פרויקט בכופרה 236499

End-point Ransomware detection and remediation

דו"ח סיום

שם הפרויקט: RansomWatch

מגישים: אביעד גפני, רפאל וורף

מנחה: אסף רוזנבאום

תוכן עניינים

פרק 1: מבוא.....2
פרק 2: הגדרת הבעיה.....3
פרק 3: סקירת ספרות 4
פרק 4: מודל האיום 5
1.4 סוגי כופרות.....5
2.4 הקורבן 5
3.4 טכניקות הדבקה.....5
4.4 יכולת התוקף ואופן הפעולה הבסיסי של כופרה מודרנית 5
פרק 5: אופן עבודת הפתרון.....7
פרק 6: תיאור תהליך הזיהוי 8
1.6 אזור מוגן 8
2.6 אנטרופיה.....8
3.6 קבצי מלכודת 8
4.6 מערכת GID 8
5.6 מדדי זיהוי.....9
פרק 7: ממשק ומימוש 11
1.7 דרייבר 11
1.1.7 אופן המימוש 11
2.1.7 מערכת GID.....11
3.1.7 איסוף מידע בדרייבר.....12
4.1.7 שמירת מידע בדרייבר.....13
2.7 תוכנת צד המשתמש.....14
1.2.7 אופן המימוש 14
2.2.7 קבצי מלכודת.....14
3.2.7 איסוף פעולות אפליקציות 15
4.2.7 זיהוי אפליקציה כזדונית.....15
5.2.7 מודל הגיבוי 16
6.2.7 אזורים מוגנים.....16
3.7 תקשורת בין הדרייבר לאפליקציה.....16
פרק 8: מכלול הבדיקות שנבדקו.....18
פרק 9: תוצאות הפרויקט.....19
פרק 10: ניתוח בעיות והמלצות לעתיד 21
פרק 11: ביבליוגרפיה.....22
נספח א: בניית הפרויקט ותפעולו.....23
נספח ב: תוצאות זיהוי 25
נספח ג: קטגוריות סיומות 26

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 באמצעות הצפנה או נעילת המחשב [1]. כיום הדרך המועדפת לתשלום לכופרות היא תשלום באמצעות מטבעות קריפטוגרפים [2]. אנו נתמקד בפרויקט בסוג השני של כופרות, כלומר בכופרות מצפינות.

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).

יעדים שנקבעו בתחילת הפרויקט:

- 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, ובמקרה של אובדן קבצים שחזורם.
- זיהוי פעילות של אפליקציות בזמן אמת.

פרק 3: סקירת ספרות

ב-2015 פתחה חברת Symantec את הדוח השנתי שלה במילים: "מעולם בהיסטוריה האנושית לא היו אנשים מרחבי העולם נתונים לסחיטה בממדים העצומים כפי שהם נתונים היום" [1]. לא תמיד כופרות היו כה מאיימות – בתחילת דרכן הן השתמשו בטכנולוגיות פשוטות יחסית. הן מנעו גישה למידע על ידי הסתרתו מהמשתמש, או השתמשו בצפנים חלשים. פרט לחולשות אלה, באותם זמנים היה קשה לקבל את תשלום הכופר מבלי להשאיר עקבות, ולכן הן לא היו מוצלחות במיוחד ולא השיגו את מטרתן [3][1].

עם המצאתו של המטבע הקריפטוגרפי חלה גם עלייה בתקיפות כופרה והתחילו מופיעות כופרות המקבלות את תשלום הכופר במטבעות קריפטוגרפים [9][2]. כיום, הכופרות הן די מתוחכמות. הן משתמשות בצפנים סימטריים כדי להצפין את הקבצים מטעמי יעילות, ומצפינות את המפתחות בעזרת צפנים א-סימטריים המנוהלים בעזרת מרכז שליטה [9][3][1]. היקפי הנזק הכלכלי שכופרות גורמות נאמדים במיליארדי דולרים [2][1], ועל כן יכולות להיות רווחיות מאוד לתוקפים. לדוגמה, בתחילת 2018 עלתה מתקפת כופרה למועצת העיר אטלנטה יותר מ-17 מיליון דולר [5].

מאז עלו הכופרות לתודעה נרחבת שוב בשנות האלפיים נעשו מאמצים למצוא פתרונות לבעיה. טכניקות הזיהוי הקיימות מנסות לאפיין תוכנה כזדונית בשתי דרכים – סטטית ודינאמית. בטכניקה הסטטית משתמשים באנטי-וירוסים ומערכות לגילוי חדירות (IDS) – מנתחים את התוכנה בצורה סטטית על פי סימנים המאפיינים תוכנות זדוניות שזוהו ככאלה בעבר. טכניקה זו היא שימושית וחזקה מאוד, אבל לבדה אינה מספיקה מכיוון שהיא מכילה את החיסרון המובנה שתוכנה זדונית שמעולם לא זוהתה בעבר עשויה שלא להתגלות [7]. יתר על כן, נעשו מחקרים המראים כי ניתן להתחמק משיטות כאלו [8].
הטכניקה הדינאמית משתמשת במערכות לומדות וניתוח סטטיסטי של התנהגות תוכנה כדי לקבוע אם היא מבצעת פעולות לא שגרתיות. חוקרים הראו שבעזרת איסוף נתונים סטטיסטים כמו כתיבה באנטרופיה גבוהה, כמות רבה של קריאות וכתיבות וכדומה, ובנייה של מסווגים המשתמשים בטכניקות של מערכות לומדות ניתן להגיע לרמות גבוהות של דיוק בזיהוי כופרות [10].

