



נוזקה בתוך תמונה - הסתרת קוד זדוני בקבצי גרפיקה

וזיהוי באמצעות YARA

מאת עמית גבאי

הקדמה

בעולם אבטחת המידע המודרני, קבצי תמונה נחשבים בעיני רבים לתוכן "תמים" - קבצים שרק מוצגים על המסך ואינם יכולים לגרום נזק. תפיסה זו, אף שהיא נכונה ברובה, מוטעית במקרים רבים מספיק כדי להפוך קבצי גרפיקה לאחד מכלי ההסתרה המעניינים עבור תוקפים.

מאמר זה סוקר את הדרכים העיקריות שבהן ניתן להשתמש בקובץ תמונה כאמצעי הובלה או הסתרה של תוכן זדוני, את ההיסטוריה של הטכניקה, ואת האופן שבו צוותי אבטחה, ובפרט באמצעות מנוע חוקים בשם YARA - מזהים ומסכלים מתקפות מסוג זה.

חשוב להבהיר כבר בפתיחה: המאמר נכתב למטרות הבנה, מודעות, חינוך והגנה. הוא אינו כולל קוד או payload פונקציונלי או הוראות ליצירת נזקה.

למה בכלל אפשר "להחביא" קוד בתוך תמונה?

כדי להבין את האיום, צריך קודם להבין משהו בסיסי על מבנה קבצים דיגיטליים: כמעט כל פורמט קובץ, כולל JPEG, PNG, GIF, BMP ואחרים - הוא בסופו של דבר רצף של Bytes המאורגן לפי מבנה (format) מוגדר מראש. תוכנת הצפייה או העיבוד בתמונה (הדפדפן, מציג התמונות של מערכת ההפעלה, או ספרייה כגון libpng, libjpeg או ImageMagick) יודעת לקרוא את רצף הבייטים, לפרש את הכותרות (headers), את מרחב הצבעים, את הדחיסה ומבני הנתונים השונים, ולבסוף להפיק את הייצוג החזותי של התמונה.

הבעיה, ומכאן נובעת כל הסוגיה, היא שפורמטים ותוכנות עיבוד רבות מאפשרים קיום של מידע שאינו חלק ישיר מהייצוג החזותי.



לדוגמה:

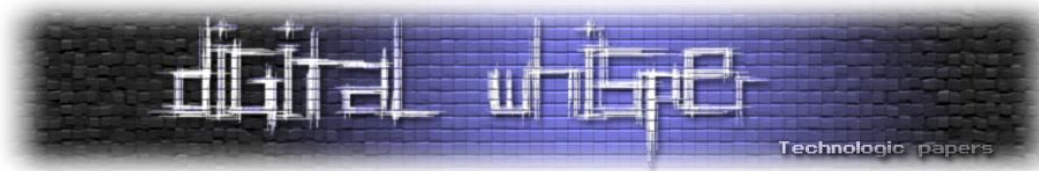
- מטא-דאטה - שדות מידע המיועדים לתאר את התמונה, כגון המצלמה שצילמה אותה, תאריך, מיקום גיאוגרפי או הערות של הצלם.
 - אזורי ריפוד (padding) - בייטים שעשויים להופיע במבני קבצים מסוימים מסיבות טכניות, כגון יישור או דרישות מבניות, ואינם בהכרח חלק מהייצוג החזותי.
 - מבנה מקונן (chunk-based) - פורמטים כמו PNG בנויים מ"נתחים" (chunks) בעלי מבנה מוגדר. קיימים סוגי chunks שאינם חלק מהפיקסלים עצמם, ובמקרים מסוימים ניתן לנצל שדות או מבנים שאינם נחוצים להצגת התמונה כדי לשאת מידע נוסף.
 - גמישות בפענוח - תוכנות מסוימות אינן קפדניות לחלוטין לגבי כל חריגה מהתקן. אם parser מקבל קלט שאינו נחוץ להצגת התמונה ומתעלם ממנו, ניתן לעיתים להשתמש בהתנהגות זו כדי לשלב מידע נוסף בקובץ.
- התכונות הללו, שנועדו במקור לגמישות, להרחבה או לצורכי metadata, הופכות לנקודות שבהן ניתן להסתיר מידע שאינו קשור ישירות לתצוגה החזותית.
- חשוב להבחין כבר כאן בין שני מושגים שונים: הסתרת מידע בתוך תמונה אינה שקולה להרצת קוד. תמונה יכולה להכיל מידע שנראה חשוד או אפילו קוד בינארי, מבלי שהקוד הזה ירוץ אי פעם. כדי שהמידע המוסתר יהפוך לחלק משרשרת תקיפה, נדרש מנגנון נוסף. למשל, תוכנה זדונית שכבר קיימת במערכת ויודעת לחלץ אותו, או פגיעות בתוכנה שמעבדת את הקובץ.

הטכניקות המרכזיות

סטגנוגרפיה קלאסית - Least Significant Bit (LSB)

סטגנוגרפיה היא, במקור, תחום אמנות הסתרת מידע בתוך מדיום אחר, כך שקיומו של המידע המוסתר אינו ניכר לעין. בהקשר הדיגיטלי, אחת השיטות המוכרות היא מניפולציה על "הביט הפחות משמעותי" (LSB) של ערכי הפיקסלים.

