تشخیص بدافزارهای اندرویدی با رویکرد تخلیهٔ بار در محاسبات ابری

معصومه قاسمی [۱]، عباس حری [۲*]، محمداحسان بصیری [۳]
دانش‌آموختهٔ کارشناسی ارشد مهندسی کامپیوتر دانشکدهٔ فنی‌ومهندسی، دانشگاه شهرکرد، شهرکرد، ایران[۱]
استادیار مهندسی کامپیوتر، دانشکدهٔ فنی‌ومهندسی، دانشگاه شهرکرد، شهرکرد، ایران[۲]
دانشیار مهندسی کامپیوتر، دانشکدهٔ فنی‌ومهندسی، دانشگاه شهرکرد، شهرکرد، ایران[۳]

چکیده

اندروید در سال‌های اخیر به‌عنوان محبوب‌ترین سیستم‌عامل گوشی‌های هوشمند و دستگاه‌های تلفن همراه ظاهر شده‌است؛ بااین‌حال، با توجه به خاصیت متن‌بازبودن این سیستم‌عامل، بدافزارهای زیادی در میان نرم‌افزارها در بازارهای اندروید پنهان شده‌اند که امنیت آن را با خطر جدی مواجه کرده‌است؛ لذا یافتن راه‌حل‌هایی برای تشخیص این بدافزارها از کارهای ضروری جهت جلوگیری از آلوده‌شدن تلفن همراه است؛ به‌این‌منظور، در این پژوهش، برای تشخیص بدافزارها استفادهٔ روش تخلیهٔ محاسباتی در ساختار محاسبات ابر پیشنهادشده‌است که این روش باعث می‌شود تشخیص بدافزارها در زمان معقول، با دقت بالا و با استفاده از منابع کمتر انجام شود. در روش پیشنهادی ویژگی‌های برنامه‌های اندروید را در هنگام نصب و زمان اجرا سمت تلفن همراه استخراج می‌کنیم و ویژگی‌های استخراج‌شده به سمت سرور ابر برای تجزیه و تحلیل ارسال می‌شود و با الگوریتم یادگیری ماشین بدافزارهای اندروید از برنامه‌های تمیز تشخیص داده می‌شوند. در این پژوهش، رویکرد پیشنهادی با استفاده از مجموعه‌دادهٔ Drebin آموزش و آزمایش شده‌است. نتایج به‌دست‌آمده، نشان می‌دهد که رویکرد پیشنهادی به‌دقت ٪۹۶/۴۴ برای شناسایی بدافزار دست‌یافت.

واژگان کلیدی: بدافزار اندروید، یادگیری ماشین، محاسبات ابر.

Detecting Android malware with offloading approach in cloud computing

Masoume Ghasemi[1], Abbas Horri[2*] & MohammadEhsan Basiri[3]
Master, Department of Computer Engineering, Faculty of Engineering, Shahrekord University,Shahrekord, Iran[1]
Assistant Professor, Department of Computer Engineering, Faculty of Engineering, Shahrekord University,Shahrekord, Iran[2]
Associate Professor, Department of Computer Engineering, Faculty of Engineering, Shahrekord University,Shahrekord[3]

Abstract

Today, the mobile phone is one of the smart devices that have become a necessity in everyday life and are used for various tasks such as shopping, banking, communicating with friends, family, etc. In recent years, the Android operating system has been able to gain more popularity than other mobile phone operating systems. The number of software related to this operating system is also expanding at a remarkable speed. Unfortunately, this issue is not hidden from the profit-seeking people, and the production of malware of this operating system has also grown in parallel with its development. Third-party Android app stores that have emerged in recent years have become a very strong source of malware distribution, as these stores have weak to non-existent measures to prevent malicious apps from being uploaded and distributed to users' devices. Therefore, one of the challenges that programmers are dealing with in this field is to find solutions to establish security in these types of

* نویسندهٔ عهده‌دار مکاتبات
* Corresponding author

• تاریخ ارسال مقاله: ۱۳۹۹/۱۲/۲۹ • تاریخ پذیرش: ۱۴۰۳/۶/۱ • تاریخ انتشار: ۱۴۰۳/۱۰/۲۸ • نوع مطالعه: پژوهشی

[Downloaded from jsdp.rcisp.ac.ir on 2026-10-30]

[DOI: 10.61186/jsdp.21.3.137]

devices, in such a way that it provides powerful security analysis capabilities while consuming few resources on the device itself.
Software products such as Lookout, Norton, and Comodo Mobile Security mainly use signature-based methods to detect malware threats. However, malware attackers use techniques such as repackaging and obfuscation to circumvent signatures and defeat attempts to analyze their internal mechanisms. The ever-increasing sophistication of Android malware requires new defense techniques that can protect users against new threats while not using up all of a mobile device's processing and storage resources. Therefore, in the current research, a computational offloading method is presented in the cloud structure to identify Android malware.
The solution proposed by this research first extracts the features of Android applications during installation and execution on the mobile phone, then sends these extracted features to the cloud servers. On the cloud server side, these features are analyzed and using machine learning algorithms, malware is distinguished from clean programs. The proposed approach is trained and tested using the Drebin dataset. The obtained results show that the proposed approach has achieved 96.44% accuracy for malware detection.
Keywords: Android Malware, Machine Learning, Cloud Computing.

۱-مقدمه

امروزه تلفن همراه یکی از دستگاه‌های هوشمند است که به نیازی ضروری در زندگی روزمره تبدیل شده‌است. تا ژانویهٔ سال ۲۰۲۱، تعداد دستگاه‌های تلفن همراه هوشمند در سراسر جهان به سه میلیارد رسیده‌است. سیستم‌عامل اندروید نیز در سال‌های اخیر توانسته‌است محبوبیت بیشتری نسبت به دیگر سیستم‌عامل‌های تلفن همراه از آن خود کند و ۷۱/۹۳ درصد از بازار را به خود اختصاص داده‌است[۱]. تعداد نرم‌افزارهای مربوط به سیستم‌عامل اندروید نیز با سرعت چشمگیری در حال گسترش است؛ به‌طوری‌که تا سه ماه دوم سال ۲۰۲۲، بیش از سه‌ونیم میلیون برنامهٔ تلفن همراه اندروید در فروشگاه Google play در دسترس بود، شکل (۱)، تعداد برنامه‌های موجود در فروشگاه Google Play از سه ماههٔ نخست سال ۲۰۱۵ تا سه ماههٔ دوم سال ۲۰۲۲ را نشان می‌دهد[۲].



(شکل-۱): تعداد برنامه‌های موجود در فروشگاه Google Play از سه ماههٔ نخست ۲۰۱۵ تا سه ماههٔ دوم ۲۰۲۲ [۲]

(Figure-1): The number of apps available in the Google Play Store from the first quarter of 2015 to the second quarter of 2022 [2]

یکی از مهم‌ترین دلایل محبوبیت اندروید این است که کاربران تلفن همراه می‌توانند انواع مختلف برنامه‌ها را از فروشگاه‌های برنامه دانلود کنند. متأسفانه این موضوع از نظر افراد سودجو پنهان نمانده و تولید بدافزارهای[۱] این سیستم‌عامل نیز موازی با توسعهٔ آن رشد فراوانی کرده‌است. پروژهٔ ژنوم[۲] بدافزار اندروید با بودجهٔ ملی ایالات‌متحده یک مجموعه‌داده از نمونه‌های بدافزار اندروید را جمع‌آوری و منتشر کرده‌است. در میان این بدافزارها، نود درصد ربات‌ها تحت‌کنترل شبکه‌اند. ربات‌های موبایل به سرورهای فرمان و کنترل (c&c[۳]) متصل می‌شوند تا دستورات را از ربات استاد[۴] دریافت و اطلاعات دستگاه‌های آلوده را برای انجام فعالیت‌های مخرب مانند انتقال پیام‌های هرزنامه، سرقت داده‌ها که منجر به سرقت هویت، از دست‌دادن داده‌ها و ضررهای مالی می‌شود، ارسال کنند؛ علاوه‌براین، کاربران این سیستم‌عامل به‌ندرت نرم‌افزار آنتی‌ویروس را روی دستگاه‌های خود نصب می‌کنند؛ حتی کسانی که آن را نصب می‌کنند ممکن است نتوانند از آن استفادهٔ مؤثر کنند؛ ازاین‌رو، یکی از چالش‌هایی که برنامه‌نویسان در این زمینه با آن دست‌وپنجه نرم می‌کنند، یافتن راه‌حل‌هایی برای برقراری امنیت در این نوع دستگاه‌هاست.

