות ברצוני לממש, וכן רוב הטכניקות שהכרתי דאז היו בשפת CPP בלבד. עם הזמן, כשהתקדמתי בתכנון וכתיבת הפרויקט, הבנתי שעליי לשלב עוד שפה מרכזית על מנת להגיע לחקירה מעמיקה יותר של מטרת הקובץ. בחרתי את Python מכיוון שיש לה המון ספריות שמתעסקות ב Malware Analysis ו Reverse Engineering וחקירת קובץ בצורה סטטית ו לפי Hash. יתרה מזאת, הגעתי להבנה שאני עומד לבצע את הגרפיקה בpyqt ולא ב Qt שכתוב ב CPP . לבסוף שילבתי בהצלחה בין השפות על ידי הפיכת קבצי ה C++ לExe שאני מריץ דרך Python.
* בתחילת הפרויקט, מצאתי קושי להבין איזה טכניקת Hooking ברצוני ליישם על מנת לחקירה כמה שיותר עמוקה על מטרת הקובץ, והתנהגותו. התלבטתי בין IAT Hooking שאמורה רק למצוא את הפונקציות שהקובץ עושה להן Import מכל DLL, לבין Relative Inline Hooking , שמשנה את כתובת ה Jump הרלטיבית לכתובת של הפונקציה החליפית שלי, לבין Trampoline Inline Hooking, שבעזרת פקודות אסמבלי כגון PUSH ו RETN , אני יכול להכריח את התכנית לקפוץ לפונקציה שלי במקום לפונקציה המקורית. לבסוף, החלטתי להשתמש ב Trampoline Inline hooking, מכיוון שאני חושב שהיא מספקת הכי הרבה מידע על הפונקציה כגון פרמטרים וסוגם, והיא הרבה יותר נוחה לשימוש וגם קריאה.
* במהלך כתיבת המערכת החלטתי שאני רוצה להשתמש בDebuggers שונים על מנת לייעל ולהעמיק את המחקר על קבצי Executables. היה לי אתגר אך צלחתי אותו בהבנת הDebuggers, שימושם ואיך הם יתרמו לי, אך הצלחתי להריצם רק בצורה ידנית. מכיוון שהפרויקט שלו הוא Automated כלומר רץ בצורה אוטומטית, ולא מבצע חקירה מסוימת בצורה ידנית בשום שלב, חיפשתי דרכים שונות להריצם באופן אוטומטי במגוון שפות כמו CPP, Python, C# , אך לא מצאתי דרך לעשות זאת. לבסוף החלטתי על האופציה ולהתמקד בכיווני מחקר אחרים.
* כשרציתי להתחיל לכתוב את ממשק המשתמש, הייתה לי התלבטות בנוגע לאיזה כלי אני אשתמש – Pyqt או WinForms. בהתחלה נטיתי לכיוון WinForms בגלל שיש לו Designer שלדעתי יותר קל ונוח למשתמש מאשר Pyqt . אך כאשר התחלתי להשתמש ב Python לטובת מחקר וכתיבת הפרויקט, החלטתי להשתמש ב Pyqt, מחד גיסא כדי להרחיב את אופקיי בכלי, ומאידך גיסא כדי להיצמד לPython שתהיה השפה העיקרית של הפרויקט.

