****

**עבודת גמר 5 יח"ל**

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

**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 3](#_Toc122813856)

[2.תיאוריה 4](#_Toc122813857)

[3.תוצר סופי 10](#_Toc122813858)

[4.תהליך כתיבת הפרויקט 19](#_Toc122813859)

[5.מרכיבי הפרויקט 20](#_Toc122813860)

[6.תסריטי בדיקה 23](#_Toc122813861)

[7.רפלקציה 24](#_Toc122813862)

[8.הוראות התקנה ותפעול 25](#_Toc122813863)

[9.ביבליוגרפיה 25](#_Toc122813864)

[10.נספחים 26](#_Toc122813865)

# 1.מבוא

## 1.1 נושא העבודה

נושא הפרויקט שלי הוא סביבת הרצה בטוחה Sandbox. התכנה **SafeBox** תקבל מהמשתמש קובץ הרצה שנוצר במגוון שפות (C++/C/C#/Python) ותעקוב אחר קריאות המערכת שלו לקבצים, registry, sockets , RAM ועוד, על מנת לגבש מסקנה האם מדובר בווירוס או בתכנה תמימה. במהלך הריצה המשתמש יוכל לראות את הקריאות לפונקציות השונות אשר ביצע הקובץ הנבחן. הסביבה תמומש באמצעות userland inline hooking, driver, mini filter driver wrapper.

## 1.2 מטרות מרכזיות

המטרות המרכזיות של הפרויקט הן:

* Dynamic Analysis של קובץ הרצה ללא חסימה במחשב. לא הבנתי.
* דיווח למשתמש על הפעולות אשר הקובץ מנסה לעשות תוך שמירה על בטחון המחשב.
* מתן גמישות בבחירת פרמטרי הריצה שהמשתמש מעוניין בהם, ודרך כך השגת גמישות בסריקת הקובץ
* ניתוח קבצים במגוון שפות (C++/C/C#/Python)

המטרות האישיות המרכזיות של הפרויקט הן:

* למידה 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 השליטה בתחום התקפות הסייבר תוך למידה על אמצעי הגנה רבים
* הבנה לעומק של 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 חולשות רבות. 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

בתור sandbox המטרה של הפרויקט היא למנוע מהקובץ הנבדק גישה לא מפוקחת למשאבי מערכת. וירוסים ניגשים למשאבי המערכת כמו sockets, registry, file system, RAM ועוד ומשתמשים בהם כדי להזיק למחשב היעד, לגנוב מידע חשוב ולפגום בפעילות המערכת בדרכים שונות. שכך, זוהי אחריותו של ה sandbox למנוע מהווירוס שהוא בוחן גישה למשאבים הללו, ולפקח על פעילותו באמצעותן.

לפני שאסביר על הדרכים השונות בהן sandbox פועל, צריך תחילה להבין מהו וירוס, כיצד הוא מתפשט ואיך הוא מבצע פעולות גניבת מידע \ כופר.

**וירוס** הינו קובץ ריצה רגיל אשר משתמש במשאבי המחשב בו הוא רץ, על מנת לבצע פעולות נוזקה, וזאת בין אם מדובר בזיכרון המחשב, בגישה מהמחשב לאינטרנט, או במערכת הקבצים של המחשב. הוירוס משתמש במשאבים הללו על מנת שיוכל להפיץ את עצמו למחשבים אחרים לא נכון! אתה מתבלבל בין וירוס לבין תולעת. להישאר חבוי במחשב הקיים או לבצע תקיפה. המשאבים המרכזיים בהם מתמקד ה sandbox הינם ה communication bandwith, RAM, file system, ו-registry. להלן מספר דוגמאות איך וירוס יכול להשתמש במשאבים הללו כדי לבצע פעולות נוזקה:

* באמצעות אמצעי תקשורת הווירוס יכול לקבל פקודות משרתcommand & control C)&C) או לשלוח מידע הנגנב מהמחשב.
* באמצעות גישה למערכת הקבצים הווירוס יכול לגנוב או להרוס מידע חשוב כמו פרטי בנק, תרשימים של חברה, מידע של עובדים וכו'.
* באמצעות גישה בלתי מוגבלת ל RAM הווירוס יכול לשבש את פעילות המחשב או השרת, ובעקיפין את הפעילות של חברות וחוות שרתים שלמות.
* באמצעות גישה ל registry הווירוס יכול להישאר במחשב חבוי (persistent) לתמיד, תוך גניבת מידע ושיבוש פעילות.

כל המשאבים אשר צוינו לעיל הינם משאבי יסוד אשר כל תכנה במחשב משתמשת בהם, ועל כן בחסימתם יכול ה sandbox להבטיח את הישארות הווירוס בתחום אשר הוגדר לו ללא פגיעה במחשב. בטרם שאסביר כיצד למנוע מהווירוס גישה למשאבים הללו, אפרט כיצד תכנה ניגשת למשאבים האלו.