راه‌حل‌های امنیتی موجود برای تلفن‌های همراه هوشمند، برای عملکرد منابع زیادی مانند حافظه، پردازنده و باتری را مصرف می‌کنند؛ این امر قابلیت استفادهٔ آن‌ها را به خطر می‌اندازد و کاربران را تشویق می‌کند از چنین راه‌حل‌هایی اجتناب کنند؛ در واقع، برای مؤثربودن یک راه حل امنیتی نیاز به نگه‌داشتن یک پایگاه‌دادهٔ جامع از امضاهای بدافزار است که نیاز به فضای ذخیره‌سازی زیادی از دستگاه تلفن همراه دارد؛ همچنین، هر رفتار مشکوکی باید با لیست بزرگی از امضاهای ذخیره‌شده اسکن شود که این امر باعث می‌شود منابع پردازشی و حافظهٔ زیادی از

[1] Malware
[2] MalGenom
[3] Command and Control
[4] BotMaster

[Downloaded from jsdp.rcisp.ac.ir on 2026-10-30]
[DOI: 10.61186/jsdp.21.3.137]

دستگاه‌های تلفن همراه استفاده شود؛ بااین‌حال، این راه حل برای یک دستگاه تلفن همراه با عمر باتری محدود، منابع محاسباتی کم و پهنای باند شبکهٔ محدود قابل اجرا نیست و یک چالش کلیدی در ایجاد راه حل‌های امنیتی مؤثر، محدودیت‌های محاسباتی و منابع ذخیره‌سازی تلفن‌های همراه است. درنتیجه نیاز به راه حل‌های امنیتی جامع برای تلفن‌های همراه است، به‌طوری‌که قابلیت‌های آنالیز امنیتی قدرتمند را با مصرف کم منابع روی خود دستگاه فراهم کند.

تشخیص بدافزار مبتنی بر ابر می‌تواند از اشتراک داده‌ها و منابع محاسباتی قدرتمند سرورهای امنیتی برای بهبود عملکرد شناسایی استفاده کند؛ به همین منظور رویکردی در این پژوهش برای تشخیص بدافزارها پیشنهاد شده که شامل استخراج ویژگی‌های برنامهٔ اندروید در دستگاه‌های تلفن همراه و سپس ارسال ویژگی‌های استخراج‌شده به سمت سرویس ابری برای تجزیه و تحلیل و استفاده از روش‌های یادگیری ماشین برای تشخیص بدافزار است.

در ادامه بخش‌های بعدی مقاله به صورت زیر سازمان‌دهی شده‌است: در بخش دوم مرور مختصری بر پژوهش‌های انجام‌شده در زمینهٔ تشخیص بدافزار ارائه شده‌است. در بخش سوم مفاهیم اولیه‌ای که در این پژوهش به‌کار برده‌شده، بیان شده‌است. در بخش چهارم طرح مسئله شرح داده‌شده و بخش پنجم شامل پیاده‌سازی و ارزیابی روش پیشنهادی است و در نهایت، بخش ششم به نتیجه‌گیری مقاله اختصاص داده شده‌است.

۲-ادبیات پژوهش

در پژوهش‌های پیشین روش‌هایی برای تشخیص بدافزارها پیشنهاد شده‌است که برخی از این روش‌ها عبارت‌اند از:

عمر و همکاران [۳]، از پنج الگوریتم طبقه‌بندی که عبارت‌اند از: ماشین بردار پشتیبان[1]، نزدیک‌ترین همسایه‌ها[2]، درخت تصمیم[3]، جنگل تصادفی[4] و Naive Bayes در شناسایی بدافزار اندروید استفاده کردند، سپس نتایج به‌دست‌آمده را با رویکرد یادگیری عمیق GRU بر روی مجموعه‌دادهٔ CICandMal2017 مقایسه کردند.

الزیلایی و همکاران [۴]، یک سیستم تجزیه و تحلیل پویا مبتنی بر یادگیری عمیق DL-Droid برای شناسایی بدافزار اندروید را ارائه دادند. در این سیستم از ویژگی‌های پویای استخراج‌شده از برنامه‌ها با استفاده از تلفن‌های واقعی استفاده کردند.

یوان و همکاران [۵]، چارچوبی ارائه کردند که با استفاده از یادگیری عمیق می‌تواند بدافزارهای اندروید را طبقه‌بندی کند. این روش با استفاده از دویست ویژگی استخراج‌شده با یک روش ترکیبی (ایستا + پویا) ارزیابی‌شده در ۲۵۰ برنامهٔ تمیز و ۲۵۰ بدافزار اندروید، به‌دقت ۹۶/۵٪ دست‌یافت.

هو و همکاران [۶]، یک سیستم خودکار شناسایی بدافزار اندروید Deep4MalDroid را پیشنهاد کردند که با استفاده از شبیه‌ساز Genymotion، فراخوانی‌های سیستم هستهٔ لینوکس را به‌صورت پویا استخراج می‌کند.

دی‌پیر و حری [۷]، معیاری به نام IRS ابداع کردند که این معیار مقادیر ریسک کمابیش بالایی را برای بدافزارهای شناخته‌شده به جای برنامه‌های معمولی محاسبه می‌کند؛ زیرا می‌تواند از بدافزارهای پیشین و نمونه بدافزارها به‌خوبی استفاده کند.

تام و همکاران [۸]، یک سیستم تجزیه و تحلیل پویای خودکار مبتنی بر VMI برای بازسازی رفتارهای بدافزار اندروید ارائه کردند؛ بااین‌حال، روش‌های تشخیص پویا اغلب نمی‌توانند مطمئن شوند که چه زمانی و چگونه تمام رفتارهای مخرب را فعال کنند؛ علاوه‌براین، تجزیه و تحلیل پویا به زمان زیادی برای تجزیه و تحلیل برنامه‌ها نیاز دارد که برای تلفن همراه مناسب نیست.

کیائو و همکاران [۹]، یک رویکرد یادگیری ماشین برای شناسایی بدافزار با استخراج الگوهای مجوز و فراخوانی‌های تابع API[5] که به‌وسیلهٔ برنامه‌های اندروید به‌دست‌آمده را پیشنهاد کردند. آن‌ها ارتباط بین مجوز و API را برقرار کردند؛ بااین‌حال، نمونه‌های مخربی که جمع‌آوری کرده‌اند کافی نیست.

لانگ و همکاران [۱۰]، یک سیستم تشخیص برای سیستم‌عامل اندروید مبتنی بر طبقه‌بندی‌کنندهٔ یادگیری ماشین به نام SVM پیشنهاد کردند. این سیستم به مرحلهٔ استخراج ویژگی و انتخاب ویژگی توجه می‌کند و از یک روش ترکیبی (ایستا + پویا) برای به‌دست‌آوردن ویژگی‌های برنامهٔ اندروید استفاده می‌کند، سپس یک الگوریتم انتخاب ویژگی جدید به نام PCA-RELIFE برای یافتن متمایزترین زیرمجموعهٔ ویژگی پیشنهاد کردند.