תמונה דיגיטלית מורכבת מפיקסלים, וכל פיקסל מיוצג בדרך כלל על ידי ערכי צבע - למשל אדום, ירוק וכחול (RGB). כאשר כל ערוץ מיוצג על ידי 8 ביטים, ערכו נע בין 0 ל-255. שינוי הביט האחרון, כלומר הביט הפחות משמעותי, משנה את הערך במספר קטן מאוד ולכן בדרך כלל יוצר שינוי חזותי שקשה מאוד להבחין בו בעין האנושית.



דוגמה להמחשה: נניח שיש לנו פיקסל שחור לחלוטין שערכי ה-RGB שלו הם (0,0,0). אם נשנה את הביט האחרון בערוץ האדום, הערך יכול להפוך ל-(1,0,0). מבחינה חזותית השינוי זעיר ביותר, אך תוכנה שיודעת כיצד לקרוא את אותם ביטים יכולה לאסוף אותם ולבנות מהם רצף מידע.

באופן דומה, אם שינוי כזה מתבצע במספר גדול של פיקסלים, ניתן ליצור ערוץ סמוי שיכול לשאת טקסט או נתונים בינאריים.

Lossy מול Lossless: טכניקות LSB המבוססות ישירות על ערכי הפיקסלים דורשות שהמידע שהוטמע יישמר באופן עקבי. לכן הן מתאימות במיוחד לפורמטים השומרים את ערכי הפיקסלים ללא אובדן, כגון PNG או BMP. לעומת זאת, בפורמטים עם דחיסה מאבדת מידע (Lossy), כמו JPEG, שלבי הדחיסה והקידוד מחדש עלולים לשנות את ערכי הפיקסלים ולפגוע במידע שהוטמע.

חשוב להדגיש LSB steganography כשלעצמה אינה מריצה קוד. היא רק מסתירה מידע. כדי שהמידע המוסתר יהפוך ל-payload פעיל, צריך מנגנון נוסף שיחלץ אותו ויעביר אותו לשלב הבא. לדוגמה, תוכנה זדונית שכבר נמצאת במחשב יכולה לאתר את התמונה, לחלץ ממנה את המידע ולהשתמש בו בהמשך שרשרת התקיפה.

דוגמה אמיתית לכך היא Worok, [קמפיין שתועד על ידי ESET](#). אחד הרכיבים שלו, PNGLoad, חיפש קבצי PNG והפיק מהם מידע באמצעות הביטים הפחות משמעותיים של ערוצי הצבע וה-alpha ולאחר מכן עיבד את המידע שהתקבל. כך התמונה שימשה למעשה כמעין מיכל למידע נוסף, בעוד שהקובץ עצמו יכול להישאר תמונה תקינה למראה.

ניצול לרעה של מטא-דאטה EXIF ודומיו

תקן EXIF (Exchangeable Image File Format) מאפשר לשלב בקובץ תמונה מידע תיאורי, כגון: דגם מצלמה, הגדרות הצילום, מיקום, GPS, תאריך ולעיתים גם שדות טקסט נוספים.

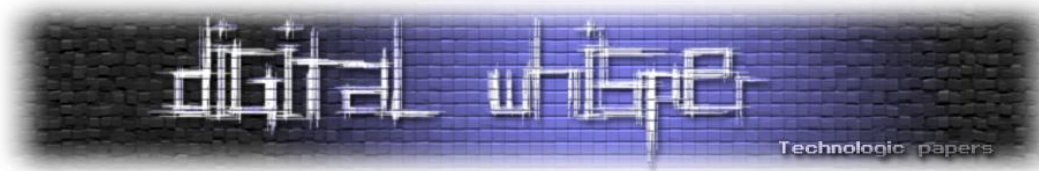
קובץ EXIF תמים של תמונה שצולמה בסמארטפון עשוי להכיל מידע כגון:

```
Make: Apple  
Model: ePhone 13  
GPS Latitude: 32.0853
```

עם זאת, העובדה שניתן להכניס טקסט לשדה EXIF אינה אומרת שהטקסט יהפוך אוטומטית לקוד.

לדוגמה, אם שדה EXIF מכיל מחרוזת שנראית כמו קוד PHP:

```
Model: <?php system($_GET['cmd']); ?>
```



עצם הופעת המחרוזת בתוך הקובץ אינה גורמת לה לרוץ. הסיכון נוצר כאשר אפליקציה או שרת מעבירים את הנתון למנגנון ביצוע בצורה לא בטוחה - למשל כאשר תוכן שהגיע מהמשתמש משולב בתוך פקודת מערכת, שאילתה או מנגנון פרשנות ללא הפרדה וסינון מתאימים.

במילים אחרות, מדובר במקרה שבו התמונה משמשת אמצעי הובלה לנתון זדוני, והפגיעות האמיתית נמצאת ברכיב שמעבד את הנתון.

היבט נוסף שהופך metadata למעניין הוא שהמידע נמצא באזור נפרד יחסית מהייצוג החזותי. לכן ניתן לעיתים לשמר אותו גם כאשר התמונה עצמה עוברת תהליכי דחיסה, כל עוד תוכנת העיבוד אינה מסירה את ה-metadata.

קבצי Polyglot - קובץ שהוא "שניים באחד"

טכניקה מתוחכמת יותר היא יצירת קובץ פוליגלוט - קובץ שניתן לפרש באופן תקין בשתי דרכים שונות על ידי שני parsers או מנגנוני עיבוד שונים.

