تاریخ دریافت مقاله: ۹۸/۰۵/۱۵
تاریخ پذیرش مقاله: ۹۹/۰۱/۲۴

بررسی ابزارهای تحلیل ایستا برای شناسایی بدافزارهای مبتنی بر ارتباط بین مولفه کاربردهای اندروید

آزاده سروعظیمی
دانشجوی کارشناسی ارشد مهندسی کامپیوتر، دانشکده فنی و مهندسی ـ دانشگاه بوعلی سینا ـ همدان ـ ایران
پست الکترونیکی: a.sarveazimi@basu.ac.ir

مهدی سخایی‌نیا*
استادیارگروه مهندسی کامپیوتر، دانشکده فنی و مهندسی ـ دانشگاه بوعلی سینا ـ همدان ـ ایران
پست الکترونیکی: sakhaei@basu.ac.ir

چکیده

در دنیای مدرن، گوشی‌های هوشمند به دلیل قابلیت‌ها و ویژگی‌های جذابی که دارند بسیار مورد توجه کاربران قرار گرفته و برای همه ضروری هستند. در بین سیستم‌عامل‌های موجـود، اندروید به دلیل ماهیت منبـع باز آن بزرگ‌ترین سـهم بازار را از لحاظ تعداد کاربران در سراسـر جهان دارد. مدل ارتباطی در اندروید به نام ارتباط بین مولفه(ICC) سبب افزایش توسعه برنامه‌های کاربردی و افزایش تعداد کاربران و تولیدکنندگان شـده اسـت. تعداد کاربران زیاد سبب تشـویق مهاجمان شده است تا دستگاه‌های اندروید را مورد هدف قرار دهند. در حالی که اندروید، دارای مجوز پایه‌ای برای حفاظت از دستگاه و منابع است، اما چارچوب امنیتـی را برای دفاع از هرگونه حملـه فراهم نمی‌کند. در این بررسی، درباره تهدیدات امنیتی اندروید و راه‌حل‌های امنیتی موجود که آسیب‌پذیری‌ها را از طریق تحلیل ارتباط بین مولفه‌ها شناسـایی می‌کنند، بحث می‌گردد. ابزارهای تحلیل ایسـتای مبتنی بر تحلیل بین‌مولفـه‌ای از نقطه نظر توانمندی، روش‌شناسـی و محدودیت‌ها بررسی می‌گردد. نتایج این بررسی‌ها نشان داد که ابزارها با هدف قراردادن دو آسـیب‌پذیری ربودن و جعل Intent (توصیف انتزاعیِ عملی که باید اجرا شـود) از نشت اطلاعات و ترفیع امتیاز جلوگیری می‌نماینـد. محدودیت ابزارهای موجود در عدم پشتیبانی از reflection و چندریسگی است که در تحقیقات بعدی مورد توجه قرار گرفته است.

واژه‌هـای کلیـدی: ارتبـاط بیـن مولفـه‌ای،ICC، آسیب‌پذیری‌های اندروید، تحلیل ایستا.

۱.مقدمه

تلفن‌های همراه در گذشـته، فقط برای برقراری تماس استفاده می‌شدند، اما با معرفی گوشی‌های هوشمند، تلفن همراه به یک سیسـتم پردازش دستی محدود تبدیل شده اسـت و بسـیار مورد توجه کاربران قـرار گرفته و برای همه ضروری هسـتند [۱]. توانمندی گوشی‌های هوشمند در پردازش و ذخیره‌سـازی اطلاعات، به دلیل بهره‌گیری

* نویسندهٔ مسئول

از سیستم‌عامل در این گوشی‌هاست[۲]. در صنعت سیستم‌عامل گوشی‌های هوشمند، اندروید به دلیل منبع باز بودن، سریع‌ترین رشد را در بین سیستم‌عامل‌های تلفن همراه داشته است[۳]. همین موضوع سبب شده است که دستگاه‌های هوشمند مورد علاقه بسیاری از مصرف‌کنندگان و تولیدکنندگان برنامه‌های کاربردی قرار گیرد[٤]. از نقطه نظر دیگر موفقیت اندروید از نظر تعداد کاربران و تولیدکنندگان آن، تا حدی می‌تواند به مدل ارتباطی آن به نام ارتباط بین مولفه (ICC)[1] مربوط شود؛ چرا که این مدل می‌تواند به افزایش تولید برنامه‌های کاربردی کمک کند[٥].

داشتن پایگاه کاربری بزرگترِ دستگاه‌های هوشمند نسبت به رایانه‌های شخصی از یک سو و ذخیره‌سازی اطلاعات شخصی حساس و مهم بر روی این دستگاه‌ها از سوی دیگر سبب شده است که حفظ حریم خصوصی و امنیت کاربران گوشی‌های هوشمند به یک نگرانی تبدیل گردد[۳] .سیستم‌عامل اندروید نسبت به سیستم‌عامل‌های دیگر از جمله iOS آسیب‌پذیرتر است[۳]. برای مثال، بر خلاف iOS، صاحبان دستگاه‌های اندروید نیاز به قفل‌شکنی[2] برای نصب برنامه‌ها از منابع ناشناخته ندارند. این به کاربران اندروید اجازه می‌دهد به سادگی با تغییر تنظیمات سیستم، برنامه‌های مخرب یا ممنوع شده‌ای را از گوگل پلی[3] یا بازارهای شخص ثالث نصب کنند که این حریم خصوصی آن‌ها را در معرض خطرات امنیتی قابل توجهی قرار می‌دهد. بنابراین تولیدکنندگان بدافزار می‌توانند از چنین آسیب‌پذیری استفاده کنند و اطلاعات خصوصی کاربر را سرقت نمایند، باتری بیش از حد مصرف کنند، از خدمات تلفنی برای سرقت پول از حساب‌های بانکی کاربران استفاده کنند و حتی دستگاه را به یک بات‌نت زامبی[4] تبدیل کنند[۳]. همچنین می‌توانند از آسیب‌پذیری‌های مدل ارتباط بین مولفه استفاده کنند تا موجب نشت اطلاعات خصوصی کاربران شوند. تمامی این آسیب‌ها حریم خصوصی کاربران را تهدید کرده و می‌توانند به بازار برنامه‌ها و شهرت تولیدکنندگان آسیب برسانند[٥].

با شناخت این نواقص در معماری فعلی اندروید، تلاش‌های زیادی در جهت حل این مشکلات صورت گرفته است. علاوه بر اقدامات امنیتی مختلف مانند جعبه‌شنی[5] و مدل مجوز اندروید، بسیاری از راه‌حل‌های امنیتی و حفظ حریم خصوصی برای مقابله با آسیب‌پذیری‌های امنیتی موجود در سیستم‌عامل اندروید ارائه شده است. هدف این مقاله بررسی فن‌ها و ابزارهایی است که از طریق تحلیل ارتباط بین مولفه بدافزارها را شناسایی می‌کنند.

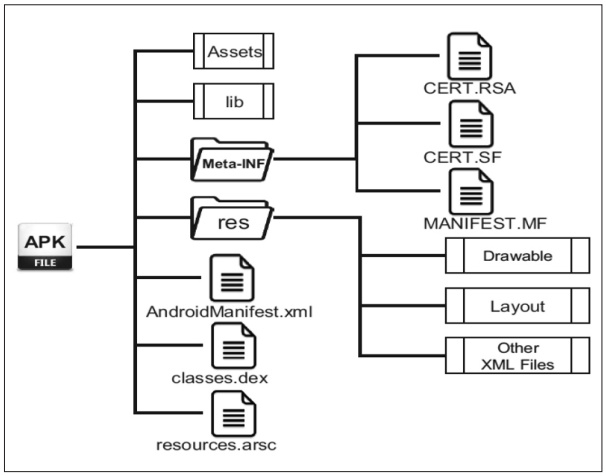
در این مقاله به موضوع‌های زیر پرداخته شده است:

- بررسی برنامه‌های کاربردی اندروید و مفاهیم مرتبط با تهدیدهای امنیتی و سازوکارهای امنیتی موجود
- بررسی سازوکار ارتباط بین مولفه و آسیب‌پذیری‌های آن
- ارائه تحلیلی از ابزاهای تحلیل ایستا مبتنی بر تحلیل درون مولفه و تحلیل ارتباط بین مولفه
- برجسته کردن مشکلات موجود و سیر تحقیقات آتی

نتایج نشان داد که آسیب‌پذیری‌های ارتباط بین‌مولفه ربودن و جعل Intent می‌باشد. ربودن Intent منجر به نشت اطلاعات گردیده و جعل Intent منجر به ترفیع امتیاز خواهد شد. دو ابزار IccTA و Amondriod از سایر ابزارها محدودیت کمتری دارند. مهم‌ترین محدودیت ابزارهای موجود عدم پشتیبانی از reflection و چندریسگی می‌باشد که کارهای آتی در این حوزه بر روی این موضوع‌ها متمرکز است.

ساختار ادامه مقاله به شرح زیر است: در بخش دوم مفاهیم پایه مرتبط با مولفه‌های برنامه‌های کاربردی اندروید و تهدیدات امنیتی و سازوکارهای امنیتی ارائه شده توسط اندروید تشریح شده است. در بخش سوم سازوکار ارتباط بین مولفه و آسیب‌پذیری‌های آن معرفی می‌شود.

1- Inter Component Commmunication(ICC)
2- Jailbreaking
3- Google play
4-Botnets of zombie
5- Sandbox



شکل ۱: ساختار بسته نرم‌افزاری اندروید[۳]

بخش چهارم به شرح تفصیلی ابزارهای تحلیل ایستا مبتنی بر ارتباط بین مولفه اختصاص دارد. در بخش پنجم ابزارها با هم مقایسه شده و محدودیت ابزارها و سیر تحقیقات آتی آن‌ها نشان داده می‌شود. در بخش ششم با نتیجه‌گیری مقاله پایان می‌یابد.

۲.مفاهیم پایه

این بخش مفاهیم ضروری را مرور می‌کند که برای مطالعه بخش‌های بعدی لازم هستند. نمای کلی اندروید و چهار مولفه اصلی آن معرفی می‌شوند. مسائل و تهدیدات امنیتی اندروید تشریح می‌گردد و سپس سازوکارهای امنیتی ارائه شده توسط اندروید مرور می‌شوند.

