# פרק 0: הקדמה

## פרטים

* **שם הקובץ**: bc98c8b22461a2c2631b2feec399208fdc4ecd1cd2229066c2f385caa958daa3
* **גודל**: 180 קילו-בייט.
* **סוג**: בינארי

## מטרות

* להבין מה הקובץ החשוד יכול לעשות.
* האם הוא מבצע תקשורת החוצה.
* האם כותב / מוחק קבצים ל/ממערכת ההפעלה.
* כיצד לחסום את הקובץ.
* יצירת חוקים לחסימת הקובץ במערכת הגנה ארגונית.
* הקמת EDR בענן וקבלת התראות בזמן אמת על הקובץ החשוד.

## טכניקות

בשביל להבין את הקובץ החשוד (קובץ בינארי בלתי ניתן לקריאה (Non-Human Readable)), אשתמש בשתי טכניקות:

* ניתוח סטטי – מערב כלים שונים, אשר כל אחד נותן פיסות מידע שונות על הקובץ. ניתוח סטטי מתבצע ללא הרצה של הקובץ. הכלים השונים יקנו תמונה ברורה יותר לאופי הקובץ ומה הוא יכול לעשות מבלי להריץ אותו על מערכת ההפעלה.
* ניתוח דינאמי – הרצת הקובץ הבינארי עם כלי ניטור שונים בשביל לעקוב אחרי הפעילות של הקובץ על המערכת הן בתעבורה ברשת והן על קבצים, קבצי רג'יסטרי, תהליכים וכו'...
* מיתון ודרכי הגנה – לאחר שלבי הניתוח, צריך לייצר כללי הגנה בכדי למנוע מהקובץ לחדור לרשת ארגונית ו/או לתפוס הפעלת הקובץ אצל משתמשי קצה באם חלילה חדר לארגון. מפורט [בנספח א'](#_נספח_א':_דוגמה) בסוף הקובץ.

## גילוי נאות

***לא ידעתי כלום על ניתוח מאלוורים עד לזמן שהתחלתי עם האתגר הזה. כל מה שרשמתי במסמך זה הגיע מתוך לימוד עצמי וחקירה. מפאת זמן ועצם העובדה שהמאלוור לא פעיל יותר, לא חקרתי אותו עם דיבאגרים כמו IDA ו-OllyDBG כחלק מהניתוח הדינאמי הרצוי. מה שמאוד התחברתי אליו הוא הקמת EDR ברשת הביתית וניתוח בזמן אמת כאילו אני יושב בארגון ומקבל התראות בזמן אמת ממשתמשי קצה.***

תוכן עניינים:

[פרק 0: הקדמה 1](#_Toc64544701)

[פרטים 1](#_Toc64544702)

[מטרות 1](#_Toc64544703)

[טכניקות 1](#_Toc64544704)

[גילוי נאות 1](#_Toc64544705)

[פרק 1: ניתוח סטטי 4](#_Toc64544706)

[אילו פונקציות הקובץ משתמש 4](#_Toc64544707)

[ADVAPI32.dll 4](#_Toc64544708)

[USER32.dll 5](#_Toc64544709)

[IMM32.dll 5](#_Toc64544710)

[RASAPI.dll 5](#_Toc64544711)

[KERNEL32.dll 5](#_Toc64544712)

[זיהוי סוג הקובץ,(PE File Headers & Sections) : 6](#_Toc64544713)

[סיכום 7](#_Toc64544714)

[חתימות מאלוור (Malware Hashing) 8](#_Toc64544715)

[יצירת חתימות רלוונטיות עם HashMyFiles: 8](#_Toc64544716)

[PEStudio Hashes: 8](#_Toc64544717)

[Sections MD5 Hashes 8](#_Toc64544718)

[ניתוח סטטי ב-VirusTotal: 9](#_Toc64544719)

[Relations Tab 9](#_Toc64544720)

[ניתוח סטטי ב-Any.run ע"י חיפוש ה-MD5 10](#_Toc64544721)

[חליצת סטרינגס (STRINGS) 11](#_Toc64544722)

[PE Header 12](#_Toc64544723)

[Resources 12](#_Toc64544724)

[דרכים לחסימת המאלוור 13](#_Toc64544725)

[Yara rules 13](#_Toc64544726)

[זיהוי על פי Indicators of Compromise 14](#_Toc64544727)

[פרק 2: ניתוח דינאמי 15](#_Toc64544728)

[שימוש ב-Malware Sandbox 15](#_Toc64544729)

[Any.Run 15](#_Toc64544730)

[הרצה על מכונה וירטואלית 19](#_Toc64544731)

[ניטור עם Process Monitor 19](#_Toc64544732)

[ניתוח עם Process Explorer 24](#_Toc64544733)

[השוואת שינויים ברג'יסטרי עם RegShot 30](#_Toc64544734)

[זיוף רשת 31](#_Toc64544735)

[תפיסת קבצים שנמחקים אוטומטית 32](#_Toc64544736)

[Microsoft Network Monitor 33](#_Toc64544737)

[סיכום דברים / ממצאים 34](#_Toc64544738)

[נספח א': דוגמה להקמת EDR בענן, קביעת חוקי D&R, הרצת המאלוור במכונה אחרת וזיהוי בזמן אמת 36](#_Toc64544739)

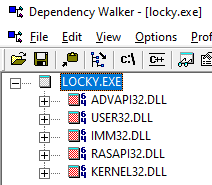
[יצירת חוקי Detection & Response (D&R) 36](#_Toc64544740)

[הרצת המאלוור במכונה ובדיקת אמינות החוקים שנוצרו 39](#_Toc64544741)

# פרק 1: ניתוח סטטי

## אילו פונקציות הקובץ משתמש

פתיחת הקובץ עם Dependency Walker ו-PEStudio, כלים לניתוח קבצים והצגת ספריות שהוא מייבא. מתוך המודולים אפשר לראות את הפונקציות השונות שמיוצאות מהספרייה בתצורת עץ.



### ADVAPI32.dll

הפונקציות מהמודול הזה מספרות לנו שיש שימוש ברג'יסטרי. פונקציות כגון RegCreateKeyExW, RegDeleteKeyA, RegSetValueA וכדומה.

MITRE-ATT&CK:

[**MITRE-Technique T1112**](https://attack.mitre.org/techniques/T1112/) פונקציות אלה מזוהות ע"י **MITRE** כאלה שמשומשות בטכניקת [MITRE T1112](https://attack.mitre.org/techniques/T1112/). ***"תוקף אפשרי יכול לפעול עם מפתחות רג'יסטרי כדי להסתיר מידע על קונפיגורציה, למחוק מידע כחלק מניקוי, טשטוש עקבות, או כחלק מטכניקות אחרות כדי לסייע בעבודה עקבית והרצה של התוכנה****.****"***

[**MITRE Technique T1106**](https://attack.mitre.org/techniques/T1106/) פונקציית CreateProcessAsUserA מזוהה כשמשומשת כחלק מטכניקה T1106 ע"פ MITRE-ATT&CK. *"תוקפים עשויים לקיים אינטראקציה ישירה עם ממשק תכנות היישומים המקורי של מערכת ההפעלה (API). ממשקי API מקומיים מספקים אמצעי מבוקר לקרוא לשירותי מערכת הפעלה ברמה נמוכה בתוך הליבה, כגון אלה הקשורים עם חומרה / מכשירים, זיכרון ותהליכים. ממשקי API מקוריים אלה ממונפים על ידי מערכת ההפעלה במהלך אתחול המערכת (כאשר רכיבי מערכת אחרים עדיין לא מאותחלים) וכן ביצוע משימות ובקשות במהלך פעולות שגרתיות."*

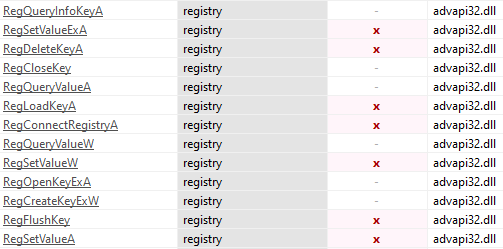


Figure: מודולים בשימוש ע"י הקובץ החשוד

שימוש ב-AddAce מוסיפה רשומה ל-**Access Control List** אשר מכילה סטים של זכויות גישה המזהות למי הזכויות מונגשות, חסומות או משונות. [[MSDN](https://docs.microsoft.com/en-us/windows/win32/secgloss/a-gly)]



עוד פונקציה מעניינית היא הפונקציה EnctyptFileW, אשר **מצפינה קבצים או תיקיות**. אם תיקייה הוצפנה כל הקבצים תחתיה מוצפנים גם כן. פונקציה זו היא רמז ראשוני לכך שהקובץ החשוד הוא מאלוור מסוג **Ransomware**.



### USER32.dll

היבואים מ-***USER32.dll*** אומרות שקיים GUI לתוכנה, אך לא בהכרח שהוא מוצג למשתמש. ShowWindow, DrawText, וכו'... כמו כן יש שימוש בפונקציית mouse\_event אשר עוקבת אחרי תנועות ולחיצות עכבר. רומז על כך שקיים **Keylogger** על התוכנה.

### IMM32.dll

ספרייה שמשומשת ע"י **Microsoft Windows Input Method Manager (IMM**), חייבת לפעול בשביל שווינדוס יפעל כראוי. הקובץ משתמש בספרייה כדי **לדמות לחיצות קיצורי מקשים** (*אשר כביכול קיימים בקובץ (ראה* ***Accelerators******Objects***)) **כאילו אותו משתמש לחץ עליהם**. דוגמה לפונקציה מסוג כזה היא ImmSimulateHotKey:



### RASAPI.dll

***RASAPI.dll*** קשור לכל הנוגע בהתחברות **RAS** **(*Remote Access Service*)** עם סרבר וקליינט. המאלוור משתמש בפונקציית RasDial האחראית להקמת ערוץ תקשורת RAS בין קליינט לסרבר. המידע בחיבור יכלול בתוכו פרמטר CALLBACK ואמצעי זיהוי של המשתמש (*קליינט*).



Figure I: שתי הפונקציות מתוך הספרייה RASAPI

### KERNEL32.dll

שלושת הפונקציות המיובאות מתוך KERNEL32 מספרות לנו שהקובץ אולי עושה **מניפולציות תהליכים ועל קבצים במערכת**.

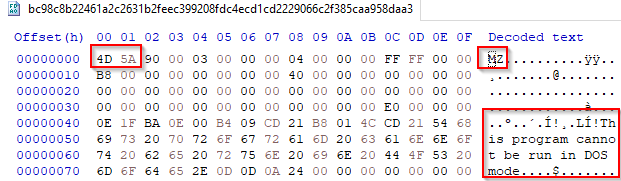
* GetLongPathNameA הופכת את הנתיב שניתן לנתיב בפורמט הארוך שלו. (Converts specified path to its long form).
* WriteFileGather לוקחת מידע מתוך מערך וכותבת אותו לתוך קובץ.
* PulseEvent מסנכרנת תהליכים. מציבה את אובייקט האירוע שצוין למצב מסומן ואז מאפסת אותו למצב המקורי (?) לאחר ששוחררו כל מספר התרדים (Threads) הממתינים המתאימים.



## זיהוי סוג הקובץ,(PE File Headers & Sections) :

קובץ **Portable Exe**, מזוהה ע"י שני הבייטים הראשונים 4D 5A (MZ) + המחרוזת:

“This program cannot run in DOS mode”.



קובץ **PE** מכיל מידע רב בתוך ההדרים. ניתן להשתמש ב-PEView כדי לסקור את המידע בתוך ההדרים.

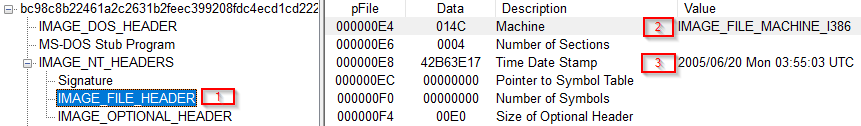


Figure II: פתיחת קובץ עם PEView

בתמונה הנ"ל מספר **1** מייצג את ה-Header הנבחר.

**2** – מציג עם איזה סוגי מעבדים הקובץ רץ: i386

**3** – מציג את תאריך הקימפול: **20 ליוני 2005** בשעה **03:55 UTC**.

* תאריך הקימפול חשוב בשביל לדעת אם הקובץ ישן או חדש. אם התאריך ישן כנראה שכבר גורמים אחרים עשו עליו ניתוח, ואפשר יהיה להצליב נתונים עם חוקרים אחרים.

