دانشگاه صنعتی شریف
دانشکده مهندسی کامپیوتر

پایان‌نامه کارشناسی ارشد
مهندسی رایانش امن

بهبود کارایی روش‌های تشخیص برنامک‌های اندرویدی بازبسته‌بندی شده

نگارش

مجتبی موذن

استاد راهنما

دکتر مرتضی امینی

بهمن ۱۴۰۱

به نام خدا

دانشگاه صنعتی شریف

دانشکده مهندسی کامپیوتر

پایان‌نامه کارشناسی ارشد

این پایان‌نامه به عنوان تحقق بخشی از شرایط دریافت درجه کارشناسی ارشد است.

عنوان: بهبود کارایی روش‌های تشخیص برنامک‌های اندرویدی بازبسته‌بندی شده

نگارش: مجتبی موذن

کمیته ممتحنین

استاد راهنما: دکتر مرتضی امینی امضاء:

استاد مشاور: استاد مشاور امضاء:

استاد مدعو: استاد ممتحن امضاء:

تاریخ:

سپاس

از استاد بزرگوارم که با کمک‌ها و راهنمایی‌های بی‌دریغشان، مرا در به سرانجام رساندن این پایان‌نامه یاری داده‌اند، تشکر و قدردانی می‌کنم. همچنین از همکاران عزیزی که با راهنمایی‌های خود در بهبود نگارش این نوشتار سهیم بوده‌اند، صمیمانه سپاسگزارم.

چکیده

با گسترش روزافزون استفاده از برنامک‌های اندرویدی در سالیان اخیر حملات موجود بر روی این سیستم‌عامل با افزایش قابل توجهی همراه بوده‌است. متن‌باز بودن برنامک‌های اندرویدی و در نتیجه، دسترسی به کد منبع این دسته از برنامک‌ها، در کنار افزایش حملات بر روی آن‌ها ، لزوم توجه به مقابله با حملات مطروحه در این زمینه را افزایش داده‌است.حملات بازبسته‌بندی روی برنامک‌های اندرویدی، نوعی از حملات هستند که در آن مهاجم، پس از دسترسی به کد منبع برنامک و کپی‌کردن آن و یا ایجاد تغییراتی که مدنظر مهاجم است، مجدداً آن‌را بازبسته‌بندی می‌کند.تغییر کدهای برنامک، اهداف متفاوتی نظیر تغییر کتابخانه‌های تبلیغاتی، نقض امنیت کاربر و یا ضربه به شرکت‌های تولید برنامک از تغییر گسترش برنامک‌های جعلی را دنبال می‌کند. بازبسته‌بندی برنامک‌های اندرویدی علاوه بر ماهیت تهدید کاربران و شرکت‌ها، ماهیتی پیشگیرانه نیز دارد. در این حالت توسعه‌دهندگان نرم‌افزار از طریق ایجاد مبهم‌نگاری در برنامک‌های اندرویدی، سعی در پیشگیری از بازبسته‌بندی به وسیله‌ی مهاجمان دارند. تشخیص بازبسته‌بندی در برنامک‌های اندرویدی از آن‌جهت دارای اهمیت است که هم کاربران و هم شرکت‌های توسعه‌دهنده، می‌توانند از این موضوع ذی‌نفع،باشند. تشخیص برنامک‌های بازبسته‌بندی شده، به جهت چالش‌های پیش‌رو، نظیر مبهم‌نگاری کدهای برنامک جعلی به دست مهاجم و همچنین تشخیص و جداسازی صحیح کدهای کتابخانه‌ای مسئله‌ای چالشی محسوب می‌شود. پژوهش‌های اخیر در این زمینه به صورت کلی، از روش‌های تشخیص مبتنی بر شباهت‌سنجی کدهای برنامک و یا طبقه‌بندی برنامک‌های موجود استفاده کرده‌اند. از طرفی برقراری حدواسطی میان سرعت و دقت در تشخیص برنامک‌های جعلی، چالشی است که استفاده از این دست پژوهش‌ها را در یک محیط صنعتی ناممکن ساخته‌است. در این پژوهش پس‌از استخراج کدهای برنامک به وسیله‌ی چارچوب سوت و ابزارهای دیس‌اسمبل، در یک روش دو مرحله‌ای کدهای برنامک‌های موجود با یکدیگر مقایسه می‌شود. پس از دیس‌اسمبل کدهای هر برنامک، در طی یک فرایند طبقه‌بندی مبتنی بر ویژگی‌های انتزاعی و دیداری، برنامک‌های کاندید برای هر برنامک مبدا استخراج می‌شود. سپس برای هر کلاس برنامک اندرویدی، امضایی متشکل از مهم‌تری ویژگی‌های کدپایه از آن استخراج و پس از انجام مقایسه با کلاس‌های کتابخانه‌های اندرویدی موجود در مخزن، کتابخانه‌های اندرویدی حذف می‌شوند و در نهایت با مقایسه‌ی کدهای اصلی، برنامک بازبسته‌بندی شده مشخص می‌شود. در قسمت آزمون روش پیشنهادی در این پژوهش، توانستیم روش موجود در این زمینه‌را با بهبود امضای تولیدشده از هر برنامک و اضافه‌شدن مرحله‌ی پیش‌پردازش، سرعت تشخیص را ۴ برابر افزایش داده و در عین‌حال دقت روش موجود را نیز حفظ کنیم.

کلیدواژه‌ها: پایان‌نامه، حروف‌چینی، قالب، زی‌پرشین

فهرست مطالب

فهرست جدول‌ها

فهرست شکل‌ها

فصل ۱

مقدمه

سیستم‌عامل اندروید به دلیل سهولت در توسعه توسط توسعه‌دهندگان موبایلی و در نتیجه فراوانی استفاده از آن در تلفن‌های همراه، تلوزیون‌های هوشمند و دیگر دستگاه‌های موجود، حجم بالایی از بازار سیستم‌عامل‌های موبایلی را به خود اختصاص داده‌است. بر طبق گزارش پایگاه استاتیستا [۲] سیستم‌عامل اندروید سهمی معادل ۷۱ درصدی از سیستم‌عامل‌های موبایلی را در سه ماهه‌ی پایانی سال ۲۰۲۲ به خود اختصاص داده‌است. در سال‌های اخیر به دلیل گسترش استفاده از این پلتفرم، فروشگاه‌های اندرویدی زیادی به جهت ارائه‌ی خدمات به کاربران به وجود آمده‌است. برخی از فروشگاه‌های رسمی مانند فروشگاه اندرویدی گوگل، از ابزارهایی نظیر پلی پروتکت[۳] برای بررسی برنامک‌های اندرویدی موجود در فروشگاّه استفاده می‌کنند. علاوه بر این، در سال‌های اخیر فروشگآه‌های متعدد رایگانه به وجود آمده‌اند که صرفا برنامک‌های اندرویدی موجود در سطح وب را غربان و آن را به کاربران ارائه می‌دهند. فروشگاه‌های رایگان غالباً ابزارهای مشخصی را برای حفظ امنیت کاربران استفاده نمی‌کنند و امنیت کاربران این دسته از فروشگاه‌های اندرویدی، ، همواره تهدید می‌شود یکی از راه‌های مورد استفاده توسط مهاجمان برای وارد ساختن بدافزار به تلفن‌های همراه، بازبسته‌بندی نرم‌افزار است. مطابق تعریف، بازبسته‌بندی شامل دانلود یک برنامک، دسترسی به محتوای کدهای برنامک اصلی از طریق روش‌های مهندسی معکوس و در نهایت بازبسته‌بندی به همراه تغییر و یا بدون تغییردادن کدهای برنامک اصلی است. زبان اصلی توسعه در برنامک‌های اندرویدی، زبان جاوا می‌باشد که یک زبان سطح بالا محسوب می‌شود. در طی فرآیند کامپایل برنامک‌های اندرویدی، مجموعه‌ی کدهای منبع در طی فرایندی به بایت‌کدهای دالویک تبدیل می‌شوند و در ادامه ماشین مجازی جاوا، بایت‌کدها را برروی ماشین مقصد اجرا می‌کند[۴]. فهم و در نتیجه مهندسی معکوس زبان میانی دالویک بایت‌کدها آسان است و به همین علت موجب سهولت در بازبسته‌بندی برنامک‌های اندرویدی می‌شود.

