[תאריך]

Rafael Wurf

[שם חברה]

[כותרת מסמך]

[כותרת משנה של מסמך]

תוכן עניינים

[פרק 1: מבוא 2](#_Toc12314155)

[פרק 2: הגדרת הבעיה 3](#_Toc12314156)

[פרק 3: סקירת ספרות 4](#_Toc12314157)

[פרק 4: מודל האיום 5](#_Toc12314158)

[1.4 היסטוריה קצרה של כופרות 5](#_Toc12314159)

[2.4 הקורבן 5](#_Toc12314160)

[3.4 טכניקות הדבקה 5](#_Toc12314161)

[4.4 יכולת התוקף ואופן הפעולה הבסיסי של כופרה מודרנית 5](#_Toc12314162)

[פרק 5: אופן עבודת הפתרון 7](#_Toc12314163)

[פרק 6: תיאור תהליך הזיהוי 8](#_Toc12314164)

[1.6 אזור מוגן 8](#_Toc12314165)

[2.6 אנטרופיה: 8](#_Toc12314166)

[3.6 קבצי מלכודת: 8](#_Toc12314167)

[4.6 מערכת GID: 8](#_Toc12314168)

[5.6 מדדי זיהוי 8](#_Toc12314169)

[פרק 7: ממשק ומימוש 10](#_Toc12314170)

[1.7 דרייבר 10](#_Toc12314171)

[1.1.7 מימוש 10](#_Toc12314172)

[2.1.7 מערכת GID 10](#_Toc12314173)

[3.1.7 איסוף מידע בדרייבר 11](#_Toc12314174)

[4.1.7 שמירת מידע בדרייבר 12](#_Toc12314175)

[2.7 תוכנת צד המשתמש 13](#_Toc12314176)

[1.2.7 מימוש 13](#_Toc12314177)

[2.2.7 קבצי מלכודות 13](#_Toc12314178)

[3.2.7 מידע הנאסף עבור קבוצת תהליכים משותפי GID 13](#_Toc12314179)

[4.2.7 זיהוי תהליך כזדוני 14](#_Toc12314180)

[5.2.7 מודל הגיבוי 14](#_Toc12314181)

[7.2.7 אזור מוגן 15](#_Toc12314182)

[3.7 תקשורת בין הדרייבר לתוכנת צד משתמש 15](#_Toc12314183)

[פרק 8: מכלול הבדיקות שנבדקו 16](#_Toc12314184)

[פרק 9: תוצאות הפרויקט 17](#_Toc12314185)

[פרק 10: ניתוח בעיות והמלצות לעתיד 18](#_Toc12314186)

[פרק 11: ביבליוגרפיה 19](#_Toc12314187)

[נספח א: בניית הפרויקט ותפעולו 20](#_Toc12314188)

[נספח ב: תוצאות זיהוי 21](#_Toc12314189)

[נספח ג: קטגוריות סיומות 22](#_Toc12314190)

# פרק 1: מבוא

הפרויקט שלנו עוסק בזיהוי כופרה והשבה למצב תקין של נקודת-קצה.

החלק הארי של הפתרון שלנו עוסק בזיהוי של תקיפה מכופרה. לצורך כך בנינו היוריסטיקה שנוסתה על כופרות מהעולם האמיתי כפי שנתאר בפרק 8, המשתמשת במדדים שונים שחריגה מהם אופיינית לכופרה ומבדילה אותה מתוכנות בלתי מזיקות. מכיוון שפתרון זה משתמש בזיהוי התנהגות "כופרתית" תמיד הכופרה תייצר נזק כלשהו, והמטרה שלנו בפרויקט הייתה להפחית את הנזק ככל הניתן יחד עם מזעור מספר מקרי הזיהוי הכוזבים (false-positive detection).

לצורך הזיהוי פיתחנו אפליקציה ודרייבר, כאשר הדרייבר מבצע את איסוף הנתונים והאפליקציה משמשת לניתוח הנתונים. בחרנו בתצורה כזו כדי לאפשר כמה שיותר שליטה על מערכת ההפעלה, הן מבחינת יכולת איסוף המידע והן מבחינת היכולת להשפיע על אופן הפעולה שלה. נתאר בהרחבה על הארכיטקטורה ומימושה בפרק 7.

לבסוף פיתחנו שירות גיבוי ושחזור בענן המתבסס על Microsoft Azure. השירות הוא בסיסי ונועד בעיקר לצורך הוכחת היתכנות והשלמות של הפתרון. נתאר בפרק 10 כיצד ניתן להרחיב את השירות כך שיענה על דרישות רחבות יותר.

פרט למטרות העל שתוארו לעיל, השתדלנו לייצר פתרון המערב את המשתמש מעט ככל הניתן, ולא מסתמך על הבנתו או יכולתו, ובאופן כללי שיעבוד באופן עצמאי כמעט לחלוטין.

# פרק 2: הגדרת הבעיה

*נקודת קצה* בהקשר של הפרויקט היא מחשב בסביבת עבודה של Windows 10.

*כופרה* (Ransomware) היא סוג של תוכנה זדונית המשתמשת בטכניקה של קריפטו-וירולוגיה כדי לאיים בפרסום המידע של הקורבן או למנוע גישה אליו כל עוד הוא אינו משלם כופר [מקור – ויקיפדיה]. כיום התשלום הוא באמצעות מטבעות קריפטוגרפים. אנו נתמקד בפרויקט בסוג השני של כופרות, כלומר זה המונע גישה למידע באמצעות הצפנה.

הנחות העבודה של הפרויקט:

* כופרות משתמשות בהצפנות טובות ובטכניקות הצפנה בטוחות. כלומר, מרגע שכופרה הצליחה להצפין קבצים, אין דרך קלה לפענח אותם.
* כופרות יכולות לרוץ מתוך קובץ הרצה עצמאי או לרוץ דרך קובץ הרצה לגיטימי שלא קשור לכופרה.
* כופרות רצות בהרשאות רגילות.
* כופרות תמיד יצליחו לייצר נזק מסוים (אבל שניתן לשחזור).