לדוגמה, ניתן ליצור קובץ שהוא בו-זמנית:

- קובץ JPEG תקין מבחינת תוכנת הצפייה בתמונות.
- קובץ ZIP תקין מבחינת כלי חילוץ מתאים.
- ובתרחישים מסוימים, גם קובץ המכיל תוכן קוד או סקריפט שניתן לפרש על ידי רכיב אחר במערכת.

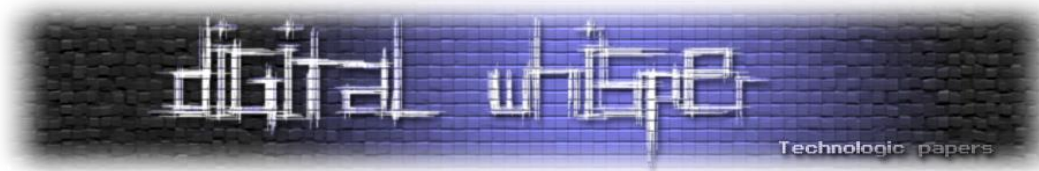
הדבר אפשרי משום שפורמטים שונים משתמשים במבנים שונים כדי לזהות את תחילתו וסופו של המידע הרלוונטי. כאשר שני parsers מתייחסים לאותם בייטים בצורה שונה, ניתן לעיתים ליצור קובץ שעובר בהצלחה את הבדיקות של מערכת אחת, אך מקבל משמעות שונה לחלוטין אצל מערכת אחרת.

סיכון משמעותי נוצר כאשר שרת בודק קובץ לפי מנגנון אחד, למשל כ"תמונה", אך רכיב אחר במערכת מתייחס אליו לפי פורמט או תוכן אחר.

חשוב להדגיש שגם כאן, עצם קיומו של polyglot אינו אומר שהקוד שבו ירוץ. ההרצה תלויה באופן שבו מערכת היעד מפרשת את הקובץ ובאילו רכיבים היא מאפשרת להשתמש בו.

ניצול פגיעויות בספריות פענוח תמונה (Image Parsers)

זוהי אולי הטכניקה הישירה ביותר מבחינת הקשר בין קובץ התמונה לבין הרצת קוד, מכיוון שהיא יכולה לנצל חולשה בתוכנה שמעבדת את הקובץ עצמו.



ספריות כמו GDI+, libpng, libjpeg של מייקרוסופט, ImageMagick ואחרות הן תוכנות פענוח תמונה מורכבות. חלקן כתובות בשפות low-level כמו C ו-C++ שבהן קיימים סיכונים הקשורים לניהול זיכרון. לאורך השנים התגלו בספריות מסוג זה פגיעויות כגון buffer overflow, out-of-bounds access ושגיאות parsing אחרות.

כאשר קיימת פגיעות ניתנת לניצול, תוקף עשוי ליצור קובץ תמונה שתוכנן לגרום ל-parser לבצע פעולה בלתי צפויה. בהתאם לאופי הפגיעות, התוצאה יכולה לנוע מקריסה של התוכנה ועד להרצת קוד תחת ההרשאות של התהליך המעבד את הקובץ.

דוגמה מפורסמת היא ImageTragick, שם שניתן לקבוצת פגיעויות ב-ImageMagick שהתגלתה ב-2016. אחת מהן, CVE-2016-3714, אפשרה בתנאים מסוימים הרצת פקודות דרך עיבוד קובץ תמונה מעוצב במיוחד. לפי [NVD](#), הפגיעות נבעה בין היתר מעיבוד לא בטוח של קלט והעברה לפונקציונליות delegate.

במקרה כזה, בניגוד ל-LSB, הבעיה היא לא שמידע הוחבא בתוך התמונה וממתין לתוכנה אחרת, אלא ה-parser עצמו הופך להיות חלק ממשטח התקיפה.

דוגמאות מהעולם האמיתי

- לאורך השנים תועדו מספר מקרים בולטים שבהם טכניקות אלו נוצלו במתקפות אמיתיות:
- (2015) Stegoloder - משפחת נזקה שתועדה משתמשת בסטגנוגרפיה כדי להסתיר רכיבים של שרשרת התקיפה בתוך קבצי תמונה, ובכך להקשות על זיהוי התוכן הזדוני.
 - (2022) Worok - קמפיין שבו רכיב בשם PNGLoad השתמש בסטגנוגרפיה בקובצי PNG. לפי [ESET](#), הרכיב חילץ מידע מהביטים הפחות משמעותיים של ערוצי הצבע וה-alpha ולאחר מכן עיבד את המידע שהתקבל.
 - (2016) ImageTragick - סדרת פגיעויות ב-ImageMagick ובהן [CVE-2016-3714](#), שאפשרו בתנאים מסוימים לעבד קבצים מעוצבים באופן שהוביל להרצת פקודות על המערכת.
 - שימוש ב-Polyglot בהעלאת קבצים באתרי אינטרנט - חוקרי אבטחה תיעדו לאורך השנים תרחישים שבהם מנגנוני העלאת קבצים חלשים אפשרו להעלות קבצים שנראו כתמונות אך הכילו מבנים נוספים. הסיכון נוצר כאשר רכיב אחר בשרת פירש את אותו קובץ בצורה שונה מזו שהתכוון אליה מנגנון ההעלאה.
 - Malvertising - קמפיינים של פרסום זדוני יכולים להשתמש בתוכן מדיה ובשרשראות אספקת פרסום כדי להעביר קוד או להפנות משתמשים לתוכן זדוני. יש להבחין בין מצב שבו קוד ממש מוטמע בתוך נתוני התמונה לבין מצב שבו התמונה או משאב הפרסום משמשים כחלק משרשרת תקיפה רחבה יותר.

