ارائه راه‌کاری کارا برای تشخیص بدافزارهای ناشناخته بر مبنای تحلیل دستورات اسمبلی

فرنوش معنوی[۱]، دانشجوی دکتری؛ علی حمزه[۲]، استاد

۱- دانشکده مهندسی برق و کامپیوتر - دانشگاه شیراز- شیراز - ایران – F.manavi@cse.shirazu.ac.ir
۲- دانشکده مهندسی برق و کامپیوتر - دانشگاه شیراز- شیراز - ایران – Ali@cse.shirazu.ac.ir

چکیده: امروزه با گسترش سیستم‌های کامپیوتری نرم‌افزارهای مخرب رشد چشم‌گیری داشته‌اند. نرم‌افزارهای مخرب یا بدافزارها، یک برنامه هستند که باهدف آسیب‌رساندن به کامپیوتر، شبکه، اطلاعات و غیره توسعه داده‌شده‌اند. تشخیص نرم‌افزارها مخرب شاخه‌ای از امنیت کامپیوتر است که برای تجزیه‌وتحلیل برنامه‌های مشکوک، تشخیص نرم‌افزارهای مخرب و درنهایت، ازبین‌بردن تهدید در تلاش است. روش‌های مبتنی بر آپکد، ازجمله روش‌های متداول در شناسایی بدافزارها می‌باشد. با توجه به این‌که همه‌ی آپکدهای سازنده‌ی فایل‌ها برای شناسایی بدافزارها مهم نیستند می‌توان از برخی از آن‌ها در فرآیند تشخیص صرف‌نظر کرد. لذا در این مقاله، برای کلاسه‌بندی فایل‌ها از آپکدها استفاده خواهدشد با این تفاوت که فقط چند آپکد مهم و موثر برای تشخیص فایل‌ها در نظر گرفته خواهدشد. درروش ارائه‌شده نخست آپکدهای مهم فایل‌ها شناسایی می‌شود و با استفاده از این آپکدها، تصاویر ساخته می‌شود. سپس از این تصاویر، ویژگی استخراج می‌شود و در مرحله‌ی کلاسه‌بندی، مورداستفاده قرار می‌گیرد. مزیت روش پیشنهادی این است که براساس آپکدهای مهم، تصاویر ساخته می‌شود و مسئله‌ی تشخیص بدافزارها، به مسئله‌ی پردازش و کلاسه‌بندی تصاویر تبدیل می‌شود. ازاین‌رو روش پیشنهادی نسبت به روش‌های پیشین بهینه‌تر عمل می‌کند و پیچیدگی کمتری دارد.

واژه‌های کلیدی: آپکد، بدافزار، تشخیص بدافزار، تصویر، کلاسه‌بندی.

An Efficient Approach for Unknown Malware Detection Based on Opcode Analysis

Farnoush Manavi[1], PhD Student; Ali Hamzeh[2], Professor

1- Faculty of Electrical and Computer Engineering, University of Shiraz, Shiraz, Iran, Email: F.manavi@cse.shirazu.ac.ir
2- Faculty of Electrical and Computer Engineering, University of Shiraz, Shiraz, Iran, Email: Ali@cse.shirazu.ac.ir

Abstract: Today, with the development of computer systems, malware has grown dramatically. Malware is defined as a program that is developed with malicious purpose, such as sabotaging the computer system, information theft or other malicious actions. Malware detection is a branch of computer security which attempts to analyze suspicious programs, detect malware and ultimately eliminate the threat. Opcode-based methods are commonly used in malware detection. Given that, all Opcode are not important for detecting malware, some of them can be ignored in the detection process. In this research, the proposed method is based on Opcode Analysis, but only some of the important and effective Opcodes will be considered for file detection. First, momentous Opcodes of file are identified and employed for generating images. Then, features are extracted from the images in order to accomplish the classification. The advantage of the proposed method is that images are created based on important Opcodes and detecting malware is converted into image classification. Therefore, the proposed method is more optimized compared to the previous methods and also has acceptable accuracy and less complexity.

Keywords: Classification, image, malware, malware detection, opcode.

تاریخ ارسال مقاله: ۱۳۹۷/۱۰/۱۶
تاریخ اصلاح مقاله: ۱۳۹۸/۰۴/۲۶و ۱۳۹۸/۰۷/۰۹
تاریخ پذیرش مقاله: ۱۳۹۸/۱۰/۰۴
نام نویسنده مسئول: علی حمزه
نشانی نویسنده مسئول: ایران - شیراز – خیابان ملاصدرا - دانشگاه شیراز - دانشکده مهندسی برق و کامپیوتر.

۱- مقدمه

در چند سال اخیر با گسترش فناوری اطلاعات، سیستم‌های کامپیوتری مورد تهدید و حمله‌های بسیاری قرارگرفته‌اند. این تهدیدها و حمله‌ها شامل ویروس‌های جدید و یا انواع هک و نفوذ در سیستم‌های کامپیوتری است، که برای کاربران سیستم‌های کامپیوتری یک تهدید امنیتی جدی تلقی می‌شود. توصیفات متفاوتی برای معرفی نرم‌افزارهای مخرب، بیان شده‌است. به‌طور مثال، هر کدی که کاربر سیستم‌های کامپیوتری را بیازارد و یا عمداً باعث آسیب‌رسانی یا متوقف‌کردن عمل‌کرد سیستم شود به‌عنوان بدافزار شناخته خواهدشد [۱، ۲]. براساس نوع حملات بدافزارها به سیستم قربانی، آن‌ها به دسته‌هایی مانند ویروس‌ها، کرم‌ها، تروجان‌ها، درب‌های پشتی، جاسوس‌افزارها و باج‌افزارها تقسیم‌شده‌اند [۳، ۴]. تشخیص نرم‌افزارهای مخرب[1] شاخه‌ای از امنیت کامپیوتر است، که برای تجزیه‌وتحلیل برنامه‌های مشکوک، تشخیص نرم‌افزارهای مخرب و درنهایت، ازبین‌بردن تهدید در تلاش است [۵، ۶]. روش‌های بسیاری برای تشخیص نرم‌افزارهای مخرب وجود دارد که عمدتاً در سه دسته‌ی زیر جای می‌گیرند [۴]:

- تکنیک‌های مبتنی بر امضا: امضا دنباله‌ای از بایت‌های فایل مخرب است که برای شناسایی فایل مشکوک از آن استفاده می‌شود [۶]. تشخیص در این‌گونه روش‌ها بدین‌صورت می‌باشد که ابتدا یک پایگاه داده از فایل‌های مخرب براساس امضاهایشان ایجاد می‌شود. سپس از فایل مشکوک امضا گرفته می‌شود و این امضا را براساس تطبیق الگو با امضاهای پایگاه داده مقایسه می‌کنند. درصورتی‌که این امضا قبلاً مشاهده شده‌باشد، بدافزاربودن فایل تشخیص داده خواهدشد. این روش، اکثراً در آنتی‌ویروس‌ها به کار گرفته می‌شود که نسبتاً سریع می‌باشد اما توانایی تشخیص بدافزارهای جدید را ندارد [۷].
- تکنیک‌های مبتنی بر رفتار: روش‌های مبتنی بر رفتار معمولاً به این نیاز دارند که برنامه حتماً اجرا شود تا بتواند از آن برای استخراج ویژگی‌ها کمک بگیرند. لذا باید هر برنامه را در محیطی مجازی اجرا کرد و رفتار واقعی برنامه را به‌منظور ایجاد ویژگی‌های رفتاری برای دسته‌بندی برنامه مشاهده نمود. بسیاری از بدافزارهای جدید با مستقرشدن در سیستم و پس از اجرا برای چند لحظه بدون انجام هیچ‌گونه فعالیتی در حالت سکون باقی‌می‌مانند تا بتوانند روش‌های مبتنی بر رفتار را دور بزنند؛ به همین دلیل، این روش‌ها، با توجه به‌سرعت عمل‌کرد پایینشان برای تشخیص‌های سریع مناسب نمی باشند [۸، ۹].
- تکنیک‌های مبتنی بر روش‌های اکتشافی: روش‌های اکتشافی انواع متفاوتی دارند، مانند تجزیه‌کردن فایل به کدهای عملیاتی (آپکد) و یا سطوح پایین‌تر مثل بایت. در این نوع روش‌ها تلاش می‌شود تا براساس ویژگی‌هایی که از فایل‌ها استخراج می‌شود و با کمک روش‌های یادگیری ماشین الگوریتمی برای تشخیص بدافزارهای ناشناخته ارائه شود و درنهایت فایل‌های بدافزار از فایل سالم تشخیص داده شود. این روش‌ها معمولاً پیچیدگی قابل‌قبول و دقت بالایی دارند [۱۱-۹].

با توجه به این‌که روش‌های مبتنی بر امضا با مقایسه‌ی امضای بدافزارهای شناخته‌شده با امضای فایل مشکوک کار می‌کنند، فقط قادر به تشخیص بدافزارهای شناخته‌شده هستند و در تشخیص بدافزارهای جدید و ناشناخته ناتوان می‌باشند. همچنین روش‌های مبتنی بر رفتار هم با توجه به‌سرعت عمل‌کرد پایینی که دارند برای تشخیص‌های سریع و موثر مفید نمی‌باشند. تحلیل‌های مبتنی بر روش‌های اکتشافی، از ویژگی‌های مختلف فایل برنامه، مانند کدهای عملیاتی در دستورات اسمبلی، اطلاعات ساختاری نظیر اطلاعات هدر در فایل‌های اجرایی و فراخوان‌های API[2] استفاده می‌کنند و از آن‌ها برای استخراج ویژگی در فرآیند تشخیص بدافزارها بهره می‌برند [۴]. تشخیص و کلاسه‌بندی بدافزارها، معمولاً مبتنی بر الگوریتم‌های یادگیری ماشین مانند، الگوریتم بیز[3] [۸، ۹]، روش دسته‌بندی‌کننده ساده بیز[4]، درخت‌های تصمیم‌گیری[5]، ماشین بردار پشتیبان[6] و شبکه‌های عصبی انجام می شود. سیستم‌های ضد بدافزار مبتنی بر روش‌های اکتشافی قادرند با بدافزارهای ناشناخته دست‌وپنجه نرم کنند. هم‌چنین در مقایسه با تکنیک‌های رفتاری، پیاده‌سازی راحت‌تری دارند. لذا این پژوهش براساس روش‌های اکتشافی با تحلیل کدهای عملیاتی (آپکد)، بدافزارها را کلاسه‌بندی خواهدکرد.

به‌دست‌آوردن مجموعه ویژگی از فایل‌های اولیه‌ی بدافزارها کار چالش‌برانگیزی می‌باشد. ویژگی‌های متمایز سیگنال‌ها نسبت به داده‌های عادی باعث شده تا بتوان از روش‌های پردازش سیگنال برای تشخیص و دسته‌بندی بدافزارها استفاده نمود. یکی از قابلیت‌های مفید سیگنال، امکان ترکیب چندین سیگنال با یک‌دیگر و تولید سیگنالی جدید با حفظ ویژگی تمامی سیگنال‌های اولیه است. روش‌های متعددی درزمینه‌ی پردازش سیگنال صوتی و تصویری و دیگر سیگنال‌ها وجود دارد که می توان از این روش‌ها برای استخراج ویژگی، تشخیص شباهت و پیدانمودن ناسازگاری استفاده کرد. لذا با توجه به قابلیت حذف نویز و داده‌های پرت در سیگنال‌ها، در این مقاله از روش‌های پردازش سیگنال‌های تصویری برای کلاسه‌بندی نمونه‌های جدید استفاده شده‌است.

در این مقاله، ابتدا در بخش ۲ به بررسی چند کار مرتبط در حوزه‌ی تشخیص بدافزارها پرداخته خواهدشد. در بخش ۳ روش پیشنهادی که تبدیل آپکدها به عکس و استخراج ویژگی از آن‌ها می‌باشد ارائه خواهدشد و آپکدها، مورد بررسی قرار خواهندگرفت و چگونگی استخراج آپکدهای مهم مشخص خواهدشد. در بخش ۴ به بررسی معیارهای ارزیابی، خصوصیات مجموعه داده‌ها و نتایج اجرا پرداخته خواهدشد. سپس در بخش ۵ تحلیل روش پیشنهادی بیان خواهدشد و در بخش آخر نتیجه‌گیری از کار، ارائه خواهدشد.

۲- کارهای مرتبط

در حوزه‌ی یادگیری ماشین روش‌های متفاوتی برای شناسایی بدافزارها وجود دارد. این روش‌ها با کمک یکی از خصوصیات، گراف کنترل جریان [۱۲-۱۴]، N-گرم [۱۵-۱۷]، فراخوانی API [۱۸-۲۰]، کدهای عملیاتی[۵، ۸، ۲۴-۲۱]، بایت‌ها تشکیل‌دهنده‌ی فایل‌ها [۲۵، ۲۶] و یا ترکیبی از آن‌ها کار می‌کنند. در این بخش نگاهی بر روی کارهای انجام شده و نحوه‌ی استخراج ویژگی‌ها خواهدشد.

گراف کنترل جریان، یک نمایش از مسیرهای اجرای برنامه با استفاده از نشان‌گذاری گراف می‌باشد [۱۲]. وقتی گراف کنترل جریان ساخته شد، روش‌های متفاوتی روی آن به کار گرفته می‌شود تا از آن ویژگی استخراج شود. به‌عنوان مثال دینگ و همکارانش[۱۳] بعد از ساخت گراف و درنظرگرفتن همه‌ی مسیرهای ممکن برای اجرا، با استفاده از مفاهیم بهره اطلاعات[۷] و تکرار سند[۸]، بهترین ویژگی‌ها را انتخاب می‌کنند. درنهایت با استفاده از روش‌های یادگیری ماشین مانند ک-نزدیک‌ترین همسایه و درخت تصمیم‌گیری و ماشین بردار پشتیبان مدل خود را آموزش می‌دهند. یکی از مشکلات اساسی این مقاله این است که درنظرگرفتن همه‌ی مسیرهای ممکن برای اجرای یک قطعه کد امری پیچیده و دشوار می‌باشد که تحلیل مسئله را زمانی که کدهای برنامه زیاد باشد مشکل‌تر می‌کند و در بعضی شرایط ارائه همه‌ی مسیرهای ممکن از لحاظ زمانی مقرون به‌صرفه نیست. همچنین زمانی که معیار افزایش اطلاعات و تکرار سند برای چند مسیر مختلف، تقریبا یکسان می‌شود انتخاب ویژگی از بین این شرایط برای مقایسه‌ی فایل‌ها کار ساده‌ای نیست و ممکن است ویژگی‌های موثر به‌درستی انتخاب نشوند.