ב-2015 ערכו חוקרים לראשונה איסוף מידע המוני על התנהגות של כופרות, והציעו כי מערכת הקבצים היא נקודה אסטרטגית חשובה לניטור התנהגות כופרתית [9]. בפרויקט שלנו עסקנו בזיהוי כופרה על ידי התנהגותה מול מערכת הקבצים. הפתרון שלנו אוסף נתונים סטטיסטים על תוכנות המבצעות פעולות מול מערכת הקבצים, ומאפשר להם לעבוד כרגיל כל עוד לא זוהו כמתאפיינות בהתנהגות חריגה. החידוש שלנו הוא שניתוח זה מתבסס על מערכת שבנינו היוצרת קשרי משפחה בין תהליכים שונים הנוצרו זה על ידי זה (קשרים בין התהליכים שמערכת ההפעלה Windows לא בונה בעצמה) וכל הניתוח התנהגות מתבצע למול משפחת תהליכים כזו.

פרק 4: מודל האיום

נתייחס לאיום בהקשר של הנחות העבודה והדרישות שהצבנו בפרק 2.

1.4 סוגי כופרות

כיום נמצאים בתפוצה שני סוגים של כופרות: כופרה *נועלת* וכופרה *מצפינה*. "כופרה נועלת" מונעת גישה למכשיר הנגוע – למשל באמצעות הוספת מסך פתיחה למערכת ההפעלה שלא ניתן לעבור אותו ללא הכנסת קוד. "כופרה מצפינה" מונעת גישה למידע במכשיר הנגוע באמצעות הצפנת המידע במכשיר. שני הסוגים של הכופרות מציעים את הגישה חזרה על ידי תשלום, בדרך כלל בביטקוין [9][1].

2.4 הקורבן

ניתן לחלק לשניים את קורבנות הכופרה:

משתמשים ביתיים – במקרים רבים מתקפות הכופרה יכולות להימנע באמצעות נקיטת כללי זהירות בסיסיים, אך משתמשים רבים אינם מודעים לסכנה. למשתמשים אלה יש לעיתים קרובות מידע רגיש, קבצים, מסמכים וכדומה. מחקר שערכה חברת Symantec הראה שרבע מהמשתמשים לא מגבים מידע כלל, חצי מהם מגבים חלק מהקבצים ורק רבע מגבים את המידע שלהם אחת לשבוע. אפילו אם חלק מהמשתמשים מגבים את המידע, חלק מהכופרות ניגשות למחוק או להצפין גיבויים מקומיים [13].

עסקים וארגונים – אלה הם קורבנות בעייתיים במיוחד שכן לעיתים קרובות במקומות אלה משתמשים לא מנוסים מתפעלים את המחשבים, והם מהווים מטרה עם פרופיל גבוה. בתי חולים הם דוגמה לקורבנות כאלה. לאיבוד של מידע במקרים אלה יהיו השלכות קטסטרופליות על העסק. חלק מהמקומות דואגים לשמור גיבויים, אבל ישנן לא מעט מקומות שלא. לדוגמה, ב-2015 התגלה מקרה שבו תוקפים "התלבשו" על ארגון מסוים במשך חודשים, כשהם מצפינים את המידע שלו באופן הדרגתי יחד עם כל הגיבויים, וכשהיה להם מספיק מידע הם דרשו את הכופר [12].

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 [3]. 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 [14].

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.

כופרה נוהגת להצפין קבצים רגשים כגון מסמכים, תמונות וכדומה. בנוסף, ראינו כי כופרות נוהגות 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.

- מתן שמות לגיטימיים לתהליכים שהיא מריצה, כגון "Windows Update" וכדומה. כך גם במידה והמשתמש חושד בפעילות זדונית במערכת, יהיה קשה לזהות תהליכים אלה.
- חלוקת העבודה בין מספר תהליכים שונים ובכך לחלק את האַחְרָיוּתִיוּת (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.

מרבית הכופרות ממשיכות את עבודתן לאחר סגירתן עקב אתחול המערכת או עצירתן.