[1] Support Vector Machine
[2] K-Nearest Neighbors
[3] Decision Tree
[4] Random Forest
[5] Application Programming Interface

[Downloaded from jsdp.rcisp.ac.ir on 2026-10-30]
[DOI: 10.61186/jsdp.21.3.137]

عبدالرحمان و همکاران [۱۱]، مکانیسم جدیدی برای شناسایی بدافزارهای اندروید با استفاده از تجزیه و تحلیل شبه‌پویا و ساخت یک نمودار فراخوانی API که برای هر مسیر اجرا ساخته و آموزش داده‌شده است، ارائه کردند.

سلیمان و همکاران [۱۲]، یک رویکرد طبقه‌بندی مبتنی بر یادگیری ماشین موازی برای شناسایی بدافزار اندروید پیشنهاد کردند. یک مدل طبقه‌بندی مرکب از مجموعه‌ای موازی از طبقه‌بندی‌کننده‌های ناهمگن، یعنیSimple Logistic ، Naive Bayes، Decision Tree، PART وRIDOR ایجاد کردند. نتایجی که به‌دست آوردند، PART توانست از تمام طبقه‌بندی‌های دیگر بهتر عمل کند.

شین و همکاران [۱۳]، مجوز خطرناک را تجزیه و تحلیل و در پایگاه داده ذخیره کردند، سپس مجوزها را از برنامهٔ ناشناختهٔ اندروید استخراج کردند و پس از مقایسه با مجوز خطرناک، توانستند به نتیجه برسند.

کیائو و همکاران [۱۴]، چارچوبی به نام CBM ارائه کردند که توالی فراخوانی API را توسط ابزار تحلیل رفتار پویا استخراج می‌کند.

گاوفنگ‌هه و همکاران [۱۵]، روش جدیدی برای شناسایی بدافزار با خوشه‌بندی ترافیک برنامه در گره‌های محاسباتی لبه را پیشنهاد کردند. در این روش سرور لبه[1] ویژگی‌های ترافیک شبکهٔ برنامه‌ها را استخراج و این ویژگی‌ها را برای تجزیه و تحلیل بیشتر به سرور ابر ارسال می‌کند، ابر شباهت‌های بین برنامه‌ها را محاسبه و این مقادیر شباهت را برای جداسازی خودکار برنامه‌های اصلی و بدافزار دسته‌بندی می‌کند.

شیائو و همکاران [۱۶]، یک بازی تشخیص بدافزار را فرموله و Nash Equilibrium بازی را استخراج کردند تا رقابت شبکه، منابع محاسباتی و اشتراک داده در سرور امنیتی مبتنی بر ابر را مطالعه کنند.

زولکیفلی و همکاران [۱۷]، روشی را برای شناسایی بدافزار اندروید پیشنهاد کردند که بر اساس هفت ویژگی ترافیک شبکه و الگوریتم درخت تصمیم J48 است.

کاردلینی و همکاران [۱۸]، یک بازی تخلیهٔ غیرهمکاری فرموله و سیستم سه‌لایهٔ بارگذاری ابر را بررسی کردند که نرخ بارگذاری را به تکه ابر با منبع محدود تنظیم می‌کند تا زمان اجرای کار را کاهش دهد.

وانگ و همکاران [۱۹]، یک سیستم تعاملی محاسبات ابری سیار متشکل از چندین دستگاه تلفن همراه و سیستم محاسبات را درنظر گرفتند و یک فرمول بازی بیزی برای این سیستم تعامل پیشنهاد کردند. در این بازی، هر دستگاه تلفن همراه بخشی از درخواست خدمات خود را برای پردازش ازراه‌دور در سیستم محاسبات تعیین می‌کند.

سعیک و همکاران [۲۰]، یک بررسی جامع از بارگذاری وظایف در سه زمینهٔ الگوریتم‌های بهینه‌سازی، تکنیک‌های هوش مصنوعی و تئوری کنترل ارائه دادند و تکنیک‌های بالا را بر اساس عملکرد هدف، سطح دانه‌بندی، استفاده از لبه و زیرساخت‌های ابری و گنجاندن قابلیت تحرک در راه‌حل کلی، بسته به نوع دستگاه‌های لبه دسته‌بندی کردند.

دی‌پیر [۲۱]، ابزار نرم‌افزاری جدیدی به‌منظور سنجش میزان ریسک امنیتی برنامه‌ها در دستگاه‌های همراه طراحی و پیاده‌سازی کرده‌است؛ این ابزار از یک معیار جدید به‌منظور اندازه‌گیری ریسک بهره می‌برد. برای سنجش معیار، مجوزهای درخواستیِ ده‌ها بدافزار و صدها برنامهٔ تلفن همراه بررسی و تحلیل شده‌است؛ علاوه‌براین، به‌منظور ارزیابی دقیق‌تر، مجموعه‌داده‌های جدیدی از برنامه‌های ارائه‌شده در فروشگاه‌های داخلی و بدافزارهای جدید را گردآوری کرده‌است. آزمایش‌های صورت‌گرفته بر روی بدافزارها و نرم‌افزارهای بی‌خطر شناخته‌شده، نشان‌دهندهٔ دقت روش ارائه‌شده نسبت به معیارهای ارائه‌شدهٔ پیشین از نظر تخصیص ریسک امنیتی بالا به بدافزارها و ریسک پایین به نرم‌افزارهای بی‌خطر است.

در [۲۲] نویسندگان سعی کردند عدم‌قطعیت پیش‌بینی مدل‌های شبکهٔ عصبی عمیق (DNN) در شناسایی بدافزارها را برآورد کنند و از این برآوردها برای بهبود تکنیک‌های شناسایی بدافزارهای اندروید استفاده کردند. آنها علاوه بر آموزش مدل DNN برای پیش‌بینی بدافزار، چندین روش برآورد عدم‌قطعیت را برای آموزش یک مدل تصحیح به‌کار بردند؛ سپس با استفاده از خروجی عدم‌قطعیت تخمینی مدل تصحیح، نتایج پیش‌بینی را اصلاح کردند و دقت مدل DNN را بهبود دادند.

در این مقاله برای بهبود عملکرد شناسایی بدافزار و حل مشکل محدودیت‌های منابع محاسباتی ذخیره‌سازی دستگاه‌های تلفن همراه، از تلفیق روش‌های یادگیری ماشین و تخلیهٔ بار محاسباتی در ساختار ابر استفاده شد که در پژوهش‌های پیشین از این روش در شناسایی بدافزارهای اندروید استفاده نشده‌است.

۳- تعریف مفاهیم اولیه

در این بخش تعریفی اولیه از مفاهیمی که در این پژوهش به‌کار برده شده‌است، ارائه می‌شود.

بدافزارهای اندروید، نرم‌افزارهای مخربی هستند که برای سیستم‌عامل اندروید ساخته‌شده‌اند که در ظاهر برای

[1] Edge

[Downloaded from jsdp.rcisp.ac.ir on 2026-10-30]

[DOI: 10.61186/jsdp.21.3.137]

بهینه‌سازی، افزایش امنیت، حذف فایل‌های بیهوده، تقویت‌کنندهٔ سرعت تلفن همراه مناسب است یا به‌عنوان نرم‌افزار کاربردی و نرم‌افزاری همه‌کاره برای وقت‌گذراندن با دوستان، اجرای بازی‌ها و چت سریع ارائه می‌شوند، اما در واقع به جای ازبین‌بردن فایل‌های اضافی و زائد، هزاران نوع از بدافزارها را روی دستگاه اندروید دانلود می‌کنند و از همین طریق دسترسی‌های مختلفی را برای خود مقدور می‌سازند؛ یکی از روش‌های کاری این بدافزارها، استفاده از حساب facebook یا google کاربران و جمع‌آوری اطلاعات خصوصی از این طریق است که همین اطلاعات به شرکت‌های تبلیغاتی فروخته می‌شود.