אזור ה-IMAGE\_OPTIONAL\_HEADER:

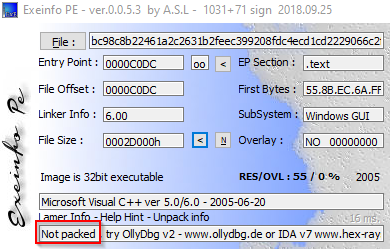
* Subsystem חושף אם הקובץ פועל עם **ממשק משתמש** (*GUI*) דרך מערכת ההפעלה או דרך **שורת פקודות** (*Command Line Interface*).



הערך IMAGE\_SUBSYSTEM\_WINDOWS\_GUI מצביע על כך שלקובץ יש ממשק משתמש (ראוי לציין, שה-GUI לא תמיד נראה למשתמש, גם אם הוא קיים)

* + (אם הערך היה IMAGE\_SUBSYSTEM\_WINDOWS\_CUI משמע שהתוכנה משתמשת ב-CLI.)

פתיחת הקובץ עם exeinfope, מגלה שהקובץ **לא ארוז**:



פתיחה עם pestudio: זיהוי **ארכיטקטורה**:



### סיכום

* **סוג הקובץ**: Portable Executable (PE)
* ארכיטקטורה: 32-ביט.
* **ארוז**: שלילי.
* **Compiler-Stamp**: 0x42B63E17 (Mon Jun 20 06:55:03 2005)
* **מכונה**: i386.

## חתימות מאלוור (Malware Hashing)

### יצירת חתימות רלוונטיות עם HashMyFiles:

|  |  |
| --- | --- |
| **MD5** | b06d9dd17c69ed2ae75d9e40b2631b42 |
| **SHA1** | b606aaa402bfe4a15ef80165e964d384f25564e4 |
| **SHA256** | bc98c8b22461a2c2631b2feec399208fdc4ecd1cd2229066c2f385caa958daa3 |

### PEStudio Hashes:

#### ImpHash



### Sections MD5 Hashes



hash\_md5 = "b06d9dd17c69ed2ae75d9e40b2631b42"

hash\_sha1 = "b606aaa402bfe4a15ef80165e964d384f25564e4"

hash\_sha256 = "bc98c8b22461a2c2631b2feec399208fdc4ecd1cd2229066c2f385caa958daa3"

imp\_hash = "0FCEA3AF550AD0A893E93808DCCF17F4"

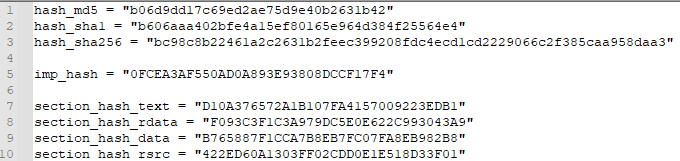
section\_hash\_text = "D10A376572A1B107FA4157009223EDB1"

section\_hash\_rdata = "F093C3F1C3A979DC5E0E622C993043A9"

section\_hash\_data = "B765887F1CCA7B8EB7FC07FA8EB982B8"

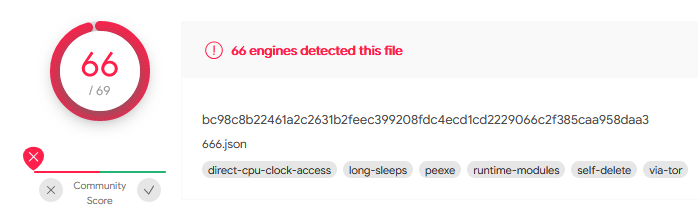
section\_hash\_rsrc = "422ED60A1303FF02CDD0E1E518D33F01"

#### שמירה לקובץ בפורמט הנכון לשימוש YARA מאוחר יותר:



## ניתוח סטטי [ב-VirusTotal](https://www.virustotal.com/gui/file/bc98c8b22461a2c2631b2feec399208fdc4ecd1cd2229066c2f385caa958daa3/detection):

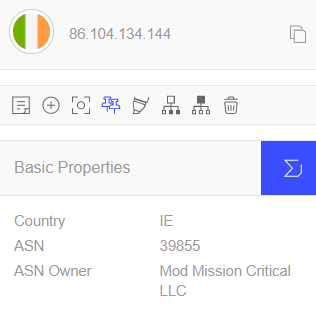
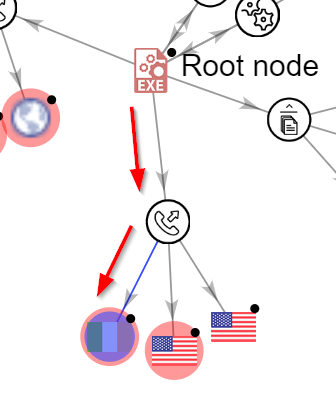
* **66 מתוך 69** מנועי אנטי וירוס זיהו את הקובץ **כמסוכן**.
* **סוג קובץ**: Win32 EXE
* **Magic**: PE32 executable for MS Windows 32-bit
* **מטרה**: מעבדי Intel 386 ומעלה.



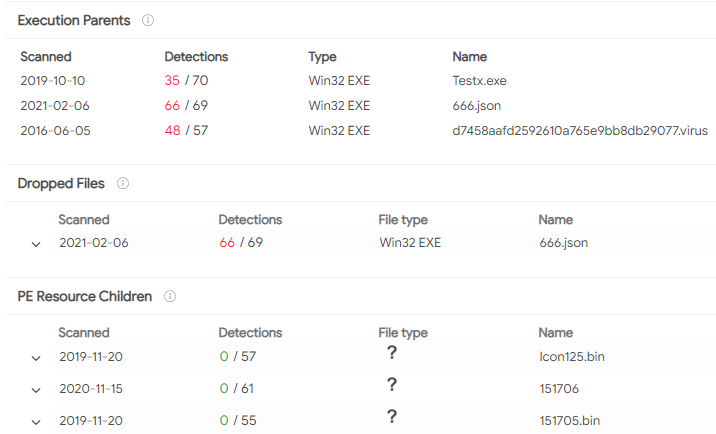
### Relations Tab

מידע על האינטראקציות שיש לקובץ עם ארטיפאקטים אחרים כמו כתובות IP (*כנראה C&C*)

* תקשורת עם כתובת IP מאירלנד (*מיוצגת ע"י החצים האדומים*):



* מידע על התהליכים והקבצים שהקובץ משתמש בהם:



## ניתוח סטטי ב-[Any.Run](https://app.any.run/tasks/25e35127-c5a4-4b09-b2a0-618054815ec3/) ע"י חיפוש ה-MD5

חיפוש האש MD5 של הקובץ מקנה לי את המסקנות הבאות:

1. ההאש מזוהה כ-**Malware** מסוג **Ransomware** בשם **Locky**.



Figure III: תוצאות חיפוש ההאש בAny.Run



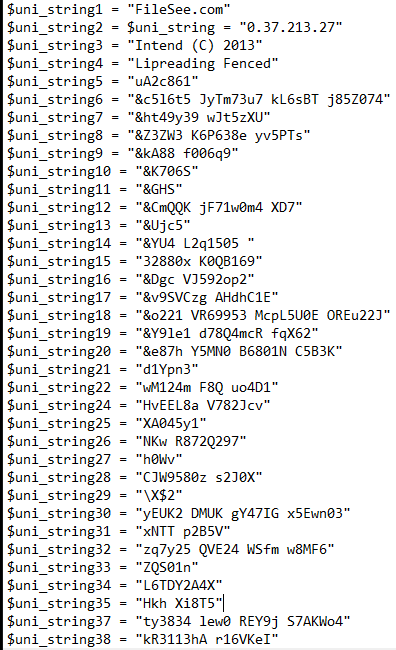
Figure IV: ההאשים שהוגשו עם הניתוח

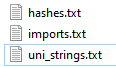
שלושת ההאשים השונים תואמים את ההאשים שג'ונרטו מהקובץ הנחקר.

1. יוצר תקשורת החוצה בבקשת POST עם IP **86.104.134.144** בפורט **80**. ספציפית לכתובת: <http://86.104.134.144/main.php>, שמקורה באירלנד. (*קוד ארץ:* ***IE***)

## חליצת סטרינגס (STRINGS)

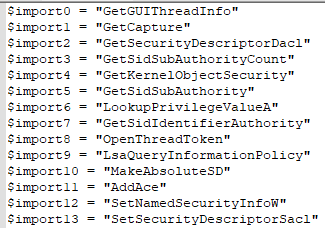
floss Locky > flossLocky.txt 

כשפותחים את קובץ הטקסט אפשר לראות את כל מחרוזות ה-ASCII וגם ה-UTF-18. רוב המחרוזות הם רק הוראות שפורשו כמחרוזות. כל המחרוזות המיוחדות ישמשו כסט חוקים ב-YARA. במצב הנוכחי, נראה שיש המון מחרוזות לכן בחרתי רק מחרוזות ארוכות ומיוחדות בתור מחרוזות שישמשו ל-**YARA**. דבר זה נועד כדי למנוע עומס בתהליך איתור המחרוזות בחוק **YARA**.

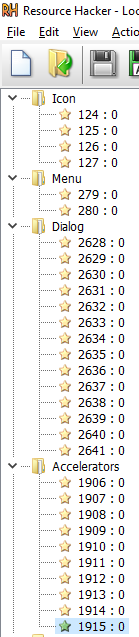


## PE Header

מציאת הפונקציות שמיובאות ע"י הקובץ. פתיחת הקובץ **ב-PESTUDIO** תיתן מידע אודות הפונקציות השונות וגם אינדיקציה לאם **PESTUDIO** מזהה את הפונקציה כפונקציה שמאלוורים אחרים משתמשים בה. פונקציות כאלה מסומנות ב-**X** תחת לשונית **BLACKLIST**. ניתן להשתמש בשמות של פונקציות אלה לחוקי YARA לצורך זיהוי המאלוור.



## Resources

עם **ResourceHacker** אפשר לבחון את רכיבי ה-GUI של המאלוור. רכיבי GUI הם רכיבים שלא נראים בזמן הרצה דינאמית של המאלוור אלה קיימים בתוך הקובץ. רכיבים אלה יכולים לסייע בהבנה עמוקה יותר של פעילות המאלוור. כפי שניתן לראות מהתמונה, לקובץ יש **4 אייקונים**, **2 אובייקטים שונים לתפריט**, **14 פריטי דיאלוגים** שמוצגים ע"י ביצוע פעולות שונות.

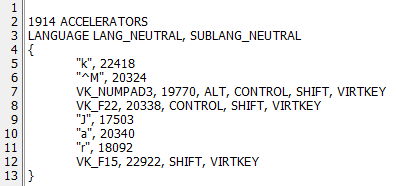
ראוי לציין שלמאלוור הזה יש **10 אובייקטים מאיצים (Accelerators)**. אלה קיצורי-מקשים לפעולות המקושרות לקובץ הבינארי ורומז לנו שאולי המאלוור עוקב אחרי הקלדות של משתמש.