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) באזור מוגן:
   1.1. אסוף מידע אודות הפעולה.
   1.2. העברת הפעולה לניטור ולבניית תמונת מצב של התהליך.
   1.3. ניתוח תמונת המצב החדשה של התהליך.
   1.4. קבלת החלטה האם תהליך הוא זדוני:
      1.4.1. 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:

$$e = \sum_{i=0}^{255} P_{B_i} \log_2\left(\frac{1}{P_{B_i}}\right)$$

כאשר P_{B_i} הוא היחס של מספר מופעי ערך הבית i בבלוק B למספר הבתים הכולל בבלוק, כלומר נרמול של מספר מופעי הבית i בבלוק B.
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 (אפליקציה) 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.
משלב זה נתייחס לקבוצת תהליכים משותפת GID כאפליקציה.
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 אפליקציה ומהווה בסיס לתהליך הזיהוי. לכל אפליקציה מחושבים מחדש המדדים לאחר סדרת פעולות שנקלטו עבורה. 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.

על מנת להימנע מזיהויים כוזבים הוספנו דגל בעת בדיקת זדוניות הבודק האם ישנו שינוי לפחות בקובץ מוגן אחד, כל עוד אפליקציה אינה משנה קבצים מוגנים אנו נניח שאינה זדונית. לא ניתן להשתמש בתכונה זו למטרת תקיפה מכיוון שאנו עוקבים אחר כל תהליך הרץ תחת אפליקציה תחת GID יחיד.

למלכודות ניתן משקל רב יותר עבור חלק ממדדי הזיהוי, פעולה על מלכודת נספר כנגיעה ב-10 קבצים במקום קובץ בודד.

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 לפי שיטת שנון של פעולות הכתיבה ביחס לממוצע האנטרופיה של פעולות הקריאה. בדיקה מבוצעת לפי נוסחה: $averegeWriteEntropy - averegeReadEntropy > \frac{maxEntropy - averegeReadEntropy}{2}$	הצפנות מאופיינות באנטרופיה גבוהה. השוואת ממוצע הכתיבה לממוצע הקריאה מאפשר לצמצם זיהויים כוזבים של אפליקציות העובדות עם קבצים בעלי אנטרופיה גבוהה.	בהתאם לנוסחה
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 פורט.

1.7 דרייבר

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

1.1.7 אופן המימוש

בחרנו לממש את הדרייבר בארכיטקטורת Windows Minifilter Driver [11].
דרייבר מסוג זה מאפשר רישום לפעילויות במערכת הקבצים על ידי פונקציות שמוגדרות על ידי כותב הדרייבר. פעולות כגון כתיבה לקובץ, פתיחת קובץ, קריאה, סגירה, ושינוי שם קובץ. לכן הוא מאפשר לנטר בזמן אמת אחר פעולות לפני ואחרי ביצוען ולהשפיע עליהן במידת הצורך.

בנוסף, על מנת להתמודד עם כופרות המחלקות את עבודתן על מספר תהליכים שונים יצרנו מערכת GID (Group Identifier) 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 (3.7).
בעת ניטור פעולה אנו מייצרים הודעה מסוג DRIVER_MESSAGE (3.7) וממלאים אותה בהתאם.

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: הקובץ מייצג תיקייה, מדובר בקובץ חדש או דריסה של קובץ קודם או אירוע פתיחת קובץ בלבד, 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 מחיקת קובץ אנו מדווחים בנוסף בהודעה שהקובץ עובר מחיקה. עבור שינוי שם אנו בודקים מהי הסיומת החדשה של הקובץ, ובנוסף מדווחים האם הקובץ מוכנס לאזור מוגן או האם הקובץ מוצא מהאזור המוגן. מידע זה מצורף להודעה עבור הפעולה.

4.1.7 שמירת מידע בדרייבר

כל המידע בדרייבר 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 שאפליקציית צד המשתמש מעבירה בעת שליחת ההודעה. לאחר העברת הודעה לצד משתמש ההודעה נמחקת מהרשימה המעגלית.
רשימה זו ממומשת כרשימה מעגלית (LIST_ENTRY), כל הפעולות על רשימה זו מתבצעות תחת 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), כל הפעולות על רשימה זו מתבצעות תחת מנעול directoriesSpinLock.

- GidsList - רשימה מעגלית
משמש לשמירת מבני GID, המכילים את מספרי התהליכים הנמצאים בתוכם.
מבנה זה משמש לשחרור זיכרון של הדרייבר למעקב.
רשימה זו ממומשת כרשימה מעגלית (LIST_ENTRY), כל הפעולות על רשימה זו מתבצעות תחת מנעול GIDSystemLock.

- 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. כל הפעולות על טבלה זו מתבצעות תחת מנעול GIDSystemLock.

- PidToGids – טבלת גיבוב
מיפוי של PID לערך ה-GID אליו הוא שייך. כל הפעולות על טבלה זו מתבצעות תחת מנעול GIDSystemLock.

2.7 תוכנת צד המשתמש

אחראית על מעקב אחר אפליקציות וסיווגן לפי מדיניות הזיהוי, שליחת בקשת לעצירת אפליקציות לדרייבר, קביעת האזורים המוגנים על ידי המערכת והעברת מידע זה לדרייבר, יצירת מלכודות בהתאם למדיניות הזיהוי וגיבוי ושחזור של קבצים שנפגעו.

