מערכת אינטליגנטית לגילוי וירוסים באנדרואיד.

בימינו, השימוש בפלאפונים חכמים פופולרי מאד, בעיקר בזכות הניידות שלהם בשילוב עם השימושיות, כלומר היכולת של המשתמש לבצע מספר רב של משימות באמצעותם. אנדרואיד היא מערכת ההפעלה הכי פופולרית בקרב סמארטפונים (81% ע"פ מחקר של IDC) בזכות נתח השוק הרחב שלה והעובדה שהיא מערכת קוד פתוח. למעשה, לאנדרואיד יש קהילה גדולה שלמפתחים אשר כותבים אפליקציות המגדילות את הפונקציונליות של המכשירים והן נכתבות בדרך כלל בשפת ג'אוה. את האפליקציות ניתן להוריד דרך Google Play או אתרי צד שלישי.

עם זאת, אנדרואיד מושכת אליה לא רק מפתחים "כשרים" אלא גם מתקיפים המפתחים אפליקציות המכילות תוכנות זדון או נוזקה (מלוור - Malware, הלחם בסיסים של Malicious Software). קיימות כבר כמה דוגמאות לאפליקציות זדוניות שכאלה(למשל Geinimi, DriodKungfu and Lotoor) שמהווים מגוון איומים על משתמשים כגון גניבת זהויות, סיסמאות ופרטים חסויים או נעילת\הצפנת המכשיר על מנת לבקש כופר(ransomware). למעשה, דווח שאחת מכל חמש אפליקציות הינה מלוור. לכן ישנו צורך הולך וגובר בטכניקות הגנה חסינות ויעילות שיגנו על המשתמשים מפני איומים חדשניים אלו.

במאמר זה, בניגוד לעבודות קיימות (כנראה), הכותבים מעוניינים לא רק לבחון את ה"קריאות למערכת הAPI" (API calls -קריאות למערכת ההפעלה ובעצם, הדרך של אפליקציות להשתמש בפונקציונליות של מערכת האנדרואיד ולגשת למשאבי המערכת כגון זיכרון וכו') אלא גם לבדוק לעומק את ההקשרים ביניהן, בין אם הן לקוחות מאותו קטע קוד או מאותה חבילה או משתמשות באותה שיטת הפעלה. על מנת לאפיין הקשרים בין קריאות מערכת מציגים הכותבים מבנה של "רשת מידע הטרוגנית"(מחקר קודם) על מנת לתאר אפליקציות ואת קריאות הAPI שלהן. לאחר מכן הם משתמשים בעוד עבודה קודמת הנקראת meta-path כדי לקבוע סמנטיקה של קשרים בי אפליקציות. לבסוף הם משתמשים באלגוריתם מולטי קרנל כדי למשקל meta-path-ים שונים כך שנבחן רק את אלו הנחוצים לנו עם ההקשר הנכון. כתוצר סופי הם פיתחו את ה – HinDroid שאמור לקבוע האם אפליקציה מסוימת מסווגת כמלוור. הפתרון נבדק בניסוי שנערך בחברת Comodo Cloud Security Center ע"י כלי ייעודי שלהם לאבטחת מידע במכשירי מובייל.

מערכת ה-הינדרואיד מורכבת מ5 שלבים עיקריים:

1- Unzipper and Decompiler – קודם כל עושים אנזיפ לקובץ הAPK (Android Application Package) כדי לקבל קובץ דקס (DEX.) שהוא למעשה קובץ הרצה של תוכנית אנדרואיד מקומפלת (Dalvik Executable) הניתנת לתרגום ע"י DalvikVM. לאחר מכן עושים דקומפייל לקובץ זה ומקבלים קוד סמאלי (Smali). סמאלי הוא אסמבלר\דיסאסמבלר המספק לנו קבצים ניתנים לקריאה בשפת סמאלי, שהיא שפה אמצעית בין ג'אוה לדאלוויק.

2- Feature Extractor – כעת שיש בידינו קובץ סמאלי קריא, נוכל להתבונן באילו קריאות מערכת משתמש הקוד הזדוני וכמובן לאפיין את הקשר.

למה יש חשיבות לקריאות המערכת ולקשרים ביניהן?

כפי שציינו, קריאות מערכת הן הדרך של אפליקציות לקבל שירות ממערכת ההפעלה, ניתן להציג ולאפיין התנהגות של אפליקציה בדרך שבה היא משתמשת בקריאות המערכת. הקשר בין אפליקציה לקריאות המערכת שבה "R0" יסומן באופן הבא: מטריצה A אשר כל אלמנט בה $A_{ij} \in \{0, 1\}$ מציין האם אפליקציה i מכילה את קריאת המערכת j .

