****

**עבודת גמר 5 יח"ל**

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

**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:** 24.11.22

תוכן עניינים

Contents

[מבוא 3](#_Toc120988316)

[נושא העבודה 3](#_Toc120988317)

[מטרות מרכזיות 3](#_Toc120988318)

[רציונל 3](#_Toc120988319)

[קישור לחומר הנלמד 3](#_Toc120988320)

[תיאוריה 4](#_Toc120988321)

[2.1 תיאוריה 4](#_Toc120988322)

[2.2 מוצרים קיימים בשוק 4](#_Toc120988323)

[תוצר סופי 6](#_Toc120988324)

[תיאור הפרויקט 6](#_Toc120988325)

[דרישות ואילוצי פתרון 6](#_Toc120988326)

[אלגוריתמים עיקריים 6](#_Toc120988327)

[תסריטים עיקריים 7](#_Toc120988328)

[תהליך כתיבת הפרויקט 8](#_Toc120988329)

[תהליך הפרויקט 8](#_Toc120988330)

[אתגרים ואופציות שונות למימוש 9](#_Toc120988331)

[מרכיבי הפרויקט 9](#_Toc120988332)

[תיחום הפרויקט 9](#_Toc120988333)

[סביבת העבודה (טכנולוגיה) 10](#_Toc120988334)

[מבט טופולוגי 10](#_Toc120988335)

# מבוא

## נושא העבודה

נושא הפרויקט שלי הוא סביבת הרצה בטוחה – sandbox. במסגרת הפרויקט סביבת ההרצה הבטוחה תדע לקבל קובץ מקומפל של C++ או C# ולעקוב אחר קריאות המערכת שלו לקבצים, רגיסטרי, סוקטים, RAM ועוד. הסביבה תמומש באמצעות userland inline hooking ו - mini filter drivers. הhooking נעשים באמצעות פקודות win32/winapi ובאמצעות השפה C++.

## מטרות מרכזיות

המטרה של סביבת ההרצה היא לבדוק את קריאות המערכת של הקובץ הexecutable ולפקח שהוא אינו פוגע במחשב. הsandbox ידע לעצור קריאות למקומות אשר הקובץ לא אמור לגשת אליהם כגון registry וקבצים חשובים של windows. בנוסף ישלח למשתמש log של איזה קריאות נעשו, עם אלו פרמטרים ומתי.

## רציונל

המוטיבציה שלי לפיתוח הפרויקט נבעה מהרצון שלי ללמוד את נושא הlow level ומערכות ההפעלה לעומק, יותר מאשר שנלמד בכתה. במסגרת הפרויקט הייתי צריך ללמוד את שפת C++ אשר היה לי אפס ניסיון בה וללמוד את נושא מערכות ההפעלה ובפרט windows. אני מוצא את שני הנושאים האלה מעניינים מאוד ואחרי שנה שלמה שבה עסקתי בתקשורת רציתי להשיג הבנה רחבה גם בנושא זה.

## קישור לחומר הנלמד

על אף שנושא הC++ לא נלמד בכתה, אני עושה שימוש נרחב בכל הנלמד בכתה י"ב, בתחום מערכות ההפעלה כגון PE,DLL,ntdll וכו', בונסף, בתחום הC# winforms אשר באמצעות כלי זה אפתח את הGUI של הפרויקט.

ה source control נעשה ע"י git ושמירה בענן ע"י github אשר נלמדו במהלך כיתה י"א.

אני עושה שימוש קל בנושא התקשורת דרך sockets אך לא באמצעות python כפי שנלמד בכיתה י"א.

# תיאוריה

## תיאוריה

## מוצרים קיימים בשוק

יש בשוק מוצרי sandbox רבים אשר כל אחד מהם עובד בצורה שונה, מיושם בצורה שונה ופועל על סוגי קבצים שונים. אני אשווה את תכנת ה sandbox שלי לתכנה של חברת אבטחת המידע Kaspersky. אני אשווה את עצמי למוצר של חברה זו מכיוון שמחד גיסא זוהי חברה מאוד מוכרת, בעלת שם בעולם הסייבר ופיתחה עוד מוצרים מעולים מעבר ל sandbox אשר בעלי שם עולמי, מאידך גיסא יש לי היכרות רבה עם המוצר שלהם כיוון שבשלבים של חקר הפרויקט כאשר רציתי להחליט באיזה נושא לבחור, נחשפתי לעולם ה sandboxing דרכם ואני מכיר את המוצר שלהם יחסית טוב.