۲-۱ نمای کلی اندروید و چهار مولفه اصلی آن

یک برنامه اندروید شامل چندین فایل و پوشه است که به‌عنوان یک بسته با پسوند .apk برای توزیع و نصب نرم‌افزار بر روی سیستم‌عامل اندروید مورد استفاده قرار می‌گیرد. شکل ۱ ساختار یک بستهٔ نرم‌افزاری اندروید را نشان می‌دهد. برخی از مولفه‌های خاص در فایل‌های برنامه کاربردی نقش مهمی ایفا می‌کنند. پوشه META-INF شامل MANIFEST.MF است که حاوی یک امضای رمزگذاری است و کلیه محتویات بسته توزیع را تأیید می‌کند. پوشه lib حاوی کد کامپایل شده است، که مخصوص لایه نرم‌افزاری یک پردازنده است و پوشه Assets حاوی دارایی‌های برنامه است که می‌تواند توسط مدیر دارایی بازیابی شود. AndroidManifest.xml یک فایل کلیدی در ساختار برنامه است. این فایل اظهارنامهٔ اندروید است که نام، نسخه، حقوق دسترسی و فایل‌های کتابخانه ارجاع داده شده را برای برنامه توصیف می‌کند[۳]. اظهارنامه یک فایل پیکربندی است که در طول نصب همراه با برنامه است. این اظهارنامه شامل لیستی از مولفه‌هایی است که توسط یک برنامه میزبانی می‌شوند. برخی از اطلاعات آن برای ارتباط مولفه‌ها در زمان اجرا مورد استفاده قرار می‌گیرند و برخی مربوط به مجوزهای اعمال شده و درخواست شده توسط برنامه هستند.

۲-۲ مولفه‌های برنامه‌های کاربردی اندروید

برنامه‌های کاربردی، گوشی‌های هوشمند را فراتر از یک تلفن ساخته‌اند. در اندروید، برنامه‌ها شامل مولفه‌هایی هستند که بلوک‌های کلیدی آن محسوب می‌شوند. چهار مولفه برنامه در اندروید تعریف شده است که همه برنامه‌ها حاوی یک یا چند مورد از این چهار مولفه هستند[۳] :

۱- فعالیت[۶]: نشان دهنده یک صفحه نمایش است که کاربر می‌تواند از طریق آن با برنامه تعامل نماید.

۲- خدمت[۷]: برای نگه داشتن یک برنامه در حال اجرا در پس‌زمینه است که برای انجام بخش‌های پردازشی برنامه در پس‌زمینه و یا انجام کار برای یک فرایند از راه دور استفاده می‌شوند. خدمت دارای رابط کاربری نیست.

۳- گیرنده پخش[۸]: یکی از مولفه‌های برنامه اندروید هست. این گیرنده‌ها پیام‌هایی را که توسط سیستم اندروید یا سایر برنامه‌های اندروید انتشار می‌یابد دریافت می‌کند. نمونه‌هایی از پخش‌های ارسال شده توسط سیستم کم‌بودن باتری[۹]، تغییر وضعیت شبکه و عکس گرفته شده از دوربین است.

ارائه‌دهنده محتوا[۱۰]: مدیریت دستیابی به داده‌های مانا در یک مخزن داده را فراهم می‌نماید. این داده‌ها

6- Activity
7- Service
8- Broadcast Receiver
9- Battery low
10- Content Provider

در داخل برنامه‌هـــا و بین برنامه‌های کاربردی اســـتفاده می‌گردد.

۲-۳ مسائل و تهدیدهای امنیتی اندروید

امنیت اندروید براســـاس ســـازوکار مبتنی بر مجوز[11] است. این سازوکار اجازه دسترسی برنامه‌های کاربردی اندرویـــد شـــخص ثالث بـــه منابع حیاتی در یک دســـتگاه اندرویـــدی را مدیریت می‌کند. ســـازوکار مبتنی بر مجوز به‌دلیـــل کنترل نادرســـت مجوزهـــای برنامـــه و مدیریت ناکارآمد مجوزها، توســـط تولیدکنندگان، فروشـــندگان و کاربران نهایی مورد انتقاد قرار گرفته است. به‌عنوان مثال، کاربران یا همه درخواســـت‌های مجـــوز از برنامه را برای نصـــب آن باید بپذیرند یا برنامه را نصـــب نکنند. این نوع مدیریت مجوز برای امنیت دســـتگاه‌ها مطلوب نیســـت. در ادامه چالش‌های امنیتی اصلی اندروید که منجر به نشـــت اطلاعات کاربر می‌شـــود و حریم خصوصی کاربر را در معرض خطر قرار می‌دهد تشریح گردیده است[۳, ٦].

۲-۳-۱ نشت اطلاعات[12]

در طراحـــی حال حاضر معماری اندروید، دسترســـی برنامه‌ها به منابع یا برنامه‌های دیگر محدود شـــده است. البته کاربر می‌تواند این مجوز را به برنامه‌ها بدهد. اعطای این مجوز قبل از نصب و استفاده از برنامه می‌باشد. نشت اطلاعات زمانی اتفاق می‌افتد کـــه کاربران بدون هیچ‌گونه محدودیتی منابع را در اختیار دیگران قرار می‌دهند. کاربران در اعطای حق دسترسی به برنامه‌ها اغلب بی‌توجه هستند و کمتر به مجوزهای اعطا شـــده دقـــت می‌نمایند. رویکرد «توصیـــه به کاربران» برای محافظـــت از کاربران اندروید منجر به شکســـت شده اســـت[۳, ٦]. عدم آگاهی و تجربه کاربران در اعطای دسترسی به منابع از یک سو و مجبور بودن کاربران به اعطای دسترســـی به منظور استفاده از برنامه‌ها از سوی دیگر، عمده دلایل ضعف سیستم کنترل مجوز کنونی اندروید می‌باشد[۳].

11-Permission-based
12-Information leakage

۲-۳-۲ ترفیع امتیازی[13]

حملات ترفیع امتیازی یـــا ترفیع مجوز با بهره‌برداری از آســـیب‌پذیری‌های هســـته اندروید که برای عموم قابل دسترس است انجام می‌شـــود. هدف این حملات، دریافت مجوز بیشـــتر برای دسترســـی به منابعی است که به‌طور معمـــول در مقابل دسترســـی آن برنامه یا کاربر محافظت شده‌اند. چارچوب امنیتی اندروید مدل مجوز حفاظت شده را اجرا می‌کند که به کاربر اجازه می‌دهد اجازه دسترسی به منابع و داده‌ها بـــرای یک برنامه تنظیم کند. برنامه‌های کاربردی می‌توانند این مدل امنیتی را از طریق اســـتفاده از انتقال مجوز که به‌عنوان افزایش امتیاز شناخته شده است دور بزنند. افزایش امتیاز به ســـناریویی اشاره دارد که در آن دو یا چند برنامه کاربردی با مجموعه محدود مجوزها بـــا یکدیگر ارتبـــاط برقرار می‌کنند تا به‌طور غیرمســـتقیم مجوزهای بیشـــتری را به دســـت آورند و بتوانند اقدامات غیرمجاز انجام دهند.

۲-۳-۳ بازبسته‌بندی[14] برنامه‌های کاربردی

بازبسته‌بندی یکی از مهم‌ترین و رایج‌ترین چالش‌های امنیتی سیستم‌عامل اندروید است. در عملیات بازبسته‌بندی به کمک دی‌کامپایلرها و دیس‌اسمبلرها فایل‌های استخراج شـــده از .apk (کد اجرایی) با اســـتفاده از فن‌های مهندسی معکوس به کد منبع برنامه تبدیل می‌گردد. سپس کد مخرب بـــه کد منبع اصلـــی تزریق می‌گردد و دوبـــاره کد اجرایی تولید می‌گردد. کدهای تولید شده دوباره در یک فایل .apk بسته‌بندی و آماده استفاده می‌گردد. در این حالت، کدهای مخرب به‌عنوان یک برنامه عادی ظاهر می‌شوند. تشخیص دادن یک کد مخرب بازبسته‌بندی شده از یک برنامه عادی کار دشواری است زیرا برنامه بازبسته‌بندی شده معمولا به همان شیوه‌ای که مشروع است عمل می‌کند[۳, ٦].

۲-۳-۴ حمله رد خدمت(DoS)[15]

اهـــداف اصلـــی حمـــلات رد خدمت، اســـتفاده بیش از حد ازپردازنـــده، حافظه، پهنای باند شـــبکه و توان باتری

13-Privilege escalation
14-Repackaging
15-Denial of Service(DoS)

اســـت[۳, ٦] به نحوی که گوشـــی یا دستگاه هوشمند قادر به ارائه خدمات به سایر برنامه‌ها نباشد. از آنجایی که اکثر گوشی‌های هوشمند مجهز به برنامه‌های محافظ(به‌عنوان مثال برنامه‌های ضد ویروس) نیســـتند، برنامه‌های مخرب آن‌ها را به‌عنوان یک بستر مناسب برای حملات رد خدمت پیدا و استفاده می‌کنند.

۲-۳-۵ تبانی[۱۶]

تهدید تبانی یک حمله ســـمت مشـــتری اســـت. در این حمله، کاربران مجموعه‌ای از برنامه‌های تولیدشده توسط تولیدکننده مشـــابه و گواهینامه مشابه را نصب می‌کنند و انواع مختلفی از مجوزها را از جمله حساس و غیرحساس به هریک از برنامه‌ها اعطا می‌کنند. به‌عنوان مثال یک برنامه می‌توانـــد اطلاعات خصوصی فرد را جمـــع‌آوری نماید و برنامه دیگر توانایی انتقال اطلاعات به یک کارساز وب[۱۷] را دارا باشد. چون برنامه‌ها جداگانه نصب می‌گردد کاربران مجوزهای دسترسی به هر برنامه را جداگانه اعطا می‌نماید. اما برنامه دوم با مجوزهـــای برنامه اول قادر خواهد بود که اطلاعات خصوصی کاربر را به یک کارساز وب منتقل نماید. در عمل کد آلوده تقســـیم شده و در دو برنامه قرار گرفته است که حتی برنامه‌های شناسایی کننده بدافزارها هم قادر به شناسایی آن‌ها نیستند[۳, ٦].

۲-٤ سازوکارهای امنیتی اندروید

برنامه‌ها تحت بســـتر متن‌بـــاز و رایـــگان اندروید، با بهره‌گیری از ســـخت‌افزار و نرم‌افزار پیشرفته و همچنین داده‌های مختلف، سعی دارد بستری برای تغییر و نوآوری و دارای ارزش برای مصرف کننـــدگان خود فراهم آورد. بـــرای محافظت از ایـــن ارزش، این بســـتر باید محیطی را ارائـــه کند که امنیـــت کاربران، داده‌ها، برنامه‌ها، دســـتگاه و شـــبکه را تضمین کند. حفاظت از یک بســـتر نیاز به یک معمـــاری امنیتی قوی و برنامه‌هـــای امنیتی دقیق دارد. در این بخش ســـازوکارهایی را که اندروید برای ایجاد محیط برنامه کاربردی امن مورد استفاده قرار می‌دهد ارائه شده اســـت[۳, ٤]. برای دستیابی به امنیت مولفه‌هایی که در بالا توصیف شـــدند اندروید این ویژگی‌هـــای کلیدی امنیتی را فراهم می‌کند[٤]:

۱- امنیت در سطح سیستم عامل از طریق هسته لینوکس: اندروید بر پایه هســـته لینوکس بوده که یک سیستم‌عامل پایدار و امن می‌باشد. اندروید از سازوکارهای امنیتی تعبیه شده در لینوکس بهره می‌برد.

