بررسی آسیب‌پذیری Next-Intent در برنامه‌های کاربردی اندروید: رویکردها و چالش‌ها

زهرا کلوندی[۱]، مهدی سخایی‌نیا [۲،*]

۱- دانشجوی کارشناسی ارشد نرم‌افزار، دانشکده فنی و مهندسی- دانشگاه بوعلی‌سینا- همدان- ایران

۲- استادیار گروه مهندسی کامپیوتر، دانشکده فنی و مهندسی- دانشگاه بوعلی‌سینا- همدان- ایران

چکیده

با پیشرفت روزافزون بسترهای تلفن همراه، سیستم‌عامل اندروید به‌عنوان گسترده‌ترین و محبوب‌ترین سیستم‌عامل شناخته شده است. این امر باعث ایجاد حمله‌های زیادی از سوی مهاجمان به سمت اندروید می‌شود. اغلب این حمله‌ها در صورت وجود آسیب‌پذیرهای متنوع در برنامه‌ها رخ می‌دهد. این آسیب‌پذیری‌ها با استفاده ناامن از قابلیت‌های اندروید به وجود می‌آیند. یکی از قابلیت‌های حساس اندروید، سازوکار ارتباط بین مؤلفه‌ای است که به برنامه‌ها اجازه می‌دهد با مؤلفه‌های عمومی یکدیگر در تماس باشند. همچنین برنامه‌ها برای محافظت از مؤلفه‌های حساس خود می‌توانند آنها را بصورت خصوصی تعریف کنند؛ به این معنی که اجازه ارتباط با سایر برنامه‌ها را ندارند. اما مهاجمان این محافظت را از طریق آسیب‌پذیری Next-Intent دور می‌زنند و با مؤلفه‌های خصوصی ارتباط برقرار می‌کنند. در این مقاله، قابلیت Next-Intent به همراه آسیب‌پذیری و اثرات مخرب آن در برنامه‌های کاربردی بررسی شده است. همچنین رویکردهای امنیتی که قادر به شناسایی آسیب‌پذیری Next-Intent و اثرات آن می‌باشند، مورد مقایسه قرار گرفته‌اند. نتایج مقایسه رویکردها نشان می‌دهد که تا به حال رویکرد مناسبی با مقیاس بالا جهت جلوگیری از وقوع حمله Next-Intent ارائه نشده است. همچنین وجود محدودیت‌ها و هشدارهای کاذب در تحلیلگرهای موجود، از مسائلی است که می‌تواند در تحقیقات آتی این حوزه مورد توجه قرار گیرد.

کلمات کلیدی: اندروید، برنامه کاربردی، مؤلفه‌های خصوصی، سازوکار ارتباط بین مؤلفه‌ای، آسیب‌پذیری Next-Intent، حمله مهاجم.

۱. مقدمه

محبوبیت و سهولت استفاده از اندروید، توسعه‌دهندگان و مشاغل جدیدی را به خود جذب کرده است. درعین‌حال داده‌های حساس ذخیره‌شده در تلفن‌های همراه، مهاجمان بسیاری را وسوسه می‌کند تا به دنبال روش‌های جدیدی برای سرقت اطلاعات و به خطر انداختن سیستم اندروید باشند[۱]. به همین دلیل، محققان به دنبال یافتن رفتارهای مخرب و آسیب‌پذیری‌های موجود در برنامه‌ها هستند تا بتوانند راه‌حل‌های مناسبی برای جلوگیری از اهداف مهاجمان بدست آورند[۲].

* **Corresponding author:** مهدی سخایی‌نیا، استادیار گروه کامپیوتر، دانشکده مهندسی، دانشگاه بوعلی سینا، همدان، ایران
Email: sakhaei@basu.ac.ir

یکی از آسیب‌پذیری‌های پرخطر برنامه‌ها، NIV (آسیب‌پذیری Next-Intent) است که سبب شده مؤلفه‌های خصوصی برنامه آسیب‌پذیر در دسترس برنامه‌های مخرب قرار گیرند. به‌طورکلی یک برنامه کاربردی اندروید فقط می‌تواند از طریق کانال‌های ارتباطی به مؤلفه‌های عمومی یا صادرشده[1] سایر برنامه‌ها دسترسی پیدا کند. مؤلفه‌های که صادر نشده‌اند، به‌عنوان مؤلفه خصوصی فقط توسط مؤلفه‌های همان برنامه مورد دسترسی قرار می‌گیرند[۳]. اما NIV امکان ارتباط با مؤلفه‌های خصوصی را نیز فراهم کرده است. یکی از کانال‌های ارتباطی بین مؤلفه‌ای، Intentها هستند که هم ارتباط درون برنامه‌ای و هم ارتباط برون برنامه‌ای را فراهم می‌کنند[۴]. به همین ترتیب، حمله NIV زمانی اتفاق می‌افتاد که یک برنامه مخرب با ایجاد یک Intent که حاوی Intent دیگری است، ابتدا مؤلفه عمومی برنامه قربانی را مورد هدف قرار می‌دهد و سپس از طریق Intentدرونی، مؤلفه خصوصی آن را فراخوانی می‌کند. در نتیجه مهاجم می‌تواند داده‌ها و APIهای حساس مؤلفه خصوصی را به سرقت ببرد و عملکرد آن را مختل کند.

آسیب‌پذیری‌های مبتنی بر Intent تا کنون مورد مطالعه زیادی قرار گرفته است[۳-۹]. اکثر رویکردهای موجود بر تحلیل جریان داده‌های حساس از منابع به سمت چاهک‌ها متمرکز هستند و توجه بسیار کمی به فراخوانی مؤلفه‌های خصوصی از طریق ارتباط با مؤلفه عمومی شده است. این مورد باعث می‌شود که برنامه در معرض حمله‌های نشت اطلاعات، تزریق داده مخرب، حمله رد خدمت[2] و حمله میانبر قرار گیرد. در این مقاله ضمن تشریح NIV، روش‌های ارائه‌شده برای کشف این آسیب‌پذیری، بررسی و مقایسه شده است. هدف این بررسی، شناسایی مشکل‌ها و چالش‌های موجود در این آسیب‌پذیری و تبیین سیر تحقیقات آتی در این خصوص می‌باشد.

ساختار ادامه مقاله بصورت زیر تنظیم شده است: بخش دوم به مفاهیم پایه در مورد مؤلفه‌های برنامه کاربردی، ارتباط بین‌مؤلفه‌ای و همچنین تهدیدهای موجود می‌پردازد. بخش سوم، قابلیت Next-Intent و آسیب‌پذیری آن را تشریح می‌کند. بخش چهارم، رویکردهای امنیتی موجود که توانایی شناسایی NIV را دارند، مورد بررسی قرار می‌دهد. بخش پنجم و ششم به مقایسه رویکردهای تشریح‌شده و سیر تحقیقات آتی در این زمینه پرداخته‌اند. بخش هفتم نیز به نتیجه‌گیری مقاله اختصاص دارد.

۲. مفاهیم پایه

این بخش ابتدا به تشریح مؤلفه‌های اصلی درون هر برنامه مانند فعالیت، خدمت، گیرنده اعلان و ارائه‌دهنده محتوا می‌پردازد. سپس مدل ارتباطی این مؤلفه‌ها را از طریق کانال ارتباطی Intent و تهدیدهای آن، مورد بررسی قرار می‌دهد.