דרישות מהפתרון:

* אוטונומיות – הפתרון ידע לתת מענה ללא התערבות המשתמש.
* אמינות – הפתרון ידע לזהות כל כופרה וימנע ככל הניתן ממתן זיהויים כוזבים.
* שלמות – הפתרון ידע לתת מענה כולל לכופרה, גם אם מידע אבד.

# פרק 3: סקירת ספרות

בתחילת העבודה נעשתה סקירה של שיטות לזיהוי כופרה, מודל האיום של כופרות ופעילותן ועל כלים אותם מספקת מערכת ההפעלה למעקב אחר פעילות במערכת.

את מודל האיום מצד כופרות אנו נפרט לעמוק בפרק 4.

מערכת ההפעלה (Windows) מאפשרת למפתחים להתממשק עם מערכת הקבצים בעזרת hooking לפעולות בעזרת הממשק של Windows minifilter driver [Minifilter], המודל מאפשר לכתוב דרייבר העוקב בזמן אמת אחר פעולות במערכת הקבצים.

חקרנו שיטות זיהוי לכופרות, וראינו מדדים שונים אפשריים לזהות התנהגות כופרתית.

ShieldFS, המאמרים באתר, תוכנת CryptoMonitor

# פרק 4: מודל האיום

להוסיף הסתמכויות

נתייחס לאיום בהקשר של הנחות העבודה והדרישות שהצבנו בפרק 2.

## 1.4 היסטוריה קצרה של כופרות

בתחילת דרכן של הכופרות הן מנעו גישה למידע על ידי הסתרתו מהמשתמש, או השתמשו בצפנים חלשים. בנוסף, באותם זמנים היה קשה לקבל את תשלום הכופר מבלי להשאיר עקבות. עקב כך, הן היו לא מוצלחות ולא הצליחו להשיג את מטרתן.

עם המצאתו של המטבע הקריפטוגרפי החלו מופיעים שוב מתקפות כופרה המקבלות את תשלום הכופר במטבעות קריפטוגרפים. כיום, הכופרות הן די מתוחכמות, משתמשות בצופן סימטרי כדי להצפין את הקבצים עצמם מטעמי יעילות ומצפינות את המפתח בעזרת צפנים אסימטריים המנוהלים של מפתחות בעזרת Command & Control. היקפי הנזק הכלכלי שכופרות גורמות נאמדים במיליארדי דולרים [https://cybersecurityventures.com/ransomware-damage-report-2017-5-billion/], ועל כן יכולות להיות רווחיות מאוד לתוקפים. לדוגמה, הכופרה WannaCry שפרצה במאי 2017 הדביקה מעל ל-200,000 מחשבים ב-150 מדינות ודרשה 300 דולר למחשב.

ההערכות הן כי ב-2019 היקפי הנזק הכלכלי שיגרם מכופרה צפויים להגיע לשיא של מעל 11 מיליארד דולר.

## 2.4 הקורבן

במקרים רבים מתקפות הכופרה יכולות להימנע באמצעות נקיטת כללי זהירות בסיסיים, אך משתמשים רבים אינם מודעים לסכנה. קורבנות בעייתיים במיוחד הם מוסדות וארגונים בהם משתמשים לא מנוסים מתפעלים מחשבים, ואשר מהווים מטרה עם פרופיל גבוה כדוגמת בתי חולים.

## 3.4 טכניקות הדבקה

בדרך כלל כופרות מועברות בשיטה של סוס טוראני. מרמים את הקורבן כדי שיפתח את קובץ המכיל את קוד ההרצה של הכופרה בהסוואה של קובץ לגיטימי, לעיתים בשילוב טכניקות של הנדסה חברתית. אך יש גם יוצאות מן הכלל – כופרת WannaCry שילבה תולעת שניצלה חולשה במערכת ההפעלה Windows, וידעה להפיץ את עצמה למשתמשים אחרים ברשת גם ללא התערבות המשתמש.

## 4.4 יכולת התוקף ואופן הפעולה הבסיסי של כופרה מודרנית

אופן הפעולה של כופרה משתנה מכופרה אחת לשנייה, אבל העקרונות הבסיסיים דומים. המטרה של הכופרה בסופו של דבר היא להצפין כמה שיותר מידע, ולמנוע יכולת מהמשתמש לשחזר את המידע מבלי לשלם את הכופר. לכן בדרך כלל הכופרה תנסה למחוק את הגיבויים של המשתמש, ולאחר מכן (או במקביל) תפנה להצפנת קבצים מסוגים מסוימים.

לאחר תהליך ההצפנה תדווח בדרך כלשהי למשתמש המותקף על התקיפה ותדרוש כופר על הקבצים שהוצפנו.

היא נוהגת להצפין קבצים רגשים כגון מסמכים, תמונות וכדומה. בנוסף, ראינו כי כופרות נוהגת להימנע מקבצים גדולים. אופן פעולה זה נועד כדי לגרום נזק גדול בפרק זמן קצר ככל הניתן, כדי שזיהוי של תקיפה בכל מקרה יהיה מאוחר מדי.

לכופרה יש מספר דרכים להימנע מזיהוי, נתאר דרכים עיקריות בהן נתקלנו.

* אחת השיטות הבסיסיות שחוזרות על עצמן היא מתן שמות לגיטימיים לתהליכים שהיא מריצה, כגון “Windows Update” וכדומה. כך גם במידה והמשתמש חושד בפעילות זדונית, יהיה קשה לזהות תהליכים אלה.
* חלוקת העבודה בין מספר תהליכים שונים ובכך לחלק את האַחְרָיוּתִיּוּת (accountability) בין התהליכים. בצורה זו, אם אוסף תהליכים ככלל מתאפיין בהתנהגות כופרתית, ייתכן שכל תהליך בנפרד לא נראה כזה. כך למשל ראינו ב-WannaCry שתהליך אחד מבצע את ההצפנות עצמן ותהליך אחר מבצע את איסוף המידע על מערכת הקבצים. ניתן גם כמובן לבצע חלוקות עדינות יותר שיגרמו לקושי רב עוד יותר בזיהוי.
* הזזת קבצים לאזורים לתיקייה זמנית לשם הצפנתם. חלק מהכופרות בנוסף להצפנת הקבצים, מעבירות אותם לאזורים העלולים להיות לא מוגנים, ובכך להקשות על מלאכת הזיהוי.