Figure V: דוגמא ל-Accelerator (מיוצג כ-1914)

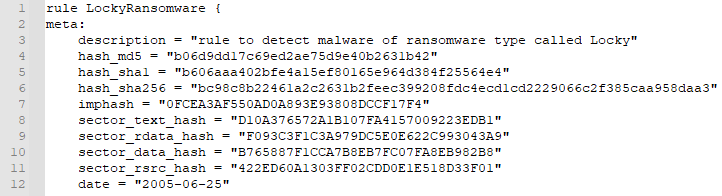
## דרכים לחסימת המאלוור

### Yara rules

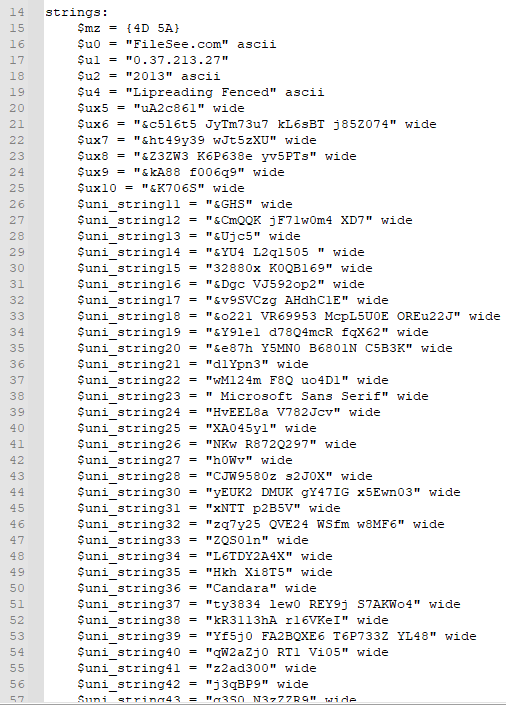
אחרי הניתוח הסטטי ואחרי שהוצאתי את כל המידע הרלוונטי, אשתמש בו כדי ליצור חתימות YARA עם סט חוקים. לקובץ חוקי YARA שלושה אזורים, **Meta**, **Strings** וה-**Condition**. כל אזור אחראי למשהו אחר כפי שמפורט:

#### Meta Section

אזור זה הוא לשימוש קורא הקוד. הוא **מכיל מידע על החוק** כמו: שם החוק, כותב החוק, האשים רלוונטיים, תאריך... לא משומש בבדיקת החוק עצמו או איתור התנאים.

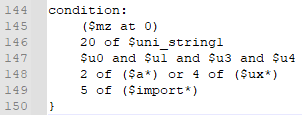


#### Strings

🡨 אזור זה מכיל את כל המחרוזות שהחוק ישתמש בהן לזיהוי המאלוור בעת התהליך. התנאים השונים נכתבים בצורה שיכללו את כל המחרוזות בקובץ, כל תנאי על פי אופיו:

#### Condition

אזור שבו נכתבים התנאים השונים לבדיקה בהצלבת המאלוור עם החוק:



($mz at 0) // Check for MZ ‘\x4D\x5A’ Bytes at offset 0.

20 of $uni\_string1 // Checks if 20 of the uni\_string strings are present.

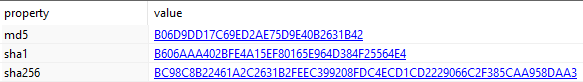
$u0 and $u1 and $u3 and $u4 // Checks if all $u are present.

2 of ($a\*) or 4 of ($ux\*) // Checks if 2 of the text strings are present. Checks if 4 of $ux strings are present.

5 of ($import\*) // Checks if 5 of the blacklisted $import strings are present.

### זיהוי על פי Indicators of Compromise

#### חתימות ההאש של הקובץ:



הדרך הפחות מאובטחת. מספיק שישתנה \ יוסף תו אחד (*Garbage Chars*) וכל ההאש של הקובץ יהיה אחר לגמרי **אך הפונקציונליות של הנוזקה תישאר שלמה**, מה שהופך את ההאש הנ"ל ללא יעיל. כל ה $a לדוגמה שנוספו ע"י מפיץ הנוזקה "שיבשו" את חתימת ההאש.

#### בקשות DNS שזוהו:

|  |  |
| --- | --- |
| **xwmaghtuu.pm** | Domain |
| **dyoqumjpbrk.pw** | Domain |
| **pjoaycpw.fr** | Domain |
| **ieehxmtcpwgdq.us** | Domain |
| **vtbqg.uk** | Domain |
| **qoldrpqbisbyly.in** | Domain |

#### חיבורים

כתובת IP חשודה: ***86.104.134.144***

#### בקשות HTTP

<http://86.104.134.144/main.php> בבקשת ***POST***.

שילוב של כל האינדיקטורים במערכת הגנה ארגונית תהיה הגישה היעלה ביותר

# פרק 2: ניתוח דינאמי

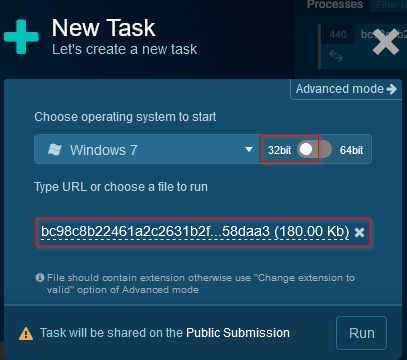
## שימוש ב-Malware Sandbox

### Any.Run

[Any.Run](https://app.any.run/) הוא שירות מצוין להעלאת קבצים חשודים ובחינתם בסביבה מאובטחת מבוססת ענן.

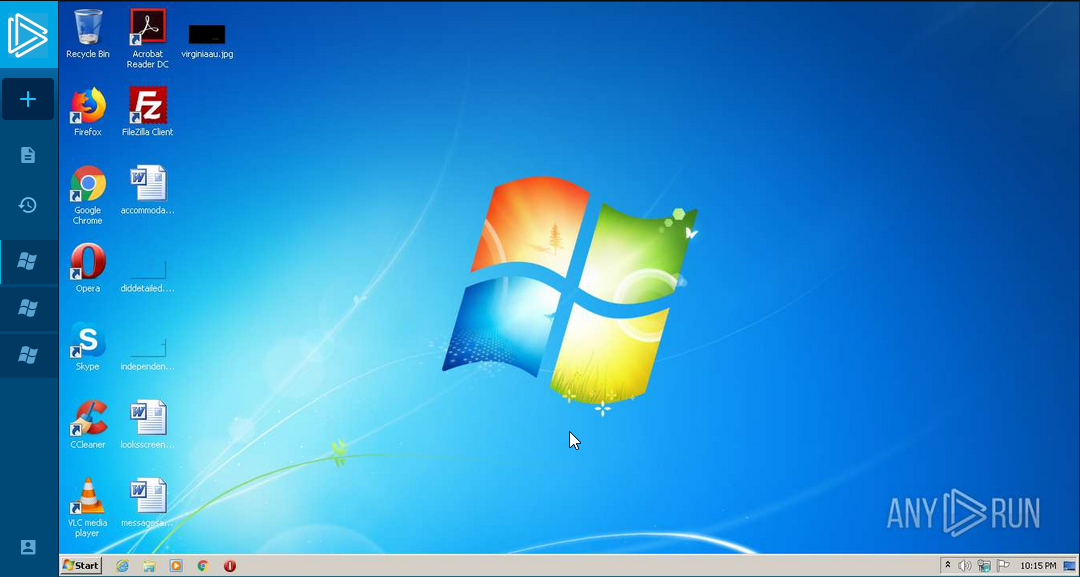
מתוך הניתוח הסטטי הבנתי שהקובץ הוא **Portable Executable** למערכות הפעלה מסוג **ווינדוס 32 ביט** בעלי מעבדים מסדרת **Intel i386** ומעלה.

1. בניית מערכת ההפעלה המתאימה וטעינת הקובץ החשוד:

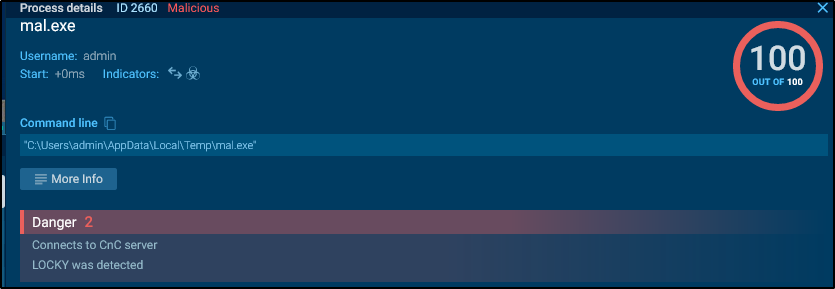


המערכת שתעלה תהיה Windows 7 32-bit. מיד אחרי שהמכונה עולה הקובץ יפעל ויוצג ניתוח על כל **השינויים, בקשות HTTP, DNS, חיבורים ותהליכים.**

1. אחרי לחיצה על Run, מערכת ההפעלה תעלה בסביבה מאובטחת ותריץ את הקובץ החשוד. (אם אין סיומת לקובץ Any.Run, תנתח את הקובץ כדי שיוכל לפעול כראוי)



**זיהוי הקובץ כמזיק**: מנסה להתחבר ל-C&C Server ומזוהה כ-**LOCKY**. כמו כן, **100** מתוך **100** מנועי אנטי וירוס מזהים את הקובץ **כמזיק**:



#### HTTP Requests

1. נראה שבוצעה בקשה אחת לכתובת 86.104.134.144 לדף בשם main.php בתצורת **POST**. הבקשה **לא צלחה** – **אין תגובה מהשרת**.

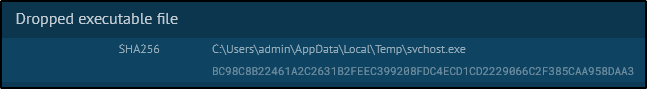
#### DNS Requests

1. ניסיון ל-**6** בקשות DNS שונות:
2. xwmaghtuu.pm
3. dyoqumjpbrk.pw
4. pjoaycpw.fr
5. ieehxmtcpwgdq.up
6. vtbqg.uk
7. qoldrpqbisbyly.in

לששת הבקשות **לא נמצאה** כתובת IP על כן לא התבצע חיבור והמאלוור נתקע בהרצה.

#### הנחתת קבצים ניתנים להרצה (Dropped Exe Files)

המאלוור יצר קובץ EXE בתיקיית %TEMP% בשם svchost.exe. לקובץ אותו האש SHA256 כמו למאלוור. כלומר, הקובץ הוא העתק מדויק של קובץ המאלוור עצמו המתחזה לתהליך svchost.exe על כן, זוהתה פעילות חשודה וזה נחשב כ-IoC:



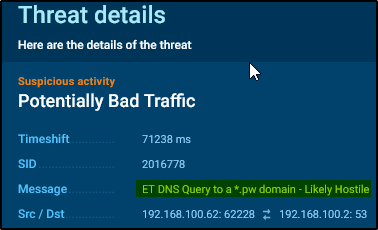
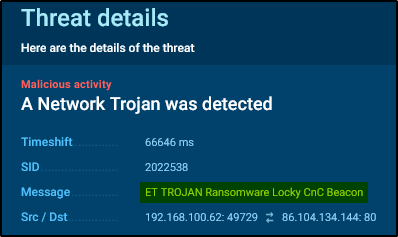
#### Indicators of Compromise

|  |  |  |
| --- | --- | --- |
| **IOC** | **Type** | **Title** |
| * **Main Object – “mal.exe”:** | | |
| b06d9dd17c69ed2ae75d9e40b2631b42 | MD5 |  |
| * **DNS Requests:** | | |
| xwmaghtuu.pm | Domain |  |
| dyoqumjpbrk.pw | Domain |  |
| pjoaycpw.fr | Domain |  |
| ieehxmtcpwgdq.us | Domain |  |
| vtbqg.uk | Domain |  |
| qoldrpqbisbyly.in | Domain |  |
| * **Connections:** | | |
| 86.104.134.144 | IP |  |
| * **HTTP Requests:** | | |
| http://86.104.134.144/main.php | URL |  |

אינדיקטורים אלה ישומשו במערכת הגנה ארגונית להגנה מפני המאלוור באם אחד או יותר מהאינדיקטורים הנ"ל נקלטו ע"י המערכת (*כגון EDR*)

#### איומים מרכזיים

* פעילות חשודה ומאיימת ע"י התחברות או ניסיון להתחברות לשרת Command & Control.
* בקשת DNS לדומיין עם פוטנציאל מזיק בתבנית \*.pw



#### מסקנות ניתוח דינאמי עם Any.Run

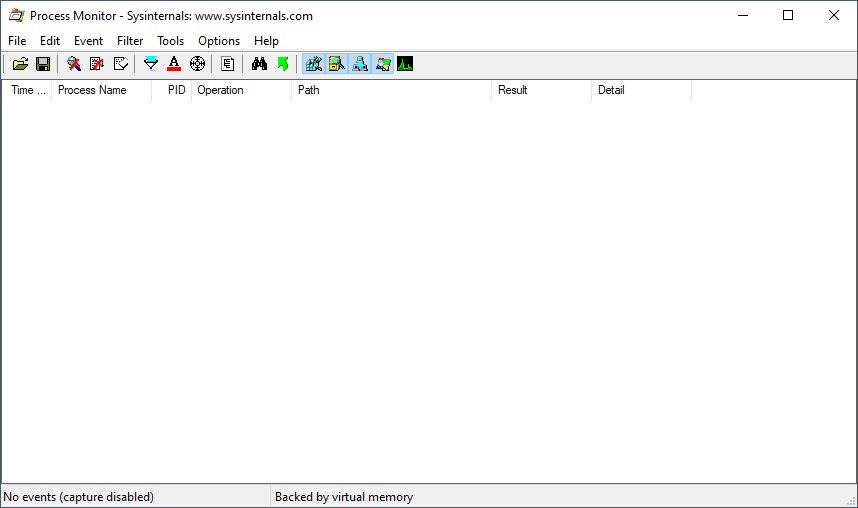
1. מאלוור שמנסה לתקשר עם **שרת באירלנד** המזוהה כ**מזיק** בשרתי Any.Run (כנראה לצורך הורדת קבצים מזיקים אחרים המכילים מידע / הוראות / פקודות נוספות להפעלת המאלוור)
2. המאלוור שולח בקשות DNS הנ"ל בצורה אינסופית אך אין תגובה מבקשות אלו, ואין תגובה משרת ה-C&C, לכן המאלוור לא מממש את מטרותיו.

