دوفصل‌نامه علمی منادی امنیت فضای تولید و تبادل اطلاعات (افتا)
سال ۱۴۰۰، جلد ۱۹، شماره ۲ (۵۱-۶۰)
https://monadi.isc.org.ir

ارائه‌شده در هفدهمین کنفرانس بین المللی انجمن رمز ایران در دانشگاه علم و صنعت ایران، ۱۹ و ۲۰ شهریور ۱۳۹۹

تشخیص حمله وکیل سرگردان در برنامه‌های اندرویدی با استفاده از رویکرد مهندسی معکوس مدل‌رانده★

مینا حیدری*، عاطفه نیرومند، بهروز ترک لادانی و بهمن زمانی

دانشکده مهندسی کامپیوتر، دانشگاه اصفهان، اصفهان، ایران

اطلاعات مقاله

کلمات کلیدی:
برنامه‌های اندروید
ساز و کار مجوزدهی
مهندسی معکوس مدل‌رانده
حملات افزایش امتیاز
وکیل سرگردان

dor: 10.0000/000000000

نوع مقاله: پژوهشی

چکیده

بسیاری از برنامه‌های اندرویدی دارای اطلاعات یا وظائف حساس و مهمی هستند. این برنامه‌ها اغلب از چندین مؤلفه تشکیل می‌شوند که می‌توانند از طریق ساز و کار اینتنت در دو سطح درون‌برنامه‌ای و یا بین‌برنامه‌ای با یکدیگر در تعامل باشند. اندروید به منظور برقراری امنیت در این تعاملات، ساز و کارهای امنیتی مختلفی را از جمله ساز و کار مجوزدهی ارائه داده است. با این حال، یک برنامه با کسب مجوزهایی که خود از داشتن آن‌ها محروم است، می‌تواند اعمالی را فراتر از امتیازات خود انجام دهد که به آن حمله افزایش امتیاز گفته می‌شود. در این مقاله ابزار VAnDroid که قبلاً بر اساس رویکرد مهندسی نرم افزار نرم‌افزار مدل‌رانده توسط مؤلفین مقاله ارائه شده، تکمیل می‌شود تا مجموعه مهمی از حملات افزایش امتیاز تحت عنوان «وکیل سرگردان» را نیز در هر دو سطح درون‌برنامه‌ای و بین‌برنامه‌ای شناسایی کند. برای ارزیابی ابزار جدید که در قالب یک افزونه جدید اکلیپس به‌نام VAnDroid+ پیاده‌سازی شده است، ده برنامه اندرویدی با بالاترین رواج در بین کاربران ایرانی تحت ابزار ارائه شده در شرایط مختلف تحلیل شدند. نتایج حاصل، نشان می‌دهد که VAnDroid+ می‌تواند به صورت صحیحی آسیب‌پذیری از نوع وکیل سرگردان را در برنامه‌های اندرویدی کشف کند.

۱ مقدمه

امروزه، سکوی اندروید [1] سهم چشم‌گیری از بازار گوشی‌های هوشمند را به خود اختصاص داده است [۱]. همچنین، استفاده‌ی گسترده از برنامه‌های اندروید به منظور انجام وظایف مختلفی مانند ذخیره‌ی داده‌های خصوصی، آن‌ها را به هدفی جذاب برای مهاجمان تبدیل کرده است [۲]. مطالعات متعدد نشان داده‌اند که حملات ناشی از سوء استفاده از مجوزها، از مهم‌ترین مسائل امنیتی در این برنامه‌ها هستند. سکوی اندروید، یک ساز و کار امنیتی تحت عنوان سیستم مجوزدهی را برای محافظت از برنامه‌ها ارائه داده است [۳]. این ساز و کار، مانع از دسترسی برنامه‌ها به منابع حساس سیستم و سایر برنامه‌های محافظت‌شده می‌شود. با این حال، پیاده‌سازی نادرست این ساز و کار در برنامه‌ها و امکان سوء استفاده از آن، زمینه‌ساز انجام انواع مختلفی از حملات است. این حملات، داده‌های حساس موجود در دستگاه کاربر را به مخاطره می‌اندازند و می‌توانند منجر به نشت اطلاعات حریم خصوصی کاربر شوند [۴]. در نتیجه، نیازمند به وجود تکنیک‌های خودکار هستیم که به تحلیل برنامه‌ها و تشخیص مسائل مهم امنیتی از جمله آسیب‌پذیری‌ها در آن‌ها بپردازند [۵].

مهندسی معکوس روشی اساسی برای ارزیابی کیفی از جمله ارزیابی

★از کمیته علمی هفدهمین کنفرانس بین‌المللی انجمن رمز ایران برای داوری این مقاله تشکر می‌شود.
*نویسنده مسئول
آدرس‌های رایانامه: mheidari@eng.ui.ac.ir (مینا حیدری)، atefehnirumand@eng.ui.ac.ir (عاطفه نیرومند)، ladani@eng.ui.ac.ir (بهروز ترک لادانی)، zamani@eng.ui.ac.ir (بهمن زمانی)

[1] android platform

امنیتی سیستم‌های نرم‌افزاری است. اما مهندسی معکوس به تنهایی یک فرآیند وقت‌گیر و مستعد خطا است. از سوی دیگر، مهندسی مدل‌رانده روشی جدید و نویدبخش در این عرصه است که مسائل مربوط به مهندسی معکوس سنتی را می‌توان با کمک آن برطرف کرد و به ایجاد نمایش‌هایی مبتنی بر مدل از جنبه‌های مختلف سیستم‌های نرم‌افزاری موجود پرداخت که به راحتی قابل درک و فهم باشند [۶]. این ترکیب از مهندسی معکوس و مهندسی مدل‌رانده، تحت عنوان مهندسی معکوس مدل‌رانده[1] شناخته می‌شود. بنابراین، در این پژوهش از رویکردی ایستا مبتنی بر مهندسی معکوس مدل‌رانده به منظور کشف آسیب‌پذیری از نوع حملات افزایش امتیاز[2] استفاده شده است و با اضافه کردن قابلیت تشخیص آسیب‌پذیری وکیل سرگردان[3] به ابزار VanDdroid [۳]، ابزار VAnDroid+ ارائه شده است. توسعه‌دهندگان و همچنین کاربران نهایی برنامه‌های اندروید می‌توانند با استفاده از این ابزار، برنامه‌های خود را به منظور کشف آسیب‌پذیری‌های امنیتی مورد بررسی قرار دهند. به طور خلاصه، دستاوردهای این پژوهش: ۱) توسعه‌ی چارچوب ارائه‌شده در ابزار VAnDroid به منظور تشخیص نوعی از آسیب‌پذیری افزایش امتیاز به‌نام وکیل سرگردان ۲) توسعه فرامدل ارائه شده در VAnDroid با هدف افزایش قابلیت تحلیل و کشف حملات وکیل سرگردان در دو سطح درون‌برنامه‌ای و بین‌برنامه‌ای ۳) تکمیل ابزار VAnDroid به منظور افزایش قابلیت تشخیص حمله‌ی وکیل سرگردان در برنامه‌های اندروید