N-گِرم درواقع تمام زیررشته‌های ممکن با طول N از یک رشته‌ی بزرگ‌تر از طول N می‌باشد. به‌عنوان مثال در رشته‌ی "VIRUS " ۳-گرم‌های آن به‌صورت VIR، IRU، RUS می‌باشد [۵]. یکی از روش‌های تشخیص فایل بدافزار استفاده از روش N-گِرم می‌باشد. کنگ و همکارانش [۱۷] فایل‌های برنامه را به تکه‌های N تایی با استفاده از قاعده‌ی ساخت N-گِرم که در بالا گفته شد، تجزیه می‌کنند و سپس با استفاده از تکه‌های به‌وجودآمده با توجه به‌دفعات تکرار آن‌ها، آنتروپی و بهره اطلاعات را به‌دست می‌آورند و بردار ویژگی از N-گِرم‌ها استخراج می‌نمایند. در آخر از روش‌های یادگیری ماشین مانند ک-نزدیک‌ترین همسایه برای طبقه‌بندی استفاده می‌کنند. یکی از اشکالات عمده در این مدل روش‌ها این است که تعیین مقدار N امری پیچیده و دشوار است و با توجه به مقادیر مختلف N دقت مسئله متفاوت خواهدشد. ازطرفی بین پارامتر N و پیچیدگی مسئله رابطه وجود دارد و باید بین آن‌ها توازن برقرار کرد که همین عامل تعیین پارامتر بهینه N را سخت می کند.

رابط برنامه‌نویسی مجموعه توابعی است که یک برنامه می‌تواند از یک برنامه دیگر فراخوانی کند [۱۹]. یکی از راه‌های جذاب برای تشخیص یک فایل بدافزار با استفاده از توالی پاسخ رابط برنامه‌نویسی نرم‌افزار است. به‌عنوان مثال بِلوید و همکارانش [۲۰] با استفاده از API تلاش می‌کنند که بدافزار بودن فایلی را طی سه گام، تشخیص دهند. در گام اول لیستی از تمام فراخوانی‌های API هر فایل PE ایجاد می‌کنند. در گام دوم عملیاتی انجام می‌دهند که لیست ایجادشده در مرحله قبل را برای تجزیه‌وتحلیل‌های بعدی آماده می‌کند. در این گام، API های تکراری از بین برده می‌شود و API های مشابه در یک گروه قرار می گیرد و فرکانس تکرار آن‌ها شمارش می‌شود. گام سوم تصمیم‌گیری می باشد، که مرحله تصمیم‌گیری مبتنی بر یک روش آماری شناخته‌شده و قدرتمند به نام MCA صورت می‌گیرد. مشکل روش آن‌ها این است که تعداد زیادی از بدافزارها به‌صورت غیرمستقیم یک API را فراخوانی می کنند به‌عنوان مثال یک تابع از یک کتاب‌خانه را فراخوانی می‌کنند که آن تابع، API را فراخوانی کرده‌است و یا این‌که به‌جای این‌که یک API را وارد یا فراخوانی کنند، در حین اجرا یک کتاب‌خانه را بارگذاری می کنند و API آن را فراخوانی می‌کنند. هم‌چنین درروش‌های فراخوانی API نیاز است که فایل اجرا بشود که همین امر تشخیص را بسیار سخت می‌کند.

مهم‌ترین پژوهش بر روی کدهای عملیاتی توسط بیلار [۲۱] انجام شده‌است. بیلار در این پژوهش به تجزیه‌وتحلیل آماری توالی کدهای عملیاتی، به‌عنوان یک معیار برای شناسایی بدافزارها پرداخته بود. او به این نتیجه رسید که توزیع توالی کدهای عملیاتی در برنامه‌های سالم و مخرب دارای تفاوت بسیار قابل‌توجهی هستند و نیز کدهای عملیاتی نادر، تفاوت بیشتری را توصیف می‌کنند و درنتیجه قدرت بالای کدهای عملیاتی را برای شناسایی بدافزارها به اثبات رساند. براساس نتیجه حاصل از این پژوهش روش‌های استخراج ویژگی متعددی برپایه دنباله کدهای عملیاتی ارائه شده‌است. به‌عنوان مثال روش ارائه‌شده توسط سانتوس و همکارانش [۸] که از فرکانس‌های دنباله‌ی کدهای عملیاتی وزن‌دهی شده برای محاسبه‌ی شباهت کسینوسی بین دو فایل اجرایی استفاده کرده‌اند. این دنباله‌ی کدهای عملیاتی براساس تحلیل آماری است. انتساب یک وزن به هر کد عملیاتی ازطریق محاسبه‌ی دفعات تکرار آن در مجموعه‌ای از برنامه‌های بدافزار و بی‌خطر به‌دست می‌آید. از این طریق آن‌ها توانستند میزان ارتباط یک کد عملیاتی به کلاس بدافزارها را به‌دست بیاورند و وزن مناسب برای کد عملیاتی را محاسبه کنند. نکته ی اصلی درروش پیشنهادی آن‌ها، تکیه‌ی به فرکانس دنباله‌ی کدهای عملیاتی برای محاسبه شباهت بین برنامه‌های اجرایی است.

سانتوس و همکارانش در پژوهش خود با استناد به پژوهش انجام شده توسط بیلار [۲۱] روشی نوین برای استخراج ویژگی‌ها بر مبنای دنباله کدهای عملیاتی ارائه کرده‌اند. آن‌ها در تحقیق خود ابتدا یک معیار اندازه‌گیری برای ارزیابی هر یک از کدهای عملیاتی بیان کرده‌اند. سپس با استفاده از این معیار به استخراج ویژگی‌ها پرداخته‌اند. ویژگی‌های استخراج‌شده را توسط کلاسه‌بندهای مختلف ازجمله درخت تصمیم‌گیری، ماشین بردار پشتیبانی، ک-نزدیک‌ترین همسایه و شبکه بیزین مورد آزمایش قرار داده‌اند. در این روش، به‌دست‌آوردن مقدار

مناسب برای دنباله‌ی کدهای عملیاتی یکی از چالش‌های مسئله خواهدبود. با افزایش طول دنباله‌ی کدهای عملیاتی، میزان محاسبات نیز افزایش چشمگیری خواهدداشت. برای هر فایل بسیار بزرگ با افزایش طول دنباله‌ی کدهای عملیاتی، ساخت بردار ویژگی زمان‌بر خواهدشد. در این‌صورت عملیات کاهش ابعاد، ساختن مدل کلاسه‌بندی و پیش بینی نمونه‌های جدید، زمان‌بر و با دقت کمتری انجام خواهدشد. براساس پیشرفت‌های حاصل از پژوهش‌های پیشین روش‌های استخراج ویژگی موثرتری برپایه دنباله کدهای عملیاتی ارائه شد.

هاشمی و همکارانش [۵] بعد از تجزیه‌کردن فایل‌ها در سطح اسمبلی شروع به ساخت گراف می‌کنند. به این صورت که ترکیبات دوتایی از دستورات اسمبلی پشت سر هم که به ۲-گِرم معروف است را در نظر می‌گیرند و از روی فرکانس تکرار این ترکیبات ۲-گِرمی ماتریس مجاورت گراف برای هر دنباله از آپکدها می‌سازند. بعد از ساخت ماتریس مجاورت گراف، با استفاده از روش پاور ایترشن[۹] [۳۱]، ماتریس ساخته شده را به بردار ویژگی تبدیل می‌کنند و سپس با استفاده از روش‌های یادگیری ماشین مانند ک-نزدیک‌ترین همسایه و ماشین بردار پشتیبان داده‌ها را طبقه‌بندی می‌کند. استفاده از روش پاور ایترشن برای استخراج ویژگی‌ها باعث می‌شود هر فایل به بهترین بردار ویژه‌ی خود همگرا شود. گرچه این روش دقت قابل‌قبولی دارد ولی پروسه‌ی همگراشدن به بهترین بردار ویژه، فرآیند زمان‌بری می‌باشد. لذا لازم است تا به‌صورت بهینه‌تر عملیات استخراج ویژگی از آپکدها صورت گیرد.

معنوی و همکارانش [۲۴] فایل‌های اجرایی را در سطح اسمبلی تجزیه می‌کنند و سپس ترکیبات ۲-گِرمی از همه‌ی دستورات را درنظر می‌گیرند و با توجه به این دستورات، ماتریس مجاورت گراف می‌سازند و این ماتریس‌ها را به عکس تبدیل می‌کنند. آن‌ها تلاش می‌کنند از تصاویر ویژگی استخراج کنند و با توجه به این ویژگی‌ها فایل‌های بدافزارها را تشخیص دهند. استفاده از همه‌ی آپکدهای سازنده فایل‌های اجرایی باعث می‌شود فرآیند تشخیص زمان‌بر شود و شناسایی بدافزار به‌صورت موثر انجام نپذیرد.

در برخی دیگر از پژوهش‌ها، با استفاده از همه‌ی بایت‌های تشکیل دهنده‌ی فایل‌های بدافزار و سالم، یا قسمتی از بایت‌هایشان، تجزیه‌وتحلیل انجام می‌شود، مثلا از هدر فایل‌های اجرایی که حاوی اطلاعات مفیدی می‌باشد بهره گرفته می‌شود و با استفاده از این بایت‌ها برای هر فایل، بردار ویژگی ساخته می‌شود. در برخی از این روش‌ها، مسئله کلاسه‌بندی بدافزارها با استفاده از بایت‌های تشکیل‌دهنده‌ی فایل‌های اجرایی به مسئله‌ی کلاسه‌بندی تصاویر، صوت، موسیقی و ... نگاشت می‌شود. سپس مسئله کلاسه‌بندی جدید، بررسی می‌شود و در حوزه‌ی جدید استخراج ویژگی انجام می‌شود. از آن‌جایی که در این تحقیق، مسئله کلاسه‌بندی بدافزارها با استفاده از آپکدهای تشکیل دهنده‌ی فایل‌های اجرایی به مسئله‌ی کلاسه‌بندی تصاویر نگاشته می‌شود، چند پژوهش که مسئله کلاسه‌بندی بدافزارها را به مسئله‌ی کلاسه‌بندی جدید تبدیل می‌کند، بررسی می‌شود تا بتوان، چالش‌های تبدیل مسئله کلاسه‌بندی بدافزارها به مسئله‌ی کلاسه‌بندی جدید را بررسی کرد. برای مثال در این حوزه می‌توان به پژوهش فرخمنش و همکارانش [۲۵] و ناتاراج و همکارانش [۲۶] اشاره کرد.

فرخمنش و همکارانش [۲۵] مسئله کلاسه‌بندی بدافزارها را به مسئله کلاسه‌بندی موسیقی تبدیل می‌کنند. آن‌ها درصورتی‌که فایل اجرایی ساختار PE داشته باشد ،PE هدر از فایل‌ها را در نظر می‌گیرند و در غیر این صورت بلاک اول از فایل‌ها را در مدنظر قرار می‌دهند و با استفاده از مبدل، بایت‌ها را به پیام MIDI [۲۵] تبدیل می‌کنند. MIDI یک واسطی بین ادوات موسیقی و کامپیوتر برای تولید صوت‌های مصنوعی می‌باشد. آن‌ها این MIDI را توسط یک ترکیب‌کننده[۱۰] نرم‌افزاری پیاده‌سازی کرده‌اند. کار ترکیب‌کننده تولید سیگنال‌های صوتی از دستورات MIDI می‌باشد. بعد از این‌که هر فایل به یک سیگنال صوتی تبدیل شد از تکنیک‌های پردازش موسیقی مانند MFCC [۲۵] برای استخراج ویژگی‌ها استفاده می‌کنند. با این ویژگی‌ها مدل خود را آموزش می‌دهند و در مراحل بعدی برای مشخص‌کردن هویت فایل‌ها از آن استفاده می‌کنند. یکی از عیوب این روش تعیین مقدار پارامترهایی است که در آن استفاده شده‌است و باید به دنبال پارامترهای بهینه گشت تا مدل بهتری را آموزش داد. ازطرفی تولید یک صوت باکیفیت که دقت بالاتری را حاصل کند روش را زمان‌بر و پر مصرف می‌نماید. ازاین‌رو باید یک تناسب بین مقادیر پارامترها رعایت شود که گاهی تعیین مقدار مطلوب آن‌ها مشکل است. در این روش یک بایت به مجموعه‌ای از نوت‌های هم زمان تبدیل می‌شود که از دید الگوریتم‌های استخراج ویژگی صوتی، باید تمایز بین بایت‌های مختلف به‌خوبی رعایت شود تا تشخیص ماهیت فایل‌ها به‌درستی صورت گیرد، که این امر مشکل است و بر روی دقت نهایی تاثیر چشمگیری می‌گذارد.

در پژوهش ناتاراج و همکارانش [۲۶] مسئله کلاسه‌بندی بدافزارها به مسئله کلاسه‌بندی تصاویر تبدیل می‌شود. آن‌ها فایل‌های اجرایی موجود را به تصاویر خاکستری تبدیل می‌کنند، به‌این‌ترتیب که هر بایت معادل با یک پیکسل در تصویر خواهدشد. سپس با استفاده از روش استخراج ویژگی GIST [۲۷] از تصاویر به‌دست‌آمده، ویژگی استخراج می‌کنند و از روش ک-نزدیکترین همسایه برای فرآیند کلاسه‌بندی استفاده می‌کنند. مشکل عمده روش آن‌ها حجم و حافظه‌ی مصرفی است. تولید تصاویر از فایل‌هایی که حجم بالایی دارند زمان زیادی لازم دارد و حافظه‌ی زیادی را مصرف خواهدکرد. ازطرفی همه‌ی بایت‌های تشکیل‌دهنده‌ی فایل، دارای ارزش یکسانی نیستند لذا لازم نیست از تمام بایت‌های تشکیل‌دهنده‌ی فایل، تصویر ساخته شود.

