****

**עבודת גמר 5 יח"ל**

**נושא העבודה:** סביבת הרצה בטוחה – SafeBox

**שם תלמיד:** עומר כהן

**ת.ז:** 326705142

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

**שם המנחה:** ערן בינט

**מועד הגשה:** 24/12/22

תוכן עניינים

[1.מבוא 2](#_Toc122813856)

[2.תיאוריה 5](#_Toc122813857)

[3.תוצר סופי 9](#_Toc122813858)

[4.תהליך כתיבת הפרויקט 16](#_Toc122813859)

[5.מרכיבי הפרויקט 18](#_Toc122813860)

[6.תסריטי בדיקה 20](#_Toc122813861)

[7.רפלקציה 20](#_Toc122813862)

[8.הוראות התקנה ותפעול 21](#_Toc122813863)

[9.ביבליוגרפיה 21](#_Toc122813864)

[10.נספחים 21](#_Toc122813865)

# 1.מבוא

## 1.1 נושא העבודה

נושא הפרויקט שלי הוא סביבת הרצה בטוחה Sandbox. התכנה תקבל מהמשתמש קובץ הרצה של C++/C/C#/Python ותעקוב אחר קריאות המערכת שלו לקבצים, registry, sockets , RAM ועוד, על מנת לגבש מסקנה האם מדובר בוירוס או בתכנה תמימה, התוכנה תתעד את הבדיקות שלה ותאפשר למשתמש לצפות ב log הרצה. הסביבה תמומש באמצעות userland inline hooking ו - mini filter drivers. במסגרת הפרויקט יבוצע שימוש בפקודות win32/winapi מעל שפת ++C~~.~~

## 1.2 מטרות מרכזיות

המטרות המרכזיות של הפרויקט הן:

* Dynamic Analysis של קובץ הרצה ללא חסימה במחשב.
* דיווח על קריאות למשאבי המחשב (sockets, files, registry) למשתמש – מה הכוונה?
* מתן גמישות בבחירת פרמטרי הריצה שהמשתמש מעוניין בהם, ודרך כך השגת גמישות בסריקת הקובץ
* ניתוח של קבצים של C++/C/C#/Python שהן? #C, C++, PYTHON?

המטרות האישיות המרכזיות של הפרויקט הן:

* למידת עומק על תהליכי העבודה של מערכת Windows במובן ה WinAPI וגם במובן קונספטים שונים כמו DLL, PE, RAM and memory management, kernel
* שיפור יכולות הפיתוח הגרפי שלי, תוך התמקצעות בכלי C# winforms
* התמקצעות בשפה C++
* הרחבת הבנתי בתחום התקפות הסייבר תוך למידה על אמצעי הגנה רבים
* הבנה רחבה יותר על תהליך ה compilation של C#/C++ וה interpretation של python

## 1.3 רציונל

מאז ומעולם אני זוכר שכל נושא הווירוסים ריתק אותי, ישנן מתקפות שבוצעו בידי אדם יחיד אשר עשו נזק של מיליארדי דולרים ושיתקו חברות ענק לשבועות, ותמיד רציתי לדעת איך דברים כל כך הרסניים אך גם מעניינים בעצם עובדים. אני בחרתי בפרויקט אשר דווקא מתמקד בצד המגן מכיוון שאני לא מעוניין להסב נזק אך אני מעוניין בידע של איך אותם ווירוסים עובדים.

הבחירה במערכת sandbox זימנה עבורי אתגר המחייב העמקה ב kernel של מערכת הפעלה ושפות low level. בחרתי להתמקד בפיתוח מעל מערכת ההפעלה Windows מאחר שזוהי מערכת ההפעלה הנפוצה ביותר בעולם וגם בעלת הכי הרבה חולשות. בנוסף, אני משתמש בה מהיום בו נחשפתי למחשבים ויש לי הכרות רבה איתה. מעבר לזה נחשפתי ל API העשיר אשר Microsoft מציעה עם documentation מפורט מאוד והבנתי שיהיה לי מאוד נוח לעבוד על פלטפורמה זו. אני בחרתי ב ++C מכיוון שזוהי שפה שאני מעוניין ללמוד כבר שנים וניתן להגיד כי בחרתי את השפה לפני הפרויקט. אני חושב שזוהי שפה מרתקת אשר מציעה למתכנת אפשרויות אשר מעט שפות אחרות מאפשרות.

## 1.4 קישור לחומר הנלמד

על אף שנושא ה ++C לא נלמד בכתה, אני עושה שימוש נרחב בכל הנלמד בכתה י"ב, בתחום מערכות ההפעלה כגון PE,DLL,ntdll וכו'. יתר על כך, אני משתמש בכלי C# winforms על מנת לפתח את ה GUI לפרויקט. ה source control נעשה ע"י git ושמירה בענן ע"י github אשר נלמדו במהלך כיתה י"א.

# 2.תיאוריה

## 2.1 תיאוריה

## 2.2 מוצרים קיימים בשוק

שוק תוכנות ה Sandboxing הינו שוק רחב עם מספר רב של תוכנות המוצעות למכירה, כל מערכת מתמקדת בנושא שונה, מיושמת בצורה שונה ועובדת עם מערכות הפעלה וקבצים שונים.