מקרים אלו ממחישים שמדובר באיום אמיתי ולא בתיאוריה בלבד. עם זאת, חשוב גם להבין שהטכניקות אינן שקולות זו לזו: סטגנוגרפיה, polyglot וניצול parser מייצגים מודלים שונים של תקיפה ודורשים מנגנוני הגנה שונים.



[בתמונה - תרשים זרימה שממחיש כיצד עבד קמפיין ה-Stegolader. [מקור](#)]

נוזקה בתוך תמונה - הסתרת קוד זדוני בקבצי גרפיקה

זיהוי באמצעות YARA

www.DigitalWhisper.co.il

שרשרת המתקפה - איך זה בפועל מתחבר יחד

חשוב להבין שברוב המקרים, "התמונה הזדונית" אינה המתקפה כולה, אלא חוליה אחת בשרשרת ארוכה יותר, המכונה בעגה המקצועית Kill Chain. שרשרת טיפוסית עשויה להיראות כך:

1. **גישה ראשונית (Initial Access)** - הקורבן מקבל קובץ, קישור או תוכן שמוביל לתמונה או למערכת המעבדת אותה. לדוגמה, התמונה יכולה להגיע בדוא"ל, באתר אינטרנט או כחלק ממערכת להעלאת קבצים.

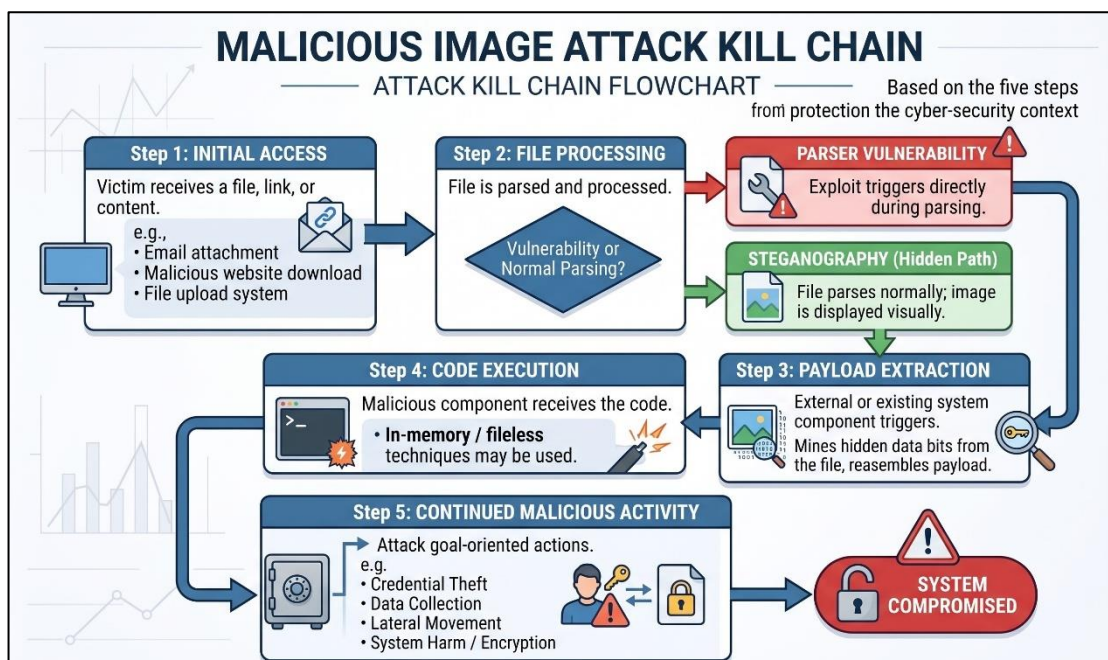
2. **עיבוד הקובץ** - אם מדובר בניצול פגיעות ב-parser עיצוב עצמו עשוי להיות הרגע שבו מתרחשת התקיפה. אם מדובר בסטגנוגרפיה, הקובץ יכול לעבור עיבוד תקין לחלוטין מבלי ששום קוד ירוץ.

3. **חילוץ ה-payload** - במקרה של סטגנוגרפיה, רכיב שכבר קיים במערכת יכול לפנות לתמונה, לחלץ ממנה את הביטים או הנתונים המוסתרים ולהרכיב מהם מידע נוסף.

4. **הרצה** - אם המידע שחולץ הוא קוד שנועד לביצוע, רכיב זדוני יכול להעביר אותו לשלב ההרצה. בחלק מהתרחישים התוקף עשוי לנסות לצמצם את הכתיבה לדיסק ולהשאיר חלק מהפעילות בזיכרון - טכניקה המקושרת לעיתים למונח fileless malware.