## הרצה על מכונה וירטואלית

### ניטור עם Process Monitor

Process Monitor או procmon, חלק מ-Windows Sysinternals הוא כלי ניטור מתקדם למערכות ווינדוס המקנה את היכולת לעקוב אחר פעילות רג'יסטרי, קבצי מערכת, תקשורת, ותהליכים.

#### ניקוי התצוגה לפני הרצת המאלוור:



#### הפעלת פילטרים

Procmon מציג מידע רב. לפעמים מידע רב מדי שיהיה קשה מדי לנסות ולנתח את כולו. לכן ל-procmon יש אפשרות להפעלת פילטרים כדי להציג רק מידע רלוונטי ושחשוב לנו.

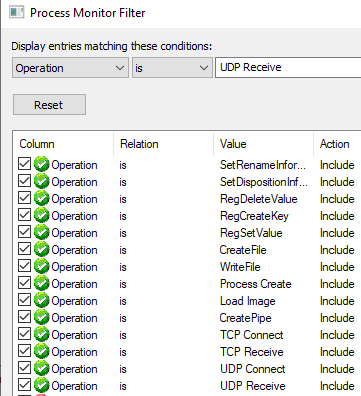


Figure VIII: Operations included for procmon display

**הפילטרים שנבחרו לתצוגה:**

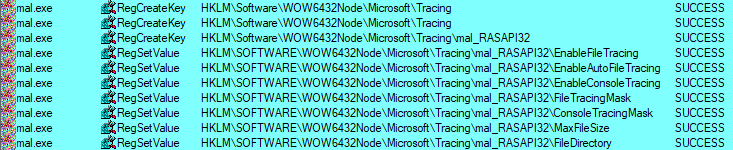
**הפעולות** שבוצעו: אלה פעולות הכי נפוצות לניתוח מאלוורים ובהם אתמקד:

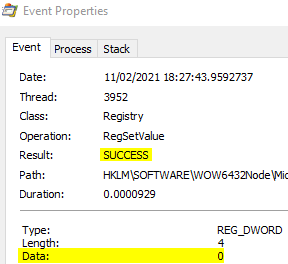
* + SetRenameInformationFile: Rename operation occurred.
  + SetDispositionInformationFile: Deletion of file occurred.
  + RegDeleteKey: Key gets deleted from the registry.
  + RegDeleteValue: The value of a key is deleted from the registry.
  + RegCreateKey: Registry key is created.
  + RegSetValue: Data in the registry is set in the registry.
  + CreateFile: Process wants to create a file.
  + WriteFile: Process writes data to file.
  + Process Create: Process creates another process.
  + Load Image: Process loads any DLL’s \ Executables.
  + CreatePipe: Process creates a Pipe.
  + TCP / UDP Connect / Receive: Process is sending \ receiving TCP connection.

##### שימוש ברג'יסטרי

ברגע הרצה נוצרים מפתחות רג'יסטרי בשם של הקובץ + \_RASPI32. מפתחות אלה נוצרים כאשר אפליקציה עובדת עם ה-API של **גישה מרחוק** (*Remote Access*), והספרייה raspi32.dll. מפני שהקובץ עובד עם הספרייה הזו, ניתן להסיק שיש ניסיון לתקשורת.

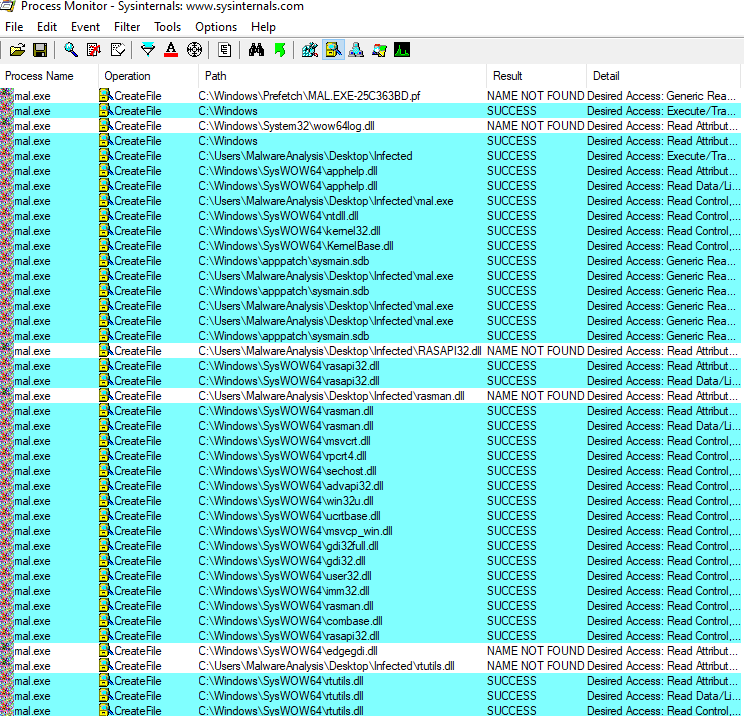
לאחר שנוצרו המפתחות, הערכים ב-EnableFileTracing ו-EnableConsoleTracing מוגדרים ל-**0**. כתוצאה מכך, לא ייווצרו לוגים בתיקייה המוגדרת FileDirectory לפעולות שהמאלוור מבצע.



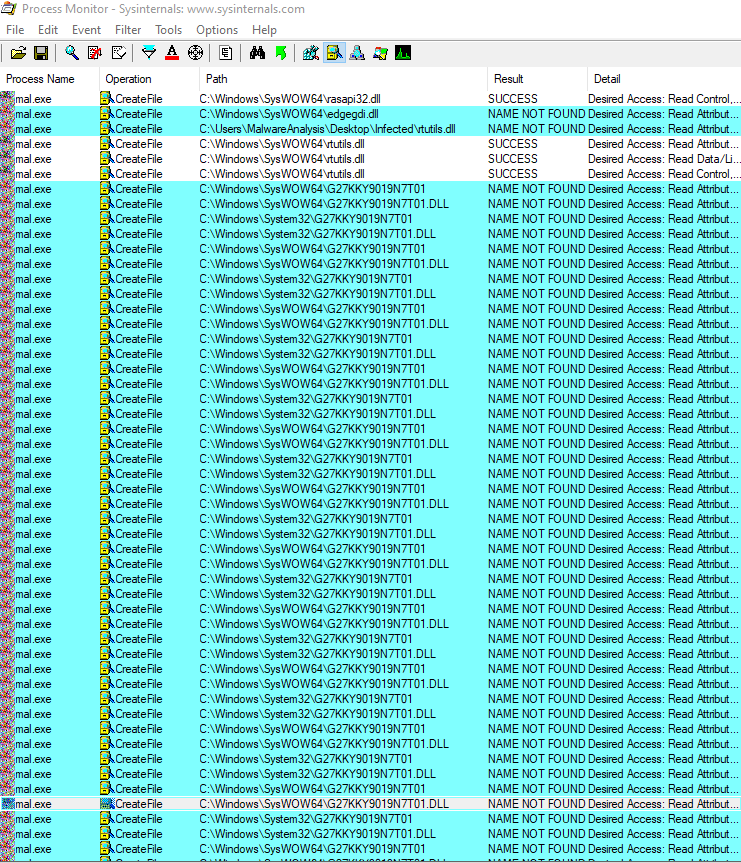


##### קבצים

המאלוור מנסה לגשת לקבצים ולפי procmon נראה שהוא רק מנסה לקרוא. מתבטא בטור ה-Detail: תחת Desired Access: Read Attributes\Generic Read.



כמו כן, יש המון קבצים שהוא מנסה לגשת אליהם אך שמם לא נמצא לפי procmon.



##### תהליכים

המאלוור יצר שני תהליכים תחתיו: svchost.exe ו-cmd.exe.

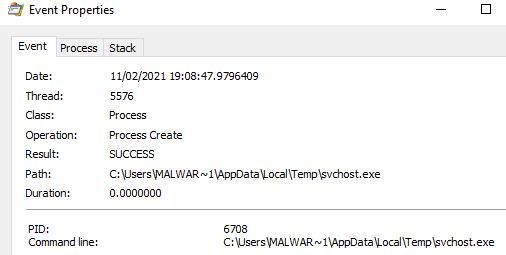


Figure IX: svchost.exe Event Properties

בתהליך השני מתבצעת פקודה דרך ה-**CMD** למחיקת קובץ בשם sys270E.tm***p***, שנמצא בתיקיית **Temp** של המשתמש הנוכחי.

Command line:

cmd.exe /C del /Q /F "C:\Users\MALWAR~1\AppData\Local\Temp\sys270E.tmp"

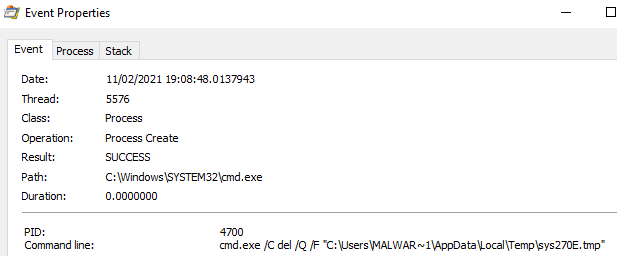


Figure X: cmd.exe Event Properties

דרך ה-CMD שנוצר מקובץ המאלוור, נוצר תהליך משני בשם Conhost.exe.

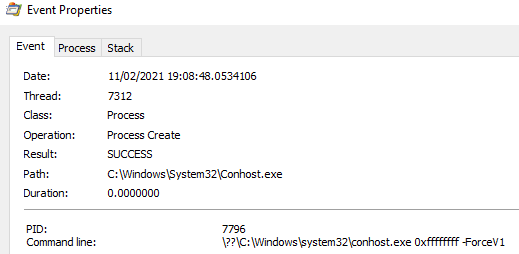
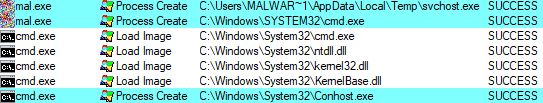


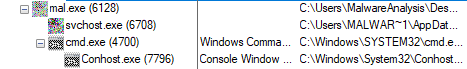
Figure XI: cmd.exe Event Properties

Command Line:

\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1



תצורת עץ של התהליכים שנוצרו מתוך procmon:



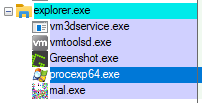
##### תעבורת רשת

לא נתפסה תעבורה ברשת דרך procmon בפרוטוקולים **TCP** או **UDP**. ראוי לציין מהניתוח ב-Any.Run, כי המאלוור שולח בקשת **HTTP** בתצורת **POST** לכתובת IP מ**אירלנד**. לכן כנראה הבקשה הזו לא נתפסה ב-procmon.

### ניתוח עם Process Explorer

Process Explorer מביא את האפשרות לנתח תהליכים שרצים על המערכת, לראות ספריות DLL שהוטענו מתהליך מסוים, תקשורת, ולנתח סטרינגים כפי שהם נטענו לזיכרון בעת הרצה.