در جامعهٔ پژوهشی معمولاً برای انجام آزمایش‌ها از مجموعه‌داده استفاده می‌شود. مجموعه‌داده‌ای که در این پژوهش مورد استفاده قرار گرفته‌است، مجموعه‌دادهٔ Drebin است؛ مجموعه‌دادهٔ Drebin، مجموعه‌داده‌ای از برنامه‌های اندروید است که پرمصرف‌ترین مجموعه‌دادهٔ بدافزار اندروید در جامعهٔ پژوهشی است. پژوهش‌گران برای ارزیابی و مقایسهٔ الگوریتم‌هایی که طراحی می‌کنند به این مجموعه‌داده تکیه می‌کنند [۲۳]. این مجموعه‌داده در مجموع دارای ۱۲۳۴۵۳ داده نمونه از برنامه‌های اندروید است که شامل ۵۵۶۰ نمونهٔ مخرب است. این نمونه‌ها در بازه زمانی اوت ۲۰۱۰ تا اکتبر ۲۰۱۲ جمع‌آوری شده‌اند. ویژگی‌های برنامهٔ اندروید در مجموعه‌دادهٔ Drebin، شامل هشت دسته به شرح زیر است[۲۴]:

- اجزای سخت‌افزاری که برای تنظیم مجوزهای سخت‌افزاری مورد نیاز نرم‌افزار استفاده می‌شوند.
- مجوزهای درخواستی که توسط کاربر در زمان نصب اعطا می‌شود و به نرم‌افزار کاربردی اجازهٔ دسترسی به منابع را می‌دهد.
- مؤلفه‌های برنامه که شامل چهار نوع مختلف رابط است: فعالیت‌ها، خدمات، ارائه‌دهندگان محتوا و گیرنده‌های پخش.
- اهداف فیلترشده که برای ارتباط فرایندی بین اجزا و برنامه‌های مختلف استفاده می‌شود.
- تماس‌های محدود API، دسترسی به یک‌سری تماس‌های کلیدی API.
- مجوزهای استفاده‌شده، زیرمجموعه‌ای از مجوزهایی که به‌طور واقعی در تماس‌های محدود API استفاده و درخواست می‌شوند.
- تماس‌های مشکوک API، تماس‌های API برای اجازهٔ دسترسی به داده‌ها و منابع حساس.
- آدرس‌های شبکه، آدرس‌های IP مورد دسترسی برنامه، از جمله نام میزبان و URL.

Androguard ابزاری مبتنی بر پایتون است که برای مهندسی معکوس برنامه‌های اندروید استفاده می‌شود. این ابزار فایل‌های خام برنامهٔ اندروید (apk) را می‌گیرد، آن‌ها را برای تجزیه و تحلیل، تفکیک می‌کند و در آنجا می‌توان آزمایش نفوذ برای بدافزار و آسیب‌پذیری‌ها را انجام داد. همان‌طورکه در بخش‌های پیشین ذکر شد در این پژوهش برای شناسایی بدافزار از روش تخلیهٔ بار محاسباتی استفاده شده‌است که در این بخش توضیح مختصری در مورد تخلیهٔ محاسباتی داده می‌شود.

(جدول-۱): فهرستی از ویژگی‌های برنامهٔ اندروید در مجموعه‌دادهٔ Drebin

(Table-1): A list of Android app features in the Drebin dataset

مجموعه‌ویژگی	توضیح
ویژگی‌های سخت‌افزاری	برنامه از ویژگی‌های سخت‌افزاری استفاده می‌کند.
مجوزهای درخواستی	برنامه اجازهٔ دسترسی می‌خواهد.
مؤلفه‌های برنامه	برنامه حاوی جزء مشکوک است.
مقاصد فیلترشده	برای ارتباط فرایندی بین اجزا و برنامه
تماس‌های محدود API	برنامه از تابع API برای دسترسی استفاده می‌کند.
مجوزهای استفاده‌شده	برنامه از مجوزها برای دسترسی استفاده می‌کند.
تماس مشکوک API	برنامه از تماس‌های مشکوک API استفاده می‌کند.
آدرس‌های شبکه	ارتباط با میزبان برای تبادل اطلاعات

تخلیهٔ محاسباتی یا بارگذاری وظایف[1] را می‌توان به‌عنوان انتقال وظایف محاسباتی با منابع فشرده به یک پلتفرم خارجی و غنی از منابع مانند مواردی که در رایانش ابری، لبه یا مه استفاده می‌شود، تعریف کرد. بارگذاری کل یا بخشی از وظایف به یک پردازنده یا سرور دیگر می‌تواند برای تسریع برنامه‌های کاربردی با منابع فشرده و حساس به تأخیر مورد استفاده قرار گیرد. بارگذاری وظایف یک فرایند پیچیده است و می‌تواند تحت‌تأثیر تعدادی از عوامل مختلف قرار گیرد. هدف بارگذاری وظیفه، بهینه‌سازی بارگذاری وظایف محاسباتی فشرده از دستگاه کابر نهایی به یک سایت راه دور تحت محدودیت‌های محاسباتی، ارتباطی و تحرکی مختلف است. فرایند بارگذاری وظیفه، همان‌طور که در شکل (۲) نشان داده‌شده‌است، شامل موارد زیر است[۲۰]:

- اجزای سخت‌افزاری مختلف مانند دستگاه‌های کاربر نهایی و دستگاه‌های ابر یا لبه

[1] Task Offloading

[Downloaded from jsdp.rcisp.ac.ir on 2026-10-30]

[DOI: 10.61186/jsdp.21.3.137]

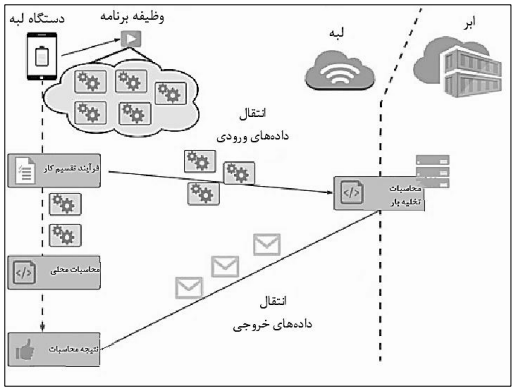
- فرایندهای محاسباتی متعدد از جمله تقسیم کار و پردازش محاسباتی به‌صورت محلی یا ازراه‌دور
- اجزای شبکه برای انتقال داده بین اجزای سخت‌افزاری درگیر

شبکهٔ عصبی مصنوعی[1] یا به اختصار شبکه عصبی، در حقیقت ایده‌ای برای پردازش اطلاعات است که از سیستم عصبی زیستی الهام گرفته و مانند مغز به پردازش اطلاعات می‌پردازد.