הפלטפורמה שלי מתרכזת רק במערכות הפעלה מסוג windows. במערכות אלו הדרך הכי נפוצה לגשת למשאבי המערכת הינה באמצעות WinApi. ניקח לדוגמה את מערכת הקבצים. בכל שפה יש דרך שונה לפתוח קובץ:

* Python – with open(“tmp.txt”) as file
* C# - FileStream fs = File.Open(…)
* C++ - HANDLE hFile = CreateFile(…)

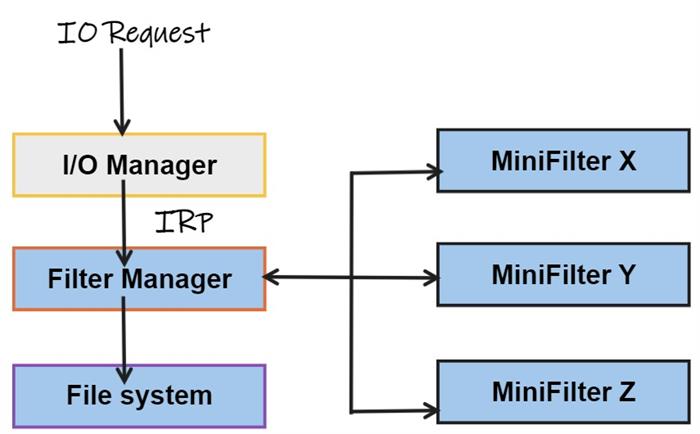
מרבית הפקודות לעיל מתורגמות לקריאות WinApi (או ישירות ל NT functions). על ידי האפשרות לפקח על קריאות אלה נפתחות למשתמש אפשרויות מחקר וזיהוי התנהגות הקובץ הנבדק. ניתן לפקח על קריאות ה WinApi במספר דרכים:

**Inline Hook WinApi Functions** - תהליך זה מתבצע באמצעות שיטת הטרמפולינה. בשיטה זו התכנה דורסת את חמשת הבתים הראשונים של הפונקציה, כדאי תרשים! ובמקומם מכניסה פקודת push עם כתובת הפונקציה החדשה אליה מעויניים לקפוץ, ולאחריה פקודת retn. באמצעות שיטה זו ניתן לקבל שליטה על ריצת התכנית ולמנוע את הגישה לקריאות מערכת ולמשאביה. את כתובת הפונקציה השרת ניתן להשיג באמצעות GetProcAddress – פקודת WinAPI. מאחר שרוב הקריאות במערכת windows למשאבים (לדוגמה למערכת הקבצים) מתורגמות ל WinApi functions, באמצעות ה hook ניתן לראות לא רק כי שתוכנה ביצעה בקשה לקובץ מסוים (לביצוע על ידי ה driver המתאים), אלא גם את הפונקציה הספציפית ב WinApi בה השתמש, לדוגמא למטרות של חיפוש חולשות. ביצוע ה inline hooking לצד השיטות האחרות כלליות יותר עליהן אסביר מאפשר לחוקרים לחקור את הווירוס לעומק ולגלות עליו מידע רב.

אך מה אם תכנה מצליחה לעקוף את מנגנון ה WinApi או ה inline hooking בין אם תוך קריאה לפונקציה אחרת או ביטול ה hook?לכן ישנה שכבת הגנה נוספת, קשה לאין ערוך לעקיפה, אשר ממומשת ב sandbox - שימוש ב drivers/mini filter wrappers.

**Driver** הינו קוד אשר רץ ב kernel אשר עונה על מספר מטרות. אחת המטרות הכי שכיחות שלו הינה לקשר בין רכיבי התכנה השונים לבין רכיבי חומרה. באמצעות ה driver ניתנת אפשרות לרכיבי התכנה לגשת ל API של ה driver ולהפעיל את הרכיב החומרתי. אך driver יכול למלא גם מטרה אחרת - לפקח על המערכת. כאשר קוד רץ ב kernel מחד, הוא צריך לעמוד בהרבה סטנדרטים שונים ומאידך, הוא גם מקבל הרשאות וגישות ליכולות אשר אין לקוד אשר רץ ב user mode. דוגמה אחת היא גישה לשירות שנקראית notify routines. לדוגמה הפונקציה PsSetCreateProcessNotifyRoutineEx אשר מאפשרת ל driver להכניס callback function שתקרא עבור כל process שנוצר. יכולות כאלו הן מה שמייחדות את שכבת ההגנה שמוסיף ה driver לעומת תכנית user mode רגילה.