1. הרצת המאלוור:



1. המאלוור רץ, הורג את עצמו (*מסומן באדום*) ופותח תהליך חדש בשם svchost.exe בעצמו (*מסומן בירוק*):

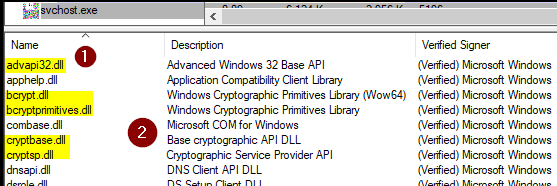


1. בסופו של דבר רק svchost.exe נשאר פועל ברקע:

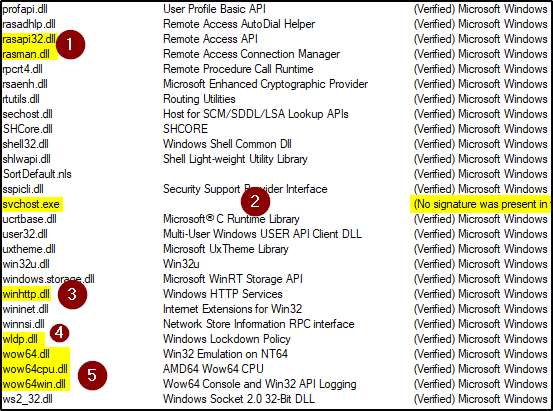


#### חקירת svchost.exe

##### DLLים מעניינים בשימוש:

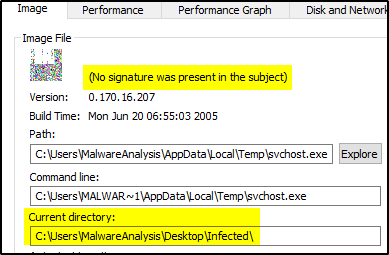


1. Advapi32 – שימוש ברג'יסטרים, פונקציות כגון: AddAce, EncryptFileW...
2. ספריות crypto של ווינדוס – מצאתי את ה-DLL הזה מעניין משום שהוא רומז שיש שימוש במפתחות קריפטוגרפים מה שמחזק את ההשערה שהמאלוור מריץ Ransomware ע"י הצפנה של קבצים ותיקיות עם הפונקציה EncryptFileW.



1. Rasapi32 ו-rasman – עוזרים לשימוש בשליטה מרחוק (Remote Access API & Manager)
2. Svchost.exe – התחזות ל-svchost, פותח תהליכים תחת תהליך אחד כדי לשמור על צריכת משאבים מהמחשב.
3. Winhttp – מנגישה את המערכת לממשק ברמה גבוהה עם תמיכה בשרתים לפרוטוקולים **HTTP2** ו-**1.1**. משומש בעיקר בסביבה של שרת ע"י אפליקציות שמתקשרות עם שרתי HTTP. רומז על כך שיש תקשורת או לפחות ניסיון לתקשורת בפרוטוקול **HTTP.**
4. Wldp – **Windows Lockdown Policy**. קורא לספריה בשביל לקבל את מצב האבטחה ביחס למערכת והסקריפט (תהליך ה-svchost בהקשר זה) **או** מנהל ההתקנות של ווינדוס (MSI) הרץ.
5. Wow64 – מנגישה אפליקציות של 32-ביט למערכות 64-ביט.

###### svchost: מאפיינים

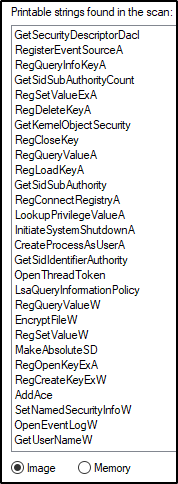
**אפשרות שימושית מאוד כשמנתחים מאלוור היא אפשרות ה- Verify בלשונית ה-Image במאפיינים של תהליך מסוים**. לחיצה על Verify תוודא את האמינות של הקובץ / תהליך באם הוא **באמת** קובץ בינארי חתום ממייקרוסופט. מייקרוסופט מטמיעה חתימות דיגיטאליות לרוב הקבצים הניתנים להרצה מטעמה. כש-Process Explorer מוודאת את החתימה הדיגיטאלית אפשר להיות בטוחים שהתהליך הוא באמת תוצר מבית מייקרוסופט ולא תהליך מתחזה בשם דומה.

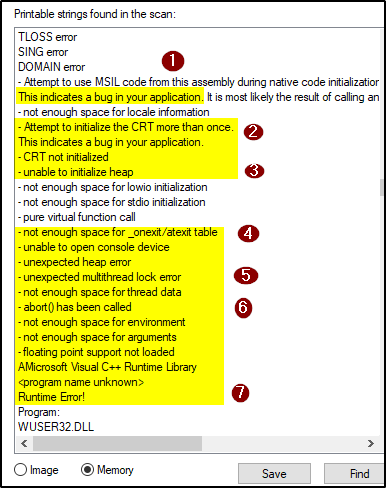
ל-svchost.exe שנפתח דרך התהליך של המאלוור אין חתימה דיגיטאלית מה שאומר שהוא לא תהליך ממייקרוסופט ויכול להיות מזיק.

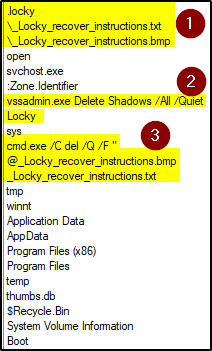
###### השוואת סטרינגים

דרך לאמת החלפת תהליכים במאלוורים היא להשוות בין המחרוזות ב-**Image** לבין המחרוזות שבאמת נטענו **לזיכרון**. Process Explorer נותן לנו את האפשרות לבחון את זה בלשונית Strings בתוך המאפיינים של התהליך.

ב-svchost.exe המחרוזות שנטענו לזיכרון המערכת שונות מהמחרוזות ב-Image.



חלק מהמחרוזות שנטענו לזיכרון הם הודעות שגיאה על פעולות שהמאלוור מנסה לעשות.

1. ניסיון שימוש ב-Microsoft Intermediate Language לא צלח. יש באג בתוכנה. “This indicates a bug in your program”
2. אתחול **CRT** לא מוצלח.
3. הודעת שגיאה על HEAP שלא אותחל.
4. הודעות שגיאה על חוסר מיקום.
5. שגיאה לא צפויה ב-HEAP + שגיאה על נעילת טרדים.
6. פונקציה בשם abort נקראה. (ביטול המאלוור (?))
7. **Runtime Error!** שגיאה בזמן הרצה.
8. מחרוזות Locky מופיעות לכל אורך העמוד. Locky הוא שם של ה-Ransomware.
9. Vssadmin.exe Delete Shadows /All /Quiet: מוחק Shadow Copies מהמערכת.

/All – כל הערכים (All Volumes)

/Quiet – אל תוציא פלט, תעבוד ברקע.

1. cmd.exe /C del /Q /F – פקודת PowerShell להרצת פקודות עם CMD.

/C – **מריץ** את הפקודה הבאה ויוצא מ-CMD.

del – **הפקודה שתתבצע**. בהינתן נתיב, מוחק קובץ אחד או רשימה של קבצים.

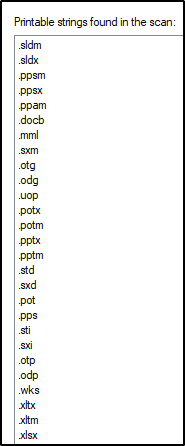
/Q – **מצב שקט**. לא שואל "כן/לא".

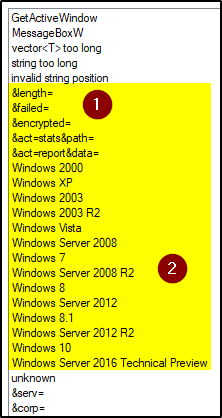
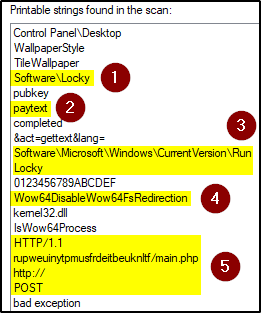
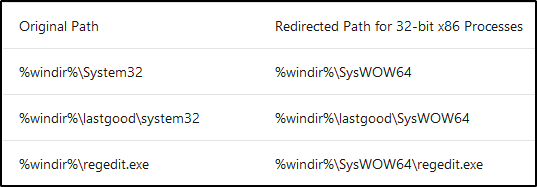
/F – **Force**. אם קובץ מוגדר "לקריאה בלבד", מתעלם מההגדרה ומוחק בכל זאת.

1. Locky\_recover\_instructions: שם הקובץ בו כתובות **ההוראות לתשלום הכופר**. הוראות אלה יוצגו למשתמש כשהמאלוור יפעל.

מספר 2 אומר שהמאלוור מוחק קבצי גיבוי מהמערכת כדי שלא יהיה ניתן להחזיר את המערכת לקדמותה טרם הרצת המאלוור.

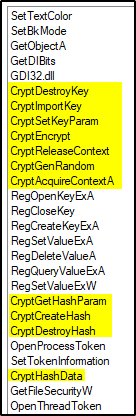
מספר 3 מצביע על ניסיון למחיקת קבצים עם הפקודה בהינתן נתיב רלוונטי.

סיומות קבצים כגון אלה. המאלוור מתעסק עם עשרות, אם לא מאות סיומות קבצים.

1. **קישורים (Reference) קבועים** למשתנים שונים, מסומנים עם הכרזת ‘&’ בתחילת המשתנה. אלה ישומשו את התוכנה בשביל להגיע למשתנים בזיכרון אם נקראו. יש משתנה אחד מעניין בשם **encrypted** – כנראה משומש בתהליך הצפנת הקבצים ו/או בדיקה אם קובץ הוא מוצפן או לא.
2. **מערכות הפעלה**: החל מוינדוס 2000 עד ווינדוס 10, ווינדוס סרבר 2008 עד 2016.
3. ******נתיב לרג'יסטרי בשם Locky תחת Software.
4. אולי קשור להוראות תשלום של הכופר המוצגות לקורבן.
5. נתיב רג'יסטרי בשימוש המאלוור – המאלוור כותב לנתיב זה מפתחות חדשים. כל מה שנמצא בתיקיית Run יפעל אוטומטית כל פעם שהמערכת עולה.
6. ביטול Redirection לקבצי מערכת. במערכות 64 ביט, נתיב %windir%\System32 שמור לאפליקציות 64 ביט. רוב שמות קבצי DLL לא השתנו כשקבצי DLL ל-64 ביט נוצרו, אז קבצי DLL של 32 ביט אוחסנו בתיקייה נפרדת. WOW64 מחביאה את השינוי עם *File System Redirector*, ***ניתוב מחדש לקבצי מערכת***. כשאפליקציה מנסה לגשת אל תיקיות System32 או regedit.exe, קורה ניתוב מחדש לנתיב הספציפי המתאים לארכיטקטורה הרצה. אפליקציות לא אמורות לגשת לתיקיות אלה, אלא להינתב ל-API כמפורט:

Wow64DisableWow64FsRedirection מבטל את הניתוב מחדש מה שמקנה לאפליקציה גישה ל-**System32** לכל הפעולות שאותו תהליך מנסה לעשות. כך לדוגמה, אפליקציית 32-ביט שרצה תחת WOW64 יכולה לפתוח אפליקציות 64-ביט מתוך SystemRoot%\System32% במקום להינתב מחדש לגרסת ה-32-ביט בתוך SystemRoot%\SysWOW64%.

1. בקשת POST בפרוטוקול HTTP/1.1 – ניסיון לתקשורת עם כתובת: rupweuinytpmusfrdeitbeuknltf/main.php

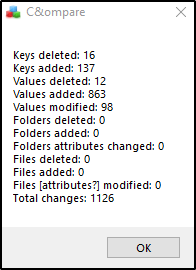
**פונקציות** Crypt **מתוך** Wincrypt.h: **CryptEncrypt** מצפינה מידע ע"פ הפרמטרים שניתנו לה מהפונקציות האחרות המתחילות ב-***Crypt***. לדוגמה CryptGenRandom ממלאת באפר בבייטים קריפטוגרפים רנדומליים. CryptCreateHash יוצרת ומחזירה **Handle** לאובייקט האש של ספק שירות קריפטוגרפיה או בלעז, Cryptogaphic Service Provider (CSP). ה-Handle הזה משומש בקריאות פונקציות אחרות במשפחת Crypt.