۲- جعبه شـــنی بـــرای همه برنامه‌ها: جعبه شـــنی یک سازوکار امنیتی برای جداسازی برنامه‌های در حال اجرا و محدود کردن منابع دســـتگاه برای این برنامه است. این سازوکار اغلب برای آزمون برنامه‌های غیرقابل اعتماد که ممکن اســـت شـــامل یک ویروس یا کد مخرب دیگر باشد، بدون این‌که به نرم‌افزار یا کد اجازه آســـیب رســـاندن به دستگاه میزبان داده شود، مورد استفاده قرار می‌گیرد.

۳- ارتباطات امن بین مولفه‌ها: اگرچه هر برنامه در یک جعبه شنی اختصاصی اجرا می‌شود، اندروید به برنامه‌های کاربـــردی اجـــازه می‌دهد از طریق ســـازوکار ارتباط بین مولفه بـــا یکدیگر ارتباط برقرار کننـــد. میان‌افزار اندروید، ارتباط بین مولفه‌ها را بین مولفه‌های برنامه واسطه می‌کند. برنامه‌ها می‌توانند مولفه‌ها یا خدمت‌های برنامه‌های دیگر را به‌عنوان خدمت فراخوانی کنند[۳].

٤- امضـــای برنامـــه کاربردی: برای نصـــب و اجرای برنامه‌هـــا بر روی سیســـتم‌عامل اندرویـــد، برنامه‌ها باید به‌صـــورت دیجیتالـــی امضا شـــوند. بـــا اســـتفاده از این سازوکار، سیستم‌عامل اندروید شناسه برنامه را شناسایی می‌کند. این ویژگی همچنین برای ایجاد ارتباط قابل اعتماد بین برنامه‌های کاربردی استفاده می‌شود.

۵- مجوزهای تعریف شده در برنامه و مجوزهای اعطا شده توســـط کاربر: قبل از نصب یک برنامه، نسخه فعلی سیســـتم‌عامل اندروید تمام مجوزهای مـــورد نیاز برنامه را نمایـــش می‌دهد. پس از بررســـی ایـــن مجوزها، کاربر می‌توانـــد تصمیم بگیرد که آن‌ها را بپذیـــرد یا آن‌ها را رد کند، نصب برنامه فقط در صورت پذیرش انجام می‌شود.

16-Colluding
17-Web server

هدف مجوز، حفظ حریم شـــخصی کاربر اســـت. برنامه‌ها نیـــاز دارند مجوز دسترســـی به داده‌های حســـاس کاربر مانند پیام‌های متنی و مخاطبان و همچنین اجزای سیستم خاصی مانند میکروفون، دوربین و دسترسی به شبکه، را درخواست کنند[۳, ۷].

۳. سازوکار ارتباط بین مولفه‌ها

مدل ارتباطی سیســـتم‌عامل اندروید به‌صورت ارتباط بین مولفه اســـت، بدین معنی کـــه برنامه‌ها در اندروید به یکسری مولفه‌ها تقسیم می‌شوند که می‌توانند داده‌ها را در یک برنامه واحد یا حتی در چند برنامه مختلف مبادله کنند. این مدل ارتباطی می‌تواند قابلیت استفاده مجدد از نرم‌افزار را تقویت کند و به این صورت به کاهش بار تولیدکنندگان کمک کرده و موجب افزایش توســـعه برنامه‌های کاربردی شود[۵]. اندروید برای این‌که بتواند ارتباطات بین‌مولفه‌ای را در بین هر ترکیبی از مولفه‌ها به وجود آورد، روش‌های خاصی، که روش‌های ارتباط بین مولفه‌ای نامیده می‌شود، را ارائه می‌دهد. این روش‌ها به‌عنوان پارامتر، نوع خاصی از شـــیء به نام Intent را می‌گیرنـــد که مولفه مقصد پیام را مشـــخص می‌کند. تمام روش‌هـــای ارتباط بین مولفه‌ای با حداقل یک شـــیء Intent به‌عنوان یک پارامتر فراخوانی می‌شوند (بجز روش‌های مرتبط به ارائه دهنده محتوا).

۳-۱ آسیب‌پذیری‌های ارتباط بین مولفه‌ای

سازوکار ارتباط بین مولفه‌ای از یک شیء پیام‌رسانی برای انتقال پیام اســـتفاده می‌کند. متاسفانه سیستم انتقال پیـــام اگر به‌طور نادرســـتی مورد اســـتفاده قـــرار بگیرد می‌تواند به سطح حمله تبدیل شود. در این موارد محتویات پیام‌ها می‌توانند خراب، اصلاح، دزدیده شـــده یا جایگزین شـــوند که می‌تواند حریم خصوصـــی کاربران را به خطر بیندازد. همچنین یک برنامه مخرب می‌تواند پیام‌های جعلی یا مخـــرب را تزریق کند که می‌تواند منجر به دسترســـی/ افشای داده‌های کاربر و نقض سیاست‌های امنیتی برنامه شود. آسیب‌پذیری‌های ناشی از ارتباط بین مولفه‌ها به‌طور مختصر در زیر تشریح شده‌اند.

۱- آســـیب‌پذیری‌های ربودن[18] Intent : اگر فرســـتنده پیـــام به درســـتی گیرنده را مشـــخص نکند، یـــک مهاجم می‌تواند جلوی پیام را بگیرد و محرمانگی یا یکپارچگی آن را به خطر بیندازد. این آسیب‌پذیری شامل ربودن خدمت، ربودن فعالیت و دزدی پخش است[۸].

• ربودن فعالیت: زمانی می‌تواند اتفاق بیفتد که یک Intent برای راه‌اندازی یک فعالیت بدون مقصد مشـــخص ارسال شود.

• ربودن خدمت: زمانی می‌تواند اتفاق بیفتد که یک Intent برای راه‌اندازی یک خدمت بدون مقصد مشـــخص ارسال شود.

• دزدی پخش: زمانی می‌تواند اتفاق بیفتد که یک Intent بدون این‌کـــه با مجوزهای Signature یـــا signatureOrSystem محافظت شود پخش شود.

در هر ســـه مورد ممکن است Intent همراه با اطلاعات حساس توسط یک مولفه مخرب دریافت شود.

۲- آســـیب‌پذیری‌های جعل[19] Intent: اگر در یک مولفه عمومی، ارســـال کننده پیام محدود نگردد، مهاجم می‌تواند پیام‌های مخـــرب را به آن تزریق کنـــد. راه‌اندازی خدمات یا فعالیت مخرب و تزریق پخش، آســـیب‌پذیری‌های جعل Intent هســـتند. یک مولفه عمومی بـــا مجوز signature یا signatureOrSystem محافظـــت نمی‌شـــود. این امر ممکن اســـت سبب گردد که این مولفه توســـط مولفه‌های مخرب راه‌اندازی شوند. این آسیب‌پذیری‌ها می‌تواند منجر به نشت مجوز شود[۸].

۳-آســـیب‌پذیری پخش سیســـتم[20]: بعضی پخش‌های Intent می‌توانند تنها توســـط سیســـتم‌عامل ارسال شوند که در حوزهٔ Action آن‌ها مشـــخص شده است. گیرندگان پخش، با مشـــخص نمـــودن فیلترهـــای Intent با Action مناســـب، می‌توانند بـــرای دریافت این نـــوع پخش ثبت‌نام نمایند. با این‌حال، مولفه‌های عمومی هنوز می‌توانند به‌طور

18- Hijacking
19-Spoofing
20-System Broadcast

مستقیم توسط Intentهای صریح هدف قرار بگیرند. به همین دلیل، گیرنده‌های هدف باید حوزهٔ Action مربوط به Intent دریافت شده را بررسی کنند تا اطمینان حاصل شود که توسط سیستم ارسال شده است[۸].

پیامد این آسیب‌پذیری‌ها حملاتی است که منجر به ترفیع امتیازی و نشت داده‌ها می‌گردد. از این رو شناسایی این آسیب‌پذیری‌ها دارای اهمیت است.

۴.ابزارهای تحلیل ایستا مبتنی بر تحلیل درون مولفه و ارتباط بین مولفه

از آنجایی که رایج‌ترین تهدید امنیتی برنامه‌های اندروید نشت حریم خصوصی می‌باشد، شناسایی این تهدید امنیتی مهم و قابل توجه است. اگرچه بیشتر نشت‌های حریم خصوصی به راحتی قابل شناسایی هستند، زیرا آن‌ها در یک مولفه واحد عمل می‌کنند. اما نشت‌های حریم خصوصی بین مولفه‌ها نیز وجود دارند. بنابراین، تحلیل مولفه‌ها به‌طور جداگانه برای شناسایی نشت‌ها کافی نیست. لذا لازم است که یک تحلیل بین مولفه‌های برنامه‌ها نیز انجام شود. از این‌رو ابزارهایی نیز برای تحلیل ارتباط بین مولفه‌ها طراحی شده‌اند. این بخش شرح ۱۰ ابزار امنیتی است که مبتنی بر تحلیل ایستا هستند و از طریق تحلیل ارتباط بین مولفه، به تحلیل و شناسایی آسیب‌پذیری‌ها پرداخته‌اند. برای هر ابزار ابتدا روش پیشنهادی بیان می‌شود و سپس جزئیاتی از ارزیابی روش و نتایج به دست آمده گزارش می‌شود و در نهایت محدودیت‌ها و چالش‌های موجود تشریح می‌گردد.