به طور کلی بازبسته‌بندی را می‌توان از دو جهت مورد بررسی قرار داد، از دید توسعه‌دهندگان، بازبسته‌بندی

شامل فرآیندی است که توسعه‌دهنده با انجام مبهم‌نگاری در برنامک مورد توسعه، فهم بدنه‌ی اصلی برنامک را برای مهاجم سخت می‌کند. از این دید، بازبسته‌بندی یک روش تدافعی تلقی می‌شود تا مهاجم پس از دسترسی به کد برنامک اصلی، نتواند بدنه‌ی برنامک اصلی برنامک را شناسایی و در نتیجه آن‌را تغییر دهد. از جهت دیگر، بازبسته‌بندی توسط فردی که برنامک متعلق به او نیست یک عمل تهاجمی محسوب می‌شود. در این حالت، مهاجم پس از دسترسی به کد برنامک‌اصلی، بسته به هدف او، برنامک را مجددا بازبسته‌بندی می‌کند و آن را در فروشگاه‌های اندرویدی خصوصا فروشگاه‌هایی که نظارت کمتری بر روی آن‌ها وجود دارد منتشر می‌کند. در این حالت مهاجم به جهت اهدافی متفاوتی نظیر تغییر کدهای تبلیغاتی در برنامک اصلی، تغییر درگاه‌های پرداخت و یا بازپخش بدافزار، اقدام به بازبسته‌بندی می‌کند. بازبسته‌بندی یکی از راه‌های محبوب مهاجمان برای انتقال بدافزارهای توسعه‌داده‌شده به تلفن همراه قربانی است[۵]. مطابق پژوهش آقای ژو و همکاران[۶] حدود۸۵ درصد بدافزارهای موجود، از طریق بازبسته‌بندی منتشر می‌شوند. همانطور که گفته شد، برخی فروشگآهای اندرویدی نظیر گوگل، سازوکار مشخصی را برای تشخیص بازبسته‌بندی ارائه‌داده‌اند اما بسیاری از فروشکاه‌های اندرویدی فعال و پربازدید، خصوصا فروشگاه‌های رایگان، یا از هیچ ابزاری استفاده نمی‌کنند و یا در صورت توسعه‌ی نرم‌افزار بومی خود برای شناسایی برنامک‌های بازبسته‌بندی شده، مشخصات و یا دقت آن را گزارش نکرده‌اند.

همانطور که اشاره‌شد، محبوبیت و در نهایت استفاده‌ی زیاد برنامک‌های اندرویدی و همچنین نظارت کم در فروشگاه‌های اندرویدی، بازبسته‌بندی، یک روش پر استفاده به جهت انتقال بدافزار به تلفن همراه کاربران است. آقای خانمحمدی و همکاران[۷]، پس از بررسی پایگاه‌داده‌ای برنامک‌های اندرویدی اندروزو، دریافتند که ٪۵۲/۲۲ از برنامک‌های موجود در این مخزن توسط ویروس‌توتال، بدافزار شناسایی شده‌اند. ویروس‌توتال، ابزاری متشکل از ۳۰ ضدبدافزار برای بررسی یک برنامک اندرویدی اندرویدی است. مطابق این پژوهش، ٪۷۷/۸۴ از برنامک‌های این مخزن‌داده که بازبسته‌بندی شده‌اند، دارای نوعی از بدافزار ضدتبلیغاتی بوده‌اند که موجب می‌شود تبلیغات موجود در برنامک تغیر و اهداف مالی و امنیتی کاربران و توسعه‌دهندگان مخدوش شود . علاوه بر این، مطابق پژوهشی که توسط ویداس وهمکاران[۸] انجام شده‌است، پس از پیاده‌سازی ۷ روش پربازدید به جهت تشخیص بازبسته‌بندی، در بهترین حالت، روش‌های موجود قادر به تشخیص ٪۷۲/۲۲ از برنامک‌های بازبسته‌بندی شده‌ی سه فروشگاه مطرح اندرویدی است. بنابراین مشخص است که تشخیص برنامک‌های بازبسته‌بندی شده، به چه میزان می‌تواند اهداف مالی و امنیتی توسعه‌دهندگان و کاربران برنامک‌ها را ارضا کند. در سال‌های اخیر ارائه‌ی یک راهکار پرسرعت به همراه دقت مناسب، همواره یکی از دغدغه‌های مهم پژوهش‌کنندگان در این زمینه بوده‌است.

همانطور که گفته‌شد، بازبسته‌بندی برنامک‌های اندرویدی از دو دیدگاه تهاجمی و تدافعی قابل بررسی است. در حالتی که کاربر متقلب، برنامک اندرویدی اصلی را دجار تغییراتی می‌کند و آن را در اختیار عموم قرار می‌دهد، تشخیص بازبسته‌بندی، با استفاده از مقایسه‌ی برنامک اصلی و برنامک جعلی صورت می‌گیرد.

تشخیص بازبسته‌بندی در این حالت را می‌تواند در حالت کلی به دو طبقه تقسیم کرد. در حالت اول توسعه‌دهنده روش خود را مبتنی بر تحلیل برنامک مبدا و مقصد پیاده‌سازی می‌کند. عمده‌ی روش‌های موجود در این طبقه مبتنی بر تحلیل ایستای جفت برنامک‌ها است و استفاده از تحلیل پویا به جهت سرعت پایین آن، محبوبیت فراوانی ندارد و بیشتر از تحلیل ایستای برنامک‌های اندرویدی استفاده می‌شود[۹]. در سمت دیگر طبقه‌بندی برنامک‌های اندرویدی وجود دارد.روش‌های موجود در این دسته، عمدتا سرعت بالایی دارند اما در تشخیص جفت بازبسته‌بندی شده دقت پایینی را ارائه می‌دهند.

برنامک‌های اندرویدی متشکل از دو قسمت اصلی کدهای برنامک و منابع آن هستند. کدهای برنامک، منطق برنامک را تشکیل می‌دهند و رفتار برنامک با توجه به این قسمت مشخص می‌شود. از طرفی منابع برنامک، ظاهر آن را تشکیل می‌دهند. روش‌های مبتنی بر تحلیل برنامک و یا طبقه‌بندی آن، عمدتاً از ویژگی‌های موجود در منابع و یا کد استفاده می‌کنند. مهاجم در حالاتی که می‌خواهد از محبوبیت برنامک مبدا استفاده کند، سعی در یکسان‌سازی ظاهر برنامک‌های مبدا و مقصد دارد به همین جهت از منابع برنامک مبدا استفاده می‌کند و منطق برنامک را مطابق با اهداف خود تغییر می‌دهد. در حالتی دیگر، متقلب سعی می‌کند که با استفاده از تغییر منابع برنامک و تولید یک برنامک تقلبی و گاهاً بدون هیچ تغییری در کد برنامک، ادعای توسعه‌ی یک برنامک جدید را اثبات کند. لازم به ذکر است استفاده از ویژگی‌های کدپایه و منبع‌پایه، به وفور در پژوهش‌های سال‌های اخیر یافت می‌شود که هر کدام معایب و مزایای خود را دارد.

در روش‌های مبتنی بر طبقه‌بندی عمدتا تعریف تشخیص بازبسته‌بندی محدود به تشخیص دسته‌ی مشکوک و یا دسته‌ای از برنامک‌ها است که احتمال بازبسته‌بندی بودن جفت‌های داخل این دسته، بیش از سایر دسته‌ها است. بنابراین تشخیص بازبسته‌بندی در این روش‌ها، محدود به تشخیص طبقه‌ی برنامک ورودی می‌باشد و جفت بازبسته‌بندی شده مشخص نمی‌شود. از طرفی در روش‌های مبتنی بر تحلیل ایستا، بررسی دوبه‌دوی برنامک‌های ورودی و مجموعه‌داده مدنظر است. در این روش‌ها تعریف تشخیص بازبسته‌بندی گسترش یافته و یافتن جفت بازبسته‌بندی به صورت مشخیص، هدف پژوهش می‌شود. تغییر منابع برنامک و همچنین مبهم‌نگاری در برنامک بازبسته‌بندی شده، دوجالش مهم در راستای تشخیص بازبسته‌بندی است. متقلب پس از بازبسته‌بندی برنامک، با استفاده از مبهم نگاری سعی می‌کند تغیرات خود و شباهت ساختار منطقی برنامک تقلبی با برنامک اصلی را پنهان کند. به همین جهت، تشخیص بازبسته‌بندی نیازمند ویژگی‌هایی است که مقاومت بالایی مقابل مبهم‌نگاری داشته‌باشد بدین معنا که تغییر و ایجاد ابهام در کد، به راحتی در این ویژگی‌ها قابل انجام نباشد.

در هنگار کامپایل برنامک‌های اندرویدی، کتابخانه‌هایی که در برنامک مورد استفاده قرار گرفته‌اند به همراه کد مورد توسعه، کامپایل شده و دالویک بایت‌کدهای آن در کنار برنامک قرار می‌گیرد. بر اساس پژوهش آقای زیانگ و همکاران [۱۰] ٪۵۷ از کدهای برنامک‌های مورد بررسی در این پژوهش، شامل کدهای کتابخانه‌ای بودند که دجار مبهم‌نگاری نشده‌اند. بنابراین تشخیص کدهای بازبسته‌بندی شده بدون تشخیص

درست و دقیق و جداسازی کدهای کتابخانه‌ای امکان‌پذیر نیست و می‌تواند نتایج منفی غلط و مثبت غلط را کاهش دهد. به صورت کلی دو روش برای تشخیص کدهای کتابخانه‌ای استفاده می‌شود، روش مبتنی بر لیست سفید و یا روش تشخیص مبتنی بر شباهت سنجی. در روش لیست سفید، لیسیتی از مشهور ترین کتابخانه‌های موجود در مخازن کتابخانه‌های اندرویدی نظیر ماون را جمع آوری می‌شود و با استفاده از نام کلاس‌ها و بسته‌های موجود کلاس‌های کتابخانه ای تشخیص‌داده می‌شود. مشخص است که این روش مقاومت بسیار کمی مقابل ساده‌ترین روش‌های مبهم‌نگاری در کتابخانه‌های اندرویدی دارد. در حالت دیگر از روش‌های مبتنی بر شباهت سنجی برای تشخیص کدهای کتابخانه‌ای استفاده می‌شود که در این روش، تحلیل ایستا روی کدهای برنامک مبدا و مخزن کتابخانه‌های اندرویدی صورت می‌گیرد و در نهایت با یکدیگر مقایسه می‌شوند. مشخص است که روش‌های مبتنی بر شباهت‌سنجی از دقت بیشتری برخوردار هستند و تمایز بهتری میان کدهای کتابخانه‌ای و کدهای اصلی قرار می‌دهند اما اینگونه روش‌ها سرعت پایینی دارند.