**Kaspersky Sandbox** – התוכנה בבעלות חברת אבטחת המידע Kaspersky, לחברה זו שורה ארוכה של מוצרים איכותיים ותוכנת ה Sandboxing שלה נמצאת בין המוצרים האיכותיים ביותר ברשימה זו. ה Sandbox פעיל על מספר פלטפורמות ועובד עם מספר רב של קבצים, כדלקמן:

פלטפורמות – , Windows 10,8.1, 8, 7, MacOS, IOS, Android

סוגי קבצים – , Executables, DLLs, NET objects, PDFs. APKs.URLs

כאשר משתמש מעלה קובץ יש לו אפשרות לבחור לעשות בדיקות ספציפיות על הקובץ על מנת שיוכל לבדוק האם הוא ווירוס (worm, trojan, etc.) מעבר לבדיקות הרגילות אשר עושה ה Sandbox ובאפשרותו לבחור כמה זמן הוא מעוניין שהקובץ יוכל לרוץ עד שהוא יקבל timeout אך אילו כל ההגדרות אשר ניתן לשנות.

בנוסף, ה Sandbox ממומש כ Virtual Machine ולא רץ על המחשב האמיתי של המשתמש, כלומר אם המשתמש רוצה לראות את תהליך הריצה ואת ההשפעות על המחשב בזמן אמת, אין לו אפשרות כזו. אך המשתמש כן מקבל log מפורט על כל הריצה אשר כולל:

* API function calls
* Memory dumps
* Loaded modules dumps
* Changes in file system, registry
* Network traffic
* Artifacts of exploit activity

**McAfee sandbox** – ה sandbox של McAfee הינו כלי אשר פותח על ידי חברת McAfee ומאפשר לבחון קבצים או אתרים חשודים בסביבה בטוחה ומבודדת. ה Sandbox מתוכנן למנוע מ malware להתרבות ולפגוע במחשב וברשת. המוצר מתאים למספר רב של סוגי קבצים ופלטפורמות:

פלטפורמות – Windows, Linux, MacOS,

סוגי קבצים – Executables, DLLs, DOC/DOCX, PDFs,URLs

ה Sandbox של McAfee יותר גמיש מ Kaspersky והוא נותן אפשרות להגדיר

* Timeout
* Forbidden protocols
* Specific OS to run on (if on a VM)
* Check if file might qualify as malware

בנוסף, לעומת Kaspersky, McAfee מציעים למשתמש בחירה האם הוא מעוניין להריץ את ה Sandbox ב VM או local.

ה log אשר McAfee מספקים תוכן מעט דל אשר מכיל את המידע הבא:

* Details about the security checks preforms on the file
* Any threats or vulnerabilities detected
* Basic behavior of the file – processes, network activity and changes made to the system or application

המערכת המוצעת במסגרת הפרויקט - **SafeBox qualifying factor**

הרכיב אשר מייחד את SafeBox מה sandbox של McAfee ו- Kaspersky מורכב משני חלקים.

1 – ה GUI הפשוט אשר מאפשר בקלות לראות את כל הפרמטרים האפשריים לריצה ומציג בצורה נוחה לניתוח את תוצאות הריצה. ב sandbox של שתי החברות האחרות המשתמש צריך להעמיק בהגדרות התוכנה על מנת להגדיר חלק מהפרמטרים שציינתי, או שהוא בכלל לא יהיה מודע אליהם. ב SafeBox התפריט הראשי כולל את פרמטרי הריצה והתוכנה מחייבת את המשתמש להגדיר אותם. גם מסך ניתוח התוצאות מציג את המידע בצורה נוחה שתאפשר למשתמש לבחון את התוצאות בקלות.

2 – פרמטרי ריצה גמישים ביותר – בשני ה sandbox האחרים הפרמטרים אשר המשתמש יכול להגדיר שטחיים ולא מאפשרים שליטה מספקת לטעמי. ב SafeBox כל היבט של הריצה ניתן לשינוי על ידי המשתמש, בין אם זה פונקציות ספציפיות לחסימה, הגנה על קבצים ספציפיים או תיקיות שלמות, ה drivers אשר יהיו בשימוש ומספר process/threads שמותר לקובץ לפתוח. SafeBox מאפשרת למשתמש לשנות את בדיקת הקובץ בצורה יותר low-level, בחירת פונקציות ספציפיות, drivers ספציפיים וכו' אך משתמש אשר ידע להשתמש בפרמטרים אלו טוב יוכל לחקור בצורה טובה יותר קבצים מאשר ב sandbox של Kaspersky או McAfee.

ניתן לסכם זאת בטבלה כדלקמן:

|  |  |  |  |
| --- | --- | --- | --- |
| תבחין | SafeBox | McAfee | Kaspersky |
| מערכות הפעלה עליהן פועל ה sandbox | Windows בלבד | Windows, Linux, MacOS | Windows, MacOS, IOS, android |
| סוגי קבצים עליו פועל ה sandbox | .exe של שפות כמו C# וC++ | .exe, .dll, DOC/DOCX, PDSs, URLs | .exe, .dll .NET object, PDFs, APK, URLs |
| על איזו מכונה רץ ה sandbox? | על אותה מכונה של ה client מה שנותן אפשרות לצפות ולעשות monitor על המתרחש בזמן הריצה | המשתמש מקבל אפשרות בין ריצה על VM מרוחק או ריצה local | הקובץ רץ בענן |
| בחירת security parameters | ה sandbox מאפשר להגדיר פונקציות ספציפיות לחסימה, קבצים להגן עליהם ובאילו drivers להשתמש ובאילו לא. | ה sandbox מאפשר להגדיר פרמטרים כלליים כמו מערכת ההפעלה עליה לרוץ, אילו network protocols לחסום, זמן timeout לריצת הקובץ. | אפשרות להגדיר זמן timeout לריצת הקובץ ולבצע בדיקות ספציפיות האם הקובץ עלול להיות סוג ספציפי של malware לדוגמה fishing, trojan, etc. |