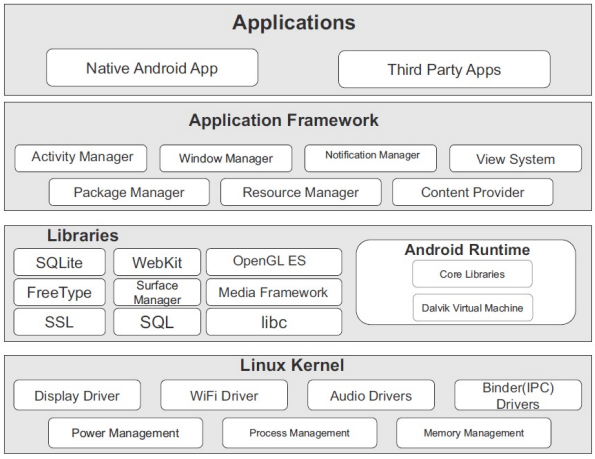
در ادامه‌ی مقاله، ابتدا مفاهیم اولیه و پیش‌زمینه در بخش ۲ بیان می‌شوند. بخش ۳ مروری بر کارهای مرتبط را ارائه می‌دهد. رویکرد پیشنهادی و مراحل اصلی آن در بخش ۴ شرح داده می‌شوند. در بخش ۵ ارزیابی رویکرد پیشنهادی بیان می‌شود. در نهایت، در بخش ۶ به نتیجه‌گیری و بیان کارهای آینده پرداخته می‌شود.

۲ پیش‌زمینه

به منظور درک بهتر مباحث مطرح شده در این مقاله، در این بخش به معرفی مختصر مفاهیم مورد نیاز خواهیم پرداخت.

اندروید را می‌توان به عنوان یک پشته نرم‌افزاری در نظر گرفت. همان‌طورکه در شکل ۱ مشاهده می‌شود، اندروید دارای لایه‌های مختلفی از جمله لایه‌ی هسته‌ی لینوکس، لایه‌ی کتابخانه‌ها، لایه‌ی چارچوب کاربرد[4] و لایه‌ی برنامه می‌باشد که برنامه‌ها در لایه‌ی برنامه نصب می‌شوند [۷]. یک برنامه‌ی اندروید در فایلی با پسوند APK توزیع می‌شود و از چهار نوع مؤلفه فعالیت[5]، خدمت[6]، دریافت‌کننده همه‌پخشی[7] و ارائه‌دهنده محتوا[8] تشکیل شده است. مؤلفه‌های فعالیت، خدمت و دریافت‌کننده همه‌پخشی را می‌توان از طریق اینتنت با استفاده از متدهای خاصی فعال کرد. اگر از طریق اینتنت، اتصال بین مؤلفه‌های یک برنامه ایجاد شود، آن را ارتباط درون‌برنامه‌ای و در صورتی که اتصال بین مؤلفه‌های دو

[1]Model-Driven Reverse Engineering (MDRE) [2]privilege escalation attacks [3]confused deputy [4]application framework layer [5]activity [6]service [7]broadcat reciver [8]content provider



شکل ۱. معماری سیستم‌عامل اندروید [۹]

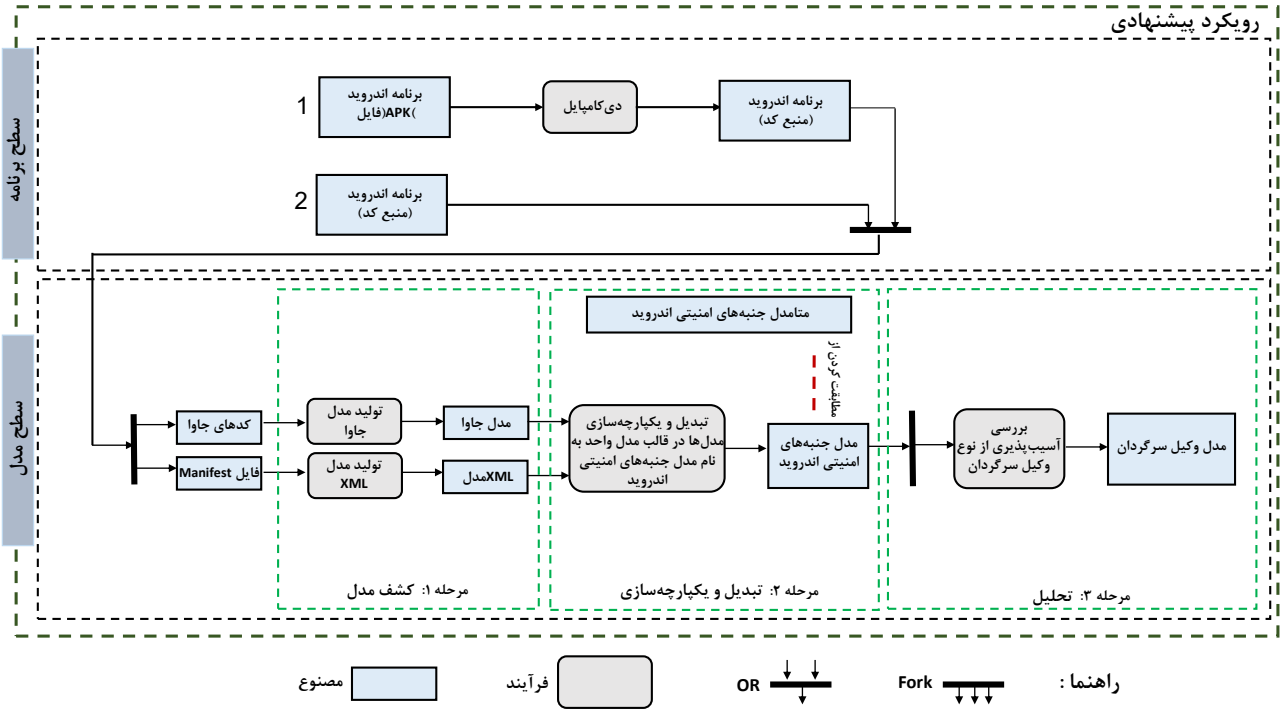
برنامه‌ی مجزا ایجاد شود، آن را ارتباط بین برنامه‌ای می‌نامند [۸].

مجوزها در اندروید، سنگ بنای اصلی مدل امنیتی اندروید هستند که توسط چارچوب اندروید برای حفاظت از برنامه‌ها ارائه شده‌اند [۱۰]. مجوزهای مورد نیاز باید در فایل مانیفست برنامه مشخص شوند تا دسترسی ایمن به منابع حساس و همچنین تعامل برنامه‌ها را ممکن سازند. مجوزهای اندروید به چهار سطح محافظتی عادی، خطرناک، Signature و SignatureOrSystem تقسیم می‌شوند [۱۱]. از نسخه‌ی ۶ اندروید، گوگل سیستم مدیریت مجوز اندروید را از حالت ایستا به پویا تغییر داده است. قبل از نسخه‌ی ۶، کاربر مجبور است که قبل از نصب، تمام مجوزهای درخواستی برنامه را قبول یا رد کند. اگر کاربر از قبول مجوزهای درخواستی خودداری کند، نصب برنامه ادامه نخواهد داشت. اما به دلیل تغییر صورت گرفته، از نسخه‌ی ۶ به بعد، کاربران می‌توانند در زمان اجرا نیز به اعطاء یا ابطال مجوزها بپردازند [۱۲].