پژوهش‌های ارائه‌شده در زمینه‌ی تشخیص برنامک‌های بازبسته‌بندی شده در سال‌های اخیر، عمدتاً در تلاش برای بهبود دقت و سرعت روش‌های پیشین بوده‌اند.مبهم‌نگاری باعث می‌شود که دقت روش‌های تشخیص مبتنی بر تحلیل ایستا و شباهت سنجی پایین بیاید و استفاده از ویژگی‌هایی را که مقاومت بالایی مقابل مبهم‌نگاری داشته باشند را واجب کند. از طرفی استفاده از ویزگی‌های مقاوم به مبهم‌نگاری، می‌تواند سرعت تشخیص را بسیار پایین آورده تا حدی که عملا استفاده از این روش‌ها در یک محیط صنعتی را غیر ممکن سازد. در این پژوهش ما با استفاده از ترکیب روش‌های تحلیل ایستا و طبقه‌بندی منابع، به همراه شباهت‌سنجی، روشی را ارائه کرده‌ایم که در حالی که مقاومت بالایی نسبت به مبهم‌نگاری داشته‌باشد، سرعت روش‌های پیشین را نیز افزایش دهد. در این پژوهش به عنوان پیش‌پردازش، از یک طبقه‌بند نزدیک‌ترین همسایه برای کاهش فضای مقایسه‌ی دودویی و با استفاده از ویژگی‌های مبتنی بر منبع، استفاده شده‌است. با کاهش فضای مقایسه‌ی دودویی و طبقه‌بندی برنامک‌های مشکوک در یک دسته، مقایسه‌ی برنامک‌های موجود در آن دسته آغاز می‌شود. مقایسه‌ی دودویی در هر دسته مبتنی بر تحلیل ایستا و شباهت‌سنجی کدهای برنامک انجام می‌شود. ابتدا ویژگی‌هایی از هر کلاس و متد در بسته‌های برنامک استخراج شده و امضای هر کلاس ساخته می‌شود به طوری که امضای هر کلاس منحصر به فرد و تا حد امکان مختص همان کلاس باشد. نوآوری روش مطروحه، ترکیب روش‌های مبتنی بر طبقه‌بندی و روش‌های مبتنی بر تحلیل می‌باشد که در نهایت منجر به افزایش سرعت و در عین حال دقت خوب در تشخیص برنامک‌های بازبسته‌بندی شده‌است. حذف کدهای کتابخانه‌ای با استفاده از روشی مبتنی بر مقایسه‌ی کدهای موجود در مخزن کتابخانه‌ها و کلاس‌های برنامک انجام می‌شود. مخزن کتابخانه‌ها متشکل از ۴۵۳ کتابخانه‌ی اندرویدی جمع‌آوری شده از مخزن ماون مب‌باشد. در نهایت پس از تشخیص کلاس‌های کتابخانه‌های اندرویدی و حذف آن‌ها از کد برنامک، کدهای مورد توسعه به عنوان ورودی برای

مقایسه‌ی دودویی مورد تحلیلی قرار می‌گیرند.

در ادامه‌ی این نگارش، در فصل ۲ به بررسی و تعریف مفاهیم اولیه‌ی مورد نیاز در این پژوهش می‌پردازیم. در فصل ؟؟ به تعریف مسئله می‌پردازیم و همچنین مروری از کارهای پیشین را خواهیم داشت. در ادامه و در فصل ۴ روش مورد استفاده در این پژوهش، شرح داده خواهدشد و در فصل ۵ مقایسه و ارزیابی روش پیشنهادی خود را ارائه می‌دهیم. در نهایت و در فصل ۶ ضمن جمع‌بندی این گزارش علمی، به بررسی نقاط ضعف و قوت این پژوهش و همچنین ارائه‌ی پیشنهاداتی جهت بهبود این پژوهش می‌پردازیم.

فصل ۲

مفاهیم اولیه

در این فصل مفاهیمی را که به صورت مستقیم و غیرمستقیم در این پژوهش از آن‌ها استفاده شده‌است را شرح می‌دهیم. آشنایی با مفاهیم مطروحه در این فصل، منجر به درک بهتر پژوهش و راه‌حل پیشنهادی در فصل ۴ خواهد شد.

۲ـ۱ مبهم‌سازی

آن‌چنان که در فصل پیشین گفته‌شد، مبهم‌سازی را می‌توان از دو دیدگاه تهاجمی و تدافعی بررسی کرد. در این قسمت ما با توجه به هدف پژوهش که تشخیص بازبسته‌بندی به جهت دفاع می‌باشد، مبهم‌سازی را فرایندی در نظر می‌گیریم که در آن فرد مهاجم، برنامک اصلی را دانلود کرده و آن را پس از دیکامپایل کردن، به نوعی تغییر می‌دهد که منطق کلی برنامک، تغییری نمی‌کند. مبهم‌سازی یکی از ارکان اصلی در فرایند بازبسته‌بندی شده‌است و هدف اصلی آن این است که ابزارهای تشخیص بازبسته‌بندی، خصوصا در مواردی که از تحلیل ایستا استفاده می‌کنند، را به اشتباه بیاندازد.
روش‌های مبهم‌سازی را از نظر میزان سختی در تشخیص به ۳ دسته‌ی کلی می‌توان تقسیم کرد[۱۱]:

۲ـ۱ـ۱ روش‌های بدیهی

راهکارهای موجود در این دسته عمدتا بدون تغییر در برنامک اصلی رخ می‌دهد. در این روش متقلب پس از آن‌که به کدهای برنامک اصلی دسترسی پیدا کرد، آن را بدون هیچ گونه تغییری تغیر می‌دهد. بازبسته‌بندی تنها موجب تغیر در امضاء توسعه‌دهنده‌ی برنامک و جمع‌آزما می‌شود و روش‌هایی که مبتنی بر این دو

خصوصیت هستند در این سطح دچار مشکل می‌شوند.

۲ـ۱ـ۲ روش‌های میانی

این دسته از روش‌های مبهم‌سازی، شامل روش‌هایی است که در آن بیشتر ویژگی‌های مبتنی بر معناشناسی تغییر می‌کند و ویژگی‌های مبتنی بر نحو ثابت باقی می‌ماند. بنابراین، روش‌هایی که بیشتر مبتنی بر معناشناسی برنامک‌های اندرویدی هستند، دچار خطای بیشتری در این سطح از مبهم‌نگاری می‌شوند.در ادامه به معرفی مختصری از انواع روش‌های مبهم‌نگاری مطابق با پژوهش [۱۲] در این دسته می‌پردازیم:

- **تغییر نام شناسه‌ها:** تغییر نام شناسه‌های موجود در برنامک شامل نام کلاس‌ها، متدها و یا متغیرهای موجود [۱۱]

```
public class a{
    private Integer a;
    private Float = b;
    public void a(Integer a, Float b){
        this.a = a + Integer.valueOf(b)
    }
}
```

شکل ۲ـ۱: نمونه‌ای از مبهم‌نگاری با استفاده از تغییر نام شناسه‌ها

- **تغییر نام بسته:** در این روش مبهم‌نگاری با استفاده از تغییر نام بسته‌های برنامک صورت می‌گیرد.
- **رمزنگاری رشته‌ها:** استفاده از رمزنگاری در رشته‌های مورد استفاده از فایل‌های دکس، موجب کاهش سطح معناشناسی می‌شود.
- **فراخوانی غیرمستقیم:** یکی از روش‌های ساده‌ی تغییر گراف فراخوانی، استفاده از یک تابع واسط به عنوان تابع فراخواننده‌ی تابع اصلی است. در این حالت تابع اولیه یک تابع واسط و تابع واسط به صورت زنجیرای تابع اصلی را فراخوانی می‌کند. بدنه‌ی تابع واسط در این حالت، بسیار ساده و شامل یک فراخوانی تابع اصلی است.

  فقره **جابه‌جایی دستورات:** جابه‌حایی دستورات موجود در برنامک اصلی، یکی از روش‌های پرکاربرد توسط ابزارهای مبهم‌نگاری است. جابه‌جایی دستورات به شکلی انجام می‌شود که استقلال هر قسمت حفظ گردد.

- **جابه‌جایی ساختار سلسله‌مراتبی:** در این روش، ساختار سلسله مراتبی کلاس‌های برنامک به نوعی تغییر می‌کند که منطق کلاس‌ها دچار تغییر نشود.

- **ادغام و شکستن:** می‌توان توابع و یا کلاس‌های موجود در برنامک‌های اندرویدی را ادغام کرد. برای مثال می‌توان هر جایی که یک تابع صدا زده‌شده بود، فراخوانی تابع با بدنه‌ی تابع جایگزین شود. از طرفی می‌توان بدنه‌ی چند تابع را تحت یک تابع با یکدیگر ادغام کرد. این کار ساختار توابع فراخواننده را نیز تغییر می‌دهد. از طرفی می‌توان یک تابع را به چندین تابع مشخص شکست و بدین صورت گراف جریان برنامک را تغییر داد.