drivers גם מקשרים בין רכיבי חומרה לתכנה. שניים מהמשאבים החשובים ביותר שה sandbox צריך לחסום גישה אליהם הם מערכת הקבצים ואמצעי התקשורת, שניהם בסופו של דבר הם רכיבי חומרה – כרטיס רשת ודיסק קשיח. נבחן כיצד ניתן להשתמש בעובדה הזו כדי לחזק את ההגנה של המערכת. ניקח לדוגמה את מערכת הקבצים. קובץ הינו רצף של bytes ששמורים בדיסק הקשיח, כלומר רכיב חומרה, אשר מנוהל על ידי driver שמקשר בינו לבין הצרכן. כל קריאה לקובץ לכתיבה, לקריאה או למחיקה מחייבת מעבר דרך ה driver האחראי על הקבצים. לכל driver קיימים **Mini filter driver** (MFTs) שונים, אשר רצים ב kernel ומוסיפים את עצמם למערכת הקריאות לצורך עיבוד הקריאה וחסימה במקרה הצורך. ה call back שה MFT יכול להוסיף צריך להיות מסוג post/pre, הכוונה היא האם ה call back יקרא אחרי שהבקשה נעשתה או לפני. לכל סוג יש יתרון שונה – אם מעוניינים שהקוד יבחן את הקריאה לפני שהיא תבוצע (הסוג היותר רלוונטי למקרה של sandbox) נדרש סוג pre. אך אם מעוניינים לשמור כל פעולה שקוראת במחשב למטרות logging ומחקר, נדרשת פונקציית post.



השימוש ב MFT מאפשר פיקוח הדוק יותר על אמצעי תקשורת או גישה לקבצים ותעבוד על כל סוג של פונקציה ולכן יעילה יותר לשימוש. קיימים גם **MFT wrappers** כלומר MFT אשר אדם פיתח עבורו API שמאפשר לפתח אפליקציות user mode בעלות היכולות של MFT (כמובן שצריך הרשאות admin במערכת כדי להשתמש בהן), לדוגמה pydivert, ספרייה מאוד פופולרית, אשר משתמשת ב MFT. באמצעות wrappers אלו ניתן לפתח הגנות בלתי עקיפות עם הנוחות של פיתוח user mode ובלי הקושי של לפתח ל kernel.

## 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 המובילים 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 לבצע בדיקות ספציפיות על הקובץ כדי לבדוק חשד לנוזקה (worm, trojan, etc.) מעבר לבדיקות הרגילות אשר עושה ה Sandbox שלי 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 פיזית על מחשב 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 and registry
* Network traffic
* Artifacts of exploit activity

**McAfee sandbox** – ה sandbox של McAfee מאפשר 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 a malware

בנוסף, לעומת Kaspersky, McAfee מציעים למשתמש בחירה האם הוא מעוניין להריץ את ה Sandbox ב VM או local. ה log אשר McAfee מספק בעל תוכן מעט דל ומכיל 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** באמת!

היתרונות אשר מייחדים את SafeBox מה sandbox של McAfee ו- Kaspersky מתרכזים בהיבטי השליטה ובהירות התצוגה, ומכאן נגזר 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. בנוסף, 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 threads / processes 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++, קבצי python | .exe, .dll, DOC/DOCX, PDSs, URLs | .exe, .dll .NET object, PDFs, APK, URLs |

|  |  |  |  |
| --- | --- | --- | --- |
| תבחין | SafeBox | McAfee | Kaspersky |
| 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. |
| 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 שהווירוס לא יבין שהוא ב sandbox | אין צורך, כיוון שהווירוס 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. תקציר מהעמוד הראשון

המערכת מכילה ממשק גרפי, אשר מאפשר למשתמש לבחור פרמטרים לריצה, להעלות קובץ לניתוח ומציג במהלך הריצה את התוצאות. לאחר בחירת הקובץ והפרמטרים, מופעלים פונקציות שונות אשר מבצעות את החסימות (כל פונקציה וההיבט אשר צריך לחסום) ומדווחים חזרה לממשק שמציג את הממצאים למשתמש. חלבי ביותר.

הרכיבים המרכזיים במערכת הינם – אם מדובר בשרת, מצוין, אחרת, מה זה קובץ?!

* הממשק הגרפי אשר מציג למשתמש נתונים ומאפשר לו לבחור פרמטרים ולהעלות קבצים.
* קובץ controller מרכזי אשר מריץ את שאר הקבצים ה – "collectors" אשר אוספים מידע על הקובץ הנבדק בזמן שחוסמים את פעולותיו.
* ה – "collectors" כמו DLL ?? שאחראי על hooking, קובץ .sys של ה driver, קבצים המתממשקים עם mini filter driver wrappers. בלאגן אטומי.

הרצת המערכת מבוצעת בשלושה שלבים: בחירת פרמטרים לריצה, הרצת הקובץ החשוד, איסוף מידע לגביו, והצגת תוצאות למשתמש

פרמטרים אפשריים לריצה – המערכת מציעה למשתמש גמישות בנושא ההגבלות אשר המשתמש מעוניין לכפות על קובץ ההרצה. היא מאפשרת לבחור אחת משלושה אפשרויות לגבי כל תחום שהיא מאפשרת לחסום (normal warn, (block, :

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

המשתמש צריך לבחור אחת מהרשאות אלה לגבי כל אחד מהפרמטרים הבאים –

* פונקציית WinApi ספציפית
* משפחת פונקציות WinApi (file system, registry, communication)
* Drivers / mini filter drivers wrappers

מה הדיווח? מה קורה לקובץ לאחר הריצה? לא הגדרת SERVER אבל אתה משתמש במושג הזה בהמשך.

