בס"ד

PDF malware detection - preliminary research

הקדמה:

מבנה כללי של PDF:

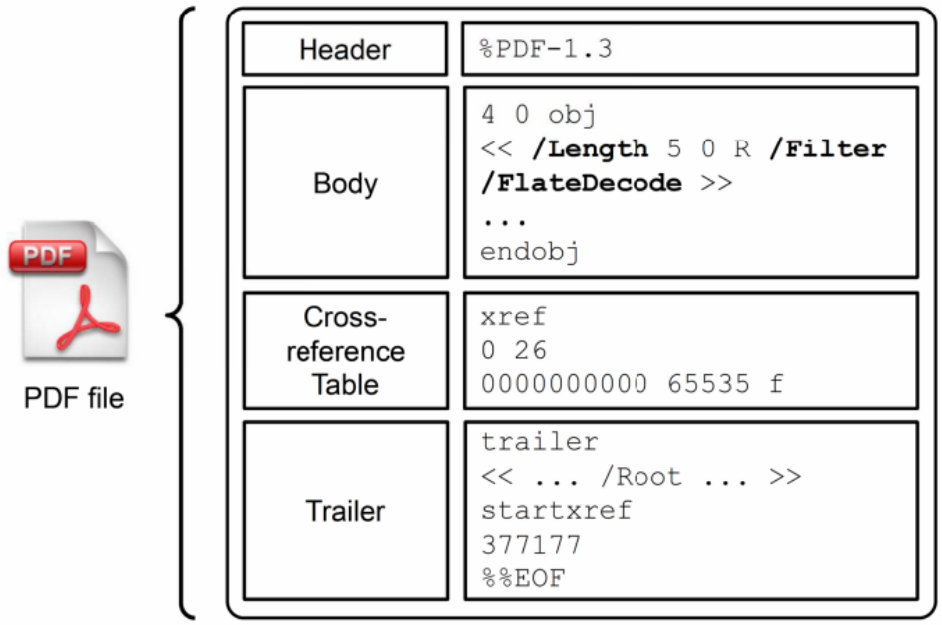
מבנה PDF מחולק ל2 חלקים עיקריים:

1. **General structure** - header, body, Cross-Reference (X-Ref) Table, Trailer
2. **Objects-** כפי שצוין קודם לכן, אובייקטים מזוהים בדרך כלל על ידי מספר, והם יותר המכונה באופן רשמי אובייקטים עקיפים. עם זאת, כל אלמנט בתוך הגוף נחשב בדרך כלל כאובייקט, גם אם מספר אינו מזהה אותו. כאשר חפץ אינו מזוהה במספר (כלומר, כאשר הוא חלק מאובייקטים אחרים), זה נקרא ישיר.
אובייקטים עקיפים מורכבים בדרך כלל משילוב של אובייקטים ישירים

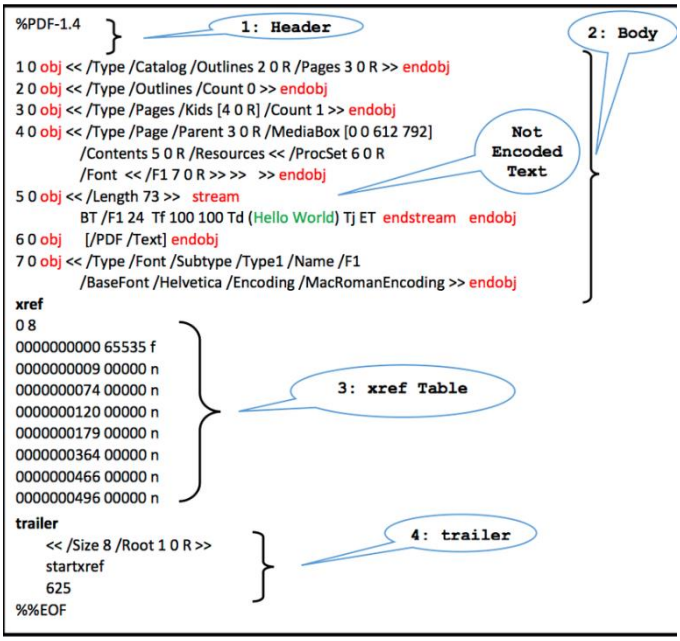
.