۲-۱. مؤلفه‌های برنامه‌های کاربردی اندروید

هر برنامه کاربردی اندروید از چندین مؤلفه تشکیل شده است که هر مؤلفه نقش خاصی را بر عهده دارد و برای هدفی مشخص تعریف می‌شود. هر یک از مؤلفه‌ها دارای چرخه‌حیات مشخصی هستند که نحوه ایجاد و پایان آنها را تعیین می‌کند. انواع مؤلفه‌های یک برنامه کاربردی شامل[۳]: ۱) فعالیت[3]: مؤلفه‌ای که صفحه نمایشی را به همراه رابط‌کاربری ارائه می‌دهد. یکی از فعالیت‌های برنامه به عنوان فعالیت اصلی هنگام راه‌اندازی برنامه به کاربر ارائه می‌شود. ۲) خدمت[4]: مؤلفه‌ای که پردازشی را در پس‌زمینه برنامه مدیریت می‌کند. خدمت‌ها رابط‌کاربری ارائه نمی‌دهند و بدون خلل در تعامل کاربر با برنامه،

[1] Exported
[2] Denial of service
[3] Activity
[4] Service

عملیات را در پس‌زمینه اجرا می‌کنند (برای مثال پخش موسیقی). ۳) گیرنده اعلان[1]: رویدادهای سیستم‌عامل اندروید و سایر برنامه‌ها را دریافت می‌کند و متناسب با آن پاسخی بر می‌گرداند (برای مثال پیغام کم شدن باتری). ۴) ارائه‌دهنده محتوا[2]: مجموعه داده‌های مشترک بین برنامه‌ها را مدیریت می‌کند. داده‌ها می‌توانند در سیستم‌فایل، پایگاه داده SQLite یا هر مکان ذخیره‌سازی دیگری وجود داشته باشند.

۲-۲. سازوکار ارتباط بین مؤلفه‌ای

یکی از ویژگی‌های اصلی اندروید، سازوکار ارتباط بین‌مؤلفه‌ای[3] است که برقراری ارتباط مؤلفه‌ها را امکان‌پذیر می‌کند. یکی از مهم‌ترین روش‌های ارتباطی اندروید، Intentها هستند[۱۱]. یک Intent را می‌توان به عنوان یک شیء مستقل در نظر گرفت که یک مؤلفه را برای فراخوانی مشخص می‌کند. برنامه‌ها از Intentها هم برای ارتباط‌های بین‌برنامه‌ای و هم برای ارتباط‌های درون‌برنامه‌ای استفاده می‌کنند. همچنین سیستم‌عامل نیز از Intentها برای ارسال اطلاعیه‌ی رویدادها به برنامه‌های کاربردی استفاده می‌کند. برای فراخوانی سه مؤلفه فعالیت، خدمت و گیرنده اعلان به ترتیب از APIهای startActivity()، startService() و sendBroadcast() استفاده می‌شود.

هنگامی که صراحتاً نام مؤلفه مقصد در تعریف یک Intent مشخص شود، این Intent از نوع صریح خواهد بود. در غیراینصورت، Intent ضمنی است و فقط اقدام مورد نظر را بدون نام مؤلفه مقصد اعلام می‌کند. اقدام‌های هر مؤلفه از طریق IntentFilter در فایل اظهارنامه[4]برنامه اعلام می‌شوند[۱۲]. آنگاه سیستم‌عامل هنگام دریافت Intentها از طریق IntentFilter هر برنامه، تشخیص می‌دهد که چه مؤلفه‌هایی امکان اجرای آن درخواست را دارند. همچنین به همراه یک Intent می‌توان کد، شیء و یا داده‌های اضافی را برای انجام عملیات به مؤلفه مقصد ارسال کرد. سپس مؤلفه مقصد از طریق APIهای getIntent()، getParcelableExtra()، getData() و غیره می‌تواند داده‌های داخل یک Intent را استخراج و پردازش کند. با توجه به اینکه اغلب منشأ پیام‌های Intent بررسی نمی‌شود، با برقراری ارتباط با یک برنامه مخرب، امکان به وجود آمدن انواع حمله‌ها وجود دارد. بنابراین سازوکار ارتباط بین‌مؤلفه‌ای اندروید به بزرگ‌ترین سطح حمله در برنامه‌ها تبدیل شده است.

۲-۳. تهدیدهای مبتنی بر Intent

بطور کلی تهدیدهایی که ممکن است از طریق ارتباط Intentها در یک برنامه به وجود آید، به دو دسته تقسیم می‌شود[۵]: ۱) Intent غیرمجاز[5]: این تهدید هنگامی رخ می‌دهد که برنامه آسیب‌پذیر یک Intent ضمنی تعریف کرده که توسط برنامه مخرب دریافت می‌شود. در این صورت برنامه در معرض تهدیدهای ربودن Intent[6] در مؤلفه‌های فعالیت، خدمت و گیرنده اعلان قرار می‌گیرد[۴]. همچنین اگر برنامه فرستنده به همراه Intent، مجوزی را نیز صادر کرده باشد، برنامه مهاجم از طریق آن به داده‌های ارائه‌دهنده محتوا دسترسی پیدا می‌کند. ۲) Intent جعلی[7]: در این تهدید برنامه مهاجم، Intent مخربی را برای برنامه آسیب‌پذیر ارسال می‌کند. در این صورت حمله تزریق Intent[8] مخرب رخ می‌دهد و

[1] Broadcast receiver
[2] Content provider
[3] Inter Component Communication
[4] Manifest
[5] Unauthorized Intent
[6] Hijacking
[7] Spoofing
[8] Intent injection

اگر برنامه قربانی هیچ بررسی بر روی هویت مبدأ پیام نداشته باشد، باعث راه‌اندازی مؤلفه‌های فعالیت، خدمت و گیرنده اعلان مخرب در برنامه می‌شود[۵]. NIV جزء یکی از عامل‌های این تهدیدها محسوب می‌شود که در بخش بعدی مورد بررسی قرار می‌گیرد.

۳. قابلیت Next-Intent

بطور پیش‌فرض تمام مؤلفه‌ها، خصوصی هستند؛ به این معنی که فقط مؤلفه‌های داخل برنامه، مجاز به فراخوانی آنها می‌باشند. اما در صورت صادر کردن آنها، اجازه ارتباط با سایر برنامه‌ها را دارند. در کل مؤلفه‌های عمومی برنامه‌ها کم هستند اما فقط وجود یک مؤلفه عمومی، تهدیدهای زیادی را از جمله دریافت Intent جعلی از برنامه‌های مخرب ایجاد می‌کند. یکی از این تهدیدها از طریق قابلیت Next-Intent ممکن است رخ دهد. این قابلیت به مؤلفه‌ها این امکان را می‌دهد تا بتوانند Intentهای درون داده‌های اضافی یک Intent را استخراج کنند و مؤلفه Intent داخلی را راه‌اندازی نمایند.