1.2.7 אופן המימוש

בחרנו לממש את תוכנת צד לקוח בשפת C++/cli, המשלבת את שפת C++ עם סביבת .NET, ורצה תחת המכונה הווירטואלית CLR [16], בחרנו לממש בסביבה זו מטעמי נוחות.

תוכנת צד המשתמש פועלת בעזרת שלושה חוטים עיקריים:

- חוט 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 המשמשים לקבלת פעולות מול מערכת הקבצים של תהליכים עם GID מהדרייבר. חוטים אלו אחראים על איסוף מידע על אפליקציות הרצות, הוספת רשומה ל- GID חדש במידה ואין כזה, בדיקת תקינות לאפליקציות מול המדדים שנקבעו (5.6), ושליחת בקשה לסיום קבוצת ה-GID במקרה ואפליקציה זוהתה כזדונית. לאחר זיהוי אפליקציה כחשודה מתבצע ניסיון שחזור (5.2.7) של הקבצים שזוהו כנגועים או קבצים שנמחקו מתוך האזורים המוגנים (1.6).
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 מלכודת

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

עבור כל סיומת בתיקיה הנוספת 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 איסוף פעולות אפליקציות

עבור כל אפליקציה בעלת 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 ומספרן של התיקיות המכילות מלכודות בהן שונו קבצי מלכודת.
- מספר פעולות כתיבה בעלות אנטרופיה (2.6) גבוהה.
- קבצים שנמחקו לפי מזהה קובץ (FILE_ID).
- קבצים שנוצרו לפי מזהה קובץ.
- קבצים שבוצעה מולם פעולת קריאה, כתיבה או שינוי שם לפי מזהה קובץ.
- קבצים שנכתב אליהם מידע לפי מזהה קובץ.
- קבצים שנקרא מהם מידע לפי מזהה קובץ.
- קבצים ששם שונה לפי מזהה קובץ.
- קבצי מלכודות ששונו לפי מזהה קובץ.
- 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.

המבנה שתואר עבור כל אפליקציה נשמר במילון אליו ניתן לגשת משני חוטי הניטור וטיפול שתיארנו באופן המימוש (1.2.7). מפתח המילון הוא ה- GID של האפליקציה וערכו הוא מצביע למבנה GProcessRecord שתיארנו קודם.

איסוף המידע נעשה באופן הבא: האפליקציה שולחת בקשה לפעולות אחרונות של אפליקציות בעלות 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-GID זה מסומן לבדיקה בהתאם למודל הזיהוי.

4.2.7 זיהוי אפליקציה כזדונית

לאחר קבלת הודעות עבור פעולות של אפליקציות מול מערכת הקבצים מהדרייבר, נבדק האם ניתן לקבוע אם הן זדוניות.

עבור כל אפליקציה בעלת GID, נבדק האם המידע שנשמר לגביה תקין ביחס למדדי הזיהוי (5.6).
במידה ו- GID מזוהה כזדוני ביחס למדדים, האפליקציה שלנו אוספת את המדדים שהופרו ומדווחת עליהם, שולחת בקשה לדרייבר לעצור את אפליקציות זדוניות לפי המזהה GID, מייצרת קובץ דיווח ומנסה לשחזר קבצים שנפגעו בעקבות אפליקציות זדוניות.

5.2.7 מודל הגיבוי

מודל הגיבוי הינו מבוסס ענן Azure, מתבסס על אחסון מבוסס Blob [17].
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 וזוהתה אפליקציה זדונית.
לאחר זיהוי אפליקציה כזדונית נעשה שחזור לקבצים אשר זוהו כקבצים אותם שינתה האפליקציה 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 שהאפליקציה החלה לפעול לראשונה, ובמידה וקיים קובץ שכזה התוכנה מורידה אותו ומחליפה בקובץ ששונה.

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

6.2.7 אזורים מוגנים

את האזורים המוגנים הגדרנו בפרק 6.

הוספה והסרה של תיקיות האזורים המוגנים נעשית דרך תפריטי המשתמש.
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 לחצן Remove Directory, בלחיצה על כפתור זה נפתח חלון המאפשר בחירה של התיקייה אותה 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 האפליקציה.
- MESSAGE_KILL_GID – בקשה לעצירת אפליקציה בעלת 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 חוצץ ומעבירה אותו לדרייבר בצירוף עם ההודעה ומחכה למילוי החוצץ. בתגובה לבקשה זו הדרייבר ממלא את החוצץ 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.