|  |  |  |  |
| --- | --- | --- | --- |
| תבחין | SafeBox | McAfee | Kaspersky |
| הפרמטרים אשר עליהם מדווח ה sandbox | כל הקריאות לפונקציות שבוצעו, פרמטרים, זמני קריאה, קבצים אליהם ניגש הווירוס, הודעות אשר ניסה לשלוח והבקשות אשר חסם ה mini filter driver | פרטים לגבי ה security checks אשר נעשו על הקובץ, threats or variabilities detected, basic behavior of the file | API function calls, memory dumps, loaded modules dumps, changes in file system/registry, network activity, artifact of exploit activity |
| שימוש ב hooking על מנת לעצור קריאות API | האפשרות **קיימת** במערכת | | האפשרות **לא קיימת** במערכת |
| זיוף קיום משתמש על מנת שהוירוס לא יבין שהוא ב sandbox | אין צורך, כיוון שהוירוס נמצא על אותו מחשב כמו המשתמש הוא חושב שהוא הגיע למטרה שלו. | בריצה local אין צורך לדמות קיום משתמש, בריצה ב VM אין מנגנון לזיוף משתמש | קיים מנגנון זיוף משתמש. |
| סוגי malware אשר ה sandbox מזהה ומדווח | אין דיווח על סוג malware ספציפי, ה sandbox רק מודיע האם הקובץ הינו malware או לא. | יכול לזהות ולדווח האם הקובץ הוא – virus, worm, trojan, rootkit, fishing. | |
| אילו פעולות עושה ה sandbox עם התוצאות אשר גילה | כל ה sandboxes מקבלים מהשתמש הגדרות מה לבצע לקובץ אשר התגלה כחשוד –  Quarantine/Delete/Alert | | |

# 3.תוצר סופי

## 3.1 תיאור הפרויקט

חסר מבוא...

לא הבנתי לגמרי איזה מבוא צריך להיות פה

המערכת מכילה ממשק גרפי, אשר מאפשר למשתמש לבחור פרמטרים לריצה, להעלות קובץ לניתוח ומציג בסוף הריצה את התוצאות, ושרת אשר מקבל את קובץ ההרצה ומבצע את ה dynamic analysis בהתאם לפרמטרים שהוגדרו. אם השרת ממתין בתקשורת לנתונים אז הוא שרת, אחרת הוא לא. ש

הניתוח נשען על שלושה רכיבים, המאפשרים את איסוף המידע:

* DLL injector – אשר מריץ את הקובץ ומזריק אליו את ה hooking DLL
* Hooking DLL – רכיב אשר "מוזרק" לתוך ה process של קובץ ההרצה ואחראי לבצע user land hooking על פי פרמטרי הריצה.
* Mini filter driver – רץ ב kernel של מערכת ההפעלה ומבצע redirect ו log לבקשות לא מאושרות. לא הבנתי מה חלוקת העבודה הפונקציונלית בין 2 הרכיבים האחרונים

Useland hooking מטפל גם ב registry ו communication בעוד שכרגע mini filter driver מטפל רק במערכת הקבצים. אולי אם יהיה לי זמן אפתח גם עוד אחד שמטפל גם הוא ב communication. אבל בכל מקרה אם המשתמש רוצה דיווח על פונקציה ספציפית לדוגמה CreateFileA רק user land hooking מסוגל לעשות זאת.

הרצת המערכת מבוצעת בשלושה שלבים:

* בחירת פרמטרים לריצה
* הרצת הקובץ החשוד, ואיסוף מידע לגביו.
* הצגת תוצאות למשתמש

פרמטרים אפשריים לריצה – המערכת מציעה למשתמש גמישות בנושא ההגבלות אשר המשתמש מעוניין לכפות על קובץ ההרצה. היא מאפשרת לבחור אחת משלושת האפשרויות normal/warn/block אשר ישפיעו על ההרשאות שה sandbox מאפשר לקובץ. ברמת ריצה, ברמת גישה לקובץ / רגיסטרי...?

* Normal – המשתמש לא מעוניין במידע בנושא זה ולא דורש מהמערכת דיווח או חסימה.
* Warn – המשתמש מעוניין לדעת מתי נעשה שימוש במשאב זה (לדוגמה קובץ מסוים) או מתי נקראה פונקציה ספציפית אך אינו מעוניין שהמערכת תחסום את הגישה אליו.
* Block – המשתמש מעוניין כי המערכת תחסום כל גישה למשאב זה או קריאה לפונקציה שנבחרה ותודיע לו מתי נעשית בקשה אליו\קריאה אליה.

