

לוטן יהב

MALWARE ANALYSIS AND DETECTION



לוטן יהב

326243920

גל בר און

30.5.2024

## MALWARE ANALYSIS AND DETECTION

4.....	הצעת פרוייקט שאושרה על ידי משרד החינוך.....
21.....	פרק 1 מבוא: .....
21.....	רקע פרוייקט: .....
22.....	סקירת ספרות.....
26.....	סיכונים ואתגרים בהכנת הפרוייקט .....
27.....	מטרות ויעדים .....
28.....	רקע תאורטי .....
28.....	ניתוח סטטי: .....
30.....	ניתוח דינאמי: .....
33.....	תיאור מצב קיים.....
34.....	ניתוח חלופות ומדוע בחרתי דווקא בחלופה הקיימת .....
34.....	ניתוח סטטי: .....
35.....	ניתוח דינאמי .....
36.....	בסיס נתונים.....
37.....	אפיון המערכת:.....
37.....	ניתוח דרישות המערכת: .....
40.....	מטרות המערכת: .....
40.....	מודלים במערכת: .....
41.....	אינטראקציה בין המודלים:.....
42.....	ארכיטקטורה.....
42.....	תיאור הארכיטקטורה: .....
43.....	הארכיטקטורה בצורת top down level design.....
45.....	אבטחת מידע:.....
46.....	ניתוח Use Cases.....
47.....	תרשים UML .....
48.....	רכיבי ממשק: .....
49.....	תיאור תוכנה: .....

MALWARE ANALYSIS AND DETECTION

50.....	תרשים מסכים.....
52.....	קוד.....
52.....	manager.py.....
56.....	Disinfection.py.....
58.....	HashManager.py.....
61.....	HybridManager.py.....
67.....	VirusTotal.py.....
72.....	YaraManager.py.....

## הצעת פרוייקט שאושרה על ידי משרד החינוך

### תיאור פרוייקט

אני עומד להכין תוכנת הגנה שתגן מפני התקפות של תוכנות ריגול שונות. בימים אלה, הנוכחות הגדולה של תוכנות ריגול בכל מקום מהווה איום משמעותי החודר בעדינות לאזורים הדיגיטליים שלנו. spywaren , שנועדה לאיסוף מידע אישי בחשאי, מסכנת את פרטיות המשתמשים במחשב בקנה מידה חסר תקדים. בעודנו גולשים באינטרנט בין אם בשביל עסקאות מקוונות ועד לאינטראקציות חברתיות, תוכנת הריגול מנצלת את טביעת הרגל הדיגיטלית שלנו, ויוצרת איום נרחב המגיע גם מעבר לתחומים אישיים ועשוי להשפיע על מוסדות פיננסיים, עסקים ואפילו כלכלות שלמות. כדי להפחית את הסיכון הזה, מודעות מוגברת ופיתוח טכנולוגיות מתוחכמות לזיהוי תוכנות ריגול הופכים חיוניים בשמירה על המידע הדיגיטלי שלנו ושמירה על שלמות העולם הדיגיטלי שלנו.

בנוסף לכך, תוכנות ריגול מהוות איום משמעותי לפרטיות של האדם בפרט. כאשר התוכנת ריגול נכנסת לתוך מחשב תמים שלא מודע לכך היא יכול לשבת ולהסוות את עצמה כתוכנה רגילה, אך בזמן זה היא תגנוב מידע אישי וחשוב כמו:

- כרטיסי אשראי
- סיסמאות
- אימיילים
- קבצים ותיקיות חשובות

מסיבה זו תוכנות רבות ובעיקר אנטי וירוסים עובדים על מנת למצוא ולהסיר את תוכנות הריגול האלה.

## הבעיה האלגוריתמית

וירוס מסוג תוכנת ריגול שהוא מאוד מפורסם בימינו הוא הסוס הטרויאני. דוגמה טובה לכך היא וירוס הידוע בשם FLAME, וירוס זה היה סוס טרויאני מתוחכם אשר שימש לריגול והתגלה לראשונה ב-2012 על ידי חברת האבטחה Kaspersky.

Flame היה וירוס חזק אשר אפשר להאקרים לא רק גניבת נתונים אלא גם הקלטת אודיו וצילומי מסך על המחשב המותקן. תוכנת הריגול הזו הופצה באמצעות שילוב בין ניצול חולשות של ווינדוס וניצול חיבורי דיסק און קי, שילוב זה אפשר לתוכנה לנצל עדכון של ווינדוס ולתקוף מחשבים כך. על מנת להגן מפני תקיפות כאלו אני עומד להכין אלגוריתם שידע לזהות תוכנות ריגול אלו ובהתאם לכך לנטרל אותם ולנקות אחריהם (לסדר דברים שנפגעו מהפעולות שלה).

אני אשתמש באלגוריתמים שונים, לארגזי חול אני אשתמש ב –

Cuckoo Sandbox, Firejail, Docker, Microsoft Azure Sandbox,  
Sandboxie, VirtualBox, QEMU

עבור ניתוחים דינאמיים אני אשתמש ב-

Wireshark, Process Monitor, ProcMon (Process Monitor),  
Sysinternals Suite, ApateDNS, INetSim, Capture-IPC  
PANDA (Platform for Architecture-Neutral Dynamic Analysis)

ועבור ניתוחים סטטיים אני אשתמש ב –

בנוסף אני אשלב את האלגוריתמים האלה בפרוייקט שלי בדרכים הבאות:  
ארגז חול, ניתוח סטטי וניתוח דינאמי.

## **1. כיצד הניתוח הדינאמי תורם לזיהוי תוכנות ריגול?**

מבין פתרונות הניתוח הדינאמי אני אשתמש בכמה פעולות מרכזיות:

### **תצפית התנהגות,**

אני עומד לתצפת על התוכנה כאשר פעולות התוכנה מנוטרות מקרוב בתוך סביבה מבוקרת (ארגז חול). גישה זו מאפשרת למערכות אבטחה שאני אבנה לבחון כיצד תוכנית מתנהגת במהלך הביצוע, כך אוכל לחשוף דפוסים ופעולות שעשויות להצביע על כוונת זדון. על ידי בידוד התוכנה במרחב מצומצם, אני אוכל להכיל איומים פוטנציאליים, ולמנוע השפעות שליליות על המערכת הכוללת. המעקב בזמן אמת של ריצת הקוד מספק לי תובנות לגבי התנהגות התוכנית, ומאפשר לי זיהוי של פעילויות חשודות כגון גישה לא מורשית, מניפולציה של נתונים או ניסיונות לנצל נקודות פגיעות.

### **בידוד הסביבה,**

הוא היבט מרכזי של ניתוח דינאמי אשר אני אשתמש בו לביצוע קוד שעלול להזיק בתוך סביבה מוגבלת ומבוקרת. על ידי כך, אני יכול לצפות ולנתח את התנהגות התוכנה מבלי לסכן את המערכת כולה. בידוד הסביבה מאפשר לי ביצוע בטוח של קוד שעלול להיות זדוני, ומאפשר לי בדיקה מפורטת של

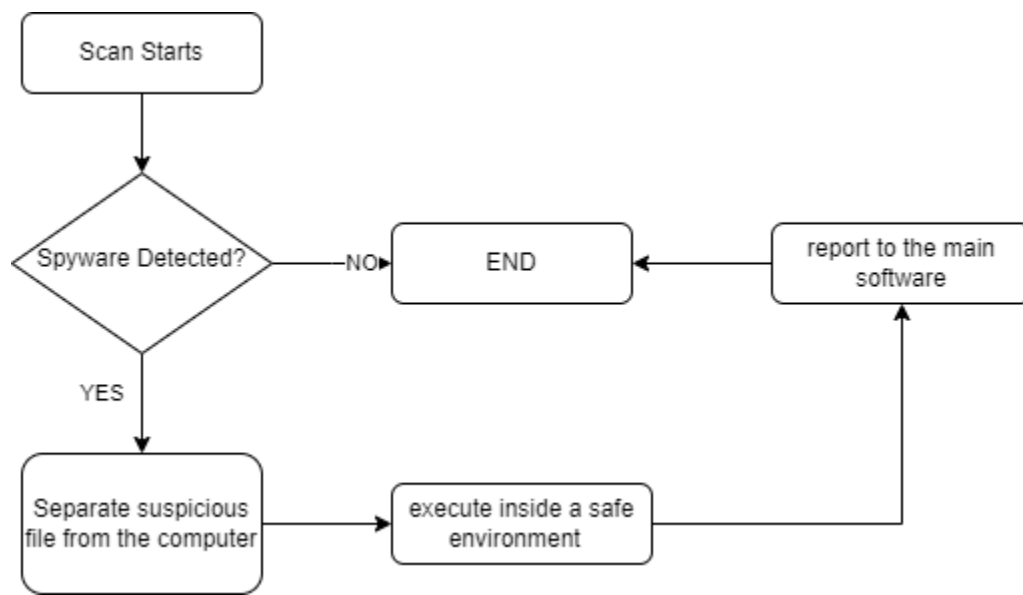
## MALWARE ANALYSIS AND DETECTION

פעולותיו. הגדרה מבוקרת זו מקלה עליו בתהליך זיהוי פעילויות של תוכנות ריגול ומאפשרת לי לזהות פעולות כגון ניסיונות לחדור למערכת, לשנות קבצים קריטיים או ליצור חיבורי רשת לא מורשים בצורה פשוטה יותר ובטוחה יותר. בידוד סביבת העבודה שלי תורמת להבנה טובה יותר של הטקטיקות של תוכנת הריגול, ומסייעת לי בפיתוח של אמצעי נגד יעילים.

שני הפתרונות של ניתוח דינאמי לבעיית תוכנות הריגול שהצגתי עד כה מובילות לעוד פתרון יותר מרכזי שגם הוא נמצא תחת ניתוח דינאמי והוא נקרא Sandbox או ארגז חול בעברית.

## פתרון ארגז חול – Sandbox:

אני אשתמש בארגז חול כפתרון דינמי לזיהוי תוכנות ריגול על ידי יצירת סביבה מאובטחת ומבודדת שבה אני אפעיל ואנתח את התוכניות חשודות. הגדרה מבוקרת זו מאפשרת למערכות האבטחה שאני אכין לצפות בהתנהגות של קוד שעלול להיות זדוני מבלי לסכן את המערכת הכוללת. במהלך הביצוע בתוך ארגז החול, אני אוכל לזהות ולנתח פעולות המעידות על תוכנות ריגול, כגון גישה לא מורשית, מניפולציה של נתונים או ניסיונות לנצל נקודות תורפה של מערכת ההפעלה. על ידי מתן מרחב בטוח לבדיקה



## MALWARE ANALYSIS AND DETECTION

מעמיקה, אני אוכל להשתמש בארגז חול כדי לסייע בזיהוי והבנת פעילויות של תוכנות ריגול וכך לתרום לתובנות חיוניות לפיתוח אמצעי נגד יעילים לתוכנות ריגול.

ארגז החול הוא מערכת חשובה מאוד בתהליך הזיהוי של תוכנות ריגול ובנוסף יש לארגז חול שימוש רב בתוכנות של ניתוח דינאמי לכן החלטתי לכלול אותו תחת הניתוח הדינאמי.

## 2. כיצד הניתוח הסטטי תורם לזיהוי תוכנות ריגול?

ניתוח סטטי כולל בחינת קבצים מבלי לבצע אותם, תוך הסתמכות על דפוסים, חתימות וכללי התנהגות מוגדרים מראש כדי לזהות ולזהות תוכנות ריגול פוטנציאליות על סמך מאפיינים והתנהגויות ידועים. שיטה זו שמה לה למטרה לזהות איומים לפני שהם יכולים לצאת לפועל ולגרום נזק למערכת.