5. **המשך הפעילות הזדונית** - בהתאם למטרת התוקף, הדבר עשוי לכלול איסוף מידע, גניבת credentials, יצירת תנועה רוחבית, פגיעה במערכות או הצפנת מידע.

הבנת השרשרת הזו חשובה כי היא ממחישה מדוע קובץ תמונה יחיד, שנשלף ובדק ב-sandbox, לא תמיד "יראה" זדוני. לעיתים הזדוניות אינה נמצאת בתמונה עצמה, אלא במערכת היחסים בין התמונה לבין הרכיבים שמעבדים אותה.



נוזקה בתוך תמונה - הסתרת קוד זדוני בקבצי גרפיקה

זיהוי באמצעות YARA

www.DigitalWhisper.co.il

שיטות זיהוי והגנה

ניתוח סטטי (Static Analysis)

ניתוח סטטי הוא בדיקת הקובץ מבלי להריץ אותו. הבדיקה יכולה לכלול את מבנה הקובץ, הכותרות, גודל הקובץ, chunks, metadata, חתימות וסימנים לנתונים שאינם תואמים את הפורמט. כלים לניתוח סטטי יכולים לחפש סימנים כגון:

- ✓ פער בין מבנה הקובץ הצפוי לבין גודלו בפועל.
- ✓ נוכחות של חתימות (magic bytes) של סוגי קבצים אחרים בתוך קובץ שאמור להיות תמונה.
- ✓ נתונים חריגים המופיעים לאחר סוף המבנה הלוגי הצפוי של הקובץ.
- ✓ אנטרופיה גבוהה באופן חריג באזורים שבהם לא צפוי להופיע מידע דחוס או מוצפן. חשוב לזכור שאנטרופיה גבוהה אינה הוכחה לנוזקה, מכיוון שגם דחיסה לגיטימית יכולה לייצר אנטרופיה גבוהה.
- ✓ שדות metadata החורגים מאורך או מבנה סביר, או מכילים תוכן שאינו אופייני לשדה התיאורי.
- ✓ מבנים שאינם תואמים את התקן או שאינם ניתנים לפענוח תקין על ידי parser תקין.

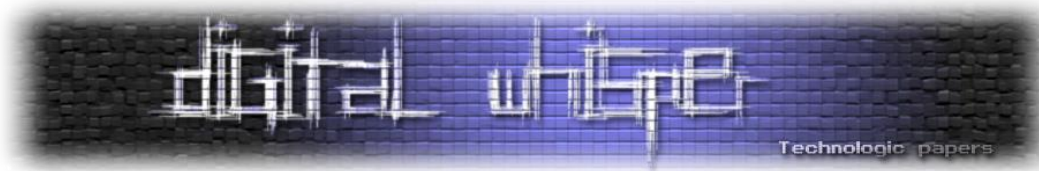
ניתוח דינמי (Sandbox / Dynamic Analysis)

בניתוח דינמי, הקובץ נפתח או מעובד בסביבה מבודדת הדומה למערכת הפעלה אמיתית, אך מנוטרת בקפידה. אם עיבוד התמונה גורם להתנהגות חריגה, כגון: יצירת תהליכים חדשים, קריאות רשת בלתי צפויות, יצירת קבצים, שינויים במערכת או קריסה חריגה של תוכנת העיבוד - הדבר עשוי להוות סימן אזהרה.

עם זאת, גם sandbox אינו פתרון מושלם. נזקות יכולות לנסות לזהות שהן רצות בסביבה מבודדת, לעכב את הפעילות שלהן או להפעיל אותה רק כאשר מתקיימים תנאים מסוימים.

חוקי YARA - "טביעות אצבע" לזיהוי דפוסים

זהו החלק שבו נכנס לתמונה (מקווה שאהבתם את משחק המילים) הכלי [YARA](https://yara.algomia.com/). YARA הוא כלי קוד פתוח שנועד לסייע לחוקרי נזקות ולצוותי אבטחה לזהות ולסווג קבצים על בסיס דפוסים טקסטואליים ובינאריים. באמצעות YARA ניתן לתאר מאפיינים של משפחות נזקה או של תבניות חשודות אחרות, ולאחר מכן להריץ את הכללים על קבצים או מקורות מידע מתאימים.



מה שהופך את YARA לרלוונטי במיוחד למקרה שלנו הוא הגמישות שלו: במקום להסתמך רק על hash או על חתימה מדויקת של קובץ אחד, ניתן להגדיר שילוב של מחרוזות, רצפי בייטים ותנאים לוגיים.

לדוגמה, כלל יכול לחפש קובץ שמזוהה כ-PNG, אך מכיל גם מאפיינים חריגים הקשורים לסוג קובץ אחר. עם זאת, עצם הופעת חתימת MZ בתוך PNG אינה הוכחה לכך שמדובר בנוזקה, אלא אינדיקציה בודדת שאפשר לשלב עם תנאים נוספים. מבנה בסיסי של כלל YARA כולל בדרך כלל:

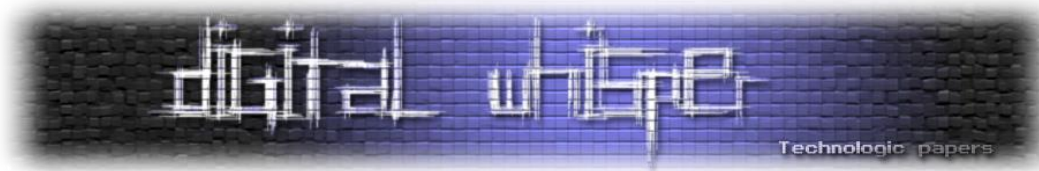
- meta - מידע תיאורי על הכלל עצמו, כגון מי כתב אותו, מה מטרתו ומהי רמת החומרה.
- strings - הגדרת דפוסי חיפוש. אלה יכולים להיות מחרוזות טקסט, רצפי בייטים הקסדצימליים או ביטויים רגולריים.
- condition - התנאי הלוגי שקובע מתי הכלל יתאים לקובץ. ניתן להשתמש בתנאים על נוכחות דפוסים, מספר התאמות, מיקום, גודל הקובץ ועוד.