طبق تعریف بیان شده توسط داوی[9] و همکاران [۱۳] حمله‌ی افزایش امتیاز زمانی رخ می‌دهد که یک برنامه با مجوزهای کمتر، از دسترسی به مؤلفه‌های یک برنامه با مجوزهای بیشتر محدود نشود. این نوع حمله را می‌توان براساس ساز و کار مورد استفاده به دو دسته‌ی حملات وکیل سرگردان و حملات از طریق تبانی برنامه‌ها[10] تقسیم کرد. در حمله‌ی وکیل سرگردان، برنامه‌ی مهاجم از آسیب‌پذیری‌های موجود در برنامه‌ی هدف، به منظور انجام عملیات غیرمجاز استفاده می‌کند. در حمله‌ی تبانی برنامه‌ها، هر دو برنامه‌ی اندروید درگیر در حمله مخرب هستند و برای به‌دست آوردن مجوزی تبانی می‌کنند که برای هر دو غیر مجاز می‌باشد [۱۴].

استفاده از اصول و تکنیک‌های مهندسی مدل‌رانده در مهندسی معکوس باعث ساخت راه‌حل‌های مؤثرتری می‌شود که به فهم بهتر سیستم‌های نرم‌افزاری کمک می‌کند. راه‌حل‌های مهندسی معکوس مدل‌رانده مقیاس‌پذیر، سازگار، قابل حمل و قابل استفاده‌ی مجدد هستند. این راه‌حل‌ها مستقیماً از مهارت‌های عمومیت، توسعه‌پذیری، ادغام، پوشش و فناوری‌های مهندسی مدل‌رانده بهره می‌برند [۱۵]. یک

[9]Davi [10]colluding applications attack



شکل ۲. رویکرد پیشنهادی +VAnDroid (مبتنی بر رویکرد VAnDroid [۳])

فرآیند مهندسی معکوس مدل‌رانده شامل سه فاز اصلی کشف مدل [1]، فهم مدل [2] و تولید دوباره‌ی مدل [3] است [۱۶].

۳ کارهای مرتبط

تاکنون تحقیقات گسترده‌ای در زمینه‌ی تحلیل امنیتی برنامه‌های اندروید صورت گرفته‌اند. در این بخش، ابتدا تحقیقاتی مرور می‌شوند که به شناسایی آسیب‌پذیری افزایش امتیاز در سطح درون برنامه‌ای می‌پردازند. سپس تحقیقات مربوط به شناسایی آسیب‌پذیری افزایش امتیاز در سطح بین برنامه‌ای معرفی می‌شوند.

تحقیقات مربوط به سطح درون‌برنامه‌ای SAndroid [۱۷] رویکردی قابل گسترش و مبتنی بر امضا است که امنیت در برابر حمله‌ی افزایش امتیاز را از طریق تشخیص سریع فراهم می‌کند. این رویکرد شامل دو بخش اصلی نظارت فعال و تشخیص است که در بخش تشخیص از لیست امضاهای برنامه‌های مخرب شناخته شده استفاده می‌شود. DroidChecker [۱۸] یک ابزار تحلیل خودکار به منظور کشف آسیب‌پذیری در برابر حمله‌ی افزایش امتیاز و به طور خاص حمله‌ی وکیل سرگردان است. این ابزار با استفاده از گراف کنترل جریان درون‌پردازشی و وارسی آلودگی [4] ایستا، مسیرهای نشت داده‌ای را در برنامه تشخیص می‌دهد که منجر به نشت داده می‌شوند. DroidAlarm [۱۹] ابزاری است که مؤلفه‌های آسیب‌پذیر در برابر حمله‌ی وکیل سرگردان را شناسایی می‌کند. در این ابزار هر مؤلفه‌ی حساس شامل واسط عمومی، به عنوان یک مؤلفه‌ی آسیب‌پذیر شناسایی می‌شود. در نهایت ابزار با ساخت گراف فراخوانی برنامه و همچنین گراف کنترل جریان [5]، مسیرهای نشت ناشی از حمله‌ی افزایش امتیاز را شناسایی می‌کند.

تحقیقات مربوط به سطح بین برنامه‌ای. ابزار Covert [۲۰] مجموعه‌ای از برنامه‌ها را به عنوان ورودی دریافت می‌کند و فهرستی از آسیب‌پذیری‌های کشف‌شده را به عنوان خروجی ارائه می‌دهد. این ابزار در دو لایه‌ی front-end و back-end پیاده‌سازی شده است. لایه‌ی front-end، یک محیط تعاملی به منظور استفاده کاربران نهایی ارائه می‌دهد. مؤلفه‌های اصلی covert برای کشف آسیب‌پذیری‌های امنیتی در لایه‌ی back-end پیاده‌سازی شده‌اند.

ابزار DIALDroid [۲۱] به منظور تشخیص حملات افزایش امتیاز ناشی از ارتباطات بین مؤلفه‌ها ارائه شده است. ویژگی کلیدی این ابزار، دقت و مقیاس‌پذیری آن است و از کارآمدترین ابزارهای موجود برای شناسایی آسیب‌پذیری‌های بین برنامه‌ای می‌باشد. DIALDroid در چهار مرحله، سعی در کشف آسیب‌پذیری‌ها دارد. در اولین مرحله، مجوزها و مشخصه‌های Intent Filter از فایل مانیفست استخراج می‌شوند. همچنین اینتنت‌ها و فرستنده و گیرنده‌ی آن‌ها نیز شناسایی می‌شوند. در مرحله دوم، تحلیل ایستا به منظور تعیین مسیر از منابع تولیدکننده‌ی داده‌های حساس به منابع ارسال‌کننده‌ی این داده‌ها انجام می‌شود. سپس از پایگاه داده‌ی رابطه‌ای به دلیل مقیاس‌پذیری و کارآمد بودن آن، برای ذخیره‌ی اطلاعات استفاده می‌شود. در نهایت، با استفاده از پرس‌وجوهایی بر روی پایگاه داده، حملات شناسایی می‌شوند.

[1] model discovery [2] model understanding [3] model (re)generation [4] taint analyses

[5] Control Flow Graph (CFG)

۴ رویکرد پیشنهادی

در این پژوهش رویکرد قبلی ارائه‌شده توسط مؤلفین در ابزار VAnDroid توسعه یافته است. همان‌طور که در شکل ۲ مشاهده می‌شود، این رویکرد دارای دو سطح، یعنی سطح برنامه و سطح مدل است.

ابتدا در سطح برنامه، منبع کد برنامه‌ی اندروید حاصل می‌شود. در این قسمت ممکن است برنامه به دو صورت در اختیار باشد: (۱) فایل APK، که در این صورت بسته‌ی APK دی‌کامپایل می‌شود. (۲) منبع کد برنامه در اختیار باشد که در این صورت نیازی به دی‌کامپایل کردن برنامه نیست. ابزارهای مختلفی به منظور دی‌کامپایل کردن برنامه موجود هستند که در رویکرد پیشنهادی از ابزار Jadx [۲۲] استفاده شده است. سطح دوم شامل سه مرحله‌ی کشف مدل، تبدیل و یکپارچه‌سازی و تحلیل است.

در مرحله‌ی کشف مدل، مدل‌های متناظر با کد منبع به دست می‌آیند. برای به دست آوردن این مدل‌ها، از کشف‌کننده‌های ابزار مدیسکو [۱۵] استفاده می‌شود. از آن جایی که یکی از مهم‌ترین فایل‌های پیکربندی هر برنامه، فایل مانیفست می‌باشد، بنابراین، ابتدا توسط کشف‌کننده‌ی مدل XML، مدل XML از روی این فایل حاصل می‌شود. همچنین با استفاده از کشف‌کننده‌ی جاوا، مدل جاوا از روی منابع کد جاوای برنامه استخراج می‌گردد. از آنجایی که یکی از سیاست‌های اصلی مهندسی معکوس مدل‌رانده ایجاد هر چه سریع مدل‌های اولیه از روی مصنوعات سیستم نرم‌افزاری بدون از دست رفتن کوچک‌ترین اطلاعات است [۲۳]، بنابراین، می‌توان اطمینان داشت که هیچ اطلاعاتی از برنامه از دست نخواهد رفت.