נסביר על כל אחד מהחלקים של **General structure** :

- **Header** הצגת שורת טקסט אחת המכילה מידע על גרסת קובץ הPDF
- **Body-** - רצף של אובייקטים המגדירים את הפעולות שמבצע הקובץ
יכול להכיל גם נתונים מוטעים דחוסים או לא דחוסים (לדוג' טקסט, תמונות, קוד)
לכל אובייקט יש מספר סימוכין ייחודי
- **טבלת הפניות מוצלבות (X-Ref).** רשימה של קיזוזים המציינים את המיקום של כל אובייקט ב הקובץ. רשימה כזו נותנת לקוראי PDF אינדיקציות מדויקות היכן להתחיל לנתח כל אחד מהם לְהִתְנַגֵד. טבלת ההפניה ההצלבה מוצגת על ידי הסמן xref, ואחריו רצף של מספרים, שהאחרון שלהם מציין את המספר הכולל של אובייקטים בקובץ. כל שורה בטבלה מתאים לאובייקט מסוים, אבל רק קווים המסתיימים ב-n קשורים לאובייקטים שכן מאוחסן באופן קונקרטי בקובץ. ראוי לציין שכל קוראי ה-PDF מנתחים רק את האובייקטים שאליהם מתייחסים טבלת X-Ref. לכן, אפשר למצוא חפצים שמאוחסנים בקובץ, אך אין לכך התייחסות בטבלה.
- **טריילר-** אובייקט מיוחד המתאר כמה אלמנטים בסיסיים של הקובץ, כמו האובייקט הראשון של הגרף (כלומר, היכן קוראי ה-PDF צריכים להתחיל לנתח את פרטי הקובץ). יתר על כך, הוא מכיל הפניות למטא נתונים של הקובץ, המאוחסנים בדרך כלל באובייקט בודד אחד. מילת המפתח טריילר תמיד מציגה את אובייקט הטריילר.



בס"ד



קבצי PDF מנותחים באופן הבא:

ראשית אובייקט הטריילר מנותח ראשון.
ולאחר מכן כל אובייקט בגוף נחקר על ידי המיקום שהוא צריך להיות לפי קרוס רפרנס
כל קובץ פי די אף מסתיים בEOF

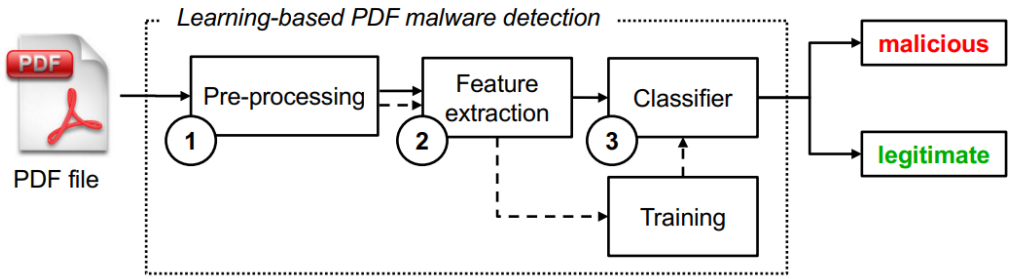
נדבר על 3 מתקפות עיקריות בPDF:

- **JavaScript** מתקפות בהן בתוך אובייקט או כמה אובייקטים מוכל סקריפט שיפעיל קטע קוד זדוני.
- **ActionScript** התקפות בהן יש קוד זדוני שרץ ברקע ברגע שקובץ הPDF נפתח
- **File Embedding** טכניקת ניצול המשתמשת בקבצים חיצוניים כגון exe על מנת לנתח את הקובץ ולהטמיע בו ניצול (למשל בתוך תמונה וכו')

