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[۱] اندروید[۲] به دلیل سهولت در توسعه[۳] توسط توسعه‌دهندگان[۴] موبایلی و در نتیجه فراوانی استفاده از آن در تلفن‌های همراه، تلوزیون‌های هوشمند و دیگر دستگاه‌های موجود، حجم بالایی از بازار مصرفی سیستم‌عامل‌های موبایلی را به خود اختصاص داده‌است. بر طبق گزارش پایگاه استاتیستا[۵] [۳] 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 بستر[۶]، فروشگاه‌های اندرویدی زیادی به جهت ارائه‌ی خدمات به کاربران به وجود آمده‌است. برخی از فروشگاه‌های رسمی مانند فروشگاه اندرویدی گوگل[۷]، از ابزارهایی نظیر پلی پروتکت[۸] [۴] برای بررسی برنامک‌های اندرویدی موجود در فروشگاّه استفاده می‌کنند. علاوه بر این، در سال‌های اخیر فروشگآه‌های متعدد رایگانی به وجود آمده‌اند که صرفا برنامک‌های اندرویدی موجود در سطح وب را غربان[۹] 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[۱۰] به تلفن‌های همراه، بازبسته‌بندی نرم‌افزار[۱۱] است. مطابق تعریف، بازبسته‌بندی شامل

[۱] Operation System
[۲] Android
[۳] Development
[۴] Developers
[۵] Statista
[۶] Platform
[۷] Google
[۸] Play Protect
[۹] Crawl
[۱۰] Malware
[۱۱] Software Repackaging

دانلود[۱۲] یک برنامک، دسترسی به محتوای کدهای برنامک اصلی از طریق روش‌های مهندسی معکوس[۱۳] و در نهایت بازبسته‌بندی به همراه تغییر و یا بدون تغییردادن کدهای برنامک اصلی[۱۴] است. زبان اصلی توسعه در برنامک‌های اندرویدی، زبان جاوا[۱۵] می‌باشد که یک زبان سطح بالا[۱۶] محسوب می‌شود. در طی فرآیند کامپایل[۱۷] برنامک‌های اندرویدی، مجموعه‌ی کدهای منبع در طی فرایندی به بایت‌کدهای دالویک[۱۸] تبدیل می‌شوند و در ادامه ماشین مجازی جاوا[۱۹]، بایت‌کدها را برروی ماشین مقصد اجرا می‌کند[۵]. 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[۲۰] در برنامک مورد توسعه، فهم بدنه‌ی اصلی برنامک را برای مهاجم[۲۱] 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 دیدگاه تهاجمی، مهاجم به جهت اهدافی متفاوتی نظیر تغییر کدهای تبلیغاتی[۲۲] 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[۶]. مطابق پژوهش آقای ژو و همکاران[۷] حدود۸۵ درصد بدافزارهای موجود، از طریق بازبسته‌بندی منتشر می‌شوند. همانطور که گفته شد، برخی فروشگآهای اندرویدی نظیر گوگل، سازوکار مشخصی را برای تشخیص[۲۳] بازبسته‌بندی ارائه‌داده‌اند اما بسیاری از فروشگاه‌های اندرویدی فعال و پربازدید، خصوصا فروشگاه‌های رایگان، یا از هیچ ابزاری استفاده نمی‌کنند و یا در صورت توسعه‌ی نرم‌افزار بومی[۲۴] خود برای شناسایی برنامک‌های بازبسته‌بندی شده، مشخصات و یا دقت آن را گزارش نکرده‌اند.

همانطور که اشاره‌شد، به دلیل محبوبیت و در نهایت استفاده‌ی زیاد برنامک‌های اندرویدی و همچنین

---

[۱۲] Download
[۱۳] Reverse Engineering
[۱۴] Orginal Application
[۱۵] Java
[۱۶] High Level
[۱۷] Compile
[۱۸] Dalvic Byte Code
[۱۹] Java Virtual Machine
[۲۰] Obfuscation
[۲۱] Attacker
[۲۲] Ad Code
[۲۳] Detection
[۲۴] Native

نظارت کم در فروشگاه‌های مرتبط، بازبسته‌بندی یک روش پر استفاده به جهت انتقال بدافزار به تلفن همراه کاربران است. آقای خانمحمدی و همکاران[۸]، پس از بررسی برنامک‌های اندرویدی مجموعه‌داده‌ی[۲۵] اندروزو[۲۶]، دریافتند که ٪۵۲/۲۲ از برنامک‌های موجود در این مخزن توسط ویروس‌توتال[۲۷]، بدافزار شناسایی شده‌اند. ویروس‌توتال، ابزاری متشکل از ۳۰ ضدبدافزار برای بررسی یک برنامک اندرویدی است. مطابق این پژوهش، ٪۷۷/۸۴ از برنامک‌های این مجموعه‌داده که بازبسته‌بندی شده‌اند، دارای نوعی از بدافزار ضدتبلیغاتی[۲۸] بوده‌اند که موجب می‌شود تبلیغات موجود در برنامک تغیر کرده و اهداف مالی و امنیتی کاربران و توسعه‌دهندگان مخدوش شود. علاوه بر این، مطابق پژوهشی که توسط ویداس وهمکاران[۹] 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 بوده‌اند. 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 دچار 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 می‌توان به صورت 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 ایستا[۲۹]ی جفت برنامک‌ها است و استفاده از تحلیل پویا[۳۰] به جهت سرعت پایین آن، محبوبیت فراوانی ندارد[۱۰]. در سمت دیگر طبقه‌بندی[۳۱] 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[۳۲] هستند. کدهای برنامک، منطق[۳۳] برنامک را تشکیل می‌دهند و رفتار برنامک با توجه به این قسمت مشخص می‌شود. از طرفی منابع

[۲۵] Data Set
[۲۶] Androzoo
[۲۷] Virus total
[۲۸] AdWare
[۲۹] Static
[۳۰] Dynamic
[۳۱] Classification
[۳۲] Resouurces
[۳۳] Logic

برنامک، رابط کاربری[۳۴] آن را تشکیل می‌دهند. روش‌های مبتنی بر تحلیل برنامک و یا طبقه‌بندی آن، عمدتاً از ویژگی‌های موجود در منابع و یا کد آن 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[۳۵] و منبع‌پایه[۳۶]، 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 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 برنامک، بررسی دوبه‌دوی برنامک‌های ورودی و مجموعه‌داده مدنظر است. در این روش‌ها تعریف تشخیص بازبسته‌بندی گسترش یافته و یافتن جفت بازبسته‌بندی به صورت مشخیص، از اهداف اصلی پژوهش است. 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، کتابخانه‌ها[۳۷]یی 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 [۱۱] ٪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 در صورتی که به درستی جداسازی صورت گیرد، 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[۳۸] و یا روش تشخیص

[۳۴] User Interface
[۳۵] Code Base
[۳۶] Resource Base
[۳۷] Library
[۳۸] White List

مبتنی بر شباهت سنجی[۳۹]. در روش لیست سفید، لیستی از مشهور ترین کتابخانه‌های موجود در مخازن کتابخانه‌ای اندروید نظیر ماون[۴۰] 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 اندروید صورت می‌گیرد و در نهایت از طریق شباهت‌سنجی، کدهای کتابخانه‌ای تشخیص داده می‌شوند. مشخص است که روش‌های مبتنی بر شباهت‌سنجی از دقت بیشتری ،خصوصا در صورت وجود ابهام، 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 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[۴۱] نزدیک‌ترین همسایه[۴۲] برای کاهش فضای مقایسه‌ی دودویی[۴۳] و با استفاده از ویژگی‌های مبتنی بر منبع، سرعت تشخیص بهبود داده‌شده است. 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[۴۴] در بسته‌های برنامک استخراج شده و امضا[۴۵]ی 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 ایستا 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

[۳۹] Similarity
[۴۰] Maven Repository
[۴۱] Classifier
[۴۲] Nearest Neighbor
[۴۳] Pairwise Comparison
[۴۴] Method
[۴۵] Signature

کتابخانه‌ها متشکل از ۴۵۳ کتابخانه‌ی اندرویدی جمع‌آوری شده از مخزن ماون می‌باشد. در نهایت پس از تشخیص کلاس‌های کتابخانه‌های اندرویدی و حذف آن‌ها از کد برنامک، کدهای مورد توسعه به عنوان ورودی برای مقایسه‌ی دودویی و طبقه‌بندی مورد تحلیل قرار می‌گیرند.