### זיהוי מבוסס חתימות:

בדרך זו אשתמש בניתוח סטטי כדי למנף זיהוי מבוסס חתימות על ידי השוואת קבצים למסד נתונים של חתימות ידועות של תוכנות ריגול. אני אשתמש בו כדי ליצור מזהים ייחודיים עבור דפוסי ריגול ידועים. כאשר קובץ תואם לאחת מהחתימות הללו, אני אסמן אותו כקובץ שעלול להיות זדוני. שיטה זו יעילה לזיהוי גרסאות ידועות של תוכנות ריגול, ובעזרתה אני יכול בקלות לזהות תוכנות ריגול חלשות מרמה נמוכה אך כאשר אני אגיע אל תוכנת ריגול חדשה שלא נראתה לפני או שמסתירה טוב את הדפוס שלה השיטה הזו נכנסת לבעיה ואצטרך לעבור לשיטה אחרת.



### **זיהוי דפוסים:**

אני אשתמש בניתוח סטטי כדי לבחון את המבנה והתוכן של קבצים ובכך לזהות דפוסים או התנהגויות חשודות הקשורות לתוכנות ריגול. זה כולל ניתוח מבני קוד, חיפוש אחר רצפים ספציפיים או פקודות נפוצות בתוכנות ריגול. היכולת לזהות דפוסים אלו מסייעת לי בזיהוי מוקדם של התוכנה הזדונית לפני הפעלתה.

### **הגדרות כללים התנהגותיים:**

בנוסף אני אשתמש בניתוח סטטי כדי לקבוע חוקי התנהגות המבוססים על הפעולות הצפויות של תוכנות ריגול. כללים אלה הם תנאים מוגדרים מראש אשר משמשים אותי כדי לזהות תכונות חשודות בתוכנה ואם הם יופעלו, הם מעלים התראה. לדוגמה, אם קובץ מנסה גישה לא מורשית או מפגין התנהגות קוד חריגה, אני אשתמש בניתוח הסטטי כדי לסמן אותו בתור פוטנציאל זדוני. גישה זו מאפשרת לי זיהוי של איומי ריגול חדשים או מתפתחים בהתבסס על התנהגותם הצפויה לפני שהם יכולים להפעיל את הפעולות הזדוניות שלהם.

## **מבנה הפרוייקט**

אני אחלק את הפרוייקט שלי ל-3 חלקים, חלק ראשון שיעבוד בצד השרת הוא יזהה את תוכנות הריגול ינתח אותם ימחק אותם ויסדר בנוסף החלק הראשון שהוא צד השרת יהיה אחראי על כל מה שקורה ושני החלקים האחרים יפעלו דרכו, לצד השרת תהיה שליטה על הפרוייקט. החלק השני יעבוד עם GUI והוא יכלול בתוכו את צד הלקוח, צד הלקוח יציג בפני המשתמש את הפעולות הקיימות בצד השרת ואת הפעולות שהוא יכול להפעיל על מנת לחפש ולמחוק תוכנות ריגול. החלק השלישי הוא בסיס נתונים שישמור בתוכו פעולות שנעשו על ידי החלק הראשון הבסיס נתונים יכיל את השם והמידע על תוכנות ריגול שנתפסו בעבר ונמחקו מהמחשב ובנוסף על כל תוכנת ריגול שנמחקה ישמרו שמות הקבצים שנפגעו במידה ונפגעו קבצים.

## חלק 1 – צד שרת

צד השרת בעצם ינהל את כל הפרוייקט מבפנים.

צד השרת יהווה כגשר בין הבסיס נתונים וצד הלקוח.

### 1. סריקה:

פעולת הסריקה היא פעולה מרכזית וחשובה ביותר, הפעולה הזאת תרוץ כל פרק זמן קבוע על מנת לשמור הבטיחות של המחשב בנוסף בצד הלקוח יהיה למשתמש אופציה להריץ סריקה בכל זמן שבוא הוא ירגיש שהמצב של המחשב לא טוב.

פעולת הסריקה תשתמש בשני מנגנונים שונים על מנת לזהות וירוסים והם ClamAV ו YARA התוכנית תבנה בצורה שבה יהיה שימוש בשני על מנת לספק לפחות 2 דרכים שונות לזהות וירוסים, בנוסף הפעולה תשתמש בניתוח היוריסטי על מנת שתהיה אמינה יותר.

YARA לביצוע ניתוח סטטי, YARA הוא כלי רב עוצמה להתאמת דפוס התנהגות של קבצים שיכול לסרוק קבצים או תהליכים ולאתר מאפיינים ספציפיים המעידים על תוכנות ריגול. YARA מאפשר לי להגדיר כללים מותאמים אישית כדי לזהות דפוסים חשודים בקבצים בינאריים או בזיכרון.

ClamAV לביצוע ניתוח דינאמי, ClamAV פועל על פי זיהוי חתימות באמצעות חתימות ידועות של תוכנות ריגול. כלים כמו ClamAV מציעים בסיס נתונים מוכן מראש של חתימות של תוכנות זדוניות ידועות. אני יכול לשלב את ClamAV במודול הזיהוי שלך כדי לסרוק קבצים לאיתור התאמות מול חתימות אלה.

Cuckoo sandbox הוא כלי קוד פתוח המתמחה בניתוח תוכנות זדוניות אוטומטיות, Cuckoo גם הוא משתמש בניתוח דינאמי. Cuckoo מריץ קבצים חשודים בסביבה מבוקרת, לוכד את התנהגותם, שיחות המערכת ופעילות הרשת שלהם. פעולות משלימות את הטכניקות אחרות, כגון זיהוי מבוסס חתימה באמצעות ClamAV, ניתוח היוריסטי ו-YARA עבור חוקי התאמת דפוסים, אשר בהם אני משתמש.

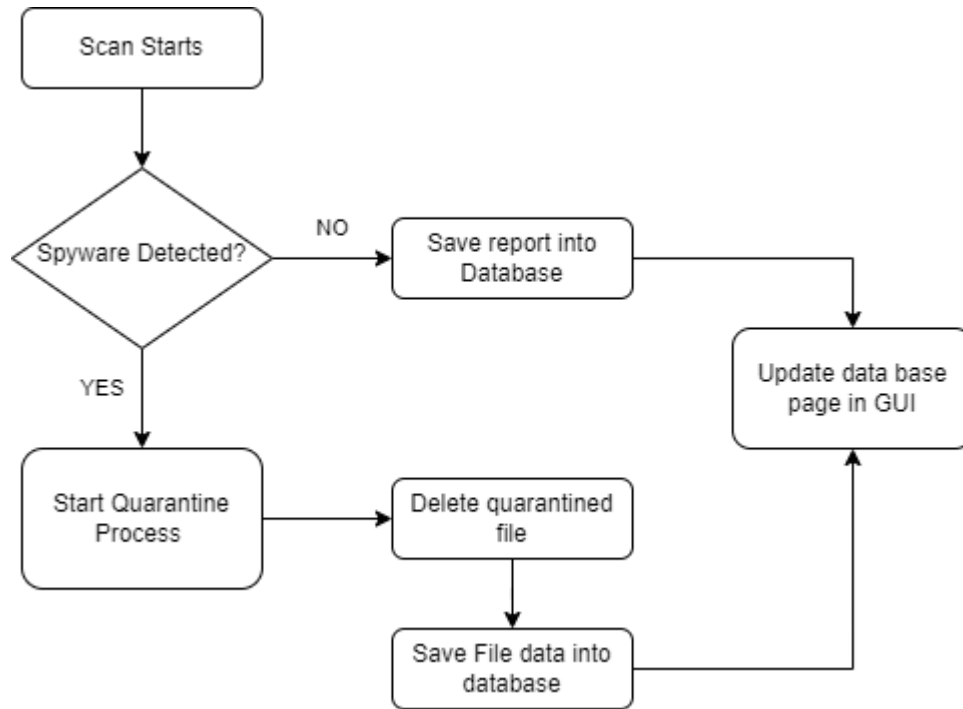
## 2. הסרה:

פעולת ההסרה היא פעולה נוספת שחשובה מאוד להצלחת הפרוייקט והתוכנית כולה. הפעולה הזאת תרוץ אך ורק כאשר תימצא תוכנת ריגול על המחשב, כלומר יהיה חיבור ישיר בין פעולה זו לפעולת הסריקה. המערכת תפעל בהתאמה ביחד עם המערכות שמזהות את תוכנת הריגול על מנת ליצור מערכת היוזמת מנגנון הסגר ומחיקה. זה כרוך בבידוד תוכנת הריגול שזוהתה על ידי העברתה לספריית הסגר ייעודית, ובכך להגביל למעשה את יכולתה לבצע או לגרום נזק נוסף. הקוד יכול להשתמש בפקודות מספריית shutil, כגון shutil.move(), על מנת להעביר את הקבצים החשודים לתיקיית ההסגר. לאחר מכן, המערכת יכולה להמשיך למחוק או לנטרל קבצים אלה, תוך הבטחת הבלימה וביטול של איומים פוטנציאליים. ניטור ותחזוקה שוטפים של ספריית ההסגר תורמים ליעילות הכוללת של מערכת הסרת תוכנות הריגול. לאחר הסרה של תוכנה זדונית ישמר בתוך בסיס הנתונים שם התכונה התאריך בה היא נמחקה ובמידה ויש קבצים שנפגעו ממנה שמם ישמר גם

### **3. אינטגרציה**

סקריפט האינטגרציה מתפקד כמתאם הליבה במערכת האיתור וההסרה של תוכנות ריגול, ומקשר בין אלמנטים מגוונים בצורה חלקה. הוא משלב אלגוריתמי זיהוי כמו YARA ו-ClamAV לניתוח קבצים ותהליכים נרחבים כדי לאתר תוכנות ריגול פוטנציאליות. הסקריפט יוזם ניתוח התנהגותי, זיהוי מבוסס חתימה, ומשתמש בארגז החול של Cuckoo כדי לבחון קבצים ותהליכים לאיתור התנהגות חשודה. לאחר זיהוי תוכנות ריגול, סקריפט האינטגרציה מפעיל את תהליך ההסרה, תוך הפעלת מנגנוני ההסגר והמחיקה. הוא גם מתקשר עם המודולים בצד השרת, ומבטיח תגובה מגובשת ומסונכרנת לאיומים פוטנציאליים. ריכוזיות זו משפרת את יעילות המערכת, ומאפשרת עדכונים בזמן אמת, ניטור וכוונון עדין של אסטרטגיות זיהוי והסרה של תוכנות ריגול.

## MALWARE ANALYSIS AND DETECTION



## חלק 2 – צד הלקוח

בצד זה אני אבנה GUI שדרכו יהיה למשתמש גישה לראות את מה שבעצם קורה בתוכנה. אני אשתמש בספריות כמו Tkinter ו pyQT כדי ליצור תפריט מודרני שנראה טוב.

צד זה יהיה מחולק לחלקים שבהם המשתמש יוכל לעבור ויהיה מסודר בצורה נוחה לעין:

שם העמודה	לוגו
בית	
סריקה	
בסיס נתונים	
הגדרות	

בצורה זו הגישה עבור המשתמש תהיה פשוטה ונוחה.

צד הלקוח יכלול:

### **בית:**

בעמוד זה יוכל המשתמש לראות את המצב הביטחוני של המחשב ואת תוצאות הסריקה האחרונה.

### **סריקה:**

בעמוד זה יהיה למשתמש אופציה לבחור לעשות סריקה מיידית ואיזה סוג סריקה, פעולה זו תפעל בשילוב עם פעולת הסריקה בצד השרת ותסרוק כמות קבצים בהתאם לבקשת המשתמש. הפעולה תסרוק את כל המחשב עבור סריקה מלאה ותסרוק קבצים חשובים כמו קבצי מערכת עבור סריקה מהירה.