- **واردساختن کدهای بیهوده:** کدهای بیهوده، کدهایی هستند که اجرا می‌شوند ولی تاثیری در ادامه‌ی روند اجرایی برنامک، ندارند. کدهای مرده عموماً دارای ساختارهای کنترلی و حلقه‌های nop هستند که تاثیری در روند اجرای برنامک ندارند. ذکر این نکته حائز اهمیت است که در صورتی که در ساختار کدهای مرده از شروط کنترلی مبتنی بر متغیرهای پویا استفاده شود آن‌گاه دیگر تحلیل ایستای برنامک‌های اندرویدی قادر به تشخیص این قسمت‌ها نیست.

- **واردساختن کدهای مرده:** یکی دیگر از روش‌های تغییر گراف‌های برنامه از جمله گراف جریان، اضافه‌کردن کدهای مرده‌ای است که در ساختار گراف جریان برنامک‌های اندرویدی هیچ‌گاه اجرا نمی‌شوند اما به عنوان یک گره در گراف حضور دارند.

- **روش‌های دیگر:** روش‌های دیگری نظیر تغییر نام منابع مورد استفاده در برنامک‌های اندرویدی و حذف فایل اشکال‌زدایی از روش‌های دیگری است که در این سطح به وفور مورد استفاده قرار می‌گیرد.

۲ـ۱ـ۳ روش‌های خاکستری

روش‌های موجود در این دسته، مبتنی بر نحو برنامک‌های اندرویدی و خصوصا زبان جاوا به وجود آمده‌است. عمده‌ی روش‌های مورد استفاده در این سطح، از خصوصیات مهم زبان جاوا به عنوان زبان اصلی در پیاده‌سازی برنامک‌های اندرویدی، استفاده می‌کنند. در ادامه به بررسی مهم‌ترین روش‌های موجود در این دسته می‌پردازیم.

- **بازتاب:** بازتاب یکی از ویژگی‌های مهم و پیچیده‌ی زبان جاوا می‌باشد[۱۳] که امکان فراخوانی مندها و ارتباط با کلاس‌های برنامک را به صورت پویا فراهم می‌سازد. مهاجمان با استفاده از فراخوانی متدها به وسیله‌ی قابلیت بازتاب، می‌توانند نام واسط فراخوانی‌شده را پنهان سازند و بدین وسیله سطح جدیدی از مبهم‌نگاری را در برنامک‌های اندرویدی ایجاد سازند. استفاده از قابلیت بازتاب و رمزنگاری رشته‌ی واسط مورد نظر، به طور کامل واسط فراخوانی‌شده را مبهم می‌سازد.

```
Object object = new Object();
Method getService = Class.forName("android.os.
    ServiceManager").getMethod("getService",
    String.class);
Object obj = getService.invoke(object, new
    Object[]{new String("batteryinfo")});
```

شکل ۲ـ۲: نمونه‌ای از مبهم‌نگاری با استفاده از قابلیت بازتاب به منظور پنهان‌سازی واسط فراخوانی‌شده به نام batteryinfo

- **رمزنگاری دالویک بایت‌کدها:** در این روش، مهاجم در حین ساختن برنامک بازبسته‌بندی شده، قسمتی مهمی از کدهای برنامک را رمزنگاری کرده و در هنگام اجرا با استفاده از یک رویه‌ی رمزگشایی، کدهای اصلی را بارگیری می‌کند. این روش عمدتاً زمانی استفاده می‌شود که مهاجم نیاز به فراخوانی توابع واسط‌های برنامه‌نویسی داشته‌باشد و قسمتی را که واسط‌ها فراخوانی می‌شوند را رمزنگاری می‌کند.
- **بارگذاری پویای کلاس‌ها:** زبان جاوا از قابلیت مهمی به نام بارگیری پویای کد پشتیانی می‌کند که اجازه می‌دهد تکه کدی را که پیش از این در منابع مورد توسعه‌ی یک برنامک موجود نبود را در حین اجرا به برنامک اضافه کنیم. مهاجم با استفاده از این قابلیت زبان جاوا می‌تواند قسمت‌هایی از برنامک را در حین اجرای آن تغییر دهد که عملاً تشخیص آن‌ها با استفاده از تحلیل‌های ایستا امکان‌پذیر نیست.

۲ـ۱ـ۴ روش‌های ترکیبی

هر ترکیبی از روش‌های گفته‌شده در سطوح مختلف را می‌توان در مبهم‌نگاری استفاده کرد. به صورت کلی روش‌های میانی ۲ـ۱ـ۲ و روش‌های خاکستری ۲ـ۱ـ۳ را می‌توان دو دسته‌ی مهم از انواع مبهم‌نگاری به حساب آورد که به صورت گسترده در مبهم‌نگارهای رایگان و یا تجاری مورد استفاده قرار می‌گیرد.

۲ـ۱ـ۵ انواع مبهم‌نگارها

در قسمت پیشین، دریافتیم که مبهم‌نگاری، سطوح متفاوتی دارد که متقلبان برای تولید برنامک‌های بازبسته‌بندی شده از آن‌ها استفاده می‌کنند. برای ابداع یک روش مفید جهت تشخیص برنامک‌های بازبسته‌بندی شده ابتدا

باید انوع مبهم‌نگارهای موجود را بررسی کرد. در پژوهشی که توسط ژانگ و همکاران[۱۴] انجام شده، ٪۴۳ از برنامک‌های بازبسته‌بندی شده‌ی مورد بررسی در این پژوهش از مبهم‌نگاری‌های بسیار ساده‌ای نظیر تغییر نام و با استفاده از مبهم‌نگارهای رایگان، انجام شده‌است. در ادام به بررسی چند مبهم‌نگار رایگان و تجار می‌پردازیم.

- **پروگارد**

پروگارد یک نرم‌افزار متن‌باز رایگان به جهت بهینه‌سازی و مبهم‌نگاری در برنامه‌های جاوا مورد استفاده قرار می‌گیرد. بهینه‌سازی از طریق حذف کدهای مرده و منابع بلااستفاده انجام می‌شود و مبهم‌نگاری با استفاده روش‌های مطروحه در بخش ۲ـ۱ـ۳ انجام می‌شود.[۱۵]

- **آلاتوری**

آلاتوری یک مبهم‌نگار رایگان تولیدشده توسط شرکت روسی Smardec می‌باشد که سطوح مختلفی از مبهم‌نگاری را با توجه به فایل‌های پیکربندی پوشش می‌دهد. این مبهم‌نگار از تغییرنام، مبهم‌نگاری مبتنی بر تغییر گراف‌های جریان، مبهم‌نگاری فایل‌های اشکال‌زدایی و رمزنگاری داده‌های رشته‌ای پشتیبانی می‌کند.[۱۶، ۱۷]

- **دکس‌گارد**

این مبهم‌نگار نسخه‌ی تجاری پروگارد است که توسط شرکت گارداسکوار تولید شده‌است. دکس‌گارد را می‌توان مشهورترین و یکی از پیچیده‌ترین مبهم‌نگارهای موجود به حساب آورد. آخرین نسخه‌ی این نرم‌افزار انواع مبهم‌نگاری‌های سطح خاکستری نظیر بارگیری پویای کد و همچنین رمزنگاری کلاس‌ها و توابع را به صورت پویا انجام می‌دهد.

۲ـ۲ ساختار فایل‌های برنامک‌های اندرویدی

هر برنامک اندرویدی یک یک فایل فشرده‌شده با پسوند APK است که به اختصار شامل ۴ پوشه‌ی مهم و ۳ فایل است.در ادامه هر کدام از این قسمت‌ها را معرفی و کارکرد آن را بررسی خواهیم‌کرد.[۱۸]

- **پوشه‌ی res:** این پوشه شامل منابع برنامک‌های اندرویدی است که مربوط به ظاهر برنامک می‌شود. این فایل در نهایت به فایل‌های R. نگاشت می‌شود و هر کدام از منابع با یک شناسه مشخص می‌گردد.

- **پوشه‌ی lib:** فایل‌های کامپایل‌شده ی بومی در این پوشه قرار می‌گیرند که شامل کتابخانه‌ها نیز

می‌گردد. استفاده از فایل‌هایی که کامپایل شده‌اند سرعت اجرای برنامک‌های اندرویدی را بالا می‌برد لذا استفاده از آن‌ها به عنوان ماژول‌های از پیش آماده محبوبیت دارد.

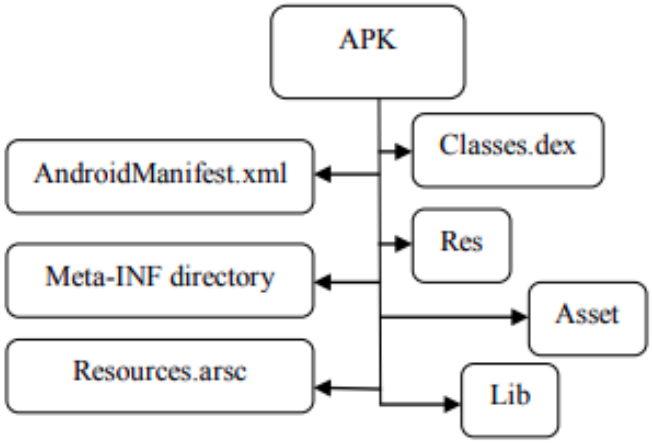
- **فایل Classes.dex:** فایل‌های با پسوند dex فایل‌های دودویی هستند که اطلاعات را در سطر و ستون‌های خود ذخیره می‌کنند. این فایل در برنامک‌های اندرویدی حاوی بایت‌کدهای دالویک است که توسط ماشین مجازی دالویک اجرا می‌شود.