המערכת מאפשרת הגדרה של פרמטרים נוספים, לשליטה מיטבית על תהליך הניתוח –

* בחירת ה mini filter drivers אשר המשתמש מעוניין שידלקו. מבין אילו אופציות? File system and network
* בחירת קבצים ותיקיות (מה הבעיה) אשר המשתמש מעוניין שתיחסם כל גישה אליהם. כאשר מבצע המשתמש חסימה ליכולות כתיבה וקריאה מקבצים, קובץ הריצה החשוד יקבל גישה אליהם אך כל השינויים שהוא ינסה לעשות יימחקו, אם המשתמש מעוניין שקובץ הריצה לא יוכל לקבל גישה אל קובץ מסוים או תיקייה מסוימת הוא צריך לעשות זאת ידנית (אופציה זו מציעה גמישות רחבה יותר למשתמש ואפשרות לנתח את הווירוס בצורה יותר מעמיקה)
* הגדרת כמות process/threads שמותר לקובץ לפתוח.
* תוצאת ניתוח רצויה בהתאם לממצאים: quarantine/delete/warn

## 3.2 אלגוריתמים עיקריים

1. שליחת הקובץ ל server (?) והעלאת קבצים נדרשים

~~לאחר שה server מקבל את קובץ ההרצה מתרחשות מספר פעולות –~~ איך הוא מקבל?

לאחר שהקובץ נשלח ל server מה client מעל socket מתרחשות מספר פעולות -

* ה server יוצר process חדש עבור הווירוס
* ה process של הווירוס מושהה להמשך פעולות הכנה
* מתבצע תהליך ה DLL injection לתוך ה process של הווירוס
* העלאת mini filter drivers ~~נדרשים~~ אשר בחר המשתמש איזה?
* הכנת קבצים של ה sandbox (log, folder structure is correct) איזה? ובדיקת תקינות ריצה (אסור לקרות מצב בו ריצת ה sandbox נופלת כיוון שה host machine עלול להינזק). למה? ומה פתאום? אם המשתמש שיחק עם המבנה של הקבצים בתוך התיקיות של ה sandbox (כלומר הזיז חלק מהקבצים), ה sandbox עלול לא לתפקד כראוי. לדוגמה, אני מעלה את ה inlineHook DLL בכך שאני מתבסס על מקומו הרלטיבי לדוגמה ואם הוא לא יהיה שם ה user land hooking לא יעבוד וכך עלול להיגרם נזק למחשב.

1. בחירת פרמטרים והכנת הריצה

בזמן שה server (?) מקבל את קובץ הריצה המבוקש הוא גם מקבל מה client (?) את הפרמטרים לריצה. בהמשך, ה server מתחיל בתהליך העלאה של objects מה זה העלאה של אובייקט?! (אני מסביר בהמשך, כל hook מבוצע על ידי יצירת Hook object מתאים ולאחר מכן גם כתיבה למקומות שונים בזיכרון של ה process על מנת שה inline hook יהיה במקומו כאשר מתחיל הקובץ לרוץ) שונים בהתאם לפרמטרים אשר נבחרו על ידי המשתמש –

* נוצר Hook object (class פנימי) עבור כל פונקציה אשר המשתמש מבקש לחסום.
* אתחול של קבצים (?) להם ה server זקוק והעלאת DLLs דרושים למרחב הכתובות שלו.
* בדיקת תקינות של קבצים פנימיים של ה server, לדוגמא? (הסברתי למעלה)

רק לאחר שה server מסיים לבצע פעולות אלו הוא מאפשר ~~לווירוס~~ לקובץ החשוד להתחיל לרוץ לקובץ החשוד!

1. החדרת ה DLL אשר אחראי על ה hooking

על מנת שה hooking DLL אשר אחראי על ה userland hooking יעבוד כראוי נדרש להחדיר אותו ל process של ~~הווירוס~~ הקובץ החשוד. שכך, ה server ~~פותח~~ מריץ את התכנית הזדונית ב process מרוחק מתוך התכנית הראשית באמצעות winapi ואז מחדיר לתוך ה process המרוחק את ה DLL אשר אחראי על ה hooking באמצעות DLL injection.

תהליך החדרת ה DLL עובד כך –

* יצירת process חדש עבור הווירוס
* הקצאת זיכרון ב process של הווירוס
* כתיבת שם ה DLL אשר אחראי על ה hooking בתוך הזיכרון המוקצה
* יצירת thread מרוחק בתוך ה process החדש עם הפונקציה LoadLibraryA
* מסירת כתובת זיכרון המציינת את שם ה DLL לפונקציה זו כ parameter

1. Inline hooking WinAPI functions

תהליך זה מתבצע באמצעות שיטה שונה מעט משיטת הjmp-return (שיטת הטרמפולינה). ה server מקבל את הכתובת של הפונקציה לה נדרש hook ודורס את חמשת הבתים הראשונים שלה, ובמקומו מכניס פקודת push עם כתובת הפונקציה החדשה אליה נדרש לקפוץ, ולאחריה פקודת retn . באמצעות שיטה זו השרת מקבל שליטה על ריצת התכנית ומונע את הגישה לקריאות מערכת וגישה למשאבי המערכת. את כתובת הפונקציה השרת משיג באמצעות GetProcAddress – פקודה של WinAPI. תפסיק לאמר אני אני אני אני אני, שכתבתי הכל.

מה החסרון בשיטת הטרמפולינה? (הסיבה העיקרית היא שזוהי כבר שיטה מיושנת שווירוסים מסוגלים לגלות לפעמים, אני יכול להתחיל להסביר למה אבל זה הסבר טכני שאני לא חשבתי שמתאים פה)

1. הסתרת הפעילות ~~מהווירוס~~ מהקובץ החשוד כדי שיפעל באופן רגיל

ווירוסים רבים מבצעים כיום בדיקות האם הם נמצאים במצב של ריצה רגילה או ב Sandbox. המערכת מערימה על הווירוס ~~וגורם~~ וגורמת לו לחשוב כי הוא בתהליך של ריצה רגילה.