## 3.2 אלגוריתמים עיקריים

1. בחירת פרמטרים והכנת הריצה

לפני שאפשר להריץ את הקובץ הנבדק, יש לבחור את הפרמטרים הדרושים לריצה ולהעביר אותם לקבצים האחראים על החסימה. לאחר שהמשתמש סיים לבחור את הפרמטרים שהוא רוצה בממשק הגרפי, ומחליט להתחיל את הריצה, רץ הקובץ controller המרכזי אשר מתחיל את כל הקבצים הדרושים בשביל ה sandbox ומעביר להם את הפרמטרים שנוגעים לתחום שלהם. עברית / רוסית? תחליט!

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 הקובץ החשוד. שכך, ה server 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. תיקנתי לך מאוד בסבלנות את פרק 1+2, מפה אתה אחראי!! דאג לעברית שוטפת, לא סיפור, לא חידות, ולא חזרות!

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

כמפורט בפרק התיאוריה, ויש להרחיב

מה לגבי יישום WINAPI למשפחות?

1. ניתוב לקבצים על פי פעולות קודמות

כאשר 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 המתאים (יכול להיות תרחיש שיהיו כמה קבצי temp). סוגיה זו נפתרת בשיטה הבאה:

* כאשר ה Sandbox מזהה בקשת גישה לקובץ של read/write הוא בודק ב hashmap האם השם של הקובץ נמצא באחד מן ה keys.
* 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.

1. מציאת ה PID של port ספציפי

כדי למנוע שליחת הודעות תקשורת (ומה בדבר קבלה?) התכנה של ה communication mini filter driver wrapper צריכה לדעת עבור כל port מהו ה PID אשר מתקשר אליו, כדי שתוכל לדעת האם לחסום את הפקטה או לא. כדי לעשות משימה זו אני משתמש בשילוב של pydivert ו windivert - אני עושה זאת כיוון שב pydivert וב python בכללי מאוד נוח לתכנת פעולות מסובכות של string manipulation או logging אך ב ++C יש את היכולת לנתר כל event שקשור ל communication - יכולת אשר חסרה ב pydivert. אני משתמש ב windivert על מנת שאוכל לדעת מתי PID ספציפי יוצר socket חדש או מבצע פעולות של bind, accept, connect וכו' ובאמצעות pydivert אני מבצע את החסימה עצמה ושולח את ה log לממשק הגרפי.

איך אתה משלב python ו C++?

1. חסימת התכנית משימוש יתר ב RAM

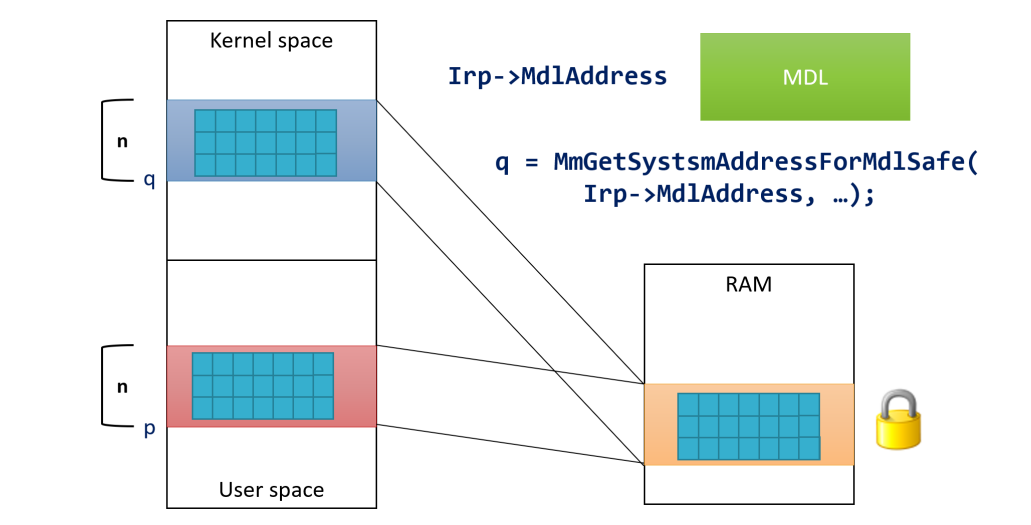
שימוש ב WinApi על מנת למצוא את כמות ה RAM (כולל DLLs) אשר תכנית מסוימת משתמשת בו, הרצה ב thread ובדיקה כל כמה מילי שניות האם הקובץ הנבדק עבר את כמות הזיכרון המותרת לו, ואם כן אני ביצוע terminate ובכך שחרור המקום ודיווח על כך למשתמש.