- **فایل AndroidManifest.xml:** پیکربندی‌های مهم فایل‌های APK از جمله لیست مجوزهای مورد نیاز، لیست مولفه‌ها و نام بسته‌ی برنامک در این فایل نوشته می‌شود.

- **پوشه‌ی assets:** این پوشه همانند پوشه‌ی res برای منابع ایستا مورد استفاده قرار می‌گیرد با این تفاوت همه توسعه‌دهندگان در این پوشه می‌توانند عمق زیرپوشه‌ها را به تعداد نامتناهی افزایش دهند تا ساختار بهتری را فراهم سازند.

- **پوشه‌ی META-INF:** این پوشه شامل اطلاعات کلیدهای عمومی کاربر توسعه‌دهنده‌ی برنامک است که برنامک با کلید خصوصی متناظر آن امضا شده‌است. امضای موجود در این پوشه، خاصیت صحت‌سنجی دارد اما اطلاعاتی را از توسعه‌دهنده نشر نمی‌دهد و به صورت خودامضا ساخته می‌شود.

- **فایل resources.arsc:** این فایل برای انجام نگاشت میان منابع موجود در پوشه‌ی resources و شناسه‌ی هر منبع استفاده می‌شود تا بتوان در حین اجرای برنامک‌ها، هر شناسه را با منبع مورد نظر ترجمه شود.



شکل ۲ـ۳: ساختار پوشه‌ها و فایل‌های فایل‌های Apk [۱]

۲ـ۳ کتاب‌خانه‌های اندرویدی

کتابخانه‌های اندرویدی، نمونه‌های از پیش توسعه‌یافته هستند که توسط توسعه‌دهندگان نوشته شده و توسعه‌دهندگان اندروید به جهت سهولت در پیاده‌سازی و کمک به تسریع پیاده‌سازی به وفور از این نمونه‌ها استفاده می‌کنند. کتابخانه‌های اندرویدی به صورت کلی به دوبخش کتابخانه‌های مختص برنامه‌نویسی اندرویدی و کتاب‌خانه‌های زبان جاوا تقسیم می‌شوند. در هنگام کامپایل، تمامی کتاب‌خانه‌هایی که توسعه‌دهنده هنگام توسعه‌ی برنامک آن‌ها را استفاده کرده‌است به همراه کدهای مورد توسعه، کامپایل شده و در ساختار سلسله مراتبی تحت فایل‌های classes.dex قرار می‌گیرد[۱۹]. ذکر این نکته قابل توجه است که تشخیص برنامک‌های بازبسته‌بندی شده بدون شناسایی کتابخانه‌های برنامک اندرویدی مورد نظر امکان‌پذیر نیست. واضح است که در صورتی که نتوانیم کتاب‌خانه‌های اندرویدی شاخص را از جفت برنامک‌های مورد بررسی جدا کنیم، آن‌گاه بخش زیادی از شباهت دو برنامک ناشی از کتاب‌خانه های اندرویدی و اشتراکات موجود در آن‌ها است چرا که بسیاری از کتابخانه‌ها خصوصا کتاب‌خانه های زبان جاوا، در هر برنامک اندرویدی موجود است. از طرفی، مرز عدم وجود مرز مشخصی میان کدهای کتابخانه‌ای و کدهای مورد توسعه توسط توسعه‌دهندگان، شناسایی کتابخانه‌های اندرویدی را تبدیل به یک چالش در زمینه‌ی تشخیص برنامک‌های بازبسته‌بندی در این حوزه کرده‌است.

۲ـ۴ طبقه‌بندی

طبقه‌بندی اطلاعات ورودی یکی از روش‌های مرسوم در هوش‌مصنوعی و یادگیری ماشین است که توسط الگوریتم‌های طبقه‌بند انجام می‌شود. یک طبقه‌بند شامل مجموعه‌ای از الگوریتم‌ها است که برای طبقه‌بندی و یا مرتب‌سازی داده‌های ورودی مورد استفاده قرار می‌گیرد [۲۰]. یکی از ساده‌ترین مثال‌های موجود برای طبقه‌بندی، جداسازی هرزنامه‌ها در سرویس‌های ایمیل است.روش‌های طبقه‌بندی نیازمند مجموعه‌ای از ویژگی‌های اطلاعات ورودی به عنوان ورودی مسئله می‌باشند تا پس از اجرای الگوریتم، ورودی‌های مدنظر را بر اساس آن‌ها در چند طبقه قرار دهند.

۲ـ۵ بازبسته‌بندی برنامک‌های اندرویدی

با پویش عمیق در پژوهش‌های مرتبط با این حوزه در سالیان اخیر متوجه می‌شویم که تعاریف متنوعی برای بازبسته‌بندی در نظر گرفته‌شده است. برخی از پژوهش‌ها نظیر [۲۱، ۲۲] بازبسته‌بندی را در تغییر منابع و

ظاهر برنامک‌ها در نظر می‌گیرند و در نهایت ویژگی‌های مبتنی بر ظاهر آن‌ها را با یکدیگر مقایسه می‌کنند در حالی که برخی از پژوهش‌های اخیر دیگر نظیر [۲۳، ۲۴] باربسته‌بندی مبتنی بر تغییر ویژگی‌های کدپایه بیان شده‌است. علاوه بر این یکی دیگر از اختلافات موجود در تعریف بازبسته‌بندی، وجود مبهم‌نگاری در برنامک‌های بازبسته‌بندی شده‌است. برخی از پژوهش‌ها نظیر [۲۵] بازبسته‌بندی را منطوط به تغییر در امضای برنامک می‌دانند اما بسیاری از پژوهش‌های به‌روزتر، نظیر [۲۶، ۲۷] بازبسته‌بندی را تنها به تغییر منابع و یا کدهای برنامک تقلبی نسبت به برنامک اصلی می‌دانند. همانطور که مشاهده شد، هنوز تعریف مشخصی از بازبسته‌بندی در پژوهش‌ها ارائه نشده‌است اما به طور کلی می‌توان گفت که برنامک A بازبسته‌بندی یک برنامک دیگر است اگر تغییرات آن نسبت به برنامک مبدا محدود و با حفظ کارکرد و منابع برنامک اصلی باشد. این تعریغ در این پژوهش نیز به عنوان تعریف مبنای بازبسته‌بندی در نظر گرفته شده‌است.

فصل ۳

تعریف مسئله و مرور کارهای پیشین

پژوهش‌های اخیر در حوزه‌ی تشخیص برنامک‌های اندرویدی بازبسته‌بندی شده نشان می‌دهد که تشخیص این دسته از برنامک ها تحت تاثیر دو عامل مبهم‌نگاری و جداسازی درست کتابخانه‌های اندرویدی قرار دارد. برخی از پژوهش‌های اخیر انجام‌شده در این حوزه، تشخیص کتاب‌خانه‌های بسته‌ی تقلبی را با فرض عدم مبهم‌نگاری کتابخانه‌ها انجام داده‌اند که مشخصاً این فرضی نادرست است چرا که بسیاری از مبهم‌نگارهای ابتدایی نیز این کار را در کتابخانه‌های اندرویدی انجام می‌دهند. در اکثر روش‌های پیشنهادی قسمتی از روش، مختص تشخیص و جداسازی کتابخانه‌های اندرویدی است. شناسایی کدهای کتابخانه‌ای از آن‌جهت اهمیت دارد که تشخیص درست آن‌ها می‌تواند نتایج مثبت غلط و منفی غلط را کاهش دهد. در بیشتر مواقع، خصوصا در ابزارهای مبهم‌نگاری، متقلب هنگام بازبسته‌بندی اقدام به مبهم‌نگاری در کتابخانه‌های اندرویدی می‌کند و بدین صورت سعی در افزایش منفط غلط در ابزارهای تشخیص دارد. در صورتی که کدهای کتابخانه‌ای به درستی تشخیص و جداسازی نشوند، شباهت‌های موجود میان برنامک‌های مورد بررسی، خصوصا در روش‌های مبتنی بر تحلیل ایستا، ناشی از کدهای کتابخانه‌ای خواهد بود. از سوی دیگر، تشخیص مبهم‌نگاری در کدهای مورد توسعه توسط متقلب، نیازمند ویژگی‌هایی از برنامک مورد نظر است که مقاومت بالایی در برابر مبهم‌نگاری داشته باشند. بدین معنا که متقلب برای تغییر این دسته ازویژگی‌ها ناچار به پرداخت هزینه‌ی زمانی و فنی باشد و در نهایت از تغییر این دست از ویژگی‌ها، پرهیز کند. در بسیاری از روش‌های ارائه‌شده در سال‌های اخیر، تشخیص برنامک‌های بازبسته‌بندی شده مبتنی بر ویژگی‌هایی صورت گرفته است که در عین مقاومت در مقابل مبهم‌نگاری، هزینه‌ی محاسباتی تشخیص برنامک‌های بازبسته‌بندی شده را افزایش می‌هد به طوری که استفاده از این روش‌ها را عملا در یک محیط صنعتی غیر ممکن ساخته‌است.