### **בסיס נתונים:**

בעמוד זה יוצג עבור המשתמש בסיס הנתונים שלו, בסיס הנתונים יהיה בנוי ללא שרת ולכן יהיה נוח לגישה ולהצגה מתוך הפרוייקט ויעשה שימוש בSQLite. עמוד זה יציג בתוכו את כל תוכן בסיס הנתונים בצורת GUI נוחה לעין המשתמש.

### **הגדרות:**

עמוד זה יציג את ההגדרות הבסיסיות:

MALWARE ANALYSIS AND DETECTION

1. סריקה כל \_\_\_\_ שעות – הגדרה זו תיתן למשתמש לבחור כל כמה שעות תתבצע סריקה קבוע (יכול להיות שאני אשנה את זה ל, באיזה שעה ביום תתבצע הסריקה האוטומטית)
2. רקע – אופציה זו תאפשר למשתמש לבחור בין עיצובים שונים כגון : רגיל, כהה, בהיר וכו'. אולי יתווספו עוד בחירות בהמשך.
3. הגדרות נכות – אופציה להפעיל פעולות ספציפיות על מנת להקל שימוש עבור אנשים עם מוגבלות, דברים כמו הגדלת הכתב והקראה קולית.



### **חלק 3 – בסיס נתונים**

בחלק זה אני אשתמש בSQLite על מנת ליצור בסיס נתונים שישמור בתוכו מידע מהסריקות. הפעולה תרוץ בצורה אוטומטית אחרי כל סריקה

עבור כל סריקה שתרוץ על המחשב יוכנס לבסיס הנתונים מידע עבור:

תאריך הסריקה

שעת הסריקה

האם נמצא וירוס?

שם הוירוס

התיקיה בה נמצא

סוג הוירוס

נמחק?

שעת המחיקה

MALWARE ANALYSIS AND DETECTION

מס סריקה	תאריך	שעה	וירוס	שם הוירוס	PATH	סוג הוירוס	שעת המחיקה
1	10.1.20 24	8:55:31	TR UE	virus_na me	C:\PA TH	Spywa re	8:57:25
2	10.1.20 25	9:55:31	TR UE	virus_na me	C:\PA TH	Spywa re	9:57:25
3	10.1.20 26	10:55:3 1	TR UE	virus_na me	C:\PA TH	Spywa re	10:57:25
4	10.1.20 27	11:55:3 1	TR UE	virus_na me	C:\PA TH	Spywa re	11:57:25
5	10.1.20 28	12:55:3 1	TR UE	virus_na me	C:\PA TH	Spywa re	12:57:25
6	10.1.20 29	13:55:3 1	TR UE	virus_na me	C:\PA TH	Spywa re	13:57:25

וכו'.

ביבליוגרפיה:

<https://www.techtarget.com/searchsecurity/definition/sandbox#:~:text=A%20sandbox%20is%20an%20isolated,to%20test%20potentially%20malicious%20software.>

<https://www.perforce.com/blog/sca/what-static-analysis#:~:text=Static%20analysis%20is%20a%20method,compliant%2C%20safe%2C%20and%20secure.>

<https://www.parasoft.com/blog/static-analysis-and-dynamic-analysis/>

<https://www.w3schools.com>

<https://wiki.python.org/moin/GuiProgramming>

10.01.2024	כתיבת הצעת פרויקט
10.01.2024	שליחת הצעת פרויקט
31.01.2024	פיתוח צד שרת חלק 1
15.02.2024	פיתוח צד שרת חלק 2
28.02.2024	פיתוח צד שרת חלק 3 + בסיס נתונים
10.3.2024	בניית צד לקוח בעל GUI וחיבור כל החלקים
20.03.2024	בדיקת תקינות המערכת
01.04.2024	הכנת המערכת לבדיקה

## פרק 1 מבוא:

### רקע פרוייקט:

בזמן ובתקופה שבא אנחנו חיים אנשים רבים משתמשים בכלים טכנולוגיים שונים בין אם זה מחשב נייד, מחשב נייד, סמארטפון או קונסולת משחקים אנחנו יוצרים חשבונות רבים וגולשים זמן רב ברשתות אונליין.

החשיפה הממושכת של אנשים רבים לאינטרנט חושפת אותנו לסוגים רבים של תוכנות זדוניות מסוגים שונים, תוכנות זדוניות אלו יכולות לעשות דברים רבים בין אם זה להאט את הפעולה של המחשב ועד לגנוב פרטי אשראי מאתרים בהם קנית בעבר.

אני שמטבעי מתעניין בכל הנושא של אבטחת מידע וביטחון סייבר החלטתי להכין בפייתון תוכנה נגישה למשתמשים אלה אשר מבליים את רוב זמנם באינטרנט. תוכנה זו תגן עליהם מפני הסכנות שקיימות באינטרנט בהתמקדות על תוכנות ריגול ועל ידי שימוש בסריקות סטטיות ודינאמיות על מנת לגלות ולהיפטר מתוכנות זדוניות.

הפעולות יתבצעו בצורה הבאה, המשתמש יכניס מיקום של תיקייה (path) והתוכנה תבצע פעולה של סריקה אשר תעבור על הקובץ או על הספרייה במידה ומדובר באחת.

פעולת הסריקה תתבצע על ידי ארבעה כלים שונים לניתוח הקובץ שני כלים של ניתוח סטטי ושני כלים של ניתוח דינאמי אשר השימוש בכולם ייכתב בשפת python.

התוכנה תשתמש בהאשים וב yara על מנת לזהות קבצים זדוניים על ידי השוואה מול דוגמאות קיימות, האשים של וירוסים מפורסמים שיילקחו מ virusshare והשוואת חתימות של תוכנות זדוניות קיימות על ידי חוקים של yara.

בנוסף התוכנה תשתמש ב api 2 של sandbox קיימים אשר יאפשרו העלאה של קובץ לארגז החול ניתוח וקבלת התוצאות ישירות אל המשתמש. בחלק זה יעשה שימוש ב וירוס טוטל אשר מאפשר שליחת קבצים לארגזי חול של חברות אבטחה קיימות וקבלת התשובה, וגם בהיבריד אנליסיס אשר מאפשר ניתוח קבצים בארגז חול פתוח לשימוש חופשי. התוצאות של הסריקה ישמרו בבסיס נתונים של mongodb.

## סקירת ספרות

**Python** - פייתון היא שפת תכנות ברמה גבוהה, אך קלה להבנה, אשר זכתה לפופולריות רבה בזכות הפשטות והגמישות שלה. פייתון נוצרה על ידי ג'ידו ואן רוסום ושחררה לראשונה בשנת 1991. הפילוסופיה של עיצוב פייתון מדגישה קריאות של קוד, מה שמאפשר למפתחים להביע רעיונות בפחות שורות קוד בהשוואה לשפות כמו C++ או Java.

פייתון היא שפה מתקדמת מאוד ומעוצבת בצורה שנוח מאוד להשתמש בה. השפה קלה מאוד לקריאה, וזאת מכיוון שיש בה פקודות רבות שמבצעות פעולות מסובכות בשורה או שתיים, דבר שלא קיים בשפות אחרות כמו Java או C++ שבהן יש צורך ליצור קלאסים לכל דבר. המינימליזם הזה מפחית את זמן הפיתוח ומקל על תחזוקת הקוד.

**Visual Studio Code (VS Code)** - הוא פלטפורמה וסביבת פיתוח עוצמתית לעריכה וקתיבה של קוד תוכנה. סביבת הפיתוח של VS Code נוצרה על ידי חברת Microsoft והיא זמינה בחינם כתוכנה מסוג קוד פתוח. הייחודיות של סביבת ה VS Code נובעת מהאפשרות של לאנשים לגשת לראות, לשנות ולשפר את הקוד הבסיסי של התוכנה, מה שמוביל לפיתוח מתמשך ושיפור תמידי על ידי קהילת המפתחים הגלובלית.

VS Code תומכת בכתיבת קוד בשפות תכנות רבות ומגוונות, כולל פייתון, C++, Java, JavaScript, TypeScript ועוד רבות. התמיכה הרחבה בשפות התכנות השונות מאפשרת למפתחים מכל התחומים להשתמש בסביבה זו לפרויקטים מגוונים, בין אם הם עוסקים בפיתוח אתרים, יישומי מובייל, משחקים, ניתוח נתונים או פיתוח מערכות תוכנה מורכבות.

**YARA – YARA** היא כלי רב עוצמה המשמש בעיקר לזיהוי ואיתור תוכנות זדוניות

באמצעות יצירת חוקים מותאמים אישית. יישום זה, שפותח על ידי וירוס טוטאל (VirusTotal) נועד לסייע בחקירות של אבטחת מידע, ומאפשר לחוקרי אבטחה לתאר דפוסי התנהגות של תוכנות זדוניות ולזהות אותן על בסיס דפוסים אלו.

YARA מאפשרת למשתמשים לכתוב חוקים בשפה פשוטה וקריאה, המתארים מאפיינים מסוימים של קבצים או דגימות. החוקים יכולים לכלול רצפי מחרוזות, חתימות בינאריות, או כל מאפיין אחר שניתן להגדיר בקובץ. ברגע שהחוק נכתב, YARA יכולה לסרוק קבצים, תיקיות, או זיכרון מערכת ולהשוות את התוכן שלהם לחוקים שנכתבו, מה שמקל על זיהוי איומים חדשים ולא מוכרים.

דוגמה לחוק כתוב ב-YARA:

```
rule TestRule {
  strings:
    $malicious = "malicious"

  condition:
    any of them
}
```

הקוד הנל כאשר יורץ על קובץ ינתח אותו ויחפש את הסטרינג malicious ואם אחד מהסטרינגים קיימים בקובץ הוא יחזיר True

**פונקציות Hash** - הם כלים קריפטוגרפיים חיוניים בתחום האבטחה המידע, המאפשרים המרת נתונים מכל גודל לערך קבוע בגודל שנקרא Hash. הערך הזה משמש בעיקר לבדיקת שלמות נתונים וזיהוי, והן חלק בלתי נפרד ממערכות אבטחת מידע רבות.

**SHA-256 (Secure Hash Algorithm 256-bit)** - הוא אחד מהאלגוריתמים ממשפחת SHA-2, שפותחה על ידי הסוכנות לביטחון לאומי של ארצות הברית (NSA). הוא משמש ליצירת ערכי Hash בגודל 256 ביט (32 בתים). SHA-256 נחשב לבטוח ויעיל לשימושים רבים, כולל חתימות דיגיטליות, תעודות אבטחה, ובדיקה של שלמות קבצים.

**MD5 (Message Digest Algorithm 5)** - הוא אלגוריתם Hash שנוצר על ידי רונלד ריבסט בשנת 1991. הוא מייצר ערכי Hash בגודל 128 ביט (16 בתים). למרות שבעבר היה פופולרי מאוד, הוא נחשב כיום לפחות בטוח לשימושים קריפטוגרפיים בשל חולשות שהתגלו בו.

אני עשיתי שימוש בהאשים על ידי לקיחת האשים ידועים של וירוסים קיימים והשוואתם עם ההאשים של הקבצים שאני רוצה לנתח. דוגמאות של האשים:

md5 – 'hello world' יהפוך ל - fc3ff98e8c6a0d3087d515c0473f8677

sha256 – 'hello world' יהפוך ל -  
c0535e4be2b79ffd93291305436bf889314e4a3faec05ecffcb7df31b93d8d  
b

לא משנה כמה פעמים אריץ את הטקסט בהאשים האלו תמיד אקבל את אותה תשובה



## MALWARE ANALYSIS AND DETECTION

**HybridAnalysis Sandbox API** - הוא כלי עוצמתי וחיוני המשמש חוקרי אבטחה לצורך ניתוח דינמי של קבצים חשודים. פלטפורמה זו מאפשרת למשתמשים להעלות קבצים לסביבה מבודדת (sandbox), שבה ניתן לבצע את הקבצים בצורה בטוחה ולנתח את ההתנהגות שלהם. באמצעות HybridAnalysis, ניתן לזהות תוכנות זדוניות וללמוד על התנהגותן מבלי לסכן את מערכת ההפעלה הראשית.