### השוואת שינויים ברג'יסטרי עם RegShot

#### דרכי פעולה

* + צילום (*Snapshot*) של הרג'יסטרי עם RegShot טרם הפעלת המאלוור.
  + הרצת המאלוור, המתנה לסיום.
  + צילום של הרג'יסטרי אחרי שהמאלוור פעל.
  + השוואה בין שני הצילומים וחקירת שינויים שנעשו.

#### תוצאות

סה"כ נעשו **1126** שינויים אחרי הרצת המאלוור. רוב השינויים לא קשורים להרצת המאלוור וידוע שיהיו רעשי רקע.

#### מפתחות שנוספו

HKLM\SOFTWARE\WOW6432Node\Microsoft\Tracing\**mal\_RASAPI32**

HKLM\SOFTWARE\WOW6432Node\Microsoft\Tracing\**svchost\_RASAPI32**

HKU\S-1-5-21-2427678116-2504986317-361849973-1001\SOFTWARE\**Locky**

#### שינויים במפתחות

כולם נמצאים באותו נתיב המתחיל:

HKLM\SOFTWARE\WOW6432Node\Microsoft\Tracing

|  |  |
| --- | --- |
| **Key** | **Value** |
| \mal\_RASAPI32\**EnableFileTracing** | **0x00000000** |
| \mal\_RASAPI32\**EnableAutoFileTracing** | **0x00000000** |
| \mal\_RASAPI32\**EnableConsoleTracing** | **0x00000000** |
| \mal\_RASAPI32\FileTracingMask | 0xFFFF0000 |
| \mal\_RASAPI32\ConsoleTracingMask | 0xFFFF0000 |
| \mal\_RASAPI32\MaxFileSize | 0x00100000 |
| \mal\_RASAPI32\FileDirectory | "%windir%\tracing" |
| \svchost\_RASAPI32\**EnableFileTracing** | **0x00000000** |
| \svchost\_RASAPI32\**EnableAutoFileTracing** | **0x00000000** |
| \svchost\_RASAPI32\EnableConsoleTracing | 0x00000000 |
| \svchost\_RASAPI32\FileTracingMask | 0xFFFF0000 |
| \svchost\_RASAPI32\ConsoleTracingMask | 0xFFFF0000 |
| \svchost\_RASAPI32\MaxFileSize | 0x00100000 |
| \svchost\_RASAPI32\FileDirectory | "%windir%\tracing" |

הגרדת הערכים הנ"ל ברג'יסטרי **מבטלים** **יצירת לוגים** על פעולות שהמאלוור מבצע כנראה לצורך לחשאיות.

### זיוף רשת

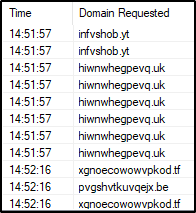
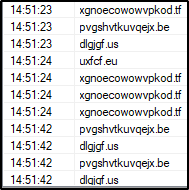
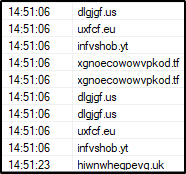
עם ApateDNS, אפשר לראות בקשות **DNS** שנעשו על ידי תוכנות זדוניות בדרך המהירה ביותר. ApateDNS מתחזה לתגובות *DNS* לכתובת *IP* שצוינה על-ידי המשתמש על-ידי האזנה ל**פורט 53** במחשב המקומי.

#### דרכי פעולה

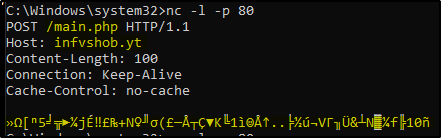
1. ניתוק המכונה מהרשת. (*Host-only adapter*)
2. הרמת שרת WEB במכונה אחרת. אשתמש במכונה שלי של Kali Linux.
3. הגדרת כתובת IP שתישלח בתגובות DNS: ***83.130.134.238*** – הכתובת של Kali Linux.
4. התחלת השרת.
5. הקשבה לפורט **53** עם ApateDNS.
6. בדיקת אילו בקשות נעשו.

#### ממצאים

אחרי הרצת המאלוור, בזמן ההאזנה, ובזמן ששום תהליך אחר לא פועל והרשת מנותקת. ApateDNS מציג רשימה של בקשות DNS שנעשו:



* המאלוור חוזר על עצמו ומגיש בקשות DNS בשביל כתובות אלה שוב ושוב.

בשילוב עם Netcat והאזנה לפורט **80**, ניתן לראות **בקשות** **HTTP** שהמאלוור מבצע:

המאלוור מבצע בקשת פוסט ל: infvshob.yr/main.php. המידע ב-POST Data לא קריא.

### תפיסת קבצים שנמחקים אוטומטית

**CaptureBAT** כבר לא זמין ולא מצאתי הורדה קיימת, לכן אשתמש בכלי חלופי שמצאתי ב-GitHub הנקרא [CapturePy](https://github.com/fbruzzaniti/Capture-Py). CapturePy הוא כלי ניתוח שעושה עותק של כל הקבצים שנמחקו או שונו בתיקייה נתונה ותתי-תיקיות שלה אל תיקייה אחרת.

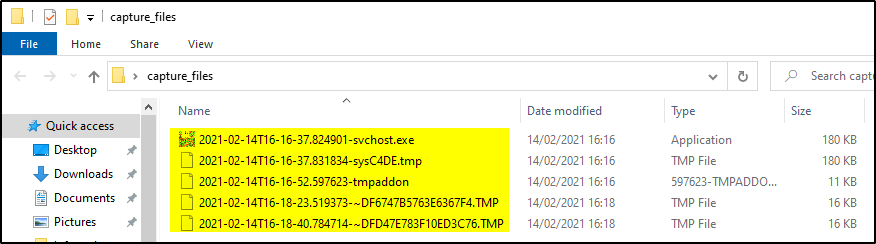
שימוש: python capture-py.py <capture-directory> <save-directory>

המאלוור יוצר ומוחק קבצים מתיקיית %TEMP% של המערכת. לכן הגדרתי לכלי לעקוב אחר התיקייה ולשמור קבצים שנמחקו בתיקיית capture\_files על שולחן העבודה.

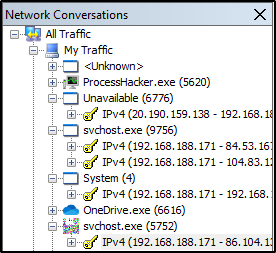
C:\Users\MalwareAnalysis\Desktop\Capture-Py-master> python capture-py.py C:\Users\MALWAR~1\AppData\Local\Temp C:\Users\MalwareAnalysis\Desktop\capture\_files

תוצאות:

**5** קבצים נמחקו והועברו לתיקייה **Capture\_files**.



### Microsoft Network Monitor

Microsoft Network Monitor מאפשר לתפוס חבילות רשת רצות על המכונה. בחרתי בכלי על פני Wireshark, מפני שהוא יותר מסודר מבחינת תצוגת התעבורה. בשונה מ-Wireshark, MNM **מציג את החבילות שנתפסו לפי התהליכים שיצרו את החבילות באופן מסודר בתצורת עץ**. לכן, קל יותר לסנן חבילות ולבדוק חבילות שנתפסו מתהליכים שאנחנו רוצים לבחור.

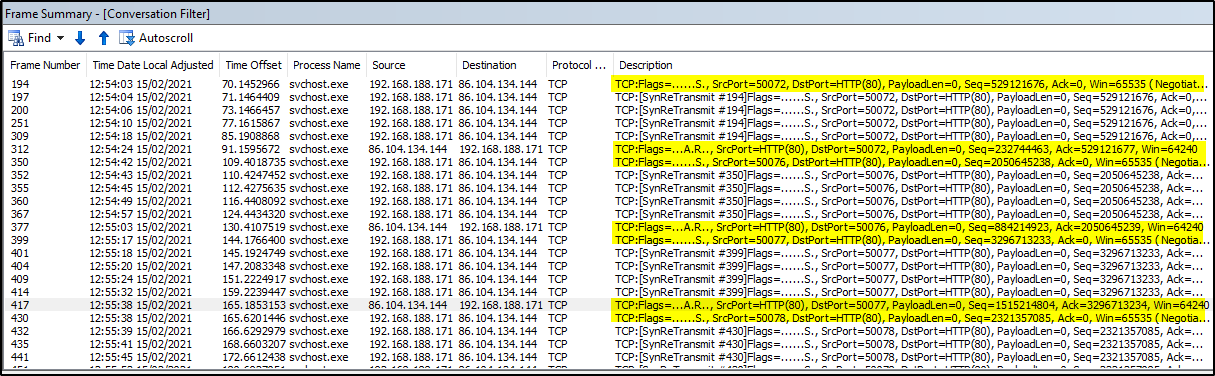
לאחר הרצת המאלוור, אפשר לבחון את תעבורת הרשת של ***svchost.exe***, תהליך הבן של המאלוור.



Figure XII: תצוגת MNM

כל בין חצי דקה ל-40 שניות svchost.exe מנסה ליצור תקשורת TCP עם

ה-IP החשוד **מאירלנד**, ***86.104.134.144***. התקשורת לא מצליחה והוא מנסה שוב ושוב ונתקע בלופ אינסופי.



# סיכום דברים / ממצאים

אחרי ניתוח המאלוור אלה הממצאים המסוכמים. **המאלוור לא מממש את המטרות שלו.** **תקשורת בין המאלוור** ל**בין** מה שיכול להיות **C&C Server לא מוצלחת**, כנראה להורדת קבצים או משיכת פקודות שהמאלוור זקוק להם בשביל שיפעל כראוי. למרות שהמאלוור אינו פעיל ניתן היה להוציא מידע על מטרת המאלוור, ניסיון לתקשורת ובחינת תהליכים שהוא יוצר.

1. ברגע שהמכונה נגועה, **המאלוור יכול להצפין את כל צורות הקבצים, ממסמכים, תמונות וסרטונים**. הוא יכול להצפין נתונים, ואולי בעל אפשרות **הפצה למחשבים אחרים** באותה רשת. כדי לקבל את הקבצים בחזרה, התוקף מבקש תשלום בביטקוין. סימן היכר נוסף של Ransomware הוא שתינתן מגבלת זמן קצרה כדי לשלם את הכופר או שנאבד את הנתונים לתמיד. ראוי לציין, כי בזמן **הרצה במכונה וירטואלית מקומית** בהגדרות שונות וגם **הרצה בארגז-חול מבוסס ענן** כמו Any.Run, **לא נראתה פעילות ולא הוצפנו קבצים**, **נראה כי המאלוור לא פעיל יותר**.
2. מנסה ליצור תקשורת החוצה בבקשת POST עם IP **86.104.134.144** בפורט **80**. ספציפית לכתובת: <http://86.104.134.144/main.php>, שמקורה באירלנד. (קוד ארץ: **IE**) כמו כן, קיימות בקשות DNS לכתובות שונות. בזמן הרצה על מכונה וירטואלית ובדיקה במנועים אחרים, לבקשה זו, **אין תגובה מהשרת המבוקש**, על כן המאלוור לא ממשיך בפעילותו.
3. יוצר קבצי TMP בתיקיית %TEMP% של מערכת ההפעלה ומוחק אותם לאחר שנעשה בהם שימוש. יוצר עותק של התהליך svchost.exe תחת אותה תיקייה, כנראה לתחזוקת המאלוור והרצה מחדש באם מערכת ההפעלה עשתה אתחול מחדש.
4. דרכי התמודדות במערכות הגנה ארגוניות:
5. שימוש ב-NAC: **בקרת גישה לרשת** (Network Access Control) מסייעת לארגונים ליישם מדיניות לשליטה במכשירים ובגישה של משתמשי קצה לרשת. בהקשר למאלוור הנחקר ניתן להגדיר מדיניות שלא תאפשר תקשורת עם ה-IP **86.104.134.144** וכן לכל בקשות ה-DNS שנמצאו בניתוח הנ"ל. דרך הגנה זו לבד כשלעצמה לא היעילה ביותר, לכן שילוב פתרון זה עם עוד דרכי הגנה יהיה יעיל ביותר.
6. SIEM: פלטפורמת *SIEM* אוספת **נתונים מפיירוולים** שעשויים להצביע על **תקשורת מוצלחת עם דומיינים או כתובות IP**. *SIEM* גם **מזהה תוכנות זדוניות** המשויכות לדומיינים אלה וכולל תוכנת אנטי-ספאם **המזהה קבצים שיכולים לגרום נזק לרשת הפנימית** - והכל בזמן אמת ומסוכם בהתראת אבטחה אחת. על-ידי הצלבת מידע זה עם *Indicators of Compromise* ציבוריים (*IoCs*), אנליסטים יכולים לזהות ולהגיב לפעילות רשת זדונית - במיוחד תוכנות כופר (כמו התוכנה הנחקרת) - במהירות ובדייקנות רבה יותר.