Vulnerability	Vuln. Type	Exploitation Type
CVE-2008-0655	API Overflow (*Collab.collectEmailInfo*)	JavaScript
CVE-2008-2992	API Overflow (*util.printf*)	JavaScript
CVE-2009-0658	Overflow	File Embedding (JBIG2)
CVE-2009-0927	API Overflow (*Collab.getIcon*)	JavaScript
CVE-2009-1492	API Overflow (*getAnnots*)	JavaScript
CVE-2009-1862	Flash (Memory Corruption)	ActionScript
CVE-2009-3459	Malformed Data (FlateDecode Stream)	JavaScript
CVE-2009-3953	Malformed Data (U3D)	JavaScript
CVE-2009-4324	Use-After-free (*media.newPlayer*)	JavaScript
CVE-2010-0188	Overflow	File Embedding (TIFF)
CVE-2010-1240	Launch Action	File Embedding (EXE)
CVE-2010-1297	Flash (Memory Corruption)	ActionScript
CVE-2010-2883	Overflow (*coolType.dll*)	JavaScript
CVE-2010-2884	Flash (Memory Corruption)	ActionScript
CVE-2010-3654	Flash (Memory Corruption)	ActionScript
CVE-2011-0609	Flash (Bytecode Verification)	ActionScript
CVE-2011-0611	Flash (Memory Corruption)	ActionScript
CVE-2011-2462	Malformed U3D Data	JavaScript
CVE-2011-4369	Corrupted PRC Component	JavaScript
CVE-2012-0754	Flash (Corrupted MP4 Loading)	ActionScript
CVE-2013-0640	API Overflow	JavaScript
CVE-2013-2729	Overflow	File Embedding (BMP)
CVE-2014-0496	Use-After-Free (*toolButton*)	JavaScript
CVE-2015-3203	API Restriction Bypass	JavaScript
CVE-2016-4203	Invalid Font	File Embedding (TFF)
CVE-2017-16379	Type Confusion (*IsAVIconBundleRec6*)	JavaScript
CVE-2018-4990	Use-After-Free (ROP chains)	JavaScript

למידת מכונה בשביל גילוי נוזקות בקבצי PDF - Introduction and Background

למידת המכונה כאן עובדת באופן דומה של למידת מכונה כללית.



יש לנו 3 שלבים בלמידת מכונה בכדי לסווג בסוף האם הקובץ זדוני או לגיטימי.

1. Pre-processing – עיבוד מוקדם ניתוח קבצי PDF על ידי בידוד נתונים חיוניים למשל : ג'אבה סקריפט, מילות מפתח, ActionScript וכו'.
2. Feature extraction – חילוץ פיצ'רים, רכיב זה פועל על המידע שחולץ לנו ע"י תהליך מס' 1, וליצור ממידע זה ווקטור מספרים. ווקטור זה יכול לייצג מילות מפתח, שימוש ב API וכו'.
3. Classifier – שלב הסיווג, לא ניתן את כל המידע רק לאימון אלה גם לבדיקות, וגם בשלב זה נבדוק שאכן אנו לא ב over fitting , ומשם נסווג האם זדוני או לגיטימי.

כעת נסביר כיצד ניתן לאפיין כל מרכיב ומהם החוזקות והחולשות שלו.

Pre-processing – תהליך זה חיוני מאוד לבחירת הנתונים בכדי שנוכל לזהות.
ישנם 2 סוגי תהליכי עיבוד:

1. סטטי – לא נריץ את הקובץ, אלא ננסה לנתח את התוכן שלו, הבנה כמות מספר האובייקטים, קבצי JavaScript אשר בקובץ כמה מקודדים. למשל ע"י pdf-parser, peePdf ניתוח זה משמעותית לוקח פחות זמן ופחות יקר, אך חסרונו הוא עירפול מידע, כלומר יכול להיות שבסופו של דבר לא נדע בוודאות האם הקובץ זדוני או לא.
2. דינאמי – הרצה של הקובץ בתוך סאנד-בוקס וכדומה. החיסרון הוא שזה פעולה יקרה וצורכת הרבה זמן, אך בסופו של דבר תביא לתוצאות חד משמעיות.

כאשר אנו מדברים על כלים ל pre-processing ישנם 2 קטגוריות עיקריות:

1. על ידי גורמי צד שלישי – יתרון: אתה לא צריך להמציא בעצמך את הכלי לניתוח וכבר מקבל פונקציות מובנות. חיסרון: בגלל שזה גינארי יכול להיות שישנם הרבה פונקציות שאתה בכלל לא צריך וסתם מעמיס על המערכת וכמו כן יכולים להיות באגים ובגלל שזה צד שלישי לא נדע עליהם.
2. ניתוח מותאם אישית – ניתוח שאתה עושה בעצמך, יתרון: מותאם אישית לבעיה שלך. חיסרון: לבנות את הכל מאפס וכמו לכל מתכנת גם לך יכול להיות באגים וגם לוקח הרבה זמן לבנות.