**VirusTotal API** - הוא שירות חיוני נוסף בתחום אבטחת המידע, המספק לחוקרי אבטחה ולמפתחים את היכולת לסרוק קבצים וכתובות URL כנגד מספר רב של מנועי אנטי-וירוס ושירותי סריקה מקוונים. השירות פותח על ידי חברת VirusTotal ומאפשר למשתמשים לבדוק במהירות וביעילות אם קובץ או כתובת URL חשודים כנגועים בתוכנות זדוניות.

באמצעות VirusTotal API, ניתן להעלות קבצים לבדיקה, לסרוק כתובות URL, ולשאול מידע על קבצים שכבר נבדקו בעבר. ה-API מחזיר דוחות מפורטים הכוללים תוצאות סריקה ממנועי אנטי-וירוס רבים, מידע על הקובץ, ומידע על התנהגות הקובץ במידה וישנם ניתוחים דינמיים זמינים.

**MongoDB - MongoDB** הוא מסד נתונים NoSQL רב עוצמה שמיועד לאחסון וניהול של כמויות גדולות של נתונים במבנה גמיש וניתן להרחבה. פלטפורמה זו, שנוצרה על ידי חברת MongoDB Inc, מציעה פתרון מתקדם לניהול נתונים שמתאים במיוחד לעבודה עם נתונים לא מובנים או משתנים, מה שהופך אותה לכלי חשוב בארגז הכלים של מפתחים ומנהלי נתונים.

## סיכונים ואתגרים בהכנת הפרוייקט

במהלך עבודתי על הפרוייקט התייצבו לפני מס בעיות וקשיים מכיוון שהגעתי אל הנושא ללא יידע מוקדם, בהצעת הפרוייקט רשמתי שאשתמש בשרת ולקוח על מנת לחבר את קבצי הניתוח אל קבצי התצוגה שאיתם מתעסק במשתמש אך לאחר חשיבה הגעתי למסקנה ששימוש בשרת ולקוח על מנת לחבר את שני החלקים שנמצאים על אותו המחשב ייעקב את מעבר המידע וייסכן כניסה של זדון חיצוני למערכת. בנוסף לכך נתקלתי בבעיה עם הרצת כלים לניתוח דינאמי של קבצים, מכיוון שהחומר היה חדש בשבילי ובנוסף הוא חומר שקשוח ללמידה היה לי קשה ללמוד את הנושא והגעתי למסקנה שהכלים הנוחים ביותר לניתוח דינאמי אוטומטי של קבצים הם API לשימוש חופשי של חברות אבטחה ואתרי אבטחה קיימים כגון וירוס טוטל והייבריד אנליסיס אשר מציעים שירותים שונים לניתוחים דינאמיים של קבצים על ידי הרצה בארגז חול על השרת שלהם.

בנוסף הייתי צריך ללמוד איך להשתמש ולעצב GUI בעזרת PyQT משהו שלא התעסקתי בו לפני ולקח ממני הרבה זמן של עבודה לחבר בין כל החלקים והתצוגה עצמה

## מטרות ויעדים

מטרת הפרוייקט היא יצירת תוכנה שתאפשר למשתמש לסרוק ולנתח קבצים שהוא מוצא כחשודים ולקבל תשובה מהירה על האם הם באמת מסוכנים או לא. התוכנה תשלב ניתוחים סטטיים ודינמיים ותשמור את תוצאות הסריקות בבסיס נתונים נוח לגישה. מטרת הפרוייקט כוללות:

מטרות הניתוחים הסטטיים:

1. בדיקה של קבצים על ידי הרצתם דרך פונקציות האש שונות
2. השוואת ההאשים של הקבצים להאשים של וירוסים ידועים
3. בדיקת התוכן הבינארי של הקוד על ידי שימוש בYARA
4. בדיקת חותמות וחוקים של וירוסים קיימים.

מטרות הניתוחים הדינמיים:

1. הרצת קבצים בסביבת עבודה מבודדת משאר המחשב
2. מעקב אחר הפעולות של הקובץ ומציאה האם הוא באמת זדוני או לא

מטרות בסיס הנתונים:

1. לשמור בתוכו את תוצאות הסריקות על מנת לאפשר גישה נוח למשתמש.

התוכנה שתפותח תהיה כלי מקיף ואמין לזיהוי וניתוח קבצים חשודים, עם ממשק משתמש ידידותי ונוח לשימוש, המיועדת הן למשתמשים פרטיים והן לארגונים.

## רקע תאורטי

### ניתוח סטטי:

ניתוח סטטי של קבצים הוא שיטה לאיתור ופענוח תוכן של תוכנות מחשב מבלי להפעיל אותן בפועל. שיטה זו נועדה לבדוק את הקוד או הנתונים של הקבצים באופן אנליטי כדי לזהות מאפיינים חשודים או זדוניים. בניתוח סטטי נעשה שימוש רב באבטחת מידע לצורך זיהוי תוכנות זדוניות (malware) ולמניעת התקפות סייבר.

### עקרונות הניתוח הסטטי

פונקציות האש (Hash Functions): פונקציות האש הן כלים קריפטוגרפיים היוצרים מחרוזת ייחודית עבור כל קובץ, המבוססת על התוכן שלו. תוצאת פונקציית האש, המכונה hash, משמשת לזיהוי קבצים בצורה מהירה ומדויקת. בניתוח סטטי, השוואת ההאש של קבצים להאשים ידועים מאפשרת לזהות תוכנות זדוניות מוכרות.

חתימות דיגיטליות (Digital Signatures): חתימות דיגיטליות הן שיטה להבטחת שלמות האותנטיות של קבצים. הן מבוססות על הצפנה אסימטרית ומאפשרות לבדוק אם קובץ עבר שינויים מאז נחתם. בניתוח סטטי, ניתן להשתמש בחתימות כדי לזהות אם קובץ עבר שינוי זדוני או אם מקורו אמין.

כלי YARA: YARA הוא כלי רב-עוצמה שנועד לזהות ולתאר משפחות שונות של תוכנות זדוניות על ידי יצירת חוקים מותאמים אישית. חוקים אלו מבוססים על מחרוזות ודפוסים שניתן למצוא בתוכן הבינארי של הקבצים. ניתוח סטטי בעזרת YARA מאפשר לזהות תוכנות זדוניות על סמך מאפיינים ידועים של הקוד.

## MALWARE ANALYSIS AND DETECTION

בדיקת תוכן בינארי: ניתוח התוכן הבינארי של קבצים מאפשר לזהות חתימות ודפוסים שמאפיינים תוכנות זדוניות. בדיקות אלו כוללות חיפוש של רצפי מחרוזות חשודות, ניתוח מבנה הקובץ וזיהוי קטעי קוד זדוניים. כלים לניתוח בינארי מסייעים לחשוף את התנהגות הקובץ מבלי להפעילו.

### **יתרונות הניתוח הסטטי**

מהירות ויעילות: ניתוח סטטי מאפשר לזהות תוכנות זדוניות במהירות רבה מבלי להפעיל את הקבצים בפועל. הדבר חשוב במיוחד בסביבות בהן יש צורך לבדוק כמות גדולה של קבצים בזמן קצר.

הימנעות מסיכון: מאחר והקבצים אינם מופעלים, הניתוח הסטטי אינו מסכן את מערכת ההפעלה או את הסביבה שבה מתבצע הניתוח. כך ניתן לבדוק קבצים חשודים בצורה בטוחה.

זיהוי מוקדם: ניתוח סטטי מאפשר לזהות קבצים זדוניים עוד לפני שהם מבוצעים, מה שמפחית את הסיכון להדבקה ואת הפגיעה במערכת

## **ניתוח דינאמי:**

ניתוח דינמי של קבצים הוא שיטה לבדיקת התנהגות ופעולתם של תוכניות מחשב תוך הרצתן בסביבה מבודדת או במערכת פעילה. השיטה מקבלת קבצים כקלט, מפעילה אותם בתוך סביבת מחשב מבודדת ומבצעת מעקב אחר פעולותיהם ושימושיהם במערכת ההפעלה.

### **עקרונות הניתוח הדינמי**

הרצת קבצים בסביבת מבודדת: ניתוח דינמי מתבצע על ידי הרצת הקבצים בתוך סביבת מחשב מבודדת אשר מופעלת בכל פעם לצורך בדיקת התוכניות. זה מקל על הביצוע והניתוח של קבצים פוטנציאלית מסוכנים בלי לסכן את המערכת הפעילה.

מעקב אחר הפעולות של הקובץ: ניתוח דינמי כולל מעקב מפורט אחר כל הפעולות שהקובץ מבצע במערכת ההפעלה, כולל יצירת קבצים חדשים, שינויי רישום ותקשורת עם שרתים חיצוניים. זה מאפשר לזהות התנהגות זדונית או לא רגילה.

ניתוח תגובה ותגובה: ניתוח דינמי כולל אף את ניתוח התגובה של המערכת לפעולות שהקובץ ביצע, כולל השוואה בין התוצאות הצפויות לבין התוצאות האמיתיות. זה מאפשר לזהות בצורה יעילה איומים חדשים או מתקפות מתפתחות.

### **יתרונות הניתוח הדינמי**

זיהוי התנהגות זדונית: הניתוח הדינמי מאפשר זיהוי מתקפות סייבר על פי התנהגות ופעולות של התוכניות במערכת ההפעלה, מה שיכול לכלול פעולות דפוסים, גישה למידע רגיש ועוד.

תרחישי התקפה במציאות: ניתוח דינמי מאפשר דימוי של תרחישי התקפה אמיתיים וזיהוי נקודות חולשה במערכת הממוקמת במציאות הארגונית.

## YARA

YARA הוא כלי חזק לניהול ולהתאמה אישית של חוקים לזיהוי דפוסים של תוכנות זדוניות. על ידי שימוש בפייתון אני כתבתי חוקים ממחרוזות ותנאים המאפשרים לזהות קבצים חשודים. YARA מאפשר לבדוק קבצים, תהליכים וספריות שלמות בהתאם לחוקים שנכתבים, מה שמסייע בזיהוי והבנת תוכנות זדוניות על סמך תכונות והיבטים ספציפיים של התוכן שלהן.

## Hash

אני הפכתי קובץ ל Hash באמצעות ספריית hashlib, ספרייה זו היא ספרייה המאפשרת תהליך של יצירת מחרוזת ייחודית המייצגת את תוכן הקובץ. על ידי קריאת הקובץ במצב בינארי והזנת התוכן לאובייקט ניתן לייצר hash שמתאים לקובץ בצורה ייחודית. תהליך זה מאפשר לי להשוות Hash של קובץ קיים ולהשוות אותו ל HASH של קבוצה זדוני מרשימת האשים שהכנתי מראש.

## Hybrid Analysis

Hybrid API מספק ממשק לשליחת נתונים וניתוח קבצים באמצעות HybridAnalysis Sandbox השימוש ב API-כולל העלאת קבצים חשודים לסביבה מבודדת (sandbox) באמצעות בקשות HTTP (requests) והמתנה לניתוח ההתנהגות של הקובץ. השירות מחזיר דוחות מפורטים על התנהגות הקובץ, אשר בהם השתמשתי כדאי לדעת האם הקובץ אכן זדוני או לא.