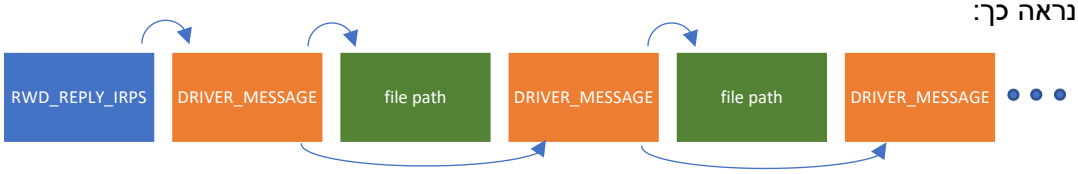
מבנה DRIVER_MESSAGE:

- סיומת של הקובץ עליו בוצעה הפעולה.
- מזהה ייחודי של הקובץ [15].
- מספר הבתים ששומשו במסגרת הפעולה (רלוונטי לכתיבה וקריאה בלבד).
- אנטרופיה במידה וחושבה (רלוונטי לכתיבה וקריאה בלבד).
- PID של התהליך שביצע את הפעולה.
- GID של האפליקציה מבצעת 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).
- מצביע לשם הקובץ (שם הקובץ מוקצה ומועבר בנפרד).
- מצביע למבנה DRIVER_MESSAGE הבא במידה ויש אחד כזה.

מילוי החוצץ בעת הודעה מסוג MESSAGE_GET_OPS מבוצע באופן הבא:

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 החוצץ.

לאחר המבנה RWD_REPLY_IRPS מוכנסות הודעות מסוג DRIVER_MESSAGE, ובצמוד לכל הודעה מסוג DRIVER_MESSAGE מוצמד שם הקובץ עליו מבוצעת הפעולה. לאחר מילוי החוצץ הוא נראה כך:



פרק 8: מכלול הבדיקות שנבדקו

בדיקת התוכנה והדרייבר נעשתה מול כופרות קיימות ומול פעולות לגיטימיות מצד משתמש כגון דחיסה של קבצים בעזרת תוכנות, מחיקת קבצים, קריאה, כתיבה ושינוי שם קובץ.
בעת ניסוי התוכנה זיהינו בהצלחה מספר כופרות. ביניהן:
Katyusha, Cerber, Mamba, Jigsaw, Vipasana, Wannacry.

את הדוגמיות השגנו מתוך מאגר ב-GitHub [4].

בנספח ב' מופיעים מספר זיהויים של הכופרות הנ"ל יחד עם תיאור קצר.

ערכנו ניסויים לבדיקת זיהויים כוזבים על ידי יצירת קבצים דחוסים (zip, 7z) מקבצי האזורים המוגנים. הרעיון הוא שדחיסה היא פעולה שהתנהגותה דומה לכופרה: היא (עשויה) לעבור על מלכודות בתיקיות שונות, לייצר 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. 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

העבודה כללה למידה מעמיקה על סוגי כופרות שונים, התנהגות כופרות בעת עבודה מול המערכת והגדרת מודל איום של כופרה. בהתבסס על התנהגות כופרה במערכת בזמן ריצתה, ולמידת הפתרונות הקיימים לזיהוי ניסחנו מודל זיהוי ובדקנו אותו מול כופרות שונות. כחלק מביצוע הפרויקט הוכנו פוסטר המפרט את עיקרי הפרויקט, דו"ח סיום זה ומצגת סיום.

היעדים שהוצבו בתחילת הפרויקט, המפורטים בפרק 2, הושגו. הפתרון שלנו מסוגל לזהות כופרות מסוגים שונים, ישנות וחדשות, בזמן אמת ולפי פעילות אפליקציות. לאחר מכן, הוא יודע לשחזר מידע שנפגע כתוצאה מפעילות כופרתית. הפתרון שלנו יודע לטפל בכופרה ללא התערבות המשתמש, פרט לבחירת האזורים עליהם המשתמש מעוניין להגן בהפעלה ראשונית.
לשם השלמת מודל הזיהוי ביצענו בהצלחה פעולות לגיטימיות של המשתמש מול המערכת כמו דחיסת קבצים, הצפנה ופעולות של משתמש כגון מחיקת קבצים, עריכה ויצירה. במקרים מסוימים המערכת תיתן זיהויים כוזבים (false positive) לפעילות של משתמש, למשל כאשר משתמש דוחס באופן מוצפן קבצים רבים בעזרת אפליקציה כגון zip, שבמסגרת התהליך האפליקציה מוחקת קבצים שנדחסו. בחרנו להתייחס לפעילות כזו כהתנהגות זדונית מכיוון ששיטה זו עשויה לשמש למימוש כופרה.

במהלך הפרויקט עלו קשיים ולעיתים חריגות מלוח הזמנים שתוכנן תחילה עקב מימוש חלק מהפתרון באמצעות דרייבר. זיהוי פעולות מול מערכת הקבצים, תכנון הדרייבר, בדיקתו ותפעולו לקח זמן רב ביחס לשאר חלקי הפרויקט.
בתחילת הפרויקט השתמשנו במודל תקשורת שונה ממה שפורט בדו"ח זה (פרק 7). במודל הראשוני הדרייבר היה יוזם התקשורת מול האפליקציה. הדרייבר שלח מידע על פעולות של תהליך כלשהו מול מערכת הקבצים לאפליקציית צד המשתמש, והיה ממתין לתשובה האם לאשר את הפעולה. מודל זה גרר בעיות תקשורת ומערכת כגון: איבוד מידע אודות פעולות, קריסת האפליקציה ולעיתים אף לקריסה או קפיאת המערכת כתוצאה מעצירת פעולות קריטיות. במהלך הפרויקט מימשנו מחדש מודל תקשורת זה למודל הנוכחי כפי שתואר בפרק 7.