להתייחס למאמר <https://work.delaat.net/rp/2017-2018/p05/report.pdf>, בהצעות לשיפור אנחנו צריכים להתייחס למאמר ולבעיה הזו ולהתייחס לפתרון אפשרי של מניעת התקנה של דרייברים נמוכים יותר על מערכת הקבצים, רישום במקום נמוך וחסימת הודעות מסוג SL\_FORCE\_DIRECT\_WRITE שהן בעצם פעולות IRP שיוצר דרייבר למערכת הקבצים או לנסות לזהות תקשורת בין דרייבר לאפליקציה ולנטר את התקשורת.

להתייחס לכופרות המנסות להשיאר במערכת על ידי יצירת קובץ נוסף בתיקיית start-up, שינוי רגיסטרי או הוספת task sched.

# פרק 5: אופן עבודת הפתרון

אזור מוגן - ההגדרה המדויקת לאזור מוגן תינתן בפרק 6, באופן עקרוני אלו אזורים במערכת הקבצים אשר המערכת מגנה עליהם.

תמונת מצב – אוסף מידע שנאסף עד כה על התהליך בצירוף מידע חדש (3.2.7).

נתאר את אופן הפעולה העקרוני של המערכת באלגוריתם:

1. לכל תהליך שמבצע פעולה מול מערכת הקבצים (כגון קריאה, כתיבה, פתיחה, מחיקה ועוד) מול אזור מוגן:
   1. אסוף מידע אודות הפעולה.
   2. העברת הפעולה לניטור ולבניית תמונת מצב של התהליך.
   3. ניתוח תמונת המצב החדשה של התהליך.
   4. קבלת החלטה האם תהליך הוא זדוני:
      1. עבור תהליך זדוני מתבצע ניסיון עצירה של התהליך וניסיון שחזור של קבצים שנפגעו.
2. המתנה לפעולה חדשה של תהליך וחזרה לשלב 1.

# פרק 6: תיאור תהליך הזיהוי

## 1.6 אזור מוגן

תיקיות במערכת הקבצים עליהם בחר המשתמש להגן, תיקיות אלו מוגנות באופן רקורסיבי.

עבור תיקיות אלו מתבצע גיבוי של התוכן לגיבוי ענן (5.2.7), ולאחר זיהוי של כופרות נעשה ניסיון שחזור לקבצים מאזורים אלו שנפגעו.

עבור כל תיקייה מוגנת אנו אוספים מידע סטטיסטי על מספר הקבצים והתיקיות באופן רקורסיבי, מידע המסייע בזיהוי התנהגות חשודה.

תיקיות אלו ניתנות להוספה ולהסרה על ידי המשתמש במערכת. (6.2.7)

## 2.6 אנטרופיה:

אנטרופיה היא מדד פשוט המספק מידע על אי הוודאות של מידע.

חישוב האנטרופיה מתבצע לפי נוסחת שנון לאנטרופיה:

כאשר הוא היחס של מספר מופעי ערך הבית בבלוק למספר הבתים הכולל בבלוק, כלומר נרמול של מספר מופעי הבית בבלוק .

בחישוב זה e הינו מספר בטווח 0 ל- 8.

קבצים מסוימים כמו קבצים מוצפנים או דחוסים מאופיינים באנטרופיה גבוהה.

## 3.6 קבצי מלכודת:

קבצים שנשתלים על ידי מערכת.

מכיוון שאלו קבצים שנוצרו על ידי המערכת ולא על ידי המשתמש, כל פעולה המשנה אותם עשויה להצביע על התנהגות זדונית. קבצים אלו משתתפים במדדי הזיהוי.

פירוט נוסף אודות המלכודות בפרק 7 – מימוש. (2.2.7).

## 4.6 מערכת GID:

לכל אוסף תהליכים הקשורים זה לזה בקשרי יצירה ניתן מזהה ייחודי זהה, שנועד לאגד אותם לצורך זיהוי משותף. המערכת מנתחת התנהגות של קבוצת תהליכים משותפי GID לצורך זיהוי ובכך מעלה את האחריותיות של אפליקציות מרובות תהליכים.

פירוט נוסף אודות מערכת ה- GID בפרק 7 – מימוש. (2.1.7).

## 5.6 מדדי זיהוי

בטבלה א' נתאר את המדדים הבאים מהווים ניתוח ראשוני עבור התנהגות קבוצת GID ומהווה בסיס לתהליך הזיהוי. לכל GID מחושבים מחדש המדדים לאחר סדרת פעולות שנקלטו עבור תהליכים מקבוצת ה-GID. המדדים נורמלו על מנת לאפשר התאמה של המודל לתרחישים שונים, וערכי הסף נקבעו לפי ניסויים על כופרות ותהליכים לגיטימיים.

צירוף משוקלל של מספר מדדים יחד יכול להבדיל בין אפליקציות זדוניות ללגיטימיות, רק אפליקציה העוברת על מספר משוקלל של מדדים יחד מעלה אינדיקציה על אפליקציה זדונית.

על מנת להפעיל את מודל הזיהוי נדרש מספר קבצים מינימלי באזור המוגן של 30 קבצים ו- 5 תיקיות, אזור קטן מידי ייטה לריבוי זיהויים כוזבים.