## VirusTotal API

Virus Total API מאפשר ניתוח קבצים באמצעות מספר רב של מנועי אנטי-וירוס ושירותי סריקה מקוונים. על ידי שליחת קבצים ל API אני קיבלתי דוחות המכילים תוצאות סריקה ממנועים שונים. אני לקחתי את התוצאות אך ורק של מנועים מפורסמים ומהימנים על מנת לוודא שהתוצאה שאני מקבל הינה תקינה ונכונה.

## OS library

ספריית OS בפייתון מספקת ממשק לפעולות מערכת הפעלה כמו עבודה עם קבצים ותיקיות, ניהול תהליכים והרשאות מערכת. היא מאפשרת לבצע פעולות כמו יצירת קבצים ותיקיות, הזזה ומחיקה של קבצים ותיקיות. אני ביצעתי שימוש מורחב בספרייה זו על מנת לעבוד עם הקבצים והספריות שאותם אני רוצה לסרוק.

## PyQt6

PyQt6 היא ספרייה המאפשרת פיתוח ממשקי משתמש גרפיים (GUI) בפייתון באמצעות שימוש בQT Framework היא מספקת כלים לעיצוב והגדרת ממשקי משתמש מורכבים ואינטראקטיביים, כולל חלונות, תפריטים, כפתורים וטפסים. PyQt6 איפשרה ליישם ממשק גרפי נוח ומודרני בקלות וביעילות ולכן בחרתי להשתמש בה.

## Pymongo

Pymongo היא ספרייה בפייתון המאפשרת חיבור וניהול של מסד הנתונים MongoDB. באמצעות Pymongo, יצרתי קשר עם בסיס הנתונים על מנת להוסיף ולעדכן נתונים בצורה פשוטה. הספרייה מאפשרת לי לשמור את תוצאות הסריקות והניתוחים באופן מקומי, לניהול ארגון והצגת המידע למשתמש בצורה פשוטה.



## תיאור מצב קיים

במערכת קיום יש ממשק משתמש (gui) אשר מאפשר למשתמש להציב קבצים לסריקה.

המערכת סורקת את הקובץ שומרת את התוצאה לבסיס נתונים, למשתמש יש את האפשרות להצגת המידע הקיים בבסיס הנתונים על ידי ממשק המשתמש.

## ניתוח חלופות ומדוע בחרתי דווקא בחלופה הקיימת

במהלך הכנת הפרוייקט נאלצתי לבחור בין אופציות לטיפול בשלושה נושאים:

ניתוח סטטי

ניתוח דינאמי

ממשק משתמש (GUI)

ניתוח סטטי:

ישנם כלים רבים לניתוח סטטי של קבצים המשמשים בזיהוי ואבחון תוכנות זדוניות ותוכנות אחרות. כלים אלו מספקים מגוון רחב של יכולות וכוללים, בין היתר:

ClamAV: מערכת אנטי-וירוס פתוחה לגילוי וירוסים ותוכנות זדוניות.

PEiD: כלי לזיהוי מארזים ודחיסות בקבצי PE.

Radare2: מסגרת קוד פתוח לניתוח בינארי מתקדם.

IDA Pro: כלי חזק לפירוק והרכבה של קוד בינארי.

Ghidra: כלי לפירוק והרכבה מבית ה-NSA לניתוח בינארי.

## MALWARE ANALYSIS AND DETECTION

לאחר מחקר אני החלטתי להשתמש בYARA Hash מכמה סיבות:

פשטות ויעילות: ניתוח מהיר ויעיל של קבצים.

זיהוי דפוסים מותאמים אישית: חוקים מותאמים לזיהוי תוכנות זדוניות.

זיהוי חד ערכי: יצירת hash ייחודי לזיהוי מדויק של קבצים.

תמיכה בקבצים רבים: תמיכה במגוון פורמטי קבצים.

שימוש רחב ואינטגרציה קלה: שימוש נרחב ואינטגרציה פשוטה במערכות קיימות.

## ניתוח דינאמי

בנוסף לניתוח סטטי רצתי להגביר את היכולת של המערכת לזהות קובץ זדוני ולכן החלטתי להוסיף דרכים לניתוח דינאמי בנוסף לניתוח הסטטי.

בשוק קיימים כלים רבים לניתוח דינאמי של קבצים כאשר רובם הם ארגזי חול המאפשרים הרצה מדומה במערכת מבודדת, דוגמה לכאלה שיכולתי להשתמש בהם:

Cuckoo Sandbox: מסגרת פתוחה לניתוח דינמי של תוכנות זדוניות בסביבת ארגז חול. בהתחלה אני תכננתי להשתמש בסאנדבוקס זה מכיוון שהוא יעיל מאוד. הסאנדבוקס מותקן על המערכת ומריץ את הקבצים בסביבה מבודדת על אותו המחשב בעזרת VM בנוסף לכך הוא מנתח בצורה אוטומטית את התוכנה ומחזיר תשובה על אותה הרשת, כלומר הוא אינו צורך שימוש באינטרנט על מנת לנתח קבצים. אך מכיוון שהוא מריץ את ה-VM על המחשב המחשב שמריץ אותו צריך להיות עוצמתי מספיק מה שהמחשב שלי לא ולכן הורדתי את הרעיון ועברתי ל-API.

FireEye: פלטפורמת אבטחה לניהול איומים וניתוח דינמי.

Joe Sandbox: כלי לניתוח דינמי של תוכנות זדוניות עם דיווח מפורט.

Malwr: שירות מבוסס קהילה לניתוח דינמי של תוכנות זדוניות.

אז מדוע בחרתי דווקא ב-HybridAnalysis-VirusTotal-

בחרתי ב-VirusTotal מכמה סיבות:

וירוס טוטאל מאפשר מבחר רחב מאוד של מנועי ניתוח דינאמי (וסטטי) שיעברו על הקובץ ויחזירו תוצאה מפורטת.

בנוסף וירוס טוטל מאפשר גישה חופשית לנתונים קיימים כלומר אני יכול לחסוך את זמן ההעלאה והניתוח אם מדובר בוירוס שכבר נותח בעבר באתר.

בחרתי ב-HybridAnalysis מכמה סיבות:

להיברייד אנליסיס יש ממשק ממש נוח לשימוש. תהליך שליחת הבקשה וקבלת המידע הוא תהליך מהיר אשר מספק לי תוצאה מפורטת של הסריקה אשר כוללת בתוכה דברים הרבה מעבר כמו לדוגמא, שימוש באינטליגנציה מלאכותית כדי לעבור על הקבצים שנסרקו.

## בסיס נתונים

אני בחרתי להשתמש ב-MongoDB מכיוון שלעומת רוב בסיסי הנתונים הקיימים מונגו לא משתמש ב-SQL דבר שהופך אותו למאוד מאוד פשוט ונוח לשימוש. בנוסף מונגו קל להתקנה ושימוש בצורה מהירה ולכן החלטתי להשתמש בו


## אפיון המערכת:

### ניתוח דרישות המערכת:


הפרוייקט שלי הינו פרוייקט פשוט שלא דורש הרבה כוח מהמערכת עליה הוא יושב.

המערכת תרוץ בסביבת ווינדוס 11 אך יכולה לעבוד גם בווינדוס 10.

המערכת רצה על המחשב הנייד שלי שהמפרט שלו הוא:

 Windows specifications	
Edition	Windows 11 Home
Version	23H2
Installed on	10/5/2022
OS build	22631.3593
Experience	Windows Feature Experience Pack 1000.22700.1003.0
<a href="#">Microsoft Services Agreement</a>	
<a href="#">Microsoft Software License Terms</a>	

## MALWARE ANALYSIS AND DETECTION


**Device specifications**

Device name	LotanYahav
Processor	11th Gen Intel(R) Core(TM) i5-11400H @ 2.70GHz 2.69 GHz
Installed RAM	16.0 GB (15.6 GB usable)
Device ID	51978C11-8E81-4C39-9D13-315E97D5E751
Product ID	00342-21966-25405-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display


Generic PnP Monitor and Intel(R) UHD Graphics Properties

Adapter

Monitor

Color Management

Adapter Type



Intel(R) UHD Graphics

Properties

Adapter Information

Chip Type:

Intel(R) UHD Graphics Family

DAC Type:

Internal

Adapter String:

Intel(R) UHD Graphics

Bios Information:

Intel Video BIOS

Total Available Graphics Memory:

8138 MB

Dedicated Video Memory:

128 MB

System Video Memory:

0 MB

Shared System Memory:

8010 MB

List All Modes

OK

Cancel

Apply

Requirements.txt:

cryptography==42.0.5

itemadapter==0.8.0

pymongo==4.7.2

PyQt6==6.7.0

PyQt6\_sip==13.6.0

Requests==2.32.3

Scrapy==2.11.1

yara\_python==4.3.1

Modern Operating System:

Windows 7 or 10 or 11

x86 64-bit CPU (Intel / AMD architecture).

4 GB RAM

GB free disk space 5

## MALWARE ANALYSIS AND DETECTION

### מטרות המערכת:

- לאפשר סריקה נוחה ומהירה של קובץ חשוד עבור המשתמש.
- מחיקה של הקובץ
- הצגה ויזואלית של התהליך

### מודולים במערכת:

1. קבצי הסריקה – סקריפטים אלו כתובים בפייתון וכל אחד מהם אחראי על ניתוח אחר, ישנם ארבעה סקריפטים והם: yara, hash, hybrid, vt כל אחד מהם אחראי על הניתוח שלו וחייב כל אחד מהם כדי שהפרוייקט יעבוד.
2. אינטגרציה – בפרוייקט קיים קוד אינטגרציה. הקוד הזה שנמצא בקובץ Manager.py מנהל את כל התהליכים שקורים בפרוייקט בין אם זה להפעיל את הסריקות להעביר את התוצאות בין הקבצי סריקה לקבצי ה GUI להצגת המשתמש ולשמור את הנתונים בבסיס הנתונים. כל הפעולות שקורות במערכת חייבות לעבור דרך קובץ האינטגרציה.
3. ממשק המשתמש – ממשק המשתמש יאפשר הצגה ויזואלית של הסריקות ושל בסיס הנתונים. המשתמש יוכל לבחור בעזרת סרגל בצד המסך את הקטגוריה ובהתאם לכך יוצג מידע על המסך.
4. בסיס הנתונים – הנתונים שיתקבלו מהסריקות ישמרו בצורה אוטומטית לתוך בסיס נתונים של MongoDB על מנת לאפשר גישה נוחה ומהירה אליהם כאשר המשתמש ירצה לראות מידע מסריקות קודמות.



### אינטראקציה בין המודלים:

המודולים יעבדו בצורה שבה כל הפעולות יעברו דרך סקריפט האינטגרציה (המנהל) שהוא כביכול כמו מנהל.

1. המשתמש מכניס מיקום של קובץ ושולח לסריקה.
2. המנהל מקבל את המיקום ושולח אותו לארבעת הניתוחים.
3. כל אחד מהסקריפטים של הניתוח עובר על המיקום שנשלח עליו ומחזיר תוצאות.
4. כאשר כל התוצאות הגיעו בהצלחה המנהל ישמור אותם בבסיס הנתונים וישלח לממשק המשתמש.
5. ממשק המשתמש יציג את התוצאות למשתמש עצמו.

## ארכיטקטורה

### תיאור הארכיטקטורה:

הארכיטקטורה תהיה בנויה בצורה כזאת שכל המידע עובר דרך המנהל(סקריפט האינטגרציה) ולכן כל החלקים והמודלים מחוברים עליו בצורה כזו שתאפשר גישה מהירה ונוחה למידע.