اکثر روش‌های موجود پیچیدگی زمانی زیادی دارند و باید زمان زیادی را صرف تحلیل و بررسی کنند تا بتوانند بهترین پارامترها را برای مسئله تشخیص بدافزارها به‌منظور شناسایی آن‌ها تنظیم نمایند. لذا در این پژوهش تلاش شده‌است تا با استفاده از بررسی چند آپکد که وجودشان در شناسایی ماهیت فایل‌ها موثرتر است تشخیص بدافزارها صورت گیرد. از همین رو، نقطه قوت و نوآوری روش پیشنهادی این است

که در زمان کمتری تشخیص بدافزارها صورت می‌گیرد و روش پیشنهادی بهینه‌تر عمل کرد.

۳- روش پیشنهادی

زمانی که فایل‌های اجرایی در سطح اسمبلی به‌درستی تجزیه می‌شوند، آپکدهای تشکیل‌دهنده‌ی آن‌ها مشخص خواهدشد. هر فایل با توجه به عمل‌کرد آن، از دنباله‌ای از آپکدها تشکیل خواهدشد. به عبارتی در فایل‌های مختلف، معمولاً دنباله‌ی آپکدهای سازنده‌ی آن‌ها با توجه به عمل‌کرد برنامه متفاوت خواهدبود. لذا با توجه به این تفاوت‌ها و این‌که هر فایل با درنظرگرفتن ماهیت آن معمولاً یک دنباله‌ی خاص از آپکدها را شامل خواهدشد، می‌توان از دنباله‌ی آپکدهای هر فایل برای شناسایی عمل‌کرد آن فایل، استفاده نمود. پردازش خام دنباله‌های آپکد هر فایل، به دلیل تکنیک‌های مبهم‌سازی‌هایی که انجام می‌شود، ممکن است زیاد دقیق نباشد و زمان زیادی را صرف خود کند. یکی از ساده‌ترین حالت‌هایی که برای تحلیل آپکدها در نظر می‌گیرند این است که ترکیبات چندتایی از آپکدهای پشت سرهم که به N-گِرم معروف است را در نظر می‌گیرند و با توجه به‌دفعات تکرار N-گِرم‌ها تجزیه‌وتحلیل انجام می‌شود.

در اکثر پژوهش‌ها میزان N، دو در نظر گرفته شده‌است، دلیل این انتخاب این است که هرچه میزان N بیشتر شود زمان تحلیل به‌صورت نمایی رشد خواهدکرد. بعد از این‌که ترکیبات دوتایی از آپکدهای پشت سرهم و یا همان ۲-گِرم‌ها مشخص شد فرکانس تکرار آن‌ها را در قالب ماتریس می‌توان نمایش داد. اگر تعداد آپکدهای غیرتکراری فایل برابر با M باشد آن گاه سایز این ماتریس برای هر فایل M×M خواهدشد. پردازش خام ماتریس تکرار ترکیبات ۲-گِرمی، باتوجه‌به حجمشان و رنج گسترده‌ی اعدادی که درون آن‌ها می‌باشد کار ساده‌ای نیست. از تکنیک‌های مختلف برای استخراج ویژگی از این ماتریس‌های خام می‌توان استفاده کرد. برای مثال در برخی از پژوهش‌ها [۵] بهترین بردار ویژه از این ماتریس را به‌عنوان بردار ویژگی برای هر فایل در نظر می‌گیرند و با استفاده از این بردار و روش‌های یادگیری ماشین یک مدل برای تشخیص بدافزارها ارائه می‌کنند. گرچه این روش‌ها دقت قابل‌قبولی دارد ولی فرآیند مشخص‌کردن بهترین بردار ویژه، فرآیند زمان‌بری می‌باشد. لذا می‌توان به‌جای استفاده از بهترین بردار ویژه، برای تبدیل ماتریس به بردار ویژگی، از فرآیندهای دیگری کمک گرفت. با توجه به این‌که در حوزه‌ی پردازش تصاویر الگوریتم‌های نسبتاً توانمند و سریعی برای پردازش تصاویر و استخراج اطلاعات از آن‌ها وجود دارد، در این پژوهش از آن‌ها کمک گرفته می‌شود. برای این منظور ماتریس فرکانس تکرار ترکیبات ۲-گِرمی به یک تصویر خاکستری تبدیل می‌شود و از این تصاویر، ویژگی استخراج می‌شود و برای گام کلاسه‌بندی و تشخیص بدافزارها مورداستفاده قرار می‌گیرد.

باید توجه داشت که هر آپکد برای تحقق عملیات خاصی در برنامه به کار گرفته می‌شود. مثلاً آپکد call به‌منظور فراخوانی یک روند در برنامه مورداستفاده قرار می‌گیرد. در این پژوهش، بعد از بررسی و تحلیل در سطح اسمبلی از فایل‌های بدافزار و سالم، این نتیجه محقق شد که دنبال‌کردن برخی از آپکدها در فایل‌های بدافزار و سالم تفاوت بیشتری در مورد خانواده‌هایشان و ماهیت آن‌ها ایجاد می‌کند. به عبارتی حضور برخی از آپکدهای در دنباله‌های N-گِرمی مهم‌تر از بقیه می‌باشد. لذا در این مقاله تلاش گردیده‌است که از این آپکدها استفاده شود تا فرآیند تشخیص بدافزارها به‌صورت بهینه‌تر و سریع‌تر انجام گیرد.

در این قسمت ابتدا روش پیشنهادی و نحوه‌ی تبدیل مسئله‌ی تشخیص و کلاسه‌بندی بدافزارها به یک مسئله کلاسه‌بندی تصاویر بیان خواهدشد. سپس در ادامه به چگونگی استخراج آپکدهای مهم‌تر در فایل‌ها پرداخته خواهدشد و درستی این تحلیل، صحت‌سنجی خواهدشد.

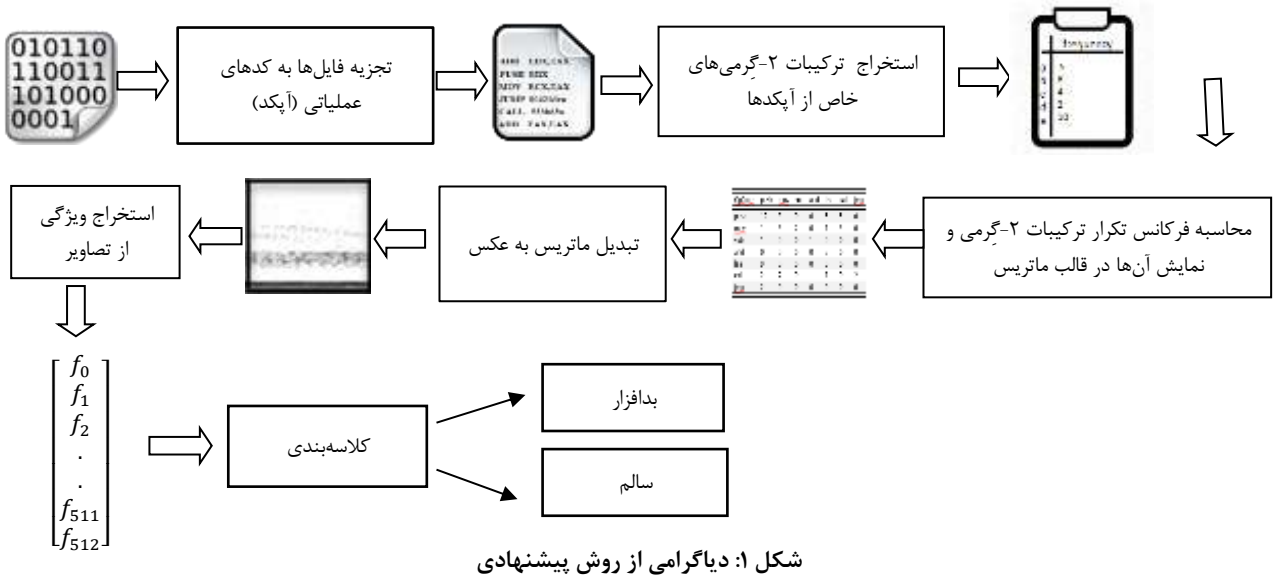
۳-۱- تبدیل مسئله‌ی تشخیص و کلاسه‌بندی بدافزارها به یک مسئله کلاسه‌بندی تصاویر

در این مقاله مسئله‌ی تشخیص و کلاسه‌بندی بدافزارها به مسئله‌ی کلاسه‌بندی تصاویر تبدیل می‌شود. از آن‌جایی که آپکدهای سازنده‌ی فایل‌های اجرایی به اجزای تشکیل‌دهنده‌ی تصاویر تبدیل خواهندشد، یکی از چالش‌های پیش‌روی در این فرآیند نحوه نگاشت آپکدهای سازنده‌ی فایل‌های اجرایی به اجزای تشکیل‌دهنده‌ی تصاویر خواهدبود. از این دید که داده‌های مسئله وارد فضای دیگری می‌شود این فرآیند مهم و چالش‌انگیز است. زیرا، اگر نگاشت به‌درستی صورت نگیرد اطلاعات مسئله از بین خواهدرفت و تشخیص بدافزارها به‌درستی صورت نخواهدگرفت. شکل ۱ مراحل روش پیشنهادی را گام‌به‌گام نشان می‌دهد. مراحل اصلی مدل ارائه‌شده به‌صورت زیر است:

۱- نگاشت آپکدهای سازنده‌ی فایل‌های اجرایی به یک تصویر
۲- استخراج ویژگی از تصاویر
۳- ایجاد یک مدل برای دسته‌بندی تصاویر
۴- استفاده از مدل ایجادشده برای تشخیص کلاس یک فایل دیده‌نشده

گام اول: در این مرحله باید فایل‌های موجود در مجموعه داده را با توجه به نرم‌افزارهای مناسب در سطح اسمبلی تجزیه کرد تا بتوان کدهای عملیاتی سازنده‌ی آن‌ها را استخراج کرد. بعد از به‌دست‌آوردن کدهای عملیاتی باید از عملوندها که مقابل آپکدها هستند صرف‌نظر کرد و عملگرها، که همان آپکدها هستند را در نظر گرفت. به‌این‌صورت، آپکدهای سازنده‌ی هر فایل در قالب دنباله‌ای پشت سرهم به‌دست خواهدآمد. برای فایل‌های که به‌صورت بسته‌بندی‌شده[11] هستند به روشی که فایل‌های عادی در سطح اسمبلی تجزیه شده‌اند، تجزیه انجام می‌شود. واضح است که آپکدهای به‌دست‌آمده، آپکدهای اصلی سازنده‌ی فایل‌ها نیستند، درواقع آپکدهایی هستند که توسط بسته‌بندی‌کننده ایجادشده‌اند و برای خارج‌کردن برنامه از حالت فشرده جهت اجرا استفاده می‌شوند.

گام دوم: با توجه به آن‌چه که گفته شد، یکی از ساده‌ترین حالت‌هایی که برای تحلیل آپکدها در نظر گرفته می‌شود این است که



شکل ۱: دیاگرامی از روش پیشنهادی

ترکیبات چندتایی از آپکدهای پشت سرهم که به N-گِرم معروف است را مشخص کرد و با توجه به‌دفعات تکرار N-گِرم‌ها تجزیه‌وتحلیل انجام شود. با افزایش پارامتر N تا یک آستانه ممکن است دقت تحلیل بیشتر شود ولی زمان اجرا به‌صورت نمایی رشد خواهدکرد. در بیشتر پژوهش‌ها میزان N دو در نظر گرفته شده‌است، که توازنی بین دقت و زمان است. ازاین‌رو بعد از تجزیه در سطح اسمبلی از فایل‌های اجرایی و استخراج آپکد از آن‌ها، ترکیبات دوتایی از آپکدها در نظر گرفته خواهدشد.

با توجه به این که گفته شد، حضور برخی از آپکدهای در دنباله‌های N-گِرمی مهم‌تر از بقیه است و با کمک آن‌ها می‌توان برای فرآیند تشخیص در حافظه و زمان صرفه‌جویی کرد، در محاسبه‌ی ترکیبات ۲-گِرمی از این خاصیت استفاده شده‌است و ترکیبات ۲-گِرمی، در نظر گرفته شده‌است که یکی از این ۲-گِرم‌ها شامل یک آپکد مهم باشد. با توجه به تحلیل‌های انجام‌شده برای فایل‌های اجرایی که در بخش ۳-۲ به آن پرداخته خواهدشد، آپکدهای {jmp, mov, push, call, add, lea} نسبت به بقیه پراهمیت‌تر هستند. لذا در این پژوهش، ۲-گِرم‌هایی درنظر گرفته شده‌است که یکی از ۲ شرط زیر را دارا باشند، تا بتوان با این کار در زمان و حافظه مصرفی، صرفه‌جویی انجام داد.

- ابتدای ترکیب دوتایی یکی از آپکدهای پراهمیت باشد و آپکد بعدی هر نوع آپکدی می‌تواند باشد.
- انتهای ترکیب دوتایی یکی از آپکدهای پراهمیت باشد، آپکد قبل هر نوع آپکدی می‌تواند قرار گرفته‌باشد.

گام سوم: برای هر فایل موجود در مجموعه داده، فرکانس تکرار ترکیبات ۲-گِرمی که در مرحله‌ی قبل مشخص شد، محاسبه می‌شود و در قالب یک ماتریس ذخیره می‌شوند، تا در مراحل بعدی از این ماتریس استفاده شود. به‌عنوان مثال فرض کنید آپکدهای ۱ تا n نشان‌دهنده‌ی کل کدهای عملیاتی در مجموعه داده باشند. برای هر فایل اجرایی ماتریسی مانند جدول ۱ ساخته خواهدشد. سطر اول از این ماتریس، فرکانس تکرار ترکیبات ۲-گِرمی از آپکدهای فایل را نشان می‌دهد که ابتدای ترکیب آپکد jmp قرار دارد و آپکد دوم در ترکیب دیگر آپکدهای ۱ تا n می‌باشد.