טבלה א'

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| מדד | משקל | תיאור | רציונל | ערך סף |
| מחיקת קבצים | 1 | מספר הקבצים שנמחקו בתוספת מספר המלכודות שנמחקו (עם משקל רב יותר), ביחס למספר הקבצים המוגנים ולמספר הקבצים אליהם האפליקציה ניגשה. | כופרות רבות מוחקות את הקבצים המקוריים לאחר הצפנה | 0.3 |
| יצירת קבצים | 1 | מספר הקבצים שנוצרו ביחס למספר הקבצים אליהם האפליקציה ניגשה . | כופרות רבות כותבות קובץ מוצפן חדש עבור כל קובץ אותן הן מצפינות. | 0.5 |
| שינוי שמות קבצים | 2 | מספר הקבצים ששם שונם בתוספת מספר המלכודות ששמן שונה (עם משקל רב יותר), ביחס למספר הקבצים אליהם ניגשה האפליקציה. | כופרות רבות משנות שמות קבצים לפני או לאחר תהליך ההצפנה. | 0.3 |
| קריאת תוכן תיקיות | 1 | מספר התיקיות שנקראו ביחס למספר התיקיות הכולל. | כופרות חייבות לסרוק את התיקיות בהן הקבצים אותם היא הולכת להצפין. | 0.4 |
| אנטרופיה גבוהה | 2 | ממוצע האנטרופיה של פעולות הכתיבה ביחס לממוצע האנטרופיה של פעולות הקריאה. | הצפנות מאופיינות באנטרופיה גבוהה. השוואת ממוצע הכתיבה לממוצע הקריאה מאפשר לצמצם זיהויים כוזבים של אפליקציות העובדות עם קבצים בעלי אנטרופיה גבוהה. | 7.6 עבור אנטרופיה גבוהה. |
| שימוש בסיומות קבצים | 2 | חלקנו את סוגי סיומות הקבצים לקטגוריות (נספח ג). היחס בין מספר הקטגוריות בהם אפליקציה השתמשה ביחס למספר הקטגוריות, והיחס בין מספר הסיומות אותם כתבה האפליקציה ביחס למספר הסיומות שנפתחו. | כופרות ניגשות למספר קטגוריות שונות בדר"כ במספר רב יותר ביחס לתוכנות לגיטימיות. בנוסף כופרות מספר מועט של סיומות ביחס למספר הסיומות בהן הן משתמשות. | 0.25 עבור שימוש בקטגוריות.  0.2 עבור היחס בין כתיבה לפתיחה |
| שינוי סיומות קבצים | 2 | מספר שינויי הסיומת של קבצים ביחס למספר הקבצים אליהם ניגשה האפליקציה. | כופרות רבות משנות סיומת של קבצים לאחר הצפנת קבצים, על ידי הוספת סיומת לקובץ. | 0.2 |
| שימוש במלכודות | 2 | דגל המציין האם מספר התיקיות המכילות מלכודות שנפתחו עבר את הסף. | אפליקציות לגיטימיות לא ייפתחו קבצי מלכודת רבים, מכיוון שקבצים אלו לא נוצרו על ידי המשתמש. כופרה אינה מסוגלת להבדיל קבצים אלו מקובץ רגיל ולכן תנסה לפתוח או להצפין קבצי מלכודת. | 4 |
| קריאת קבצים | 1 | מספר הקבצים שנקראו בתוספת מספר המלכודות שנקראו (עם משקל רב יותר), ביחס למספר הקבצים המוגנים. | כופרות קוראות את הקבצים אותן היא מצפינה. | 0.4 |
| גישה לקבצים | 2 | מספר הקבצים שנכתב אליהם בתוספת מספר המלכודות שנכתב אליהן (עם משקל רב יותר), ביחס למספר הקבצים שאליהם האפליקציה ניגשה. | כופרות כותבות לאותו מספר קבצים בערך כמו מספר הקבצים אותם היא קוראת.  כופרה צריכה לקרוא כל קובץ לפני הצפנתו. | 0.5 |
| הזזת קבצים | 2 | מספר הקבצים שהוצאו מאזור מוגן ביחס למספר הקבצים שהוכנסו. | כופרות מסוימות מזיזות קבצים על מנת להקשות על זיהוי תהליכי הצפנה. | 0.5 |
| מספר נקודות לזיהוי כופרה | 8 |  |  |  |

# פרק 7: ממשק ומימוש

הפתרון מחולק לשני חלקים עיקריים: דרייבר - Windows kernel driver (KMDF) ותוכנת צד משתמש - (C++ cli on CLR) עם תפריטי משתמש (GUI), בין חלקים אלו ישנה תקשורת מבוססת Port.

## 1.7 דרייבר

אחראי על איסוף מידע מהמערכת בנוגע לעבודה של תהליכים מול מערכת הקבצים, זיהוי תהליכים חדשים במערכת וסיווגם, יצירת קבוצות GID עבור קבוצות תהליכים, סינון הודעות שנאספות ממידע השייך לתהליכי מערכת בלבד והעברת מידע שנאסף לצד משתמש לשם ניתוח.

### 1.1.7 אופן המימוש

בחרנו לממש את הדרייבר כ- Windows Minifilter driver.

Windows Minifilter driver הינו דרייבר עבור מערכת הקבצים במערכות מבוססות Windows, המאפשר רישום למעקב אחר פעילויות במערכת הקבצים על ידי פונקציות שמוגדרות על ידי כותב הדרייבר, פעולות כגון כתיבה לקובץ, פתיחת קובץ, קריאה, סגירה, ושינוי שם קובץ. עבור פעולות מול מערכת הקבצים ניתן לנטר פעולות לפני פעילותן ולהשפיע עליהן במידת הצורך, ולנטר פעולות ותוצאתן לאחר השלמתן.

דרייבר זה מאפשר מעקב בזמן אמת אחר פעולות של אפליקציות מול מערכת הקבצים, וגם תוצאת הפעילות לאחר ביצוען.

על מנת להתמודד עם כופרות המחלקות את עבודתן על מספר תהליכים שונים יצרנו מערכת GID פנימית של הדרייבר. לשם מערכת זאת, בעת טעינת הדרייבר מתבצע רישום של הדרייבר לעדכוני מערכת על יצירת תהליכים וסיומם.

### 2.1.7 מערכת GID

ה- GID הוא מזהה ייחודי שנוצר על ידי הדרייבר עבור אפליקציות צד משתמש ועבור התהליכים הנוצרים על ידן, ה- GID עבור אפליקציה נוצר לראשונה כאשר התהליך הראשון באפליקציה נוצר. ה-GID משותף לכל תהליך הנוצר מתהליך עבורו קיים GID קודם לכן. GID מייצג קבוצה של תהליכים בעלי קשר משותף דרך יצירה.