### המרכיבים העיקריים במערכת:

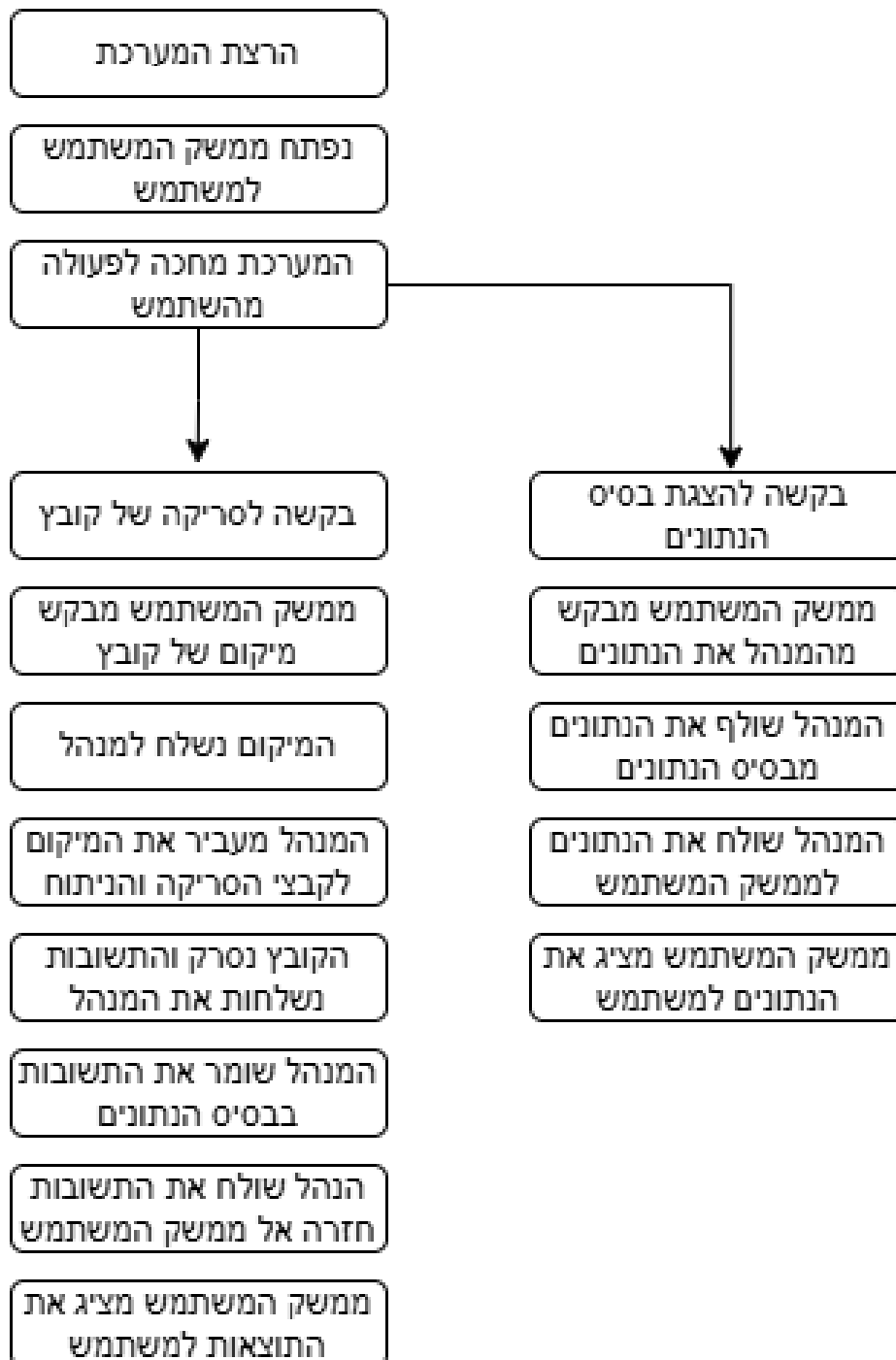
ממשק המשתמש: נותן ויזואליות מודרנית ונוחה לשימוש שתקל על המשתמש בעת ההרצה של המערכת.

ניתוח של הקובץ: פעולות הניתוח שחוזרות על עצמן שוב ושוב, פעולות אלה חשובות מאוד הן מקבלות קובץ מנתחות אותו ושלחות את המידע חזרה על המנהל. הפעולות יקרו בצורה הבאה:

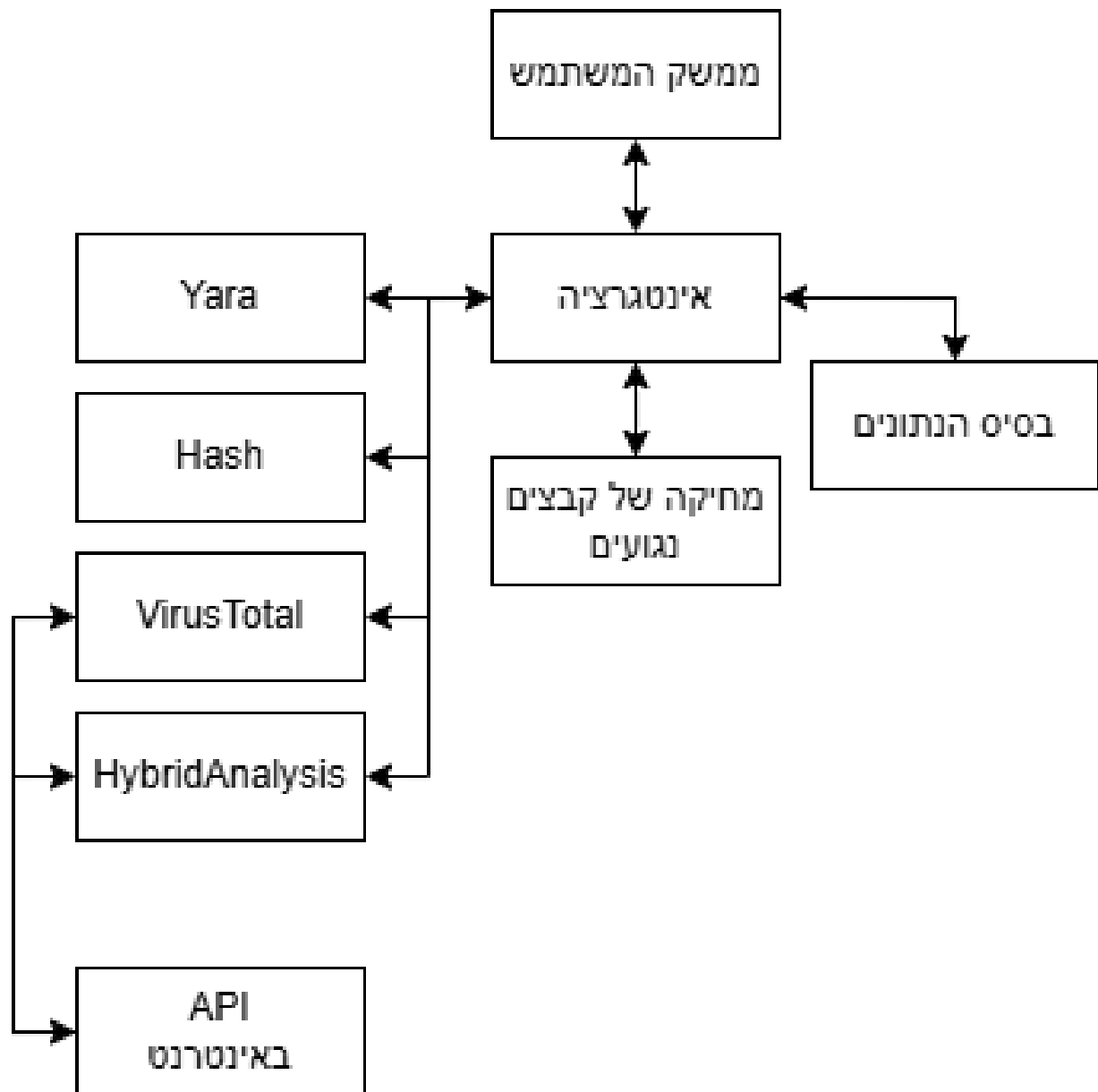
- המיקום של הקובץ ישלח לסקריפט הניתוח
- הסקריפט יריץ את פעולות הניתוח שהגדרתי לו
- הסקריפט יחזיר תשובה בהתאם לתוצאות
- המנהל יקבל את התשובה ויעביר אותה הלאה

בסיס הנתונים: בסיס הנתונים אחראי על שמירת המידע בצורה מאובטחת אך נוחה לשימוש.

## הארכיטקטורה בצורת top down level design



מבנה המערכת:



## אבטחת מידע:

תיאור ההגנה:

המערכת אינה מגנה מהתקפות המערכת הינה יותר מיועדת למניעת ההתקפות לפני שהן קורות.

המערכת שיועדת לנתח קבצים תנתח את הקובץ לפני ההרצה הראשונית ובכך תמנע את התקיפה לפני שהיא מתחילה לפעול ולפגוע במחשב המשתמש.

פעולת הניתוח לפני הרצה ברוב המקרים תהיה פעולה חכמה שתמנע תקיפות רבות אך למרות זאת ישנם תקיפות שיכולות לעקוף את המערכת. דוגמאות מאוד טובות לכך הן:

מתקפות יום אפס (Zero-Day) – מתקפת Zero-Day מנצלת פרצות אבטחה בתוכנה שטרם נודעו ליצרנים ולמפתחים. מאחר והפרצות אינן ידועות, גם לא קיימות חתימות זיהוי עבורן, ולכן כלים לסריקה וניתוח קבצים אינם מסוגלים לזהות או לעצור מתקפות אלו בזמן אמת. מתקפות אלו נשארות מסוכנות עד לזיהוי ועדכון התוכנה.

מתקפות פשינג – מתקפות מסוג זה לרוב אינן כוללות הורדה של קבצים שניתן לסרוק ומכיוון שהמערכת שלי לא מגנה מפני URL זדוניים פשינג יכול לעקוף את המערכת.

הצפנות:

בחרתי שכן להצפין את המידע שאני שומר בבסיס הנתונים למרות שאינו מידע רגיש על מנת להוסיף להרגשה של האבטחה.

השתמשתי ב Fernet שהוא חלק מספריית cryptography בפייתון שמאפשרת הצפנות סימטריות בעזרת מפתח, מכיוון שהמידע שמור על מחשב המשתמש ולא נעשה העברה באינטרנט אז אין צורך לוודא שלמות ולהצפין בצורה שתאפשר שליחה ברשת ולכן הכנתי פעולה חד פעמית שתיצור מפתח ותשמור אותו על המחשב של הלקוח בתור משתנה סביבתי (environmental variable) שיאפשר גישה אליו אך ורק למשתמש.