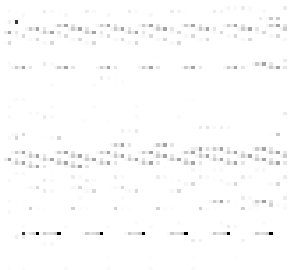
گام چهارم: ممکن است با توجه به تعداد آپکدهای غیرتکراری از مجموعه داده، ماتریس‌هایی که از گام سوم ساخته می‌شوند مربعی نباشند، در این صورت نزدیک‌ترین عدد مربعی به سایز ماتریس را مشخص کرده و با اضافه‌کردن صفر به ماتریس و یا صرف‌نظرکردن چند درایه از آن به ماتریس مربعی تبدیل می‌شود. این مرحله، برای آن است که تبدیل‌کردن ماتریس‌ها به عکس در مراحل بعدی راحت‌تر صورت گیرد.

جدول ۱: ساختار ماتریس یک فایل اجرایی

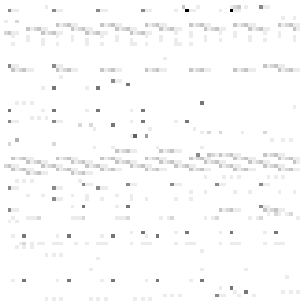
	آپکد ۱	آپکد ۲	آپکد ۳	:	آپکد n-1	آپکد n
jmp …	.	.	.	.	.	.
mov…	.	.	.	.	.	.
push …	.	.	.	.	.	.
call …	.	.	.	.	.	.
add …	.	.	.	.	.	.
lea …	.	.	.	.	.	.
… jmp	.	.	.	.	.	.
… mov	.	.	.	.	.	.
… push	.	.	.	.	.	.
… call	.	.	.	.	.	.
…add	.	.	.	.	.	.
… lea	.	.	.	.	.	.

گام پنجم: عناصر درون ماتریس‌های مرحله‌ی قبل نرمال‌سازی می شوند تا بتوان در مراحل بعد، از هر ماتریس یک عکس خاکستری ایجاد کرد. عناصر درون هر ماتریس به اعدادی بین ۰ تا ۲۵۵ نگاشته خواهندشد. برای این کار بیشترین درایه از هر ماتریس به ۲۵۵ تبدیل شده و بقیه درایه‌ها به این نسبت تغییر می‌کنند و نرمال می‌شوند. اگر در نرمال‌سازی اعداد درون ماتریس‌ها، بزرگترین درایه از کل ماتریس‌ها محاسبه شود و این عدد به ۲۵۵ نگاشته شود و با توجه به این تبدیل مابقی اعداد هم نگاشته شوند عکس‌هایی از ماتریس‌ها، حاصل خواهدشد که بیشتر پیکسل‌های آن سیاه می‌باشند، که عملاً معنی و مفهومی ندارد و در حین نگاشت اطلاعات از بین رفته است. این مشکل ناشی از آن است که دامنه‌ی اعداد برخی از ماتریس‌ها نسبت به بزرگ‌ترین درایه از کل ماتریس‌ها، عدد بسیار کوچکی می‌باشد، به همین خاطر هر ماتریسی با توجه به رنج اعدادی که درون خود دارد به بازه‌ی ۰ تا ۲۵۵ نگاشته خواهدشد.

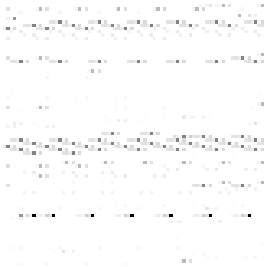
گام ششم: با توجه به این‌که در حوزه‌ی پردازش تصاویر الگوریتم‌های نسبتاً توانمند و سریعی برای پردازش تصاویر و استخراج اطلاعات از آن‌ها وجود دارد، در گام ششم، ماتریس‌های نرمال‌سازی‌شده از مرحله‌ی قبل به تصاویر خاکستری تبدیل می‌شوند. برای تبدیل‌کردن ماتریس‌های به‌دست‌آمده به تصاویر، به این صورت عمل می‌شود که مقدار هر درایه از ماتریس میزان وضوح رنگ یک پیکسل در عکس را معین می‌کند و مختصات هر درایه از ماتریس جایگاه آن پیکسل در عکس را نشان می‌دهد. به‌عنوان مثال اشکال ۲ تا ۵ چند نمونه از عکس‌هایی که از این طریق به‌دست‌آمده است را نشان می‌دهد.



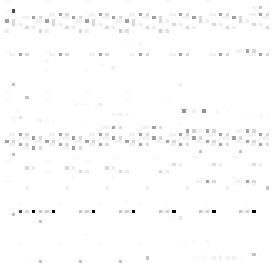
شکل ۲: عکس فایل LuncherBmp.dll از مجموعه داده‌های سالم PE



شکل ۳: عکس فایل Backdoor.IRC.Sobet.a از مجموعه داده‌های بدافزار PE



شکل ۴: عکس فایل FileSystemBrowser.dll از مجموعه داده‌های سالم PE



شکل ۵: عکس فایل Trojan.Win32.Xircon از مجموعه داده‌های بدافزار PE

گام هفتم: روش‌های متعددی برای استخراج ویژگی از تصاویر وجود دارد که در این پژوهش، پس از ساخت تصاویر با استفاده از روش GIST [۲۷]، از هر تصویر تعداد ۵۱۲ ویژگی به‌دست می‌آید. توصیفگر GIST ، برای شناسایی صحنه در تصاویر بکار می‌رود. این توصیفگر به‌صورت پیش‌فرض تصویر را به‌صورت شبکه‌های ۴ به ۴ تقسیم‌بندی می‌کند و از این طریق هیستوگرام‌های جهت‌گیری را استخراج می‌کند، که این کار مشابه به عمل‌کرد توصیفگر محلی SIFT[۲۷] می‌باشد. توصیفگر GIST، اشیای محلی را در نظر نمی‌گیرد و براساس روابط بین خطوط، خواص آن‌ها و فاصله اقلیدسی کار می‌کند. GIST پیچیدگی زمانی کمی دارد و می‌تواند از تصاویر خاکستری ویژگی‌های کم بعد قابل‌قبولی را استخراج نماید. در بیشتر کارهایی که از GIST استفاده شده‌است ابتدا تصویر را به یک عکس مربعی کوچک تغییر اندازه داده‌اند (معمولاً به اندازه‌های ۳۲ تا ۱۲۸ پیکسل تغییر اندازه می‌دهند) و از جزییات کم اهمیت چشم‌پوشی کرده‌اند [۲۹، ۳۰]. از همین رو با توجه به مشخصات تصاویر حاصل از روش پیشنهادی که در گام‌های قبل ذکر شد این توصیفگر برای این پژوهش و استخراج ویژگی از تصاویر روش پیشنهادی بسیار مناسب می‌باشد.

برای استفاده از GIST، پیاده‌سازی که توسط اولیوا و تروالبا [۲۷] با پارامترهای پیش‌فرض انجام شده‌است مدنظر قرار داده شده‌است. این پارامترها بردار ویژگی ۵۱۲ مولفه‌ای را فراهم می‌کند.

گام هشتم: با ویژگی‌های به‌دست‌آمده از مرحله‌ی قبل یک مدل برای کلاسه‌بندهای ماشین بردار پشتیبان، ک-نزدیکترین همسایه، جنگل تصادفی[۱۳] و کلاسه‌بند گروهی برای مسئله مشخص می‌شود.

گام نهم: با استفاده از مدل به‌دست‌آمده، داده‌های آزمایش مورد ارزیابی قرار می‌گیرد. برای این کار گام‌های ۱ تا ۷ باید تکرار شود و ویژگی ها از داده‌های آزمایش استخراج گردد و با توجه به این ویژگی‌ها و مدل به‌دست‌آمده در مرحله قبل، سنجش انجام شود و داده‌های آزمایش به دو دسته‌ی بدافزار و غیربدافزار طبقه‌بندی شود.

۳-۲- استخراج آپکدهای مهم از فایل‌های موجود در مجموعه داده‌ها

در این قسمت بررسی خواهدشد که چگونه می‌توان آپکدهای مهم فایل را شناسایی نمود تا در مراحل کلاسه‌بندی از آن‌ها برای تشخیص فایل‌های بدافزارها استفاده کرد و در زمان و فضا صرفه‌جویی نمود. در برخی از پژوهش‌ها [۵]، از همه‌ی آپکدها استفاده می‌شود و براساس همه‌ی آپکدها ویژگی به‌دست‌می‌آید. در این مقاله تلاش شده‌است تا آپکدهایی که تاثیر بیشتری در آشکارکردن تفاوت‌های میان خانواده‌های بدافزارها و فایل‌های سالم ایجاد می‌کنند شناسایی شود و از آن‌ها در جهت تشخیص سریع و موثر بدافزارها استفاده کرد. به‌منظور تحقق این هدف بعد از تجزیه‌کردن فایل‌ها در سطح اسمبلی و استخراج آپکدهای سازنده‌ی فایل، همه‌ی دنباله‌های ۲ تایی از آپکدها در نظر گرفته می‌شود. فرکانس تکرار این دنباله‌های ۲-گِرمی در قالب یک ماتریس نمایش داده می‌شود. اگر تعداد آپکدهای غیرتکراری در فایل M باشد ماتریس به دست‌آمده به‌اندازه‌ی M×M خواهدبود. بردارهای ویژه این ماتریس محاسبه می‌شود. بردارهای ویژه جهت پراکندگی داده‌ها که در این‌جا دنباله‌هایی از آپکدها هستند، را نشان می‌دهد. هرچه مقادیر ویژه‌ی، بردار ویژه بیشتر باشد یعنی پراکندگی داده، در آن جهت بیشتر بوده‌است و داده‌ها در آن جهت قابلیت تفکیک‌پذیری بیشتری دارند. غالب‌ترین بردار ویژه را انتخاب و اسم آن V گذاشته شده‌است. بر روی بردار V الگوریتم‌های مختلف انتخاب ویژگی اعمال می‌شود. اگر الگوریتم‌های انتخاب ویژگی بر روی یک سری بردار اعمال شوند، ازمیان درایه‌های موجود در بردارها، آن دسته از درایه‌هایی که اهمیت بیشتری دارند را انتخاب خواهندکرد. برای آن‌که آپکدهای بهتری انتخاب شود، از بین الگوریتم‌های موجود برای انتخاب ویژگی، نتیجه‌ی جمعی آن‌ها در نظر گرفته می‌شود. جدول ۲ پنج آپکد پراهمیت را با استفاده از الگوریتم‌های انتخاب ویژگی مختلف بر روی بردار V حاصل از آپکدهای سازنده‌ی فایل‌های PE نشان می‌دهد و جدول ۳ فرکانس تکرار آپکدهایی که در جدول ۲ بود را بیان می‌کند.

الگوریتم انتخاب ویژگی‌های شماره ۱ تا ۷ درون جدول ۳ از Toolbox آماده‌ی متلب هستند. این الگوریتم‌های انتخاب ویژگی اکثراً به این صورت کار می‌کنند که برای هر ویژگی یک امتیاز یا یک وزن محاسبه می‌کنند و با توجه به رتبه‌بندی‌کردن این امتیازها یا وزن‌ها، ویژگی‌های

جدول ۲ : نتیجه اعمال الگوریتم‌های انتخاب ویژگی مختلف برروی مجموعه داده فایل‌های PE

شماره	نام روش انتخاب ویژگی	نام کدهای عملیاتی (آپکدهای) انتخاب شده
۱	Relief-f	add, push, call, mov,lea
۲	FSV	Fcompp, fucompp, fcmovbe, popfw, fincstp
۳	Laplacian	Push ,mov , call, add,lea
۴	ECFS	Push,mov,call,add,jmp
۵	inffs	Push,mov,call,add,jmp
۶	fisher	Test,xor,adc,push, jnb
۷	Mutinffs	Cmpps, cvtpi2pd, lzcnt, maskmovdqu, mpsadb
۸	Gain ratio	Setnz, jl, neg, push ,jle

جدول ۳ : نمایش دفعات تکرار هر آپکد با الگوریتم‌های انتخاب ویژگی مختلف بر روی مجموعه داده اول

نام آپکد	فرکانس تکرار	نام آپکد	فرکانس تکرار	نام آپکد	فرکانس تکرار
push	۶	test	۱	mpsadbw	۱
add	۴	abd	۱	jnb	۱
mov	۴	Setnz	۱	fincstp	۱
call	۳	popfw	۱	neg	۱
lea	۲	fcmovbe	۱	Cmpps	۱
jmp	۲	fucompp	۱	cvtpi2pd	۱
jl	۱	Fcompp	۱	Lzcnt	۱
jle	۱	maskmovdqu	۱	Xor	۱

جدول ۴ : مقایسه میزان دقت کلاسه‌بندی با استفاده از تمام آپکدها و ۶ آپکد مهم، درروش ارائه شده توسط هاشمی [۵] برروی مجموعه داده اول

تعداد آپکدها / روش کلاسه‌بندی	با استفاده از همه‌ی آپکده‌ها	با استفاده از ۶ آپکد مهم
ماشین بردار پشتیبان	٪۸۸,۵۰	٪۸۷,۸۹
ک-نزدیکترین همسایه	٪۸۷,۶۸	٪۸۵,۷۱
کلاسه‌بندهای گروهی	٪۹۱,۱۴	٪۸۵,۳۰

$$\text{Entropy} = -\sum_{i=0}^{n} p(Xi)\log(p\ (Xi)) \quad (۱)$$

$$\text{Gain Split} = \text{Entropy of the whole file} - \left(\sum_{i=0}^{n}\left(\left(\frac{\text{ni}}{\text{n}}\right) * \quad Entropy(i)\right)\right. \quad (۲)$$

$$\text{Split Info} = -\sum_{i=0}^{n}\left(\left(\frac{\text{ni}}{\text{n}}\right) * \log\left(\frac{\text{ni}}{\text{n}}\right)\right) \quad (۳)$$

$$\text{Gain Ratio} = \frac{\text{Gain Split}}{\text{Split Info}} \quad (۴)$$

الگوریتم ۱: استخراج آپکدهای مهم از فایل‌ها

```
Input: Dataset
Output: Important Opcode
Step1: Dis_file= Disassembling( Dataset)
Step2: Opc_file= Extracting_Opcode( Dis_file)
Step3: Opc_matrix= Constracting_2gram_matrix( Opc_file)
Step4: Eigen_vector= decomposing(Opc_matrix)
Step5: V= Find_dominant_vector (Eigen_vector)
Step6: Important_Opcode= Vote(Feature_selection(V))
```

بارز در داده‌ها را که قابلیت تفکیک‌پذیری زیادی را دارند مشخص می کنند. الگوریتم انتخاب ویژگی Gain ratio براساس مجموعه روابط (۱) تا (۴) در این پژوهش به‌دست آمده‌است.