٤-۱- Amandroid

روش: Amandroid یک چارچوب عمومی برای انجام تحلیل ایستا برای ارزیابی امنیت برنامه‌های اندروید است[۹]. از آن‌جایی که تحلیل Amandroid به‌طور مستقیم جریان کنترلی و جریان داده‌ای بین مولفه‌ها را مدیریت می‌کند، می‌توان از آن برای حل مشکلات امنیتی که حاصل تعاملات بین مولفه‌های مختلف یک برنامه یا چند برنامه مختلف است، استفاده کرد. تحلیل Amandroid صحیح است زیرا می‌تواند اطمینان از عدم وجود مشکلات امنیتی خاص در یک برنامه با فرضیه‌های خاص و منطقی در سیستم زمان اجرای اندروید و کتابخانه آن را فراهم کند. تحلیل Amandroid قادر است حملات نشت اطلاعات، تزریق داده و سوء استفاده از Api را تشخیص دهد. مراحل اصلی Amandroid به‌صورت زیر است:

۱. Amandroid بایت کد Dalvik برنامه را به یک نمایش میانی قابل قبول برای تحلیل ایستا تبدیل می‌کند.
۲.یک مدل محیطی ایجاد می‌کند که تعاملات سیستم اندروید با برنامه‌ها را شبیه‌سازی می‌کند تا حوزه تحلیل را برای مقیاس‌پذیری محدود کند.
۳.Amandroid یک گراف جریان داده‌ای بین مولفه‌ای (IDFG)[۲۱] از کل برنامه را ایجاد می‌کند. IDFG شامل گراف جریان کنترلی است که همه مولفه‌های قابل دسترس برنامه را پوشش می‌دهد و همچنین مجموعه‌ای از محل‌های ایجاد شیء که به هر نقطه برنامه می‌رسند را ردیابی می‌کند.
٤.گراف وابستگی داده (DDG) را در راس IDFG ایجاد می‌کند که جریان اطلاعات دقیق را شامل می‌شود.
۵.سپس Amandroid می‌تواند در انواع مختلف تحلیل امنیتی با استفاده از اطلاعات ارائه شده در IDFG و DDG اعمال شود. به‌عنوان مثال، می‌توان از DDG برای پیدا کردن این‌که آیا هیچ نشت اطلاعاتی از یک منبع[۲۲] حساس به یک چاهک[۲۳] بحرانی وجود دارد، استفاده کرد. با این پرس‌وجو که: آیا یک زنجیره وابستگی داده از منبع به چاهک وجود دارد یا خیر؟ منبع، داده‌های حساس(مکان کاربر یا دفترچه نشانی) می‌باشد و چاهک یک کانال بالقوه(اتصال شبکه یا امکانات ارسال پیامک) که می‌تواند داده‌ها از طریق آن‌ها به سمت یک دشمن نشت پیدا کند.

آزمایش‌ها و نتایج: Amandroid در انواع تحلیل های

21- Inter-Component Data Flow Graph
22- Source
23- Sink

امنیتی آزمایش شده اســت و در این آزمایش‌ها از چندین مجموعه از برنامه‌ها اســتفاده شــده اســت: ۷۵۳ برنامه محبوب از Google Play، یک مجموعه نمونه بدافزار (شامل ۱۰۰ برنامه) از Arbor Networks و دو محک DroidBench و ICC-Bench. نتایج آزمایش‌ها نشــان می‌دهد که Aman-droid مشــکلات متعــددی را در برنامه‌ها یافته اســت که بســیاری از این نتایج، هرگز توســط پژوهشــگران دیگر گزارش نشده‌اند؛ طبق آزمایش‌های انجام شده میزان دقت (F-measure) Amandroid در محک DroidBench ٦٨درصد است و در محک ICC-Bench ۹۲ درصد گزارش شده است.

محدودیت‌ها و چالش‌ها

۱. پــردازش Amandroid مقــداری آهســته‌تر از آنچه که FlowDroid انجام می‌دهد است [٤] و همچنین استخراج گــراف از بخش قابل توجهــی از نمونه‌ها به شکســت می‌انجامد[۱۰].
۲. Amandroid تحلیــل رشــته دقیــق و کلــی را انجــام نمی‌دهد[۹].
۳. Amandroid در حال حاضر Java reflection، بارگذاری ردهٔ پویــا و فراخوانــی روش‌هــای بومــی را مدیریت نمی‌کند[۹].
٤. اجرای همروند مبتنی بر ریســه در یــک مولفه در حال حاضر توسط Amandroid مدیریت نمی‌شود[۹].
۵. Amandroid به‌طور قابل اعتماد تمام APIهای کتابخانه را تشــخیص نمی‌دهد و مدل دقیق و درستی برای آن‌ها ارائه نمی‌کند[۹].

IccTA –۲–٤

روش: تحلیلگــران برنامه اندرویــد می‌توانند از ابزار IccTA برای شناســایی نشــت‌های حریم خصوصی بین مولفه‌ای استفاده کنند[۵]. IccTA از یک رویکرد دو مرحله‌ای استفاده می‌کند:

۱) استخراج لینک‌های ارتباطی بین مولفه‌ها
۲) تحلیل جریان آلودگی[۲٤] برای ارتباط بین مولفه‌ها

IccTA ابتدا با استفاده از Dexpler بایت کدهای Dalvik را به Jimple که یک نمایش داخلی Soot است تبدیل می‌کند و سپس ارتباطات بین مولفه‌ها را استخراج می‌کند و آن‌ها را ذخیره می‌کند و همچنین تمام داده‌های جمع‌آوری شده (مانند پارامترهــای فراخوانی یا مقادیر فیلتــر Intent) را در یک پایگاه داده ذخیره می‌کند. بر اســاس ارتباطات بین مولفه‌هــا، IccTA نمایش Jimple را اصلاح می‌کند تا به‌طور مســتقیم مولفه‌ها را متصل کرده و قادر به تحلیل جریان داده بین مولفه‌ها باشــد. ســپس IccTA با اســتفاده از یک نسخه اصلاح شده از FlowDroid (یک ابزار تحلیل آلودگی درون مولفه‌ای با دقت بالا بــرای برنامه‌های اندروید) یک گراف جریان کنترلی کامــل از تمام برنامه‌های اندروید را ایجاد می‌کند. این اجــازه می‌دهد که محتوا (به‌عنوان مثال، مقدار Intentها) بین مولفه‌های اندروید منتشر شود و یک تحلیل جریان داده‌ها با دقت بســیار بالا ساخته شود. این اولین رویکردی اســت که مولفه‌هــا را برای تحلیل جریان داده‌هــا به‌طور دقیق به هم متصــل می‌کند. در آخر IccTA مسیرهای آلودگی گزارش شــده (نشت) را در پایگاه داده ذخیره می‌کند.

آزمایش‌ها و نتایج: در آزمایش‌های انجام شده IccTA با دو ابــزار FlowDroid و AppScan (یــک ابزار تجاری توسعه یافته توســط IBM)مقایسه شده است که در نتایج مشــخص شده است که IccTA با دستیابی به میزان دقت[۲۵] ۹٦,٦٪ و بازنمایی[۲٦] ۹٦,٦٪ در DroidBench و ICC-Bench نســبت به دو ابــزار دیگر بهتر عمل می‌کند و قادر اســت نشــت‌های ارتباطات بین مولفه‌ها را در مجموعه بزرگی از برنامه‌های دنیای واقعی پیدا کند[۵].

محدودیت‌هــا و چالش‌هــا: در حــال حاضــر، IccTA فراخوانی‌هــای reflective را تنهــا در صورتــی کــه آرگومان‌های آن‌ها ثابت رشــته‌ای باشــند، پیــدا می‌کند. همچنین این ابزار به چندریســگی[۲۷] بی‌توجه اســت. برای فراخوانی‌هــای بومــی، IccTA محدودیــت FlowDroid را

24-Taint

25-Percision
26-Recall
27- multi-threading

دارا می‌باشـد. در حال حاضر، IccTA برخی از روش‌های ارتباطـات بین مولفه‌ها کـه به ندرت بـه کار می‌روند را مدیریت نمی‌کند. IccTA نمی‌تواند عملیات رشته‌های پیچیده را حـل کند و تحلیل رشـته در یک روش واحد اسـت که ممکن است هشدارهای کاذب ایجاد کند. طبق آزمایش‌هایی که انجام شده اسـت IccTA نمی‌تواند به درستی برخی از برنامه‌ها را تجزیه و تحلیل کند (مصرف بیش از حد حافظه یا متوقف شدن اجرا) [۵].

۳-٤ ScanDroid

روش: ScanDroid، یـک ابزار برای اسـتدلال خودکار در مـورد امنیت برنامه‌هـای اندروید اسـت[۱۱]. Scan-Droid مشـخصات امنیتی را از فایـل اظهارنامه که همراه با چنین برنامه‌هایی هسـتند، اسـتخراج می‌کند و بررسی می‌کند که آیـا جریان‌های داده‌ای از طریق این برنامه‌ها با آن مشخصات سازگار هستند یا خیر. ScanDroid به‌طور ایستا جریان داده‌ها را از طریق برنامه‌های اندروید تحلیل می‌کنـد و می‌تواند تصمیمات مربوط بـه امنیت را به‌طور خودکار بر اسـاس چنیـن جریاناتی تنظیم کنـد. به‌طور خاص می‌تواند تصمیم بگیـرد که آیا اجرای یک برنامه با مجوزهای خاص، بر اسـاس مجوزهای اجرا شده توسط برنامه‌های دیگر، امن هسـت یا نه. ScanDroid یک تحلیل ماژولار از جریان‌ها را درون هر مولفه به اجرا در می‌آورد و سـپس یک مرحلهٔ نهایی که جریان‌ها را در همه مولفه‌ها متصـل و تحلیل می‌کند. کاربر اجـازه می‌یابد برنامه‌های کاربردی را همان‌طور که آن‌ها نصب می‌شوند به‌صورت افزایشـی تحلیل کند و قابلیت استفاده مجدد از مولفه‌ها را در سرتاسر برنامه بهتر منعکس می‌گردد[۱۱].

آزمایش‌ها و نتایج: تولیدکنندگان ScanDroid این ابزار را در چندین نمونه برنامه آزمایش کرده‌اند که نتایج نشان می‌دهـد که این ابزار در کمتـر از چند ثانیه قادر به تحلیل برنامه‌ها است. نویسـندگان ScanDroid هنوز تحلیل خود را به برنامه‌های کاربردی در دنیای واقعی اعمال نکرده‌اند.

محدودیت‌هـا و چالش‌ها: محدودیـت جزئیِ فعلی این تحلیل این اسـت که یا نیاز به کـد منبع جاوا دارد یا این‌که بایت‌کد JVML کامپایل شده از برنامه‌ها در دسترس باشد. با این‌که این روش ممکن است یک مدل معقول برای صدور تاییدیه برون‌خط باشـد، تحلیـل ScanDroid نمی‌تواند بر برنامه‌های بسـته‌بندی شده در یک دستگاه اندروید اعمال شـود. دلیل این موضوع این است که اندروید با یک زبان بایت‌کد سفارشـی کار می‌کند که برای آن هنوز استاندارد مشـخصی وجود ندارد. علاوه بر این، مدل‌سـازی فعلی ScanDroid از اشـیاء پشته، حسـاس به جریان نیست تا به‌طور محافظه‌کارانه‌ای مسائل چندریسگی را حل نماید.