سپس در مرحله‌ی تبدیل و یکپارچه‌سازی، مدل‌های خام اولیه به دست آمده در مرحله‌ی قبل، از طریق تبدیلات مدل به مدل ATL [۲۴] به یک مدل امنیتی تبدیل می‌شوند که از سطح انتزاع مناسبی برای درک برخوردار است. به منظور استخراج اطلاعات مورد نیاز و یکپارچه‌سازی آن‌ها در قالب این مدل واحد، از فرامدل طراحی شده در ابزار VAnDroid استفاده شده است. به منظور تکمیل این فرامدل با هدف تشخیص آسیب‌پذیری وکیل سرگردان، ویژگی‌هایی به این فرامدل اضافه شده است که در بخش ۱.۴ بررسی شده‌اند.

در نهایت، در مرحله‌ی تحلیل، با گردآوری اطلاعات در قالب مدل جنبه‌های امنیتی اندروید و با استفاده از زبان تبدیل ATL و قوانین OCL، این مدل برای تشخیص آسیب‌پذیری حمله‌ی وکیل سرگردان در دو سطح درون‌برنامه‌ای و بین‌برنامه‌ای تحلیل می‌شود. فرآیند تحلیل برنامه به منظور کشف این آسیب‌پذیری در هر دو سطح به ترتیب در بخش‌های ۲.۴ و ۳.۴ توضیح داده شده است.

۱.۴ توسعه‌ی فرامدل VAnDroid با هدف تشخیص آسیب‌پذیری وکیل سرگردان

در این بخش ویژگی‌های اضافه شده در فرامدل طراحی شده در ابزار VAnDroid بررسی می‌شوند که در شکل ۳ این ویژگی‌ها با رنگ قرمز نشان داده شده‌اند.

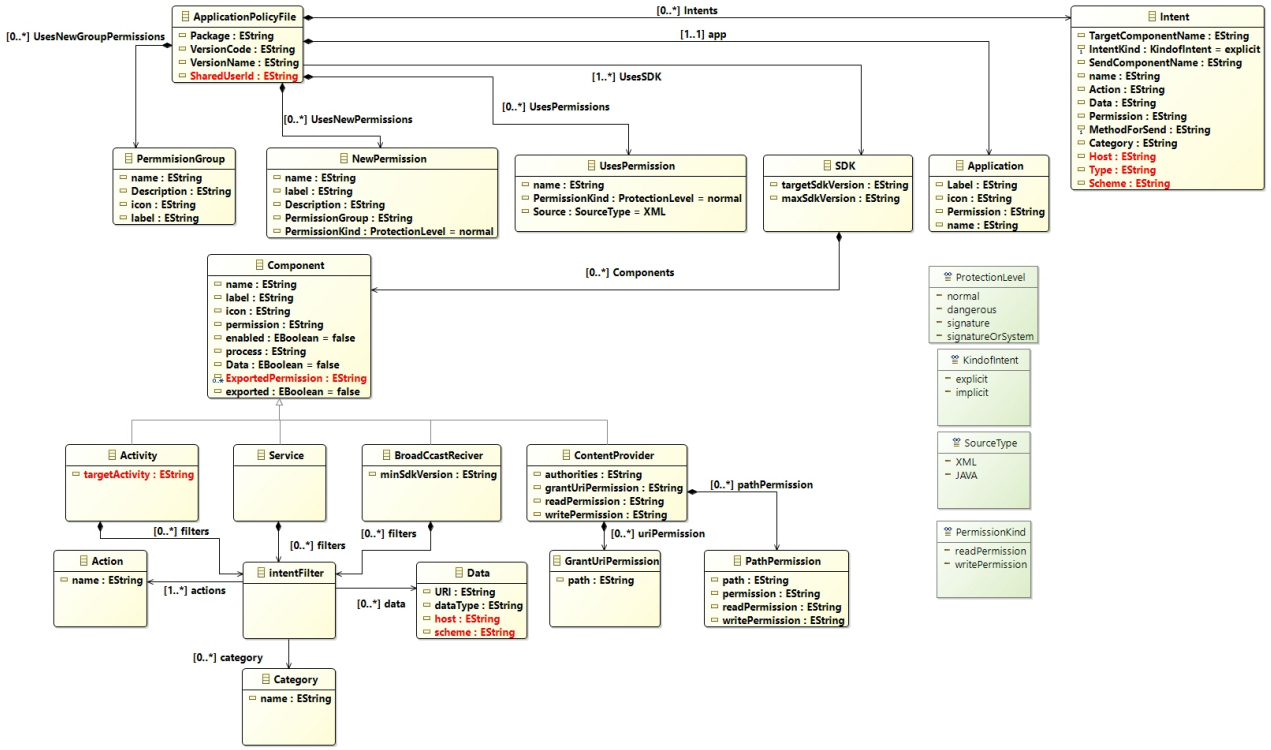
عنصر Activity. علاوه بر چهار نوع مؤلفه‌ای که در فرامدل VAnDroid در نظر گرفته شده است. مؤلفه‌ی دیگری نیز با عنوان فعالیت مستعار وجود دارد. یک مؤلفه‌ی فعالیت را می‌توان در فایل مانیفست با استفاده از مؤلفه‌ی فعالیت مستعار، چندین بار اعلان کرد که به عنوان مؤلفه‌ای جداگانه با مجوزها و Intent Filterهای مخصوص به خود، به برنامه معرفی خواهد کرد. این نوع مؤلفه به فرامدل اضافه شده است.

عنصر UsesPermission. این عنصر در فرامدل VAnDroid تنها بیانگر مفهوم تگ <uses-permission> بوده است. به منظور داشتن دید کاملی از مجوزهای درخواستی، مجوزهای درخواستی توسط تگ <uses-permission-SDK-23> نیز در نظر گرفته شده است. بنابراین، عنصر UsesPermission در فرامدل موجود بیانگر مفهوم دو تگ <uses-permission> و <uses-permission-SDK-23> می‌باشد.

عنصر Intent. در فرآیند مقایسه‌ی اینتنت ارسالی با IntentFilterهای سایر مؤلفه‌های موجود در سیستم، سه ویژگی اصلی Action، Category و Data بررسی می‌شوند. ویژگی Data در ساز و کار اینتنت و فرآیند مقایسه به بخش‌های کوچکتری چون Host، Schema و Type تقسیم می‌شود. بنابراین، به بیان دقیق‌تر، ویژگی‌های Action، Category، Host، Schema و Type مقایسه می‌شوند. در فرامدل VAnDroid تنها ویژگی Action در نظر گرفته شده است. بنابراین بایستی به منظور کشف روابط بین مؤلفه‌ها، سایر ویژگی‌های شرکت‌کننده در فرآیند مقایسه به عنصر Intent اضافه شوند.

عنصر Data. با توجه به اینکه در عنصر Intent ویژگی Data به سه ویژگی Type، Host و Schema تجزیه شده است، در نظر گرفتن این سه ویژگی به منظور انجام مقایسه و یافتن مؤلفه مقصد الزامی است. در فرامدل VAnDroid تنها ویژگی Type در نظر گرفته شده است. بنابراین بایستی دو ویژگی Schema و Host به عنصر Data اضافه شوند.