1. Monitor על יצירת process/threads של הקובץ הנבדק

באמצעות ה driver והפונקציות PsSetCreateProcessNotifyRoutineEx ו- PsSetCreateThreadNotifyRoutine - הרצת call back function אשר בודקת את ה PID של ה process שיזם את התהליך. אם זהו ה process אשר נבדק על ידי ה sandbox ה driver לא מאשר את הפעולה (בהתאם לפרמטרי הריצה) וחוזרת שגיאה לקובץ הנבדק.

ה driver צריך גם לשלוח את המידע חזרה לממשק אשר רץ ב user mode והיות וה driver רץ בקרנל אני עושה זאת באמצעות Direct IO buffer כלומר buffer שנמצא ב system memory אשר יש גישה לתכניות גם ב kernel וגם ב user mode. אם אתה כל הזמן אומר כלומר, אז אתה לא מסביר טוב, תסביר כמו שצריך בפעם הראשונה!

להלן תרשים שממחיש את הפעולה –



## 3.3 דרישות ואילוצי פתרון

האילוצים אשר חלים על הפרויקט –

* קובץ ההרצה המועבר לניתוח חייב להיות קובץ C++/C/C#/Python
* הפרויקט חייב לרוץ מעל מערכת הפעלה Windows 7 and above לאור השימוש הנרחב ב WinAPI
* 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) .
* שמירה הדוקה על ה integrity של מחשב ה host במהלך הריצה, ללא כל השפעת ריצה.
* מה עוד?

## 3.4 ממשקים למערכות חיצוניות

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 כדי לתת למשתמש מידע האם הקובץ עלול להיות malware ואם כן איזה סוג. למה?

MSDN documentation – השתמשתי בקובץ crawler ועברתי על כל ה documentation של MSDN כדי להוריד משם את ה function prototypes של כל ה WinApi functions. באמצעות המידע הזה ביצעתי inline hook לכל הפונקציות WinApi הקיימות.

## 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 המערכת חייבת למנוע כל גישה למשאב אשר ציין המשתמש ולדווח לו על הנעשה. הרבה חזרתיות!

## 3.6 ממשק משתמש

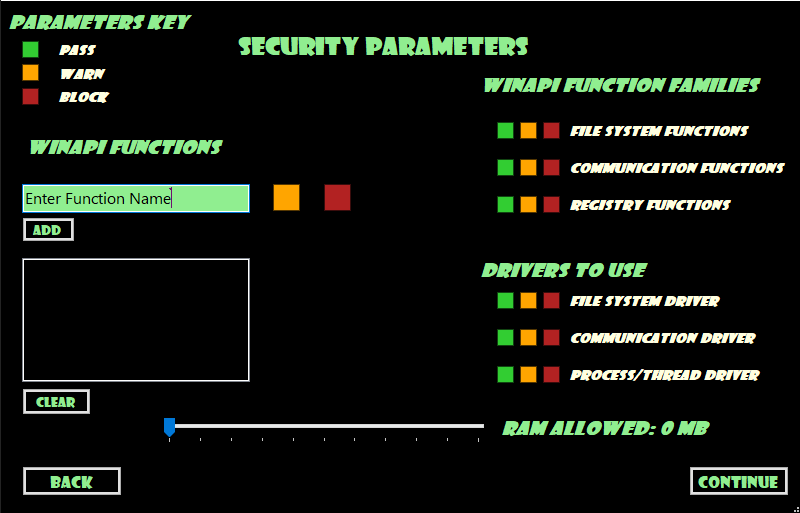
מסך הפתיחה אשר יוצג למשתמש כאשר מאתחל את התכנה –