להלן מספר דוגמאות כיצד ה Sandbox מבצע פעולה זו –

* אם הווירוס מבצע פעולת כתיבה לקובץ, ה Sandbox יוצר קובץ temp אשר יימחק לאחר הריצה. כך הווירוס מקבל status code של הצלחה ורואה שפעולת הכתיבה בוצעה, אך בעצם לא ביצע פעולה אמיתית, קובץ ה temp יימחק מיד אחרי סיום הריצה. (ארחיב יותר באלגוריתם הבא)
* במקרה שהווירוס מנסה לשלוח הודעה על גבי socket, ה Sandbox מחזיר לו status code של הצלחה אך לא שולח את ההודעה וכאשר הוא מנסה לקבל הודעה מוחזר אליו buffer ריק.
* מה לגבי שליפת ערך מה registry?
* מה לגבי ביצוע subprocess ע"י הוירוס, למשל dir() לתיקיה אסורה, del/p לתיקיה ועוד...

(זה גם דברים שהמערכת חוסמת אבל אני רק נתתי שתי דוגמאות לא התחלתי לפרט על כל דבר שנחסם)

1. ניתוב לקבצים על פי פעולות קודמות

כאשר ~~הווירוס~~ קובץ ההרצה החשוד מנסה לכתוב לקובץ (אם יש לו הרשאות גישה כמובן), נוצר קובץ חדש אשר עליו יערכו השינויים הנדרשים, וכך לא ייפגעו קבצים של ה host machine. אך במקרה ובו הווירוס ירצה לקרוא את השינויים שהוא עשה ולהמשיך ריצה, ה Sandbox נדרש לנתב את המשך הריצה לקובץ temp המתאי? (יכול להיות תרחיש שיהיו כמה קבצי temp). סוגיה זו נפתרת בשיטה הבאה:

אני אני אני

* כאשר ה Sandbox מזהה בקשת גישה לקובץ ש read/write הוא בודק ב hashmap האם השם של הקובץ נמצא באחד מן ה keys. יפה!
* אם הבקשה הייתה read והקובץ לא אחד מן ה keys, הווירוס מקבל גישה לקרוא את הקובץ (כמובן אם מותר לו על פי פרמטרי הריצה).
* אם הבקשה הייתה write והקובץ לא אחד מן ה keys, נוצר קובץ temp חדש והווירוס מופנה לקובץ החדש להמשך פעולה
* במקרה שהקובץ כן נמצא באחד מן ה keys, ה value של אותו key יהיה שם ה temp אליו יופנה הווירוס. אבל מה התוכן המוחזר? למה ריק? (מה הכוונה? התוכן מה hashmap לא ריק הוא שם הקובץ החדש)

לא דיברת בכלל על כל נושא ה mini filter. לא ברור אילו אתה מיישם, איך, מה שיתוף האחריות בין שיטה זו ל inline hooking וכו'

לא ציינת כלום לגבי RAM?!

(אמרת שנוסיף דברים בספר הפרויקט, כפי שכתוב בלוח הזמנים אני עדיין לא מתעסק ב RAM וה mini filter לא מוכן עדיין, בספר הפרויקט או בתיקון שיבוא בהמשך אוסיף את שני התחומים האלה)

## 3.3 דרישות ואילוצי פתרון

האילוצים אשר חלים על הפרויקט –

* קובץ ההרצה המועבר לניתוח חייב להיות קובץ C++/C/C#/Python לא אמרת שגם פייתון? ואחרכך ציינת שגם C?!
* הפרויקט חייב לרוץ מעל מערכת הפעלה Windows 7 and above לאור השימוש הנרחב ב WinAPI איזה גירסא?
* ~~ה driver חייב להיות מותקן ומוכן לריצה לפני תחילת ה dynamic analysis איזה DRIVER? ומה מונע זאת ממך?~~
* קובץ ההרצה חייב להיות מקומפל לארכיטקטורה x86

דרישות הפרויקט –

* המערכת תידרש לתמוך במניעה של מספר מספק של פעולות ו WinAPI functions במספר תחומים שונים (registry, files, communication, RAM) . לא ציינת כלום לגבי RAM?!
* ~~המערכת חייבת לעמוד בדרישות פרמטרי הריצה ולא לאפשר לאף פעולה לא מאושרת להתבצע על מנת שה host machine לא יפגע~~. שמירה הדוקה על ה integrity של מחשה ה host במהלך הריצה, ללא כל השפעת ריצה.
* ~~המערכת תידרש להערים על הווירוס בכך שלא ידע כי הוא נמצא ב Sandbox מבודד ויחשוב כי הוא מתפקד על מחשב רגיל.~~ לא דרישה, זו פונקציונליות!

## 3.4 ממשקים למערכות חיצוניות

~~המערכת מתממשקת עם WinAPI רבות על מנת ביצוע פעולות שונות ביניהן שליטה על process, ביצוע hooking, כתיבת log וכו'.~~ זה לא ממשק! אין לך ממשקים, אלא אם רוצה לפנות ל VIRUSTOTAL וכדומה.