این سیستم از شمار زیادی عناصر پردازشی فوق‌العادهٔ به‌هم‌پیوسته به نام نورن‌ها[2] تشکیل شده‌است که برای حل یک مسئله، با هم هماهنگ عمل می‌کنند. شبکه‌های عصبی مصنوعی نیز مانند انسان‌ها با مثال یاد می‌گیرند و برای انجام وظیفه‌های مشخص مانند شناسایی الگوها و دسته‌بندی اطلاعات، در طول یک پروسهٔ یادگیری تنظیم می‌شوند. یک شبکهٔ عصبی مصنوعی به‌طور معمول شامل سه لایه است: لایهٔ ورودی[3] برای دریافت داده‌های اولیه، لایهٔ پنهان[4] یا مخفی برای وزن‌دهی به ورودی‌ها و ارتباط بین آن‌ها و لایهٔ خروجی[5].

(شکل-۲): فرایند بارگذاری وظیفه [۲۰]

(Figure-2): Task offloading process [20]



شبکهٔ عصبی پیش‌خور[6] که در این پژوهش از این نوع شبکه استفاده شده‌است. پردازش داده‌های ورودی روبه‌جلو انجام می‌شود و مسیر پردازش به نورون‌های لایهٔ پیشین برنمی‌گردد و خروجی هر لایه فقط بر لایهٔ بعدی تأثیر می‌گذارد. در این نوع شبکه مقدار پارامتر خروجی بر اساس پارامتر ورودی و یک‌سری وزن‌های اولیه تعیین می‌شود، مقادیر ورودی با هم ترکیب و سپس در لایه‌های پنهان از آن‌ها استفاده می‌شود و در آخر مقادیر لایه‌های پنهان نیز برای محاسبهٔ مقادیر خروجی ترکیب می‌شوند.

[2] Artificial neural network
[3] Nerouns
[4] Input Layer
[5] Hidden Layer
[6] Output Layer
[7] Feed Forward

دقت[7]، یکی از معیارهای ارزیابی در یادگیری ماشین است و به این معناست که مدل تا چه اندازه خروجی را درست پیش‌بینی می‌کند. فرمول (۱)، فرمول دقت است که با توجه به جدول (۲) که به ماتریس درهم‌ریختگی[8] معروف است، به‌دست می‌آید.

$$\text{دقت} = \frac{TP+TN}{TP+TN+FP+FN} \qquad (۱)$$

صحت[9]، یکی از معیارهای ارزیابی در یادگیری ماشین به معنای درصدی از پیش‌بینی‌های مدل است که مرتبط هستند و طبق فرمول (۲) محاسبه می‌شود.

$$\text{صحت} = \frac{TP}{TP+FP+FN} \qquad (۲)$$

حساسیت[10]، یکی از معیارهای ارزیابی در یادگیری ماشین است و اشاره به درصدی از کل پیش‌بینی‌هایی که توسط مدل، درست دسته‌بندی‌شده‌اند، دارد و طبق فرمول (۳) محاسبه می‌شود.

$$\text{حساسیت} = \frac{TP}{TP+FN} \qquad (۳)$$

(جدول-۲): ماتریس درهم‌ریختگی، عملکرد الگوریتم‌های یادگیری ماشین را نشان می‌دهد.

(Table-2): The Confusion Matrix shows the performance of machine learning algorithms.

	پیش‌بینی توسط الگوریتم		
		بلی	خیر
برچسب واقعی	بلی	True Positive(TP)	False Negative(FN)
	خیر	False Positive(FP)	True Negative(TN)

تابع ضرر[11] یا تابع هزینه، در واقع میزان خطا در هر بار اجرای شبکهٔ عصبی را برای داده‌های آموزشی نشان می‌دهد. با توجه به اینکه پایه و اساس آموزش در شبکه‌های عصبی بر مبنای تابع ضرر است؛ درنتیجه انتخاب تابع ضرر مناسب برای شبکهٔ عصبی از اهمیت بالایی برخوردار است.

۴- طرح مسئله

با توجه به پیشرفت‌های اخیر فنّاوری، دستگاه‌های تلفن همراه و گوشی‌های هوشمند برای انجام کارهای مهم از جمله آموزش، تراکنش‌های بانکی، خرید از فروشگاه‌های آنلاین و اوقات فراغت به نیاز روزمره تبدیل شده‌اند[۲۵]. اندروید نیز به‌عنوان گسترده‌ترین سیستم‌عامل مورد استفاده برای گوشی‌های هوشمند و دستگاه‌های تلفن همراه ظاهر شده‌است. امنیت این سیستم‌عامل بیشتر به

[8] Accuracy
[9] Confusion Matrix
[10] Precition
[11] Recall
[1] Loss Function

[Downloaded from jsdp.rcisp.ac.ir on 2026-10-30]
[DOI: 10.61186/jsdp.21.3.137]

برنامه‌های نصب‌شده توسط صاحب دستگاه متکی است[۷]. ابزار اصلی توزیع برنامه‌های اندروید از طریق بازارهای برنامه است و چندین فروشگاه برنامهٔ آنلاین غیررسمی در کنار فروشگاه رسمی برنامهٔ Google Play در حال ظهورند[۱۲]؛ بااین‌حال، بازبودن بازار اندروید آن را به یک هدف داغ برای حملات بدافزار تبدیل می‌کند که تهدید جدی برای امنیت و حریم خصوصی کاربران است. درنتیجه، نیاز فوری به برداشتن بدافزار از برنامه‌های معمولی وجود دارد[۵]. تعداد زیادی بدافزار تلفن همراه از جمله ویروس، کرم، تروجان و ابزارهای جاسوسی برای حمله به دستگاه‌های تلفن همراه در گردش‌اند که منجر به نشت حریم خصوصی، ضرر اقتصادی، کاهش قدرت و کاهش عملکرد شبکه می‌شوند[۲۶]؛ بااین‌حال، به‌دلیل محدودبودن منابع دستگاه‌های تلفن همراه مانند عمر باتری محدود، منابع محاسباتی، پهنای باند شبکه، ظرفیت ذخیره‌سازی و عملکرد پردازنده، پیداکردن روش‌هایی که بتواند با سرعت بالا و در زمان معقول تشخیص را انجام دهد و درعین‌حال منابع کمتری مصرف کند مورد توجه قرار گرفته‌است. انتقال وظایف محاسباتی روی یک پلتفرم خارجی و غنی از منابع مانند رایانش ابری از طریق یک شبکه می‌تواند قدرت محاسباتی را فراهم کند و بر محدودیت‌های سخت‌افزاری دستگاه تلفن همراه مانند توان محاسباتی محدود، ذخیره‌سازی و انرژی غلبه کند.

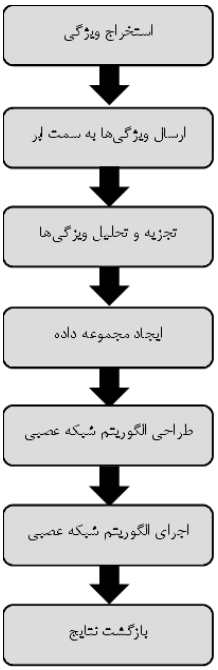
۵- راه حل پیشنهادی

در این پژوهش برای تشخیص بدافزارهای اندرویدی با توجه به محدودیت‌های منابع محاسباتی، ذخیره‌سازی و عملکرد پردازندهٔ تلفن همراه، رویکرد تخلیهٔ محاسباتی در ساختار محاسبات ابر پیشنهاد شده‌است که برای بررسی نتایج این رویکرد سه سناریو پیاده‌سازی شده‌است و این سه سناریو عبارت‌اند از:

۱) استخراج ویژگی‌های برنامه‌های اندروید در سمت تلفن همراه و ارسال این ویژگی‌ها به سمت سرورهای ابر برای شناسایی بدافزار با استفاده از الگوریتم‌های یادگیری ماشین.
۲) شناسایی مستقیم بدافزار بر روی تلفن همراه
۳) ارسال برنامه‌های اندروید به سمت سرور ابر و شناسایی بدافزار با الگوریتم‌های یادگیری ماشین.