با توجه به اهمیت تشخیص مبهم‌نگاری و در نهایت تشخیص برنامک‌های بازبسته‌بندی شده و همچنین، در

نظر گرفتن سرعت تشخیص به عنوان یک عامل مهم ، در این فصل به بررسی و مرور کارهایی می‌پردازیم که روش‌های گوناگونی را برای تشخیص برنامک‌های بازبسته‌بندی استفاده کرده‌اند و مزایا و معایب هر کدام را به صورت جدا بررسی خواهیم‌کرد.از آن‌جایی که هدف این پژوهش بهبود کارایی روش‌های تشخیص برنامک‌های بازبسته‌بندی شده‌است و تمرکز پژوهش بر روی تشخیص کدهای کتابخانه‌ای نبوده‌است، در ابتدا روند کلی تشخیص برنامک‌های بازبسته‌بندی شده را در پژوهش‌های مرتبط بیان کرده و به اختصار، روش‌های جداسازی کتابخانه‌های اندرویدی از کدهای مورد توسعه را توضیح می‌دهیم و از مرور کارهای پیشین انجام‌شده در این حوزه پرهیز خواهیم کرد.

در ادامه ابتدا به روند کلی تشخیص برنامک‌های بازبسته‌بندی شده می‌پردازیم و مسئله‌ی تشخیص برنامک‌های بازبسته‌بندی شده را از دیدگاه این پژوهش، شرح می‌دهیم. همچنین، دسته‌بندی انواع روش‌های تشخیص را با توجه به پژوهش‌های سال‌های اخیر بیان می‌کنیم و از هر دسته چند پژوهش انجام‌شده را بررسی خواهیم کرد. برای درک بهتر روش پیشنهادی در هر قسمت به بیان مزایا و معایب هر روش خواهیم پرداخت و علاوه بر این روش تشخیص کدهای کتابخانه‌ای در هر روش مشخص خواهیم‌کرد.

۳ـ۱ تعریف مسئله

علارغم پژوهش‌های متعدد صورت‌گرفته در این زمینه، همانند تعریف بازبسته‌بندی، هنوز تعریف مشخصی نیز برای تشخیص بازبسته‌بندی ارائه‌نشده است. پژوهش‌های سال‌های اخیر در حالت کلی تشخیص بازبسته‌بندی را به دو صورت تعریف می‌کنند:

تعریف ۳ـ۱ (تشخیص بازبسته‌بندی مبتنی بر برنامک مبدا) */ تشخیص بسته‌ی بازبسته‌بندی شده، یعنی تشخیص جفتی از برنامک‌های درون مخزن که دقیقا جفت مشابه برنامک ورودی باشد. به بیان دیگر در این تعریف مشخص می‌شود که برنامک ورودی بازبسته‌بندی شده است یا خیر و در صورتی که بود، جفت برنامک آن درون مخزن نیز مشخص می‌شود.*

تعریف ۳ـ۲ (تشخیص بازبسته‌بندی مبتنی بر تصمیم‌گیری برنامک مقصد) *تشخیص بسته‌ی بازبسته‌بندی شده، یعنی مشخص کنیم برنامک ورودی بازبسته‌بندی شده است یا خیر. در این حالت تشخیص برنامک اصلی اهمیتی ندارد و مسئله، تصمیم‌گیری درباره‌ی بازبسته‌بندی بودن یک برنامک ورودی است.*

در سال‌های اخیر، اکثر پژوهش‌ها از یکی از تعاریف بالا برای تشخیص بازبسته‌بندی استفاده کرده‌اند. برای پاسخ به تعریف ۲ ، پژوهش‌هایی نظیر [۲۸، ۲۹، ۳۰] از روش‌های مبتنی بر مدل‌های یادگیری ماشین برای تشخیص برنامک‌های بازبسته‌بندی شده استفاده کرده‌اند. حال آن‌که پژوهش‌های مرتبط با تعریف ۱ ،

نظیر [۳۱، ؟] بیشتر از روش‌های مقایسه‌ی دودویی و مبتنی بر شباهت‌سنجی استفاده کرده‌اند.
تعریفی که در این پژوهش مشخص مورد استفاده قرارگرفته است، تعریف ۱ است. یعنی تشخیص بازبسته‌بندی منوط به ت**شخیص جفت برنامک اصلی** در مخزن برنامک‌های پژوهش می‌باشد. بنابراین در طی فرایند تشخیص به ۲ سوال اساسی پاسخ می‌دهیم :

- آیا برنامک ورودی بازبسته‌بندی شده‌ی یک برنامه‌ی دیگر است؟
- در صورتی که برنامک مورد بررسی، بازبسته‌بندی شده‌ی برنامک دیگری بود، آن‌گاه جفت بازبسته‌بندی شده‌ی برنامک ورودی مشخص گردد.

۳ـ۲ روند کلی تشخیص برنامک‌های بازبسته‌بندی شده

با بررسی پژوهش‌های صورت‌گرفته در حوزه‌ی تشخیص برنامک‌های بازبسته‌بندی شده، درمی‌یابیم که به طور مشخیص عمده‌ی این روش‌ها مراحل مشابهی را برای حل این مسئله، دنبال کرده‌اند. به طور کلی عمده‌ی روش‌های تشخیص، به عنوان ورودی، یک برنامک اندویدی شامل یک فایل با پسوند Apk را دریافت کرده و پس از گذر از سه مرجله، مسئله را حل می‌کنند. در ادامه به بررسی این سه مرحله می‌پردازیم:

۳ـ۲ـ۱ پیش‌پردازش برنامک‌های اندرویدی

یکی از مراحل مهم در تشخیص برنامک‌های بازبسته‌بندی شده، مرحله‌ی پیش‌پردازش است که تاثیر به سزایی در سرعت و دقت روش تشخیص خواهد داشت. حذف کدهای کتابخانه‌ای ، حذف کدهای مرده و یا بیهوده و اعمال فیلترهای ساختاری از موارد نمونه در قسمت پیش‌پردازش است. در این قسمت روش‌های کلی مورد استفاده توسط پژوهش‌های اخیر جهت حذف کدهای کتابخانه‌ای را توضیح می‌دهیم. با توجه به مرور کارهای پیشین انجام‌شده در این حوزه، به صورت کلی دو دیدگاه در مورد تشخیص و جداسازی کتابخانه‌های اندرویدی وجود دارد:

- **مبتنی بر لیست سفید:** در این روش، لیستی از نام بسته‌ای مشهور کتابخانه‌ای در برنامک‌های اندرویدی در دسترس است و با استفاده از نام بسته‌های موجود در برنامک، کدهای کتابخانه‌ای از کدهای مورد توسعه جدا می‌شوند. راه حل‌های مبتنی بر این روش، عموماً در مقابل مبهم‌نگاری‌های ساده‌ای نظیر تغییر نام بسته نیز مقاوم نیستند و به راحتی می‌توان آن‌ها را دور زد. مزیت این روش

آن‌است که سرعت بالایی دارد چرا که فقط نام بسته‌ها با یکدیگر مقایسه می‌شوند اما دقت خوبی را ارائه نمی‌دهند.

- **مبتنی بر شباهت‌سنجی و کدهای تکراری:** در این روش، ابتدا مخزن بزرگی از کتابخانه‌های اندرویدی تهیه می‌شود و به روش‌های گوناگون کدهای کلاسی برنامک و کدهای کتابخانه‌ای موجود در مخزن، با یکدیگر مقایسه می‌شوند و بدین طریق کتابخانه‌های اندرویدی از کدهای مورد توسعه در برنامک، جدا می‌شود. روش‌های مبتنی بر شباهت‌سنجی، بسته به این‌که از چه روشی برای یافتن کدهای تکراری استفاده می‌کنند، دقت‌های متفاوتی دارند اما به صورت کلی می‌توان گفت که مقاومت آن‌ها در مقابل مهم‌نگاری بسیار بیشتر از روش‌های مبتنی بر لیست سفید است.

۳ـ۲ـ۲ استخراج ویژگی

پس از حذف کدهای کتاب‌خانه‌ای در قسمت قبلی و انجام پیش‌پردازش‌های مورد نیاز، کدهای منبع برنامک هدف، به یک طرح کلی مدل می‌شود. به صورت کلی می‌توان روش‌های تشخیص برنامک‌های بازبسته‌بندی شده را در پژوهش‌های سالیان خیر، ناشی از تفاوت در دیدگاه در مرحله‌ی استخراج ویژگی دانست. همانطور که در شکل ـــ- مشاهده می‌شود، روش‌های تشخیص برنامک‌های بازبسته‌بندی به صورت کلی به دو بخش تحلیل ایستا و تحلیل پویا تقسیم می‌شود. از آن‌جایی که هدف ما در این پژوهش، تنها بررسی پژوهش‌هایی است که راه‌حل تدافعی ارائه داده‌اند بنابراین روش‌هایی که توسعه‌دهندگان و شرکت‌های توسعه‌دهنده جهت جلوگیری از انجام بازبسته‌بندی پیاده‌سازی می‌کنند را توضیح نمی‌دهیم. به صورت کلی، می‌توان روش‌های تشخیص برنامک‌های بازبسته‌بندی شده را به دو بخش روش‌های تحلیل پویا و یا روش‌های تحلیل ایستا تقسیم کرد که در ادامه به بررسی هر کدام از این روش‌های می‌پردازیم.