Kaspersky בחרו ליישם את ה sandbox שלהם בצורה שונה משלי והם בחרו להביא למשתמש פחות שליטה על התכנה, אציג את ההבדלים בהמשך, אך גם הגיעו לרמה מאוד גבוהה של ניתוח מעבר לגבולות הפרויקט שלי. היתרון ההכי גדול שלהם לעומתי הוא שהם תומכים באין ספור שפות, סוגי קבצים, פלטפורמות ועוד. מגבלות אלו איני מסגול לעבור משני סיבות: האחת היא כי אני מפתח יחיד אשר לומד ומיישם את החומר בו זמנית לעומת צוות עצום של מפתחים אשר בטח עבדו על מוצר זה זמן רב, הסיבה השנייה היא שאני לומד את תחום ה windows ויודע לעסוק בתחום הסייבר רק בו, אין לי ידע בתחום ה android או mac. אך איני אומר שה sandbox שלהם בהכרח יותר טוב, אני לעומת Kaspersky מעניק למשתמש הרבה יותר שליטה על ה sandbox, המשתמש יכול להחליט אילו דברים לחסום, על מה לדווח ומה לאפשר, הוא יכול לקרוא את ה LOG האמיתי שיצא מה sandbox ולא רק נתונים אשר התכנה החליטה על דעת עצמה שרלוונטיים אליו.

טבלת השוואה:

|  |  |  |
| --- | --- | --- |
| תבחין | אני | Kaspersky |
| מערכות הפעלה עליהן פועל ה sandbox | windows בלבד | windows ו android |
| סוגי קבצים עליו פועל ה sandbox | .exe של שפות כמו C# וC++ | .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 | על VM מרוחק אשר ל client אין קשר אליו. |
| האם יש אפשרות להגיד פרמטרים לריצה? | ניתן להגיד ל sandbox איזו דברים המשתמש מעוניין לחסום, לדווח ולהתיר. ה sandbox עושה כרצון המשתמש | התכנית בנויה עם אלגוריתם אשר מחליט לבד מה מותר ומה אסור. המשתמש רק מקבל את ההחלטה הסופית |
| האם חוזר LOG של הריצה? | כן,כל הקריאות לפונקציות ופעולות עליהן ביקש המשתמש לדווח | חוזר LOG אשר מתאר את כל הפעולות של הוירוס ועוד מידע שימושי כמו memory dumps, PCAP files וכו'. |
| שימוש בhooking על מנת לעצור קריאות API | כן | לא |
| האם יש צורך לשלם על התכנה? | חינם לכל משתמש. | יש צורך לשלם תשלום שנתי |
| דרושה הרשמה ומתן פרטים אישיים? | לא, אין צורך בהזדהות. דבר השומר על אנונימיות המשתמש. | יש צורך בהרשמה ארוכה הכוללת הזדהות שמגיעה עד לכתובת מגורים. |
| 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. | קיים מנגנון זיוף משתמש די מרשים. |

עוד נקודה אשר אני חושב חשוב לציין היא שה sandbox של Kaspersky מכיל אלגוריתם מאוד מתקדם של זיוף פעולות משתמש. כיוון שהם שולחים את הקובץ לVM על מנת לבחון אותו, הוירוס עלול להבין שהוא אינו במכונה אמיתית, ישמיד את עצמו וכך יישאר לא מזוהה. Kaspersky השקיעו הרבה עבודה באלגוריתם מאוד מתוחכם של זיוף פעולות משתמש. אך אין זה אומר שהוא מושלם ובעוד שהוירוס עובר אצלם לVM, אצלי הוא נמצא על המכונה האמיתית של המשתמש (כמובן בסביבה מוגנת) ולכן אין צורך לזייף פעולות משתמש כיוון שישנו משתמש אמיתי (בין אם זה משתמש שיושב מול המסך או סרבר עליו רץ ה sandbox). זהו אחד מההבדלים העיקריים בין ה sandboxes ואני חושב שזה חשוב לציון כיוון שאני רואה יתרון דווקא בתכנה שלי, גם מהבחינה של זיוף משתמש קצה וגם שאם רוצה המשתמש לקבל נתונים בזמן שהוירוס תחת בחינה יש לו אפשרות כזו, כיוון שכל הפעולות קורות במכונה שלו בזמן אמת.

# תוצר סופי

## תיאור הפרויקט

כאשר אדם יקבל קובץ ממקור לא אמין, יוכל לפתוח את סביבת ההרצה ולחקור את הקובץ על מנת שיוכל להבין אם הוא מזיק או לא. כאשר אדם יפתח את הclient הוא יקבל אפשרויות של איזה דברים הוא מעוניין לחסום, בין אם זה file system, sockets, registry, RAM וכו'. כל שעליו לעשות הוא לגרור את הקובץ לתוך סביבת ההרצה ולבחור מה הוא מעוניין לחסום, הוא יוכל לבחור האם כל חסימה תהיה במצב strict\normal\high ולקבל לאחר מכן דוח של הדברים אליהם ניסה הקובץ להגיע ומה ניסה לעשות איתם.

## דרישות ואילוצי פתרון

סביבת ההרצה לא בטוחה כלפי כל קובץ. כרגע הקבצים חייבים להיות קבצי C++\C# מקומפלים אשר עושים שימוש בWINAPI, אולי גם קבצי python יצטרפו לרשימה.