٤-٤ AndroidLeaks

روش: AndroidLeaks یک چارچوب تحلیل ایستا است که به‌طور خودکار نشـت‌های بالقوه از اطلاعات حسـاس در برنامه‌های کاربردی اندروید را در مقیاس وسـیع پیدا می‌کند. AndroidLeaks به شـدت تعـداد برنامه‌ها و تعداد ردیابی‌هایی را که یک ممیز امنیتی باید به‌صورت دسـتی وارسی کند، کاهش می‌دهد[۱۲]. روشگان AndroidLeaks به این صورت است که ابتدا یک نگاشت مجوز ایجاد می‌کند که برای همه تحلیل برنامه‌ها استفاده می‌گردد. این نگاشت بین فراخوانی‌های API اندروید و مجوزهایی است که برای اجـرا به آن‌ها نیاز دارد. از زیرمجموعه‌ای از این نگاشـت برای جریان داده منابع و چاهک‌ها استفاده می‌شود. سپس بـرای هر برنامـه، AndroidLeaks با اسـتفاده از WALA [۱۳] یـک گراف فراخوانی برای تعیین مکان‌های فراخوانی روش‌های منبع و چاهک ایجاد می‌کند. سپس تحلیل قابلیت دسترسـی را انجام می‌دهد تا تعیین کنـد که آیا اطلاعات حسـاس ممکن است در شبکه ارسـال شود یا نه. اگر یک مسیر احتمالی وجود داشته باشد، از تحلیل جریان داده‌ها بـرای تعیین این‌که آیا داده‌هـای خصوصی به یک چاهک شبکه می‌رسند یا نه، استفاده می‌کند. برنامه‌های کاربردی بدون حداقل یک منبع و چاهک تحلیل نمی‌شـوند زیرا این برنامه‌ها نمی‌توانند داده‌های خصوصی را نشت دهند

آزمایش‌هـا و نتایـج: AndroidLeaks بـرای تحلیل

برنامه‌هـــای جدید کـــه در حال حاضر به فروشـــگاه‌های دیجیتالی ارائه می‌شوند، مقیاس‌پذیر بوده و قادر به تحلیل ۲٤۳۵۰ برنامه در ۳۰ ساعت است. AndroidLeaks توانست ۵۷۲۹۹ نشت حریم خصوصی بالقوه را در بیش از ۷٤۰۰ برنامه پیدا نماید. ممیزی دستی این برنامه‌ها تأیید کرد که ۲۳٤۲ برنامه کاربردی داده‌های شخصی را نشت می‌دهند. AndroidLeaks تعـــداد برنامه‌ها و تعداد ردیابی‌هایی را که یک ممیز امنیتی باید به‌صورت دســـتی وارســـی نماید، به میزان قابل توجهی کاهش می‌دهد.

محدودیت‌هـــا و چالش‌هـــا: یکـــی از محدودیت‌هـــای AndroidLeaks این اســـت که هنوز تحلیـــل جریان داده‌ها و جریـــان کنترلی خاص نرم‌افزارهـــای اندروید، Intent و ارائه‌دهنده محتوا، را انجام نمی‌دهد. AndroidLeaks مانند هر تحلیل ایســـتایی از محدودیت‌هـــای ذاتی در خصوص مصالحه دقت، ســـرعت و مثبت‌های کاذب رنج می‌برد. این ابزار ســـریع عمل کرده که منجر به شناســـایی مثبت‌های کاذب زیـــادی می‌گردد، زیرا هدف شناســـایی نشـــت‌های بالقوه برای ممیزهای امنیتی اســـت. این درحالی است که تحلیـــل پویا از دقت بالایی برخودارســـت، زیـــرا در واقع نشت حریم خصوصی در زمان اجرا مشاهده می‌شود، اما پوشـــش تمامی مسیرهای اجرایی ممکن چالش تحلیل پویا است. [۱۲].

۵-٤ ComDroid

روش: ابـــزار ComDroid آســـیب‌پذیری‌های مرتبـــط به ارتباطـــات برنامه کاربردی را تشـــخیص می‌دهد[۱٤]. ComDroid ابتـــدا فایل‌های DEX برنامه را با اســـتفاده از ابزار Dedexer [۱۵] از حالت اسمبلی درمی‌آورد. خروجی ابـــزار Dedexer تجزیه شـــده و آســـیب‌پذیری‌های بالقوه مولفـــه و Intent را ثبت می‌کند. ComDroid ایجاد و انتقال Intent را برای شناســـایی انواع آســـیب‌پذیری‌ها بررسی می‌کنـــد. برای انجـــام این کار، با تحلیل ایســـتای خروجی Dedexer، وضعیت Intentهـــا، فیلترهای Intent، ثبّات‌ها، چاهک‌ها و مولفه‌ها را دنبال می‌نماید. برای هر روشـــی که از Intentها اســـتفاده می‌کند (چه این‌که پارامتر Intent را منتقل می‌کند یا Intent را دریافت می‌کند)، مقدار هر ثابت، رشـــته، رده، Intent و فیلتـــر Intent را نیز دنبال می‌کند. زمانی‌که ComDroid تشخیص دهد که یک Intent ضمنی با مجوز ضعیف یا بدون مجوز ارســـال شده است اخطار می‌دهد. تحلیل مولفـــه ComDroid تصمیم می‌گیرد که آیا مولفه‌هـــا ممکن اســـت به یک حمله جعل Intent حســـاس باشند.

آزمایش‌هـــا و نتایج: تولیدکننـــدگان ComDroid ۱۰۰ برنامه کاربردی را تحلیل کرده و ۲۰ مورد از نتایج تحلیل را به‌صورت دستی بررسی کرده‌اند. آن‌ها تایید کرده‌اند که از این ۲۰ برنامه کاربردی، ۱۲ برنامه با حداقل یک آســـیب پذیری شناسایی شده است.

محدودیت‌ها و چالش‌هـــا: ComDroid در حال حاضر جریان کنترل Intent را در سراســـر توابع پیگیری می‌کند، اما مســـیرها را از طریق دستورات if و switch از یکدیگر تمیز نمی‌دهد. در عوض، همه شـــاخه‌ها را دنبال می‌کند که این می‌تواند به منفی‌های کاذب منجر شـــود. علاوه بر این، این ابزار هنوز Intentهایی که مجوزهای خواندن یا نوشتن URI را در بر دارند تشـــخیص نمی‌دهد و این می‌تواند یک زمینه تحقیق برای آیندگان باشد. ComDroid هشدارها را صادر می‌کند اما وجـــود حملات را تایید نمی‌کند. بنابراین ComDroid مناسب استفاده توسط تولیدکنندگان یا تیم‌های امنیتی(که توســـط تولیدکنندگان قرارداده شده است) است زیـــرا آن‌ها با کد آشـــنا هســـتند یا به کد منبع دسترســـی دارند. با وجود این محدودیت‌ها، نتایج تجربی ComDroid نشان می‌دهد که تحلیل آن‌ها بسیاری از آسیب‌پذیری‌های کاربردی واقعی را شناسایی می‌کند.

۶-٤ Epicc

روش: Epicc یـــک ابـــزار مبتنی بر ابزارهـــای Soot و Heros [۱٦] اســـت کـــه روش‌های ارتباط بیـــن مولفه‌ای و همچنین مقادیر پارامترهای آن‌ها (مانند category، action) را شناســـایی می‌کند[۸]. هدف Epicc استخراج مشخصات

هـــر منبع و چاهک مربـــوط به ارتباطات بیـــن مولفه‌ها در برنامه‌های راه‌اندازی شـــده است. این مشخصات جزئیات نوع، فرم و داده‌های مربوط به ارتباطات را توضیح می‌دهد. اهداف Epicc عبارتند از:

۱- پیدا کردن آسیب‌پذیری‌های ارتباط بین مولفه‌ها

۲- پیـــدا کردن حمـــلات مربوط به آســـیب‌پذیری‌های ارتباط بین مولفه‌ها

۳- تحلیل جریان داده بین مولفه‌ها

همان‌طور که گفته شد Epicc مشخصاتی برای هر منبع و چاهک شناســـایی می‌کند. این شامل محل نقطه ورود یا خـــروج ارتباط بین مولفه‌ها، action مرتبط به Intent، نوع داده و category و همچنیـــن نوع‌هـــای مقدار/کلید Intent و نام مولفه هدف اســـت. مشـــخصات در یـــک پایگاه داده از جریان‌های شناســـایی شده از طریق تطابق مشخصات سازگار ثبت می‌شوند. ساختار مشخصات تضمین می‌کند که این تطابق کارآمد است.

آزمایش‌ها و نتایج: تولیدکنندگان Epicc یک بررســـی را بـــر روی ۱۲۰۰ برنامـــه از بخـــش رایگان فروشـــگاه Google Play انجـــام داده‌اند. نتایـــج حاکی از آن بودند که اکثر مشخصات ارتباط بین مولفه‌ای برای ۹۳٪ از مکان‌های ارتباطی بین مولفه‌ها در برنامه‌های مورد مطالعه شناسایی شدند Epicc با میانگین زمان تحلیل ۱۱۳ ثانیه در هر برنامه نشان داد به خوبی مقیاس‌پذیر است.

محدودیت‌هـــا و چالش‌هـــا: Epicc بـــرای شناســـایی آسیب‌پذیری‌های ارتباط بین مولفه‌ها طراحی شده است. اما تحلیل جریان داده‌ها بر اساس تشخیص آسیب‌پذیری‌های ارتباط بین مولفه‌ها را انجام نمی‌دهد. به عبارت دیگر، Epicc تنها می‌داند که کدام مولفه ممکن است چیزی را نشت دهد، اما نمی‌داند که چه داده‌هایی از طریق نشت جریان می‌یابند و از این‌رو مثبت کاذب زیادی را به دست می‌آورد.

۷-٤ DidFail

روش: هدف از تحلیل DidFail این است که مجموعه‌ای از همـــه جریان‌هـــای ممکن از منبع به چاهـــک را در داخل یـــک مجموعه از برنامه‌هـــای اندروید تولید کنـــد[۱۷]. این تحلیل شـــامل دو مرحله اســـت. در مرحله اول هر برنامه به‌صورت جداگانه تحلیل می‌شود. Intentهای دریافت شده به‌عنوان منبع و Intentهای ارســـال شده به‌عنوان چاهک در نظر گرفته می‌شـــوند. خروجی مرحلـــه اول این تحلیل برای هر برنامه شـــامل: ۱)جریان‌هـــای داخل هر مولفه که توســـط FlowDroid یافت شده‌اند، ۲)شناسایی مشخصات Intentهای ارســـال شده که توســـط Epicc یافت شده‌اند، ۳)فیلترهـــای Intent هـــر مولفه کـــه از فایـــل Manifest استخراج شده‌اند. مرحله دوم تحلیل با استفاده از خروجی مرحله یک بـــر روی یک مجموعه خاص از برنامه‌ها انجام می‌شود. خروجی مرحله دوم شامل تمام جریان‌های منبع بـــه چاهک یافت شـــده در مجموعه برنامه‌ها اســـت. DidFail در ابتدا فقط بر نشـــت‌های ارتباطات بین فعالیت‌ها از طریـــق Intentهای ضمنی متمرکز بود اما این ابزار مجدداً توسط تولیدکنندگان ارتقا یافته و در حال حاضر می‌تواند نشـــت ارتباطات بین هر چهار نوع مولفـــه اندروید را پیدا کند.