برای مثال، برنامه‌ی Dropbox را در نظر بگیرید[۱۳]. این برنامه دارای یک فعالیت عمومی به نام ActivityLogin است که عملیات ورود به سیستم را انجام می‌دهد. هنگامی که هر یک از فعالیت‌های عمومی برنامه توسط یک مؤلفه بیرونی فراخوانی شوند، ابتدا این فعالیت باید بررسی کند که آیا کاربر در وضعیت Log-In هست یا خیر. اگر نباشد، کاربر را به سمت ActivityLogin هدایت می‌کند تا عملیات ورود به سیستم را تکمیل سازد. در واقع یک Intent برای فعالیت ActivityLogin ارسال کرده و Intent دریافت شده از مؤلفه بیرونی را تحت عنوان Next-Intent در داده‌های اضافی Intent اولیه ذخیره می‌کند. حال هنگامی که کاربر فرایند ورود به سیستم را تکمیل کند فعالیت ActivityLogin، Intent درونی را بازیابی می‌کند و از آن برای فراخوانی مؤلفه عمومی اولیه استفاده می‌کند تا کاربر ادامه عملیات خود را انجام دهد. این قابلیت برای تعامل بین مؤلفه‌های فعالیت هر برنامه بسیار استفاده می‌شود اما به دلیل عدم محدودیت در استخراج Intentهای درونی، این قابلیت به یک آسیب‌پذیری تبدیل شده است.

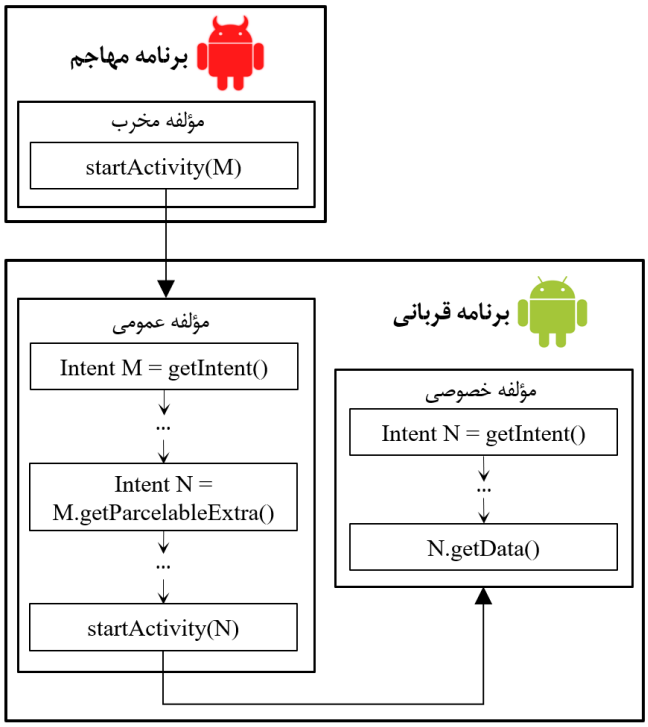
۳-۱. بهره‌برداری از NIV

طی تحقیقات انجام‌شده در سال ۲۰۱۳[۱۳] برای اولین بار، قابلیت Next-Intent به عنوان یک آسیب‌پذیری مطرح شد که از طریق آن برنامه‌های مخرب می‌توانند به مؤلفه‌های خصوصی برنامه آسیب‌پذیر دسترسی پیدا کنند. این اتفاق زمانی رخ می‌دهد که برنامه آسیب‌پذیر بدون بررسی هویتی روی Intentهای دریافتی، Intentهای اضافی آن را استخراج و پردازش می‌کند.

برای مثال، همان قابلیت Next-Intent را در برنامه Dropbox در نظر بگیرید که توسط یک برنامه مخرب مورد سوءاستفاده قرار گیرد[۱۳]. از آنجایی‌که ActivityLogin یک مؤلفه عمومی است، یک برنامه مخرب می‌تواند آن را با یک Intent راه‌اندازی کند که درون آن یک نمونه Intent دیگری با هدف مؤلفه خصوصی نهفته است. این هدف در صورتی که کاربر در وضعیت Log-In برنامه باشد مشکلی به وجود نمی‌آورد. اما اگر در این وضعیت نباشد، کاربر به فعالیت ActivityLogin هدایت می‌شود؛ این فعالیت بعد از تکمیل فرایند ورود به سیستم، Intent درونی را بازیابی می‌کند و هدف آن را راه‌اندازی می‌کند. از آنجا که اکنون Intent درونی توسط خود برنامه Dropbox فراخوانی شده است، همه مؤلفه‌های آن از جمله مؤلفه‌های خصوصی می‌توانند راه‌اندازی شوند. این مشکل نتیجه جدی را به همراه دارد به دلیل اینکه اجازه می‌دهد برنامه مخرب کنترل زیادی روی برنامه قربانی پیدا کند. این آسیب‌پذیری در بیشتر برنامه‌های محبوب و دارای حساب کاربری بالا مانند: Facebook، Dropbox، Twitter، Viber و غیره یافت شده است[۱۳-۱۵].

۳-۲. نمونه‌ی بهره‌برداری از NIV

شکل ۱ نمونه‌ی از حمله با سوءاستفاده از NIV است. در این نمونه، یک برنامه تحت تأثیر NIV و یک برنامه مخرب با هدف برقراری ارتباط با مؤلفه خصوصی برنامه آسیب‌پذیر، بر روی تلفن همراه نصب شده‌اند[۱۴]. برنامه مخرب برای سوءاستفاده از قابلیت Next-Intent برنامه قربانی، از طریق Intent(M) ابتدا با مؤلفه عمومی این برنامه ارتباط برقرار می‌کند. درون داده‌های اضافی Intent(M) یک Intent دیگری به نام Intent(N) قرار گرفته که مقصد آن یکی از مؤلفه-های خصوصی بوده و حاوی داده‌های مخرب جهت مختل کردن عملکرد آن می‌باشد. بنابراین هنگامی‌که مؤلفه عمومی، Intent(M) را دریافت می‌کند، با استخراج و پردازش داده‌های اضافی آن وادار به فراخوانی مؤلفه خصوصی و ارسال داده‌ها به آن می‌شود. بدین ترتیب مؤلفه خصوصی، Intent(N) را دریافت و هنگام پردازش آن مورد سوءاستفاده برنامه مهاجم قرار می‌گیرد.



شکل ۱ – سناریو حمله با سوءاستفاده از NIV

۳-۳. پیامدهای ناشی از NIV

NIV در برنامه‌های مختلف با توجه به قابلیت مؤلفه‌های خصوصی، حمله‌های متفاوتی را ایجاد می‌کند که عواقب جدی در برنامه‌های آسیب‌پذیر به همراه دارد. این بخش پیامدهای معمولی که NIV تا کنون بر روی برنامه‌های آسیب‌پذیر داشته را شرح می‌دهد.

۱. **تزریق داده‌های مخرب:** در این نوع آسیب‌پذیری، هدف مهاجم دستکاری اطلاعات کاربران می‌باشد. بنابراین به همراه Next-Intent، داده‌های مخربی را برای مؤلفه‌های خصوصی ارسال می‌کند. بسته به عملکرد مؤلفه خصوصی مورد هدف، داده‌های مخرب متفاوت است. به عنوان مثال در بیشتر موارد، مؤلفه‌های خصوصی یک URL را از Intentهای دریافتی استخراج می‌کنند تا در صفحه وب، بارگذاری نمایند. بدین صورت مهاجمان می‌توانند صفحات

وب دلخواه خود را در برنامه بارگذاری کنند. در برخی موارد نیز داده‌های اضافی Intentها به عنوان یک عکس، ویدئو، قیمت محصول‌ها و غیره استفاده می‌شود[۱۵].