בעת יצירת תהליך במערכת או סיומו, הדרייבר מקבל כקלט את המזהה הייחודי של התהליך במערכת ומזהה הייחודי של אב התהליך (יוצר). במידה ולאב ישנו GID קודם אנו מצרפים אותו אליו. אחרת אם התהליך אב הוא תהליך מערכת והתהליך החדש גם כן תהליך מערכת אין מעקב עליו והוא לא מוכנס למערכת ה-GID. אחרת נוצר GID חדש עבור התהליך הנוצר. אלגוריתם זה מבטיח לנו כי אפליקציה שאינה תהליך מערכת ומייצרת תהליכים נוספים לחלקות העבודה ואף קוראת לתהליכי מערכת במהלך עבודתה, יבוצע מעקב על כל תהליכיה תחת קבוצת GID יחידה.

### 3.1.7 איסוף מידע בדרייבר

הדרייבר עוקב אחר יצירה או פתיחת קובץ, סגירת קובץ, קריאה מקובץ, כתיבה לקובץ או שינוי מידע על קובץ. המעקב מתבצע רק כאשר ישנו תהליך צד משתמש המחובר לדרייבר בתקשורת מבוססת פורט (3.7).

בעת ניטור פעולה אנו מייצרים הודעה מסוג DRIVER\_MESSAGE וממלאים אותה בהתאם עם:

* מזהה ייחודי לפעולה שבוצעה.
* מזהה הייחודי של התהליך (PID) .
* המזהה הייחודי שניתן לקבוצה של התהליך (GID) על ידי הדרייבר.
* שם הקובץ עליו מבוצעת הפעולה.
* מזהה הייחודי של הקובץ:

FILE\_ID – מבנה בן 128 סיביות שבצירוף עם מספר volume מייצג באופן חח"ע קובץ (רפרנס).

* הסיומת של הקובץ הנ"ל.
* מידע נוסף הקשור בפעולה הספציפית אותה מנטרים, נתאר זאת בהסבר על הפעולות.

הפעולות אותן אנו מנטרים הן:

* פתיחה או יצירת קובץ (IRP\_MJ\_CREATE):

פעולה זו מנוטרת כאשר תהליך מבקש לפתוח קובץ קיים, לייצר קובץ, לדרוס קובץ או למחוק קובץ מיד לאחר סגירותו.

עבור קובץ הנפתח אנו בודקים האם הוא נמצא באזור מוגן שהוגדר על ידי אפליקציית צד משתמש (1.6), לפי זה מחליטים האם לנטרו.

עבור פעולה זו אנו בודקים האם הוא מייצג תיקייה, האם מדובר בקובץ חדש או דריסה של קובץ קודם, או אירוע פתיחת קובץ בלבד, ונבדק האם הקובץ נפתח במטרה להימחק לאחר סגירתו. מידע זה מצורף להודעה עבור הפעולה.

* קריאה מקובץ (IRP\_MJ\_READ):

פעולה זו מנוטרת כאשר תהליך מבקש לקרוא מקובץ שנפתח קודם לכן.

במידה וקובץ זה נמצא באזור מוגן על ידי משתמש (1.6), אנו מנטרים אחר מספר הבתים שנקראו במסגרת הפעולה, והאנטרופיה של פעולת הקריאה (2.6). מידע זה מצורף להודעה עבור הפעולה.

* כתיבה לקובץ (IRP\_MJ\_WRITE):

פעולה זו מנוטרת כאשר תהליך מבקש לכתוב לקובץ שנפתח קודם לכן.

עבור פעולה זו אנו מנטרים אחר מספר הבתים שנכתבים במסגרת הפעולה, והאנטרופיה של פעולת הכתיבה (2.6). מידע זה מצורף להודעה עבור הפעולה.

* סגירת קובץ (IRP\_MJ\_CLEANUP):

פעולה זו מנוטרת כאשר תהליך מבקש לסגור קובץ שהוא פתח קודם לכן.

עבור פעולה זו אין מידע נוסף.

* שינוי מידע קובץ (IRP\_MJ\_SET\_INFORMATION):

פעולה זו מנוטרת כאשר תהליך מבקש לשנות מידע על קובץ קיים, כגון שינוי שם, מחיקת קובץ, שינוי גודלו ועוד.

עבור פעולה זו אנו בודקים האם הפעולה נועדה למחיקת הקובץ הקיים או האם מדובר בשינוי שם של הקובץ. עבור שינוי שם אנו בודקים מהי הסיומת החדשה של הקובץ, ובנוסף מדווחים האם הקובץ מוכנס או מוצא מה/לאזור המוגן. מידע זה מצורף להודעה עבור הפעולה.

### 4.1.7 שמירת מידע בדרייבר

המידע בדרייבר עבור פעולות במערכת ומידע עבור תהליכים ומערכת ה-GID נשמר תחת אובייקט DriverData הגלובאלי לדרייבר, האובייקט נוצר בעת טעינת הדרייבר. מבנה זה מכיל:

* irpOps – רשימה מעגלית

משמש לשמירת פעולות קבוצות GID.

מכיל הודעות מסוג DRIVER\_MESSAGE המייצגות פעולות מול מערכת הקבצים ונשמרות לקראת העברתן לצד משתמש עד להגעת בקשה לפעולות אחרונות.

כאשר מגיעה הודעה מהמשתמש להעברת פעולות אחרונות מול מערכת הקבצים, נקראות ההודעות הראשונות מהרשימה המעגלית (First in – first out) ומועתקות לחוצץ שתוכנת צד המשתמש מעבירה בעת שליחת ההודעה. לאחר העברת הודעה לצד משתמש ההודעה נמחקת מהרשימה המעגלית.

רשימה זו ממומשת כרשימה מעגלית (LIST\_ENTRY) עם וכל הפעולות לרשימה זו מתבצעות יחד עם מנעול irpOpsLock.

* rootDirectories – רשימה מעגלית

משמש לשמירת תיקיות המהוות שורש (Root) לאזורים מוגנים (1.6).

מסייע בסינון חלק מההודעות וניהול מעקב אחר הזזת קבצים מחוץ ולתוך אזורים מוגנים.