## ניתוח Use Cases

תיאור המקרה העיקרי:

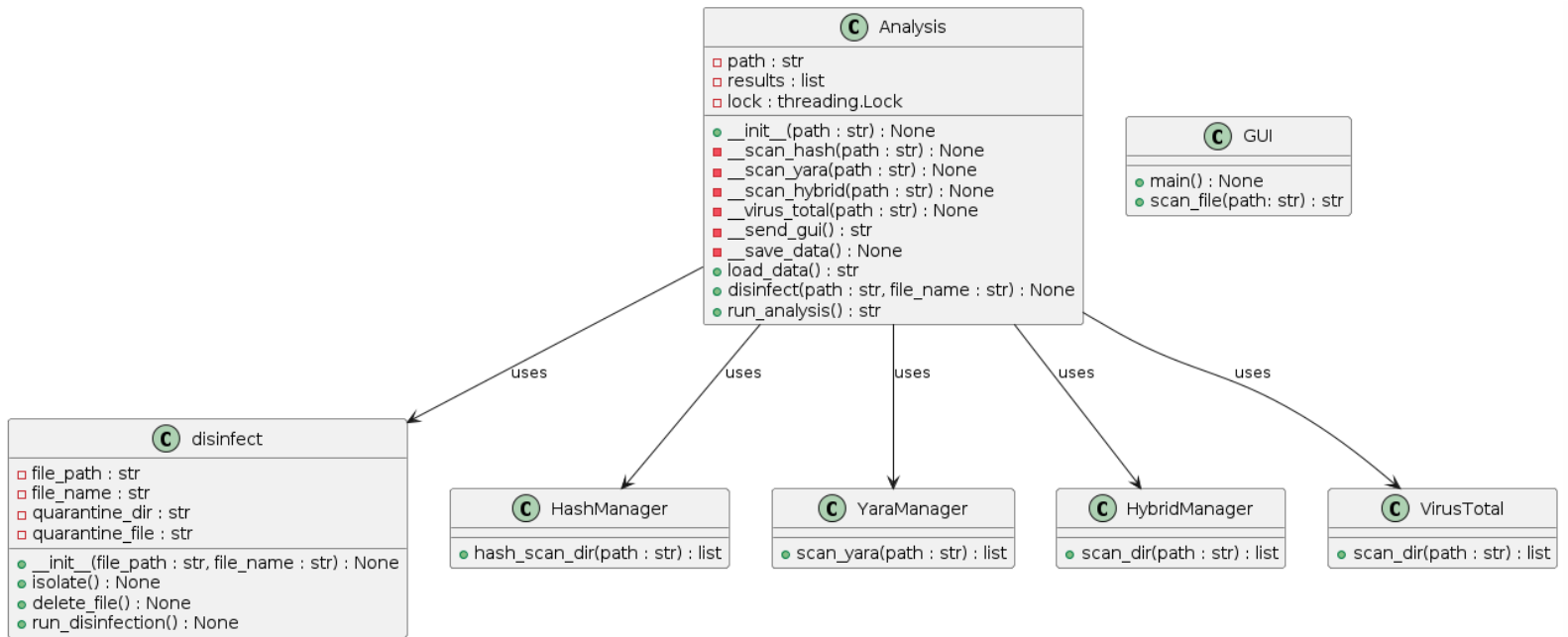
במקרה העיקרי המערכת תעשה שימוש בכל 4 הסוגים השונים של הניתוחים כלומר:

המידע יעבור ממשק המשתמש אל המנהל (מנהל אינטגרציה) אל הניתוחים ובסיס הנתונים וחזור כמו שאנשים חוצים גשר, המידע יעבור מצד לצד אך תמיד יעבור דרך המנהל.

אומנם במקרים מסוימים אנחנו יכולים להגיע למצב שחלק מהסקריפטים לא יפעלו כלל.

זה תקף בעיקר עבור HybridAnalysis אשר עושה שימוש רק בארגז חול מה שמונע ממנו לנתח קבצים מסוגים מסוימים כלומר אם הסוג קובץ לא ברשימה אז הוא לא יסרק.

במקרה כזה אין מה לדאוג מכיוון שהניתוחים האחרים יפעלו על כל סוג של קובץ בין אם זה קובץ TXT או portable executable (PE).



## רכיבי ממשק:

במערכת ישנם שני ממשקים קיימים

ממשק המשתמש:

ממשק זה אחראי על תצוגת המידע אצל המשתמש. מטרת הממשק היא לקחת את המידע הקיים ולהעביר אותו ממצב לא קריא, למייצג נעים לעין ומודרני שהמשתמש יוכל לקרוא ולהבין.

ממשק זה עושה שימוש רב ב PyQt6 על מנת לעצב את ה GUI בצורה מודרנית ויפה.

ממשק המנהל:

ממשק זה הוא בעצם סקריפט האינטגרציה.

הוא מבצע את כל הפעולות החשובות ומעביר דרכו את כל המידע, כומר מטרתו היא לנהל את התהליכים של המערכת. כאשר ממשק המשתמש רוצה להוציא מידע מבסיס הנתונים. הבקשות והמידע יעברו דרך ממשק המנהל על מנת לשמור על סדר ופעולות תקינות.



## תיאור תוכנה:

### Windows 11

Windows 11 היא המהדורה העדכנית ביותר של מערכת ההפעלה Windows מבית מיקרוסופט, שהוכרזה ב-24 ביוני 2021. גרסה שוחררה ב-20 בספטמבר 2022, והיא כוללת שיפורים משמעותיים בממשק המשתמש, בתאימות ובביצועים. היא מציעה חווית משתמש מתקדמת עם עיצוב מודרני וכלים חדשים לשיפור הפרודוקטיביות והביצועים.

### :Visual Studio Code

Visual Studio Code (VS Code) היא סביבת פיתוח משולבת (IDE) חינמית ומבוססת קוד פתוח מבית מיקרוסופט, התומכת במגוון רחב של שפות תכנות וכלים. VS code מציעה תכונות מתקדמות כגון ניפוי באגים, השלמה אוטומטית של קוד, אינטגרציה עם Git, ויכולת הרחבה באמצעות תוספים, מה שהופך אותה לאחת מסביבות הפיתוח הפופולריות ביותר בקרב מפתחים.

### :Python 3.10.12

Python 3.10.12 היא של גרסה של שפת התכנות פייתון הכוללת תכונות חדשות ואופטימיזציות שונות לשיפור הביצועים והקריאות של הקוד. פייתון היא שפת תכנות ברמה גבוהה למטרות כלליות. היא תומכת במספר פרדיגמות תכנות, כולל תכנות מובנה, מונחה עצמים ופונקציונלי. השפה נחשבת לעיתים קרובות כ"שפה הכוללת סוללות" בשל הספרייה הסטנדרטית המקיפה שלה

## תרשים מסכים:

The screenshot shows a web-based interface for malware scanning. It features a green sidebar on the left with three menu items: "Logo", "Scan", and "Database". The main content area has a dark green header with the word "SCAN" in white. Below the header, there is a light gray section containing a form. The form has a label "scan file" and a sub-label "enter file or directoy path:" followed by a text input field containing the word "path". Below the input field is a large, empty rectangular box labeled "results". At the bottom of the form is a rounded rectangular button labeled "Scan".

המסך הראשי של סריקת הקבצים יראה כך, דרך מסך זה המשתמש יוכל להכניס מיקום של תיקייה או קובץ ולסרוק את הנתונים שלה ולקבל תוצאות.

## MALWARE ANALYSIS AND DETECTION

Logo

Database

Scan

database

Database

מס סריקה	תאריך	שעה	וירוס	שם הוירוס	PAT II	סוג הוירוס	שעת המחיקה
1	10.1.20 24	8:55:31	TR UE	virus_na mc	C:\PA TH	Spywa rc	8:57:25
2	10.1.20 25	9:55:31	TR UE	virus_na me	C:\PA TH	Spywa re	9:57:25
3	10.1.20 26	10:55:31	TR UE	virus_na me	C:\PA TH	Spywa re	10:57:25
4	10.1.20 27	11:55:31	TR UE	virus_na me	C:\PA TH	Spywa re	11:57:25
5	10.1.20 28	12:55:31	TR UE	virus_na me	C:\PA TH	Spywa re	12:57:25
6	10.1.20 29	13:55:31	TR UE	virus na me	C:\PA TH	Spywa re	13:57:25

במסך זה המשתמש יוכל לעבור על הנתונים של סריקות קודמות ולראות מתי היה וירוס.

גיטהאב לקוד המלא:

[reallotlot/Spyware-Project \(github.com\)](https://github.com/reallotlot/Spyware-Project)

manager.py

```
from . import HashManager ,YaraManager
,HybridManager ,VirusTotal, Disinfection
import os
import threading
import pymongo

class Analysis():
    def __init__(self, path) -> None:
        self.path = path
        self.results = [None for i in range(4)]
        self.lock = threading.Lock()

    def __scan_hash(self, path):
        res = HashManager.hash_scan_dir(path)
        with self.lock:
            self.results[0] = ('hash', res)
        return res

    def __scan_yara(self, path):
        res = YaraManager.scan_yara(path)
        with self.lock:
```

MALWARE ANALYSIS AND DETECTION

```

        self.results[1] = ('yara', res)
    return res

def __scan_hybrid(self, path):
    res = HybridManager.scan_dir(path)
    with self.lock:
        self.results[2] = ('hybrid', res)
    return res

def __virus_total(self, path):
    res = VirusTotal.scan_dir(path)
    with self.lock:
        self.results[3] = ('vt', res)
    return res

def __send_gui(self):
    finalResult = ''
    # turn the analysis results into human readable text
    for res in self.results:
        if res[1] is not None and res[1] != [] and res[1] != {}:
            for file in res[1]:
                curPath = file['path']
                if curPath not in finalResult:
                    finalResult += f'Malicious file detected at {curPath}\n'
            if finalResult == '':
                return f"{self.path} is clear!"

```

```

return finalResult

def run_analysis(self):
    #set up the threads
    hashThread = threading.Thread(target=self.__scan_hash,
    args=(self.path,))
    yaraThread = threading.Thread(target=self.__scan_yara,
    args=(self.path,))
    hybridThread = threading.Thread(target=self.__scan_hybrid,
    args=(self.path,))
    vtThread = threading.Thread(target=self.__virus_total,
    args=(self.path,))
    threads = [hashThread, yaraThread, hybridThread, vtThread]

    #run the threads
    print("starting the threads")
    for thread in threads:
        thread.start()

    print("waiting for the threads to finish")
    #wait for threads to finish
    for thread in threads:
        thread.join()

```

```
# print the results
print("all threads finished.")
return self.__send_gui()

if __name__ == "__main__":
    pass
```

```
import os
import shutil

class disinfect():
    def __init__(self, file_path, file_name) ->
None:
        self.file_path = file_path
        self.file_name = file_name
        self.quarantine_dir =
f'..\Spyware_Manager\quarantined_files\{os.path.spli
text(self.file_name)[0]}'
        self.quarantine_file =
f'..\Spyware_Manager\quarantined_files\{os.path.spli
text(self.file_name)[0]}\{self.file_name}'

    def isolate(self):
        if not os.path.exists(self.quarantine_dir)
and os.path.exists(self.file_path) :

os.makedirs(os.path.abspath(self.quarantine_dir))
        shutil.move(self.file_path,
self.quarantine_dir)
        print("isolated")
        else:
            print("file already isolated or doesnt
exist")
```



```
def delete_file(self):
    print("deleting")
    try:
        with open(self.quarantine_file, 'wb') as
file:
            file.write(b'')
            os.remove(self.quarantine_file)
            shutil.rmtree(self.quarantine_dir)
            print("finished")
    except Exception as e:
        print(e)

def run_disinfection(self):
    self.isolate()
    self.delete_file()

if __name__ == "__main__":
    pass
```

```
import hashlib
import os

#get a file md5 hash
def get_md5(path):
    md5 = None
    with open(path, 'rb') as file:
        text = file.read()
        md5 = hashlib.md5(text).hexdigest()
    return md5

# get a file sha256 hash
def get_sha256(path):
    sha256 = None
    with open(path, 'rb') as file:
        text = file.read()
        sha256 = hashlib.sha256(text).hexdigest()
    return sha256

#check for the md5 and sha256 in the malicious hash
file (respectively)
def check_md5(md5):
    path =
f'{os.path.dirname(os.path.abspath(__file__))}\hashf
iles\md5.txt'
    with open(path, 'r') as file:
        text = list(file.read().split("\n"))
        if md5 in text:
```