۲. **نشت اطلاعات و حریم خصوصی:** در این نوع آسیب‌پذیری، مؤلفه‌های خصوصی داده‌های حساس را به همراه Intent ورودی به چاهک‌های برنامه ارسال می‌کنند[۱۳]. برای مثال یک مؤلفه خصوصی را در نظر بگیرید که URL یک فایل ویدئویی را به عنوان پارامتر از Intent ورودی دریافت می‌کند. سپس این URL را به همراه اطلاعات احراز هویت کاربر به آدرس یک دامنه ارسال می‌کند. بنابراین از آنجایی که این برنامه تحت تأثیر NIV است، برنامه مخرب به همراه URL فایل ویدئویی، آدرس صفحه مخرب خود را برای داده‌های اضافی Next-Intent تنظیم می‌کند. در نتیجه با فراخوانی مؤلفه خصوصی و ارسال اطلاعات احراز هویت کاربر به صفحه مخرب، مهاجم توانسته به اطلاعات اشخاص دسترسی پیدا کند.

۳. **حمله رد خدمت:** یکی دیگر از اثرات NIV، همانند تأثیر حمله رد خدمت در برنامه‌ها می‌باشد[۱۵]. بدین صورت که برخی متدها در مؤلفه‌های خصوصی، توانایی کنترل کردن مقادیر Null را ندارند. بنابراین برنامه مهاجم با قرار دادن مقدار Null به عنوان داده‌های اضافی Next-Intent موجب مختل شدن عملکرد برنامه می‌شود.

۴. **حمله میانبر**[1]: سوءاستفاده از NIV همانند حمله میانبر عمل می‌کند که در آن مهاجم با دور زدن محافظت‌های برنامه به منابع حساس دسترسی پیدا می‌کند[۱۴]. در NIV نیز مهاجم با دور زدن مؤلفه عمومی، مؤلفه‌های خصوصی را که دارای عملکردهای حساسی هستند تحت کنترل قرار می‌دهد و جریان‌های برنامه را مختل می‌کند.

۴. رویکردهای امنیتی NIV

جهت شناسایی NIV و اثرات آن در برنامه‌های اندروید، رویکردهای متفاوتی ارائه شده است. در این بخش، عملکرد این رویکردها به همراه نتایج و محدودیت آنها بررسی می‌شود.

۴-۱. Morbs

Morbs، توسط وانگ و همکارانش در سال ۲۰۱۳ برای اولین بار جهت تشخیص ارتباط‌های ناامن در برنامه‌های تلفن همراه توسعه یافته است[۱۳]. این ابزار یک سازوکار امنیتی برای کنترل ارتباطات بین‌برنامه‌ای در سیستم‌عامل‌های تلفن همراه می‌باشد. رویکرد Morbs بسیار مشابه سازوکارهای امنیتی مبتنی بر مرورگر عمل می‌کند. مشابه سیاست مبدأ مرورگرهای وب که از دسترسی مستقیم محتوای وب یک دامنه به دامنه‌های دیگر محافظت می‌کنند، Morbs نیز از ارتباط یک برنامه غیرمجاز با منابع حساس برنامه‌های دیگر، جلوگیری می‌کند. Morbs با حفاظت از کانال‌های ارتباطی اصلی در تلفن همراه مانند: Intent، schema و کلاس‌های کمکی وب باعث می‌شود یک برنامه تنها به درخواست‌های معتبر پاسخ دهد. برای انجام این کار، Morbs هر پیام را با اطلاعات اصلی مبدأ آن برچسب‌گذاری می‌کند. همچنین به توسعه‌دهندگان اجازه می‌دهد تا لیستی از برنامه‌های مجاز و سیاست مورد نظرکانال‌های ارتباطی را در سیستم‌عامل اعمال کنند. بدین صورت هنگام برقراری ارتباط با هر برنامه، مانیتور مرجع در سیستم‌عامل، مبدأ/ مقصد هر پیام را شناسایی می‌کند و در صورت غیرمجاز بودن آن‌ها، ارتباط را مسدود می‌کند.

نتایج: Morbs به عنوان اولین سازوکار حفاظت مبتنی بر مبدأ، توانسته حمله‌های منحصر به فردی را که از طریق ارتباط‌های ناامن رخ می‌دهد، گزارش کند. همچنین با ارائه رویکرد پیشنهادی و پیاده‌سازی آن در ۲۰ برنامه، از وقوع این

[1] Bypass

حمله‌ها جلوگیری کرده است. ارزیابی این ابزار نشان می‌دهد که بدون تغییر در کد برنامه‌های آسیب‌پذیر، توانسته NIV را به طور اثربخشی شناسایی کند. همچنین با حفظ سازگاری و حداقل تلاش توسعه‌دهنده ارائه شده است.

محدودیت‌ها: ابزار Morbs یک روش خودکار جهت شناسایی NIV و جلوگیری از آن در مقیاس وسیعی از برنامه‌ها ارائه نمی‌دهد. به دلیل اینکه جهت تعریف سیاست‌های هر برنامه و لیست مبدأهای مجاز به توسعه‌دهندگان آنها نیاز دارد. بنابراین این اصلاحات و تلاش توسعه‌دهندگان باعث عدم کاربرد Morbs در فروشگاه‌های برنامه‌های کاربردی می‌شود. همچنین عملیات ذخیره‌سازی محتوای سیاست هر برنامه در پایگاه داده و فرایند بررسی فرستنده و گیرنده هر پیام با لیست برنامه‌های مجاز، موجب تأخیر در زمان اجرای برنامه‌ها می‌شود.

NIVAnalyzer .۲-۴

ابزار بعدی که در این زمینه ارائه شده، ابزار NIVAnalyzer می‌باشد[۱۵]. NIVAnalyzer یک ابزار تحلیل ایستا و پویا است. این ابزار در تحلیل ایستا با ردیابی جریان Intentها، به بررسی NIV در تمامی مؤلفه‌ها می‌پردازد. برای این بررسی، NIVAnalyzer کد باینری Dalvik مؤلفه‌های برنامه را به کدهای smali تبدیل می‌کند. سپس در هر کلاس به دنبال نقاط کلیدی می‌گردد. نقاط کلیدی شامل دستورالعمل‌های getParcelableExtra() می‌باشد که توانایی بازیابی یک Intent را از داده‌های اضافی Intent دیگر دارند. سپس با شروع از هر نقطه کلیدی، تمام مسیرهای اجرایی احتمالی را محاسبه می‌کند. در ادامه هر مسیر را یک بار با تحلیل روبه‌جلو بررسی می‌کند که آیا جریان Intent بازیابی شده از getParcelableExtra() به متدهای فراخوانی منتقل می‌شود یا خیر.

متدهای فراخوانی، به عنوان روش‌های راه‌اندازی مؤلفه‌های دیگر استفاده می‌شوند مانند: startActivity()، startService()، sendBroadcast() و غیره. در صورت انتقال به یک متد فراخوانی، این بار Intent اولیه را بصورت تحلیل روبه‌عقب بررسی می‌کند. چناچه این Intent توسط دستور getIntent() در یک مؤلفه عمومی تعریف شده باشد، این مسیر به عنوان مسیر آسیب‌پذیر به ماژول بهره‌برداری NIV ارسال می‌شود. ماژول بهره‌برداری طبق اطلاعات دریافت شده از هر مسیر، موارد آزمایشی را جهت بهره‌برداری از برنامه آسیب‌پذیر تولید می‌کند. سپس با اجرای برنامه به همراه موارد آزمایشی بر روی شبیه‌ساز اندروید، وجود NIV را تأیید می‌کند.