להמחשה, כלל YARA כללי וברמת מושג (לא כזה שמזהה נוזקה אמיתית וספציפית) עשוי להיראות כך:

```
rule Suspicious_PNG_With_Embedded_Executable_Signature
{
  meta:
    description = "detects suspicious characteristics of a PNG containing a PE header"
    severity = "medium"
  strings:
    $png_header = { 89 50 4E 47 0D 0A 1A 0A }
    $pe_signature = "MZ"
  condition:
    $png_header at 0 and $pe_signature and filesize > 100KB
}
```

כלל כזה, במילים פשוטות, אומר: "אם הקובץ מתחיל בדיוק כמו קובץ PNG תקין, מכיל גם את המחרוזת MZ וגודלו גדול מסף מסוים -סמן אותו".

זהו כמובן כלל פשטני מאוד (הרחבה נוספת אפשר לראות [כאן](#)) הוא עלול להפיק התראות שווא, מכיוון שהמחרוזת MZ יכולה להופיע בקבצים לגיטימיים מסיבות שונות. כללים בשימוש אמיתי יהיו בדרך כלל מורכבים יותר ויכולים לשלב מיקום מדויק, מספר התאמות, מאפיינים מבניים ותנאים נוספים. YARA עצמו תומך בין היתר בתנאים על offsets, מספר התאמות וגודל קובץ.

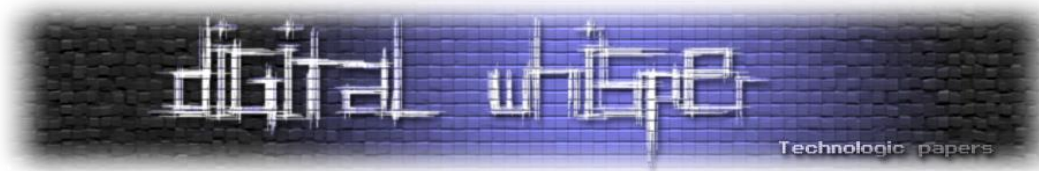


שילוב YARA בזרימת עבודה ארגונית

- בפועל, כללי YARA אינם חייבים לפעול בבידוד. הם יכולים להשתלב במגוון תהליכי אבטחה וחקירה:
- פלטפורמות ניתוח נוזקה ומאגרים כמו VirusTotal - מאפשרות להשתמש בחוקים לצורך זיהוי וסיווג דפוסים בקבצים.
 - מערכות EDR/XDR וכלי Threat Hunting - בחלק מהפתרונות ניתן לשלב חוקים או מנגנונים דומים לניתוח קבצים וארטיפקטים במערכות קצה.
 - מערכות סינון קבצים ושערי דוא"ל - ארגונים יכולים לשלב מנגנוני זיהוי מבוססי YARA כחלק מתהליך סריקת קבצים מצורפים.
 - מערכות העלאת קבצים - אתרים ומערכות המאפשרים העלאת תמונות יכולים לבצע סריקות סטטיות לפני קבלת הקובץ.
- חשוב להדגיש ש-YARA הוא מנוע זיהוי דפוסים, ולא מערכת הגנה מלאה בפני עצמו. האופן שבו הוא משתלב במוצר או במערכת מסוימת תלוי בארכיטקטורה של אותה מערכת.

מגבלות ואתגרים

- חשוב להכיר גם את המגבלות של הגישה הזו:
- התראות שווא (False Positives) - כלל רחב מדי עלול לסמן קבצים תמימים כחשודים, דבר שיוצר עומס על צוותי אבטחה ו-alert fatigue.
 - התחמקות (Evasion) - תוקפים יכולים לשנות מאפיינים של קובץ, להצפין או לדחוס payloads, לפצל מידע בין קבצים או לשנות את המבנה כך שלא תהיה התאמה לכלל קיים.
 - צורך בעדכון מתמיד - כלל מזהה את המאפיינים שהוגדרו בו. שינוי משמעותי בהתנהגות או במבנה של איום יכול לגרום לכלל קיים להחמיץ אותו.
 - ביצועים - הפעלת מספר גדול של כללים מורכבים על כמויות גדולות של נתונים דורשת משאבי חישוב, ולכן יש צורך באיזון בין עומק הסריקה, דיוק ומהירות.



המלצות מעשיות לארגונים

לסיכום החלק המעשי, אפשר לרכז מספר עקרונות הגנה מרכזיים:

הלבנת קבצים (CDR - Content Disarm and Reconstruction)

אחת משיטות ההגנה המעניינות היא Content Disarm and Reconstruction או CDR בקיצור. במקום לנסות לזהות כל payload אפשרי, מערכת CDR יכולה לנתח את הקובץ, לחלץ את התוכן הלגיטימי הנדרש ולבנות ממנו קובץ חדש.