Google cloud ai platform – מודל AI של google אשר מקבל כ input מידע על קובץ (log לדוגמה) ומסיק האם הקובץ הינו malicious או לא. אני אשתמש במודל בשביל לתת למשתמש מידע האם הקובץ עלול להיות malware ואם כן איזה סוג.

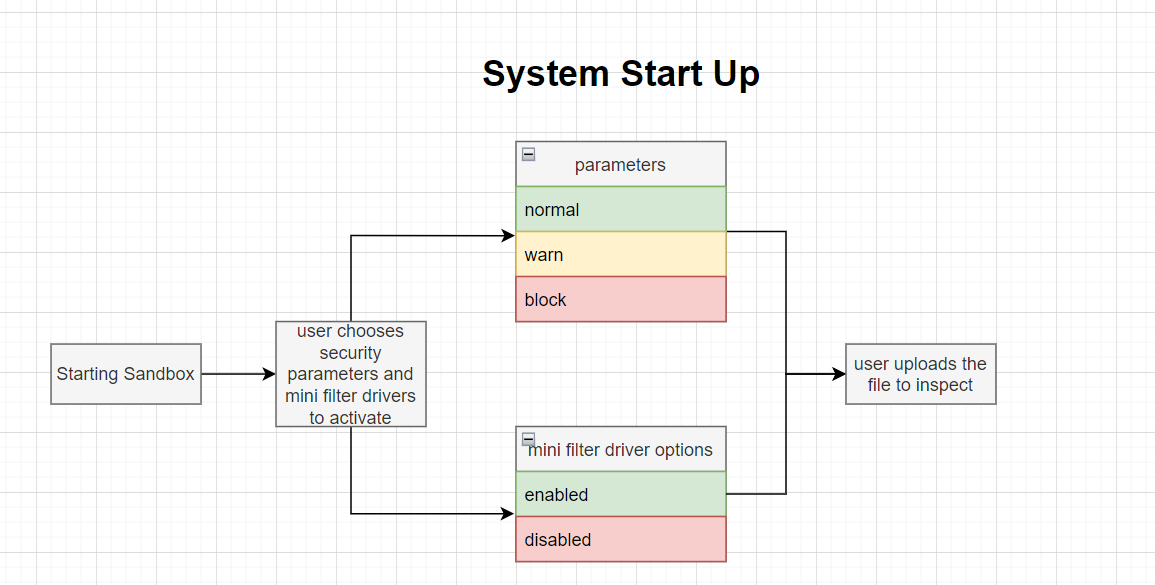
## 3.5 התייחסות לנושא האבטחה

המטרה המרכזית של הפרויקט היא לאבטח את ה host machine ולכן נושא האבטחה הינו top priority. המערכת חייבת להבטיח כי היא תגן על ה host לפי דרישת פרמטרי הריצה. במקרה ובו תיכשל המערכת למנוע מווירוס גישה למשאבי מערכת, יכול להיגרם נזק לכל המחשב ואף לכל הרשת בה הוא נמצא. לכן חייבת המערכת למנוע כל גישה למשאב אשר ציין המשתמש ולדווח לו על הנעשה.

כאשר משתמש יפתח את הקובץ הוא יידרש לספק שם משתמש וסיסמה. כך המערכת תהיה מוגנת בפני כניסת משתמשים לא מזוהים. למה זה ב FONT אחר?

## 3.6 ממשק משתמש

## 3.7 תרחישים עיקריים



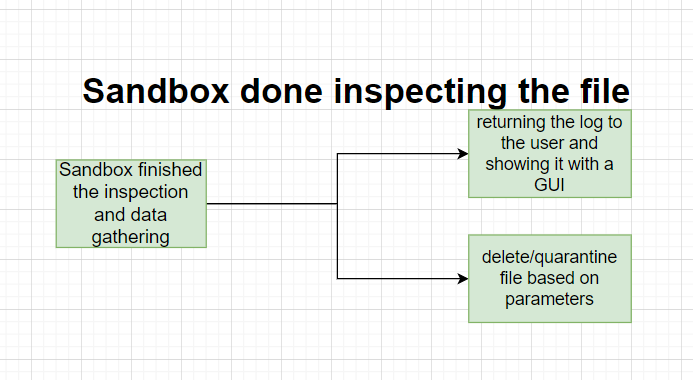
Diagram

Description automatically generated

Diagram

Description automatically generated with low confidenceרק LOG או גם ALERTS במהלך ריצה?

(לפי התכנון המשתמש לא מקבל alerts בזמן הריצה, לאחר שהקובץ סיים לרוץ הוא מקבל את התוצאות.)



לא נכון, ציינת גם אפשרות ל QUARNTINE? DELETE...ואחרכך איך אפשר להוציא מ QUARANTINE?

(תיקנתי, אני חושב שאציג ל user חלון בסוף הריצה האם הוא רוצה להוציא אותו מה quarantine או לא, בכל מקרה זה פשוט יהיה בתיקייה מוחבאת, אפשר לעשות רשימה של כל הקבצים שיש שם ב GUI ואם ה user מעוניין להוציא קובץ הוא ילך לרשימה ויבחר אותו)

# 4.תהליך כתיבת הפרויקט

## 4.1 תהליך הפרויקט

(אני לא בטוח אם הייתי צריך לכתוב את זה עכשיו או בסוף הפרויקט אז כתבתי בכל מקרה, אם צריך בסוף הפרויקט אז אפשר פשוט למחוק ולהוסיף אחר כך)

אבדוק בסוף

בתחילת הפרויקט לא כתבתי שורת קוד במשך חודשים. את הפרויקט התחלתי בחופש הגדול ובהתחלה לא פתחתי את עורך הקוד אפילו, הייתי צריך לבצע המון קריאה ומחקר.