عنصر Component. تمامی مؤلفه‌های موجود در برنامه، مجوزهای اعطاء شده به برنامه را ارث می‌برند، اما در عمل هر مؤلفه بر اساس کارکرد خود بخشی از مجوزهای اعطا شده را استفاده می‌کند. در فرآیند تشخیص حمله‌ی افزایش امتیاز، در بررسی مجوزهای مؤلفه‌ی مقصد، تنها مجوزهای مورد استفاده آن مؤلفه با مجوزهای مؤلفه‌ی مبدأ بایستی مقایسه شوند. بنابراین، نیاز به ویژگی‌ای به منظور تعیین مجوزهای مورد استفاده‌ی مؤلفه وجود دارد. ویژگی ExportedPermission در عنصر Component بیانگر این مفهوم است.



شکل ۳. توسعه‌ی فرامدل ارائه شده در VAnDroid [۳] (ویژگی‌های اضافه شده با رنگ قرمز مشخص شده‌اند)

۲.۴ تشخیص حمله‌ی وکیل سرگردان در سطح درون برنامه‌ای

در سطح درون برنامه‌ای، یک برنامه به تنهایی در نظر گرفته می‌شود. سپس برای تمام مؤلفه‌های برنامه، فرآیند تشخیص مؤلفه‌ی آسیب‌پذیر انجام می‌شود. اگر مؤلفه‌ای سه شرط مشخصی را دارا باشد، می‌تواند مورد سوء استفاده سایر مؤلفه‌ها قرار گیرد و منجر به حمله وکیل سرگردان شود. این سه شرط عبارتند از: ۱) برنامه مربوط به مؤلفه، در فایل مانیفست، مجوزهایی را درخواست کرده است. ۲) مؤلفه امکان دسترسی به خود را برای سایر مؤلفه‌ها فراهم کرده است و ۳) مؤلفه بخشی از مجوزهای درخواستی برنامه را از طریق فراخوانی APIها استفاده کرده باشد.

فرآیند به‌کار رفته برای تشخیص مؤلفه‌ی آسیب‌پذیر در برابر حمله‌ی وکیل سرگردان در الگوریتم ۱ نشان داده شده است. فرآیند تشخیص با بررسی مجوزهای درخواستی آغاز می‌شود. به عبارت دیگر اگر برنامه در فایل مانیفست خود با استفاده از تگ‌های از پیش تعریف شده `<uses-permission>` و `<uses-permission-SDK-23>` مجوزهایی از نوع خطرناک را درخواست کرده باشد، بایستی تمام مؤلفه‌های برنامه مذکور به منظور کشف مؤلفه‌ی آسیب‌پذیر مورد تحلیل قرار گیرند. در صورت درخواست مجوز از سوی برنامه (خط ۱) فرآیند تحلیل با بررسی وجود و یا عدم وجود ساختار Intent Filter برای هر از یک مؤلفه‌های برنامه ادامه می‌یابد. در صورت وجود ساختار Intent Filter (خط ۳) و همچنین عدم محافظت مؤلفه‌ی مورد نظر توسط هیچ مجوزی (خط ۴)، آن مؤلفه به عنوان یک مؤلفه‌ی آسیب‌پذیر در برابر حمله وکیل سرگردان شناسایی و گزارش می‌شود. این فرآیند تشخیص، در قالب زبان تبدل مدل ATL و همچنین قوانین OCL، با استفاده از چهار قانون و ۱۰ کمک‌کننده [1] پیاده‌سازی شده است.

الگوریتم ۱ تشخیص مؤلفه‌ی آسیب‌پذیر در برابر حمله‌ی وکیل سرگردان در سطح درون برنامه‌ای

Require: One application app1, function IntraAppPE
Ensure: Boolean

```
1: if appl use of permissions then
2:     for each Component c in app1 do
3:         if c has IntentFilter then
4:             if c has not access limitation permission then
5:                 if c use of permissions then
6:                     return True
7:                 else
8:                     return False
9:                 end if
10:            else
11:                return False
12:            end if
13:        else
14:            return False
15:        end if
16:    end for
17: else
18:    return False
19: end if
```

[1] helper

الگوریتم ۲ کشف حمله‌ی وکیل سرگردان در سطح بین برنامه‌ای

Require: Two component srcComp, trgComp, function InterAppPE (srcComp, trgComp) **Boolean**

```
if (trgComp.app.UsesPermissions-srcComp.app.UsesPermissions) is notEmpty then
    if ((trgComp.app.UsesPermissions-srcComp.app.UsesPermissions)-trgComp.permission) is notEmpty then
        return True
    else
        return False
    end if
else
    return False
end if
```

۳.۴ تشخیص حمله‌ی وکیل سرگردان در سطح بین برنامه‌ای

در این حالت با در نظر گرفتن تعاملات بین برنامه‌ها که ناشی از ارسال و دریافت اینتنت توسط مؤلفه‌ها می‌باشد، حمله‌ی وکیل سرگردان در سطح بین برنامه‌ای تشخیص داده می‌شود. در اولین قدم بایستی مؤلفه‌های گیرنده‌ی یک اینتنت تشخیص داده شوند و پس از آن با مقایسه‌ی مجوزهای هر یک از مؤلفه‌های گیرنده و فرستنده، امکان حمله‌ی وکیل سرگردان مورد بررسی قرار گیرد. فرآیند تشخیص حمله‌ی وکیل سرگردان در الگوریتم ۲ نشان داده شده است. تحلیل با هدف کشف این آسیب‌پذیری در سطح بین برنامه‌ای نیازمند کار بر روی مجموعه‌ای از مجوزها و همچنین عملگرهای مجموعه‌ها است.

۵ ارزیابی

برای ارزیابی ابزار VAnDroid+ در زمینه تشخیص حمله‌ی وکیل سرگردان و با هدف نزدیک بودن ارزیابی انجام شده به دنیای واقعی، ده برنامه با بیشترین رواج (بیشترین بارگذاری) در بین کاربران ایرانی مورد تجزیه و تحلیل قرار گرفتند. برای این منظور از سایت App Annie [1] استفاده شده است.

در جدول ۱ نتایج مربوط به کشف مؤلفه‌ی آسیب‌پذیر در برابر حمله‌ی وکیل سرگردان در سطح درون‌برنامه‌ای نشان داده شده است. در این جدول، برای هر نوع مؤلفه، تعداد مؤلفه‌های در دسترس سایر برنامه‌ها و همچنین تعداد Intent Filterها مشخص شده است. همان‌طور که مشاهده می‌شود، از ۱۷ مؤلفه‌ی فعالیت موجود در برنامه Telegram، چهار مؤلفه به صورت عمومی تعریف شده‌اند و به‌عبارت دیگر در برابر حمله‌ی وکیل سرگردان آسیب‌پذیر هستند. همچنین این چهار مؤلفه در مجموع ۱۳ Intent Filter دارند. به بیانی دیگر ۱۳ راه مختلف دسترسی به مؤلفه‌های مذکور وجود دارد. به همین ترتیب از ۲۲ مؤلفه خدمت در برنامه Telegram، ۷ مؤلفه با مجموع ۹ Intent Filter، از ۱۹ مؤلفه دریافت‌کننده همه‌پخشی، ۵ مؤلفه با مجموع ۱۱ Intent Filter می‌توانند به منظور حمله‌ی وکیل