آزمایش‌ها و نتایـــج: تولیدکنندگان DidFail این تحلیل را در دو مجموعه برنامه آزمایش کرده‌اند. مجموعه برنامه اول شـــامل ده برنامه‌ای است که خود آن‌ها تولید کرده‌اند و مجموعه برنامه دوم شـــامل ۲۰۰۰ برنامه دنیای واقعی است که از فروشگاه Google Play بارگیری شده و متعلق به بسیاری از دســـته‌های مختلف بازی فروشگاه هستند. آزمون‌های مربوط بـــه ردیابی آلودگی‌ها برای سیســـتم فایل‌ها تایید کرد که این ابزار جریان‌های آلودگی بین برنامه را از طریق فایل‌ها شناســـایی می‌کند. به‌عنوان مثال برای نمونه برنامه‌ها، دقیقا جریـــان آلودگی بین برنامه و فایلی را که برای جریان داده بین دو برنامه اســـتفاده می‌شـــود شناســـایی می‌کند. همچنیـــن برای مجموعـــه برنامه‌های دنیای واقعی، بیش از ۲۰۰ برنامه را که از سیســـتم فایل‌ها می‌خوانند یا می‌نویسند نیز شناسایی می‌کند.

محدودیت‌هـــا و چالش‌ها: محدودیت اصلی در سیستم

فایل جدید و پشـــتیبانی ارائه‌دهنده محتوا این است که پیدا کردن URI در همه موارد ســـخت اســـت، مگـــر این‌که در تابعی که فراخوانی‌های خواندن یا نوشتن در سیستم‌فایل و ارائه‌دهنده محتوا انجام می‌شـــود، کدنویسی شده باشد. محدودیت دیگر این اســـت که DidFail جریان‌های مستقیم بین دو ارائه‌دهنده محتوا را ردیابی نمی‌کند. به‌عنوان مثال، یـــک جریان که در آن یک روش از یـــک ارائه‌دهنده محتوا خوانده و در ارائه‌دهنده محتوا دیگری می‌نویســـد، توسط این تحلیل از دســـت رفته اســـت. محدودیت دیگر DidFail این اســـت که در حال حاضر یک تحلیـــل جامع و کامل را فقط برای گیرنده‌های پخش استاندارد انجام می‌دهد و نیاز به پیشـــرفت‌های بیشـــتری برای انجام تحلیل گیرنده‌های پخش sticky و ordered دارد. DidFail همچنین حوزه‌های ایســـتای شـــامل آرایه‌ها را پشـــتیبانی نمی‌کند. این تحلیل مشکلات عدم‌درستی[۲۸] و عدم‌دقت[۲۹] تحلیل‌های FlowDroid و Epicc را نیز به ارث برده است.

CHEX ۸-٤

روش: Chex یک ابزار برای شناسایی آسیب‌پذیری‌های ربودن مولفه‌هـــا در برنامه‌های اندروید با ردیابی آلودگی بیـــن منابع حســـاس و رابط‌هـــای خارجی در دســـترس اســـت[۱۸]. با ایـــن حال، این ابزار حداکثر حســـاس به یک شـــیء[۳۰] است که ســـبب دقت کمتر در عمل می‌شود. Chex ابتدا برنامه را به چند بخش تقســـیم می‌کند. هر بخش یک قسمت از کد است که از نقطه ورود برنامه قابل دسترسی اســـت. ســـپس خلاصه جریان داده برای هـــر بخش را با استفاده از ابزار Wala محاسبه می‌کند. خلاصه بخش‌ها در تمام جایگشت‌های ممکن به هم متصل می‌گردند به نحوی که توالی فراخوانی‌های سیســـتم اندروید را نقض نکنند و بتوانند بـــا جریان اطلاعات متعددی نتایـــج را ارائه دهند. Chex امکانی برای ردیابی جریـــان داده‌ها از طریق کانال ارتباطی بین مولفه‌ها برخلاف Amandroid ندارد.

آزمایش‌هـــا و نتایج: تولیدکنندگان Chex یک مجموعه بزرگ از برنامه‌های واقعی، شامل ٥٤٨٦ برنامه را آزمایش نموده‌انـــد کـــه ٣٤٨٦ برنامه آن از برنامه‌هـــای محبوب و رایـــگان Android Market و ۲۰۰۰ برنامـــه آن از ســـایر بازارهـــای برنامه‌های کاربردی جمع‌آوری شـــده اســـت. نتایج آزمایش‌ها در مقیاس بالا، نشان داد که کارایی Chex رضایت بخش است.

محدودیت‌هـــا و چالش‌هـــا: ممکـــن اســـت مولفه‌های آســـیب‌پذیری وجود داشـــته باشـــند که حملات ربودن را بدون جریان‌های داده‌ای صریح انجام دهند. در این موارد، وابســـتگی‌های داده‌ای به‌طور اساســـی به وابســـتگی‌های کنترلـــی کدگذاری می‌شـــوند و در نتیجه منابع و چاهک‌ها از طریق جریان داده‌ها متصل نمی‌شـــوند. می‌توان به‌طور انتخابی وابســـتگی کنترل را برای منابع خاص ردیابی کرد به‌طوری که بتوان در جریان تحلیل، جریان داده‌های تقسیم شده در داخل هر بخش را به‌صورت ضمنی در نظر گرفت. از ســـوی دیگر، تحلیل وابستگی کنترلی می‌تواند به راحتی مثبت کاذب زیـــادی را ارائه دهد. Chex فقط جریان داده‌ها را برای شناســـایی آسیب پذیری‌ها کنترل می‌کند که ممکن است منفی‌های نادرست را ایجاد کند.

FlowDroid ۹-٤

روش: FlowDroid یـــک ابـــزار تحلیل ایســـتا منبع باز برای شناســـایی نشت اطلاعات است[۱۹]. این ابزار ICC را مدیریت نمی‌کند و به همین علت نمی‌تواند مســـائل امنیتی مربـــوط به انتقال Intent از میان چندین مولفه را بیان کند. در واقـــع یک ابزار تحلیل آلودگـــی درون مولفه‌ای با دقت بالا برای برنامه‌های اندروید اســـت. هدف FlowDroid به حداقل رساندن تعداد نشت‌های از دست رفته و هشدارهای کاذب است. برنامه‌های اندروید در فایل‌های apk بسته‌بندی می‌شـــوند که اساسا بایگانی فشرده شـــده هستند. پس از خارج کردن یک بایگانی از حالت فشرده، FlowDroid برنامه کاربردی را برای روش‌های چرخه عمر و بازخوانی‌ها[۳۱] و

28-Unsoundness
29-Imprecision
30-object-sensetive
31- callback

نیـــز فراخوانی منابع و چاهک‌ها جســـتجو می‌کند. این کار با تجزیه فایل‌های خاص مختلف اندروید انجام می‌شـــود، از جمله فایل‌های طرح‌بندی XML، فایل‌های dex حاوی کد اجرایـــی و فایل اظهارنامه که فعالیت‌ها، خدمات، گیرندگان پخش و ارائه‌دهنده‌های محتوا در برنامه را مشخص می‌کند. بعـــد، FlowDroid یک روش main مصنوعی را از لیســـت روش‌های چرخه حیات و روش‌های بازخوانی تولید می‌کند. ســـپس این روش main برای تولید یک گراف فراخوانی و گراف جریان کنترلی بین روالی (ICFG) اســـتفاده می‌شود. با شـــروع از منابع تشخیص داده شـــده، تحلیل آلودگی با پیمایش ICFG، آلودگی‌ها را ردگیری می‌کند. FlowDroid با منابع و چاهک‌های اســـتنتاج شده توسط پروژه SuSi [۲۰] که به‌طور جامعی در دســـترس است پیکربندی شده است. فهرســـت واقعی از منابع و چاهک‌ها در وبگاه FlowDroid موجود است. در پایان، FlowDroid تمام جریان‌های کشف شده از منابع به چاهک‌ها را گزارش می‌کند. [۱۹].

آزمایش‌هـــا و نتایـــج: طبق آزمایش‌های انجام شـــده توســـط نویســـندگان FlowDroid گزارش شـــده است که FlowDroid همه نشـــت داده‌هایی کـــه در InsecureBank [۲۱] (InsecureBank یـــک نرم‌افزار اندروید آســـیب‌پذیر است که توسط شرکت Paladion Inc ایجاد شده و به‌طور خاص برای ارزیابی ابزارهای تحلیل طراحی شـــده است) وجـــود دارنـــد را پیدا می‌کند و مثبـــت کاذب و منفی کاذب وجود ندارد. FlowDroid با ابزارهای تجاری AppScan و Fortify SCA مقایســـه شده است که نتایج نشان می‌دهد که FlowDroid دارای دقت بالاتر (۸۶٪) اســـت که نتیجه مثبت کاذب را کم می‌کند.

محدودیت‌هـــا وچالش‌هـــا: اگر چه هـــدف Flowdroid به‌طور کلی تحلیل درســـت و صحیح اســـت، اما برخی از محدودیت‌هـــای ذاتی را با اکثر ابزارهای تحلیل ایســـتا به اشتراک می‌گذارد. به‌عنوان مثال، FlowDroid فراخوانی‌های بازتابـــی را تنها اگـــر آرگومان‌های آن‌ها ثابت رشـــته‌ای باشـــند پیدا می‌کند، که همیشه این‌طور نیست. عدم صحیح بـــودن نیز می‌تواند در مورد چرخه حیات اندروید شـــامل بازخوانی‌هایـــی که ما از آن‌ها آگاه نیســـتیم و یا از طریق روش‌های بومی که قوانین ما آن‌ها را اشتباها مدل می‌کنند رخ بدهد. در حال حاضر FlowDroid نیز به چندریســـگی بی‌توجه اســـت. در واقع فرض می‌کند که ریســـه‌ها در یک ترتیب دلخـــواه اما متوالی اجرا می‌شـــوند که عموما غلط است. پشـــتیبانی کامل و صحیح از چندریسگی یک چالش بزرگ اســـت و بنابراین به کارهای آینده ســـپرده شـــده است[۱۹].