בהקשר למאלוור הנחקר, אנליסטים או האחראי יכולים לייצר כללים חדשים, שיתריעו כאשר תקשורת לדומיין החשוד באירלנד נעשתה החוצה ו/או פנימה. כמו כן, הוספת ההאשים של הקובץ לרשימה השחורה של מערכת ה-*SIEM*, תתריע אם הקובץ חדר לארגון.

1. [ראה נספח א' בשביל דוגמה מעשית](#_נספח_א':_דוגמה) לשימוש במערכת EDR. בהקשר לתוכנות זדוניות המועברות במוצר ***EDR***או כל מוצר אבטחה ארגוני, אנליסטים יכולים להפעיל במהירות ובדייקנות דרכי תגובה. אם התוכנה הזדונית נמצאה כלא ידועה, כלומר חדשה או אין חתימת אנטי-וירוס זמינה, השלב הבא בתהליך התגובה צפוי להיות חקירה נוספת. אנליסט יכול להעביר את הקובץ החשוד בממשק המשתמש של ה-***EDR*** לפלטפורמת ניתוח. משם ניתן לחקור ולהנדס לאחור את הקובץ ולבצע ניתוח סטטי מלא. ניתוח סטטי יציף לפני השטח את כל המבנים הקשורים למאלוור בקובץ, קשרים לפונקציונליות ידועות של תוכנות זדוניות, וכל טכניקות ההתחמקות שהמאלוור משתמש בהן. לאחר מכן ניתן לשלוח את הקובץ לניתוח נוסף בסביבה מאובטחת (*Sandbox*), אך הפעם עם סבירות גבוהה בהרבה להרצה מוצלחת מכיוון שטכניקות התחמקות התגלו בניתוח הסטטי. בסופו של דבר, אפילו עם גרסה זדונית לא ידועה, ניתן לנתח את האיום שהיא מציבה. עם מנוע **חוקי *YARA***, ניתן לבנות חוק *YARA* ולבדוק בפלטפורמות המתאימות אם החוק תואם. לאחר בדיקה מוצלחת, ווידוא שהחוק מזהה את האיום בקובץ החשוד, ניתן לייבא את חוק *YARA* לכלי הזיהוי, כולל מוצרי ***EDR***, כך שבפעם הבאה שהתוכנה הזדונית הלא ידועה תכה, כלי הזיהוי יזהו אותה באופן מיידי.
2. מניעת Open Mail Relay: קינפוג לא נכון של שרת אימיילים (*SMTP*), יכול להיות גן-עדן לספאמרים ותוקפים לארגון. ברמה בסיסית, ***Open Mail Relay*** הוא שרת *SMTP* המוגדר לאפשר לכל אחד באינטרנט לשלוח מיילים לארגון. שיטה זו מלאה חורי אבטחה ומהווה סיכון לארגון כך שמיילים מגורמים זדוניים יכולים להגיע לכל אחד בארגון. מספיק מייל אחד עם *תולעת* / *וירוס* / *טרויאני* / *RAT* בשביל לגרום לארגון נזק רב בארגון בגלל מיס-קונפיגרציה של שרת SMTP.

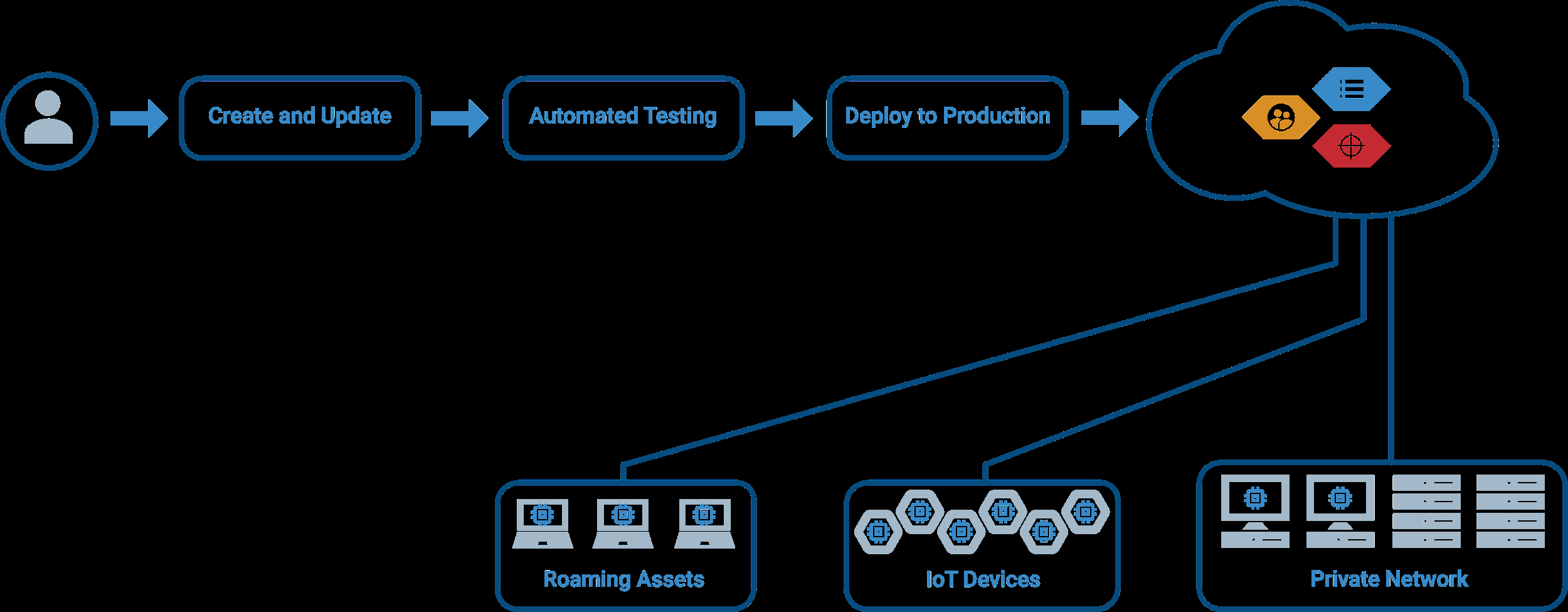
בהקשר למאלוור הנחקר: ניתן למתן סכנה זו בארגון ע"י קינפוג נכון של שרת SMTP בחברה.

1. יצירת רשימה שחורה של כתובות IP שייחסמו באם התקבלה הודעה מהכתובת. שמירה על הרשימה מעודכנת, תמנע מיילים לא רצויים מכתובות זדוניות.
2. הפעלת Reverse DNS Lookup, בודק אם הכתובת IP של השולח מתאימה לשם ולדומיין שהוגשו ע"י ה-SMTP.
3. הגבלת מספר חיבורים לשרת ה-SMTP, כולל מספר חיבורים המאופשר בו-זמנית, מספר חיבורים מקסימאלי, מספר חיבורים כולל. מיתון זה, יכול למנוע מתקפת ספאם ו-DoS על הארגון.

## נספח א': דוגמה להקמת EDR בענן, קביעת חוקי D&R, הרצת המאלוור במכונה אחרת וזיהוי בזמן אמת

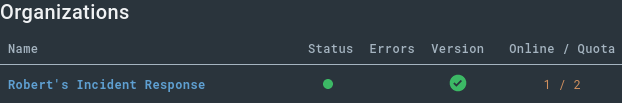
לצורך דוגמה של שימוש במערכת הגנה ארגונית, אשתמש בשירות החינמי של [LimaCharlie](file:///C:\Users\Roberto\Desktop\IR_Challenge\Mission%204%20-%20Malware%20Analysis\limcharlie.io), המאפשרת למשתמש ביתי לקבל ניסיון “Hands-On” על מערכת **EDR** ברמת **Enterprise** בחינם עד לשני משתמשי קצה.

**הסבר על המבנה:**

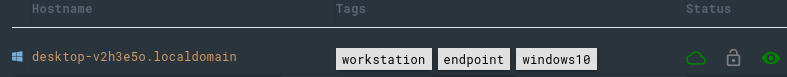


המערכת מבוססת ענן וכל ההתנהלות קורית דרך אפליקציית Web: [app.limacharlie.io](file:///C:\Users\Roberto\Desktop\IR_Challenge\Mission%204%20-%20Malware%20Analysis\app.limacharlie.io)

* אתחיל ביצירת ארגון. הארגון הוא דמו ומקבל עד שני משתמשי קצה:



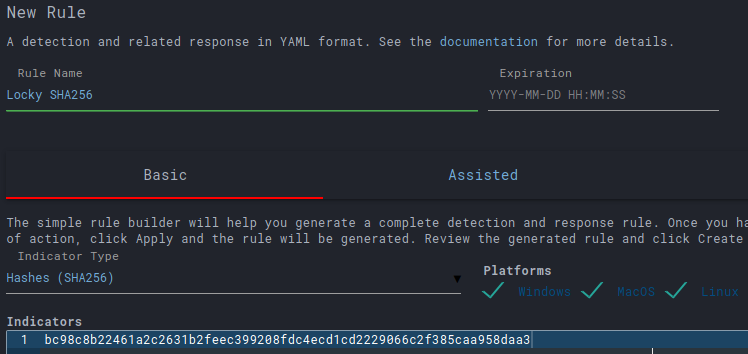
* אחרי שהארגון הוקם, התקנתי "סוכן" על מכונת ווינדוס שאני רוצה לעקוב אחריה. המכונה נוספה במערכת וכעת מנוטרת ע"י LimaCharlie תחת הארגון הנ"ל:



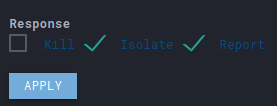
### יצירת חוקי Detection & Response (D&R)

#### זיהוי ע"פ SHA256:

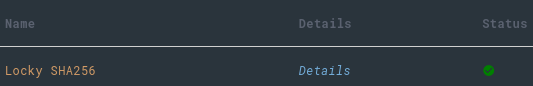
1. נותן לחוק שם.
2. בוחר סוג אינדיקטור (*Hashes SHA256*)
3. בוחר מערכות הפעלה שהחוק יחול עליהן (*Windows, Linux, MacOS*)
4. כותב את ה-IoC, במקרה זה, חתימת ההאש SHA256 של קובץ המאלוור - bc98c8b22461a2c2631b2feec399208fdc4ecd1cd2229066c2f385caa958daa3



1. בוחר את התגובה הרצויה: במקרה זה אני רוצה **לבודד** את המכונה מהרשת באם זוהה החוק ולקבל **דיווח**.



**החוק שנוצר ופרטיו:**



(שורות 1-6 בתבנית ה-Detection שנחתכו בצילום מגדירות את מערכות ההפעלה שהגדרנו קודם לכן.)

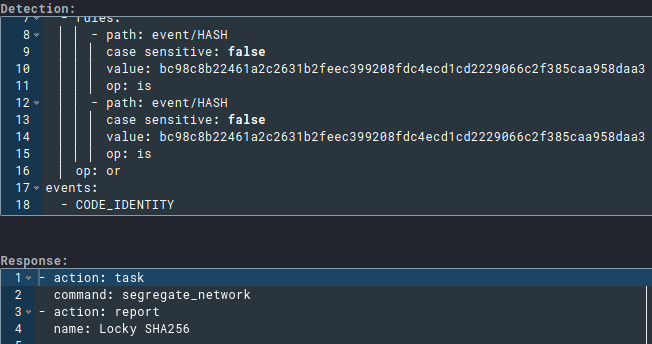
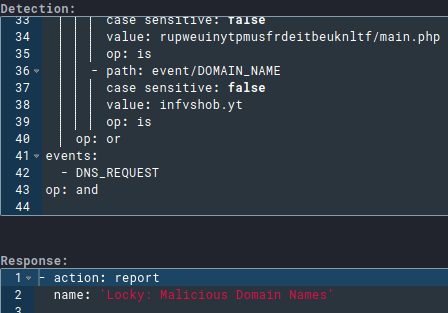
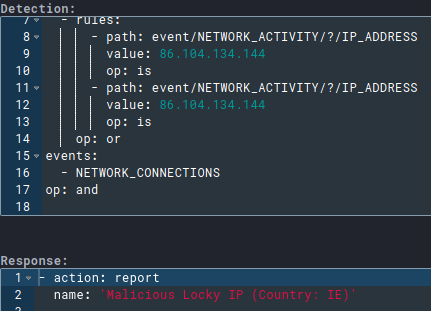


Figure VI: חוק D&R מלא

#### זיהוי ע"פ בקשות DNS (Domains) וכתובות IP

אעבור על אותם שלבים, ואכניס את ה-**IoC’s** הנותרים בתור **חוקי D&R**. (התהליך דומה לכן אצמצם בצילומי מסך והסברים ואגיע לזיהוי בזמן אמת בעת הרצה).