در ادامه‌ی این نگارش، در فصل ۲ به تعریف مفاهیم اولیه مورد نیاز این پژوهش می‌پردازیم. در فصل ۳ 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 آن خواهیم‌پرداخت.

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 یا به بیان دیگر متقلب، برنامک اصلی را دانلود کرده و پس از دیکامپایل[۱] کردن، به نوعی تغییر می‌دهد که منطق کلی برنامک، تغییری نمی‌کند. مبهم‌سازی یکی از ارکان اصلی در فرایند بازبسته‌بندی است 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[۱۲]:

۲ـ۱ـ۱ روش‌های ساده

راهکارهای موجود در این دسته عمدتاً 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 کامپایل و بسته‌بندی می‌کند.

[۱] Decompile

بازبسته‌بندی تنها موجب تغیر در امضاء توسعه‌دهنده‌ی برنامک و جمع‌آزما[۲] می‌شود. بنابراین 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[۳] تغییر می‌کند و ویژگی‌های مبتنی بر نحو[۴] ثابت باقی می‌مانند. 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 [۱۳] در این دسته می‌پردازیم:

- **تغییر نام شناسه[۵]ها:** تغییر نام شناسه‌های موجود در برنامک شامل نام کلاس‌ها، متدها و یا متغیرها[۶]ی موجود [۱۲]

```
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[۷] مورد استفاده از فایل‌های دکس[۸]، موجب کاهش سطح معناشناسی می‌شود.
- **فراخوانی غیرمستقیم:** یکی از روش‌های ساده‌ی تغییر گراف فراخوانی[۹]، استفاده از یک تابع واسط به عنوان تابع فراخواننده‌ی[۱۰] تابع اصلی است. در این حالت تابع اولیه یک تابع واسط و تابع واسط به صورت زنجیره‌ای تابع اصلی را فراخوانی می‌کند. بدنه‌ی تابع واسط در این حالت، بسیار ساده و شامل یک فراخوانی تابع اصلی است.

---

[۲] Checksum
[۳] Semantic
[۴] Syntax
[۵] Identifier
[۶] Variable
[۷] String
[۸] Dex Files
[۹] Call Graph
[۱۰] Caller

فقره **جابه‌جایی دستورات:** جابه‌حایی دستورات موجود در برنامک اصلی، یکی از روش‌های پرکاربرد توسط ایزارهای 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 می‌کتد 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 بیهوده عموماً دارای ساختارهای کنترلی[۱۱] و حلقه‌های نُپ[۱۲] هستند که تاثیری در روند اجرای برنامک ندارند. ذکر این نکته حائز اهمیت است که در صورتی که در ساختار کدهای بیهوده از شروط کنترلی مبتنی بر متغیرهای پویا[۱۳] استفاده شود آن‌گاه دیگر تحلیل ایستای برنامک‌های اندرویدی قادر به تشخیص این نوع از مبهم‌نگاری‌ها نیست.

- **واردساختن کدهای مرده:** یکی دیگر از روش‌های تغییر گراف‌های برنامه از جمله گراف جریان[۱۴]، 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[۱۵] از روش‌های دیگری است که در این سطح به وفور مورد استفاده قرار می‌گیرد.

---

[۱۱] Control's Statement
[۱۲] Nop
[۱۳] Dynamic Variables
[۱۴] Flow Graph
[۱۵] Debug File

۲ـ۱ـ۳ روش‌های خاکستری

روش‌های موجود در این دسته، مبتنی بر نحوِ 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**[۱۶]: بازتاب یکی از ویژگی‌های مهم و پیچیده‌ی زبان جاوا می‌باشد[۱۴] که امکان فراخوانی متدها 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[۱۷] رشته‌ی واسط مورد نظر، به طور کامل واسط فراخوانی‌شده را پنهان می‌کند.

```
1 Object object = new Object();
2 Method getService = Class.forName("android.os.
      ServiceManager").getMethod("getService",
      String.class);
3 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[۱۸]، کدهای اصلی را بارگیری[۱۹] 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**[۲۰]: زبان جاوا از قابلیت مهمی به نام بارگیری پویای کد پشتیانی می‌کند که اجازه می‌دهد تکه کدی را که پیش از این در کد مورد توسعه‌ی یک برنامک موجود نبوده را در حین اجرا به برنامک اضافه کنیم. مهاجم با استفاده از این قابلیت زبان جاوا می‌تواند قسمت‌هایی

[۱۶] Reflect
[۱۷] Encryption
[۱۸] Decryption
[۱۹] Load
[۲۰] Dynamic Class Loading

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 برای 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.از آن‌جایی که بسیاری از برنامک‌های تقلبی با استفاده از مبهم‌نگار[۲۱]ها توسعه یافته‌اند و علاوه بر این 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[۱۵] 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 تجاری[۲۲] می‌پردازیم.

- **پروگارد**

پروگارد[۲۳] 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 عمدتاً با استفاده روش‌های مشروحه در بخش ۲ـ۱ـ۳ انجام می‌شود.[۱۶]

- **آلاتوری**

آلاتوری[۲۴] یک مبهم‌نگار رایگان تولیدشده توسط شرکت روسی اسماردک[۲۵] می‌باشد که سطوح مختلفی از مبهم‌نگاری را با توجه به فایل‌های پیکربندی[۲۶] پوشش می‌دهد. این مبهم‌نگار از تغییرنام،

[۲۱] Obfuscator
[۲۲] Commercial
[۲۳] Proguard
[۲۴] Allatori
[۲۵] Smardec
[۲۶] Configuration

مبهم‌نگاری مبتنی بر تغییر گراف‌های جریان، مبهم‌نگاری فایل‌های اشکال‌زدایی و رمزنگاری داده‌های رشته‌ای پشتیبانی می‌کند.[۱۷، ۱۸]

- **دکس‌گارد**

این مبهم‌نگار نسخه‌ی تجاری نرم‌افزار پروگارد است که توسط شرکت گارداسکوار[۲۷] تولید شده‌است. دکس‌گارد[۲۸] 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 کامل انجام می‌دهد.