۴-۱۰ PCLeaks

روش: PCLeaks یک ابزار مبتنی بر آســـیب‌های ارتباط بیـــن مولفه‌ها اســـت . در این ابزار به تحلیـــل درون مولفه [۲۲] در برنامه‌های اندروید، نشـــت‌های بالقوه مولفه را که قابلیت اســـتفاده توسط مولفه‌های دیگر را دارد، شناسایی می‌نماید[۲]. PCLeaks نشـــت‌های بالقوه مولفه را در ســـه مولفه فعالیت، خدمات و گیرنده پخش پیدا می‌کند. این ابزار از فن تحلیل ایســـتای آلودگی برای تشـــخیص نشت‌های بالقوه مولفه اســـتفاده می‌کند. یک نشت معمولا با یک منبع شروع می‌شود و به یک چاهک ختم می‌شود. دو نوع نشت وجود دارد. نوع اول، نشت بالقوه مولفه غیرفعال[۳۲](PPCL) اســـت که در نقطه شـــروع[۳۳] یک مولفه اندروید آغاز شده و به یک چاهک ختم می‌شـــود. برای این نوع نشـــت، مولفه به‌طور غیرفعال داده‌هایی را که از مولفه‌های دیگر دریافت کرده است نشت می‌دهد. نوع دوم، نشت بالقوه مولفه فعال (PACL) اســـت که در یک منبع شـــروع می‌شود و به نقطه خروج[۳۴] یک مولفه ختم می‌شود. برای این نشت، این مولفه به‌طور فعال داده‌های حساس را به مولفه‌های دیگر ارسال می‌کند که ممکن اســـت اطلاعات حساس را به‌طور عمدی یا ســـهوی نشـــت دهد. رویکـــرد PCLeaks متکی به گراف جریان کنترلی از برنامه‌های تحلیل شـــده اســـت. دقت این رویکرد بســـتگی به دقت گراف جریان کنترلی تولید شده

32-Passive
33-Entry-point
34-Exit-point

دارد. ماهیت رویداد محور سیستم اندروید باعث ایجاد ناپیوستگی در گراف جریان کنترلی می‌شود که PCLeaks با تولید یک روش main مصنوعی برای این ناپیوستگی‌ها مدل‌سازی می‌شود[۲۳]. سپس یک تحلیل جریان داده‌ای دقیق در گراف جریان کنترلی انجام می‌دهد و در نهایت نشت بالقوه مولفه‌های تشخیص داده شده را در خروجی می‌دهد. تولیدکنندگان PCLeaks همچنین یک ابزار به نام PCLeaksValidator ارائه کرده‌اند که برنامه‌های کاربردی را برای نشت‌های گزارش شده توسط PCLeaks اعتبارسنجی می‌کند. هدف از تولید این برنامه‌ها این است که بررسی کند که آیا نشت‌ها به‌طور واقعی مثبت[۳۵] هستند یا خیر.

آزمایش‌ها و نتایج: تولیدکنندگان PCLeaks این ابزار را بر روی ۲۰۰۰ برنامه که به‌طور تصادفی از Google Play انتخاب شده است آزمایش کرده‌اند. در میان ۲۰۰۰ برنامه، PCLeaks، PACL را در ۴۳ برنامه با ۱۴۳ نشت گزارش می‌کند و همچنین PPCL را در ۱۴۷ برنامه با ۸۴۳ نشت گزارش می‌کند. در بررسی دستی که نویسندگان PCLeaks انجام داده‌اند تأیید کرده‌اند که ۷۵ درصد نتایج آن‌ها مثبت واقعی هستند.

محدودیت‌ها و چالش‌ها: PCLeaks در حال حاضر URIها را که توسط ارائه‌دهنده محتوا استفاده می‌شوند، مدیریت نمی‌کند. بنابراین، PCLeaks قادر به استخراج نشت‌های بالقوه ارائه‌دهنده محتوا نیست. این ابزار از چندریسگی، reflection و دستورات شرطی آگاه نیست. URI، PCLeaksValidatorها را برای تولید برنامه‌های مخرب ناقص مدیریت نمی‌کند، به همین دلیل قادر نیست نشت‌های بالقوه مولفه مربوطه را استخراج کند. در حال حاضر، PCLeaksValidator تنها یک تحلیل رشته را با یک روش واحد انجام می‌دهد که ممکن است منجر به هشدارهای کاذب شود. کدبومی نیز به خوبی تحلیل نشده است.

35-True Positive

۴-۱۱ سایر ابزارها

علاوه بر این ابزارها، ابزارهای دیگری هم وجود دارند که آسیب‌پذیری‌های ارتباط بین مولفه‌ها را تحلیل می‌کنند ازجمله ابزار Woodpecker [۲۴] که هدف آن شناسایی آسیب‌پذیری‌های ارتباط بین مولفه‌ها در بایت‌کد برنامه‌هایی است که از قبل در سیستم اندروید نصب شده‌اند و یا ابزار DroidChecker [۲۵] که بر یافتن آسیب‌پذیری‌های ارتباط بین مولفه‌ها متمرکز است و با انجام یک جریان بین روالی و تحلیل آلودگی‌ها در کد منبع decompile شده, مسیرهای داده قابل استخراج را پیدا می‌کند. ابزار دیگری نیز وجود دارد به نام AppAudit [۲۶] که با استفاده از تحلیل پویا و ایستا، نشت اطلاعات را شناسایی می‌کند. بخش ایستا این ابزار شامل تحلیل گراف فراخوانی است که توابع مشکوک را پیدا می‌کند و بخش پویا بایت کدهای توابع مشکوک را اجرا می‌کند تا نشت‌های حریم خصوصی را که توسط تحلیل قبلی یافته شده است، تأیید کند.

۵. مقایسه ابزارها، مشکلات موجود و سیر تحقیقات آتی

در این بخش، بر اساس تحلیل‌های مطرح شده در بخش قبلی مقایسه‌ای بین ابزارها انجام می‌گردد. جدول ۱ نتایج مقایسه روش‌های مورد بررسی را نشان می‌دهد. همان‌طور که در بخش قبلی نیز بیان شد بجز ابزارهای Flowdriod و PcLeaks که مبتنی بر تحلیل درون مولفه‌ای هستند، سایر ابزارها مبتنی بر تحلیل بین مولفه‌ای هستند. رویکرد اصلی همگی ابزارهای مورد بررسی در این جدول مبتنی بر تشخیص می‌باشد به استثنای ابزار ScanDroid که مبتنی بر تصمیم‌گیری است. ستون‌های این جدول چهار معیار را نشان می‌دهند که روش‌ها بر اساس آن‌ها با هم مقایسه شده‌اند. این معیارها به نحوی انتخاب گردیده‌اند که ضمن سادگی، دسته‌بندی را بر روی این ابزارها داشته و نقاط قوت و ضعف آن‌ها را نشان دهد. معیار اول ایده اصلی ابزار را بیان می‌کند. معیار دوم نتایج ارزیابی ابزار مربوطه را نشان می‌دهد. معیار سوم محدودیت‌های ابزار

جدول ۱: مقایسه ابزارهای موجود

نام ابزار	ایده	ارزیابی	محدودیت	کار آتی
Amandroid	ایجاد گراف جریان داده‌ای بین مولفه‌ای و ایجاد گراف وابستگی داده‌ها در راس آن برای مدیریت مستقیم جریان کنترلی و جریان داده‌ای بین مولفه‌ها	آزمایش ابزار با ۷۵۳ برنامه محبوب از Google Play، یک مجموعه نمونه بدافزار از Arbor Networks و محموعه برنامه‌های DroidBench و ICCBench	عدم مدیریت Java reflection، بارگذاری ردهٔ پویا و فراخوانی روش‌های بومی،عدم تحلیل رشته دقیق،عدم تشخیص تمام API های کتابخانه	افزودن مدیریت استثناها و reflectionها
IccTA	استخراج پیوندهای ارتباط بین مولفه‌ای تجزیه جریان آلودگی‌ها برای ارتباط بین مولفه‌ای	مقایسه با دو ابزار FlowDroid و AppScan در مجموعه برنامه‌های DroidBench و ICCBench	عدم مدیریت کامل فراخوانی‌های reflective و فراخوانی‌های بومی،عدم توجه به چندریسگی، عدم مدیریت برخی از روش‌های ICC	افزودن مدیریت فراخوانی‌های reflective
ScanDroid	تحلیل ماژولار جریان داده درون هر مولفه و سپس اتصال و تحلیل جریان‌های همه مولفه‌ها و در نهایت تولید محدودیت‌های مجوز برای تضمین محرمانگی	آزمایش ابزار با چند نمونه برنامه بدون ارزیابی با برنامه‌های واقعی	نمی‌تواند بلافاصله به برنامه‌های بسته‌بندی شده در یک دستگاه اندروید اعمال شود	ساختن decompilerها از زبان بایت‌کد اندروید به JVML
AndroidLaeks	ایجاد یک نگاشت مجوز بین فراخوانی‌های API و مجوزهای اجرای آن‌ها و تولید یک گراف فراخوانی برای تعیین مکان‌های فراخوانی و بعد تحلیل قابلیت دسترسی از منابع به چاهک‌ها	آزمایش ابزار با ۲۵۹۷۶ برنامه انتخاب شده از Android Market American market Chinese market	عدم تحلیل جریان داده‌ها و جریان کنترلی بین اندروید و مولفه‌های برنامه و ارائه‌دهنده‌های محتوا اشتباه در مثبت‌های کاذب	ترکیب AndroidLeaks با یک رویکرد پویا، انتقال خروجی این ابزار به ابزار BitBlaze به منظور تحلیل کدبومی
ComDroid	تجزیه Dedexerوثبت آسیب‌پذیری‌های پنهانی مولفه و Intent و هشدار به کاربر	آزماش ابزار با ۵۰ برنامه محبوب و رایگان از Android Market و ۵۰ برنامه هزینه دار	عدم تشخیص مسیرها از طریق دستورات if و switch و منفی کاذب، عدم تشخیص Intentهایی که مجوز خواندن یا نوشتن URI را دارند	تشخیص Intentهایی که مجوز خواندن یا نوشتن URI را دارند
Epicc	شناسایی روش‌های ارتباط بین مولفه‌ای و مقادیر پارامترهای آن‌ها، استنباط مشخصات هر منبع و چاهک و یافتن آسیب پذیری ارتباط بین مولفه‌ای	آزمایش ابزار با ۱۲۰۰ برنامه از فروشگاه Google Play	Epicc، URI ها را در اختیار ندارد از این‌رو ارائه‌دهنده‌های محتوا را تحلیل نمی‌کند، مثبت کاذب زیاد، عدم‌درستی وعدم‌دقت	انجام تحلیل Epicc برای مولفه ارائه دهنده محتوا
Chex	شناسایی آسیب‌پذیری‌های hijacking مولفه‌ها با ردیابی آلودگی بین منابع حساس و رابط‌های خارجی	آزمایش ابزار با ۳۴۸۶ برنامه محبوب رایگان از Android Market و ۲۰۰۰ برنامه از سایربازارها	مثبت کاذب بالا منفی کاذب دقت کمتر در عمل	ردیابی وابستگی کنترل به منظور شناسایی مولفه‌های آسیب‌پذیری که حملات hijacking را بدون جریان‌های داده‌ای صریح انجام می‌دهند
DidFail	شناسایی نشت‌های ارتباط بین مولفه‌ای از طریق Intentهای ضمنی با تحلیل جداگانه هر برنامه در مرحله اول و اعمال خروجی مرحله ۱ بر برنامه در مرحله ۲	آزمایش ابزار با دو مجموعه برنامه،مجموعه برنامه اول شامل ۳ برنامه تولید شده توسط نویسندگان ابزار و مجموعه برنامه دوم شامل ۳ برنامه از DroidBench	عدم پشتیبانی کامل از ارائه‌دهنده محتوا، عدم ردیابی جریان‌های مستقیم بین دو ارائه‌دهنده محتوا، عدم پشتیبانی از حوزه‌های ایستا، عدم تحلیل کامل گیرندگان پخش sticky و ordered، نرخ مثبت کاذب بالا، عدم‌درستی و دقت	ردیابی جریان‌های مستقیم بین دو ارائه‌دهنده محتوا، پشتیبانی از حوزه‌های ایستا شامل آرایه‌ها، تعیین مقدار URIها جهت پشتیبانی کامل از ارائه‌دهنده‌های محتوا
FlowDroid	تولید یک روش main مصنوعی و سپس ایجاد یک گراف فراخوانی و گراف جریان کنترلی بین روالی و گزارش مسیرهای آلودگی از منابع به چاهک‌ها	مقایسه با ابزراهای AppScan و Fortify SCA، آزمایش ابزار با نرم‌افزار InsecureBank	عدم مدیریت کامل فراخوانی‌های reflective، عدم‌درستی و عدم‌دقت، عدم توجه به چندریسگی	پشتیبانی کامل و صحیح از چندریسگی
PCLeaks	تولید یک روش main مصنوعی و سپس تحلیل جریان داده‌ها در CFG و در نهایت یافتن نشت مولفه پنهانی	آزمایش ابزار با ۲۰۰۰ برنامه که به‌طور تصادفی از Google Play انتخاب شده	عدم استخراج نشت‌های پنهانی ارائه‌دهنده‌های محتوا ، عدم آگاهی از چندریسگی، reflection، دستورات شرطی و کد بومی	ارتقا PCLeaksValidator به منظور اعتبارسنجی خودکار برنامه‌ها