[1] https://www.appannie.com/

جدول ۱. کشف مؤلفه آسیب‌پذیر در برنامه‌های در نظر گرفته شده در ارزیابی

ویژگی‌ها / برنامه	# of Activity		# of Service		# of Receiver	
	# of Intent filter	# of Exported	# of Intent filter	# of Exported	# of Intent filter	# of Exported
Telegram	۱۷		۲۲		۱۹	
	۱۳	۴	۳	۳	۴	۴
		۲۳.۵۳٪		۱۳.۶۳٪		۲۱.۰۵٪
SHAREit	۱۴۹		۳۳		۱۹	
	۲	۲	۰	۰	۰	۰
		۱.۳۴٪		۰		۰
WhatsApp	۱۷۲		۳۱		۱۴	
	۷	۳	۰	۰	۰	۰
		۱.۷۴٪		۰		۰
Anti-Filter Talaplus	۲۴		۳۸		۳۳	
	۱۲	۳	۴	۴	۶	۶
		۱۲.۵٪		۱۰.۵۲٪		۱۸.۱۸٪
Telegram X	۵		۱۰		۱۰	
	۹	۱	۰	۰	۰	۰
		۲۰٪		۰		۰
Divar	۲۸		۱۴		۹	
	۸	۸	۳	۳	۲	۲
		۲۸.۵۷٪		۲۱.۴۲٪		۲۲.۲۲٪

سرگردان مورد سوء استفاده قرار گیرند.

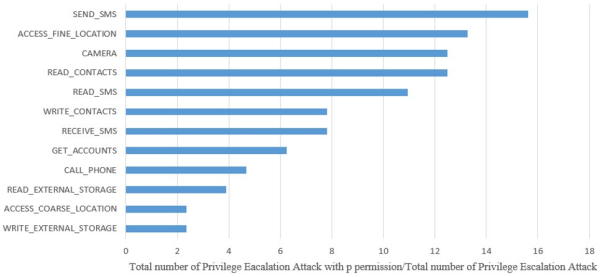
در جدول ۲، تعداد حملات وکیل سرگردان در سطح بین برنامه‌ای نشان داده شده است. هر خانه‌ای از این جدول بیانگر تعداد حملات بالقوه بین برنامه مبدأ و برنامه مقصد است. با توجه به نتایج، در مجموع ۱۲۸ حمله بالقوه وکیل سرگردان بین برنامه‌های وجود دارد. با توجه به جدول ۲، دو برنامه Psiphon و Anti-Filter Tapalus در مجموع سهم حداکثری را (در حدود نیمی از حملات) از این تعداد دارند. همچنین می‌توان دریافت که برنامه SHAREit به عنوان قربانی، بیشتر از سایر برنامه‌ها مورد سوءاستفاده قرار گرفته است.

لازم به ذکر است که همه حملات کشف شده توسط ابزار به صورت دستی مورد تحلیل قرار گرفت و صحت آن‌ها تأیید شد. البته متاسفانه ما مجموعه داده قابل دسترسی شامل برنامه‌های برچسب خورده مستعد حمله وکیل سرگردان (واقعیت زمینه [2]) پیدا نکردیم و لذا برای ارزیابی ابزار به سراغ برنامه‌هایی با بیشترین رواج رفتیم. بنابراین در ارزیابی انجام‌شده اگر چه صحت عملکرد ابزار ارزیابی می‌شود ولی به دلیل عدم دسترسی به مجموعه برنامه‌هایی واقعیت زمینه، امکان ارزیابی جامعیت و دقت عملکرد ابزار برای ما فراهم نشد. این بدان معناست که ما نشان می‌دهیم روش ارائه‌شده قادر به کشف حملات مورد نظر هست (یعنی صحیح عمل می‌کند) ولی نمی‌دانیم آیا همه حملات موجود را شناسایی می‌کند یا نه (ارزیابی جامعیت) و یا چند درصد از حملات موجود را شناسایی می‌کند (ارزیابی دقت). این یکی از محدودیت‌های روش ارزیابی به کار رفته می‌باشد.

[2] ground truth

جدول ۲. کشف حمله وکیل سرگردان در سطح بین برنامه‌ای

برنامه مبدأ / برنامه مقصد	Telegram	Secure VPN	VPN 365	SHAREi	WhatsApp	Super VPN	Anti-Filter Talaplus	Psiphon	TelegramX	Divar
Telegram	-	۰	۰	۹	۰	۰	۰	۰	۰	۰
Secure VPN	۱	-	۰	۲	۳	۰	۱	۰	۱	۰
VPN 365	۱	۰	-	۲	۳	۰	۱	۰	۱	۰
SHAREi	۰	۰	۰	-	۰	۰	۰	۰	۰	۰
WatssApp	۰	۰	۰	۰	-	۰	۰	۰	۰	۰
Super VPN	۳	۰	۰	۳	۰	-	۳	۰	۳	۰
Anti-Filter Talaplus	۰	۰	۰	۲۰	۰	۰	-	۰	۰	۰
Psiphon	۸	۰	۰	۱۴	۲۱	۰	۸	-	۸	۰
TelegramX	۰	۰	۰	۰	۰	۰	۰	۰	-	۰
Divar	۲	۰	۰	۱	۳	۰	۴	۰	۲	-



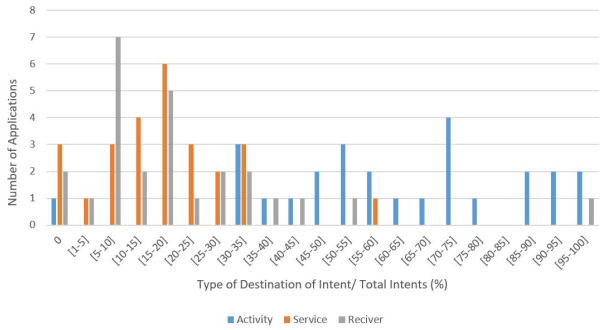
شکل ۴. رده‌بندی مجوزهای مورد سوء استفاده در برنامه‌های مورد ارزیابی

۱.۵ رده‌بندی مجوزهای به‌دست آمده درنتیجه‌ی حمله‌ی افزایش امتیاز

در شکل ۴ مجوزهای به دست آمده ناشی از حمله افزایش امتیاز در برنامه‌های مورد ارزیابی نشان داده شده است. همان طور که مشاهده می‌شود بیش از ۱۵ درصد از حملات منجر به سوء استفاده از مجوز SEND_SMS می‌شود. برنامه‌ها با کسب این مجوز می‌توانند پیام‌های مدنظر خود را به کاربر ارسال کنند. البته این مجوز توسط برنامه‌هایی با هدف کسب سود مالی از طریق ارسال پیام‌های تبلیغاتی مورد استفاده قرار می‌گیرد. از دیگر مجوزهای مورد سوء استفاده قرار گرفته، می‌توان به مجوزهای ACCESS_FINE_LOCATIN، CAMERA و READ_CONTACTS اشاره کرد که هر کدام از آن‌ها سهمی در حدود ۱۲ درصد از تمام حملات را دارند. بدین ترتیب با دستیابی به این مجوزها، برنامه‌ها می‌توانند اطلاعاتی چون مکان کاربر، دوربین دستگاه، مخاطبین و همچنین به پیام‌های کاربر دسترسی داشته باشند.