نتایج: NIVAnalyzer جهت ارزیابی، ۲۰ هزار برنامه کاربردی اندروید را مورد تجزیه‌وتحلیل قراره داده و آسیب-پذیری ۱۹۰ برنامه را تأیید کرده است. این ابزار تمامی مؤلفه‌های که مستعد NIV می‌باشند را که شامل مؤلفه‌های فعالیت، خدمت و گیرنده اعلان هستند، مورد تحلیل قرار می‌دهد. همچنین علاوه بر اثربخشی، جهت دستیابی به کارایی بیشتر از پردازش چندگانه جهت اجرای موازی برنامه‌ها استفاده می‌کند.

محدودیت‌ها: NIVAnalyzer به دلیل اینکه برای تعیین توقف تولید مسیرهای اجرایی به یک آستانه متکی است؛ هشدارهای منفی‌کاذب تولید می‌کند. ازاین‌رو، دقت تجزیه‌وتحلیل جریان Intentها در این ابزار، نیاز به بهبود بیشتری دارد. همچنین در تحلیل ایستا، تنها متدهای getParcelableExtra() را به عنوان نقاط کلیدی اولیه جستجو می‌کند، در صورتی که از روش‌های getParcelableArrayExtra() و getParcelableArrayListExtra() نیز می‌توان برای بازیابی Intentهای درونی استفاده کرد[۱۵]. همچنین در تولید موارد آزمایشی برای بهبود زمان اجرا، می‌توان در کارهای آینده از تکنیک‌های اجرای نمادین برای ساخت Intentهای بهره‌بردای استفاده کرد.

NIVD .۳-۴

یکی از تحلیلگرهای که اخیراً برای شناسایی NIV طراحی شده، ابزار NIVD است[۱۴]. این ابزار با تحلیل ایستا، برنامه‌های اندروید را جهت شناسایی NIV تجزیه‌وتحلیل می‌کند. NIVD ابتدا هر فایل apk. را از حالت کامپایل خارج می‌کند. سپس تمام مؤلفه‌های فعالیت‌های عمومی برنامه را جستجو می‌کند و با یافتن ترتیب توالی از APIهای مربوط به NIV در هر مؤلفه، مؤلفه‌های تحت تأثیر را استخراج می‌کند. این توالی به ترتیب شامل APIهای ()getIntent، ()getParcelableExtra و ()startActivity می‌باشد. در صورت وجود مسیری که دارای ترتیب توالی این APIها باشد، مسیر باید مورد تجزیه‌وتحلیل NIV قرار بگیرد. برای این کار، هر مسیر تحت تاثیر را با قوانین استنتاجی موجود در یک سیستم‌نوع[1] بررسی کرده و وجود NIV را تأیید می‌کند. در واقع توسط این قوانین، جریان داده مابین بلاک‌ها بررسی می‌شود.

نتایج: NIVD مجموعه ۱۰۰ برنامه محبوب اندروید را تجزیه‌وتحلیل کرده و NIV را در ۹ برنامه شناسایی کرده است. این رویکرد نسبت به ابزار NIVAnalyzer دارای دقت بالاتری است و به غیر از برنامه‌های آسیب‌پذیری که توسط NIVAnalyzer شناسایی‌شده، NIV را در برنامه GooglePhoto نیز گزارش کرده است[۱۴]. علاوه بر دقت، NIVD زمان اجرای سریع‌تری دارد و تعداد مسیرهای اجرایی کمتری را نسبت به NIVAnalyzer تولید و تحلیل می‌کند.

محدودیت‌ها: یکی از محدودیت‌های NIVD، تحلیل نکردن مؤلفه‌های خدمت و گیرنده اعلان می‌باشد. درست است که درصد وجود NIV در فعالیت‌ها بیشتر بوده اما احتمال وجود NIV در این مؤلفه‌ها نیز وجود دارد. همچنین NIVD همانند NIVAnalyzer، APIهای دیگری که قابلیت بازیابی Intentهای درونی را دارند را تحلیل نکرده است. از دیگر محدودیت‌های این ابزار می‌توان به تأخیر در تحلیل مسیرهای اجرایی اشاره کرد. این مورد هنگامی که یک برنامه دارای ترتیب توالی NIV باشد، خیلی چشمگیر است. به دلیل اینکه، NIVD یک بار مسیرهای اجرایی را برای بررسی ترتیب توالی APIهای مربوط به NIV تحلیل می‌کند و در صورت وجود، مجددا یک بار دیگر هر مسیر را جهت بررسی جریان داده مابین بلاک‌ها تحلیل خواهد کرد. به همین دلیل تحلیل دو بار هر مسیر، سرعت این ابزار را کند می‌کند؛ بنابراین زمان تحلیل هر برنامه برای استفاده در فروشگاه‌های برنامه‌های کاربردی جای بهبود بیشتری دارد.

سایر ابزارها .۴-۴

علاوه بر رویکردهای تشریح شده که به تشخیص NIV پرداخته‌اند، ابزارهای دیگری وجود دارند که توانایی شناسایی حمله‌های ناشی از NIV را در مؤلفه‌های خصوصی دارند. حمله‌های مانند تزریق داده‌های مخرب و نشت حریم خصوصی از جمله مواردی است که این ابزارها بطور کلی در تمام مؤلفه‌ها شناسایی می‌کنند. این حمله‌ها از طریق دریافت Next-Intent در مؤلفه خصوصی ممکن است رخ دهد. بنابراین در ادامه عملکرد این ابزارها به طور خلاصه تشریح می‌شود.

IntentSoot .۴-۴-۱

IntentSoot، یک ابزار مبتنی بر تجزیه‌وتحلیل ایستا و پویا است که جهت شناسایی آسیب‌پذیری‌های تزریق Intent مخرب طراحی شده است[۵]. IntentSoot با بررسی جریان Intentها در مؤلفه‌های عمومی و خصوصی هر برنامه قادر است حمله‌های نشت اطلاعات و افزایش مجوز را با دقت بالایی شناسایی کند. این ابزار برای انجام تحلیل خود از دو ماژول پویا و ایستا استفاده کرده است. ابتدا توسط ماژول پویا، هر برنامه را در سیستم اندروید اجرا می‌کند تا اطلاعات مربوط به

[1] Type System

فراخوانی‌های مؤلفه‌ها را به دست آورد. سپس نمودار فراخوانی بین‌رویه‌ای و درون‌رویه‌ای را با توجه به Intentهای دریافتی در هر مؤلفه تولید می‌کند. در ادامه با تعریف یک Intent دریافتی به عنوان منبع آلوده، جریان آن را به سمت APIهای حساس ردیابی می‌کند. بنابراین از این طریق حمله‌های تزریق داده مخرب ناشی از NIV را می‌تواند شناسایی کند. IntentSoot، برخلاف دقت بالا دارای زمان اجرای طولانی است و به دلیل سربار اجرای هر برنامه‌کاربردی و بررسی لاگ‌های تولیدشده نمی‌تواند به راحتی در مقیاس بالا و یا توسط فروشگاه‌های برنامه‌های کابردی جهت تست امنیت مورد استفاده قرار گیرد.