טבלה שמראה אלו חברות נותנות לנו שירות של צד שלישי(חלק מהם זה קוד פתוח לא הכל בתשלום). ואפשר לראות גם במה הם מתמחות יותר ובמה פחות עוזרות לגלות.

Parser	PDF Structure	JavaScript	Embedded Files
Origami [52]	Complete	Partial (Code Analysis)	Complete
JSand [23]	None	Complete	Partial (Analysis)
PDFId [86]	Partial (Key Analysis)	None	None
PeePDF [33]	Partial (Obj. Analysis)	Partial (Code Analysis)	Complete
PhoneyPDF [90]	None	Complete	None
Poppler [36]	Complete	Partial (Extraction)	Complete

ישנם 3 כיווני חקירה עיקריים כמו שאפשר לראות בטבלה כמו שאנו יודעים אלו הפרטים בהם בד"כ שותלים את הנוזקה:

- PDF Stracture – חלק זה מתייחס לכל המרכיבים של מבנה קובץ PDF ואינם קשורים ל Embedded Files ,למשל לניתוח של כמה אובייקטים יש לנו כמה מהם מקודדים וכו'.
- JavaScript – מתייחס לקוד הג'אבה סקריפט המוטבע בתוך הקובץ, כלומר כאן אנו מנתחים את הקוד האם הוא זדוני או לא, וכאן כמו מקודם אפשר לעשות זאת בשני אפשריות ניתוח סטטי או דינאמי שהסברנו למעלה.
- Embedded Files – כאן רעיון הניתוח הוא לחלץ קבצים אשר ניסו להחדיר למשל קיבצי exe שניסו להחדיר לקובץ אנו רוצים לנתח אותו. חילוץ של קבצים מוטבעים.

כמו שאפשר להבין מטבלה אין לנו שירות אשר יכול לטפל ב3 כיווני החקירה. לכן כדי להבין מה כל נותן שירות מספק לנו כדי שנדע האם אנו זקוקים לו או לא.

- **Origami** - מנתח זה, שנכתב כולו ב Ruby , מאפשר למשתמשים לנווט באובייקט מבנה של קבצי PDF, ליצירת קבצים זדוניים על ידי הזרקת קוד או אובייקטים אחרים, כדי לשחרר דחיסה ומפענח זרמים וכו'. יתר על כן, הוא מטמיע מידע פופולרי לזיהוי פגיעויות מבוססות JavaScript API.
- **JSand** - מנתח זה היה חלק ממנוע Wepawet לביצוע ניתוח דינמי של קבצי PDF. זה יכול להפעיל את קוד ה-JavaScript המוטבע כדי לחלץ קריאות API ולבטל את הערפול קוד. יתר על כן, זה יכול לבדוק קובצי הפעלה משובצים(Embed) כדי לחשוף את נוכחותם של נוספים התקפות. לרוע המזל, שירות Wepawet אינו זמין כעת; ולכן זה לא אפשרי כדי לבדוק את JSand יותר.
- **PDFId** - מנתח זה פותח כדי לחלץ את אובייקטי שם ה-PDF הוא אינו מבצע ניתוח נוסף על קוד או קבצים מוטבעים.
- **PeePDF** - מנתח זה, שנכתב כולו ב-Python, יכול לבצע ניתוח מלא של מבנה קובץ ה-PDF (ללא יכולת להחדיר אובייקטים). זה מאפשר להזריק ולחלץ קבצים משובצים(Embed), והוא מספק ניתוח סטטי בסיסי של קוד JavaScript.
- **PhoneyPDF-** מנתח זה, שנכתב כולו ב-Python, מבצע ניתוח דינמי של קוד JavaScript מוטבע באמצעות מכשור. ליתר דיוק, זה מחקה את ביצוע של קוד JavaScript המוטבע בקובץ PDF, על מנת לחלץ קריאות API זה קשורים אך ורק לביצוע PDF. מנתח זה אינו מבצע ניתוח מבני כלשהו או הטמעת חילוץ קבצים.
- **Popler** - היא ספריית C++ המשמשת תוכנות פופולריות בקוד פתוח כגון X-Pdf כדי להציג את תוכן הקובץ. מסיבה זו, תכונות הספרייה הושלמו תמיכה בניתוח וניהול מבנה PDF, כמו גם חילוץ קוד JavaScript והזרקת קבצים משובצים(Embed).