۲ـ۲ ساختار فایل‌های برنامک‌های اندرویدی

هر برنامک اندرویدی یک فایل فشرده‌شده با پسوند APK[۲۹] است که به اختصار شامل ۴ پوشه‌ی مهم و ۳ فایل می‌باشد.برای درک بهتر از روش پیشنهادی این پژوهش، در ادامه هر کدام از این قسمت‌ها را معرفی و کارکرد آن را بررسی خواهیم‌کرد.[۱۹]

- **پوشه‌ی res:** این پوشه شامل منابع برنامک‌های اندرویدی است که مربوط به رابط کاربری برنامک می‌شود. این پوشه در نهایت به فایل‌های R. نگاشت شده و هر کدام از منابع با یک شناسه[۳۰] مشخص می‌گردد.

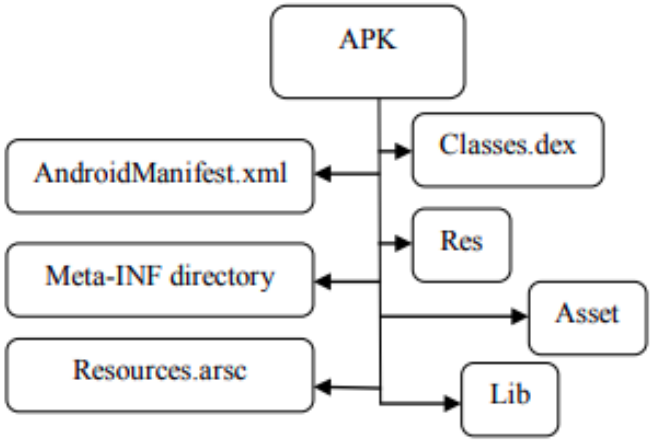
- **پوشه‌ی lib:** فایل‌های کامپایل‌شده‌ی بومی در این پوشه قرار می‌گیرند که شامل کتابخانه‌های اندرویدی و جاواییینز می‌گردد. استفاده از فایل‌هایی که کامپایل شده‌اند سرعت اجرای برنامک‌های اندرویدی را بالا می‌برد لذا استفاده از آن‌ها به عنوان بسته‌های[۳۱] از پیش آماده محبوبیت دارد.

- **فایل Classes.dex:** فایل‌های با پسوند dex فایل‌های دودویی[۳۲] 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[۳۳] اجرا می‌شود.

---

[۲۷] Guardsquare
[۲۸] DexGaurd
[۲۹] Android Package
[۳۰] Resource Id
[۳۱] Module
[۳۲] Binary
[۳۳] Dalvik Virtual Machine

- **فایل AndroidManifest.xml:** پیکربندی‌های مهم فایل‌های APK از جمله لیست مجوزهای مورد نیاز، لیست مولفه‌ها[۳۴] 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[۳۵] کاربر توسعه‌دهنده‌ی برنامک است که برنامک با کلید خصوصی متناظر آن امضا شده‌است. امضای موجود در این پوشه، خاصیت صحت‌سنجی[۳۶] دارد اما اطلاعاتی را از توسعه‌دهنده نشر نمی‌دهد و به صورت خودامضا ساخته می‌شود.
- **فایل resources.arsc:** این فایل برای انجام نگاشت[۳۷] میان منابع موجود در پوشه‌ی `resources` و شناسه‌ی هر منبع استفاده می‌شود تا بتوان در حین اجرای برنامک‌ها، هر شناسه را به منبع آن ترجمه کرد.



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

---

[۳۴] Component
[۳۵] Public Key
[۳۶] Integrity
[۳۷] Mapping

۲ـ۳ کتاب‌خانه‌های اندرویدی

کتابخانه‌های اندرویدی، بسته‌های از پیش توسعه‌یافته هستند که توسط توسعه‌دهندگان نوشته شده و توسعه‌دهندگان اندروید به جهت سهولت در پیاده‌سازی و کمک به تسریع توسعه‌ی نرم‌افزار 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[۲۰]. 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 کتاب‌خانه‌های اندرویدی و اشتراکات موجود در آن‌ها است چرا که بسیاری از کتابخانه‌ها خصوصا کتاب‌خانه‌های زبان جاوا، در هر برنامک اندرویدی موجود است. از طرفی، به دلیل عدم وجود مرز مشخصی میان کدهای کتابخانه‌ای و کدهای مورد توسعه توسط توسعه‌دهندگان، شناسایی کتابخانه‌های اندرویدی تبدیل به یک چالش در زمینه‌ی تشخیص برنامک‌های بازبسته‌بندی در این حوزه شده‌است.

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 الگوریتم[۳۸]ها است که برای طبقه‌بندی و یا مرتب‌سازی[۳۹] داده‌های ورودی مورد استفاده قرار می‌گیرد [۲۱]. یکی از ساده‌ترین مثال‌های موجود برای طبقه‌بندی، جداسازی هرزنامه[۴۰]ها در سرویس‌های ایمیل است.روش‌های طبقه‌بندی نیازمند مجموعه‌ای از ویژگی‌های اطلاعات مورد بررسی به عنوان ورودی مسئله می‌باشند تا پس از اجرای الگوریتم، اطلاعات مسئله را بر اساس آن‌ها طبقه‌بندی کنند.

---

[۳۸] Algorithm
[۳۹] Sorting
[۴۰] Spam

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 [۲۲، ۲۳] 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 [۲۴، ۲۵] باربسته‌بندی مبتنی بر تغییر ویژگی‌های کدپایه تعریف شده‌است.البته که نمی‌توان به هیچ کدام از تعاریف بالا خرده گرفت چرا که هر دو تعریف از نظر مهاجم و اهداف تعریف شده توسط اون قابل استناد است. علاوه بر این یکی دیگر از اختلافات موجود در تعریف بازبسته‌بندی، وجود مبهم‌نگاری در برنامک‌های بازبسته‌بندی شده‌است. برخی از پژوهش‌ها نظیر [۲۶] بازبسته‌بندی را منطوط به تغییر در امضای برنامک می‌دانند اما بسیاری از پژوهش‌های به‌روزتر، نظیر [۲۷، ۲۸] بازبسته‌بندی را تنها به نغییر 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 تعریف 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 صحیح 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 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 منفی 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 عبور 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 پژوهش را 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 [۱] 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 [۲۹، ۳۰، ۳۱] از روش‌های مبتنی بر مدل‌های یادگیری ماشین[۲]

[۱] Decision
[۲] Machine Learning

برای تشخیص برنامک‌های بازبسته‌بندی شده استفاده کرده‌اند. حال آن‌که پژوهش‌های مرتبط با تعریف ۱، نظیر [۳۲، ؟] بیشتر از روش‌های مقایسه‌ی دودویی و مبتنی بر شباهت‌سنجی استفاده کرده‌اند.
تعریفی که در این پژوهش مورد استفاده قرارگرفته است، تعریف ۱ است. یعنی تشخیص بازبسته‌بندی منوط به **تشخیص 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 مشخص 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[۳] 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[۴] 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

[۳] Pre process
[۴] Structural

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.غالب پژوهش‌های مبتنی بر استفاده از لیست سفید، فرض کرده‌اند که تنها کدهای مورد توسعه توسط متقلب مبهم‌نگاری شده‌است و ابهام در کدهای کتابخانه‌ای را نادیده گرفته‌اند.

- **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 چرا که در صورتی که ویژگی‌های منتخب مقابل مبهم‌نگاری مقاوم باشند، آن‌گاه می‌توان گفت که درصد بالایی از کتابخانه‌های اندرویدی را می‌توان از کد اصلی برنامک جدا کرد.