سناریوی نخست همان راه حل پیشنهادی در این پژوهش است که با بارگذاری وظایف شناسایی به سرورهای امنیتی، تشخیص بدافزار مبتنی بر ابر می‌تواند آسیب‌پذیری روز صفر[1] را برطرف کند، برنامه‌های آسیب‌پذیر را شناسایی کند و محاسبات و مصرف انرژی دستگاه‌های تلفن همراه را کاهش دهد. برای مقایسهٔ نتایج به‌دست‌آمده از سناریوی نخست (راه حل پیشنهادی)، سناریوهای دوم و سوم پیاده‌سازی می‌شوند تا عملکرد سناریوی نخست بهتر دیده شود.

شکل (۳)، شمای کلی و شکل (۴)، شبه‌کد سناریوی نخست را نشان می‌دهد.



(شکل-۳): شمای کلی سناریوی نخست (راه حل پیشنهادی)
(Figure-3): An overview of the first scenario (proposed solution)

1. Extracting the features of Android application on the mobile side
2. Send the extracted features to the cloud
3. Analyze features on the cloud
4. Creating datasets for artifical neural network
5. Design of artifical neural network algorithm
6. Implementation of artifical neural network on the dataset
7. Return result

(شکل-۴): روندنمای سناریوی نخست (راه حل پیشنهادی)
(Figure-4): Flowchart of the first scenario (proposed solution)

۶- پیاده‌سازی و نتایج

جدول (۳)، مشخصات سخت‌افزاری تلفن همراه و سرور ابری استفاده‌شده برای پیاده‌سازی سه سناریوی ذکرشده را نشان می‌دهد.

(جدول-۳): مشخصات سخت‌افزاری تلفن همراه و سرور ابری

	مشخصات سخت‌افزار			
	GPU	CPU	RAM	Disk
تلفن همراه	Intel HD Graphics 4600	2GB	4GB	60GB
سرور ابری	Nvidia Tesla T4	2.20GHz	12.68GB	78.19GB

(Table-3): Mobile phone and cloud server hardware specifications

[2] Zero day

[Downloaded from jsdp.rcisp.ac.ir on 2026-10-30]
[DOI: 10.61186/jsdp.21.3.137]

برای پیاده‌سازی سناریوی نخست، در سمت تلفن همراه با ابزار Androguard ویژگی‌های ۱۲۱۰ برنامهٔ اندروید از مجموعه‌دادهٔ Drebin، استخراج و درون فایل‌های متنی ذخیره می‌شوند. بعد از استخراج ویژگی‌های برنامه‌های اندروید، یک شبیه‌ساز task offloading به زبان پایتون نوشته شد تا این ویژگی‌های استخراج‌شده را به سمت سرور ابری ارسال کند تا در آنجا تجزیه و تحلیل لازم برای شناسایی بدافزار انجام شود.

برای تجزیه و تحلیل ویژگی‌ها در سمت سرور ابر، کدی به زبان پایتون نوشته شد تا این ویژگی‌ها را که شامل مجوزهای[1] مورد نیاز برنامه‌ها در هنگام نصب و زمان اجراست را استخراج و درون یک فایل csv ذخیره کند و داده‌ها با فرایندی ذخیره می‌شوند که یک بردار دودویی از مجوزهای مورد استفاده برای هر برنامه تحلیل شده‌است (یک = برنامه از این مجوز استفاده می‌کند، صفر = برنامه از این مجوز استفاده نمی‌کند). در این مجموعه‌داده نام برنامهٔ اندروید در ستون نخست و در ستون‌های بعد مجوزهایی که برنامه‌ها از آن‌ها استفاده می‌کنند، نوشته شده‌است که در صورت استفادهٔ هر برنامه از هر مجوز عدد یک و در غیر این صورت عدد صفر درج شده‌است. این مجوزها عبارت‌اند از: دریافت پیامک، ارسال پیامک، آمار باتری، دسترسی به حالت شبکه، خواندن وضعیت تلفن، آمار باتری و غیره. در شکل (۵)، بخشی از مجوزهای استخراج‌شده از برنامه‌های اندروید را نشان داده شده‌است.

	A	B	C	D	E
1		READ_PHONE_STATE	DELETE_CACHE_FILES	ACCESS_CACHE_FILESYSTEM	UNINSTALL_SHORTCUT
2	Plankton	1	0	0	0
3	DroidKungFu	1	0	0	0
4	Plankton	1	0	0	1
5	GinMaster	1	0	0	1
6	FakeDoc	1	0	0	0
7	GinMaster	1	0	0	0
8	FakeInstaller	1	0	0	0
9	Opfake	0	0	0	0
10	FakeInstaller	1	0	0	0
11	Opfake	1	0	0	0
12	BaseBridge	1	0	0	0
13	Nisev	0	0	0	0
14	BaseBridge	0	0	0	0
15	Opfake	1	0	0	0
16	FakeInstaller	1	0	0	0
17	Adrd	1	0	0	0
18	Kmin	1	0	0	0
19	GinMaster	1	0	0	0
20	Adrd	1	0	0	0

(شکل-۵): بخشی از مجوزهای استخراج‌شدهٔ برنامه‌های اندروید
(Figure-5): Part of the extracted licenses of Android applications

برای شناسایی و تشخیص بدافزارها، الگوریتم شبکهٔ عصبی بر روی مجموعه‌داده در سرور ابری اجرا می‌شود. مدت‌زمانی که طول می‌کشد تا بدافزار شناسایی شود با توجه به فرمول (۴) محاسبه می‌شود:

$$D_m = E_f + S_f + R_f \qquad (۴)$$

که در آن D_m مدت‌زمان تشخیص بدافزار، E_f زمان استخراج ویژگی‌ها، S_f زمان ارسال ویژگی‌ها به سمت سرور ابری و R_f زمان اجرای الگوریتم شبکهٔ عصبی در سرور ابری بر روی مجموعه‌داده است. در این فرمول، زمان برحسب ثانیه است. مدت‌زمان تشخیص برای ۱۲۱۰ برنامه با توجه به فرمول (۴) برابر است با:

$D_m = 180 + 421.44 + 34$

ترافیک ارسالی شبکه که مجموع حجم ویژگی‌های ارسال‌شده به سمت سرور ابری است برای ۱۲۱۰ برنامهٔ اندروید برابر با ۳۱۲۴۴۳۱ بایت است.

برای پیاده‌سازی سناریوی دوم، یعنی شناسایی مستقیم بدافزار روی تلفن همراه، با ابزار Androguard ویژگی‌های ۱۲۱۰ برنامهٔ اندروید را استخراج و با استفاده از کد پایتون این ویژگی‌ها درون فایل csv ذخیره و سپس الگوریتم شبکهٔ عصبی روی این مجموعه‌داده اجرا می‌شود. مدت‌زمان تشخیص بدافزار با توجه به فرمول (۴) به صورت زیر است با این تفاوت که زمان ارسال در این حالت برابر با صفر است (یعنی $S_f = 0$). ترافیک ارسالی نیز در این روش برابر با صفر است؛ زیرا تشخیص روی تلفن همراه انجام می‌شود.