הוא יתבקש לגרור קובץ ל drag and drop box ורק אז יוכל להמשיך למסך הבא. ניסוח

אחרי שיגרור קובץ, יופיע שמו בצד שמאל של המסך



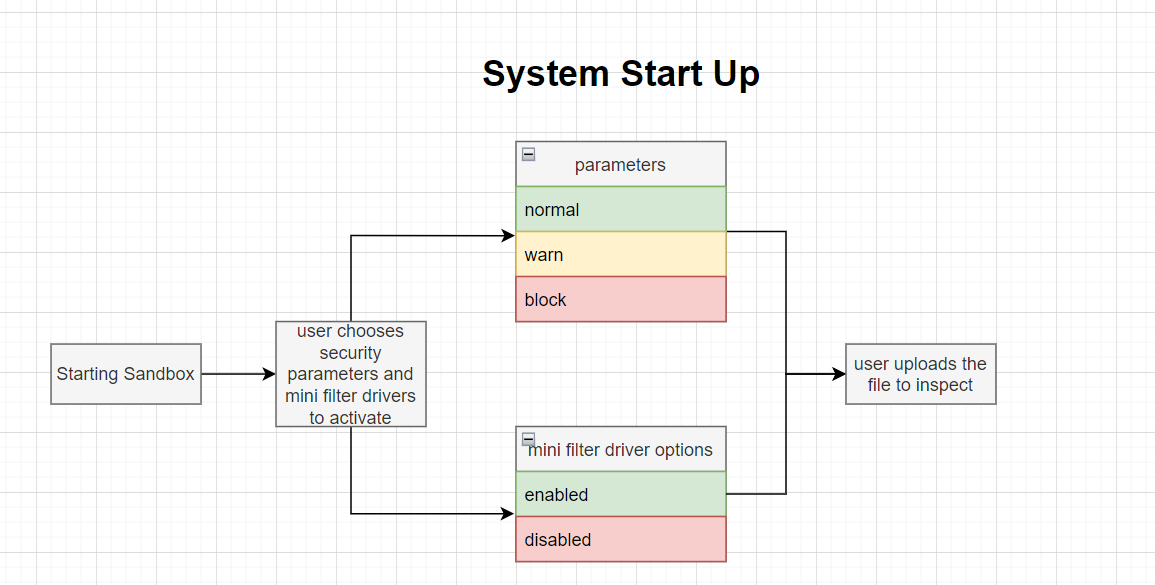
לאחר שיבחר המשתמש קובץ. יוצג מלפניו מסך הפרמטרים בו יבחר את הפרמטרים הדרושים לריצה הנוכחית – נורא ואיום!



לאחר מכן יבוא מסך התוצאות אשר עדיין לא מוכן. ☹

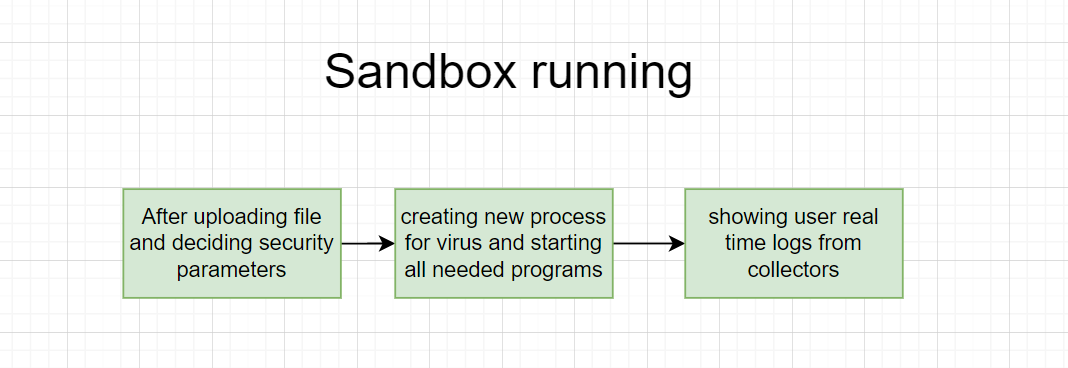
מה אתה אומר?!

## 3.7 תרחישים עיקריים

לכל תרחיש שם והסבר קצר. הפונט שלך לא נכון!

Diagram

Description automatically generated



# 4.תהליך כתיבת הפרויקט

## 4.1 תהליך הפרויקט

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 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. ניסוח 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. די! רחמים! הניסוח נוראי!

## 4.2 אתגרים ואופציות שונות למימוש

אני אפילו לא מתכוון לבדוק! מקווה שאתה יודע למה.

מימוש ה hooking : 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 ומה עם פייתון?

* #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.
* Python – מאפשרת נוחות רבה בנוגע ל string manipulation, logging ופעולות רבות שיהיו קשות מאוד ב ++C.