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[۵] 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 روش‌های تشخیص بازبسته‌بندی 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 دی‌کامپایل انجام می‌شود مقاوم هستند. اما همانطور که می‌توان

[۵] Feature Extracting

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[۶] که در برنامک‌های اندرویدی صورت می‌گیرد، می‌باشد. به همین علت روش‌های موجود در این حوزه، عمدتا برنامک‌ها را در هنگام اجرا بررسی و استخراج ویژگی عمدتا در هنگام اجرا انجام می‌گیرد. به طول کلی، روش‌های مبتنی بر تحلیل پویا از مقاومت بیشتر در مقابل استفاده از راهکارهای مبهم‌نگاری برخوردار هستند. استفاده از شبیه‌سازهای جعبه‌شن[۷] به وفور در پژوهش‌های این حوزه، یافت می‌شود. یکی از چالش‌های اصلی در تشخیص برنامک‌های اندرویدی بازبسته‌بندی شده، چگونگی پیاده‌سازی شبیه‌ساز ها[۸]ست. بسیار از شبیه‌سازها 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 شبیه‌سازی‌شده را دارند و ممکن است تمامی قابلیت‌های خود و یا بخشی از آن را به جهت دور زدن سیستم‌های تشخیص پویا، پنهان کنند.

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 مشخص 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

[۶] Execution Time
[۷] Sand Box
[۸] Simulator

روش‌ها پوشش گشترده‌تر 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 تحلیل ایستا و 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 در برنامک‌های اندرویدی بازبسته‌بندی شده پشتیبانی کرد.

روش‌های مبتنی بر آپکد و دستورات

استفاده از آپکد[۹]های موجود در فایل‌های دالویک، یکی از روش‌های تشخیص برنامک‌های بازبسته‌بندی شده است. هدف از پژوهش آقای ژو [۳۳] و همکاران، توسعه‌ی ابزاری به نام درویدمس[۱۰] بوده است

[۹] Opcode
[۱۰] DroidMoss

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[۱۱] استفاده‌شده است. پس از جداسازی کدهای کتابخانه‌ای به وسیله‌ی لیست سفید و استخراج آپکدها از فایل‌های دالویک، از یک پنجره‌ی لغزات[۱۲] روی آپکدها استفاده شده و در نهایت چکیده**Hash**ی آپکدها به همراه امضای دیجیتال برنامه‌نویس، موجود در پوشه‌ی META-INF تشکیل امضای برنامک را می‌دهند. همانطور که می‌توان فهمید، فرض پژوهش این بوده است که کلید خصوصی توسعه‌دهنده لو نرفته‌است. در نهایت برای قسمت شباهت‌سنجی، از الگوریتم فاصله ویراشی[۱۳] 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[۲۵] ارائه شده‌است، هدف پژوهش، 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[۱۴] و اطلاعاتی در مورد تعداد فایل‌های برنامک، توصیفات آن و چکیده‌ی آپکدهای دستورات برنامه است. این پژوهش با استفاده از یک مرحله پیش‌پردازش شامل بررسی فایل فراداده‌ی برنامک‌های موجود،فضای جست‌وجوی دودویی برنامک‌های مورد مقایسه را کاهش می‌دهد.دزنوز و همکاران[۳۴] روش‌دیگری را مبتنی بر شباهت‌سنجی روی آپکدها با استفاده از فاصله‌ی فشرده‌سازی نرمال‌شده ارائه کرده‌اند. در این پژوهش ابتدا برای هر متد با توجه به دنباله‌ی دستورات موجود امضای مشخصی تولید می‌شود و در مرحله‌ی بعد متدهایی که بکتا هستند از هر دو برنامک، بر اساس معیار فاصله‌ی فشرده‌سازی نرمال‌شده با یکدیگر مقایسه و بدین ترتیب متدهای مشابه استخراج می‌شود. در پژوهش [۳۵، ۳۶] پس از استخراج هیستوگرام‌های مربوط به تکرار آپکدها در قسمت‌های مختلف برنامک، هیستوگرام‌ها با استفاده از معیار فاصله‌ی مینی‌کاوسکی که یک معیارفاصله‌ی مبتنی بر هیستوگرام‌ها است مقایسه می‌شوند و در نهایت برنامک‌های بازبسته‌بندی شده مشخص می‌شوند. جرومه و همکاران[۳۷] در پژوهش خود با استفاده از آپکدها و تکرار آن‌ها و روش‌های مبتنی بر یادگیری ماشین برنامک‌های بازبسته‌بندی شده را تشخیص می‌دهند.در پژوهشی که توسط [۳۸] و همکاران، ارائه‌شده است، از نمونه‌برداری مبتنی بر n-gram در ۴ اندازه‌ی متفاوت ۱ تا ۴ استفاده شده‌است.

[۱۱] MetaData
[۱۲] Sliding Window
[۱۳] Edit Distance
[۱۴] Manifest

برای شباهت‌سنجی از روش‌های طبقه‌بندی مبتنی بر درخت‌تصمیم، شبکه‌های عصبی و بردار ماشین استفاده شده‌است.

آقای لین و همکاران[۳۹] در این پژوهش، با استفاده از فراخوانی‌های سیستمی صدا زده‌شده توسط برنامک، رفتار آن را طبقه‌بندی می‌کنند. به عقیده‌ی این پژوهش، از آن‌جایی که اکثر بدافزارهای هم‌خانواده، در بازبسته‌بندی برنامک‌های اندرویدی، رفتار مشابه یکدیگر دارند، بنابراین استفاده از فراخوانی‌های سیستم و اسخران آن‌ها از سطح دالویک بایت‌کدها و سطح نخ، می‌تواند امضاء یکتایی از هر برنامک تولید کند. پس از استخراج بردار ویژگی از فراخوانی‌های موجود با استفاده از آپکدهایر برنامک، از یک طبقه‌بندی بیز برای شباهت‌سنجی استفاده شده‌است. با توجه به روش پژوهش، شناسایی و طبقه‌بندی بدافزارهای بازبسته‌بندی شده‌ای که رفتار مشخصی ندارند و در مخزن بدافزارها موجود نیستند، یکی از ویژگی‌های مفید پژوهش ارائه‌شده است. فروکی [۴۰] و همکاران در پژوهش خود یک راه حل مبتنی بر استفاده از بلاک‌های ۶۴ بایتی روی فایل‌های دودویی برنامک‌های اندرویدی ارائه‌کرده اند. در روش ارائه‌شده پس از استخراج بلاک‌های ۶۴ بایتی از فایل‌هاو با استفاده از چکیده خلاصه تشابه و استخراج آنتروپی برای هر بلاک، بلاک‌هایی که کوچکتر و بزرگتر از یک حد کمینه و آستانه باشند حذف می‌شوند. سپس به هر بلوک با توجه اه آنتروپی آن، یک اولویت اختصاص پیدا کرده که نشان می‌دهد بلوک مورد نظر دارای آنتروپی با احتمال بیشتر است. در نهایت پس از حذف بلوک‌هایی که احتمال رخداد پایین‌تری دارند نرخ مثبت غلط پژوهش را کاهش یافته و از یک روش مبتنی بر بلوم‌فیلتر برای مقایسه و شباهت‌سنجی استفاده می‌شود.