להלן פירוט היתרונות והחידושים במסגרת הפתרון:

- מודל הזיהוי שכתבנו ניתן להתאמה לכופרות מסוגים שונים, מכיוון שהוא מנתח התנהגות בצורה דינאמית. מודל זה מאפשר לנו להתמודד עם כופרות חדשות, ובמידה ומודל האיום משתנה ניתן לשנות בקלות את מדדי הזיהוי ולהתאימו.
- מודל הזיהוי מייצר קבצי מלוכדת בצורה חכמה על מנת להקשות על זיהויים על ידי כופרות ונותן אינדיקציה טובה על התנהגות אפליקציות המשנות אותם.
- על מנת לעקוב אחר אפליקציות מרובות תהליכים יצרנו עבור כל אפליקציה מזהה GID ייחודי, מזהה זה משותף לכל תהליך של האפליקציה. כופרות המשתמשות במספר תהליכים על מנת להסוות פעילות או משתמשות בתהליכים נוספים על מנת לגבות את עבודתן ושחזורן במקרה של עצירה מטופלות ומבוצעת עצירה כוללת של כל תהליכי האפליקציה.
- מעקב אחר פעולות של אפליקציות מול מערכת הקבצים ממומשת כדרייבר, המאפשר מעקב בזמן אמת אחר כל פעילות הקורית מול מערכת הקבצים. מימוש זה מאפשר זיהוי מהיר יותר של אפליקציות ועצירתן לפני שייגרמו נזק משמעותי יותר.
- הפתרון שלנו כולל גיבוי של אזורים מוגנים ושחזור קבצים במקרה של שינוי קבצים על ידי אפליקציה שזוהתה כזדונית, בכך אנו מצמצים את הנזק שכופרה יכולה לעשות במערכת עקב ריצתה.

חסרונות במסגרת הפתרון:

- אין ניתוח סטטי של אפליקציות במודל הזיהוי, ניתוח סטטי יכול לסייע בזיהוי אפליקציות זדוניות לפני הרצתן ובכך למנוע נזק אפשרי.
- גיבוי בענן – גיבוי עלול לקחת זמן רב אם רוחב הפס לא מספיק גדול והמידע המוגן גדול מאוד, בנוסף גיבוי מסוג זה חשוף לבעיות תקשורת, כאשר אין חיבור לרשת לא התבצעו גיבויים ובבטח שלא שחזור. במידה והכופרה מצליחה להעלות הרשאות גישה במערכת, היא יכולה להשיג את מפתח ההתחברות לאחסון בענן ולהרוס גיבויים, על ידי סריקת מרחב זיכרון האפליקציה.
- בעקבות המעקב אחר פעילות אפליקציות מול מערכת הקבצים ושמירת מידע, נוצר עיכוב על פעולות המבצעות אפליקציות, לא בוצעו אומדנים מדויקים לרמת ההשפעה.
- שימוש במלכודות עלול להוסיף קבצים רבים במערכת במקומות שונים, למשתמש אין שימוש בקבצים אלו.

פרק 10: ניתוח בעיות והמלצות לעתיד

לצורך שיפור הפרויקט מומלץ לבצע שיפורים:

- מודל הזיהוי:
  - שימוש במודל זיהוי סטטי:
    ראינו כי כופרות מנסות לבצע מחיקה של גיבויים במערכת. על מנת לשפר את מודל הזיהוי ניתן לזהות מחיקת גיבויים ולסרוק את קובץ ההרצה אחר תבניות ידועות כזדוניות.
    שמירת חתימות של כופרות שזוהו ובניית מסד נתונים של חתימות אלו אשר ישמש לזיהוי עתידי.

  - שיפור במודל הזיהוי הדינאמי:
    ניתן להוסיף אומדן של זמנים למודל הזיהוי. ראינו כי כופרות בדרך כלל מבצעות סערות קריאה/כתיבה (Read/Write burst), התנהגות השונה מאפליקציות רבות, במטרה לגרום לנזק רב ככל האפשר לפני זיהוין במערכת.
    בתחילת הריצה של כופרה מתבצעת החלפת מפתחות מול מרכז שליטה, ניתן לזהות את ניסיון החלפת המפתחות בעזרת דרייבר המנטר פעילות רשת.

  - קיימת שיטת תקיפה מיוחדת [18] אשר לא ניתן לה מענה בפתרון שלנו ולא נמצא כופרות המשתמשות בשיטה זו.
    בשיטת התקיפה כופרה אפשרית פותחת HANDLE (רפרנס לקובץ) ל-Volume בו קיים מידע שהיא רוצה להצפין. במאמר מתוארת שיטת תקיפה דרך פעולה זו המאפשרת לקרוא ולכתוב קבצים שלא דרך ה-Minifilter דרייבר או hooks אחרים.