במימושים המבצעים reconstruction מלא של התוכן החזותי, ניתן ליצור מחדש את התמונה על בסיס המידע החזותי בלבד. בכך ניתן להסיר metadata ותוכן נוסף שאינו חלק מהייצוג החזותי, כולל מידע שהוסתר בשדות metadata או במבנים אחרים.

עם זאת, CDR יכול להיות מיושם בכל מוצר באופן אחר. לכן אין להניח שכל פתרון CDR מסיר כל סוג אפשרי של מידע מכל פורמט.

עדכון שוטף של ספריות עיבוד תמונה

רוב ספריות עיבוד התמונות מתוחזקות ומקבלות עדכונים אבטחה. יש להקפיד על עדכון הספריות, כגון: libjpeg, libpng, ImageMagick וספריות אחרות בהתאם לגרסאות ולמוצרים הנמצאים בשימוש. הדוגמה של ImageTragick ממחישה מדוע parser של תמונה הוא חלק ממשטח התקיפה.

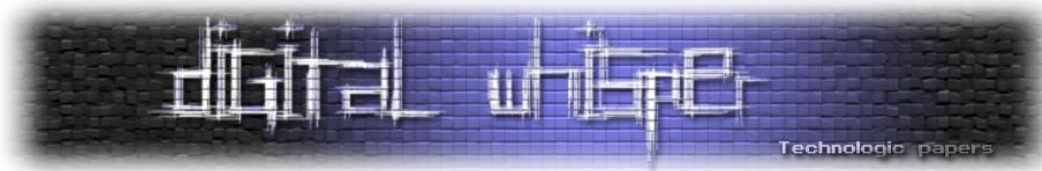
[CVE-2016-3714](#), למשל, תיאר מצב שבו עיבוד קלט מעוצב במיוחד יכול היה להוביל להרצת פקודות.

הגבלת סוגי קבצים ובדיקתם בפועל

אין להסתמך רק על סיומת הקובץ או על MIME type שסיפק הלקוח, מכיוון שניתן לזייף אותם. במקום זאת, מומלץ לשלב:

- ✓ בדיקת magic bytes / file signature
- ✓ Parsing אמיתי של הפורמט
- ✓ בדיקות מבניות
- ✓ הגבלת גודל ומאפיינים חריגים
- ✓ במקרים מתאימים, המרה מחדש לפורמט נקי
- ✓ הרצת קבצים שהועלו בסביבה מבודדת

כאשר מערכת מאפשרת העלאת קבצים ממשתמשים חיצוניים, כדאי לעבד אותם בסביבה בעלת הרשאות מוגבלות ובידוד מתאים, כך שגם אם parser ייפגע, ההשפעה תהיה מצומצמת ככל האפשר.



שילוב שכבות הגנה מרובות (Defense in Depth)

YARA הוא כלי חזק, אך הוא יעיל במיוחד כחלק ממערך רחב יותר הכולל:

- ✓ ניתוח סטטי
- ✓ ניתוח דינמי.
- ✓ EDR/XDR
- ✓ CDR
- ✓ בידוד רשת
- ✓ עדכוני תוכנה
- ✓ הגבלת הרשאות
- ✓ הדרכת עובדים נגד פשינג
- ✓ ניטור התנהגותי

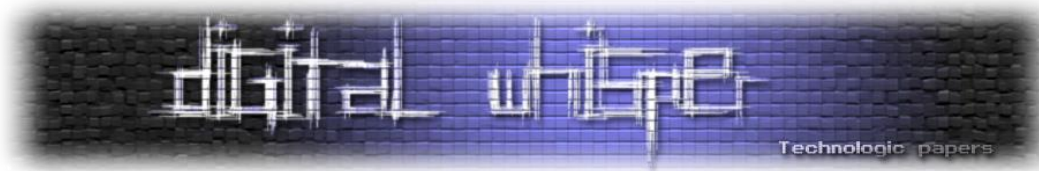
הרעיון המרכזי הוא לא להניח שמנגנון בודד יצליח לזהות כל וריאציה אפשרית של תקיפה. יש חשיבות אדירה בעדכון מתמיד של מאגרי כללי הזיהוי, שכן מדי יום נזקקות חדשות משתחררות לעולם. ארגונים יכולים להיעזר במאגרים ציבוריים של כללי YARA ובקהילת חוקרי האבטחה כדי לעדכן את יכולות הזיהוי שלהם. עם זאת, אין להעתיק כלל ציבורי באופן עיוור לסביבת production. יש לבדוק את מטרות, את איכות התנאים שלו ואת שיעור ההתראות השגויות שהוא מייצר בסביבה הארגונית.

סיכום

הסתרת קוד או מידע זדוני בתוך קבצי תמונה היא דוגמה מצוינת לאופן שבו תוקפים מנצלים את הפער בין מה שמשתמש הקצה "רואה", תמונה תמימה לכאורה, לבין מה שבאמת קיים ברמת הבייטים והמבנה של הקובץ.