# 5. מרכיבי פתרון

## 5.1 תיחום הפרויקט

* תקשורת - שרת המיושם על גבי ה Virtual Machine הממתין לקובץ ולקוח הנמצא בתוך המחשב ה Host כדי להעביר את הקובץ החשוד באמצעות תקשורת TCP מהמחשב למכונה וקבלת הדיווח מהמכונה למחשב.
* אבטחת מידע – כניסה מאובטחת של משתמש לאפליקציה בעזרת שם משתמש וסיסמה, ושימוש ב VM
* תצוגה – גרפיקה ידידותית למשתמש הכתובה בPython PYQT5
* מבנה נתונים – שמירה ברשימות, מילונים ומשתנים גלובליים - מידע בנוגע לבדיקות וניתוח הקובץ, ולשתף אותו בצורה נוחה בין קבצי הקוד.
* מערכות הפעלה – שימוש רב ב Process ו Threads, שימוש ב WinApi Hooking הכתוב בC++ , שימוש ב DLLs, DLL Injection. חקירה וניתוח מעמיק של ה PE של הקובץ
* ארכיטקטורת קוד – שימוש רב במחלקות, קבצים ומימוש של מודל ה MVC .כל זאת כדי להקל על תהליך כתיבת הקוד וגמישות בשינויים.
* תיעוד – תיעוד אקטיבי וניהול GIT מוקפד (שמירת גרסאות שונות), בנוסף לכתיבת ספר פרויקט

## 5.2 סביבת העבודה (טכנולוגיה)

**שפות התכנות:**

אשתמש בC++ על מנת לכתוב את הDLL שמכיל את הWinApi Hooks בשביל לחקור את הקובץ בצורה דינמית כאשר הוא מגיע למכונה הווירטואלית. בנוסף, את הDLL Injection אכתוב גם בשפת C++ בשימוש WinApi. בנוסף אשתמש בC++ על מנת לכתוב PE Parse ולהגיע לIAT Table של הקובץ.

אשתמש בPython בשילוב עם C++ כדי לבצע חקירה סטטית מלאה על הקובץ – כולל ניתוח מעמיק של הPE בעזרת ספריות מתקדמות ושילוב עם קבצים מסוג YARA בשביל לזהות וירוסים על פי חתימות וStrings שונים.

אשתמש בשפת Python למטרות הבאות:

* לכתוב את הלקוח ואת השרת האמורים להעביר ביניהם את הקובץ החשוד למכונה הווירטואלית, ואת הדיווח המפורט בחזרה למחשב הרגיל.
* לבצע את החקירה על פי Hash והעלאתו לVirusTotal באמצעות הVirusTotal API בשילוב עם Python.
* כדי לבצע חקירה על פי Fuzzy Hashing, וכן של ה Directory והעלאת הקבצים בתוכו אל Virus Total. יתרה מזאת, אעלה IP's הנמצאים בתוך ה Dns Cache ל Virus Total על מנת לבדוק האם הם זדוניים.

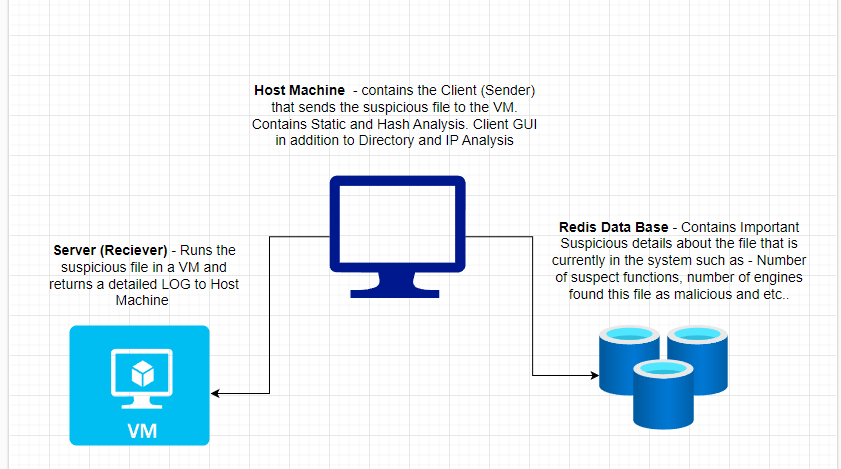
לבסוף, אשתמש גם בPython כדי ליצור גרפיקה מרשימה וידידותית למשתמש הכתובה בשפת PYQT5.