# MALWARE ANALYSIS AND DETECTION

```

        print(md5)
        return True
    return False

def check_sha256(sha256):
    hash_path =
f'{os.path.dirname(os.path.abspath(__file__))}\hashf
iles\sha256hashes.txt'
    hashes = list(open(hash_path,
'r').read().split("\n"))
    for i in range(len(hashes)):
        if sha256 in hashes[i]:
            return True
    return False

def hash_scan_file(path):
    md5 = get_md5(path)
    sha256 = get_sha256(path)
    #print(md5 + "\n" + sha256 + "\n")
    return check_md5(md5) or check_sha256(sha256)

def hash_scan_dir(path):
    results = []
    for file in os.listdir(path):
        file_path = os.path.join(path, file)
        if os.path.isdir(file_path):
            hash_scan_dir(file_path)
        else:

```

MALWARE ANALYSIS AND DETECTION

```
        result = hash_scan_file(file_path)
        if result:
            results.append({'path':
os.path.abspath(path)})
        return results

if __name__ == "__main__":
    pass
```

```
import os
import re
import subprocess
import time
import requests
import platform

from cryptography import fernet

# load the encrypted api key
def load_key():
    enc_key = os.getenv("API_ENCRYPTION_KEY")
    path =
os.path.join(os.path.dirname(os.path.abspath(__file_
_)), 'api_keys.txt')
    with open(path, "rb") as file:
        api_key = file.read().split(b"\n")[1]

    return
fernet.Fernet(enc_key).decrypt(api_key).decode()

#Global variables
API_KEY = load_key()
API_ENDPOINT = r'https://www.hybrid-
analysis.com/api/v2/submit/file'

def get_win_ver():
    command = 'systeminfo | findstr /B /C:"OS Name"'
```

# MALWARE ANALYSIS AND DETECTION

```

    result = subprocess.run(command,
stdin=subprocess.PIPE, stdout=subprocess.PIPE,
shell=True)
    match = re.search(r'Windows (\d+)', re-
sult.stdout.decode())
    if match:
        return match.group(1)
    else:
        return "Unknown version"

def get_env_id():
    env_ids = {
        'Linux': 310,
        '11': 140,
        '10': 160
    }

    env_id = ''
    syst = platform.system()

    if syst == "Windows":
        env_id = env_ids[get_win_ver()]
    else:
        env_id = env_ids[syst]

    return(env_id)

def scan_file(path, file_name):

```

# MALWARE ANALYSIS AND DETECTION

```
#check if the file is valid for sandbox dynamic
analysis
file_types = '.exe .dll .scr .sys .pdf .doc
.docx .xls .xlsx .ppt .pptx .rtf .js .vbs .ps1 .zip
.rar .7z .msi .iso'.split(' ')
for type in file_types:
    if type in file_name:
        break
else:
    #print('invalid file: ', file_name)
    return None #file is not valid for scanning

#set up the data for the sandbox analysis
payload = {
    'environment_id': get_env_id(),
    'submit_name': file_name,
}
files = {
    'file': (file_name, open(path, 'rb'),
'text/plain')
}
headers = {
    'accept': 'application/json',
    'api-key': API_KEY
}

# Submit the file for analysis in the sandbox
response = requests.post(API_ENDPOINT, head-
ers=headers, files=files, data=payload)
analysis_id = ''
```

# MALWARE ANALYSIS AND DETECTION

```

if response.status_code == 201:
    analysis_result = response.json()
    analysis_id = analysis_result.get('job_id')
    #print("submitted file")
    #print("analysis ID:", analysis_id)
else:
    #print("Failed to submit file.")
    #print("Status Code:", response.status_code)
    #print("Response:", response.text)
    return None

#wait for the result
status_endpoint = f"https://www.hybrid-
analysis.com/api/v2/report/{analysis_id}/state"
while True:
    status_response = re-
quests.get(status_endpoint, headers=headers)
    status_data = status_response.json()

    if status_data.get('state') == 'SUCCESS':
        #print("analysis complete.")
        break
    elif status_data.get('state') ==
'IN_PROGRESS':
        #print("analysis in progress.")
        time.sleep(5)
    else:
        #print("unexpected status:", sta-
tus_data)
        return None

```



```
#retrieve the data from the reprot
report_endpoint = f"https://www.hybrid-
analysis.com/api/v2/report/{analysis_id}/summary"
report_response = requests.get(report_endpoint,
headers=headers)

if report_response.status_code == 200:
    info = report_response.json()
    info_dict = {
        'name': info.get('submit_name'),
        'path': f'{os.path.abspath(path)}',
        'verdict': info.get('verdict')
    }
    if info_dict['verdict'] in ['mali-
cious', 'suspicious']:
        return info_dict
    else:
        return None
else:
    #print("failed to fetch response")
    return None

def scan_dir(path):
    results = []
    if not os.path.isdir(path):
        res = scan_file(path,
os.path.basename(path))
        if res is not None: results += res
```

# MALWARE ANALYSIS AND DETECTION

```

else:
    for file in os.listdir(path):
        file_path = os.path.join(path, file)
        if os.path.isdir(file_path):
            results += scan_dir(file_path)
        else:
            res = scan_file(file_path, file)
            if res is not None:
                results.append(res)
    return results

if __name__ == "__main__":
    pass

```

```
import time
import requests
import hashlib
import os

from cryptography import fernet

#LOAD ENCRYPTED API KEY
def load_key():
    enc_key = os.getenv("API_ENCRYPTION_KEY")
    path =
os.path.join(os.path.dirname(os.path.abspath(__file_
_)), 'api_keys.txt')
    with open(path, "rb") as file:
        api_key = file.read().split(b"\n")[0]

    return
fernet.Fernet(enc_key).decrypt(api_key).decode()

if True:
    api_key = load_key()

def scan_file(path):
    #trusted vendors
```

# MALWARE ANALYSIS AND DETECTION

```

trusted_vendors = ['google', 'avast', 'avg',
'kaspersky', 'malwarebytes', 'microsoft', 'bitde-
fender']

#set the endpoint to files and apply the api key
as the header
endpoint =
'https://www.virustotal.com/api/v3/files'
headers = {
    "accept": "application/json",
    'x-apikey': api_key
}

#upload file data to virus total
with open(path, 'rb') as file:
    sha256 = hash-
lib.sha256(file.read()).hexdigest()
    files = {'file': file}
    upload_response = requests.post(endpoint,
headers=headers, files=files)

    if upload_response.status_code == 200:
        # get the analysis id
        upload_result = upload_response.json()
        analysis_id = up-
load_result.get('data').get('id')
        #print(f"File uploaded successfully. Analy-
sis ID: {analysis_id}")

        # Check the analysis status

```

# MALWARE ANALYSIS AND DETECTION

```

analysis_url =
f'https://www.virustotal.com/api/v3/analyses/{analysis_id}'

# wait for completion
while True:
    analysis_response = requests.get(analysis_url, headers=headers)
    analysis_result = analysis_response.json()
    if analysis_response.status_code == 200:
        if analysis_result.get('data').get('attributes').get('status') == 'completed':
            #print("Analysis completed!")
            break
        else:
            time.sleep(5)
    else:
        #print(f"Error: {analysis_response.status_code}")
        #print(analysis_response.text)
        return None

#get the results
result_endpoint =
f'https://www.virustotal.com/api/v3/files/{sha256}'
result_response = requests.get(result_endpoint, headers=headers)

if result_response.status_code == 200:

```

# MALWARE ANALYSIS AND DETECTION

```

        results = result_response.json()
        res = results.get('data').get('attributes').get('last_analysis_results')
        for vendor in res:
            if vendor.lower() in trusted_vendors:
                if
res[vendor]['category'].lower() in ['malicious',
'suspicious']:
                    info = {
                        'name' : vendor,
                        'path' :
f'{os.path.abspath(path)}',
                        'category' : True
                    }
                    return info
            else:
                return None

        else:
            return None

def scan_dir(path):
    results = []
    res = ''
    if not os.path.isdir(path):
        res = scan_file(path)
        if res is not None:

```

# MALWARE ANALYSIS AND DETECTION

```

        return res
    else:
        for file in os.listdir(path):
            file_path = os.path.join(path, file)
            if not os.path.isdir(file_path):
                res = scan_file(file_path)
                if res is not None:
                    results.append(res)
            else:
                res = scan_dir(file_path)
                results = results + res

    return results

if __name__ == "__main__":
    pass

```

```
import os
import time
import yara

#DO NOT CHANGE THIS PATH UNLESS YOU KNOW WHAT YOU
#ARE DOING!
#DOING SO WILL CAUSE THE PROGRAM TO NOT BE ABLE TO
#USE MOST OF THE YARA RULES
#CAUSING SECURITY RISKS
compiled_rules_path =
f'{os.path.dirname(os.path.abspath(__file__))}\yaraf
iles\compiled-rule-master'

#one time run will only run once if the compiled
rules file is empty!!!!!!
def compile_rulemaster_rules():
    os.makedirs(compiled_rules_path)
    while not (os.path.exists(compiled_rules_path)):
        time.sleep(.3)
        print("loading")

    path =
f'{os.path.dirname(os.path.abspath(__file__))}\yaraf
iles\rule-master'
    for mal in os.listdir(path):
        rule_path = os.path.join(path, mal)
        rules = yara.compile(rule_path)
        print("saving rules")
```



```
rules.save(f'{compiled_rules_path}\compiled_{mal}')

def comp():
    if os.path.exists(compiled_rules_path):
        if os.listdir(compiled_rules_path) != []:
            pass#print("already compiled")
        else:
            compile_rulemaster_rules()
            #print("done")

def scan_yara(path):
    comp()
    results = []
    for rule_file in
os.listdir(compiled_rules_path):
        rules =
yara.load(os.path.join(compiled_rules_path,
rule_file))
        if not os.path.isdir(path):
            matches = rules.match(path)
            if matches != []:
                results.append({'path':
os.path.abspath(path), 'type': matches})
        else:
            for file in (os.listdir(path)):
                file_path = os.path.join(path, file)
                if os.path.isdir(file_path):
```

# MALWARE ANALYSIS AND DETECTION

```

        results = re-
sults.update(scan_yara(file_path))
    else:
        matches = rules.match(file_path)
        if matches != []:
            results.append({'path':
os.path.abspath(path), 'type': matches})
        return results

if __name__ == "__main__":
    pass

```

## מסקנות ורפלקציה :

התכנון המקורי היה להריץ סאנדבוקס ללא שימוש באינטרנט על המחשב שלי בתור כלי לניתוח דינאמי אך בגלל שהדרישות מהמחשב להרצת VM עליו היו גבוהות מדי בשביל המחשב שלי נאלצתי לוותר על הרעיון ולהסתפק ב-API של ארגזי חול אחרים.

בנוסף גיליתי שהפעולות של ניתוחים סטטים ודינאמיים אינם יעילים כאשר נתקלים בוירוס שפועל באינטרנט כמו לדוגמה פשינג.

העבודה על הפרוייקט עזרה לי להיכנס ולהבין את הנושא של ניתוחים סטטים ודינאמיים הרבה יותר לעומק ובצורה מעניינת. אני בטוח שבהמשך יצא לי להשתמש שוב בנושאים אלו והם תרמו לי המון לעתיד.

## ביבליוגרפיה

[VirusTotal - Home](#)

[Free Automated Malware Analysis Service \(hybrid-analysis.com\)](#)

<https://www.techtarget.com/searchsecurity/definition/sandbox#:~:text=A%20sandbox%20is%20an%20isolated,to%20test%20potentially%20malicious%20software.>

<https://www.perforce.com/blog/sca/what-static-analysis#:~:text=Static%20analysis%20is%20a%20method,compliant%2C%20safe%2C%20and%20secure.>

<https://www.parasoft.com/blog/static-analysis-and-dynamic-analysis/>

<https://www.w3schools.com>

<https://wiki.python.org/moin/GuiProgramming>