۲.۵ رده‌بندی مؤلفه‌ی آسیب‌پذیر

در بسیاری از پژوهش‌های انجام شده، نشان داده شده است که مؤلفه از نوع فعالیت در مقایسه با سایر مؤلفه‌ها، بیشتر در معرض آسیب‌پذیری قرار دارد. با توجه به نتایج به دست آمده در جدول ۱ نیز، تعداد مؤلفه‌های آسیب‌پذیر از نوع فعالیت در برنامه‌های در نظر گرفته شده در ارزیابی



شکل ۵. درصد اینتنت‌های ارسالی از نقطه نظر نوع مؤلفه‌ی مقصد

برابر با ۲۲ است. همین مقدار برای مؤلفه از نوع خدمت و دریافت‌کننده همه‌پخشی برابر با ۱۰ و ۱۲ است. بنابراین، ۵۰ درصد از مؤلفه‌های آسیب‌پذیر از نوع مؤلفه فعالیت هستند.

از سوی دیگر با توجه به تعداد مشخصه Intent Filter تعریف شده برای مؤلفه‌های آسیب‌پذیر نیز می‌توان دیدی از میزان آسیب‌پذیری مؤلفه‌ها را به دست آورد. با توجه به جدول ۱، مؤلفه‌های آسیب‌پذیر از نوع فعالیت در مجموع ۵۱ مشخصه Intent Filter را برای خود تعریف کرده‌اند. به عبارت دیگر برای مؤلفه‌های آسیب‌پذیر از نوع فعالیت ۵۱ راه دسترسی وجود دارد که نسبت به دو نوع مؤلفه‌ی دیگر بیشتر است. بالا بودن این مقدار خود بیانگر آسیب‌پذیر بودن مؤلفه‌ی فعالیت نسبت به سایر مؤلفه‌ها است.

میزان آسیب‌پذیری مؤلفه‌ها را نیز می‌توان در عمل با توجه به نوع اینتنت‌هایی که در سیستم ارسال و دریافت می‌شوند، مورد بررسی قرار داد. مؤلفه از نوع فعالیت، خدمت و دریافت‌کننده همه‌پخشی را می‌توان از طریق اینتنت و با استفاده از متدهای خاصی فعال کرد. از جمله این متدها می‌توان به متد StartActivity و StartActivityForResult اشاره کرد که برای فعال کردن مؤلفه‌ی فعالیت استفاده می‌شوند. به همین ترتیب متدهای دیگری نیز برای فعال کردن مؤلفه‌های خدمت و دریافت‌کننده همه‌پخشی وجود دارند. با بررسی متدهای ارسالی هر یک از اینتنت‌های موجود در برنامه‌های در نظر گرفته شده در ارزیابی، می‌توان نوع مؤلفه‌ای که بیشترین اینتنت را دریافت می‌کند، تشخیص داد. شکل ۵ درصد اینتنت‌های ارسالی از نقطه نظر نوع مؤلفه‌ی مقصد در برنامه‌های مورد ارزیابی را نشان می‌دهد. همان طور که در این شکل مشخص است، تعداد برنامه‌هایی که حدود ۵۰ درصد از اینتنت‌های ارسالی آن‌ها، برای مؤلفه‌هایی از نوع Activity ارسال می‌شوند، برابر دو است. همچنین تعداد برنامه‌هایی که حدود ۵۰ درصد از اینتنت‌های ارسالی آن‌ها، برای مؤلفه‌هایی از نوع Service و یا Receiver ارسال می‌شوند، برابر با صفر است. با توجه به شکل ۵ بیش از نیمی از برنامه‌ها، حداقل ۵۰ درصد از اینتنت‌های ارسالی آن‌ها برای مؤلفه‌هایی از نوع Activity ارسال می‌شوند. بنابراین، مؤلفه از نوع فعالیت به نسبت دو نوع مؤلفه‌ی دیگر بیشتر در معرض آسیب‌پذیری قرار دارد. بدین ترتیب، توسعه‌دهندگان برنامه‌های اندرویدی در هنگام استفاده از این نوع مؤلفه در برنامه‌های خود باید با دقت بیشتری عمل کنند.

جدول ۳. مقایسه ابزار +VAnDroid با سایر ابزارهای مشابه

Source App	Destination App	Covert	IccTA+APKCombiner	DIALDroid	VAnDroid+
Inter App Communication					
DeviceId_Broadcast1	Collector	○PE	○PE	✓PE	✓PE
DeviceId_ContentProvider1	Collector	○PE	○PE	○PE	✓PE
DeviceId_OrderedIntent1	Collector	○PE	○PE	○PE	✓PE
DeviceId_Service1	Collector	○PE	○PE	○PE	✓PE
Location1	Collector	○PE	○PE	✓PE	✓PE
Location_Broadcast1	Collector	○PE	○PE	✓PE	✓PE
Location_Service1	Collector	○PE	○PE	○PE	✓PE
Summary					
Number of ○PE		۷	۷	۴	۰
Number of ✓PE		۰	۰	۳	۷
Number of ×PE		۰	۰	۰	۰
Precision P :	✓PE / (×PE + ✓PE)	۰	۰	۱	۱
Recall R :	✓PE / (○PE + ✓PE)	۰	۰	۰.۴۲	۱
F1-measure :	2 P R / (P + R)	۰	۰	۰.۵۹	۱

○PE : False negative ✓PE : True positive

×PE: False positive N/I: Not Implemented

۳.۵ مقایسه با ابزارهای مشابه

مقایسه‌ای بین ابزار +VAnDroid با ابزارهایی چون Covert، IccTA+APKCombiner [۲۵، ۲۶] و DIALDroid در جدول ۳ فراهم شده است. در این مقایسه از مجموعه برنامه‌های DroidBench 3.0 استفاده شده است و براساس مثبت درست[1]، منفی غلط[2] و مثبت غلط[3] و معیارهایی چون Recall، Precision و F1-measure کارایی +VAnDroid در مقایسه با ابزارهای دیگر بررسی شده است. همان طور که مشاهده می‌شود، +VAnDroid توانسته است از ۷ حمله افزایش امتیاز، تمامی آن‌ها را به درستی تشخیص دهد. +VAnDroid در زمینه تشخیص آسیب‌پذیری از نوع حمله افزایش امتیاز توانسته است به ۱۰۰ درصد Precision، ۱۰۰ درصد Recall و ۱۰۰ درصد F1-measure دست یابد که این مقادیر از ابزارهای دیگر بیشتر است.

۶ نتیجه‌گیری

برنامه‌های اندروید، سهم چشمگیری از بازارهای برنامه‌های موبایل را به خود اختصاص داده‌اند. سیستم عامل اندروید به منظور برقراری امنیت در برنامه‌ها، ساز و کارهای امنیتی مختلفی را از جمله ساز و کار مجوزدهی ارائه داده است. با این حال، یک برنامه با کسب مجوزهایی که خود از داشتن آن‌ها محروم بوده است، می‌تواند اعمالی را خارج از محدوده اختیارات خود انجام دهد. این حمله تحت عنوان حمله‌ی افزایش امتیاز شناخته می‌شود.