الگوریتم ۱ مراحل استخراج آپکدهای مهم از فایل‌های موجود در مجموعه داده‌ها را نشان می‌دهد.

با توجه به مطالب بیان‌شده و نتایج موجود در جداول ۲ و ۳، برخی از آپکدها وجودشان در دنباله‌های ۲-گِرمی مهم‌تر از بقیه‌ی آپکدها است و حضورشان در دنباله‌های ۲-گِرمی تفاوت بیشتری در مورد خانواده‌های بدافزارها و فایل‌های سالم و ماهیت آن‌ها ایجاد می‌کند، لذا الگوریتم‌های انتخاب ویژگی مختلفی، در غالب‌ترین بردار ویژه به آن‌ها به‌عنوان بُعد تاثیرگذار اشاره می‌کنند. برای بررسی صحت این مطلب و اثبات پراهمیت بودن آپکدهای مشخص‌شده در جدول ۳، یک آزمایش انجام می‌شود. برای این کار یک روش قابل‌قبول و با نتایج معتبر که با استفاده از کل آپکدها عملیات شناسایی بدافزارها را انجام داده‌است، با استفاده از آپکدهای پراهمیت جدول ۳ مجدداً آزمایش می‌شود و نتایج هر دو حالت اجرا، باهم مقایسه می‌شود. در بخش ۳-۲-۱- به بیان این آزمایش پرداخته می‌شود.

۳-۲-۱-آزمایش برای اثبات پراهمیت بودن برخی از آپکدها

در این قسمت از روش پیشنهادی هاشمی و همکاران [۵] استفاده خواهدشد، تا اعتبار آپکدهای مهم معرفی‌شده در این پژوهش بررسی شود. هاشمی و همکاران [۵] بعد از تجزیه‌کردن فایل‌ها در سطح اسمبلی شروع به ساخت گراف می‌کنند. بدین‌صورت که ترکیبات ۲-گِرمی از آپکدها را در نظر می‌گیرند و فرکانس تکرار این ترکیبات ۲-گِرمی برای هر فایل را در قالب یک ماتریس با نام W نمایش می‌دهند. با استفاده از روش پاور ایترشن [۳۱]، ماتریس W ساخته‌شده را به بردار ویژگی V تبدیل می‌کنند و سپس با استفاده از روش‌های یادگیری ماشین مانند ک-نزدیک‌ترین همسایه و ماشین بردار پشتیبانی داده‌ها را طبقه‌بندی می‌کنند. اکنون بعد از تجزیه‌کردن فایل‌ها در سطح اسمبلی و به‌دست آوردن دنباله‌ی آپکدها، ماتریس W، براساس آپکدهای پراهمیتی که به دست آورده‌شد، ساخته می‌شود. در این‌جا ماتریس W براساس ۶ آپکد {push, add, mov, call, jmp, lea} که با توجه به نتیجه‌ی جدول ۳ پراهمیت‌تر بودند، محاسبه می‌شود. سپس با استفاده از روش پاورایترشن که به آن اشاره شد، ماتریس W به بردار V تبدیل می‌شود. بردار V بردار ویژگی فایل‌های مجموعه داده، خواهدشد که ۶ ویژگی دارد. با استفاده از بردارهای ویژگی به‌دست‌آمده فایل‌های مجموعه داده، کلاسه‌بندی می شوند. جدول ۴، نتایج ساخت ماتریسW با همه‌ی آپکدها و ۶ آپکد پراهمیت را نشان می‌دهد.

جدول ۴، نتایج اجرا بر روی مجموعه داده‌ی اول که شامل فایل‌های PE است، نمایش می‌دهد. در بخش ۴ در مورد این مجموعه داده صحبت خواهدشد. تعداد آپکدهای بدون تکرار این مجموعه داده برابر با ۵۳۰ است. یعنی اگر همه‌ی ترکیبات دوتایی از آپکدها محاسبه شوند تعداد کل حالت‌های ممکن ۲۸۰۹۰۰ (۵۳۰ * ۵۳۰) خواهدشد. درصورتی‌که اگر با استفاده از ۶ آپکدها مهم که در جدول ۲ بیان شد، ماتریس W ساخته شود فقط ۳۶ (۶*۶)حالت، ترکیب دوتایی باید محاسبه شود. جدول ۴ نشان می‌دهد که نتایج به‌دست‌آمده با ۶ آپکد نسبتاً قابل‌قبول هستند و در زمان و فضای ذخیره‌سازی صرفه‌جویی چشم‌گیری خواهدشد.

بنابراین با توجه به آزمایش انجام‌شده به این نتیجه می‌توان رسید که برخی از آپکدها در فایل‌های بدافزار و سالم تفاوت بیشتری در مورد خانواده و ماهیت‌هاشان ایجاد می‌کنند. به عبارتی حضور برخی از آپکدهای در دنباله‌های تجزیه‌شده از فایل‌ها مهم‌تر از بقیه می‌باشد و می توان برای تشخیص سریع‌تر بدافزارها از آن‌ها استفاده نمود.

۴- داده‌های مورد آزمایش و نتایج اجرا

در این بخش ابتدا خصوصیات مجموعه داده و سیستمی که با آن روش پیشنهادی ارزیابی شده‌است، بیان می‌شود. سپس معیارهای ارزیابی روش پیشنهادی، بیان خواهدشد. درنهایت به بحث و مباحثه‌ی نتایج به دست‌آمده از روش پیشنهادی، پرداخته خواهدشد.

۴-۱- مجموعه داده‌ها

نتایج به‌دست‌آمده براساس ۴ مجموعه داده بیان شده‌است. یکی ازمجموعه داده‌ها مربوط به فایل‌های آندروید است و سه مجموعه داده ی دیگر، برای فایل‌های PE می‌باشد. مشخصات آن‌ها عبارت‌اند از:

۴-۱-۱- مجموعه داده‌ی اول (فایل‌های PE)

مجموعه داده‌ی غیربدافزار استفاده‌شده در این کار شامل، ۸۵۰ فایل " EXE" و ۷۵۰ فایل "DLL " است که از پوشه‌های "Program Files" و "System32 "یک ویندوز استاندارد استخراج‌شده‌اند. این مجموعه داده با استفاده از آنتی‌ویروس Node32 اسکن شده‌است و اطمینان حاصل گردیده‌است، که فایل‌های آن غیر بدافزار هستند. اندازه این فایل‌ها از ۱ کیلوبایت تا ۱۵ مگابایت است.

فایل‌های بدافزار از پایگاه‌داده " VX Heavens Virus Collection" دریافت شدند. از تمام بدافزارهای موجود در این پایگاه‌داده، ۱۶۰۰ بدافزار به‌صورت تصادفی انتخاب شده‌اند. این فایل‌ها شامل انواع مختلف بدافزارها مانند درب‌پشتی، کرم، تروجان، ویروس و ابزار هک هستند. اندازه فایل‌های انتخاب‌شده بین ۱ کیلوبایت تا ۱۱ مگابایت است.

۴-۱-۲- مجموعه داده‌ی دوم (فایل‌های آندروید)

تعداد فایل‌های سالم در این مجموعه داده ۲۰۰۰ است. این فایل‌ها از Google Play store دانلود گردیده‌است و با VirusTotal service اسکن شده‌اند تا اطمینان حاصل شود که، بدافزار نیستند. اندازه فایل‌های انتخاب‌شده بین ۹ کیلوبایت تا ۹۶ مگابایت است.

برای داده‌های بدافزار این قسمت، از مجموعه داده Drebin، استفاده شده‌است. Drebin ، یک مجموعه داده برای فایل‌های بدافزار می‌باشد که شامل نمونه‌هایی از ۱۷۹ خانواده‌ی بدافزارهای آندروید است. این مجموعه داده از سرویس‌های مانند: Android Malware Genome Project, Google Play store, Russian and Chinese markets, Android app websites, malware forums and security blogs انتخاب شده‌است. بعد از دانلود این مجموعه داده، فایل‌های آن با VirusTotal service اسکن شده‌اند، تا اطمینان حاصل شود که بدافزار هستند. این مجموعه داده شامل ۵۵۶۰ فایل بدافزار بوده‌است که به‌صورت تصادفی ۲۰۰۰ فایل از آن‌ها انتخاب شده‌است. اندازه فایل‌های انتخاب‌شده بین ۱۰ کیلوبایت تا ۲۴ مگابایت است.

درروش پیشنهادی با تحلیل ۲ مجموعه داده قبل، آپکدهای پراهمیت به‌ترتیب برای مجموعه فایل‌های PE و آندروید شناسایی می شود. اکنون لازم است مجموعه داده‌های دیگری انتخاب شود تا با استفاده از آن و آپکدهای مهم به‌دست‌آمده، عملیات کلاسه‌بندی انجام شود. برای تحقق این هدف از مجموعه داده‌ی سوم (کگل یا ماکروسافت ۲۰۱۵) و مجموعه داده‌ی چهارم (فایل‌های بسته‌بندی‌شده) استفاده می شود.

۴-۱-۳- مجموعه داده‌ی سوم (کگل یا ماکروسافت ۲۰۱۵)

مجموعه داده غیر بدافزار استفاده‌شده در این کار، فایل‌های اجرایی از یک ویندوز استاندارد می‌باشد. این مجموعه داده با استفاده از آنتی ویروس Node32 اسکن شده و اطمینان حاصل شده‌است که فایل‌های آن غیر بدافزار هستند. اندازه این فایل‌ها از ۱ کیلوبایت تا ۱۵ مگابایت است.

از داده‌هایی که در چالش کلاسه‌بندی بدافزارهای توسط ماکروسافت با عنوان Big 2015 در سال ۲۰۱۵ ارائه شد برای پایگاه داده‌ی بدافزار این مجموعه استفاده شد. این مجموعه داده به ۲ فرمت فایل‌های اجرایی و فایل‌های تجزیه‌شده در سطح اسمبلی، در قالب ۹ خانواده از بدافزارها معرفی شد که در هر خانواده تعدادی بدافزار وجود دارد. از آن‌جایی که روش پیشنهادی این پژوهش براساس آپکدها سازنده ی فایل‌ها مسئله‌ی کلاسه‌بندی را حل می‌کند از فایل‌های تجزیه‌شده در سطح اسمبلی آن، کمک گرفته می‌شود و جمعا ۱۶۰۰ فایل به‌صورت تصادفی از همه‌ی خانواده‌ها انتخاب شده‌است.

۴-۱-۴- مجموعه داده‌ی چهارم (فایل‌های بسته‌بندی‌شده)

ازآن‌جایی که مجموعه داده‌ای در قالب فایل‌های فشرده و بسته‌بندی‌شده بدافزار و سالم به‌صورت جامع وجود ندارد. در این پژوهش مجموعه داده اول که مربوط به فایل‌های اجرایی است، با استفاده از بسته‌بندی‌کننده UPX ، بسته‌بندی و فشرده شده‌اند و سپس در سطح اسمبلی تجزیه می‌شوند. درنهایت از این آپکدهای به‌دست‌آمده، برای آزمایش روش پیشنهادی استفاده می‌شود. UPX، یک بسته‌بندی‌کننده معروف است که کدهای منبع آن به‌صورت رایگان موجود می‌باشد و از بین بسته‌بندی کننده‌های رایگان موجود، محبوب‌ترین و خوش‌نام‌ترین می‌باشد. لذا از آن برای بسته‌بندی و فشرده‌کردن فایل‌ها استفاده شده‌است. سایز فایل‌های بسته‌بندی‌شده بین ۱ کیلوبایت تا ۱۵ مگابایت می‌باشد.

۴-۲- اعتبارسنجی متقابل

اعتبارسنجی متقابل روشی برای اعتبارسنجی مدل تولیدشده است، تا بتوان برخی مسائل در رابطه با ساخت مدل‌ها مانند بیش برازش[۱۲] را حل کند. یکی از روش‌های خوب برای اعتبارسنجی، اعتبارسنجی k دسته‌ای است. درروش k دسته‌ای، مجموعه داده‌ها به k دسته مساوی تقسیم می شود. در هر بار یکی از k دسته به‌عنوان مجموعه داده آزمایش انتخاب می‌شود و k-1 دسته باقی‌مانده به‌عنوان مجموعه داده آموزشی برای ساخت مدل انتخاب می‌شود. این کار k بار تکرار خواهدشد و میانگین دقت تمامی مراحل به‌عنوان دقت نهایی انتخاب خواهدشد. در این پژوهش برای ارزیابی مدل‌ها از اعتبارسنجی به‌صورت ۱۰ دسته ۱۰ بار اجرا استفاده شده‌است.

۴-۳- معیارهای ارزیابی

برای ارزیابی صحت عمل‌کرد روش پیشنهادی خود از معیارهای ارزیابی یادگیری ماشین مانند نرخ مثبت صحیح (TPR)، نرخ مثبت کاذب (FPR)، دقت و معیار-F استفاده شده‌است، که در جدول ۵ روابط آن‌ها بیان شده است.