Feature extraction

בשלב זה נחלץ מתוך הניתוח של השלב הקודם את התכונות הרלוונטיות לקובץ שלנו וניצור ממנו וקטור שיכנס ללמידת המכונה.
התכונות בוקטור מחולקות ל3 קטגוריות:
- מבניות הקובץ: תכונות הקשורות למבנה קובץ הpdf לרוב נוגעות במילות מפתח ספציפיות בקובץ או אובייקטים\זרמים עקיפים.
- java script : תכונות הקשורות למבנה קוד ומתאפיינות באופרטורים ספציפיים וכמותם
- raw bytes : נוגעות בתכונות ספציפיות הנוגעות לרצפים של בתים שנלקחו כn-grams

Detector	*Tool*	*Year*	Structural	JS-Based	RAW Bytes
Shafiq *et al.* [80]	N-Gram	2008	x	x	N-Grams
Tabish *et al.* [89]	N-Gram	2009	x	x	N-Grams
Cova *et al.* [23]	Wepawet	2010	x	Execution-Based	x
Laskov and Šrndić [53]	PJScan	2011	x	Lexical	x
Maiorca *et al.* [64]	Slayer	2012	Keywords	x	x
Smutz and Stavrou [82]	PDFRate-v1	2012	Metadata	x	x
Šrndić and Laskov [85, 95]	Hidost	2013	Key. Sequence	x	x
Corona *et al.* [22]	Lux0R	2014	x	API-Based	x
Maiorca *et al.* [60, 61]	Slayer NEO	2015	Keywords	API-Based	x
Smutz and Stavrou [83]	PDFRate-v2	2016	Metadata	x	x

כפי שניתן לראות בטבלה לעיל, רוב הפתרונות בנושא התקפות על Pdf מיישמות מתודות חילוץ הכוללות רק סוג אחד של תכונה.
הדבר נעשה בעקבות הרצון להיות יותר אפקטיביים בנושא ספציפי אחד ולא ללכת לאיבוד בחיפוש במתודה שתכלול באופן כולל את כל 3 הסוגים.

Learning and Classification – את הוקטור תכונות שמגיע מתוך חילוץ התכונות בשלב הקודם נכניס למכונה הלומדת וממנה נקבל את הסיווג עבור הקובץ

Related works:

טכנולוגיות הזיהוי מתחלקות ל3 : סטטי, דינאמי, ומשולב (סטטי ודינאמי יחד).
ניתוח סטטי
בשנת 2008 לראשונה זוהו קבצים זדוניים באמצעות שיטת Markov n-grams שהייתה מודל סטטי באמצעות למידת מכונה. הרעיון הכללי שלו היה לאתר קודים זדוניים ולהשוותם אל מול הטכנולוגיה הקודמת שהייתה קיימת.
זמן קצר לאחר מכן בשנת 2009 הוצגה שיטת זיהוי המבוססת על עצי החלטה ובה מיצוי הפיצ'רים של הקובץ היה גם לרמת הבתים והאלגוריתם המשומש שם היה אלגוריתם סטנדרטי של כריית נתונים שניתח כל בלוק בקובץ, הוא התבסס על חוסר חתימה ויכולת לזהות תוכנות זדוניות של zero-day בצורה מדויקת.
לאחר מכן כבר בשנת 2011 פותחה PJscan טכנולוגיית זיהוי למסמכי PDF זדוניים, שהייתה שיטת ניתוח סטטי מבוססת javaScript ויכלה להשיג כבר כ85 אחוזי דיוק בזיהוי. בהשוואה לדגמים דינאמיים טכנולוגיה זו נפלה לא מעט פעמים בזיהוי קבצים תקינים כזדוניים.
במהלך 2012 הוצעה שיטה אשר השתמשה בשיטת זיהוי תבניות כדי לחלץ תכונות ממילות מפתח בקבצי PDF וסיפקה באמצעות אלגוריתם יער אקראי. שיטה זו בדקה כל התקפה על קובץ PDF ולאו דווקא מתקפת js .