آقای کو و همکاران[۴۱] از یک راه‌حل مبتنی بر استفاده از k-gram برای تشیخص بسته‌های بازبسته‌بندی شده استفاده کرده‌اند. در این پژوهش، از حذف عملوندهای موجود در کدهای دودویی، به جهت کاهش مثبت‌های غلط در تشخیص بسته‌های بازبسته‌بندی شده استفاده شده‌است. در پژوهشی [۴۲]، کیشو و همکاران از یک راه‌حل مبتنی بر ترکیبی از دستورات کلاسی و متدهای برنامک استفاده کرده‌اند. در این پژوهش، در دو مرحله، ابتدا کلاس‌های مشابه با یکدیگر مشخص می‌شود و سپس در داخل کلاس‌های مشابه، متدهایی که یکسان هستند یافت می‌شود. شباهت‌سنجی میان کلاس‌ها، با استفاده از سه ویژگی، شامل لیست تمامی متدهای کلاس شامل ورودی و خروجی، لیست متغیرهای کلاسی و لیست کلاس‌هایی که داخل این کلاس فراخوانی شده‌اند، انجام می‌شود. پس از استخراج کلاس‌های مشابه، برای یافتن متدهای مشابه میان دو کلاس، از یک امضای مشترک شامل توصیف متدها به همراه نوع ووردی و خروجی آن‌ها و همچنین نام آن‌ها استفاده می‌شود. شباهت‌سنجی با استفاده از فاصله‌ی فشرده‌سازی شده انجام شده‌است.

راهول و همکاران[۴۳]، روشی را پیشنهاد کرده‌اند که در آن استخراج ویژگی مبتنی بر درخت نحو انتزاع انجام می‌شود. پژوهش پس از دستیابی به کد میانی برنامک‌های اندرویدی و تبدیل آن به مجموعهذای از قوانین نحوی، که به صورت مجموعه‌ای از عبارات منظم هستند، درخت سطح انتزاع را در سطح تابع

تشکیل می‌دهد و سه ویژگی تعداد ورودی هر تابع، نام توابع صدازده‌شده به صورت مستقیم و مجازی و متغیرهای شرطی را استخراج می‌شود. سپس برای طبقه‌بندی از الگوریتم نزدیک‌ترین همسایه برای تشخیص بازبسته‌بندی استفاده شده‌است. نرخ منفی غلط بسیار پایین از ویژگی‌های مورد توجه این پژوهش است. همچنین برای ذخیره‌سازی درخت نحو انتزاع، از یک ساختار مبتنی بر B+ و پایگاه‌داده‌ی MySql استفاده شده‌است.

به صورت کلی می‌توان گفت که روش‌های مبتنی بر دستورات، خصوصا روش‌هایی که به صورت مستقیم از آپکد برای تشخیص برنامک‌های بازبسته‌بنده شده استفاده می‌کنند، توانایی بالایی را در تشخیص برنامک‌های بازبسته‌بندی شده ارائه نمی‌دهند. این روش‌های هم‌اکنون مقابل ساده‌ترین مبهم‌نگاری ها نظیر تغییر نام بسته‌ها و کلاس‌ها مقاوم نیستند و بخش زیادی از پژوهش‌های این حوزه، بازبسته‌بندی را بدون تغییر در کدهای برنامک اصلی تعریف کرده‌اند که به نظر و با توجه به وجود مبهم‌نگارهای امروزه، این فرضی غلط و غیر قابل اتکا است.

۳ـ۳ـ۲ روش‌های مبتنی بر گراف

در پژوهشی که توسط آقای کروسل[۴۴] و همکاران، ابزاری مبتنی بر گراف وابستگی توسعه‌داده شده‌است. در این ابزار ابتدا، برنامک‌های موجود در مخزن با استفاده از یک ابزار شباهت‌سنجی در سطح فراداده‌ی برنامک، به جهت افزایش سرعت، کاهش می‌یابد. پس از حذف کدهای کتابخانه‌ای به روش لیست سفید، امضاء هر برنام با استفاده از گراف وابستگی استخراج‌شده تشکیل می‌گردد. گراف وابستگی توابع، وابستگی اجزای یک تابع از دو منظر کنترلی و داده‌ای معرفی می‌کند. وابستگی کنترلی در این پژوهش، الزام اجرای یک دستور خاص پیش از دستور دیگری است و وابستگی داده‌ای، الزار مقداردهی متغیر پیش از اجرای دستور مرتبط با آن است. در قسمت شباهت‌سنجی پس از ساخت گراف وابستگی داده‌ای، با استفاده از الوریتم گراف‌های هم‌ریخت VF2 شباهت‌سنجی انجام شده و بسته‌های بازبسته‌بندی شده مشخص می‌شوند. در پژوهش دیگری که توسط آقای سان[۷] انجام شده‌است، هدف پژوهش افزایش دقت تشخیص برنامک‌های بازبسته‌بندی شده با تاکید بر شبیه‌سازی رفتار برناک بوده‌است. در این پژوهش، واسطهای برنامه‌نویسی برنامک‌های اندرویدی مشخص کننده‌ی رفتار اصلی برنامک در نظر گرفته‌شده است. برای ساخت گراف هر برنامک، از گراف جریان مبتنی بر فراخوانی واسطهای اندرویدی استفاده شده و در نهایت پس از استخراج گراف، هر گراف نمایانگر امضاء یک برنامک می‌باشد. در گراف حاصل هر نود گراف حاوی اطلاعات یک واسط و یال‌های گراف شامل جریان کنترلی بین واسطهای اندرویدی است. برای شباهت‌سنجی، از الگوریتم VF2 برای تشخیص گراف‌های هم‌ریخت استفاده شده‌است. در مرحله‌ی آخ برنامک‌هایی که امضاء مشابه و یکسانی در هم‌ریختی دریافت کرده‌اند به عنوان برنامک‌های بازبسته‌بنده شده در نظر گرفته

می‌شوند.

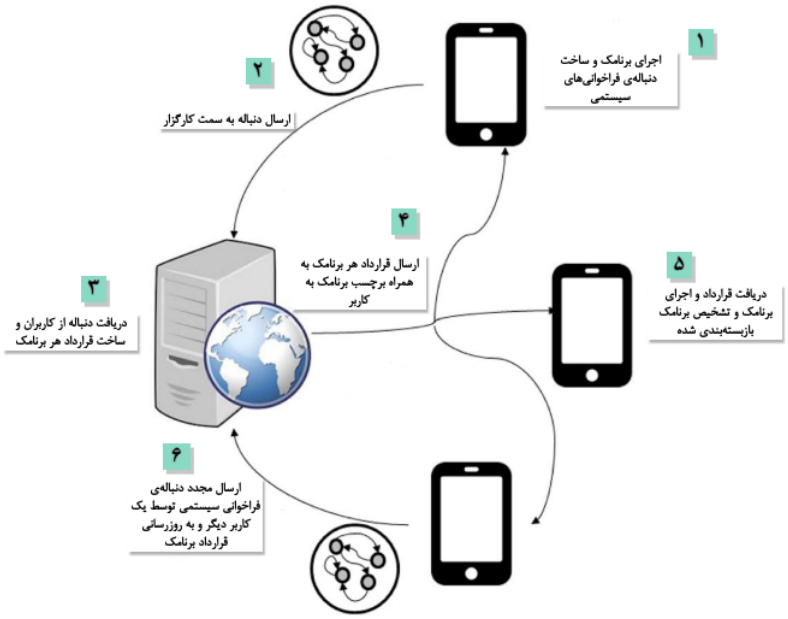
پژوهش آقای هو و همکاران[۴۵] شامل دو مرحله‌ی ساخت گراف فراخوانی متدهای برنامک و ماژول تشخصی بازبسته‌بندی است. پس از دیس‌اسمیل کردن فایل‌های برنامک، گراف فراخوانی متدهای برنامک تشکیل شده و تشکیل جنگلی از گراف‌های متصل و جدا از هم می‌دهند.سپس با استفاده از فراخوانی واسط‌های اندرویدی موجود در هر متد، گراف به دو بخش فراخوانی‌های حساس و غیر حساس تقسیم می‌شود و با توجه به میزان حساسیت واسط‌های فراخوانی‌شده، امتیاز اولویت به هر گراف نگاشت می‌شود و در نهایت با استفاده از مقایسه‌ی گرافی مبتنی بر امتیاز اولویت، شباهت‌سنجی انجام می‌شود.