- טיפול בכופרות שזוהו:
  - על מנת למנוע מכופרות להיטען מחדש לאחר עצירתן נדרש לבצע: מחיקה של קבצים שנוספו לתיקיית startup של המשתמש (קבצים בתיקייה זו נפתחים בעליית מערכת ההפעלה), ניקוי ה-Registry משינויים שיצרה הכופרה וביטול משימות שיצרה ב- Task scheduler. Task scheduler אף יכול לשמש כופרה להורדתה מחדש לאחר שנמחקה.

  - בדרייבר אנו מבצעים מעקב אחר יצירת תהליכים ובכלל ניתן לעקוב גם אחר טעינת קוד הרצה לזיכרון (image files), מידע היכול לשמש לשם ניקוי הכופרה.

- ממשק ומימוש:
  - בפתרון העכשווי נדרש לטעון את הדרייבר ידנית ולפתוח את האפליקציה. נדרש כי טעינת הדרייבר ופתיחת האפליקציה תקרה בעת עליית המערכת ללא אפשרות לסגירתם וללא צורך בהתערבות המשתמש.

  - מיקום המלכודות והתיקיות המוגנות אינו נשמר בין הרצות של האפליקציה, על האפליקציה לדעת לשחזר ריצה קודמת שלה ולבצע מעקב מחדש אחרי תיקיות שנבחרו בעבר להגנה. מחוסר זמן לא ביצענו שמירת נתונים.

  - הפתרון שלנו מבצע גיבוי ברמה בסיסית בעת הוספת תיקייה להגנה, בעת ההוספה נעשה גיבוי ראשוני ואין גיבויים נוספים לתיקייה. לשם השלמת הפתרון נדרש לבצע גיבויים נוספים. כמו כן, ניתן לייעל את תהליך הגיבוי באמצעות גיבויים חלקיים, על ידי גיבוי רק קבצים אשר זוהה ששונו.

  - על הפתרון להגן על הקבצים בהם הוא משתמש, כגון dll של Azure וסיריאליזציה של מידע הנשמר על ידי האפליקציה.

פרק 11: ביבליוגרפיה

[1] K. Savage, P. Coogan, H. Lau. Security Respone – The evolution of ransomware. 2015.

[2] S. Cook. 2017-2019 Ransomware statistics and facts. In *Comparitech,* 2019.

[3] R. Brewer. Ransomware attacks: detection, prevention and cure. In *LogRhytm*, 2016.

[4] theZoo – A Live Malware Repository

[5] S. Deere, The Atlanta Journal-Constitution. CONFIDENTIAL REPORT: Atlanta's cyber attack could cost taxpayers $17 million. 2018.

[6] A. Young; M. Yung. Cryptovirology: extortion-based security threats and countermeasures. IEEE Symposium on Security and Privacy. pp. 129–140.

[7] N. Scaife; H. Carter; P. Traynor; K. R.B. Butler. CryptoLock (and Drop It): Stopping Ransomware Attacks on User Data. 2016.

[8] S. Jana, V. Shmatikov. Abusing File Processing in Malware Detectors for Fun and Profit. 2012.

[9] A. Kharraz, W. Robertson, D. Balzarotti, L. Bilge, E. Kirda. Cutting the Gordian Knot: A Look Under the Hood of Ransomware Attacks. 2015.

[10] A. Continella, A. Guagnelli, G. Zingaro, G. D. Pasquale, A. Barenghi, S. Zanero, F. Maggi. ShieldFS: A Self-healing, Ransomware-aware Filesystem. 2016.

[11] Microsoft Inc. File System Minifilter Drivers. 2017.

[12] Virus bulletin. 'RansomWeb' ransomware targets companies' databases.

[13] Norton from Symantec. Symantec Backup Survery. 2009.

[14] B. Brenner. WannaCry: the ransomware worm that didn't arrive on a phishing hook. 2017.

[15] Microsoft Inc. FILE_ID_INFO structure. 2018

[16] Microsoft Inc. .NET Programming with C++/CLI. 2018

[17] Microsoft Inc. Introduction to Azure Blob storage. 2019

[18] R. Van Gorp. Low-level writing to NTFS file systems. 2018

נספח א: בניית הפרויקט ותפעולו

תלויות

- Microsoft Visual Studio (עבדנו עם גרסת 2019. גרסת 2017 אמורה להתאים גם, אך לא נבדק).
- התקנת Windows driver kit והוספתו ל-Visual studio בעת התקנתו (לאשר במהלך ההתקנה להוסיף תוסף ל- Visual studio).
- NET 4.7.2. ומעלה.
- Windows Software development kit.
- Visual studio build tools גרסה 1.4.2.
- C++/cli support for Visual studio build tools 1.4.2.
- Microsoft WindowsAzure storage גרסה 9.3.3.