תחילה חשבתי שאני הולך לבצע את הפרויקט בrust בכלל ולכן בחודש יולי השקעתי בעיקר בקריאת הספר הרשמי של rust שעל אף שאני לא עושה שימוש בשפה בסוף מאוד נהניתי ללמוד אותה ואני חושב שאני אחזור אליה בעתיד. אני לא משתמש בה מכיוון שהיא עדיין לא נתמכת בתכנות drivers וגם winapi, שני דברים שמרכיבים את רוב הפרויקט שלי.

ולאחר שלמדתי rust, את חודש אוגוסט השקעתי בקריאה משולבת של המדריך של learncpp.com והספר מערכות הפעלה של ברק גונן בשילוב מאמרים מזדמנים של digital whisper שמתקשרים לנושאים שלמדתי.

לאחר שסיימתי את תקופת הלימוד האינטנסיבית הגיע הזמן לבצע את החלק הקשה ביותר בפרויקט, לעשות hijack לתכנית ולהעביר את ההרצה לפונקציות שלי.

תחילה בניתי את הdll injector כדי שאוכל בכלל לקבל גישה לתכנית הזדונית. לאחר מכן בניתי את הDLL אשר מבצע את הinline hooking במהלך תקופה זו אשר כללה קריאה נרחבת של תהליכי hooking שונים מכיוון שלא ידעתי מההתחלה איזה hook אני הולך לממש והאפשרויות היו נרחבות, AIT, inline SSDT , לאחר שבחרתי את הhook והשיטה בה אבצע את הinjection התחלתי לעבוד על התכנית עם ממש ממשק console מצומצם.

לאחר מכן חילקתי את הפרויקט לקליינט ו"סרבר" כלומר במחשב אחד יהיה הממשק הגרפי שם ייבחר המשתמש את התכנית אותה הוא בודק, ובמחשב אחר תהיה סביבת ההרצה.

כעת אני משלב קריאה וכתיבה של הגנת RAM וmemory corruption ובאותו הזמן עושה mini filter drivers.

## 4.2 אתגרים ואופציות שונות למימוש

אבדוק בסוף

* מימוש הhooking , אני בחרתי באפשרות הinline hook מכיוון שהוא נראה כמו הhook הכי גמיש ונוח לביצוע לפי דעתי אך גם אפשרות הSSDT דיברה אלי ואני התלבטתי קשות באיזו שיטה לבחור. האתגר המרכזי במימוש הhooking הוא כמה שזה "קרוב לברזלים", אני נתקלתי בבעיות אשר קשורות לערך לא נכון בregister ספציפי או errors שקשורים לassertions שwindows מבצע כבדיקה שגרתית, קשה מאוד לבצע debug כשמשהו נכשל בצורה כזו ודורש קריאה מרובה וחרישה של האינטרנט למספר ימים על בעיה קטנה שעלולה לצוץ.
* למידת שפה חדשה לגמרי לבד – על אף שזהו היה תהליך מרתק ומהנה, הוא היה קשה מאוד. לעומת python/c# אשר נלמדו בכתה וחברי ללימודים יכולים לעזור לי במקרה ויש לי בעיות בהן. כאשר אתה עובד בשפה שונה ממש שנלמד ומה שעובדים איתה בכתה אין דרך לפנות לחברים \ למורה לעזרה ואתה תלוי ביכולת שלך ללמידה עצמית ופתירת בעיות לבד.

# 5.מרכיבי הפרויקט

## 5.1 תיחום הפרויקט

הפרויקט עוסק בתחומים הבאים:

* תקשורת – יש מעבר של קובץ ההרצה על גבי socket בין לקוח לשרת ולאחר מכן שליחת הlog
* אבטחת מידע –דגש על שמירה על ה host machine, חסימת קריאות למשאבי מערכת כמו קבצים, registry, RAM, communication באמצעות user land hooking ו mini filter driver
* מערכות הפעלה –ניהול processes, memory, threads and file system והתממשקות עם WinAPI. שימוש ב DLLs and DLL injection. יישום user land hooking and mini filter drivers
* תצוגה – GUI written in C# winforms
* מבנה נתונים – class עיקרי “Hook”, Functions enum, שימוש ב vectors and hash maps.
* ארכיטקטורת קוד – כתיבת classes מרובים וגם שימוש ב enums תוך שמירה על מודל ה MVC . על מנת להקל על תהליך כתיבת קוד תקין, קריא וגמיש.
* תיעוד – שימוש אקטיבי ב git עם comments על כל commit בנוסף לכתיבת ספר פרויקט