באותה שנה הוצגה PDF-rate גלאי PDF זדוני המבוסס על מטא נתונים ותכונות מבנה הקובץ. שיטה זו יכלה להשיג את התוצאות המיטביות ביותר באותה תקופה גם כנגד תכונות זדוניות לא ידועות עם שיעור סיווג גבוה של 99 אחוז דיוק, והייתה יעילה גם כנגד מתקפות חיקוי.
בשנת 2013 נבנה גלאי שהתבסס על נתיב מובנה בקבצי PDF והשתמש גם הוא בעץ החלטות ובנוסף במכשיר וקטור מובנה (SVM) כאלגוריתם הלמידה. למרות שהיכולות שלו כנגד פרצות אבטחה לא ידועות הייתה חזקה, החוסן שלו היה חלש באופן יחסי.
כשנתיים לאחר מכן בשנת 2015 הוצע מודל זיהוי אוטומטי שחילץ את התכונות מהמבנה וגם מהתוכן של הקובץ כדי למנוע התחמקות על בסיס זיהוי מבנה. הדיוק שלו היה טוב יותר ממערכות אחרות המשתמשות בטכנולוגיית ניתוח סטטי, אך היה רגיש מדי למערך הtrain .
בשנת 2016 עוד חידשו את המודל הנ"ל בעזרת סטטוס הצבעה של מסווגי משנה כדי לשפוט האם החיזוי אמין. במהלך אותה שונה עוד הרחיבו את המודל על מגוון שונה של פורמטים.

ניתוח דינאמי

את טכנולוגיות זיהוי אלו יש להפעיל בתוך ארגז חול או מכונה וירטואלית כדי להימנע מחילול המחשב האישי. כאשר התוכנית פועלת הנתונים נבנים על ידי ניתוח כלים דינמיים המאפשרים לקבל ניתוח מדויק יותר, אך זה לוקח הרבה יותר זמן.
בשנת 2010 הציעו גלאי תוכנות זדוניות מקוונות והשתמשו במסווג בייסיאני כדי לזהות קוד JS זדוני.
בשנת 2011 ראו כי מיותר להשתמש בטכנולוגיית סימולציה המבוססת תוכנה והציעו מסגרת חדשה לאיתור מהיר, מדויק של הזרקות קוד.

טכנולוגיות זיהוי משולבות סטטי ודינאמי

מצד אחד טכנולוגיה זו לוקחת את היתרונות של שתי הטכנולוגיות, אך מנגד הופך את המודל למורכב יותר.
בשנת 2011 פיתחו סורק מסמכים זדוני עצמאי המשלב ניתוח מסמכים סטטי וביצוע קוד דינמי כדי לזהות איומי PDF שלא היו ידועים בעבר.
בשנת 2014 פותח מודל זיהוי תוכנות זדוניות של JS קל משקל באמצעות הפניות של API והשיג ביצועים מצוינים במונחים של דיוק, תפוקה והכללה.
באופן כללי בכדי ליצור גלאי בעל ביצועים גבוהים נצטרך מערך נתונים בקנה מידה גדול ועלות של תיוג דגימות אלה עדיין מהווה בעיה.
(בטבלה לעיל ניתן לראות ציר זמן מסודר של הכלים וההתקדמות שלהם בנושא)

התקפות כנגד גלאי תוכנות זדוניות בPDF:

בחלק זה נציג כיצד ההתקפה "מתגוננת מפני ההגנה נגדה". (כלומר כיצד התוקף נמנע מגילוי התקיפה). ובכך נוכל ללמוד עוד גם על דרכיו של התוקף.
<u>מטרת התוקף-</u> מוגדרת על סמך הפרת האבטחה הרצויה לו. (שלמות, זמינות או הפרת פרטיות)
באופן כללי, התוקף ירצה לבלבל את המכונה שלנו בעזרת קבצים שיסווגו להפך ממה שהם- וכך בעצם לגרום לבלבול בזיהוי התכונות וסיווג המכונה הסופי.
דוגמא לכך היא סקריפטים כללים שרצים ברקע, שיוני גופנים, הוספת טקסט וכו')
<u>הידע של התוקף-</u> יכול להשתנות בין תוקף לתוקף ברמת הידע שלו על היעד, כלומר התוקף עשוי לדעת באיזה אלגוריתם הגלאי משתמש (אלגוריתם עיבוד מקדים וחילוץ תכונות) בפוטנציאל התוקף ישאף לדעת גם את הפרמטרים המסווגים שנלמדו לאחר האימון.
רמת הידע של התוקף לגבי המכונה מחולקת ל3 תתי קטגוריות:
-ידע מושלם שנקרא גם white box בתרחיש כזה אנו מניחים כי התוקף יודע הכל על המערכת שלנו למרות שזה ,קורה לעתים רחוקות, מעשית זה אפשרי וכך אפשר להיערך למקרה הקיצוני ביותר.
-ידע בינוני שנקרא גם gray box בתרחיש כזה אנחנו מניחים שלתוקף ידע מוגבל, הוא מכיר חלק מהתכונות אך אין לו את נתוני האימון ופונקציית הסיווג.
-אפס ידע שנקרא גם black box בתרחיש כזה אנו יוצאים מנקודת הנחה כי לתוקף ידע מינימלי, אם בכלל.