הוספת התיקיות לרשימה זו מתבצעת כאשר שולח צד המשתמש בקשה להוספת תיקיה לרשימת האזורים המוגנים, ומוסרת רשומה בבקשת הסרה של תיקייה.

רשימה זו ממומשת כרשימה מעגלית (LIST\_ENTRY) עם וכל הפעולות לרשימה זו מתבצעות יחד עם מנעול directoriesSpinLock.

* GidsList - רשימה מעגלית

משמש לשמירת מבני GID, המכילים את מספרי התהליכים הנמצאים בתוכם.

מבנה זה משמש לשחרור זיכרון של הדרייבר למעקב.

* GidToPids – טבלת גיבוב

מיפוי של GID לרשימה מעגלית המכילה את מספרי ה-PID המוכלים בתוכו, מאפשר חיפוש מהיר של מספרי תהליכים בעת עצירת קבוצת GID.

* PidToGids – טבלת גיבוב

מיפוי של PID לערך ה-GID אליו הוא שייך. (PidToGids).

## 2.7 תוכנת צד המשתמש

אחראית על מעקב אחר תהליכים וסיווגם לפי מדיניות הזיהוי, שליחת בקשת עצירת תהליכים לדרייבר, קביעת האזורים המוגנים על ידי המערכת והעברת מידע זה לדרייבר לשם ניטור ויצירת מלכודות (2.2.7) בהתאם למדיניות הזיהוי.

### 1.2.7 אופן המימוש

בחרנו לממש את תוכנת צד לקוח בשפת C++/cli, המשלבת את שפת C++ עם סביבת .NET, ורצה תחת המכונה הוירטואלית CLR (רפרנס), בחרנו לממש בסביבה זו מטעמי נוחות והיתכנות תפריטי משתמש (GUI).

תוכנת צד המשתמש פועלת בעזרת 3 חוטים עיקריים:

* החוט העיקרי הינו החוט האחראי על תפריטי המשתמש (GUI) ואחראי לקבלת פקודות מהמשתמש, כמו הוספת תיקייה, הסרה, קביעת מדיניות טיפול בתהליכים שזוהו ושליטה על רמת הדיווח (מצב verbose בתוכנה).
* שני החוטים האחרים משמשים לקבלת פעולות מול מערכת הקבצים של תהליכים עם GID מהדרייבר. חוטים אלו אחראים על איסוף מידע על פעולות לזיכרון של קבוצת ה-GID, הוספת רשומה ל- GID חדש במידה ואין כזה, בדיקת תקינות לקבוצת תהליכים עם GID משותף מול המדדים שנקבעו (5.6), ושליחת בקשה לסיום קבוצת ה-GID במקרה וקבוצה זאת זוהתה כזדונית. לאחר זיהוי תהליך כחשוד מתבצע ניסיון שחזור (5.2.7) של הקבצים שזוהו כנגועים או קבצים שנמחקו מתוך האזורים המוגנים (6.2.7).

בחרנו לממש את תהליך הזיהוי, טיפול ושחזור כשני חוטים עקב ריבוי המשימות המוקצות לחוטים אלו, אך ניתן להרחיב ולצמצם את מספר החוטים.

### 2.2.7 קבצי מלכודות

בעת הוספת תיקייה לרשימת האזורים המוגנים (1.6), רץ תהליך בהקשר של החוט הראשי (GUI) המייצר קבצי מלכודת עבור התיקייה שנוספה באופן רקורסיבי.

עבור כל סיומת בתיקיה נוצר קובץ מלכודת בודד לפי הפרמטרים:

* גודל רנדומלי הנמצא בין גדלי הקבצים בעלי אותה סיומת.
* תאריך יצירה ועריכה רנדומליים בין הקבצים בעלי אותה סיומת.
* שם רנדומלי עם אורך משתנה.

לאחר יצירת המלכודות מידע זה נשמר במילון שאליו ניתן לגשת מכל חוטי האפליקציה המקשר בין התיקיה בה נוצרו המלכודות, לרשימה של קבצי המלכודות בתיקייה לצורך מעקב.

בעת הסרת תיקיה ממעקב נמחקות כל המלכודות שנוצרו בתיקיה זו, והרשומות אודותיהן נמחקות.

### 3.2.7 איסוך ושמירת הפעולות לקבוצת GID

עבור כל GID נשמר מבנה הנקרא GProcessRecord המכיל:

* זמן התחלה וסיום.
* מספר הפעולות שבוצעו מול קבצי מלכודת: קריאה, כתיבה, מחיקה ושינוי שם.
* מספר הקבצים שהוכנסו או הוצאו מהאזורים המוגנים.
* מספר הסיומות של קבצים ששונו.
* שמות הקבצים ששונו (נמחקו או נכתבו לתוכם) או נוצרו באזורים המוגנים.
* שמות התיקיות ומספרן של התיקיות המכילות מלכודות בהן שונו קבצי מלכודת.
* מספר פעולות כתיבה בעלות אנטרופיה (2.6) גבוהה.
* קבצים שנמחקו לפי מזהה קובץ (FileId).
* קבצים שנוצרו לפי מזהה קובץ (FileId).
* קבצים שבוצעה מולם פעולת קריאה, כתיבה או שינוי שם לפי מזהה קובץ (FileId).
* קבצים שנכתב אליהם מידע לפי מזהה קובץ (FileId).
* קבצים שנקרא מהם מידע לפי מזהה קובץ (FileId).
* קבצים ששם שונה לפי מזהה קובץ (FileId).
* קבצי מלכודות ששונו לפי מזהה קובץ (FileId).
* שמות הסיומות של קבצים מהן נקרא מידע ושמות הסיומות של קבצים אליהן נכתב מידע.
* דגל עבור כל קבוצת קטגוריות של סיומות (נספח ג), המציין אם בוצעה פעולה כלשהי מול אותה קטגוריה.
* סכום ממוצעי האנטרופיה שחושבה בדרייבר עבור פעולות כתיבה.
* סכום ממוצעי האנטרופיה שחושבה בדרייבר עבור פעולות קריאה.
* סך הבתים שנכתבו לקבצים.
* סך הבתים שנקראו מקבצים.