در این مقاله، رویکردی ایستا مبتنی بر مهندسی معکوس مدل‌رانده به‌نام +VAnDroid ارائه شده است که با به‌کارگیری رویکردی مشابه با ابزار VAnDroid و اضافه کردن ویژگی‌های مورد نیاز به فرامدل طراحی شده، تبدیلات و تحلیل‌های این ابزار، سعی شده است که آسیب‌پذیری از نوع حمله‌ی افزایش امتیاز، به‌نام وکیل سرگردان در دو سطح درون برنامه‌ای و بین برنامه‌ای کشف شود. رویکرد پیشنهادی بر روی تعدادی از برنامه‌های دنیای واقعی مورد ارزیابی قرار گرفته است. بدین منظور ده برنامه از کشور ایران که حداکثر میزان بارگذاری را داشته‌اند، انتخاب و مورد ارزیابی قرار گرفته‌اند. نتایج حاصل از این ارزیابی در قالب رده‌بندی مجوزهای مورد سوء استفاده و همچنین رده‌بندی مؤلفه‌های آسیب‌پذیر نشان داده شده است. نتایج حاصل نشان‌دهنده‌ی اثربخشی و قابلیت +VAnDroid به عنوان یک روش امیدوارکننده در زمینه‌ی تحلیل برنامه‌های اندروید به خصوص تشخیص آسیب‌پذیری وکیل سرگردان است. رویکرد ارائه شده در این ابزار برای شناسایی نوع دیگری از حملات افزایش امتیاز، یعنی آسیب‌پذیری تبانی برنامه‌ها نیز گسترش یافته است که به دلیل محدودیت فضا در این مقاله گزارش نشد. از طرفی، علاوه بر اینتنت، کانال‌های ارتباطی دیگری نیز در سیستم اندروید وجود دارند که از آن جمله می‌توان به ارائه‌دهنده محتوا، SharedPerefernce و External Storage اشاره کرد. تمامی این کانال‌ها توسط مهاجمین می‌توانند مورد سوء استفاده قرار گیرند و منجر به حملاتی چون سرقت همه‌پخشی، ربودن سرویس، ربودن فعالیت و افزایش امتیاز شوند. +VAnDroid تنها با استخراج ارتباطات ناشی از ارسال و دریافت اینتنت بین برنامه‌ها، آسیب‌پذیری‌های موجود را کشف می‌کند. بنابراین پژوهش حاضر می‌تواند به نحوی گسترش یابد که سایر کانال‌های ارتباطی را نیز در کشف آسیب‌پذیری‌ها در نظر بگیرد. علاوه بر این، همانگونه که در بخش ارزیابی هم ذکر شد، ما نیاز به تولید یک مجموعه برنامه واقعیت زمینه برای ارزیابی دقت و جامعیت ابزار ارائه شده در کشف حملات وکیل سرگردان داریم که به عنوان کارهای آتی در ادامه این تحقیق قابل انجام است.

[1] true positive [2] false negative [3] false negative

مراجع

[1] Statista. Number of available applications in the google play store. https://www.statista.com/statistics/266210/number/of-available-applications/in-the-google-play-store/, Date Accessed: June 3, 2020.

[2] Z. Fang, W. Han, and Y. Li. Permission based Android security: Issues and countermeasures. *Computers & Security*, 43:205–218, 2014.

[3] A. Nirumand, B. Zamani, and B. Tork Ladani. VAnDroid: A framework for vulnerability analysis of Android applications using a model-driven reverse engineering technique. *Software: Practice and Experience*, 49(1):70–99, 2019.

[4] A. Misra and A. Dubey. *Android security: attacks and*

Signature-based detection of privilege-escalation attacks on Android. In *2015 conference on information assurance and cyber security (CIACS)*, pages 44–49. IEEE, 2015.

[18] P. P. Chan, L. C. Hui, and S. M. Yiu. Droidchecker: analyzing android applications for capability leak. In *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*, page 125–136, 2012.

[19] Y. Zhongyang, Z. Xin, B. Mao, and L. Xie. DroidAlarm: an all-sided static analysis tool for Android privilege-escalation malware. In *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, page 353–358, 2013.

[20] H. Bagheri, A. Sadeghi, J. Garcia, and S. Malek. Covert: Compositional analysis of android inter-app permission leakage. *IEEE transactions on Software Engineering*, 41(9):866–886, 2015.

[21] A. Bosu, F. Liu, D. Yao, and G. Wang. Collusive data leak and more: Large-scale threat analysis of inter-app communications. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 71–85, 2017.

[22] Github. Jadx: Dex to java decompiler. https://github.com/skylot/jadx, Date Accessed: June 3, 2020.

[23] H. Bruneliere. Generic model-based approaches for software reverse engineering and comprehension. *PhD thesis, Nantes University*, 2018.

[24] Frédéric Jouault, Freddy Allilaire, Jean Bézivin, and Ivan Kurtev. ATL: A model transformation tool. *Science of computer programming*, 72(1):31–39, 2008.

[25] Li Li, Alexandre Bartel, Tegawendé F Bissyandé, Jacques Klein, Yves Le Traon, Steven Arzt, Siegfried Rasthofer, Eric Bodden, Damien Octeau, and Patrick McDaniel. IccTA: Detecting inter-component privacy leaks in android apps. In *Proceedings of the 37th International Conference on Software Engineering-Volume 1*, pages 280–291. IEEE Press, 2015.

[26] Li Li, Alexandre Bartel, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. ApkCombiner: Combining multiple android apps to support inter-app analysis. In *ICT Systems Security and Privacy Protection*, pages 513–527. Springer International Publishing, 2015.

defenses. CRC Press, 2013.

[5] V. Ranganath and J. Mitra. Are free android app security analysis tools effective in detecting known vulnerabilities? *Empirical Software Engineering*, 25(1):178–219, 2020.

[6] U. Sabir, F. Azam, S. Haq, M. Anwar, W. Butt, and A. Amjad. A model driven reverse engineering framework for generating high level uml models from java source code. *IEEE Access*, 7:158931–158950, 2019.

[7] P. O. Rai. *Android Application Security Essentials*. Packt Publishing Ltd, 2013.

[8] J. Six. *Application Security for the Android Platform: Processes, Permissions, and Other Safeguards*. " O'Reilly Media, Inc.", 2011.

[9] Bahman Rashidi and Carol J Fung. A survey of android security threats and defenses. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, 6(3):3–35, 2015.

[10] A. Sadeghi. Efficient permission-aware analysis of android apps. *PhD thesis, University of California, Irvine*, 2017.

[11] D. Youchao. Android malware prediction by permission analysis and data mining. *M.S thesis, The University of Michigan-Dearborn*, 2017.

[12] Mahmoud Hammad. Self-protection of android systems from inter-component communication attacks. *PhD thesis, UNIVERSITY OF CALIFORNIA, IRVINE*, 2018.

[13] L. Davi, A. Dmitrienko, A. Sadeghi, and M. Winandy. Privilege Escalation attacks on android. In *international conference on Information security*, pages 346–360. Springer, 2010.

[14] S. Bugiel, L. Davi, A. Dmitrienko, T. Fischer, A. Sadeghi, and B. Shastry. Towards Taming Privilege-Escalation Attacks on Android. In *NDSS Symposium 2012*. Citeseer, 2012.

[15] Hugo Bruneliere, Jordi Cabot, Grégoire Dupé, and Frédéric Madiot. Modisco: A model driven reverse engineering framework. *Information and Software Technology*, 56(8):1012–1032, 2014.

[16] M. Brambilla, J. Cabot, and M. Wimmer. *Model-Driven Software Engineering in Practice*. Morgan & Claypool Publishers, 2017.

[17] R. H. Niazi, J. A. Shamsi, T. Waseem, and M. M. Khan.