**סביבות פיתוח:**

* Visual Studio לשפת C++
* Pycharm לשפת Python

## 5.3 מבט טופולוגי

המערכת מורכבת ממכונה וירטואלית, המחשב ה Host , שרת (Receiver) שפועל במכונה הווירטואלית ותפקידו לקבל את הקובץ מהלקוח (Sender) שנמצא על מחשב ה Host ומתחיל את חקירת הקובץ. התקשורת בין מחשב ה Host לבין ה VM מתבצעת על גביSOCKET . בנוסף לזאת, למערכת יש מסד נתונים (מסוג Redis) השומרת את פרטיו של הקובץ ורכיבים חשודים שנמצאו לגביו לפי ה MD5 Hash שלו.



## 5.4 מבנה נתונים

DLL Hooks:

* suspicious\_functions – רשימה השומרת את כל הפונקציות WinApi שמבוצעות עליהן Hook.
* fnMap – מילון השומר כמפתח את שם הפונקציה שעושים אליה Hook ובתור ערך את הכתובת שהפונקציה המקורית מופנית אליה
* parameters – רשימה השומרת את המידע מקובץ ה Parameters.txt שהמערכת קובעת לפני הריצה השומר פרמטרים חשודים בנוגע לRegistry, Ports, Files .
* patch – רשימה של chars המכילה את הפנייה לפונקציה של המערכת שמוחדרת לתוך כל Paralogue של כל פונקציית WinApi כחלק מתהליך ה Hooking .
* addresses – רשימה השומרת את הכתובת של כל פונקציית WinApi ברגע שהיא מועלית מה DLL שאליו היא שייכת.
* original – רשימה השומרת את הכתובות המקוריות של הפונקציות, כדי שאחר כך יהיה אפשר להמשיך את הריצה שלהן בצורה יעילה.

DLL Injection:

* name – רשימה שהיא השם של ה DLL שאותו נרצה להחדיר לתוך הקובץ החשוד
* pi – struct השומרת מידע חשוב על ה Process כלומר הקובץ שאלינו אנו מחדירים את ה DLL כמו Handle ו ID.
* memAddr – כתובת הזיכרון שהקצנו בתוך הקובץ כדי להחדיר לתוכו את הDLL .
* remote\_thread – Handle שמריץ את הDLL בצורה מרוחקת בתוך הקובץ החשוד שהמערכת מקבלת.

Yara:

* packers\_virus – רשימה השומרת את החתימות שנמצאו בקובץ בנוגע ל Packers או Obfuscators ש "מגנים" עליו.
* suspicious\_strings\_matches – רשימה השומרת את החתימות (בצורת Strings) שנמצאו בקובץ.

Hash Scan:

* upload\_url – מחרוזת השומרת את הקישור של Virus Total שאליו יועלו כל ה Hash של הקבצים החשודים.
* file\_id – מספר השומר את המזהה של כל קובץ שהועלה ל Virus Total ואיתו ניתן לקבל גם את ניתוח הקובץ.
* status – מחרוזת השומרת את סטטוס הקובץ, כלומר האם הוא מוכן לניתוח (completed) או עדיין לא וצריך לבדוק שליחה מחודשת (queued)
* results – מילון בצורת JSON השומר את תוצאות ניתוח הקובץ, ובתוכו מידע מפורט על כל המנועים שמצאו את הקובץ כחשוד. זאת ועוד, כמות המנועים שמצאו את הקובץ כזדוני וכמות המנועים שלא מצאו אותו כזדוני.
* Headers – מילון השומר פרטים החיוניים להעלאת קבצים ו Hashes לאינטרנט (KEY\_API, User Agent , Encoding)

Directory Analysis:

* num\_files – מספר הקבצים ב Directory הנבחר
* files – מילון השומר את הבתים של הקובץ הנבחר בנוסף ל Path המלא שלו.
* analyse\_result – מילון בצורת JSON השומר את כמות המנועים שמצאו את הקובץ כחשוד, נוסף למידע מפורט על כל אחד מהם.
* status – מחרוזת השומרת את סטטוס הקובץ, כלומר האם הוא מוכן לניתוח (completed) או עדיין לא וצריך לבדוק שליחה מחודשת (queued).
* hashsum – מספר Hash של הקובץ במקרה של שליחה מחודשת (הפעם לפי Hash ולא נשלח את הקובץ עצמו).