המידע עבור כל קבוצת GID נשמר במילון אליו ניתן לגשת מכל חוטי האפליקציה, מפתח המילון הוא ה- GID וערכו הוא מצביע למבנה GProcessRecord שתיארנו קודם.

איסוף המידע נעשה באופן הבא: האפליקציה שולחת בקשה לפעולות אחרונות של GID, בתגובה לכך מחזיר הדרייבר את הפעולות האחרונות. כל פעולה מיוצגת על ידי מבנה בשם DRIVER\_MESSAGE המוכר לאפליקציה ולדרייבר (3.7).

עבור כל פעולה שהתקבלה מהדרייבר, אנו מעדכנים את שדות מבנה GProcessRecord המתאים ל-GID שבהודעה, ו - GIDזה מסומן לבדיקה בהתאם למודל הזיהוי.

### 4.2.7 זיהוי GID כזדוני

לאחר קבלת הודעות עבור פעולות מול מערכת הקבצים מהדרייבר, בודקת האפליקציה האם ניתן לקבוע כי קבוצת תהליכים GID הינה זדונית, בדיקה זו נעשית עבור כל קבוצת GID שעבורה מתקבלות פעולות.

עבור כל קבוצת GID, נבדק ה-GID עם המידע שנשמר לגביו ביחס למדדי הזיהוי (5.6).

במידה ו- GID מזוהה כזדוני ביחס למדדים, האפליקציה אוספת את המדדים שנחצו ומדווחת עליהם, שולחת בקשה לדרייבר לעצור את קבוצת התהליכים בעלי המזהה GID ומייצרת קובץ דיווח על ה- GID הזדוני ומנסה לשחזר קבצים שנקבעו בעקבות ה- GID הזדוני.

### 5.2.7 מודל הגיבוי

מודל הגיבוי הינו מבוסס ענן Azure, מתבסס על אחסון מבוסס Blob.  
בעת הוספת תיקייה לרשימת האזורים המוגנים (1.6), אנו מגבים אותה באופן רקורסיבי באינטרוולים (גיבוי אינטרוולי אינו ממומש, מבוצע גיבוי בודד) לענן ללא קבצי המלכודת המוכלים בתיקייה, אנו שומרים גרסאות קודמות של הקבצים המגובים, גיבויים אלו משמשים את האפליקציה לשחזור במקרה וזוהה GID זדוני.

בעת הסרת תיקיה מרשימת האזורים המוגנים נמחקים הקבצים מגיבוי הענן וכל הגרסאות הקודמות שנשמרו עבורם.

לאחר זיהוי GID כזדוני, נעשה שחזור לקבצים אשר זוהו כקבצים אותם שינה ה- GID שזוהה באזורים המוגנים, קבצים שנמחקו, נכתבו עליהם או שונה שמם. עבור כל קובץ שזוהה כקובץ שכזה, אנו מחפשים בגיבוי הענן גרסה בעלת תאריך הקודם לתאריך בו זוהה שה - GID הזדוני החל לבצע פעולות מול מערכת הקבצים לראשונה, ובמידה וקיים קובץ שכזה התוכנה מורידה אותו ומחליפה בקובץ ששונה.

### 6.2.7 אזורים מוגנים

את האזורים המוגנים הגדרנו בפרק 6 (1.6).

הוספה והסרה של תיקיות האזורים המוגנים נעשית דרך תפריטי המשתמש (GUI).   
לחצן Add Directory מוסיף תיקייה, בלחיצה על כפתור זה נפתח חלון המאפשר בחירה של תיקייה שתהווה שורש לאזור המוגן. לאחר הוספת תיקייה נעשה גיבוי של קבצי האזור המוגן לענן, נוצרים קבצי מלכודות ונשלחת הודעה לדרייבר על הוספת תיקייה לאזור המוגן בצירוף התיקייה שנוספה.

התיקיות שנוספו נשמרות באפליקציה באוסף של תיקיות המשותף בין חוטי האפליקציה.

הסרה של תיקייה מהאזור המוגן נעשית דרך לחצת Remove Directory, בלחיצה על אזור זה נפתח חלון המאפשר בחירה של התיקייה שאותה מסירים. בעת הסרת תיקיה נמחקים קבצי המלכודות שנוצרו עבור אזור זה, נמחקים כל הגיבויים עבור התיקיה ונשלחת הודעה לדרייבר להסרת התיקייה ממעקב.

## 3.7 תקשורת בין הדרייבר לאפליקציה

התקשורת בין הדרייבר לאפליקציה בצד משתמש מבוססת על צינור תקשורת מבוסס פורט.

בעת טעינת הדרייבר, יוצר הדרייבר את צינור התקשורת ומחכה להתחברות של האפליקציה לפורט, ההתחברות לדרייבר מפעילה את המעקב אחר פעולות מול מערכת הקבצים של תהליכים בדרייבר.

בשיטת התקשורת בה בחרנו האפליקציה הינה יוזמת ההתקשרות והדרייבר עונה להודעות המגיעות מהאפליקציה. ההודעות שמועברות מהאפליקציה לדרייבר הן:

* MESSAGE\_ADD\_SCAN\_DIRECTORY – בקשה להוספת תיקייה מוגנת, בהודעה זו מועברת התיקייה אותה נוסיף. בתגובה לכך הדרייבר מוסיף תיקייה לרשימת האזורים המוגנים. הדרייבר עונה בהצלחה/כשלון.
* MESSAGE\_REM\_SCAN\_DIRECTORY – בקשה להסרת תיקייה מוגנת, בהודעה זו מועברת התיקייה אותה מסירים. בתגובה לכך הדרייבר מסיר תיקייה מרשימת האזורים המוגנים במידה והיא קיימת שם. הדרייבר עונה בהצלחה/כשלון.
* MESSAGE\_SET\_PID – קביעת ה-PID של האפליקציה שלנו. בהודעה זו מועבר ה-PID של התהליך.
* MESSAGE\_KILL\_GID – בקשה לעצירת קבוצת GID, בהודעה זו נצרף את מספר ה- GID המתאים. בתגובה עונה הדרייבר על הצלחה או כשלון.
* MESSAGE\_GET\_OPS – בעזרת בקשה זו האפליקציה מבקשת מהדרייבר לשלוח את הפעולות האחרונות מול מערכת הקבצים שביצעו אפליקציות במערכת, האפליקציה מקצה buffer ומעבירה אותו לדרייבר בצירוף עם ההודעה ומחכה למילוי ה-buffer. בתגובה לבקשה זו הדרייבר ממלא את ה-buffer בבקשות אחרונות.