<u>היכולת של התוקף-</u> לנהל את נתוני הקלט שיוצאו מהpdf לתוך הוקטור שבהמשך יוכנס למכונה. האם לתוקף יש יכולת להשפיע באופן שונה על קלטים שלו שיוכנסו לtrain או לtest
האם הוא יוכל להשפיע על הקבצים באופן כזה שישבשו את למידת המכונה בזמן שההתקפה שלו תישאר בשלמותה.

Attacker's Capability	Attacker's Goal *Integrity*	*Availability*	*Privacy*
Test data	Evasion	-	Model Extraction/Stealing Model Inversion Membership Inference
Training data	Poisoning (Integrity) Backdoor	Poisoning (Availability)	-

בטבלה לעיל מפורטות התקפות שונות והשפעתן אל מול מטרת התוקף, אנו נפרט כעת על כל אחת מהן:
התקפות התחמקות evasion attacks- בהגדרה זו התוקף מצליח לבלבל את המכונה כך שתסווג קובץ זדוני כקובץ נקי.
התקפות הרעלה poisoning attacks – התקפות אלו מכוונות לשלב האימון של המכונה, התוקף מזריק בכוונה דגימות עם תווית שגויה במטרה להפחית את אחוזי הזיהוי ובכך בקצה יגרום לסיווג כללי לא נכון.
התקפות פרטיות – privacy attacks – מטרת התקפות אלו היא לגנוב מידע על המכונה הלומדת באמצעות שליחת דגימות ספציפיות כנגד המכונה.

בס"ד

Dataset description:

במאמר PDF Malware Detection Based on Optimizable Decision Trees נמצאת הטבלה הנ"ל

Table 1. Summary of reviewed related research.

Ref.	Model	Datasets	Analysis Method	Advantages	Limitation
[22]	SVM, RF	997 virus files and 490 clean files	Hybrid	The high accuracy rate for static, dynamic, and combined techniques.	Very small dataset
[24]	Markov n-gram	37,000 malware and 1800 benign	Static	The Markov n-gram detector offers higher detection and false-positive rates than the other embedded malware detection method currently in use.	An evasion test is not available.
[25]	(J48) classifier	VX Heavens Virus Collection [50]	Static	The proposed model may identify the malware file's family, such as virus, trojan, etc.	An evasion test is not available.
[26]	RF	Contagio [27]	Static	Even though the training set, classification technique, and document features are known, the classifier is resistant to mimicking attacks.	Evasion is much more challenging because classification depends more evenly on many parameters.
[28]	RF, C5.0 Decion Tree (DT), and 2-class SVM	Contagio [27] + VirusTool [51]	Static	It gives us a thorough grasp of how these selected features affect classification, and this will improve the training time.	All datasets provided by VirusTotal are benign, and this will make decisions biased.
[30]	ensemble classifier (random sampling/bagging)	Contagio [27]	Dynamic	Using real data	It does not examine any potential embedded PDF payload.
[32]	Heuristic-based	Contagio [27]	Dynamic	More resistant to code obfuscation	Any API extraction mistakes could compromise the accuracy of the detector.
[10]	Bayesian, SVM, J48, and RF	Contagio [27]	Static	Multi-classifier system	Not efficient with different types of embedded malicious codes in PDF files
[35]	KNN	Generated Dataset	Static	It drastically lowers false negatives and improves detection accuracy by at least 15%.	An evasion test is not available.
[36]	heuristic search, RF, AND DT	Generated Dataset	Static	Identifying advanced persistent threats	It was not tested against evasion techniques and mimicry attacks.
[37]	Naive SVM	Dump [52]	Static	Prevent gradient-descent attacks	Slower than other algorithms
[39]	RF, SVM, and NB	Contagio malware dump between November 2009 and June 2018 [44]	Static	Adequately for malware detection	Not detect adversarial samples
[40]	Convolutional Neural Network (CNN)	VirusTotal	Static	Robustness against evasive samples	Can not detect adversarial samples
[41]	Coding style	HAL dataset [53]	Static	Trust generation process for PDF files	Time-consuming: the complexity depends on the file size.