$D_m = 180 + 193260$

برای پیاده‌سازی سناریوی سوم یعنی ارسال برنامه‌های اندروید به سمت سرور ابر و شناسایی بدافزار با الگوریتم‌های یادگیری ماشین، ابتدا با یک شبیه‌ساز task offloading که به زبان پایتون نوشته‌شده ۱۲۱۰ برنامهٔ اندروید را به سمت سرور ابر ارسال می‌کند. پس از ارسال، ویژگی‌های برنامه‌ها استخراج و با کد پایتون به صورت یک مجموعه‌دادهٔ csv ذخیره و الگوریتم شبکهٔ عصبی روی این مجموعه‌داده اجرا می‌شود. مدت‌زمان تشخیص بدافزار با توجه به فرمول (۴) برابر است با:

$D_m = 5844 + 43 + 34$

ترافیک ارسالی شبکه یعنی همان مجموع حجم ۱۲۱۰ برنامهٔ اندروید که به سمت سرور ابر ارسال شد برابر با ۱۴۹۸۸۲۸۲ بایت است.

(جدول-۴): عملکرد الگوریتم شبکهٔ عصبی مصنوعی
(Table-4): Performance of artificial neural network algorithm

	دقت	صحت	حساسیت	خطا	زمان اجرای الگوریتم
سناریوی نخست	۹۶/۴۴	۶۸/۳۱	۲۷/۴۸	۱۳/۴۸	۳۴
سناریوی دوم	۹۶/۴۴	۶۸/۳۱	۲۷/۴۸	۱۳/۴۸	۱۹۳۴۴۰
سناریوی سوم	۹۶/۴۴	۶۸/۳۱	۲۷/۴۸	۱۳/۴۸	۳۴

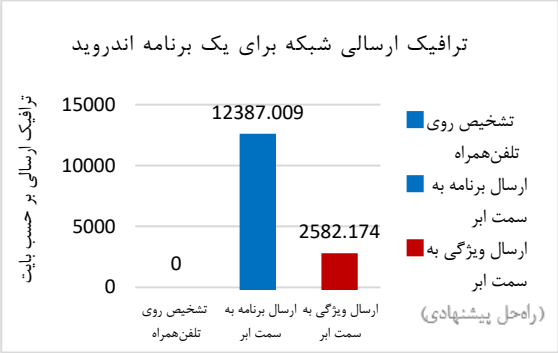
دقت و خطایی که با اجرای الگوریتم شبکهٔ عصبی بر روی مجموعه‌داده برای شناسایی بدافزار حاصل شد، به ترتیب ۹۶/۴۴ و ۱۳/۴۸ است که این دقت و خطا برای هر سه سناریویی که پیاده‌سازی شد، تغییر نکرد. بلکه زمان

[1] Permissions

[Downloaded from jsdp.rcisp.ac.ir on 2026-10-30]
[DOI: 10.61186/jsdp.21.3.137]

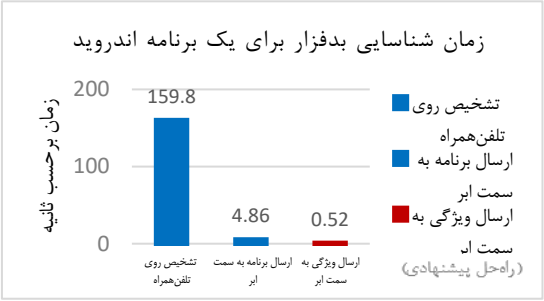
شناسایی بدافزار و ترافیک ارسالی برای هر سه سناریو متفاوت است که در جدول (۴)، عملکرد الگوریتم برای سه سناریو نشان داده شده‌است.

شکل‌های زیر نتایج به‌دست‌آمده از سه سناریو را با هم مقایسه می‌کند.



(شکل-۶): مقایسهٔ سه سناریوی شناسایی بدافزار بر حسب زمان

(Figure-6): Comparison of three scenario of malware detection by time



(شکل-۷): مقایسه سه سناریوی شناسایی بدافزار برحسب ترافیک ارسالی شبکه

(Figure-7): Comparison of three scenario of malware detection in terms of network traffic

با توجه به شکل‌های (۶) و (۷)، سناریوی نخست در این پژوهش یعنی استخراج ویژگی روی تلفن همراه و ارسال ویژگی‌ها به سمت سرور ابر، عملکرد بهتری از دو سناریوی دیگر دارد. زمان تشخیص بدافزار در سناریوی نخست نسبت به سناریوی دوم که تشخیص روی تلفن همراه انجام می‌شود ۹۹/۶۶ درصد و نسبت به سناریوی سوم که برنامه به سمت سرور ابر فرستاده می‌شود ۸۹/۰۹ درصد کاهش یافته‌است. ترافیک ارسالی شبکه در سناریوی اخیر نسبت به سناریویی که از سوی خود برنامه به سمت سرور ابر فرستاده می‌شود ۷۹/۱۵ درصد کاهش یافته‌است؛ اگرچه ترافیک ارسالی شبکه برای سناریویی که شناسایی بدافزار به‌طور مستقیم روی تلفن همراه انجام می‌شود برابر با صفر است، اما مدت‌زمانی که طول می‌کشد تا یک بدافزار شناسایی شود حدود ۱۵۹/۸ ثانیه است که این مدت‌زمان در مقایسه با سناریوی نخست برای شناسایی یک بدافزار با حدود ۰/۵۲ ثانیه بسیار زیاد است و امنیت دستگاه و کاربران تلفن همراه در خطر است و همچنین تمام منابع تلفن همراه برای شناسایی بدافزار استفاده می‌شود.

۷- نتیجه‌گیری و کارهای آینده

امروزه روش‌های مختلفی برای تشخیص بدافزارهای اندروید وجود دارد، اما بعضی از این روش‌ها دارای کارایی پایین هستند. در پژوهش حاضر نیز برای تشخیص بدافزارهای اندرویدی، رویکرد تخلیهٔ بار محاسباتی در محاسبات برای بهبود امنیت دستگاه‌های تلفن همراه ارائه شده‌است. این رویکرد با ارسال محاسبات سنگین به سرورهای ابری و دریافت نتایج از این سرورها باعث می‌شود محاسبات و مصرف انرژی دستگاه‌های تلفن همراه را کاهش دهد. همچنین از تکنیک‌های یادگیری ماشین در سرور ابر برای شناسایی مؤثر بدافزارهای اندرویدی استفاده شده‌است.

در ادامهٔ این پژوهش، به‌دلیل اینکه در ارسال بدافزارها به سمت سرورهای ابر حجم شبکه زیاد می‌شود، می‌توان از روش‌های مختلف فشرده‌سازی استفاده کرد. همچنین برای کاهش بار محاسبات می‌توان از روش‌های سبک‌سازی مانند روش‌های مختلف انتخاب ویژگی و ساده‌سازی محاسبات استفاده کرد.

۸-مراجع 8-Reference

[1] Number of Smartphone and Mobile Phone Users Worldwide in 2022/2023: Demographics, Statistics, Predictions. https://financesonline.com/number-of-smartphone-users-worldwide/. Accessed: 2022-August-16.

[2] Google Play: number of available apps by quarter 2022 — Statista. https://www.statista.com/statistics/289418/ number-of-available-apps-in-the-google-play-store-quarter. Accessed: 2022-August-16.

[3] O. N. Elayan, A. M. Mustafa, "Android Malware Detection Using Deep Learning, " *Procedia Computer Science*., vol. 184, pp. 847-852.

[4] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "DL-Droid: Deep learning based android malware detection using real devices, " Computers & Security 89 (2020) 101663, vol. 89, February 2020.

[5] Z. Yuan, Y. Lu, Z. Wang, and Y. Xue, "Droid-Sec: Deep learning in android malware detection, " *ACM SIGCOMM Computer Communication Review*, vol. 44, pp. 371–372, August 2014.