באפליקצייה זדונית מסוג כופר (Locker.apk) ניתן לראות בקוד הסמאלי 3 קריאות מערכת באותה פונקציה - "Ljava/io/FileOutputStream → write", "Ljava/io/IOException → printStackTrace", and "Ljava/lang /System → load"

בעוד ששימוש בכל אחת מהן בנפרד הוא הגיוני לגמרי, השילוב של שלושתן ביחד מנסה בעצם לרשום תוכן כלשהוא לליבת מערכת ההפעלה, דבר נדיר לחלוטין באפליקציות לג'יטיות, בכך הקשר

ששלושת הקריאות האלה מקיימות, הימצאות משותפת באותה הפונקציה, הינו מידע חיוני עבור מערכת לגילוי רנסמוור. ע"מ לאפיין קשר כזה במערכת ה HIN, קשר מסוג זה, "R1" יאופיין ע"י מטריצה B כאשר כל אלמנט בה Bij ∈ {0, 1} מציין האם הקריאות i ו j נמצאות באותה מתודה.

סוג היחס הבא שנאפיין הוא שייכות לאותו פקייג' (Package) , בהתבסס על תצפיות ונתונים קיימים, קשר כזה מעיד בדר"כ על קשר חזק בין קריאות אלו. ע"מ לאפיין קשר כזה "R2" נייצר מטריצה P אשר כל אלמנט בה Pij ∈ {0, 1} מציין שהקריאות i ו j שייכות לאותו פקייג'.

בקוד הסמאלי, ישנן חמש סוגי מתודות להוציא לפועל (invoke) קריאות מערכת והן: invoke-static, invoke-virtual, invoke-direct, invoke-super, and invoke-interface. – שימוש באותה שיטה עבור כל זוג קריאות יכול להעיד על קשר ביניהן ועל כן נרצה להגדיר גם יחס כזה. ע"מ לאפיין זאת "R3" נייצר מטריצה I אשר כל אלמנט בה Iij ∈ {0, 1} מציין שימוש באותה הוצאה לפועל עבור הקריאות i ו j .

3- HIN Constructor – בניית גרף ה HIN עבור האפליקציה המנותחת כרגע, ע"פ מטריצות ההקשרים. הקודקודים בגרף מייצגים את האפליקציות וקריאות הAPI ואילו הצלעות מייצגות את ארבעת סוגיי היחסים. בשלב זה השתמשו החוקרים ב Meta-path [Sun et al., 2011] (מחקר קודם) ע"מ לקבוע אילו יחסים מסדר גבוה יותר. המסלולים שנרצה לחקור בגרף הם אלו שמחברים בין אפליקציות כדי להשתמש בלמידת מכונה, ע"י זה שאפליקציות מוגדרות דומות מבחינת המחשב כאשר הן מכילות אותו סוג הקשרים בין קריאות הAPI.

4- Multi-kernel Learner – בהתבסס על מטריצות היחסים, ועל גרף הHIN מהשלבים הקודמים, מייצרים מעין ליבה למערכת הנותנת משקל שונה לכל סוג קשר(meta-path) ובכך ללמוד לסווג סוגי אפליקציות.

5- Malware Detector – סיווג האפליקציה כזדונית או לא ע"פ ניתוחי המערכת.

החוקרים בדקו את מערכת ה HinDroid החדשה באמצעות ניסוי כאשר תוך כדי, meta-path-ים שנכשלו בלגלות אפליקציות זדוניות במשך כמה פעמים, ממושקלות נמוך יותר ויתר ע"י המולטי קרנל. הניסוי הוכיח שמערכת ה HinDroid אכן מאד אפקטיבית והיא כבר משולבת כיום בכלי הסריקה של חברת קומודו לאבטחת מידע במובייל. כמובן שמידי פעם מתעדכנת המערכת בעקבות הגילויים של סוגי אפליקציות זדוניות חדשות.

כמו שאני רואה את הדברים, תחום אבטחת המידע הוא כמו שדה קרב שמתפתח ונעשה מתוחכם יותר, וגולש לכל מוצר חדש או מערכת שיוצאים לשוק, לכן העבודות העתידיות שעוד יעשו בתוך תחום זה יתפתחו למלא כיוונים שונים. חשוב לזכור שהמחקר הנוכחי מדבר רק על גילוי מוקדם ולא דובר בו על נתינת מענה או גילוי מאוחר וכו'.