בטבלה זו מתוארים מודלים שונים של למידת מכונה מתוך המחקרים שהצגנו קודם. כמו כן בכל שורה ניתן לראות פירוט של dataset בו היה שימוש עבור כל מודל.
אנחנו נשתמש בdataset של Contagio שהיה בשימוש בכמה וכמה מחקרים קודמים שנעשו בנושא.

http://contagiodump.blogspot.com/2013/03/16800-clean-and-11960-malicious-files.html

בס"ד

Dataset Exploration:

contagio: Version 4 April 2011 - 11,355+ Malicious documents - archive for signature testing and research (contagiodump.blogspot.com)

גדול המידע: אנחנו נשתמש ב1106 קבצי PDF נקיים ועוד 130 קבצים זדוניים.
אוסף 5 הוא אוסף הקבצים הנקיים המכיל 104 קבצים נקיים ללא סיסמא רק כדי לוודא שהמכונה תזהה אותם כנקיים.
אוסף 4 הוא אוסף קבצים זדוניים שנאספו ברחבי האינטרנט ומכילים קובץ זדוני בתוכם, רובם מכילים קוד זדוני לניצול חולשות משתמש.

בכדי להגדיל את המאגר שלנו ולא להסתמך רק על הdata set המובא במאמר הורדנו עוד 1002 קבצי PDF נקיים אשר יעזרו לנו לאמן את המודל. מאתר https://www.loc.gov/item/2020445568/ כאשר אלו מסמכי ממשלה מגוונים ובכך נוכל ליצור מודל אשר מזהה מגוון רחב יותר של פגיעויות.

אנו משערים שניתוח על מנת לחקור את הקבצים יהיה על ידי ניתוח סטטי ולא דינאמי בכדי להוציא מהם את הפיצ'רים. מכיוון שניתוח דורש המון זמן ויצירה של מכונות וירטואליות ומשאבים נוספים. לכן אנו נעשה קוד הרבה יותר טוב אשר מסתמך על ניתוחים סטטים.

שינוי של ה dataset לפני יציאה לדרך:

רגע לפני שהתחלנו לעבוד על הקוד שלנו, ורגע אחרי שסיימנו לקרוא עוד כמה מחקרים בנושא החלטנו על כיוון שונה של dataset מעודכן יותר, גמיש יותר, גדול יותר.
כך נוכל לשחק עם הדאטה עד שנגיע לתוצאות רצויות וטובות יותר ונקבל מגוון רחב יותר של קבצים.
בסך הכל נשתמש בכ10,000 קבצים מתוכו כאשר החלוקה לנקי וזדוני היא ביחס של 60:40, כאשר יש יותר קבצים נקיים.
את הדאטה סט ניתן למצוא בקישור הנ"ל:
http://205.174.165.80/CICDataset/CICEvasivePDFMal2022/Dataset/PDFs/

ההגעה אליו הייתה דרך המאמר של PDF Malware Detection Based on Optimizable Decision Trees .

לסיכום:

סקרנו את הבעיה אותה נרצה לחקור- קבצי PDF זדוניים.
הצגנו באופן כללי את מבנה של קובץ pdf
לאחר מכן פירטנו על חלקיו ובאילו מהם ניתן להחדיר נוסקה לקובץ.
כמו כן הצגנו את הנושא של למידת מכונה בהקשר זיהוי קבצי pdf זדוניים והכלים הקיימים בשוק כדי ליצור מכונה שכזו, לצד מחקרים ועבודות קודמות שנעשו בנושא.
לבסוף הצגנו את הצד התוקף, את מטרותיו וכליו כדי להימנע מהמכונה הלומדת.
כעת נפנה לפתרון הבעיה בעצמנו, ניעזר במחקרים הקודמים שנעשו ונצטייד במסקנות שלהן נשתמש בדאטה שהצגנו לעיל לטובת המכונה שנבנה, ננסה להתגבר על הגישה שאומרת שניתן לבדוק רק סוג תקיפה אחת בכל פעם, ואנחנו נלך "על כל הקופה" כלומר נבדוק במקביל את 3 סוגי התקיפה על הקבצים.