[6] S. Hou, A. Saas, L. Chen, and Y. Ye, "Deep4MalDroid: A deep learning framework for android malware detection based on Linux kernel system call graphs, " *Proc. - 2016 IEEE/WIC/ACM Int. Conf. Web Intell. Work. WIW 2016*, 2016, pp. 104–111.

[Downloaded from jsdp.rcisp.ac.ir on 2026-10-30]

[DOI: 10.61186/jsdp.21.3.137]

[7] M. Deypir, A. Horri, "Instance based security risk value estimation for Android applications, " *Journal of Information Security and Applications*, vol. 40, pp. 20-30, June 2018.
[8] K. Tam, S. J. Khan, A. Fattori, et al. "CopperDroid: Automatic Reconstruction of Android Malware Behaviors, " *Systems Security Research Lab and Information Security Group Royal Holloway University of London*, 2015, pp. 1-15.
[9] M. Qiao, A. H. Sung, Q. Liu, "Merging Permission and API Features for Android Malware Detection, " *5th IIAI International Congress on Advanced Applied Informatics, DOI 10.1109/ IIAI-AAI.2016.237*, 2016, pp. 566-571.
[10] L. Wen, and H. Yu, "An Android Malware Detection System Based on Machine Learning, " *AIP Conference Proceedings 1864, 020136-1–020136-7*, 2017.
[11] A. Pektaş, and T. Acarman. "Deep learning for effective android malware detection using api call graph embeddings, " *Soft Computing*, vol. 24, pp. 1027–1043, January 2020.
[12] S.Y.Yerima, S. Sezer, and I. Muttik, "Android malware detection using parallel machine learning classifiers, " *8th International Conference on Next Generation Mobile Applications, Services and Technologies*, 2024.
[13] Z.Qin, Y. Xu, B. Liang, et al. "An Android malware static detection method, " *Journal of Southeast University*, vol, 43, pp. 1162-1167, 2013.
[14] Y. Qiao, Y. Yang, J. He, et al, "CBM: Free, Automatic Malware Analysis Framework Using API Call Sequences, " *Advances in Intelligent Systems and Computing 214, DOI: 10.1007/978-3-642-37832-4_21*, Springer-Verlag Berlin Heidelberg, 2034, pp.225-236.
[15] G. He, B. Xu, L. Zhang, and H. Zhu, "On-Device Detection of Repackaged Android Malware via Traffic Clustering, " *Security and Communication Networks*, vol. 2020, pp. 1-19, May 2020.
[16] L. Xiao, Y. Li, X. Huangy, X. J. Du, "Cloud-based Malware Detection Game for Mobile Devices with Offloading, " *IEEE Transactions on Mobile Computing*, vol. 16, pp. 2742 - 2750, October 2017.
[17] A. Zulkifli, I. R. A. Hamid, W. M. Shah, and Z. Abdullah, "Android malware detection based on network traffic using decision tree algorithm, " *in Proceedings of the International Conference on Soft Computing and Data Mining*, Springer, Senai, Malaysia, January 2018 pp. 485–494,.
[18] V. Cardellini, V. De Nito Person´e, V. Di Valerio, et al, "A gametheoretic approach to computation offloading in mobile cloud computing, " *Springer Mathematical Programming*, vol. 157, pp. 421-449, June 2016.
[19] Y. Wang, X. Lin, and M. Pedram, "A Bayesian game formulation of power dissipation and response time minimization imobile cloud computing system, " *in Proc. IEEE Int'l Conf. Mobile Services*, pp. 7 – 14, June 2013.
[20] F. Saeik, M. Avgeris, D. Spatharakis, N. Santi, D. Dechouniotis, J. Violos, A. Leivadeas, N. Athanasopoulos, N. Mitton, S. Papavassiliou, "Task Offloading in Edge and Cloud Computing: A Survey on Mathematical, Artificial Intelligence and Control Theory Solution, " *Journal Pre-proof, May 2021 Computer Networks 195(3):108177*., vol. 195, August 2021.
[21] Deypir M. RiskMeter: "A Tool for Measuring Precise Security Risk Values of Mobile Device Applications, " *Signal and Data Processing*., vol. 14, pp. 23-36, December 2017.
[22] H. Li, et.al. "MalCertain: Enhancing Deep Neural Network Based Android Malware Detection by Tackling Prediction UncertaintyICSE," *24: Proceedings of the IEEE/ACM 46th International Conference on Software EngineeringMay*, 2024. pp 1–13.
[23] P.Irolla1, A. Dey, "The duplication issue within the Drebin dataset, " *Journal of Computer Virology and Hacking Techniques.*, vol. 14, pp. 245-249, August 2018.
[24] D.Arp, M.Spreitzenbarth, M.Hubner, H.Gascon, K.Rieck, "DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket, " Conference Network and Distributed System Security Symposium (NDSS), February 2014.
[25] S. Garg, S. K. Peddoju, A. K. Sarje, "Network-based detection of Android malicious apps, " *International Journal of Information Security.*, vol. 456, pp. 629-636, October 2021.
[26] R. Jusoh, A. Firdaus, S. Anwar, et.al. "Malware detection using static analysis in Android: a review of FeCO (features, classification, and obfuscation), " *PeerJ Comput. Sci.7 :e522*, DOI 10.7717/peerj-cs.522, June 2021.



معصومه قاسمی تحصیلات کارشناسی و کارشناسی‌ارشد خود را به‌ترتیب در رشته‌های علوم کامپیوتر و مهندسی کامپیوتر گرایش نرم‌افزار در سال‌های ۱۳۹۸ و ۱۴۰۱ در دانشگاه شهرکرد گذراند. زمینهٔ پژوهشی مورد علاقهٔ ایشان تشخیص بدافزار در اندروید و محاسبات ابری است.
نشانی رایانامهٔ ایشان عبارت‌است از:
masoume.ghasemi1995@gmail.com



عباس حری مدرک کارشناسی‌ارشد خود را در رشتهٔ مهندسی کامپیوتر گرایش نرم‌افزار در سال ۱۳۸۸ و مدرک دکتری مهندسی کامپیوتر را در سال ۱۳۹۳ از دانشگاه شیراز دریافت و اکنون استادیار گروه مهندسی کامپیوتر دانشگاه شهرکرد است. او مقاله‌های متعددی در زمینه‌های رایانش ابری، بدافزارهای اندروید و... نوشته است. زمینه‌های پژوهشی ایشان شامل رایانش ابری، بدافزارهای اندرویدی، امنیت شبکه و زنجیرهٔ بلوک است.
نشانی رایانامهٔ ایشان عبارت‌است از:
horri@sku.ac.ir

[Downloaded from jsdp.rcisp.ac.ir on 2026-10-30]
[DOI: 10.61186/jsdp.21.3.137]



محمد احسان بصیری مدرک کارشناسی خود را در رشتهٔ مهندسی کامپیوتر از دانشگاه شیراز در سال ۱۳۸۵، کارشناسی‌ارشد را در سال ۱۳۸۷ و مدرک دکتری مهندسی کامپیوتر را در سال ۱۳۹۳ از دانشگاه اصفهان دریافت کرد. وی اکنون دانشیار گروه مهندسی کامپیوتر دانشگاه شهرکرد است. او مقاله‌های متعددی در زمینه‌های داده‌کاوی، تحلیل احساسات نوشته است. علایق پژوهشی او شامل پردازش زبان طبیعی، طراحی سیستم‌های هوشمند، تحلیل احساسات و داده‌کاوی است.

نشانی رایانامهٔ ایشان عبارت‌است از:

basiri@sku.ac.ir

[Downloaded from jsdp.rcisp.ac.ir on 2026-10-30]

[DOI: 10.61186/jsdp.21.3.137]