- **روش‌های مبتنی بر تحلیل ایستا:** روش‌های مبتنی بر تحلیل ایستا، در مقابل مبهم‌نگاری‌های ایستا که در هنگام بازبسته‌بندی و انجام ری‌کامپایل انجام می‌شود مقاوم هستند. اما همانطور که می‌توان حدس زد، این دسته از روش‌ها مقابل روش‌های مبهم‌نگاری همانند بازتاب مقاومتی ندارند و ممکن است دچار خطا شوند. همچنین روش‌های مبهم‌نگاری مبتنی بر رمزنگاری پویا نیز این روش‌ها را دچار خطا می‌کند. یکی از مزایای مهم روش‌های مبتنی بر تحلیل ایستا آن است که در صورت پیاده‌سازی درست و استفاده از ویژگی‌های مقاوم، می‌توانند طیف وسیعی از برنامک‌های بازبسته‌بندی شده را تشخیص دهند.
- **روش‌های مبتنی بر تحلیل پویا:** ارائه‌ی روش‌های مبتنی بر تحلیل پویا، به هدف جلوگیری از

مبهم‌نگاری‌های در لحظه‌ی اجرا که در برنامک‌های اندرویدی صورت می‌گیرد. به همین علت روش‌های موجود در این حوزه، عمدتا برنامک‌ها را در هنگام اجرا بررسی و استخراج ویژگی عمدتا در هنگام اجرا صورت می‌گیرد. به طول کلی، روش‌های مبتنی بر تحلیل پویا از مقاومت بیشتر در مقابل استفاده از راهکارهای مبهم‌نگاری برخوردار هستند. استفاده از شبیه‌سازهای جعبه‌شن به وفور در پژوهش‌های این حوزه، یافت می‌شود. یکی از چالش‌های اصلی در تشخیص برنامک‌های اندرویدی بازبسته‌بندی شده، چگونگی پیاده‌سازی شبیه‌ساز هاست. بسیار از شبیه‌ساز ها توانایی شبیه‌سازی تمامی خدمات موجود در برنامک را ندارند و برای تحلیل دقیق‌تر نیازمند استفاده از کاربران واقعی در شبیه‌سازی و استفاده از خدمات برنامک هستند. عامل دیگری که تشخیص با استفاده از تحلیل پویا را مشکل می‌کند، این است که بسیاری از بدافزارهای توسعه‌یافته، توانایی تشخیص محیط اجرای شبیه‌سازی شده را دارند و ممکن است تمامی قابلیت‌های خود و یا بخشی از آن را به جهت دور زدن سیستم‌های تشخیص پویا، نشان ندهند.

۳ـ۲ـ۳ تشخیص بازبسته‌بندی

در این مرحله با توجه به معیارها و ویژگی‌هایی که از قسمت قبل به دست آمده است و با استفاده از روش‌های گوناگون برنامک بازبسته‌بندی شده مشخص می‌شود. به صورت کلی، روش‌های پیاده‌سازی شده در این قسمت، مبتنی بر مقایسه‌ی دودویی و یا طبقه‌بندی و یادگیری ماشین هستند.

- **مقایسه‌ی دودویی:** روش‌های مبتنی بر مقایسه‌ی دودویی، مدل استخراج شده در قسمت قبلی را با استفاده از شباهت‌سنجی با برنامک‌های موجود در مخزن مقایسه می‌کند و در نهایت برنامک بازبسته‌بندی شده را مشخیص می‌کند. اکثر روش‌های مبتنی بر مقایسه‌ی دودویی، جفت برنامک اصلی را نیز مشخص می‌کنند و از تعریف ۳ـ۱ استفاده می‌کنند بنابراین یکی از مزیت‌های این روش‌ها پوشش گشسترده‌تر از تعریف تشخیص بازبسته‌بندی است ولی در کنار آن اکثر روش‌های موجود در این زمینه، محاسبات بالایی دارند که باعث می‌شود سرعت آن‌ها کاهش یابد.
- **مبتنی بر طبقه‌بندی و یادگیری ماشین:** یکی دیگر از روش‌های تشخیص بازبسته‌بندی با استفاده از ویژگی‌های مستخرج از مرحله‌ی قبل، استفاده از طبقه‌بند ها و مدل‌های یادگیری ماشین است. اکثر پژوهش‌های موجود در این زمینه از تعریف ۳ـ۲ برای تشخیص برنامک بازبسته‌بندی شده استفاده می‌کنند. بنابراین، تنها تصمیم‌گیری در مورد بازبسته‌بندی بودن یا نبودن برنامک ورودی را انجام می‌دهند. یکی از مزایای مهم این روش‌ها، سرعت بالای آن است چرا که تنها در زمان مرحله‌ی یادگیری، نیازمند محاسبات بالایی هستند و در صورتی که مدل این روش‌ها به درستی عمل کند،

سرعت تشخیص به صورت قابل توجهی بالاتر از روش‌های مبتنی بر مقایسه‌ی دودویی است.

۳ـ۳ مرورکارهای پیشین

همانطور که در شکل —— مشاهده می‌شود، اکثر پژوهش‌های تشخیص بازبسته‌بندی از روش‌های مقایسه‌ای مبتنی بر تحلیلی ایستا و ای پویا استفاده می‌کنند. در ادامه‌ی این قسمت ابتدا روش‌های ایستا و همچنین پژوهش‌های اخیر مرتبط با این حوزه را بررسی خواهیم کرد و در ادامه روش‌های مبتنی بر تحلیل پویا شرح داده می‌شود.

۳ـ۳ـ۱ مبتنی بر تحلیل ایستا

در این قسمت، روش‌های مبتنی بر تحلیل ایستا و پژوهش‌های مرتبط با آن را بررسی خواهیم کرد. همانطور که گفتیم تحلیل ایستا، روشی محبوب در میان پژوهش‌های اخیر موجود در این حوزه است چرا که پیچیدگی‌های روش‌های پویا را ندارد و می‌توان به کمک آن‌ها طیف وسیعی از تشخیص مبهم‌نگاری‌ها را پشتیبانی کرد.

روش‌های مبتنی بر آپکد

استفاده از آپکدهای موجود در فایل‌های دالویک، یکی از روش‌های تشخیص برنامک‌های بازبسته‌بندی شده است. هدف از پژوهش آقای ژو [۳۲] و همکاران، توسعه‌ی ابزاری به نام درویدمس بوده است که توسط آن مشخص شود چه تعدادی از برنامک‌های موجود در فروشگاه‌های اندرویدی غیررسمی، بازبسته‌بندی شده‌ی برنامک‌های موجود در فروشگاه‌های رسمی هستند. همانطور که گفته شد نظارت کافی‌ای بر روی فروشگاه‌های غیر رسمی وجود ندارد، بنابراین متقلبین از این فروشگاه‌ها به عنوان یک راه امن و دردسترس برای پخش‌کردن برنامک‌های بازبسته‌بندی شده استفاده می‌کنند. برای استخراج امضای برنامک در این پژوهش از کدهای دالویک موجود در Classes.dex و امضای دیجیتال برنامه‌نویس در فراداده استفاده‌شده است. پس از جداسازی کدهای کتابخانه‌ای به وسیله‌ی لیست سفید و استخراج آپکدها از فایل‌های دالویک، از یک پنجره‌ی لغزات روی آپکدها استفاده شده و در نهایت چکیده‌ی آپکدها به همراه امضای دیجیتال برنامه‌نویس موجود در پوشه‌ی META-INF تشکیل امضای برنامک را می‌دهند. همانطور که می‌توان فهمید، فرض پژوهش آن بوده است که کلید خصوصی توسعه‌دهنده لو نرفته‌است. در نهایت برای قسمت شباهت‌سنجی، از الگوریتم فاصله ویراشی استفاده‌شده است. در قسمت شباهت‌سنجی از ۲۲۹۰۶ برنامک

موجود در فروشگاه‌های رسمی استفاده شده و نتایج پژوهش نشان می‌دهد که ۵ تا ۱۳ درصد از برنامک‌های موجود در فروشگاه‌های غیر رسمی، بازبسته‌بندی شده‌ی برنامک‌های فروشگاه‌های رسمی است. در پژوهش دیگری که توسط آقای ژو[۲۴] ارائه شده‌است، هدف، افزایش سرعت پژوهش قبلی با استفاده از نمونه‌های n تایی از آپکدها بوده است. در این پژوهش امضای هر برنامک متشکل از قسمتی از فراداده‌ی آن شامل فایل‌های منیفست و اطلاعاتی در مورد تعداد فایل‌های برنامک، توصیفات آن و چکیده‌ی آپکدهای دستورات برنامه است.

فصل ۴

نتایج جدید

در این فصل نتایج جدید به‌دست‌آمده در پایان‌نامه توضیح داده می‌شود. در صورت نیاز می‌توان نتایج جدید را در قالب چند فصل ارائه نمود. همچنین در صورت وجود پیاده‌سازی، بهتر است نتایج پیاده‌سازی را در فصل مستقلی پس از این فصل قرار داد.