از آن‌جایی که پژوهش‌های مبتنی بر گراف در تشخیص برنامک‌های بازبسته‌بندی عمدتا از روش‌های تشحیص گراف‌های همریخت استفاده می‌کنند، ژو و همکاران [۴۶] روشی برای افزایش سرعت در تشخیص ارائه کرده‌اند. در این پژوهش در ابتدا ماژول‌های اصلی برنامک که رفتار اصلی برنامک را تشکیل می‌دهند شناسایی می‌شود. برای شناسایی ماژول‌های اصلی برنامک، از یک گراف جهت‌دار مبتنی بر ارتباط بسته‌های برنامک با یکدیگر استفاده شده و در نهایت یال‌های گراف بر اساس میزان ارتباطات بین بسته‌ها، مقداردهی می‌شود. با تشکیل گراف وزن‌دار ابتدایی، بسته‌هایی که ارتباط آن‌ها بر اساس وزن یال بین دوبسته از یک مقدار آستانه بیشتر باشد، با یکدیگر ادغام می‌شوند و رویه‌ی ادغام بسته‌ها در یک روند بازگشتی تکرار می‌شود تا زمانی که هیچ بسته‌ای را نتوان با یکدیگر ادعام کرد. در این حالت بسته‌ی نهایی شامل بسته‌ی اصلی برنامک است که رفتار برنامک بر اساس آن مشخص می‌شود. رویه‌ی ساخت گراف ارتباطی بین بسته‌ها و ایده‌ی استفاده شده در قسمت ادغام بسته‌های اصلی با یکدیگر، ایده‌ای نو در این حوزه است که منجر به افزایش سرعت تشخیص نسبت به تمامی روش‌های گرافی شده‌است. برای مقایسه‌ی میان ماژول‌های اصلی برنامک، ابتدا اصلی‌ترین ماژول شامل بیشترین تعداد فعالیت، مشخص می‌شود و مقایسه میان ماژول‌های اصلی برنامک‌های اندرویدی، با استفاده از یک بردار ویژگی متشکل از فراخوانی واسط‌ها و مجوزهای درخواستی انجام می‌شود. برای مقایسه از درخت VP استفاده شده‌است که منجر به افزایش سرعت در کنار دقت تشخیص مناسب شده‌است. برخلاف روش‌های معمول گرافی و در نهایت مقایسه‌ی دودویی، روش پیاده‌سازی شده در پژوهش ژو، با مرتبه‌ی زمانی nlgn یکی از پر سرعت‌ترین روش‌های مبتنی بر تشکیل گراف می‌باشد.

از آن‌جایی که تشخیص برنامک‌های بازبسته‌بندی شده به وسیله‌ی مقایسه‌ی گرافی، نیازمند الگوریتم‌های تشخیص گراف‌های هم‌ریخت است که معمولا سرعت بسیار پایینی دارند، چن و همکاران[۴۷] با استفاده از مدل کردن گراف به یک فضای سه‌بعدی، سرعت تشخیص و مقایسه‌ی گراف‌های هم‌ریخت را به مراتب افزایش داده‌اند. ایجاد کدهای مرده در کدهای برنامک، منجر به تغییر گراف جریان برنامک‌های اندرویدی می‌شود به همین جهت در این پژوهش تمامی گره‌های گرافی که نشان‌دهنده‌ی متدهای برنامک هستند به یک فضای سه‌بعدی نگاشت شده و مرکز جرم هر گراف با توجه به مختصات گره‌های گرافی تعیین می‌شود.

در قسمت مقایسه‌ی گراف‌ی، مرکز جرم گراف‌های متناظر با یکدیگر مقایسه شده و کاندیدهای بازبسته‌بندی مشخص می‌شود. در مرحله‌ی بعد برای مقایسه‌ی گره‌های هر گراف و تطبیق گراف‌های کاملا متناظر، از مقایسه‌ی فاصله‌ی گره‌های متناظر استفاده می‌شود. روش ارائه‌شده علاوه بر مقاومت بالا مقابل مبهم‌نگاری، ناشی از مدل‌کردن برنامک در یک فضای گرافی، به دلیل استفاده از روشی نو در مقایسه‌ی گراف‌های مخزن برنامک‌ها، سرعت بسیار بالاتری از روش‌های پیشین دارد. برای حذف کتاب‌خانه‌های اندرویدی از روش لیست سفید مبتنی بر اندازه‌ی بسته‌های مورد مقایسه استفاده شده‌است.

بر خلاف پژوهش‌های رایج در حوزه‌ی تشخیص بازبسته‌بندی برنامک‌های اندرویدی، در پژوهش آقای آلدینی و همکاران[۴۸]، از یک معماری کارخواه و کارگزار برای تشحیص برنامک‌های اندرویدی بازبسته‌بندی شده استفاده شده‌است. در این معماری یک برنامک بر روی دستگاه اندرویدی کاربران نصب می‌شود و شروع به ثبت و ارسال فراخوانی‌های سیستمی به کارگزار می‌کند.

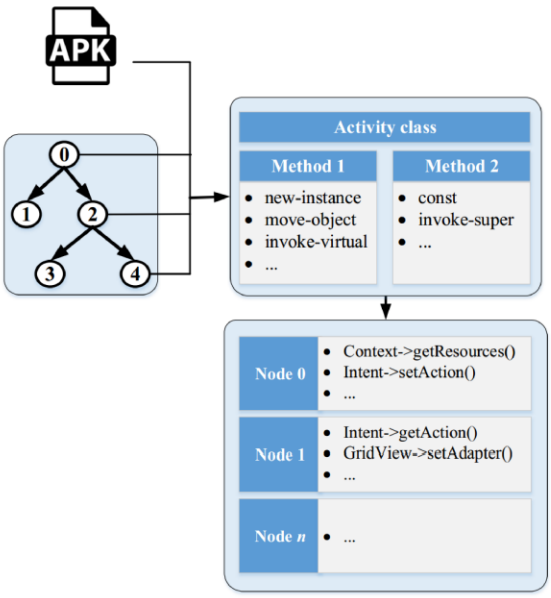


شکل ۳ـ۱: مراحل تشخیص برنامک‌های بازبسته‌بندی شده در پژوهش آقای آلدینی

ایده‌ی پژوهش، استفاده از فراخوانی‌های سیستمی برای شبیه‌سازی ایستای رفتار برنامک‌های اندرویدی بوده است. اثرانگشت برنامک توسط گراف فراخوانی‌های سیستمی ارسالی از سمت کاربران در سمت کارگزار، انجام می‌شود سپس با استفاده از اثرانگشت موجود، یک مدل برنامک تحت عنوان قرارداد ساخته شده و این مدل به برنامه‌ی کارخواه فرستاده می‌شود. در سمت کارخواه، برنامک نصب‌شده در مرحله‌ی اول، فراخوانی‌های سیتسمی برنامک موجود را با قرارداده فرستاده‌شده مطابقت می‌دهد و در صورتی که فاکتورهایی نظیر نوع و تعداد فراخوانی‌های اندرویدی برنامک با مدل پیش‌بینی شده یکسان نباشد، هشدار لازم از طریق برنامک کارخواه به کاربر داده می‌شود. در کنار استفاده از گراف فراخوانی‌های سیستمی،