## 5.2 סביבת העבודה (טכנולוגיה)

שפות תכנות

שפות התכנות בהן אני משתמש בפרויקט הינן ++C\#C –

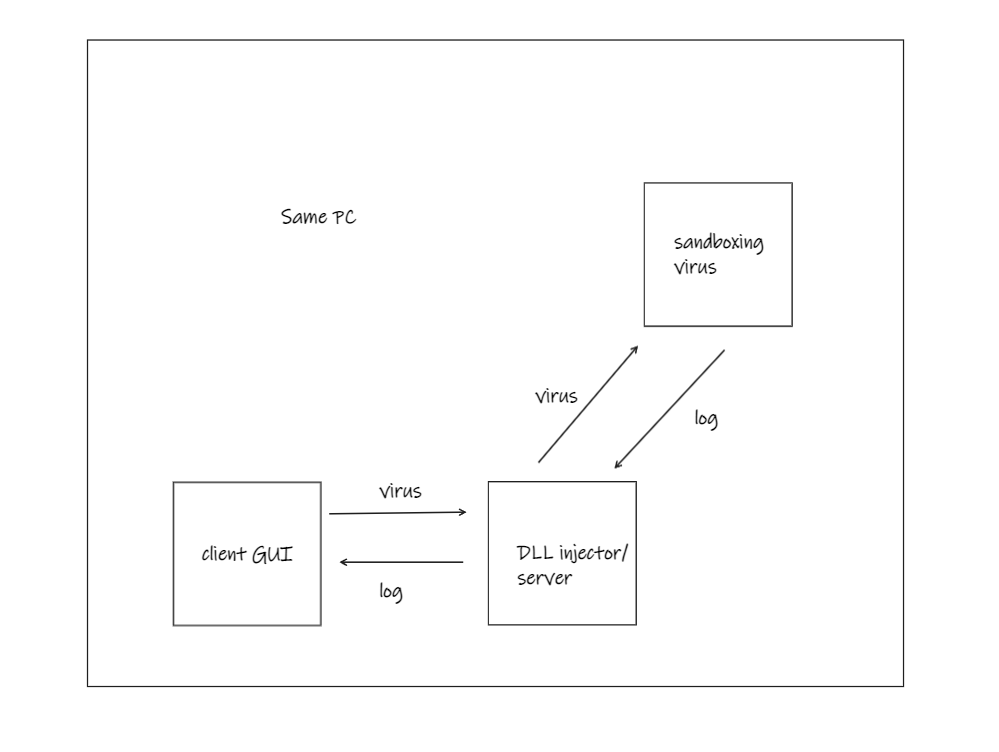
* #C – אני משתמש בשפה זאת לפיתוח ה GUI באמצעות winforms מכיוון שזהו כלי נוח וידידותי
* ++C – זוהי השפה העיקרית בה אני משתמש בפרויקט ובאמצעותה איישם את ה user land hooking, data gathering, mini filter driver מכיוון שזוהי שפה אשר נתמכת באמצעות windows על ידי winAPI וגם לפיתוח mini filter driver. בנוסף זוהי שפה בעלת יכולות גבוהות מאוד בכל הנוגע לניהול זיכרון ומאפשרת למפתח יכולות אשר לא נמצאות ברוב השפות האחרות כגון #C.

סביבת פיתוח

את כל פיתוח הקוד בשפות ++C ו- #C אני מבצע בסביבת העבודה Visual Studio 2022

## 5.3 מבט טופולוגי

אבדוק בסוף



## 5.4 מבנה נתונים

## 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 אשר כולל ביצוע user land hooking של winAPI functions על קובץ הרצה של ++C, החזרת log אשר כולל קריאות לפונקציות, פרמטרים וחותמת זמן. |
| דצמבר | End to end בסיסי אשר כולל הרצה של קובץ ומניעת פעולות registry, files, sockets, communication |
| ינואר | Mini filter driver אשר יודע למנוע גישה לקבצים ופעולות read/write. הצעת רעיון לחסימת privilege escalation |
| פברואר | חסימת פעולת privilege escalation ופיתוח גרפיקה ראשונית של בחירת פרמטרים. הצעה על חסימת RAM. |
| מרץ | תמיכה בקבצי python, חסימת RAM, אלגוריתם אשר מסוגל לבדוק האם קובץ הוא ווירוס או לא. |
| אפריל | שיפור גרפיקה של בחירת פרמטרים והוספת גרפיקה בסיסית להצגת תוצאות. |
| מאי | גרפיקה מרשימה להצגת תוצאות ריצה |

מפתח –

כל חודש נתתי לעצמי מטלה גדולה אשר תיקח את מרבית החודש ומטלה קטנה אשר תכין אותי לעבודה של החודש הבא. המטלה הגדולה עליה אשקיע את מרבית זמני מסומנת בירוק והמטלה אשר אצטרך לעשות בשביל העבודה של החודש הבא מסומנת בכתום. אני חושב שפשוט יותר להבין בצורה זו את המטרה של כל חודש והפוקוס שלו בזמן שעדיין אספיק להוסיף את כל מה שאני תכננתי.

## 7.2 אתגרים ותרומות אישיות

## 7.3 תוכנות

# 8.הוראות התקנה ותפעול

## 8.1 תצורה ודרישות קדם

## 8.2 התקנה

# 9.ביבליוגרפיה

ספרים עיקריים בהם נעזרתי –

* מערכות הפעלה – ברק גונן
* Windows kernel programming - Pavel Yosifovich
* Learncpp.com

מקורות מידע נוספים –

* <https://stackoverflow.com/>
* <https://cocomelonc.github.io/>
* winAPI msdn documentation
* <https://www.youtube.com/@TheCherno>
* <https://www.cyberark.com/>
* https://www.reddit.com/r/cpp\_questions/

# 10.נספחים