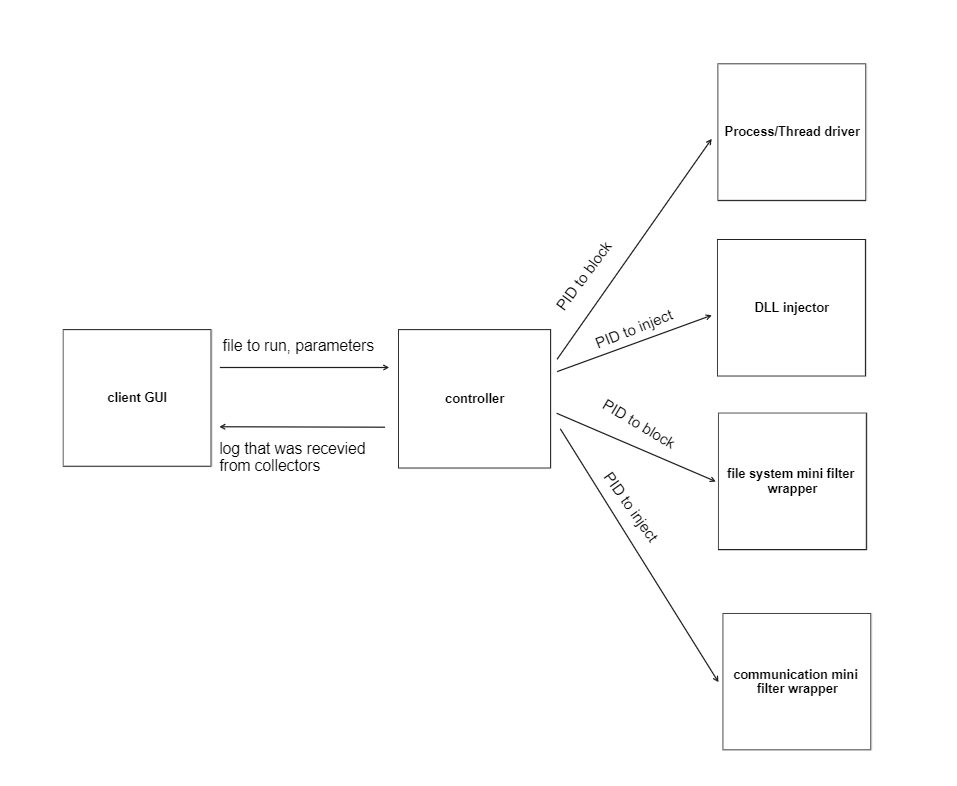
סביבת פיתוח

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

את הפיתוח ב python אני עושה ב Visual Studio Code

## 5.3 מבט טופולוגי

אבדוק בסוף



הסבר!

## 5.4 מבנה נתונים

תקן את הפונט! ותפסיק עם האני?!

הפרויקט נכתב על גבי מספר רב של קבצים אשר אינם חולקים מבני נתונים דומים בגלל מחסום השפות הרבות אשר השתמשתי בהן ולכן דאגתי להשתמש במבני נתונים אשר קיימים בכל השפות, אם אצור מחלקה ב ++C לא אוכל להשתמש בה גם ב python ולכן המבני נתונים שלי כוללים רק מבני נתונים אשר קיימים בכל השפות. עם זאת כן יש לי מספר משתנים בעלי חשיבות רבה –

**Proces/Thread driver** – ב driver אני שומר את כל הנתונים שלי ב linked list וכאשר ה controller מנסה לקרוא מה driver אני מעביר לו את ה nodes של הרשימה המקושרת אחד אחד, וכך אני מצליח לשמור את כל ההתראות שה driver מעוניין להעביר בלי לאבד אותן אך גם בצורה יעילה.

**Communication mini filter wrapper** – בתוכנה זו אני משלב קוד של ++C ו python. הקוד ++C דואג לעשות monitor לכל event שקשור ל sockets שקורה במערכת ומדווח על האירועים הרלוונטים, בקוד ה python התכנה עושה filter לכל הפקטות המגיעות ברשת. שני התוכנות הללו מתקשרות באמצעות sockets ומעבירות אחת לשנייה רשימה אשר מכילה את כל ה ports אשר התכנה הנבדקת פתחה ומקשיבה להם. כך אני לא מפספס שום פקטה אשר מנסה הקוד הנבדק להעביר (אם היה ניסיון לשלוח פקטה ומיד לסגור את ה socket, ללא קוד ה ++C הפקטה הייתה מצליחה לעבור את הסינ

**Inline hook DLL**- ב DLL אשר מוחדר לתכנה הנבדקת ישנה מחלקה Hook, מחלקה זו מכילה את הפונקציות הדרושות בשביל לבצע את ה inline hook בצורה גנרית וגם את ה log.

**Generic log** – מחלקה אשר יודעת לקבל כל משתנה (ב ++C) ולשלוח את ה log ל GUI. יש בה פונקציות שמתרגמות את כל סוגי המשתנים ל strings שיוכלו להיות מוצגות למשתמש.

## 5.5 מסד נתונים

אין מסד נתונים – מה אתה אומר!? תחשוב שוב.

## 5.6 מבט מודולרי

פונט!

הקבצים המרכזיים בפרויקט – זה לא הקבצים! זה ה MVC

Code generators- קבצים אשר עושים scraping לאתרים שונים ( MSDN לדוגמה), אוספים מידע וממנו מכינים קוד ++C אשר יבצע את תהליך ה hooking, ה code generators כתובים ב python.

injectDll – הקובץ אשר אחראי על החדרת ה DLL שמבצע את ה hooking ל process של הקובץ הנבדק. הוא מקבל מה controller את ה PID אליו יש להחדיר את ה DLL

inlineHook – ה DLL אשר מבצע את ה inline hook, DLL זה אחראי לחסום בצורה גנרית את כל פונקציות ה WinApi functions ולדווח עליהן בחזרה למשתמש.

packetDivertor – קבצי ה python ו ++C אשר חוסמים ומדווחים על כל socket event וכל ניסיון של שליחת הודעה בזמן שלא מאפשרים מעבר של שום פקטה (ללא השילוב כן יש דרכים לעבור דרך הסינון).