בניית הדרייבר והאפליקציה

1. לשם בניית הפרויקט, תחילה נדרש לטעון את הפרויקט ב-visual studio, קובץ טעינת הפרויקט נקרא RWatch.sln.
2. לאחר טעינת הקובץ לחץ build לבניית הפרויקט, יש לשים לב כי הפרויקט מקומפל לארכיטקטורת x64 בלבד.

הרצת הפתרון

1. על מנת להריץ את הפתרון ומכיוון שהדרייבר חתום בחתימת testSign שאינה מאפשרת טעינת דרייבר לסביבה שאינה ניסיונית, נדרש להעביר את המכונה עליה מעוניינים להריץ למצב טסט. ניתן לבצע זאת על ידי הרצת הפקודה הבאה ב-CMD תחת הרשאות Admin:

```
bcdedit /set testsigning on
```

לאחר ביצוע פעולה זו נדרש לבצע restart למערכת.

2. העבר את קבצי האפליקציה למכונה.
לשם העברת האפליקציה יש להעביר את הבינארי שקומפל (Application.exe) ובנוסף להעביר את קובץ ה-dll שנוצר בתיקייה של הקובץ הבינארי (Microsoft.WindowsAzure.Storage.dll). קבצים אלו נוצרים תחת :

```
<project location>/x64/Release
```

3. העבר את קבצי הדרייבר למכונה.
לשם העברת הדרייבר יש להעביר את הקבצים: FsFilter.cer, FsFilter.inf, FsFilter.sys.
קבצים אלו נוצרים תחת התיקייה:

```
<project location>/x64/Debug
```

4. טעינת הדרייבר.
על מנת לטעון את הדרייבר יש להריץ את הפקודה הבאה:

```
RUNDLL32.EXE SETUPAPI.DLL,InstallHinfSection DefaultInstall 132 <Driver files location>\FsFilter.inf
```

הפקודה מוסיפה את הדרייבר ל- Service control של מערכת ההפעלה.
הפקודה מסתמכת על כך שכל קבצי הדרייבר שהועברו בסעיף 3 נמצאים באותה תיקייה.

5. הרצת הדרייבר.
לאחר הטעינה ניתן להריץ את הדרייבר בעזרת הפקודה:

```
sc start FsFilter
```

ניתן לעצור דרייבר לאחר הרצתו בעזרת הפקודה:

```
sc stop FsFilter
```

מחיקת הדרייבר לשם התקנתו מחדש או ניקוי הדרייבר מהמערכת בעזרת הפקודה:

```
sc delete FsFilter
```

6. הרץ את האפליקציה שהועתקה, מומלץ להריץ עם הרשאת Admin.

בדיקת התוכנה

1. הרץ את האפליקציה והוסף אזורי הגנה בעזרת לחצן Add directory. יש להמתין להודעת סיום הוספת אזור מוגן.
2. הרץ כופרה.
3. לאחר זיהוי כופרה נוצר קובץ תיעוד (מיקומו מדווח ב-LogViewer) ובנוסף מדווח בתוכנה עצמה ב-LogViewer על מדדי זיהוי ושינויים להם גרמה הכופרה באזורים מוגנים.

קוד מקור (GitHub)

https://github.com/RafWu/RansomWatch

נספח ב: תוצאות זיהוי

בקישור הבא ניתן לצפות בזיהויים של WannaCry ו-Jigsaw, כופרות שזוהו על ידי הפתרון שלנו:

https://github.com/RafWu/RansomWatch/tree/Resources/Videos

ננתח את הזיהויים.

WannaCry:

כופרה זו מבצעת קריאה של קבצים רבים ובכלל מבצעת פעולות Listing רבות לתיקיות מתהליך אחד, ומתהליך נוסף מבצעת הצפנה של קבצים, הכופרה מייצרת קובץ נוסף לכל קובץ שאותו היא מצפינה, כותבת לשם את התוכן המוצפן ולאחר מכן משנה את הקובץ המקורי.

כופרה זו זוהתה על ידי כך שהיא: יצרה קבצים רבים ביחס לקבצים שהיא קראה, קראה וכתבה לסיומות מקטגוריות שונות (נספח ג'), שינתה סיומות לקבצים שהצפינה, ובכלל שינתה שמות של קבצים. פעולות אלו יחד גרמו לזיהוי WannaCry ככופרה.

Jigsaw:

כופרה זו מבצעת קריאה של קבצים, יוצרת קובץ נוסף ליד הקבצים עם שם דומה וסיומת fun, כותבת את הקובץ שנקרא מוצפן לקובץ החדש ולאחר מכן מחיקת הקובץ המקורי.
כופרה זו זוהתה על ידי שהיא: יצרה קבצים, מחקה קבצים, נגעה בקבצים בעלי קטגוריות שונות, כתבה לבערך אותה כמות קבצים כמו מה שהיא קראה, כתבה באנטרופיה גבוהה ביחס לקריאה. פעולות אלו יחד גרמו לזיהוי Jigsaw ככופרה.

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.