۴-۴-۲. ASPET

ASPET، یک ابزار تست امنیتی برای برنامه‌های اندروید است که با استفاده از تعریف الگوهای‌آسیب‌پذیر، قادر به شناسایی آسیب‌پذیری‌های مبتنی بر Intentها شده است[۶]. در واقع این الگوها شامل رفتارهای‌آسیب‌پذیر و مخرب در مؤلفه‌ها می‌باشد. ASPET بعد از تعریف الگوهای آسیب‌پذیر، برای هر برنامه نمودار ارتباط بین‌مؤلفه‌ای رسم می‌کند. از این طریق می‌تواند ویژگی‌های آسیب‌پذیر هر مؤلفه را جهت تولید نمونه‌های آزمایشی بدست آورد. سپس با اجرای نمونه‌های آزمایشی در شبیه‌ساز اندروید و تطابق هر نمونه با الگوهای آسیب‌پذیر، از امنیت برنامه مطلع می‌شود. ASPET در مقایسه با سایر ابزارهای تستی، بسیار مقیاس‌پذیر است؛ به دلیل اینکه با توجه به نیازمندی‌های جدید می‌تواند الگوهای‌آسیب‌پذیری متنوعی تولید کند. همچنین مدت زمان تست هر برنامه‌کاربردی نیز بسیار پایین است و اثربخشی بالایی در شناسایی آسیب‌پذیری‌های مبتنی بر Intentها دارد. در خصوص شناسایی NIV و حمله‌های آن نیز مستلزم تعریف الگوهای آسیب-پذیری مرتبط می‌باشد.

۴-۴-۳. AppIntent

AppIntent یکی از چارچوب‌های تشخیص نشت حریم‌خصوصی می‌باشد[۷]. این ابزار برخلاف سایر رویکردها، تنها انتقال داده‌های حساسی را گزارش می‌کند که موردنظر کاربر نبوده است. AppIntent، داده‌های ورودی و تعاملات کاربر را که منجر به انتقال می‌شوند، تولید می‌کند. سپس با اجرای کنترل‌شده برنامه به همراه ورودی‌ها و تعاملات کاربر، عملکردهای ممکن برنامه را شبیه‌سازی می‌کند. از این‌رو به تحلیلگران اجازه می‌دهد تا طبق دستکاری‌های GUI تصمیم بگیرند که کدام انتقال، قصد کاربر نمی‌باشد و نشت حریم‌خصوصی محسوب می‌شود. برای تجزیه‌وتحلیل برنامه‌ها، AppIntent از تکنیک اجرای نمادین هدایت شده با محدودیت رویداد-فضا استفاده می‌کند. تکنیک اجرای نمادین، یک روش تجزیه‌وتحلیل ایستا می‌باشد که رفتارهای برنامه را برای استخراج مسیرهای انتقال داده حساس جستجو می‌کند. همچنین در این روش جهت محدود کردن فضای جستجو و مسیرهای اجرایی از نمودار محدودیت رویداد-فضا استفاده شده است. این نمودار، کوتاه‌ترین مسیرهایی را در نظر می‌گیرد که ترتیب دستورالعمل‌های انتقال داده را پوشش می‌دهند. سپس برای درک بهتر تعاملات، به کمک یک محیط تست هر مسیر را اجرا می‌کند. این مورد باعث راحت‌تر شدن تصمیم‌گیری تحلیلگران می‌شود. AppIntent با روش پیشنهادی خود، توانسته مثبت‌های کاذب را در تشخیص نشت حریم‌خصوصی در رویکردهای مشابه کاهش دهد. از محدودیت‌های این ابزار می‌توان به عدم پشتیبانی از ورودی‌های شبکه و کدهای محلی در اجرای نمادین اشاره کرد. همچنین این ابزار قادر به شناسایی نشت اطلاعات در مؤلفه‌های خصوصی که ناشی از NIV هستند، می‌باشد اما توانایی شناسایی تزریق اطلاعات مخرب را در مؤلفه‌های خصوصی ندارد.

۴-۴-۴. FlowDroid

FlowDroid، یک رویکرد تجزیه‌وتحلیل ایستا با حساسیت بالا به متن، شیء و جریان برای بررسی دقیق برنامه‌های اندروید می‌باشد[۸]. این رویکرد، جریان داده‌های مؤلفه‌ها را جهت تشخیص نشت اطلاعات ردیابی می‌کند. این جریان داده می‌تواند اثر یک آسیب‌پذیری در برنامه خوش‌خیم باشد یا یک سوءاستفاده در برنامه مهاجم. این ابزار با مدل‌سازی دقیق چرخه‌حیات اندروید می‌تواند فراخوانی‌های مؤلفه‌ها را کنترل کند؛ بدین صورت که مسیرهای آلوده را در نمودار جریان کنترلی بین‌رویه‌ای از منبع‌های مشخص به سمت چاهک‌ها ردیابی می‌کند. با استفاده از این روش، FlowDroid توانسته با دقت بالای ۸۶٪، نشتی اطلاعات را در برنامه‌های آسیب‌پذیر و بدافزار پیدا کند. اما یکی از محدودیت‌های FlowDroid، عدم مدیریت فراخوانی Intentها بین مؤلفه‌ها می‌باشد. بنابراین اگر در کارهای آینده، این ابزار به همراه تحلیل NIVD مورد استفاده قرار گیرد علاوه بر یافتن NIV می‌تواند نشت‌های مؤلفه‌های خصوصی مورد هدف NIV را نیز شناسایی کند.

۴-۴-۵. AmanDroid

چارچوب AmanDroid با تجزیه‌وتحلیل ایستا، امنیت برنامه‌های اندروید را ارزیابی می‌کند[۹]. AmanDroid همانند FlowDroid، توانایی شناسایی نشت اطلاعات را به همراه تزریق داده‌مخرب و سوءاستفاده از APIهای حساس را دارد. رویکرد AmanDroid با تحلیل جریان داده بین‌مؤلفه‌ای، تمام ارتباطات یک برنامه را بررسی می‌کند. برای انجام این کار، از نمودار جریان داده‌ای بین‌مؤلفه‌ای و نمودار وابستگی‌داده استفاده می‌کند و به این صورت قادر به ردیابی اطلاعات از منابع به چاهک‌ها می‌شود. منابع شامل داده‌های‌حساس و یا ورودی‌های برنامه هستند و چاهک‌ها شامل متدها و APIهای که باعث ارتباط و نشتی داده می‌شوند مانند ارتباط با شبکه، بارگذاری یک URL، ارسال پیامک، برقراری تماس و غیره. AmanDroid میزان دقت بالاتری نسبت به FlowDroid دارد اما مدت زمان تحلیل آن کندتر است. همچنین فراخوانی Intentها را نیز بین مؤلفه‌ها ردیابی و بررسی می‌کند. بنابراین قادر به شناسایی حمله تزریق داده مخرب از مؤلفه‌های عمومی به سمت مؤلفه‌های خصوصی می‌باشد.