- مثبت صحیح (True Positive): تعداد فایل‌های بدافزارهایی است که به‌درستی تخمین زده شده‌اند.
- منفی کاذب (False Negative):تعداد بدافزارهایی است که اشتباهاً به‌عنوان غیربدافزار تخمین زده شده‌اند.
- مثبت کاذب (False Positive): تعداد فایل‌های غیربدافزارهایی است که اشتباهاً به بدافزار کلاسه‌بندی شده‌اند.
- منفی صحیح (True Negative): تعداد فایل‌های غیر بدافزاری می‌باشد که به‌درستی کلاسه‌بندی شده‌اند.

۴-۴- راه‌اندازی آزمایشی

در این پژوهش از کلاسه‌بندهای ماشین بردار پشتیبان، ک-نزدیکترین همسایه، جنگل تصادفی و کلاسه‌بندهای گروهی [۳۱] استفاده شده

جدول ۵: معیارهای ارزیابی

معیار	فرمول
نرخ مثبت صحیح	$\frac{TP}{TP+FN}$
نرخ مثبت کاذب	$\frac{FP}{FP+TN}$
دقت	$\frac{TP+TN}{TP+FP+TN+FN}$
بازخوانی	$\frac{TP}{TP+FN}$
صحت	$\frac{TP}{TP+FP}$
معیار- F	$2*\frac{Presision*Recall}{Precision+Recall}$

است. درروش ماشین بردار پشتیبان از کرنل RBF استفاده شد که پارامترهای آن تنظیم شده‌اند. درروش ک-نزدیکترین همسایه، تعداد همسایه‌های مختلفی از ۱ تا ۱۰ همسایه در نظر گرفته شد. کلاسه‌بندهای گروهی از نوع آدابوست با ۱۰۰۰ یادگیرنده ضعیف از نوع درخت انتخاب شده‌است.

مراحل اجرای روش پیشنهادی با استفاده از کدنویسی در متلب ۲۰۱۵ و پایتون ۲,۷ انجام شده‌است، که روی یک سیستم AMD FX(tm)-6200 six-Core Processor 3.80 GHz با ۳۲ گیگابایت حافظه ی اصلی اجرا شده‌است. برای تجزیه‌کردن فایل‌های اجرایی PE در سطح اسمبلی از نرم‌افزار IDA pro استفاده شده‌است و برای تجزیه‌کردن فایل‌های مجموعه داده آندروید، از نرم‌افزارهای Baksmali و Androguard کمک گرفته شده‌است.

۴-۵- نتایج اجرا

در این قسمت نتایج به‌دست‌آمده بررسی خواهدشد. برای این کار روش پیشنهادی با توجه به معیارهای معرفی‌شده در قسمت قبل به‌صورت عادلانه با روش‌های تحلیل آپکد مانند، پژوهش هاشمی و همکارانش[۵] مقایسه خواهدشد. از آن‌جایی که در این تحقیق مسئله کلاسه‌بندی بدافزارها با استفاده از آپکدهای تشکیل‌دهنده‌ی فایل‌های اجرایی به مسئله‌ی کلاسه‌بندی تصاویر نگاشته می‌شود، نتایج چند پژوهش که مسئله کلاسه‌بندی بدافزارها را به مسئله‌ی کلاسه‌بندی جدید تبدیل می کند، مانند پژوهش فرخمنش و همکارانش [۲۵]، ناتاراج و همکارانش [۲۶] با نتایج این پژوهش مقایسه خواهدشد تا بتوان مقایسه‌ای بین نتایج و چالش‌های تبدیل مسئله کلاسه‌بندی بدافزارها با این پژوهش داشت.

روش هاشمی و همکارانش [۵] یک روش تحلیل آپکد قوی است که از تمام آپکدهای سازنده‌ی فایل‌های اجرایی گراف می‌سازد و این گراف‌ها را با استفاده از روش پاورایترشن [۳۱] به بهترین بردار ویژه‌ی خود همگرا می‌کند. روش هاشمی از برخی روش‌های نسبتاً دقیق و سریع مانند، سانتوس و همکارانش [۸] و اسکندری و همکارانش [۳۲] با توجه به نتایجی که به‌دست‌آوردند بهتر عمل می‌کند، لذا این پژوهش با کار ایشان مقایسه خواهدشد.

در پژوهش ناتاراج [۲۶] بایت‌های حاصل از هر فایل بدافزار و سالم به یک تصویر خاکستری نگاشته خواهدشد و با استفاده از GIST [۲۷] از آن‌ها ویژگی استخراج خواهدشد. لذا این پژوهش با کار ایشان مقایسه می‌شود تا نشان داده شود که عکس‌هایی که با استفاده از آپکدهای مهم ساخته می‌شوند عکس‌های بسیار مناسب‌تری نسبت به عکس‌هایی هستند که با استفاده از تمام بایت‌های خام ساخته می‌شوند.

در کار فرخمنش و همکارانش [۲۵] بخشی از بایت‌های هر فایل بدافزار و سالم به یک سیگنال صوتی تبدیل خواهدشد و درنهایت با توجه به سیگنال‌های تولیدشده عمل استخراج ویژگی و کلاسه‌بندی انجام خواهدشد، به عبارتی آن‌ها بررسی‌های خود را در فضای صوت انجام داده‌اند و در این بخش تلاش شده‌است مقایسه‌ای بین کارکردن در فضای صوت و تصویر انجام شود.

مدت زمان تجزیه‌کردن فایل‌های PE و استخراج آپکد از فایل‌ها به مدت زمان روش پیشنهادی و روش هاشمی که مبتنی بر آپکد هستند، افزوده شده‌است. نتایج این مقایسه در جداول ۶ و۷ بیان شده‌است.

تحلیل‌های انجام‌شده بر روی فایل‌های PE، قابل تعمیم به مجموعه داده‌های دیگر مانند آندروید هم است. اگر مراحل قبل، بر روی فایل‌های تجزیه شده‌ی آندروید انجام شود آپکدهای مهم، مانند جدول ۸ به‌دست خواهدآمد. مانند آن‌چه که در قسمت‌های قبل گفته شد ترکیبات ۲-گرمی از این آپکدها در نظر گرفته‌می‌شوند و روش پیشنهادی برروی آن‌ها آزمایش می‌شود. جداول ۹ و ۱۰ نتایج این آزمایش را نشان می دهد.

همان‌طور که مشاهده می‌شود نتایج به‌دست‌آمده نشان می‌دهد روش پیشنهادی مقاله در اکثر موارد نرخ مثبت صحیح، نرخ مثبت کاذب، دقت و معیار-F بهتری نسبت به روش پیشنهادی هاشمی، فرخمنش و ناتاراج داشته‌است و عکس‌هایی که با استفاده از آپکدهای مهم ساخته می‌شوند، می‌توانند معیار مناسبی برای تشخیص فایل بدافزار از غیر بدافزار باشد که در زمان و حافظه صرفه‌جویی می‌کند و دقت بالایی را به ارمغان می‌آورد. دلیل این امر این است که آپکدهای برجسته و مهم از فایل‌ها انتخاب شده‌اند و با استفاده از این آپکدها که قابلیت تشخیص بالایی دارند مقایسه صورت گرفته‌است.

جدول ۶: نتایج میانگین زمان اجرای یک نمونه داده‌ی آزمایش از مجموعه داده‌ی اول بعد از اعمال ۴ نوع کلاسه‌بند

روش	زمان بر حسب ثانیه
روش پیشنهادی	۱/۱۱
روش هاشمی	۹۷/۱۸
روش ناتاراج	۰/۸۱
روش فرخمنش	۲.۸۱

جدول ۷: مقایسه میزان دقت، معیار Fنرخ مثبت صحیح و نرخ مثبت کاذب روش‌های ذکرشده با توجه به مجموعه داده‌ی اول

روش / کلاسه‌بند		ک-نزدیک‌ترین همسایه با k=۱	ک-نزدیک‌ترین همسایه با k=۲	ک-نزدیک‌ترین همسایه با k=۳	ک-نزدیک‌ترین همسایه با k=۴	ک-نزدیک‌ترین همسایه با k=۵	ک-نزدیک‌ترین همسایه با k=۶	ک-نزدیک‌ترین همسایه با k=۷	ک-نزدیک‌ترین همسایه با k=۸	ک-نزدیک‌ترین همسایه با k=۹	ک-نزدیک‌ترین همسایه با k=۱۰	ماشین بردار پشتیبان	کلاسه‌بندهای گروهی	جنگل تصادفی
دقت	روش پیشنهادی	۸۶/۰	۸۴/۰	**۸۵/۷**	۸۵/۳	۸۶/۶	۸۵/۰	۸۵/۰	۸۴/۸	۸۵/۵	۸۳/۸	**۹۱/۸**	۹۱/۲	۹۰/۸
	هاشمی	۸۵/۶	۸۴/۲	۸۷/۶	۸۵/۳	۸۷/۶	۸۶/۲	۸۷/۲	۸۷/۰	۸۷/۴	۸۷/۷	۸۸/۵	۹۱/۱	**۹۱/۶**
	ناتاراج	۷۶/۴	۷۳/۴	۷۸/۰	۷۵/۵	۷۸/۶	۷۸/۸	۸۰/۰	۷۸/۵	۷۹/۶	۷۹/۶	**۸۱/۷**	۷۶/۳	۸۱/۶
	فرخمنش	۸۶/۰	۸۶/۰	۸۹/۷	۸۹/۵	۹۰/۴	۹۰/۰	۸۹/۸	۹۰/۱	۸۹/۸	۸۹/۸	۸۷/۹	**۹۲/۲**	۹۲/۰
معیار-F	روش پیشنهادی	۸۵/۹	۸/۵	۸۵/۴	۸۴/۲	۸۶/۴	۸۳/۹	۸۵/۶	۸۳/۵	۸۴/۷	۸۲/۱	**۹۱/۶**	۹۱/۲	۹۰/۹
	هاشمی	۸۵/۴	۸۲/۸	۸۷/۵	۸۴/۵	۸۷/۵	۸۵/۶	۸۷/۱	۸/۵	۸۷/۲	۸۷/۲	۸۸/۸	۹۱/۷	**۹۱/۸**
	ناتاراج	۷۶/۶	۷۶/۷	۷۸/۱	۷۷/۱	۷۸/۳	۷۹/۷	۷۹/۸	۷۹/۳	۷۹/۴	۷۸/۶	۸۱/۲	۷۵/۱	**۸۱/۶**
	فرخمنش	۸۶/۰	۸۶/۰	۸۹/۶	۸۹/۴	۹۰/۲	۸۹/۲	۸۹/۶	۹۰/۰	۸۹/۷	۸۹/۷	۸۷/۹	**۹۲/۲**	۹۲/۰
نرخ مثبت صحیح	روش پیشنهادی	۸۲/۷	۷۳/۲	۸۱/۵	۷۶/۱	۸۱/۱	۷۶/۱	۷۹/۵	۷۵/۱	۷۷/۸	۷۲/۶	۸۷/۱	۸۸/۵	**۹۳/۴**
	هاشمی	۸۱/۸	۷۳/۲	۸۴/۱	۷۷/۹	۸۴/۵	۷۹/۶	۸۳/۹	۸۱/۱	۸۳/۹	۸۱/۸	۸۸/۹	۹۱/۹	**۹۲/۶**
	ناتاراج	۷۷/۷	۸۷/۹	۷۸/۷	۸۲/۷	۷۷/۳	**۸۳/۳**	۷۹/۱	۸۲/۵	۷۹/۱	۷۵/۳	۷۹/۳	۷۱/۱	۷۹/۰
	فرخمنش	۸۶/۰	۸۶/۰	۸۹/۶	۸۹/۴	۹۰/۲	۹۰/۰	۸۹/۶	۹۰/۰	۸۹/۷	۸۹/۷	۸۷/۹	**۹۲/۲**	۹۲/۲
نرخ مثبت کاذب	روش پیشنهادی	۱۰/۴	۴/۷	۹/۸	۴/۹	۷/۰	۵/۵	۶/۶	۴/۹	۶/۴	۴/۵	**۳/۲**	۵/۹	۱۱/۸
	هاشمی	۱۱/۵	**۴/۹**	۹/۱	۶/۹	۹/۲	۶/۸	۹/۴	۹/۴	۹/۰	۶/۲	۱۲/۰	۹/۲	۹/۲
	ناتاراج	**۱۵/۰**	۴۲/۰	۲۲/۸	۳۱/۸	۲۰/۲	۲۵/۸	۱۹/۰	۲۵/۶	۲۰/۰	۱۶/۲	۱۶/۰	۱۹/۲	۱۶/۰
	فرخمنش	۱۴/۰	۱۴/۰	۱۰/۴	۱۰/۶	۹/۸	۱۰/۰	۱۰/۴	۱۰/۰	۱۰/۳	۱۰/۳	۱۲/۱	**۷/۰**	۸/۰

جدول ۸: آپکدهای مهم فایل‌های آندروید براساس رتبه اهمیتشان

رتبه	۱	۲	۳	۴	۵	۶	۷
نام آپکد	array-length	move	Input-object	long/2addr	If-eq	Return-void	iget

جدول ۹: نتایج میانگین زمان اجرای یک نمونه داده‌ی آزمایش از مجموعه داده‌ی آندروید بعد از اعمال ۴ نوع کلاسه‌بند

	روش
۱/۱۹	روش پیشنهادی
۱۵۵/۷۵	روش هاشمی
۳/۷۷	روش ناتاراج
۲/۵۲	روش فرخمنش

۵- بحث در مورد نتایج

در بخش سوم، مجموعه‌ای از آپکدها به‌عنوان آپکدهای مهم، شناسایی شدند، که با توجه به این آپکدها و حضورشان در دنباله‌های N-گرمی می‌توان فایل‌های بدافزار از سالم را، تشخیص داد. در این قسمت ابتدا تاثیر هر آپکد مهم به تنهایی در فایل بررسی خواهدشد و سپس برروی مجموعه داده‌های دیگری، روش پیشنهادی بررسی می‌شود تا نشان داده شود که آپکدهای مهم به‌دست‌آمده مختص یک مجموعه داده‌ی خاص نیست و برروی مجموعه داده‌های دیگر هم جواب می‌دهد.