جدول ۲: ابزارهای استفاده شده در هر ابزار تحلیل

نام ابزار	Epicc	FlowDroid	Dex2IR	ApkCombiner	Dedexer	Soot	Heros	Decompiler	WALA	ded	dex2jar	Dare	Spark	Dexpler	DexLib	apktool
Amandroid			✔													
IccTA	✔	✔		✔												
ScanDroid								✔	✔							
AndroidLaeks									✔	✔	✔					
ComDroid					✔											
Epicc						✔	✔					✔	✔			
DidFail	✔	✔										✔				
Chex									✔						✔	
FlowDroid						✔	✔							✔		
PCLeaks		✔				✔								✔		✔

را بیان می‌کند. معیار چهارم سیر تحقیقات آتی برای هر ابزار را مشخص می‌سازد. جدول ۲ ابزارهای زیرساختی مورد استفاده توسط هر ابزار را نشان می‌دهد. بجز دو ابزار Flowdriod و PcLeaks که خود ابزارهای تحلیل آسیب‌پذیری هستند، سایر ابزارهای ابزارهای پایه تحلیل ایستا هستند.

با بررسی جدول شماره یک مشخص می‌گردد که همه ابزارها بر اساس تحلیل ایستا هستند. همچنین ابزارهایی وجود دارند که از خروجی ابزارهای دیگر برای تشخیص استفاده می‌نمایند. به نظر می‌رسد تلاش گردیده است که از نقاط قوت ابزارهای دیگر استفاده گردیده تا نتایج بهتری به‌دست آید. اگرچه تلاش‌هایی برای تحلیل بر روی کد منبع صورت گرفته است، اما بیشتر ابزارها بر روی کد اجرایی تحلیل را انجام داده‌اند. توجه پژوهشگران در این خصوص بر تحلیل نرم‌افزارها در فروشگاه‌ها بوده است و کمتر بر تحلیل در زمان توسعه پرداخته شده است. محدودیت‌ها نیز نشان می‌دهد که اغلب ابزارها از محدودیت عدم تحلیل چندریسگی و java reflection رنج می‌برند.

۶-نتیجه گیری

رشد روزافزون بدافزارها و برنامه‌های مخرب اندروید و فقدان راه‌حل‌های امنیتی جامع و موثر نیاز به تحقیقات گسترده و تولید ابزارهای کارامد را ضروری می‌سازد. در این مقاله با تشریح مفاهیم مرتبط با تهدیدات امنیتی اندروید و بررسی ۱۰ ابزار تحلیل ایستا (مبتنی بر تحلیل درون مولفه و تحلیل ارتباط بین مولفه)، زمینه مطالعه و تسلط بر کارهای تحلیل ایستا موجود و نقاط قوت و ضعف آن‌ها فراهم گردید. محققان حوزه امنیت اندروید با مطالعه این مقاله مروری قادر می‌شوند تحقیقات خود را در سمت نیاز روز پیش برده و برای تولید و بهبود ابزارهای امنیتی مبتنی بر تحلیل ایستا اقدام نمایند.

مراجع

[1]K. Kulkarni and A. Y. Javaid, "Open Source Android Vulnerability Detection Tools: A Survey," arXiv preprint arXiv:1807.11840, 2018.

[2]M. Haris, B. Jadoon, M. Yousaf, and F. Khan, "Evolution of Android Operating System: A Review," The Asia Pacific Journal of Contemporary Education and Communication Technology vol. 4, no. 1, 2018.

[3]B. Rashidi and C. J. Fung, "A Survey of Android Security Threats and Defenses," JoWUA, vol. 6, no. 3, pp. 3-35, 2015.

[4]R. Singh, "An overview of android operating system and its security," Journal of Engineering Research and Applications, pp. 519-521, 2014.

[5] L. Li et al., "Iccta: Detecting inter-component privacy leaks in android apps," in Proceedings of the 37th International Conference on Software Engineering-Volume 1, 2015: IEEE Press,

pp. 280-291.
[6]A. Bhardwaj and A. Singh, "Android Security Threats and Proposed Solutions: An Overview," International Journal of Scientific and Technical Advancements, vol. 2, no. 4, pp. 103-106, 2016.
[7]F. Guyton, "A survey of Android security threats and machine learning techniques used for detection," 2018.
[8] D. Octeau et al., "Effective inter-component communication mapping in android: An essential step towards holistic security analysis," in Presented as part of the 22nd {USENIX} Security Symposium ({USENIX} Security 13), 2013, pp. 543-558.
[9] F. Wei, S. Roy, and X. Ou, "Amandroid: A precise and general inter-component data flow analysis framework for security vetting of android apps," in Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, 2014: ACM, pp. 1329-1341.
[10]M. Mark and Z. Michael. "Paranoid Android : Android Malware Classification Using Supervised Learning on Call Graphs" snap.stanford.edu (accessed).
[11]A. P. Fuchs, A. Chaudhuri, and J. S. Foster, "Scandroid: Automated security certification of android," Tech. rep., University of Maryland, 2009.
[12] C. Gibler, J. Crussell, J. Erickson, and H. Chen, "AndroidLeaks: automatically detecting potential privacy leaks in android applications on a large scale," in International Conference on Trust and Trustworthy Computing, 2012: Springer, pp. 291-307.
[13]I. T. J. W. R. Center. "watson libraries for analysis (wala)." http://wala.sf.net (accessed).
[14] E. Chin, A. P. Felt, K. Greenwood, and D. Wagner, "Analyzing inter-application communication in Android," in Proc. of the 9th international conference on Mobile systems, applications, and services, 2011: ACM, pp. 239-252.
[15]G. Paller. "Dedexer." http://dedexer.sourceforge.net/. (accessed).
[16] E. Bodden, "Inter-procedural data-flow analysis with ifds/ide and soot," in Proceedings of the ACM SIGPLAN International Workshop on State of the Art in Java Program analysis, 2012: ACM, pp. 3-8.
[17] W. Klieber, L. Flynn, A. Bhosale, L. Jia, and L. Bauer, "Android taint flow analysis for app sets," in Proceedings of the 3rd ACM SIGPLAN International Workshop on the State of the Art in Java Program Analysis, 2014, pp. 1-6.
[18] L. Lu, Z. Li, Z. Wu, W. Lee, and G. Jiang, "Chex: statically vetting android apps for component hijacking vulnerabilities," in Proceedings of the 2012 ACM conference on Computer and communications security, 2012: ACM, pp. 229-240.
[19] S. Arzt et al., "Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps," in Acm Sigplan Notices, 2014, vol. 49, no. 6: ACM, pp. 259-269.
[20] S. Rasthofer, S. Arzt, and E. Bodden, "A Machine-learning Approach for Classifying and Categorizing Android Sources and Sinks," in NDSS, 2014, vol. 14: Citeseer, p. 1125.
[21] Paladion. "Insecurebank test app." http://www.paladion.net/.downloadapp.html. (accessed.
[22] L. Li, A. Bartel, J. Klein, and Y. Le Traon, "Automatically exploiting potential component leaks in android applications," in 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications, 2014: IEEE, pp. 388-397.
[23] L. Li, A. Bartel, J. Klein, and Y. Le Traon, "Detecting privacy leaks in Android Apps," presented at the International Symposium on Engineering Secure Software and Systems, Munich, Germany, 2014.
[24] M. C. Grace, Y. Zhou, Z. Wang, and X. Jiang, "Systematic detection of capability leaks in stock android smartphones," in NDSS, Internet Society, 2012, vol. 14, p. 19.
[25] P. P. Chan, L. C. Hui, and S.-M. Yiu, "Droidchecker: analyzing android applications for capability leak," in Proceed. of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks, 2012: ACM, pp. 125-136.
[26] M. Xia, L. Gong, Y. Lyu, Z. Qi, and X. Liu, "Effective real-time android application auditing," in 2015 IEEE Symposium on Security and Privacy, 2015: IEEE, pp. 899-914.