۴-۴-۶. AsDroid

رویکرد AsDroid از طریق تحلیل ایستا قادر به شناسایی برنامه‌های اندرویدی که دارای رفتارهای‌مخرب پنهانی هستند، می‌باشد[۱۰]. رفتارهای پنهانی مورد تحلیل AsDroid شامل: ارسال پیامک، برقراری تماس تلفنی، اتصالات HTTP و نصب برنامه‌های مخرب می‌باشد. بنابراین برنامه‌های که دارای مجوز انجام این عملیات هستند، مورد تحلیل AsDroid قرار می‌گیرند. ابزار AsDroid شامل دو بخش می‌باشد: ۱- مؤلفه تجزیه‌وتحلیل برنامه: در این مؤلفه، APIهای اندروید دسته‌بندی می‌شوند. هر دسته را به انواع Intentهایی با اهداف خاص نسبت می‌دهد مانند Intent با هدف ارسال پیامک. سپس جریان هر Intent را تا رسیدن به فراخوانی یک تابع سطح بالا در نمودارهای کنترل جریان و وابستگی، تجزیه‌وتحلیل می‌کند. ۲- مؤلفه تجزیه‌وتحلیل رابط‌کاربری: این مؤلفه وظیفه نسبت دادن متن UIهای ساختگی به یک تابع سطح بالا می‌باشد. بنابراین با بررسی سازگاری موارد استخراج‌شده از هر دو مؤلفه، رفتارهای مخرب را شناسایی می‌کند. اثر AsDroid بسیار مشابه AppIntent می‌باشد. از این‌رو، قادر به شناسایی نشت اطلاعات حساس از طریق رفتارهای مخرب پنهانی می‌باشد. AsDroid به دلیل تحلیل UIهای ساختگی، دارای هشدارهای کاذب است و همچنین میزان فضا و زمان

اجرا زیادی را مصرف می‌کند. در خصوص NIV نیز تنها قادر به ردیابی Intentها از مؤلفه‌های خصوصی به سمت نشت‌های مانند ارسال پیامک و برقراری تماس می‌باشد و APIهای حساس دیگری را در نظر نمی‌گیرد[۱۲].

۵. مقایسه رویکردهای موجود

با توجه به رویکردهای تشریح شده، این بخش به مقایسه‌ی این رویکردها می‌پردازد. طبق شاخص‌های جدول ۱ این رویکردها از نظر نوع تحلیل، ایده اصلی و محدودیت‌های آنها مورد مقایسه قرار گرفته‌اند.

Morbs به عنوان اولین رویکرد حفاظت از مبدأهای غیرمجاز، اثربخشی مؤثری در جلوگیری از NIV داشته است. اما به دلیل سربار اصلاحات سطح سیستم‌عامل و عدم مقیاس‌پذیری برای سایر برنامه‌های کاربردی، مورد استفاده قرار نگرفته است. در عوض ابزارهای NIVAnalyzer و NIVD در مقیاس بالایی از برنامه‌ها، NIV را شناسایی می‌کنند. اما از نظر زمان اجرا، NIVD به دلیل تولید مسیرها با شروع از مؤلفه‌های عمومی و تحلیل روبه‌جلو، توانسته تعداد مسیرهای کمتری را نسبت به NIVAnalyzer تولید کند. از نظر دقت نیز، NIVAnalyzer به دلیل تعیین حد آستانه جهت توقف تولید مسیرها، دارای هشدارهای منفی‌کاذب می‌باشد.

سایر ابزارها مانند IntentSoot، APSET، AppIntent، FlowDroid، AmanDroid و AsDroid بر روی شناسایی مؤلفه‌های تحت تأثیر NIV تمرکز نکرده‌اند؛ اما بطور خاص اثرات نشت اطلاعات، تزریق داده مخرب و افزایش مجوز را از طریق NIV در مؤلفه‌ها شناسایی می‌کنند. بنابراین اگر در مؤلفه‌های خصوصی موجود در برنامه‌های تحت تأثیر NIV، Intentهای دریافتی به سمت چاهک‌ها و APIهای حساس ارسال شوند، آنها را گزارش می‌کنند. بنابراین می‌توان این ابزارها را به همراه تحلیلگرهای NIVAnalyzer و NIVD برای شناسایی دقیق‌تر NIV به همراه نوع حمله‌های آن، پیاده‌سازی کرد.

۶. چالش‌ها و سیر تحقیقات آتی

همانطور که در بخش ۳-۳ مطالعه کردید، پیامدهای NIV نسبت به قابلیت‌های مؤلفه‌های خصوصی متنوع می‌باشد. در سال ۲۰۱۳ فقط یک نوع از این آسیب‌پذیری وجود داشت در حالی که در سال ۲۰۱۸ توسط NIVAnalyzer، ۱۷ نوع مختلف NIV در برنامه‌های آسیب‌پذیر یافت شد. همچنان این آسیب‌پذیری یک موضوع باز است زیرا تحقیقات کمی در مورد آسیب‌پذیری‌های مرتبط با NIV انجام‌شده است. سازوکارهای تشخیص NIV موجود نیز تنها قادر به شناسایی NIV می‌باشند و راهکار دفاعی در برابر این تهدید ارائه نمی‌دهند. در واقع این آسیب‌پذیری یک چالش ذاتی در اندروید محسوب می‌شود و رفع آن بدون پشتیبانی در سطح سیستم‌عامل، بسیار دشوار است. برای مثال هنگامی که Morbs، NIV را در Dropbox گزارش کرد، توسعه‌دهندگان آن تصمیم به تغییر معماری برنامه کردند. همچنین توسعه‌دهندگان Facebook نیز راه‌حلی برای رفع این مشکل پیدا نکردند؛ چرا که بدون استفاده از قابلیت Next-Intent، خیلی از عملکردهای برنامه قابل انجام نبود. اما رویکرد Morbs با تعریف سیاست‌های موردنظر در سیستم‌عامل برای هر برنامه، توانست مشکل NIV را در آنها رفع نماید. اما همچنان طبق گزارش NIVD، این مشکل در برنامه‌های دیگر موجود است[۱۴].