فصل ۵

نتیجه‌گیری

در این فصل، ضمن جمع‌بندی نتایج جدید ارائه‌شده در پایان‌نامه یا رساله، مسائل باز باقی‌مانده و همچنین پیشنهادهایی برای ادامه‌ی کار ارائه می‌شوند.

فصل ۶

نتیجه‌گیری

مراجع

[1] M. S. Bhatt, H. Patel, and S. Kariya. A Survey Permission Based Mobile Malware Detection. *Int.J.Computer Technology and Applications*, 6(5):852–856, 2015.

[2] Global mobile OS market share 2022 | Statista — statista.com. https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/#:~:text=Android%20maintained%20its%20position%20as,the%20mobile%20operating%20system%20market. [Accessed 02-Feb-2023].

[3] Play Protect | Google Developers — developers.google.com. https://developers.google.com/android/play-protect. [Accessed 02-Feb-2023].

[4] Decompile and modify an Android application | cylab.be — cylab.be. https://cylab.be/blog/69/decompile-and-modify-an-android-application. [Accessed 02-Feb-2023].

[5] A. Dizdar. OWASP Mobile Top 10 Vulnerabilities and How to Prevent Them — brightsec.com. https://brightsec.com/blog/owasp-mobile-top-10/. [Accessed 02-Feb-2023].

[6] D. J. Wu, C. H. Mao, T. E. Wei, H. M. Lee, and K. P. Wu. DroidMat: Android malware detection through manifest and API calls tracing. *Proceedings of the 2012 7th Asia Joint Conference on Information Security, AsiaJCIS 2012*, pages 62–69, 2012.

[7] K. Khanmohammadi, N. Ebrahimi, A. Hamou-Lhadj, and R. Khoury. Empirical study of android repackaged applications. *Empirical Software Engineering*, 24(6):3587–3629, 2019.

[8] T. Vidas and N. Christin. Sweetening android lemon markets: Measuring and combating malware in application marketplaces. *CODASPY 2013 - Proceedings of*

the 3rd ACM Conference on Data and Application Security and Privacy, 2011:197–207, 2013.

[9] P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury. A study on malicious software behaviour analysis and detection techniques: Taxonomy, current trends and challenges. *Future Generation Computer Systems*, 130:1–18, 2022.

[10] Z. Ma, H. Wang, Y. Guo, and X. Chen. Libradar: Fast and accurate detection of third-party libraries in android apps. In *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*, pages 653–656, 2016.

[11] S. Dong, M. Li, W. Diao, X. Liu, J. Liu, Z. Li, F. Xu, K. Chen, X. F. Wang, and K. Zhang. *Understanding android obfuscation techniques: A large-scale investigation in the wild*, volume 254. Springer International Publishing, 2018.

[12] V. Rastogi, Y. Chen, and X. Jiang. DroidChameleon: Evaluating Android anti-malware against transformation attacks. *ASIA CCS 2013 - Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, pages 329–334, 2013.

[13] Trail: The reflection api the javax; tutorials — docs.oracle.com. https://docs.oracle.com/javase/tutorial/reflect/index.html. [Accessed 02-Feb-2023].

[14] X. Zhang, F. Breitinger, E. Luechinger, and S. O'Shaughnessy. Android application forensics: A survey of obfuscation, obfuscation detection and deobfuscation techniques and their impact on investigations. *Forensic Science International: Digital Investigation*, 39:301285, 2021.

[15] ProGuard Manual: Home | Guardsquare — guardsquare.com. https://www.guardsquare.com/manual/home. [Accessed 02-Feb-2023].

[16] Allatori Java Obfuscator — codedemons.net. http://www.codedemons.net/allatori.html. [Accessed 02-Feb-2023].

[17] Y. Wang. Obfuscation-Resilient Code Detection Analyses for Android Apps. 2018.

[18] L. Ardito, R. Coppola, S. Leonardi, M. Morisio, and U. Buy. Automated Test Selection for Android Apps Based on APK and Activity Classification. *IEEE Access*, 8:187648–187670, 2020.

[19] L. Li, T. F. Bissyandé, J. Klein, and Y. Le Traon. An investigation into the use of common libraries in android apps. 1:403–414, 2016.

[20] N. Karankar, P. Shukla, and N. Agrawal. Comparative study of various machine learning classifiers on medical data. pages 267–271, 2017.

[21] Y. Shao, X. Luo, C. Qian, P. Zhu, and L. Zhang. Towards a scalable resource-driven approach for detecting repackaged android applications. *ACM International Conference Proceeding Series*, 2014-Decem(December):56–65, 2014.

[22] F. Zhang, H. Huang, S. Zhu, D. Wu, and P. Liu. Viewdroid: Towards obfuscation-resilient mobile application repackaging detection. page 25–36, 2014.

[23] J. Crussell, C. Gibler, and H. Chen. Attack of the clones: Detecting cloned applications on Android markets. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7459 LNCS:37–54, 2012.

[24] H. Gonzalez, N. Stakhanova, and A. A. Ghorbani. Droidkin: Lightweight detection of android apps similarity. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, 152(January 2015):436–453, 2015.

[25] X. Chen, C. Li, D. Wang, S. Wen, J. Zhang, S. Nepal, Y. Xiang, and K. Ren. Android HIV: A Study of Repackaging Malware for Evading Machine-Learning Detection. *IEEE Transactions on Information Forensics and Security*, 15(8):987–1001, 2020.

[26] A. Salem. Stimulation and Detection of Android Repackaged Malware with Active Learning. 2015.

[27] T. Nguyen, J. T. McDonald, W. B. Glisson, and T. R. Andel. Detecting repackaged android applications using perceptual hashing. *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2020-January:6641–6650, 2020.

[28] F. Alswaina and K. Elleithy. Android malware family classification and analysis: Current status and future directions. *Electronics (Switzerland)*, 9(6):1–20, 2020.

[29] F. Akbar, M. Hussain, R. Mumtaz, Q. Riaz, A. W. A. Wahab, and K.-H. Jung. Permissions-based detection of android malware using machine learning. *Symmetry*, 14(4), 2022.

[30] X. Chen, C. Li, D. Wang, S. Wen, J. Zhang, S. Nepal, Y. Xiang, and K. Ren. Android HIV: A study of repackaging malware for evading machine-learning detection. *IEEE Transactions on Information Forensics and Security*, 15:987–1001, 2020.

[31] Q. Zhang, X. Zhang, Z. Yang, and Z. Qin. An efficient method of detecting repackaged android applications. pages 056 (4 .)–056 (4 .), 01 2014.

[32] W. Zhou, Y. Zhou, X. Jiang, and P. Ning. Detecting repackaged smartphone applications in third-party android marketplaces. page 317–326, 2012.

واژه‌نامه

الف

ابتکاری ..... heuristic
ابعاد بالا ..... high dimensions
اریب ..... bias
آستانه ..... threshold
اصل لانه‌ی کبوتری ..... pigeonhole principle
ان‌پی‌ـسخت ..... NP-Hard
انتقال ..... transition

ب

برخط ..... online
برنامه‌ریزی خطی ..... linear programming
بهینه ..... optimum
بیشینه ..... maximum

پ

پرت ..... outlier
پرسمان ..... query
پوشش ..... cover
پیچیدگی ..... complexity

ت

تجربی ..... experimental
تراکم ..... density
تقریب ..... approximation
تقسیم‌بندی ..... partition
توری ..... mesh
توزیع‌شده ..... distributed

ج

جداپذیر ..... separable
جعبه سیاه ..... black box
جویبار داده ..... data stream

ح

حدی ..... extreme
حریصانه ..... greedy

خ

خوشه ..... cluster
خطی ..... linear

د

داده ..... data

داده‌کاوی ..... data mining

داده‌ی پرت ..... outlier data

دوبرابرسازی ..... doubling

دودویی ..... binary

ر

رأس ..... vertex

رسمی ..... formal

ز

زیرخطی ..... sublinear

س

سرشکن ..... amortized

سلسه‌مراتبی ..... hierarchichal

ش

شبه کد ..... pseudocode

شیء ..... object

ص

صدق‌پذیری ..... satisfiability

غ

غلبه ..... dominate

ف

فاصله ..... distance

فضا ..... space

ق

قطعی ..... deterministic

ک

کارا ..... efficient

کاندیدا ..... candidate

کمینه ..... minimum

م

مجموعه ..... set

مجموعه هسته ..... coreset

مسطح ..... planar

موازی‌سازی ..... parallelization

میان‌گیر ..... buffer

ن

نابه‌جایی ..... inversion

ناوردا ..... invariant

نقطه‌ی مرکزی ..... center point

نیم‌فضا ..... half space

هـ

هزینه‌ی آشوب ..... price of anarchy (POA)

ی

یال ..... edge

پیوست آ

مطالب تکمیلی

پیوست‌های خود را در صورت وجود می‌توانید در این قسمت قرار دهید.

Abstract

We present a standard template for typesetting theses in Persian. The template is based on the XePersian package for the LaTeX typesetting system. This write-up shows a sample usage of this template.

Keywords: Thesis, Typesetting, Template, XePersian

Sharif University of Technology

Department of Computer Engineering

M.Sc. Thesis

Performance Improvement of Android Repackaged Applications

By:

Mojtaba Moazen

Supervisor:

Dr. Amini

february 2023