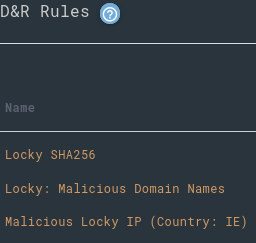
🡨 תמונה חלקית מחוק זיהוי בקשות DNS פועל ע"י זיהוי *DOMAIN\_NAME* בקטגוריית *DNS\_REQUEST* אשר הערכים שלה הם שמות הדומיין החשודים.



🡨 חלק מחוק זיהוי חיבורים לכתובות IP. פועל ע"י בדיקת אירוע *NETWORK\_ACTIVITY* בקטגוריית *NETWORK\_CONNECTIONS* אשר הערכים שווים לכתובות IP הזדוניים שסופקו.

התגובות לאירועים אלה כרגע נשארות כ"דווח" בלבד, בלי בידוד מהרשת, ובלי הריגת התהליך בשביל קבלת כמה שיותר מידע ולתת למאלוור לרוץ על המכונה ללא הפרעות לצורך ניתוח מאוחר יותר.

**כל החוקים שנוצרו מסודרים בלשונית ה-*D&R Rules*:**



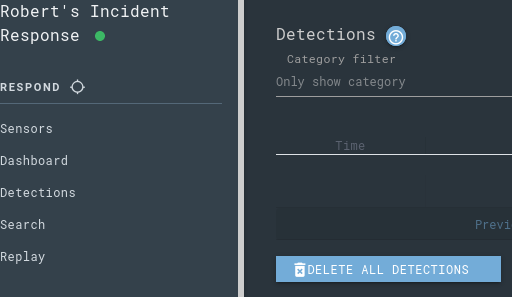
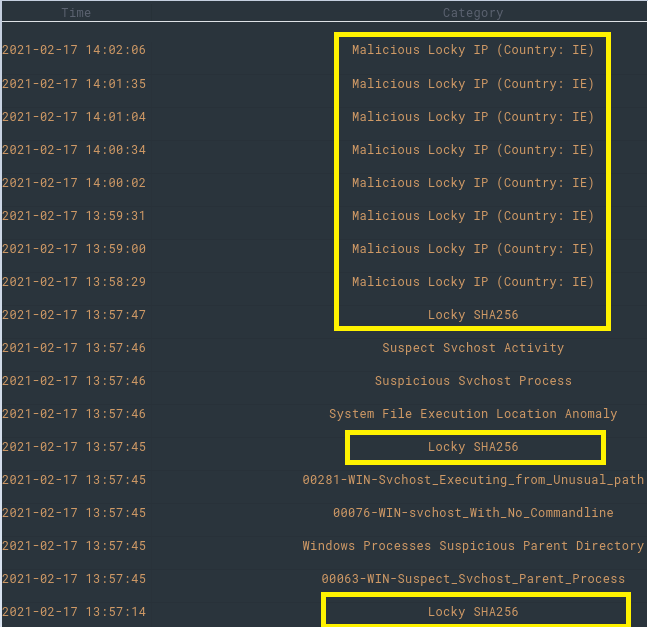
בשביל לבדוק את אמינות החוקים שנוצרו, התחלתי עם לשונית התראות ריקה ונקייה לחלוטין טרם הרצה:

Figure VII: לשונית התראות ריקה טרם הפעלה

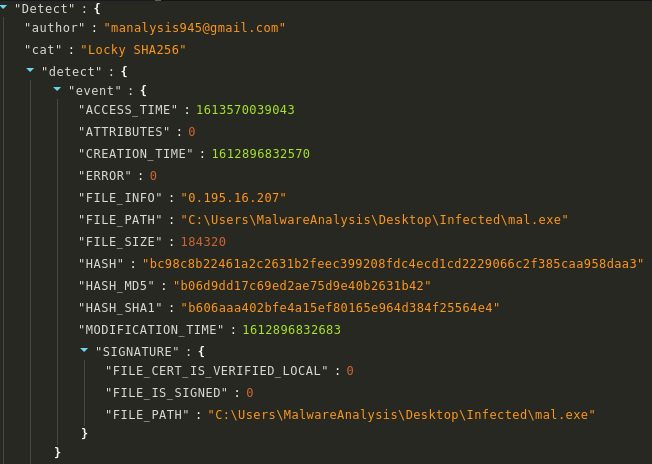
### הרצת המאלוור במכונה ובדיקת אמינות החוקים שנוצרו

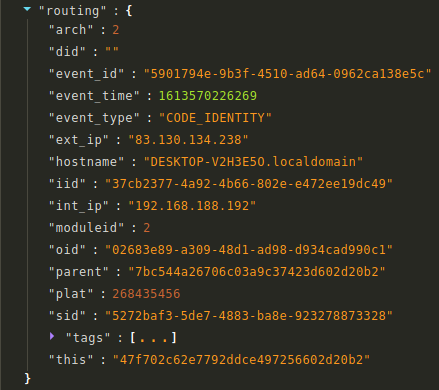
בנוסף לחוקי ה-D&R, ל-LimaCharlie יש אפשרות להתקין תוספים ממקורות שונים שתופסים פעילות חשודה בקלות כגון תוספי חוקי ***Sigma*** ו-***Soteria*** נותנים חשיפה לחוקי Sigma ו-Soteria בקלות ובצורה מסודרת ועוד... התקנתי כמה מהם כדי לראות איך הם משפיעים / מזהים את המאלוור הנחקר. כתוצאה מכך, לשונית ה-***Detections*** התמלאה בהתראות שלא אני יצרתי, אך ניתן ללמוד מהם המון על אופן הפעולה של המאלוור, כמו יצירת תהליך svchost.exe מתוך תהליך הורה חשוד, יצירת תהליכי מערכת מתיקיות חשודות, svchost ללא שורת פקודה (*חריג*), הרצת svchost.exe / ***קובץ מערכת*** מתיקייה חריגה (*Temp*).

**בהקשר לחוקים שיצרתי**, אלה עבדו והופיעו בלשונית ההתראות בזמן אמת בעת הרצה.

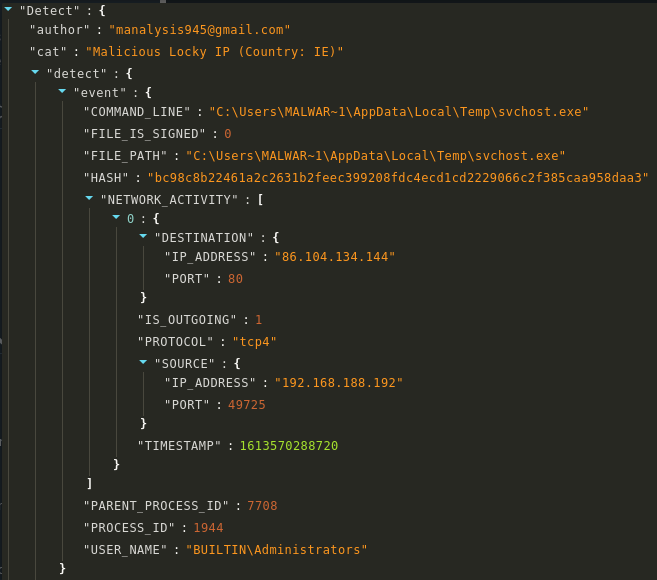
🡨 תמונה שנלקחה מתוך מערכת *LimaCharlie*, בלשונית *Detections*. אלה התראות שהתקבלו בזמן אמת ומסודרות בסדר כרונולוגי ע"פ זמן ההתרחשות. מימין לעמודת הקטגוריה (*חתוך בתמונה בגלל מקום וגודל תמונה*) יש עמודה עם שם המכונה (DESKTOP-V2H3E5O.localdomain) ממנה ההתראה נוצרה. ניתן לראות את כל ההתראות שמערכת ווינדוס העלתה אחרי שהרצתי את המאלוור. אלה המסומנות בצהוב הן התראות מחוקים שיצרתי קודם לכן. התראות אחרות הן מתוספים אחרים ממקורות אחרים. בהמשך, אפתח את שתי ההתראות מהחוקים שיצרתי ואנתח אותם לעומק.

#### Locky SHA256 D&R Rule

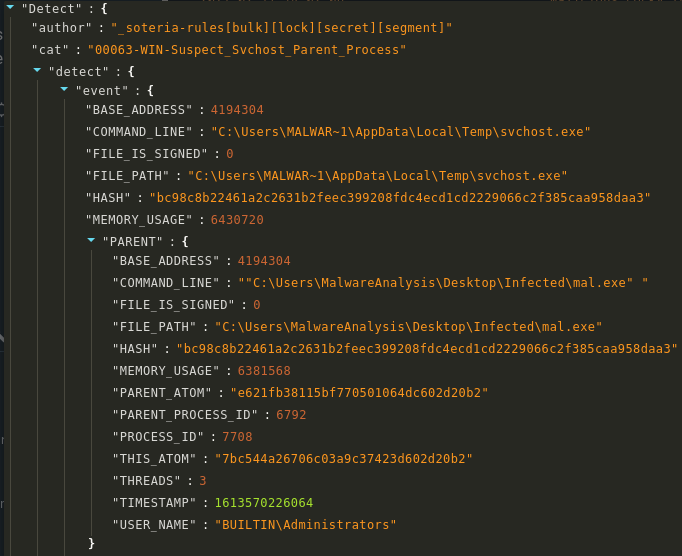
🡨 לחיצה על אחת ההתראות תפתח חלון עם פרטים נוספים ועמוקים בנוגע לאופי ההתראה כגון, *שם הקובץ*, *גודל*, *נתיב*, *זמן גישה לאירוע* + *זמן ההתראה שנוצרה* בזמן UNIX, *ההאש שהפעיל את ההתראה* וגם *MD5* ו-*SHA1*, *בדיקה האם הקובץ חתום*.

🡨 פרטים נוספים כגון מספר האירוע, זמן האירוע ב-UNIX, סוג האירוע (CODE\_IDENTITIY), כתובת IP חיצונית ופנימית, Hostname.

#### Malicious Locky IP (Country: IE)

🡨 התראה שנוצרה מחוק לתפיסת תקשורת עם IP המזוהה כזדוני מאירלנד בכתובת ***86.104.134.144***. ניתוח מעמיק מגלה כי יוצר התקשורת הוא התהליך svchost.exe הזדוני. מנסה לתקשר עם ה-IP **בפורט 80** בפרוטוקול **TCP4**.

#### 00063-WIN-Suspect\_Svchost\_Parent\_Process

🡨 לצורך הדגמה, אתעמק גם בהתראה זו שנוצרה ע"י טריגר מתוך תוסף הנותן גישה לכללי Sigma לחשיפת מאלוורים ואירועים זדוניים פוטנציאליים. קטגוריית **Suspect Svchost Parent Process**, מתארת מצב, כמו השם שלה, בו תהליך svchost.exe נוצר מתהליך-אב חשוד. svchost.exe לא צריך לפעול מתוך שום תהליך אחר חוץ מ-sercives.exe כך שהתראה נוצרת מזיהוי תהליך-אב חשוד. תהליך-האב ממנו נוצר svchost.exe הוא **קובץ המאלוור הזדוני**. יותר למטה בחלון ההתראה מופיעים פרטים על המערכת ממנה הגיעה ההתראה וגם [לינק](https://docs.soteria.io/detection/858fa2c9) לפרטים נוספים (*Documentation*) על הקטגוריה. בלינק יש הסבר מפורט על ההתראה שהתקבלה וקישורים לשימושים ב-MITRE-ATT&CK.