۵-۱- آزمایش برای بررسی تاثیر آپکدهای مهم در شناسایی بدافزارها

هر فایل بعد از تجزیه‌شدن در سطح اسمبلی به آپکدهای سازنده، معادل با یک دنباله از آپکدهای پشت سر هم خواهدشد. ابتدا یکی از آپکدهای مشخص‌شده در جدول ۳ که در بخش ۳ محاسبه شد، معادل با عدد ۱ و مابقی آپکدهای موجود در فایل، معادل با ۰ در نظر گرفته می‌شود. این کار برای آن است که بقیه آپکدها یکسان در نظر گرفته شود و تاثیر وجود یک آپکد خاص در فایل‌ها بررسی گردد. با توجه به این عمل روشن خواهدشد حضور آن آپکد خاص در فایل‌های بدافزارها و فایل‌های سالم چه میزان تفاوت در مورد ماهیت‌هایشان ایجاد خواهدکرد. به‌عنوان مثال اگر بخشی از آپکدهای سازنده‌ی یک فایل دنباله‌ای به‌صورت test, mov, {add ,jmp, push, add, pop, call, push, neg} داشته‌باشد و هدف بررسی تاثیر آپکد push باشد، به‌جای آپکد push عدد ۱ قرار داده می‌شود و به‌جای مابقی آپکدها عدد ۰ قرار می‌گیرد. از این طریق یک دنباله‌ی ۰،۱ به‌دست خواهدآمد. با توجه به این دنباله‌ها، یک تصویر سیاه وسفید ساخته می‌شود. برای این‌که پردازش و استخراج ویژگی از تصاویر راحت‌تر باشد از دنباله‌ی ۰و۱ های به‌دست‌آمده عکس مربعی ساخته می‌شود، برای این منظور نزدیک‌ترین عدد مربعی به طول دنباله‌ی از هر فایل محاسبه می‌شود و در صورت لزوم با چسباندن صفر
به هر دنباله و یا صرف‌نظرکردن چند عنصر از آن‌ها به دنباله‌های مربعی تبدیل می‌شوند. دنباله‌ی به‌دست‌آمده به یک ماتریس مربعی تغییر شکل می‌یابد و درنهایت به یک عکس تبدیل می‌شود. برای همه‌ی فایل‌های تجزیه‌شده در سطح اسمبلی از مجموعه داده‌ها این فرآیند انجام می‌شود. سپس با استفاده از روش GIST [۲۷] که در بخش ۳ معرفی شد، از این عکس‌ها ویژگی استخراج می‌شود. از ویژگی‌های به‌دست‌آمده برای کلاسه‌بندی فایل‌ها به ۲ دسته‌ی بدافزار و سالم استفاده می‌شود. جدول ۱۱ نتایج میزان دقت فرآیند کلاسه‌بندی از این تحلیل را نشان می‌دهد. در این جدول، در هر ستون بیان می‌کند با استفاده از آپکد مشخص‌شده، چه میزان دقت برای فرآیند کلاسه‌بندی به‌دست آمده‌است. همین کار برروی فایل‌های آندروید قابل انجام است. جدول ۱۲ نتایج این آزمایش بر روی فایل‌های آندروید را نمایش می‌دهد.

در جدول ۱۱ و ۱۲ مابقی آپکدهای تشکیل‌دهنده‌ی فایل‌های PE و آندروید نتایجی مانند آپکد jl و jle دارند، که تقریباً معادل با حالت کلاسه‌بندی تصادفی می‌باشد، لذا از بیان آن‌ها در جداول صرف‌نظر شده است.

با توجه به جداول، مشخص می‌شود در فایل‌های اجرایی آپکدهایی مانند{ push, add, mov } و در فایل‌های آندروید آپکدهایی مانند{ Iput-object ،move ،array-length } که الگوریتم‌های انتخاب ویژگی مختلفی به آن‌ها اشاره کرد، تفاوت‌های بیشتری در مورد ماهیت‌های فایل‌های بدافزارها و فایل‌های سالم ایجاد خواهندکرد، لذا می‌توان از آن‌ها برای تشخیص سریع‌تر فایل‌های بدافزارها استفاده نمود.

۵-۲- آزمایش تاثیر آپکدهای مهم در شناسایی بدافزارها در مجموعه داده‌های دیگر

برای آن‌که نشان داده شود، آپکدهای پراهمیت معرفی‌شده خاص یک مجموعه داده نیست از مجموعه داده سوم استفاده می‌شود. این مجموعه داده برای صحت‌سنجی آپکدهای پراهمیت به‌دست‌آمده در این پژوهش، بیان شده‌است. درروش پیشنهادی با تحلیل ۲ مجموعه داده اول و دوم آپکدهای پراهمیت برای فایل‌های PE و آندروید شناسایی می‌شود. اکنون لازم است مجموعه داده‌های دیگری انتخاب شوند تا با استفاده از آن‌ها و آپکدهای مهم به‌دست‌آمده، عملیات کلاسه‌بندی انجام شود. برای تحقق این هدف از مجموعه داده مربوط به چالش ماکروسافت، استفاده

می‌شود. در این آزمایش با استفاده از آپکدهای پراهمیت فایل‌های PE که در قسمت ۳ به‌دست آمد، روش پیشنهادی اجرا می‌شود. در جدول ۱۳ نتایج ساخت عکس و کلاسه‌بندی فایل‌ها با استفاده از تمام آپکدها و روش پیشنهادی بیان شده‌است تا نشان داده شود، با درنظرگرفتن چند آپکد مهم، نه تنها از دقت تشخیص بدافزارها کاسته نمی‌شود، بلکه به موثرتربودن روش تشخیص کمک می‌شود و فرآیند کلاسه‌بندی، بهتر انجام می‌شود.

برای این‌که نشان داده شود تاثیر روش پیشنهادی بر روی فایل‌های بسته‌بندی‌شده به چه صورت می‌باشد از مجموعه داده چهارم که با استفاده از ابزار UPX در این پژوهش بسته‌بندی شده‌است، استفاده می شود. سپس فایل‌های بسته‌بندی‌شده را به روشی که فایل‌های عادی در سطح اسمبلی تجزیه می‌شدند، تجزیه می‌شوند، واضح است که آپکدهای به‌دست‌آمده، آپکدهای اصلی سازنده‌ی فایل‌ها نخواهدبود، درواقع آپکدهایی هستند که توسط بسته‌بندی‌کننده ایجادشده‌اند و برای از خارج‌نمودن برنامه از حالت فشرده، جهت اجرا استفاده می‌شوند. لذا با توجه به این موضوع، دقت روش پیشنهادی بر روی فایل‌ها بسته‌بندی شده کمتر می‌باشد. نتایج این آزمایش در جدول ۱۴ بیان شده‌است. در این جدول هم مانند جدول ۱۳، نتایج ساخت عکس و کلاسه‌بندی آن‌ها با استفاده از تمام آپکدها، به همراه آپکدهای مهم بیان شده‌است.

نتایج به‌دست‌آمده نشان می‌دهد آپکدهایی که درروش پیشنهادی شناسایی شدند، آپکدهای مهمی می‌باشند و عکس‌هایی که با استفاده از آپکدهای مهم ساخته می‌شوند، می‌تواند معیار مناسبی برای تشخیص فایل بدافزار باشد که دقت بالایی را به ارمغان می‌آورد.

جدول ۱۰ : مقایسه میزان دقت، معیار F، نرخ مثبت صحیح و نرخ مثبت کاذب روش‌های ذکرشده با توجه به مجموعه داده‌ی آندورید

	روش / کلاسه‌بند	ک-نزدیک‌ترین همسایه با k=۱	ک-نزدیک‌ترین همسایه با k=۲	ک-نزدیک‌ترین همسایه با k=۳	ک-نزدیک‌ترین همسایه با k=۴	ک-نزدیک‌ترین همسایه با k=۵	ک-نزدیک‌ترین همسایه با k=۶	ک-نزدیک‌ترین همسایه با k=۷	ک-نزدیک‌ترین همسایه با k=۸	ک-نزدیک‌ترین همسایه با k=۹	ک-نزدیک‌ترین همسایه با k=۱۰	ماشین بردار پشتیبان	کلاسه‌بندهای گروهی	جنگل تصادفی
دقت	روش پیشنهادی	۸۸/۵	۸۹/۵	۸۶/۸	۸۶/۸	۸۵/۸	۸۶/۹	۸۵/۸	۸۵/۸	۸۴/۵	۸۵/۳	**۹۳/۵**	۹۰/۵	۹۱/۱
	هاشمی	۸۹/۸	۸۷/۱	۸۸/۷	۸۶/۲	۸۶/۸	۸۵/۸	۸۶/۳	۸۵/۱	۸۵/۸	۸۵/۳	**۹۲/۳**	۹۱/۷	۹۱/۹
	ناتاراج	**۷۳/۹**	۷۱/۸	۷۰/۳	۷۰/۴	۶۸/۸	۶۹/۵	۷۰/۰	۶۹/۲	۶۸/۲	۶۶/۵	۷۳/۷	۶۹/۹	۷۳/۸
	فرخمنش	۸۵/۷	۸۶/۲	۸۴/۴	۸۵/۱	۸۴/۳	۸۴/۲	۸۳/۶	۸۳/۰	۸۳/۰	۸۲/۷	۸۸/۱	**۸۸/۵**	۸۸/۴
معیار-F	روش پیشنهادی	۸۹/۱	۸۹/۵	۸۷/۵	۸۷/۰	۸۶/۵	۸۷/۱	۸۶/۶	۸۶/۱	۸۵/۲	۸۵/۹	**۹۳/۴**	۹۰/۷	۹۱/۷
	هاشمی	۹۰/۰	۸۸/۱	۸۸/۹	۸۶/۹	۸۷/۰	۸۶/۴	۸۶/۳	۸۵/۷	۸۵/۹	۸۵/۸	**۹۲/۱**	۹۱/۸	۹۱/۴
	ناتاراج	**۷۴/۸**	۶۸/۵	۷۰/۷	۶۷/۴	۶۸/۷	۶۷/۲	۶۹/۸	۶۶/۷	۶۷/۸	۶۴/۰	۷۰/۵	۶۸/۷	۷۳/۸
	فرخمنش	۸۵/۷	۸۵/۸	۸۴/۳	۸۴/۷	۸۴/۱	۸۳/۷	۸۳/۶	۸۲/۴	۸۲/۷	۸۲/۲	۸۷/۷	**۸۸/۰**	**۸۸/۰**
نرخ مثبت صحیح	روش پیشنهادی	۹۱/۵	۸۷/۱	۹۱/۰	۸۷/۶	۸۹/۱	۸۷/۶	۹۰/۶	۸۸/۱	۹۰/۶	۸۸/۱	**۹۲/۰**	۹۰/۱	۹۱/۹
	هاشمی	۹۱/۶	۹۵/۴	۹۱/۰	۹۱/۸	۸۸/۴	۹۰/۲	۸۶/۶	۸۹/۰	۸۶/۶	۸۸/۸	۸۹/۲	**۹۲/۸**	۸۸/۵
	ناتاراج	**۷۷/۴**	۶۱/۲	۷۱/۸	۶۱/۲	۶۴/۴	۶۲/۶	۶۹/۲	۶۱/۶	۶۶/۸	۵۹/۶	۶۲/۸	۶۶/۲	۶۸/۴
	فرخمنش	۸۵/۷	۸۶/۲	۸۴/۴	۸۵/۱	۸۴/۳	۸۴/۲	۸۳/۶	۸۳/۰	۸۳/۰	۸۲/۷	۸۸/۱	**۸۸/۵**	۸۸/۴
نرخ مثبت کاذب	روش پیشنهادی	۱۴/۵	۸/۱	۱۹/۱	۱۳/۱۱	۱۹/۶	۱۵/۱	۱۸/۶	۱۴/۶	۲۰/۱	۱۷/۱	**۵/۱**	۱۵/۱	۸/۹
	هاشمی	۱۲/۰	۲۱/۲	۱۲/۶	۹/۴	۱۴/۸	۱۸/۶	۱۴/۰	۱۹/۸	۱۵/۲	۱۸/۲	**۴/۶**	۹/۴	۵/۱
	ناتاراج	**۲۹/۶**	۱۷/۶	۳۱/۲	۳۰/۴	۳۰/۸	۲۳/۴	۲۹/۲	۲۲/۲	۳۰/۴	۲۶/۶	**۱۵/۴**	۲۶/۲	۱۸/۴
	فرخمنش	۱۷/۵	۲۱/۶	۲۰/۲	۲۲/۸	۲۱/۵	۲۴/۲	۲۳/۲	۲۵/۹	۲۴/۴	۲۶/۱	۲۰/۱	۱۹/۸	**۱۹/۹**

جدول۱۱ : نتایج میزان دقت از کلاسه‌بندی فایل‌های مجموعه داده اول با استفاده از هر آپکد مهم

آپکد / کلاسه‌بند	push	add	mov	call	jmp	lea	jle	jl
ماشین بردار پشتیان	۷۶/۹	۷۷/۰	۸۰/۷	۷۹/۲	۸۰/۳	۷۰/۹	-	۴۵/۸
ک-نزدیکترین همسایه	۸۱/۰	۷۶/۱	۷۷/۰	۷۸/۶	۷۸/۶	۷۱/۰	۴۹/۱	۵۲/۱
کلاسه‌بندهای گروهی	۷۸/۲	۷۵/۰	۸۰/۰	۷۶/۴	۸۱/۱	۷۹/۰	۴۷/۳	۴۹/۲

جدول ۱۲ : نتایج میزان دقت از کلاسه‌بندی فایل‌های آندوید با استفاده از هر آپکد مهم

آپکد / کلاسه‌بند	array-length	move	Iput-object	div-long/2addr	If-eq	Return-void	iget
ماشین بردار پشتیان	۹۳/۲	۷۹/۵	۸۴/۲	۷۲/۷	۷۶/۰	۸۳/۲	۸۱/۷
ک-نزدیکترین همسایه	۹۳/۵	۸۱/۲	۸۲/۷	۷۲/۵	۷۸/۰	۸۰/۰	۸۲/۵
کلاسه‌بندهای گروهی	۹۳/۲	۷۲/۲	۸۰/۰	۹۴/۱	۷۴/۵	۷۶/۷	۸۱/۲