בנוסף חייב הsandbox לרוץ מעל סביבת windows היות והוא עושה שימוש בפונקציות winapi.

## אלגוריתמים עיקריים

החדרת הDLL אשר אחראי על הhooking

הקושי הראשון בפרויקט היה איך לעשות hijack לתכנית אשר נמצאת בprocess אחר בכלל. על קושי זה אני עונה באמצעות כך שאני פותח את התכנית הזדונית בprocess מרוחק מתוך התכנית הראשית באמצעות winapi ואז מחדיר לתוך הprocess המרוחק את הDLL אשר אחראי על הhooking באמצעות DLL injection.

הדרך שזה נעשה היא שלאחר יצירת הprocess המרוחק אני עושה allocate לזכרון בprocess המרוחק וכותב שם את שם הDLL שלי. לאחר מכן אני יוצא thread בprocess המרוחק עם הפונקציה loadlibrarya ונותן לה את הכתובת של הזכרון בprocess המרוחק שבו כתוב השם של הDLL שלי. הפונקציה הזאת מעלה את הDLL למרחב הכתובות של הprocess המרוחק וכך אני יכול להפעיל את הhooking כאילו אני בתוך הprocess עצמו.

Inline hooking winapi functions

את תהליך זה אני עושה באמצעות שיטה שונה מעט משיטת הטרמפולינה. אני מקבל את הכתובת של הפונקציה לה אני רוצה לעשות hook ודורס את חמישת הבייטים הראשונים שלה, אני מכניס שם פקודת push עם הכתובת של הפונקצייה החדשה אליה אני רוצה לקפוץ, לאחר מכן פקודת retn . כך אני קופץ לפונקציה החדשה שלי ולאחר מכן בתוך הפונקציה שלי אני מחזיר את חמשת הבייטים הראשונים על מנת שאוכל לבצע שימוש בפונקציה זו. את כתובת הפונקציה אני משיג באמצעות getprocadress – פקודה של winapi.

לעבוד על התכנית הזדונית לחשוב שהיא רצה ריצה רגילה

על מנת שהתכנית הזדונית לא תיפול בקריאת פונקציה הראשונה אני צריך לעבוד עליה שהיא רצה ריצה רגילה ויכולה לבצע את הפעולות שהיא רוצה. בשביל לעשות דבר זה אני "עובד" עליה שהקריאות שהיא מבצעת אכן מתבצעות. כלומר, אם נגיד היא קוראת מקובץ אשר לא סומן כקובץ חשוב, אני מאפשר לה לקרוא אותו אך כאשר תרצה לכתוב אליו, אני מבצע את השינויים שהתכנית מנסה לעשות בקובץ זמני אשר נמחק בסוף ריצת התכנית. או אם התכנית מנסה לשלוח הודעה באמצעות socket.send אני מחזיר לה return code של הצלחה אך מבטל את שליחת ההודעה. אלגוריתמים כאלה קיימים כמעט לכל פונקציה, גם רגיסטרי וRAM.

## תסריטים עיקריים

להוסיף תרשים של זמן – אחד בו נשלח קובץ נכון ואחד בו נשלח קובץ לא טוב.

# תהליך כתיבת הפרויקט

## 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 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.

## אתגרים ואופציות שונות למימוש

* 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.

# מרכיבי הפרויקט

## תיחום הפרויקט

הפרויקט עוסק בתחומים הבאים:

* תקשורת – יש מעבר של התכנית הזדונית על גבי socket בין לקוח לשרת ולאחר מכן שליחת הlog
* אבטחת מידע – יש בדיקה של פעולות זדוניות לregisty או למערכת הקבצים ומניעת גישה לאזורים אסורים.
* מערכות הפעלה – שימוש רב בprocess וthreads שיוצר הסרבר על מנת להריץ את התכנית הזדונית תחת סביבת ההרצה הבטוחה. שימוש בwinapi באופן נרחב על גבי כל הפרויקט.
* תצוגה – שימוש בc# winforms
* מבנה נתונים – TBD
* מחלקות – המחלקות העיקריות הינן הserver,client ומחלקת הhook אשר בה יש את כל הפונקציות שקשורות לhooking וכתובות וכו'.

## סביבת העבודה (טכנולוגיה)

שפות תכנות

כל קוד הלוגיקה והhooking כתוב בC++ מכיוון שאני מבצע שימוש נרחב בwinapi.

הגרפיקה תיכתב בC# winforms היות והוא נוח מאוד לכתיבה, גמיש ובעל יכולות גבוהות.

סביבת פיתוח

את קוד הC++ והC# אני כותב בסביבת Visual Studio 2022

## מבט טופולוגי

לצייר בwhiteboard, סרבר ובתוכו סביבת ההרצה, לקוח ובתוכו הGUI

להמשיך לעשות על פי הדוגמה את מה שצריך להוסיף