جدول ۱- مقایسه رویکردهای تحلیلگر NIV

رویکرد	نوع تحلیل	ایده	محدودیت
Morbs	نظارت در زمان اجرا	جلوگیری از رخ دادن NIV با بررسی مؤلفه‌های مبدأ و مقصد هر پیام	سربار اصلاحات سطح سیستم‌عامل، عدم مقیاس-پذیری
NIV Analyzer	تحلیل ایستا و پویا	کشف NIV در تمام مؤلفه‌ها از طریق تحلیل مسیرهای تحت تاثیر، بهره‌برداری از هر مسیر با تولید موارد آزمایشی مخرب جهت تأیید NIV	سرعت و دقت پایین در تحلیل برنامه‌ها، تولید منفی‌کاذب
NIVD	تحلیل ایستا	تشخیص ترتیب توالی فراخوانی‌های NIV، تأیید NIV با استفاده از قوانین استنتاجی	عدم شناسایی NIV در مؤلفه-های خدمت و گیرنده اعلان، سربار تأخیر در تحلیل
Intent Soot	تحلیل ایستا و پویا	شناسایی تزریق Intent مخرب به وسیله ردیابی Intentهای دریافتی در مؤلفه‌های عمومی و خصوصی به سمت APIهای-حساس	سرعت پایین در تحلیل برنامه-ها، سربار اجرای هر برنامه و بررسی لاگ‌های‌اجرایی
APSET	تحلیل ایستا	تست امنیت و شناسایی آسیب‌پذیری‌های مبتنی بر Intent به وسیله تعریف الگوهای آسیب‌پذیر و اجرای موارد آزمایشی بر روی مؤلفه‌ها	کارایی پایین، پیچیدگی تعریف الگوهای‌آسیب‌پذیر
AppIntent	تحلیل ایستا	تشخیص نشت حریم خصوصی با بررسی انتقال داده‌های-حساس بدون قصد کاربر، استفاده از تکنیک اجرای نمادین جهت استخراج رفتار مؤلفه‌ها	عدم پشتیبانی از ورودی‌های شبکه، عدم شناسایی تزریق Intentهای‌دریافتی
FlowDroid	تحلیل ایستا	مدل‌سازی چرخه حیات اندروید، تشخیص نشت داده با ردیابی از یک منبع به سمت چاهک، حساس به جریان، شیء و داده	تخمین ارتباطات بین‌مؤلفه‌ای و عدم دقت لازم در تحلیل Intentها
Amandroid	تحلیل ایستا	شناسایی جریان داده آلوده، مدل‌سازی تعاملات برنامه با سیستم اندروید، مدیریت جریان کنترل و جریان داده بین مؤلفه‌ها از منابع به سمت چاهک‌ها	سرعت پایین در تحلیل هر برنامه
AsDroid	تحلیل ایستا	تشخیص رفتارهای پنهانی مخرب در برنامه‌ها با ردیابی Intentها به سمت APIهای حساس، تجزیه‌وتحلیل در نمودارهای کنترل جریان و نمودار وابستگی	محدودیت در فضا و زمان اجرا، گزارش هشدارهای کاذب

بنابراین یک روش مؤثر جهت پیشگیری از NIV، مجهز کردن سیستم‌عامل اندروید به سیستم ردیابی ذره‌ای است که بتواند جریان داده‌ها را بطور کامل تحلیل و NIV را تشخیص دهد. پس به عنوان یکی از کارهای آینده، می‌توان ابزارهای تحلیلگر NIV را با ابزارهای ردیابی مثل TaintDroid[16] در چارچوب اندروید ادغام کرد. این کار شامل برقراری ارتباط بین NIV و سایر آسیب‌پذیری‌های مبتنی بر Intent می‌باشد. همچنین از دیگر کارهای آینده می‌توان به بهبود تحلیلگرهای NIV موجود، جهت افزایش دقت و سرعت در تحلیل برنامه‌ها و رفع محدودیت آنها اشاره کرد.

۷. نتیجه گیری

مدل ارتباطی اندروید ضمن کاربردهای فراوان، سبب شده تهدیدهایی برای برنامه‌های کاربردی به وجود آید که منجر به حمله‌های مختلفی می‌شود. یکی از این تهدیدها، ارتباط برنامه مخرب با مؤلفه‌های خصوصی برنامه‌ها می‌باشد که از طریق NIV رخ می‌دهد. برنامه‌های مخرب با استفاده از این آسیب‌پذیری، مؤلفه‌های عمومی را دور می‌زنند و مؤلفه‌های خصوصی را مورد هدف قرار می‌دهند. از آنجایی که مؤلفه‌های خصوصی دارای حساسیت بالایی هستند و دسترسی به آنها منجر به حمله‌های جدی می‌شود، تشخیص NIV و جلوگیری از آن، اهمیت ویژه‌ای دارد. ازاین‌رو، در این بررسی سعی شد مفاهیم مربوط به Next-Intent، آسیب‌پذیری آن، پیامدهای ناشی از NIV و همچنین رویکردهای تحلیلگر NIV به همراه مزایا و معایب آنها مطرح گردد. بنابراین با مطالعه این مقاله، توسعه‌دهندگان اندروید می‌توانند با دقت بالاتری از Next-Intent در برنامه‌های کاربردی استفاده کنند. همچنین به محققان این امکان را می‌دهد تا در صدد رفع چالش‌های NIV، تحقیقات خود را پیش برده و به تولید و بهبود رویکردهای مطرح‌شده بپردازند.

۸. مراجع

[1] B. Rashidi and C. J. Fung, “A Survey of Android Security Threats and Defenses,” *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.*, vol. 6, pp. 3-35, 2015.
[2] V. J. Mozos Perez, “A study of vulnerabilities on Android Systems,” pp.10-20, 2013.
[3] E. Chin, A. P. Felt, K. Greenwood, and D. Wagner, “Analyzing inter-application communication in Android” *presented at the Proceedings of the 9th international conference on Mobile systems, applications, and services*, Bethesda, Maryland, USA, 2011.
[4] D. Octeau *et al.*, “Effective inter-component communication mapping in Android with Epicc: an essential step towards holistic security analysis” *presented at the Proceedings of the 22nd USENIX conference on Security*, Washington, D.C., 2013.
[5] B. Xiong, G. Xiang, T. Du, J. He, and S. Ji, “Static Taint Analysis Method for Intent Injection Vulnerability in Android Applications” *Cham, Springer International Publishing, in Cyberspace Safety and Security*, pp. 16-31, 2017.
[6] S. Salva and S. R. Zafimiharisoa, “APSET, an Android aPplication SEcurity Testing tool for detecting intent-based vulnerabilities” *International Journal on Software Tools for Technology Transfer* ,vol. 17, no. 2, pp. 201-221, 2015.
[7] Z. Yang, M. Yang, Y. Zhang, G. Gu, P. Ning, and X. S. Wang, “AppIntent: analyzing sensitive data transmission in android for privacy leakage detection” *presented at the Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, Berlin, Germany, 2013.

[8] S. Arzt *et al.*, “FlowDroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps” *presented at the Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation, Edinburgh, United Kingdom*, 2014.

[9] F. Wei, S. Roy, and X. O. Robby,“Amandroid: A Precise and General Inter-component Data Flow Analysis Framework for Security Vetting of Android Apps” *presented at the Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, Scottsdale, Arizona, USA, 2014.

[10] J. Huang, X. Zhang, L. Tan, P. Wang, and B. Liang, "*AsDroid: detecting stealthy behaviors in Android applications by user interface and program behavior contradiction*," presented at the Proceedings of the 36th International Conference on Software Engineering, Hyderabad, India, 2014.

[11] P. Gadient, M. Ghafari, P. Frischknecht, and O. Nierstrasz, “Security code smells in Android ICC” *Empirical Software Engineering*, pp. 1-31, 2018.

[12] “Intents and Intent filters.” https://developer.android.com/guide/components/intents-filters.html.

[13] R. Wang, L. Xing, X. Wang, and S. Chen, “Unauthorized origin crossing on mobile platforms: threats and mitigation” *presented at the Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, Berlin, Germany, 2013.

[14] M. A. El-Zawawy, E. Losiouk, and M. Conti, “Do not let Next-Intent Vulnerability be your next nightmare: type system-based approach to detect it in Android apps” *International Journal of Information Security*, vol. 20, no. 1, pp. 39-58, 2021.

[15] J. Tang *et al.*, “NIVAnalyzer: A Tool for Automatically Detecting and Verifying Next-Intent Vulnerabilities in Android Apps” in *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)*, pp. 492-499, 2017.

[16] W. Enck *et al.*,“TaintDroid :An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones” *Transactions on Computer Systems*, vol. 32, no. 2, pp. 1–29, 2014.