پیاده‌سازی یک معماری کارخواه و کارگزار یکی از ایده‌های اولیه‌ی این پژوهش بوده‌است. همچنین به دلیل استفاده از این معماری، پردازش سمت کارخواه به حداقل خود رسیده است و تحلیل برنامک نیازمند هیچ دانش اولیه‌ای از سمت کارخواه نمی‌باشد. علاوه بر این، تعداد برنامک‌های موجود در مخزن کارگزار، به صورت مرتب افزایش پیدا کرده و این موجب پویایی مخزن برنامک‌های اندرویدی پژوهش می‌شود.

پژوهش دیگری در زمینه‌ی روش‌های تشخیص بازبسته‌بندی مبتنی بر گراف توسط جنگ و همکاران[۴۹] ارائه شده‌است. در این پژوهش، پس از استخراج گراف ارتباط بین کلاس‌ها از داخل دالویک بایت‌کدهای برنامک اندرویدی، امضای هر برنامک شامل یک بردار ویژه، متشکل از فراخوانی‌های کلاسی برنامک می‌باشد. مقایسه‌ی بردارهای ویژه‌ی برنامک‌های موجود به وسیله‌ی الگوریتم یافتن بزرگ‌ترین زیردنباله‌ی مشترک انجام شده و تعیین یک حد آستانه در زیردنباله‌های مشترک، برنامک بازبسته‌بندی شده شناسایی می‌شود. روش مورد نظر را می‌توان به نوعی یک روش مبتنی بر گراف و دنباله‌ی آپکدهای برنامک توصیف کرد چرا که از هر دو ایده‌ی مدنظر استفاده نموده‌است. یکی دیگر از آخرین پژوهش‌های موجود در این دسته،در سال ۲۰۲۱ توسط نگویان[۲] مطرح شده‌است. پژوهش مورد نظر مبتنی بر استخراج گراف فعالیت در برنامک‌های اندرویدی است. گره‌های گراف مورد نظر شامل لیستی از واسطهای فراخوانی‌شده در آن فعالیت و یال‌های گراف، نشان‌دهنده‌ی یک انتقال از یک فعالیت به فعالیتی دیگراست. یکی از معایب این پژوهش استفاده از الگوریتم VF۲ به عنوان الگوریتم اصلی برای مقایسه و یافتن گراف‌های هم‌ریخت است که باعث کاهش سرعت پژوهش می‌شود.



شکل ۳ـ۲: هر گره از گراف در پژوهش نگویان شامل لیستی از واسطهای فراخوانی شده در آن فعالیت است.[۲]

به عنوان آخرین پژوهش صورت گرفته در این قسمت، پژوهش[۵۰، ۵۱] را بررسی خواهیم کرد.

پژوهش [۵۱] که الهام‌دهنده‌ی پژوهش [۵۰] می‌باشد با استفاده از پیمایش گرافی برروی گراف جریان برنامک‌های اندرویدی و استخراج ویژگی‌های متعدد نظیر واسط‌های اندرویدی، فراخوانی توابع و فیلترهای ساختاری نظیر اندازه‌ی طول امضا، امضای هر کلاس را تشکیل می‌دهد و امضاء هر برنامک حاصل از الحاق تمامی کلاس‌های برنامک (مرتب شده به صورت الفبایی) می‌باشد. تفاوت این دوپژوهش بیشتر در ساختار امضاء هر برنامک می‌باشد که به تفضیل در فصل ۴ به تفضیل بیشتر این دو پژوهش و شرح تفاوت‌های آن خواهیم پرداخت.

به طور کلی می‌توان گفت که روش‌های گرافی تشخیص برنامک‌های بازبسته‌بندی شده از دقت بالایی در تشخیص برخوردار هستند اما اکثر روش‌های ارائه‌شده در این دسته، خصوصاً آن‌هایی که در نهایت هر برنامک را به یک طرح گرافی مدل می‌کنند، روش‌های تشخیص گراف‌های هم‌ریخت را در قسمت مقایسه به کار گرفته‌اند که این موضوع باعث می‌شود تا سربار محاسباتی سنگینی به پژوهش‌های مطرح وارد و سرعت تشخیص را کند سازد. علاوه بر این همانطور که بررسی شد روش‌های ارائه‌شده، در صورتی که مدل‌هاگرافی را در فضای دیگری بررسی کنند، سرعت تشخیص پژوهش بالاتر رفته و می‌توان از دقت در تشخیص مسایل گرافی نیز استفاده نمود.

۳ـ۳ـ۳ روش‌های مبتنی بر تحلیل ترافیک شبکه و فراخوانی سیستمی

در این قسمت به بررسی پژوهش‌های صورت‌گرفته در حوزه‌ی تشخیص برنامک‌های بازبسته‌بندی شده مبتنی بر تحلیل ترافیک شبکه می‌پردازیم. به عنوان اولین پژوهش مورد بررسی، پژوهش ارائه‌شده توسط وو[۵۲] و همکاران را بررسی خواهیم کرد. در این پژوهش امضای هر برنامک مبتنی بر ترافیک http تولیدشده توسط آن درست می‌شود. در مرحله‌ی اول این پژوهش، تمامی ترافیک تولیدشده توسط برنامک جمع‌آوری شده و در مرحله‌ی بعدی ترافیک http جداسازی و تحلیل روی این دسته ادامه پیدا می‌کند. ترافیک حاصل از برنامک‌های اندرویدی در این پژوهش به دو دسته‌ی کلی تقسیم می‌شود:

- ترافیک مرجع: ترافیک تولیدشده توسط برنامک توسعه‌یافته
- ترافیک کتابخانه‌ای: ترافیک تولیدشده توسط کدهای کتابخانه‌ای

جهتر جداسازی ترافیک مرجع و ترافیک کدهای کتاب‌خانه ای از الگوریتم‌های تطبیق جریان ترافیک http و الگوریتم تطبیق هانگرین استفاده شده‌است. در قسمت شباهت‌سنجی از درخت جست‌وجوی VPT به جهت افزایش سرعت از جهت توازن درخت، استفاده شده است. پژوهش مورد نظر ایده‌ای نو در زمینه‌ی تشخیص برنامک‌های بازبسته‌بندی شده است اما مشکل اصلی این پژوهش آن است که در مقابل ترافیک