ProcessThreadDriver – ה driver אשר חוסם יצירת process/threads מהתכנה הנבדקת ומדווח על כך בחזרה למשתמש.

## 5.7 פירוט מודלים עיקריים

הערה לערן – אין לי מחלקות\מודלים עיקריים, לא בטוח מה לעשות פה. מצחיק ואפילו משעשע! תסתכל אצל ישעיהו.

# 6.תסריטי בדיקה

## 6.1 דגשים בבדיקה

* לחסום את כל מה שהמשתמש ביקש ללא פספוס של בקשות\ פעולות
* דיווח נכון של הפרמטרים וסיפוק מידע על הפעולות השונות
* תמיכה בקבצי python,C#,C++,C
* השוואה מול SANDBOX אחר?!

פונט!

## 6.2 תסריטי בדיקה עיקריים

* בדיקה שהקלטים שהתכנה מקבלת תקינים – במסך הפרמטרים ובקובץ שנבחר
* בדיקה עבור קומבינציות פרמטרים שונות שהכל מדווח וששום דבר לא מתפספס
* בדיקה שכל סוגי השפות הנתמכות עובדים

חסר! פונט!

# 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.

פונט! הרעיון של צבעים לא טוב. מצידי תכניס 2 BULLETS בכל שורה

## 7.2 אתגרים ותרומות אישיות

פונט!

בתחילת כתיבת הפרויקט לא היה לי שום ידע ב ++C, מערכות הפעלה, וירוסים, Winforms וכתיבת drivers. מצב זה גרם לכך שהייתי צריך להתחיל את הפרויקט מוקדם מאוד ולעבוד עליו קשה מאוד בכך שהייתי צריך ללמוד לצד העבודה בכל שלבי הפרויקט. בכל רגע עד שסיימתי אני המשכתי ללמוד דרכים שונות למימוש הפרויקט והעמקתי יותר באיך המחשב עובד.

נדרשתי לקרוא ספרים רבים ולהעמיק בשפות ובמושגים הרבה יותר ממה שנלמד בכתה על מנת שאוכל לממש את הפרויקט ואני חושב שבסופו חיזקתי את הידע שלי בכל הנושאים אותם ציינתי ואני מרגיש הרבה יותר בנוח לכתוב בשפת ++C, לכתוב ממשקים גרפיים ב winforms ואני הרחבתי את הידע שלי במערכות הפעלה וב kernel בצורה משמעותית.

## 7.3 תובנות

פונט!

בתחילת הפרויקט כשרק בחרתי את הנושא יחסית פחדתי להתחיל, עברית. טורקית. תחליט! זה היה נשמע נושא גדול מאוד שאין סיכוי שאוכל לעשות. אך לאחר שהתחלתי ללמוד הבנתי שבעצם זה כן משימה אפשרית וקראתי עוד ועוד ועוד עד שהמשימה הזאת באמת הפכה לאפשרית. כמה פעמים ?! לא ספרתי!

אני חושב שתהליך הזמן הארוך שבו עבדתי על הפרויקט היה עוד אתגר שהייתי צריך לעבור, באמצע השנה חשתי burnout ואבדה לי המוטיבציה להמשיך ללמוד ולעבוד על הפרויקט, אך באמצעות המנחה והחברים שדחפו אותי המשכתי לעבוד קשה בשביל לספק את התוצר הסופי.

למדתי על עצמי כי אין משימה בלתי אפשרית בתחום המחשבים, יש משימה שאתה לא רוצה מספיק להשלים. למדתי כי עם מספיק מחקר והשקעה כל דבר שאדם מנסה לעשות במחשבים אפשרי בהחלט, עובדה שגורמת לי לאהוב את התחום עוד יותר. בנוסף למדתי כי גם חשוב לקחת הפסקות כאשר עובדים על פרויקטים גדולי ממדים כמו זה, אני נתתי 150% מעצמי בחודשים הראשונים אך זה פגע בי בהמשך הדרך ואני חושב שתכנון זמן נכון ועבודה מתונה זה שני דברים אשר חשובים ללמוד ולנהל בדיוק כמו את היבטי התכנה של הפרויקט.

# 8.הוראות התקנה ותפעול

## 8.1 תצורה ודרישות קדם

פונט!

במחשב אשר מריץ את ה sandbox נדרשים –

* Windows 10 ומעלה
* .net framework
* Windows Forms
* Python 3.10 ומעלה
* ספריות – pydivert, psutil

## 8.2 התקנה

פונט!

לאחר התקנת כל המודלים והתוכנות הנדרשות יש להפעיל את ה sandbox בהרשאות מנהל ואז ניתן להתחיל להשתמש בו.

# 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