כל ההודעות מצד האפליקציה לדרייבר מועברות בעזרת המבנה: COM\_MESSAGE.  
מבנה זה מכיל את סוג ההודעה, PID של שולח ההודעה, GID במידה וצריך ושדה מחרוזת בעל גודל ארוך מספיק לקבלת שם קובץ מלא.

העברת פעולות מהדרייבר לאפליקציה מתבצעת בעזרת המבנה: DRIVER\_MESSAGE.

מבנה DRIVER\_MESSAGE:

* סיומת של הקובץ עליו בוצעה הפעולה.
* מזהה ייחודי של הקובץ (FILE\_ID).
* מספר הבתים ששומשו במסגרת הפעולה (רלוונטי לכתיבה וקריאה בלבד).
* אנטרופיה במידה וחושבה (רלוונטי לכתיבה וקריאה בלבד).
* PID של התהליך שביצע את הפעולה.
* מזהה של הפעולה, מציין האם בוצעה פעולת פתיחה/יצירה, מחיקה, סגירת קובץ, כתיבה או קריאה.
* משתנה דגל המציין האם חושבה אנטרופיה.
* משתנה המציין את סוג השינוי שנגרם עקב הפעולה או את סוגה (enum בשם FILE\_CHANGE\_INFO מכיל את כל אפשרויות שדה זה).
* מידע על מיקום הקובץ עליו בוצע הפעולה (enum בשם FILE\_LOCATION\_INFO, אומר אם הקובץ נמצא או לא נמצא באזור מוגן, אם מדובר בקובץ שמוצא מאזור מוגן או קובץ המוכנס לאזור מוגן).
* שם קובץ מלא.
* GID של התהליך מבצע הפעולה.
* מצביע למבנה DRIVER\_MESSAGE הבא במידה ויש אחד כזה.

מילוי buffer בעת הודעה מסוג MESSAGE\_GET\_OPS:

הפעולות נמשכות מהרשימה המעגלית של פעולות אחרונות בדרייבר. (4.1.7).  
לפני הכנסת ההודעות הדרייבר יוצר מבנה ראשוני הנקרא RWD\_REPLY\_IRPS, מבנה זה משמש אותנו לספירת כמות הפעולות שמוחזרות על ה-buffer.

לאחר המבנה RWD\_REPLY\_IRPS מוכנסות הודעות מסוג DRIVER\_MESSAGE, ובצמוד לכל הודעה מסוג DRIVER\_MESSAGE מוצמד שם הקובץ עליו מבוצעת הפעולה. לאחר מילוי ה-buffer הוא נראה כך:

# פרק 8: מכלול הבדיקות שנבדקו

בדיקת התוכנה והדרייבר נעשתה למול כופרות קיימות ופעולות לגיטימיות מצד משתמש כגון דחיסה של קבצים בעזרת תוכנות שונות, מחיקת קבצים, קריאה, כתיבה ושינוי שם קובץ.

בעת ניסוי התוכנה זיהינו בהצלחה כופרות כגון:

Katyusha, Cerber, Mamse, Jigsaw, Vipasana, Wannacry.

בדקנו זיהויים כוזבים על ידי יצירת קבצי zip,7z מקבצי האזורים המוגנים.

דחיסה של קבצים עלולה לעבור על מלכודות בתיקיות שונות, ליצור אנטרופיה גבוהה ביחס לפעולות קריאה, לקרוא ריבוי סיומות וכותבת סיומת בודדת ולקרוא ריבוי קבצים ותיקיות. הניקוד של כל אפשרויות אלו יכול ליצור זיהוי כוזב במידה והדחיסה מבוצעת באופן רקורסיבי על תיקיה מוגנת. לשם כך הוספנו דגל בעת בדיקת זדוניות הבודק האם ישנו שינוי לפחות בקובץ מוגן אחד. כופרה פוטנציאלית יכולה להשתמש בדחיסה והצפנה לקובץ בודד מריבוי קבצים ולאחר מכן למחוק את הקבצים שנקראו, ולכן נדרשת יכולת זיהוי במקרה זה.

# פרק 9: תוצאות הפרויקט

# פרק 10: ניתוח בעיות והמלצות לעתיד

# פרק 11: ביבליוגרפיה

<http://shieldfs.necst.it/continella-shieldfs-2016.pdf>

<https://grades.cs.technion.ac.il/grades.cgi?daceiecfieefbf1322285acdc9ef2d+2+236499+Spring2019+ho/WCFiles/Ransomwareattacks%20detection%20prevention%20and%20cure%20Ross%20Brewer.pdf>

<https://www.cise.ufl.edu/~traynor/papers/scaife-icdcs16.pdf>

<https://www.sophos.com/en-us/medialibrary/PDFs/technical%20papers/sophos-current-state-of-ransomware.pdf>

# נספח א: בניית הפרויקט ותפעולו

ע

# נספח ב: תוצאות זיהוי

# נספח ג: קטגוריות סיומות

בהתאם לתהליך הזיהוי חלקנו סיומות רגישות של קבצים לקטגוריות:

* קטגוריית מסמכים כגון doc, docx ועוד.
* קטגוריית טבלאות כגון xls, csv ועוד.
* קטגוריית מצגות: ppt, pptx.
* קטגוריית קבצי Outlook.
* קטגוריית קבצי טקסט.
* קטגוריית תמונות.
* קטגוריית קבצי ארכיון כגון zip, rar, 7z ועוד.
* קטגוריית מסדי נתונים.
* קטגוריית קבצי מקור (Source code).
* קטגוריית קבצי אודיו.
* קטגוריית קבצי וידאו.