رمزنشده مقاوم نیست و نمی‌توان از این روش استفاده کرد. در پژوهشی که توسط الشهری[۵۳] ارائه‌شده است تفکیک ترافیک به وسیله‌ی یک طبقه‌بند انجام می‌شود. پس از تفکیک ترافیک متغیرهای هر بسته شامل Request, Value, Get, Host مشخص می‌شود. ترافیک‌های از نوع Request در این قسمت به دو نوع اجباری و یا غیراجباری تقسیم می‌شود. جداسازی ترافیک‌های شبکه از آن جهت اهمیت دارد که بسیاری از کتاب‌خانه‌های رایگان و یا حتی بدافزارهای موجود، یک نقطه‌ی دسترسی توسط واسط‌های برنامه‌نویسی برای دسترسی به کتابخانه‌ها ایجاد کرده‌اند که برنامک‌های اندرویدی به وفور از این درگاه‌ها استفاده می‌کنند به جهت جلوگیری از رخداد منفی غلط زیاد در تشخیص جداسازی ترافیک اجباری از غیر اجباری، موجب افزایش دقت تشخیص می‌شود. برای شباهت‌سنجی ترافیک اجباری شامل ترافیک اصلی برنامک، از یک الگوریتم فاصله‌ـ درخواست که مبتنی بر فاصله‌ی اقلیدوسی است مورد استفاده قرار گرفته‌است. پیشنهاد جداسازی ترافیک کتابخانه‌ای و اصلی به وسیله‌ی نقطه‌های دسترسی عمومی ایده‌ای نو در این پژوهش است که هر چند کامل نمی‌تواند ترافیک شبکه را جداسازی کند اما در صورت تکامل می‌تواند دقیق‌تر عمل کند.

در پژوهش دیگری که توسط آقای هه[۵۴] ارائه شده، از یک طبقه‌بند به جهت تشخیص برنامک‌های بازبسته‌بندی شده استفاده شده است. در پژوهش اخیر، ابتدا تمامی ترافیک کاربران متصل به یک شبکه به یک کارگزار سطح فرستاده می‌شود و این کارگزار ترافیک شبکه را برچسب‌زنی کرده و پس از استخراج ویژگی‌هایی نظیر محتوای اطلاعات هر بسته آن‌ها را برای محاسبه سمت یک کارگزار مرکزی که به صورت ابری خدمات ارائه می‌کند می‌فرستد. پس از ارسال ویژگی‌های مستخرج به سمت کارگزار ابری، فرایند شباهت‌سنجی آغاز می‌گردد. به جهت حفظ کامل حریم خصوصی کاربران، تحلیل ترافیک تنها برروی ترافیک رمزگذاری نشده(http) انجام می‌شود. در سمت کارگزار، با استفاده از تحلیل ترافیک شبکه‌ی هر کاربر، جریان اطلاعات در ترافیک برچسب‌زده شده مشخص می‌شود و سپس با حذف قسمتی از ترافیک هر جریان به جهت کاهش اندازه، مانند پاسخ درخواست‌ها و قسمتی از سربرگ ترافیک، طبقه‌بندی صورت می‌گیرد. برای طبقه‌بندی از الگوریتم پرسرعت پژوهش [۵۵] استفاده شده‌است.

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] N. T. Cam, N. H. Khoa, T. T. An, N. P. Bach, and V.-H. Pham. Detect repackaged android applications by using representative graphs. In *2021 8th NAFOSTED Conference on Information and Computer Science (NICS)*, pages 102–106, 2021.

[3] 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].

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

[5] 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].

[6] 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].

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

[15] 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.

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

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

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

[19] 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.

[20] 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.

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

[22] 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.

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

[24] 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.

[25] 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.

[26] 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.

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

[28] 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.

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

[30] 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.

[31] 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.

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

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

[34] A. Desnos. Android: Static analysis using similarity distance. pages 5394–5403, 2012.

[35] B. B. Rad and M. Masrom. Metamorphic Virus Detection in Portable Executables Using Opcodes Statistical Feature. *International Journal on Advanced Science, Engineering and Information Technology*, 1(4):403, 2011.

[36] B. B. Rad, M. Masrom, and S. Ibrahim. Opcodes histogram for classifying metamorphic portable executables malware. pages 209–213, 2012.

[37] Q. Jerome, K. Allix, R. State, and T. Engel. Using opcode-sequences to detect malicious android applications. In *2014 IEEE International Conference on Communications (ICC)*, pages 914–919, 2014.

[38] C. Liangboonprakong and O. Sornil. Classification of malware families based on n-grams sequential pattern features. pages 777–782, 2013.

[39] Y. D. Lin, Y. C. Lai, C. H. Chen, and H. C. Tsai. Identifying android malicious repackaged applications by thread-grained system call sequences. *Computers and Security*, 39(PART B):340–350, 2013.

[40] P. Faruki, V. Laxmi, V. Ganmoor, M. Gaur, and A. Bharmal. Droidolytics: Robust feature signature for repackaged android apps on official and third party android markets. In *2013 2nd International Conference on Advanced Computing, Networking and Security*, pages 247–252, 2013.

[41] J. Ko, H. Shim, D. Kim, Y.-S. Jeong, S.-j. Cho, M. Park, S. Han, and S. B. Kim. Measuring similarity of android applications via reversing and k-gram birthmarking. page 336–341, 2013.

[42] S. Kishore, R. Kumar, and S. Rajan. *Towards Accuracy in Similarity Analysis of Android Applications*, volume 11281 LNCS. Springer International Publishing, 2018.

[43] R. Potharaju, A. Newell, C. Nita-Rotaru, and X. Zhang. Plagiarizing smartphone applications: Attack strategies and defense techniques. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7159 LNCS:106–120, 2012.

[44] J. Crussell, C. Gibler, and H. Chen. Scalable Semantics-Based Detection of Similar Android Applications. *Esorics*, pages 182–199, 2013.

[45] W. Hu, J. Tao, X. Ma, W. Zhou, S. Zhao, and T. Han. Migdroid: Detecting app-repackaging android malware via method invocation graph. In *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, pages 1–7, 2014.

[46] W. Zhou, Y. Zhou, M. Grace, X. Jiang, and S. Zou. Fast, scalable detection of "piggybacked" mobile applications. *CODASPY 2013 - Proceedings of the 3rd ACM Conference on Data and Application Security and Privacy*, pages 185–195, 2013.

[47] K. Chen, P. Liu, and Y. Zhang. Achieving accuracy and scalability simultaneously in detecting application clones on Android markets. *Proceedings - International Conference on Software Engineering*, (1):175–186, 2014.

[48] L. Adhianto, S. Banerjee, M. Fagan, M. Krentel, G. Marin, J. Mellor-Crummey, and N. R. Tallent. HPCTOOLKIT: Tools for performance analysis of optimized parallel programs. *Concurrency and Computation: Practice and Experience*, 22(6):685–701, 2010.

[49] J. Zheng, K. Gong, S. Wang, Y. Wang, and M. Lei. Repackaged apps detection based on similarity evaluation. pages 1–5, 2016.

[50] M. Torki. Detecting repackaged android applications. 2018.

[51] Y. Wang. Obfuscation-resilient code detection analyses for android apps," phd the- sis, the ohio state university. 2018.

[52] X. Wu, D. Zhang, X. Su, and W. Li. Detect repackaged android application based on http traffic similarity. *Sec. and Commun. Netw.*, 8(13):2257–2266, sep 2015.

[53] M. Alshehri. APP-NTS: a network traffic similarity-based framework for repacked Android apps detection. *Journal of Ambient Intelligence and Humanized Computing*, 13(3):1537–1546, 2022.

[54] G. He, B. Xu, L. Zhang, and H. Zhu. On-device detection of repackaged android malware via traffic clustering. *Security and Communication Networks*, 2020:8630748, May 2020.

[55] A. Rodriguez and A. Laio. Clustering by fast search and find of density peaks. *Science*, 344(6191):1492–1496, 2014.

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