בין אם מדובר בסטגנוגרפיה, ניצול לרעה של מטא-דאטא, קבצי polyglot או ניצול פגיעויות ב-Image-Parsers - המשותף לטכניקות הוא ההסתמכות על כך שהקובץ יעובד בדרך שלא תוכננה להתמודד עם תוכן עיין.

עם זאת, חשוב לא פחות להבחין בין הטכניקות. כפי שנסקר במאמר, סטגנוגרפיה אינה מריצה קוד בפני עצמה, metadata זדוני אינו הופך אוטומטית לקוד פעיל, polyglot תלוי באופן שבו רכיבים שונים מפרשים את הקובץ, ופגיעות parser היא בעיה מסוג שונה לחלוטין, שבה עצם עיבוד הקובץ עלול להוביל לניצול.



מנגד, כלים כמו YARA מספקים לצוותי ההגנה שפה גמישה לתיאור דפוסים חשודים. במקום להסתמך רק על זיהוי מדויק של קובץ שכבר מוכר, ניתן לחפש שילובים של מאפיינים טקסטואליים, בינאריים ומבניים. בכך YARA יכול להשתלב בתהליכי malware research, threat hunting וסריקת קבצים.

לצד זיהוי, טכנולוגיות כמו CDR מציעות גישה שונה: במקום לנסות לזהות כל דרך שבה ניתן להסתיר מידע, ניתן במקרים מתאימים לבנות מחדש את הקובץ מתוך התוכן הלגיטימי הנדרש ולסלק רכיבים שאינם נחוצים.

בסופו של דבר, ההבנה החשובה ביותר היא שתמונה אינה רק מה שהעין רואה. היא גם מבנה בינארי שמעובד על ידי תוכנה - וכל תוכנה שמקבלת קלט לא מהימן היא חלק ממשטח התקיפה.

הבנת שני הצדדים - הן טכניקות ההסתרה והן שיטות ההתגוננות והזיהוי, היא בסיס חיוני לכל מי שעוסק באבטחת מידע בעולם שבו כמעט כל קובץ, גם התמים ביותר למראה, עשוי להיות יותר ממה שנראה לעין.

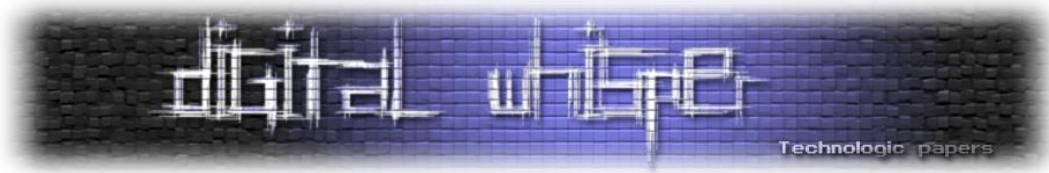
על המחבר

עמית גבאי הוא סטודנט למדעי המחשב בטכניון, שאת דרכו בעולמות הטכנולוגיה החל כבוגר תוכנית המצוינות "מגשימים" ולימודי י"ג בהנדסת תוכנה. הוא משלב ניסיון מעשי עשיר משירותו בצה"ל במערכות תקשורת ואבטחת מידע. לצד הזיקה העמוקה ליזמות וחדשנות טכנולוגית, עמית פועל באופן מתמיד להנגשת ידע מורכב לקהל הרחב, מתוך אמונה ששיתוף ידע הוא מנוע מרכזי בהתפתחות קהילת הסייבר והפיתוח בישראל.

מקורות מידע

סטגוגרפיה ונוזקות מבוססות תמונה

- Secureworks CTU, "Stegolader: A Stealthy Information Stealer" - הפרסום המקורי של יחידת המחקר שעסקה בנוזקה.
- Secureworks, "Stegolader Malware: A Wolf in Sheep's Clothing".
- SecurityWeek, "Information-Stealing Malware 'Stegolader' Hides in Image File".
- Virus Bulletin, "How It Works: Steganography Hides Malware in Image Files".



- [ESET / WeLiveSecurity, "Worok: the big picture"](#) - ניתוח טכני של השימוש ב-PNG וב-LSB במסגרת PNGLoad

- BleepingComputer, "Worok hackers hide new malware in PNGs using steganography"
- Schmid, S. et al., "Towards Modeling Hidden & Steganographic Malware Communication based on Images", Proceedings of the ACM Workshop on Information Hiding and Multimedia Security, 2025
- "Stegomalware: A Systematic Survey of Malware Hiding and Detection in Images, Machine Learning Models and Research Challenges", arXiv

ניצול פגיעויות בספריות עיבוד תמונה

- National Vulnerability Database, CVE-2016-3714 - הרשומה הרשמית של - [\(NVD\)](#)
- [imagnetragick.com](#) - האתר שהוקם סביב גילוי הפגיעויות
- Red Hat Customer Portal, "ImageTragick - ImageMagick Filtering Vulnerability"
- Bencode.io, "Technical Analysis of ImageTragick (CVE-2016-3714)"

YARA

- [YARA - התייעוד הרשמי](#) - תיעוד מבנה הכללים, strings, conditions ועוד
- [YARA - מאגר הקוד הרשמי](#)
- [VirusTotal YARA](#) - תיאור רשמי של מטרות YARA ושל יכולות התאמת הדפוסים.
- [YARA-X Documentation](#) - התייעוד העדכני של YARA-X, ממשיך דרכה המודרני של YARA.