IP Analysis:

* ip\_pattern – תבנית regex הנועדה לוודא הימצאות של כתובת IP.
* ip\_match – רשימה השומרת את ה IP's שנמצאו מתאימים לתבנית בתוך ה DNS Cache.
* dns\_cache – מחרוזת השומרת את ה DNS Cache כפי שהוא נמצא ב CMD.
* url\_to\_check – מחרוזת השומרת קישור המצפין את ה IP הנבדק כך שיהיה אפשר להעלותו ל Virus Total.
* result – מילון בצורת JSON השומר את תוצאות ניתוח ה IP, ובתוכו מידע מפורט על כל המנועים שמצאו את הקובץ כחשוד. זאת ועוד, כמות המנועים שמצאו את ה IP כזדוני וכמות המנועים שלא מצאו אותו כזדוני.

PE Analysis:

* entropy – מספר השומר את האנטרופיה של הקובץ.
* pe – משתנה השומר את הקובץ כקובץ PE לצורך בדיקות וחקירה.

## 5.5 מסד נתונים

## 5.6 מבט מודולרי

## 5.7 פירוט מודלים עיקריים

# 6. תסריטי בדיקה

## 6.1 דגשים בבדיקה

## 6.2 תסריטי בדיקה עיקריים

# 7. רפלקציה

## 7.1 לוח זמנים מוערך לניהול הפרויקט

|  |  |
| --- | --- |
| נובמבר | POC מוכן הכולל LOG מפורט של ניתוח הקובץ (כולל חמשת התחומים המגדירים וירוס). ביצוע העברה מלאה ללא שגיאות של הקובץ החשוד לתוך הVirtual Machine וקבלת דיווח מפורט חזרה |
| דצמבר | חקירה סטטית הכוללת חקירת מבנה הPE המפורט של הקובץ + שלושת הקריטריונים לנוזקה מוחבאת בקובץ תמים + חקירת האנטרופיה של הקובץ + חקירת האנטרופיה של הPE Sections שלו. |
| ינואר | חתימות מסוג YARA לחקירת "אורזי קוד" + YARA strings |
| פברואר | בדיקה לפי Hash מלא של קובץ, ולפי Hash חלקי (נוזקה מוחבאת בתוך Exe תמים) באמצעות VirusTotal וחקירה מלאה לפי Fuzzy Hashing |
| מרץ | גרפיקה בסיסית וטיפול במקרי קצה (ניהול שגיאות) |
| אפריל | גרפיקה משופרת + שיפור והעמקת התרחישים והניתוחים + שפות חדשות ופונקציות חדשות |
| מאי | גרפיקה מרשימה עם כל התרחישים, מוצר מוגמר וספר מוגמר |

## 7.2 אתגרים ותרומה אישית

## 7.3 תוכנות

# 8. הוראות התקנה ותפעול

## 8.1 תצורה ודרישות קדם

לא פירטת דברים שהעברת מלמעלה.

## 8.2 התקנה

# 9. ביבלוגרפיה

במהלך כתיבת הפרויקט הסתמכתי על מספר מקורות מידע:

* <https://stackoverflow.com/>
* <https://cocomelonc.github.io/>
* <https://www.malwaretech.com/2015/01/inline-hooking-for-programmers-part-1.html>
* <https://www.ired.team/miscellaneous-reversing-forensics/windows-kernel-internals/pe-file-header-parser-in-c>++
* <https://ssdeep-project.github.io/ssdeep/index.html>
* <https://virusshare.com/>
* <https://www.gdatasoftware.com/blog/2022/09/37511-detecting-file-manipulation-in-system-files>

# 10. נספחים

הצעת הפרויקט :

* <https://docs.google.com/document/d/1q01tweYw2wol5npAbGg08fBfX_VKZFA3/edit?usp=sharing&ouid=103036410119617128946&rtpof=true&sd=true>