جدول ۱۳: نتایج روش پیشنهادی روی مجموعه داده سوم

معیار / کلاسه بند	دقت		معیار F		نرخ مثبت صحیح		نرخ مثبت کاذب	
	روش پیشنهادی	کل آپکدها	روش پیشنهادی	کل آپکدها	روش پیشنهادی	کل آپکدها	روش پیشنهادی	کل آپکدها
ک-نزدیک‌ترین همسایه با k=1	۹۱/۷	۸۶/۰	۹۱/۹	۸۵/۹	۹۱/۲	۸۲/۷	۷/۷	۱۰/۵
ک-نزدیک‌ترین همسایه با k=2	۹۲/۱	۸۴/۰	۹۱/۹	۸۲/۵	۸۷/۶	۷۳/۲	**۳/۲**	۴/۷
ک-نزدیک‌ترین همسایه با k=3	۹۲/۱	۸۵/۷	۹۲/۲	۸۵/۴	۹۰/۶	۸۱/۵	۶/۴	۹/۹
ک-نزدیک‌ترین همسایه با k=4	۹۱/۱	۸۵/۳	۹۰/۸	۸۴/۲	۸۶/۲	۷۶/۱	۳/۹	۴/۹
ک-نزدیک‌ترین همسایه با k=5	۹۰/۸	۸۶/۷	۹۰/۸	۸۶/۴	۸۸/۲	۸۱/۱	۶/۴	۷/۱
ک-نزدیک‌ترین همسایه با k=6	۹۰/۲	۸۵/۰	۹۰/۰	۸۳/۹	۸۵/۸	۷۶/۱	۵/۱	۵/۵
ک-نزدیک‌ترین همسایه با k=7	۹۰/۲	۸۶/۲	۹۰/۲	۸۶/۶	۸۷/۰	۷۹/۵	۶/۴	۶/۶
ک-نزدیک‌ترین همسایه با k=8	۹۰/۲	۸۴/۸	۹۰/۰	۸۳/۵	۸۵/۴	۷۵/۱	۴/۷	۴/۹
ک-نزدیک‌ترین همسایه با k=9	۸۹/۹	۸۵/۵	۸۹/۹	۸۴/۷	۸۶/۲	۷۷/۹	۶/۲	۶/۴
ک-نزدیک‌ترین همسایه با k=10	۸۹/۳	۸۳/۸	۸۸/۹	۸۲/۱	۸۴/۰	۷۲/۶	۵/۱	۴/۵
ماشین بردار پشتیبان	**۹۶/۴**	**۹۱/۶**	**۹۶/۵**	**۹۱/۶**	**۹۷/۰**	**۸۷/۱**	۴/۳	**۳/۶**
کلاسه بندهای گروهی	۹۴/۸	۹۰/۹	۹۵/۰	۹۰/۱	۹۴/۶	۸۹/۱	۴/۹	۷/۳
جنگل تصادفی	۹۴/۲	۹۱/۱	۹۴/۳	۹۰/۲	۹۳/۲	۸۴/۰	۴/۷	۴/۹

جدول ۱۴: نتایج روش پیشنهادی روی مجموعه داده چهارم (فایل‌های بسته‌بندی شده)

معیار / کلاسه بند	دقت		معیار F		نرخ مثبت صحیح		نرخ مثبت کاذب	
	روش پیشنهادی	کل آپکدها	روش پیشنهادی	کل آپکدها	روش پیشنهادی	کل آپکدها	روش پیشنهادی	کل آپکدها
ک-نزدیک‌ترین همسایه با k=1	۷۵/۰	۷۴/۶	۶۸/۳	۶۷/۸	۵۳/۸	۵۳/۴	۳/۸	۴/۲
ک-نزدیک‌ترین همسایه با k=2	۷۳/۳	۷۳/۹	۶۵/۱	۶۵/۹	۴۹/۸	۵۰/۴	۶/۶	۲/۶
ک-نزدیک‌ترین همسایه با k=3	۷۴/۹	۷۴/۵	۶۸/۴	۶۷/۴	۵۴/۲	۵۲/۸	۴/۴	۳/۸
ک-نزدیک‌ترین همسایه با k=4	۷۳/۸	۷۳/۷	۶۶/۳	۶۶/۱	۵۱/۶	۵۱/۲	۴/۰	۳/۸
ک-نزدیک‌ترین همسایه با k=5	۷۴/۷	۷۴/۶	۶۸/۰	۶۷/۸	۵۳/۸	۵۳/۴	۴/۴	۴/۲
ک-نزدیک‌ترین همسایه با k=6	۷۳/۴	۷۳/۹	۶۵/۸	۶۶/۶	۵۱/۲۰	۵۲/۰	۴/۴	۴/۲
ک-نزدیک‌ترین همسایه با k=7	۷۴/۲	۷۴/۲	۶۷/۲	۶۷/۲	۵۲/۸	۵۲/۸	۴/۴	۴/۴
ک-نزدیک‌ترین همسایه با k=8	۷۳/۱	۷۳/۶	۶۵/۳	۶۶/۲	۵۰/۶	۵۱/۶	۴/۴	۴/۴
ک-نزدیک‌ترین همسایه با k=9	۷۳/۷	۷۴/۰	۶۶/۴	۶۷/۰	۵۲/۰	۵۲/۸	۴/۶	۴/۸
ک-نزدیک‌ترین همسایه با k=10	۷۳/۲	۷۳/۳	۶۵/۵	۶۵/۷	۵۰/۸	۵۱/۲	۴/۴	۴/۸
ماشین بردار پشتیبان	**۷۶/۹**	**۷۶/۱**	**۷۲/۲**	**۷۲/۷**	**۶۱/۲**	**۵۶/۳**	۷/۴	**۴/۰**
کلاسه بندهای گروهی	۷۶/۳	۷۴/۸	۷۰/۳	۶۸/۶	۵۶/۰	۵۵/۰	**۳/۴**	۵/۴
جنگل تصادفی	۷۵/۹	۷۴/۰	۶۹/۸	۷۲/۵	۵۵/۶	۵۶/۰	۳/۸	۵/۲

۵-۳- نتیجه‌گیری

این تحقیق به دنبال روشی کارآمد، با نرخ تشخیص بالا برای شناسایی بدافزارهای ناشناخته براساس آپکدهای پراهمیت بوده‌است. برای تحقق این هدف فایل‌های موردنظر در سطح اسمبلی تجزیه شده‌اند و آپکدهای سازنده آن‌ها به‌دست آمده‌است. از مجموعه دنباله آپکدهای به‌دست‌آمده، فرکانس تکرار ترکیبات دوتایی از آپکدهای پراهمیت با سایر آپکدها در نظر گرفته شد و این فرکانس تکرارها نرمال‌سازی شدند و در قالب یک عکس نمایش داده شدند. درنهایت از تصاویر به‌دست‌آمده ویژگی استخراج گردید و تلاش شد که روش پیشنهادی، با روش‌های پیشین عادلانه مقایسه شود. نتایج به‌دست‌آمده نشان می‌دهد میزان دقت روش پیشنهادی در مقایسه با روش‌های پیشین بیشتر بوده‌است و در زمان و حافظه می‌تواند صرفه‌جویی کند.

برای بهبود میزان دقت روش پیشنهادی می‌توان از راه‌کارهای مختلفی استفاده کرد. برای نمونه آپکدها پراهمیت به‌دست‌آمده در فضاهای دیگری نگاشت شوند و یا از الگوریتم‌های بهبودیافته‌ی دیگری برای استخراج ویژگی استفاده شود.

مراجع

[1] S. Mansfield-Devine, "Security guarantees: building credibility for security vendors", Network Security, vol. 2016, no. 2, pp. 14-18, 2016.

[2] G. McGraw, G. Morrisett, "Attacking malicious code: A report to the infosec research council ", IEEE Software, pp 33–44, 2000.

[3] J. Jang, H. Kang, J. Woo, A. Mohaisen, H. KangKim, "Andro-Dumpsys: Anti-malware system based on the similarity of malware creator and malware centric information", Computers & Security, vol. 58, pp. 125-138, 2016.

[4] Z. Bazrafshan, H. Hashemi, S. Fard, A. Hamzeh, "A survey on heuristic malware detection techniques", 5th Conference on Information and Knowledge Technology (IKT), pp. 113–120, shiraz, 2013.

[5] H. Hashemi, A. Azmoodeh, A. Hamzeh and S. Hashemi "Graph embedding as a new approach for unknown malware detection", Computer Virology and Hacking Techniques, vol. 12, no. 4, pp. 101-141, 2016.

[6] P. Vinod, R. Jaipur, V. Laxmi, M. Gaur, "Survey on malware detection methods", Proceedings of the 3rd Hackers' Workshop on computer and internet security (IITKHACK'09), pp. 74-79, 2009.

[7] M. Xu et aI., "A similarity metric method of obfuscated malware using fimction-call graph", Journal in Computer Virology, vol. 9, no. 1, pp. 35- 47, 2013.

[8] I. Santos, F. Brezo, X. Ugarte-Pedrero, and P. G. P. Bringas, "Opcode sequences as representation of executables for data-mining-based unknown malware detection", information Sciences, vol. 231, pp. 64–82, 2013.

[9] Y. Yanfang, L. Tao, J. Qingshan, W. Youyu, "CIMDS: Adapting Postprocessing Techniques of Associative Classification for Malware Detection", IEEE Transactions on Systems, Man, and

[10] A. Shabtai, R. Moskovitch, Y. Elovici, C. Glezer, "Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey", Information Security Technical Report, vol. 14, no. 1, pp. 16-29, 2009.

[11] B. Kinholkar, "Study of Dataset Feature Filtering of OpCode for Malware Detection Using SVM Training Phase", International Journal of Science and Research (IJSR), vol. 4, no. 12, 2015.

[12] S. Cesare, Y. Xiang, W. Zhou, "Control flow-based malware variant detection", IEEE Transactions on Dependable and Secure Computing, pp. 307–317, 2014.

[13] Y. Ding, W. Dai, Sh. Yan, Y. Zhang, "Control flow-based opcode behavior analysis for Malware detection", Computer & Security, vol. 44, pp. 65-74, 2014.

[14] T. Wüchner, M. Ochoa, A. Pretschner, "Malware detection with quantitative data flow graphs", 9th ACM symposium on Information, computer and communications security, pp. 271–282, 2014.

[15] T. Abou-assaleh, N. Cercone, V. Keselj, R. Sweidan, "N-grambased detection of new malicious code", Computer Software and Applications Conference, Proceedings of the 28th Annual International, vol. 2, no. 1, pp. 41–42, 2004.

[16] G. Canfora, A. Lorenzo, "Effectiveness of opcode ngrams for detection of multi family android malware", 10th International Conference on Availability, Reliability and Security (ARES), pp. 333–340. IEEE, 2015.

[17] B. Kang, S. Y. Yerima, K. Mclaughlin and S. Sezer, "N-opcode analysis for android malware classification and categorization", Cyber Security and Protection of Digital Services (Cyber Security), London, 2016.

[18] D. Uppal, R. Sinha, "Exploring behavioral aspects of API calls for malware identification and categorization", In 2014 International Conference on Computational Intelligence and Communication Networks, pp. 824-828, 2014.

[19] Y. Qiu, "The openness of Open Application Programming Interfaces", information, Communication & Society, pp. 1-17, 2016.

[20] M. Belaoued, S. Mazouzi, "An MCA Based Method for API Association Extraction for PE Malware Categorization", International Journal of Information and Electronics Engineering, vol. 5, no. 3, pp. 225-231, 2015.

[21] D. Bilar, "Opcodes as predictor for malware," International Journal of Electronic Security and Digital Forensics, vol. 1, no. 2, pp. 156–168, 2007.

[22] I. Santos, B. Sanz, C. Laorden, F. Brezo, P.G. Bringas, "Opcodesequence- based semi-supervised unknown malware detection", Computational Intelligence in Security for Information Systems, pp. 50–57, Berlin, 2011.

[23] F. Manavi, A. Hamzeh. "A new approach for malware detection based on evolutionary algorithm." In Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 1619-1624. ACM, 2019.

[24] F. Manavi, A. Hamzeh, "A new method for malware detection using opcode visualization", 2017 Artificial Intelligence and Signal Processing Conference (AISP), pp. 96-102, Shiraz, 2017.

[25] M. Farrokhmanesh, A. Hamzeh, "A novel method for malware detection using audio signal processing techniques", Artificial Intelligence and Robotics (IRANOPEN), pp. 85 – 91, 2016.

[26] L. Nataraj, S. Karthikeyan, G. Jacob, B. S. Manjunath, "Malware images: Visualization and automatic classification," In Proc. 8th Int. Symp. Visualization for Cyber Security, VizSec, ACM, pp. 41–47, 2011.

[27] A. Oliva, A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," Int. J. Comput. Vision, vol. 42, pp. 145–175, 2001.

[28] M. Douze, H. Jégou, H. Sandhawalia, L. Amsaleg, and C. Schmid, "Evaluation of gist descriptors for web-scale image search", The ACM International Conference on Image and Video Retrieval pp. 19, ACM, 2009.

[29] J. Hayes, A. Efros, "Scene completion using millions of photographs", ACM Transactions on Graphics, 2007.

[30] X. Li, C. Wu, C. Zach, S. Lazebnik, J.-M. Frahm, "Modeling and recognition of landmark image collections using iconic scene graphs", In ECCV, 2008.

[31] F. Lin, W.W. Cohen, "Power iteration clustering", 27th International Conferenceon Machine Learning(ICML10), pp. 655–662, 2010.

[32] M. Eskandari, Z. Khorshidpour, S. Hashemi, "HDM-analyser: a hybrid analysis approach based on data mining techniques for malware detection", Journal of Computer Virology and Hacking Techniques, vol. 9, pp. 77–93, 2013.

زیرنویس‌ها

[1] Malware Detection
[2] Windows Application Programming Interface (API) Call
[3] Bayes
[4] Naïve Bayes
[5] Decision tree
[6] Support Vector Machine (SVM)
[7] Information gain
[8] Document frequency
[9] Power iteration
[10] synthesizer
[11] Pack
[12] Over fitting
[13] Random Forest