مجله علمی پژوهشی «علوم و فناوری‌های پدافند نوین»
سال پنجم، شماره ۳، پاییز ۱۳۹۳؛ ص ۲۰۱-۱۸۷

روش تشخیص بدافزار مبتنی بر تحلیل ایستای ساختار PE

دانیال جواهری[1*]، سعید پارسا[2]

۱- مربی دانشکده تحصیلات تکمیلی، گروه رایانه، دانشگاه آزاد اسلامی واحد بروجرد

۲ – دانشیار دانشکده مهندسی رایانه، دانشگاه علم و صنعت ایران

(دریافت: ۹۳/۰۳/۲۹، پذیرش: ۹۳/۱۰/۰۶)

چکیده

این مقاله ضمن معرفی و مقایسه روش‌های تشخیص بدافزار و خانواده‌های مختلف بدافزارها، یک روش جدید و کارا جهت تشخیص بدافزارها با استفاده از تحلیل ایستا ارائه می‌کند. این تحلیل مبتنی بر بررسی ساختار فایل‌های اجرایی PE است. روش پیشنهادی با بررسی و مطالعه دقیق سرآیند بدافزارها و فایل‌ها بی‌خطر، خواصی از ساختار فایل‌های اجرایی مانند تعداد، اندازه و نام قسمت‌ها، نام توابع و کتابخانه‌های موجود در جداول IAT و EAT، نقطه شروع و میزان آنتروپی را برای تشخیص و تفکیک بدافزارها و فایل‌های بی‌خطر پیشنهاد می‌کند. خواص مذکور با انتصاب امتیازات مثبت و منفی میزان بدخیم یا خوش‌خیم بودن یک فایل ناشناس را بر اساس فرمول‌های روش پیشنهادی تعیین می‌کنند. با انجام داده کاوی در حجم انبوهی شامل ۱۵۰۰۰ نمونه بدافزار و ۱۳۵۰۰ فایل بی‌خطر خواص پیشنهاد شده استخراج و با استفاده از تکنیک‌های یادگیری ماشین مدلی هوشمند برای تشخیص و خوشه‌بندی بدافزار مبتنی بر تولید قانون آموزش داده شده است. روش پیشنهادی این مقاله بدافزارها را در ۵ خانواده و فایل‌های بی‌خطر را در ۲ خانواده خوشه‌بندی می‌کند. این مقاله در پایان دقت روش پیشنهادی را در تشخیص و خوشه‌بندی بدافزارها و فایل‌های بی‌خطر ارزیابی کرده و نشان می‌دهد که روش پیشنهادی می‌تواند با دقت بیش از ۹۵ بدافزارها را تشخیص داده و خوشه‌بندی نماید و از این حیث با روش‌های پیشین مقایسه شده و در جایگاه دوم قرار می‌گیرد.

کلید واژه‌ها: تشخیص بدافزار، ساختار PE، تحلیل ایستا، سرآیند بدافزار، داده‌کاوی، تحلیل رفتاری، یادگیری ماشین.

A Malware Detection Method Based on Static Analysis of a Portable Executable Structure

D. Javaheri[*], S. Parsa

College of Post-Graduate Education, Islamic Azad University, Borujerd Branch

(Received: 19/06/2014; Accepted: 27/12/2014)

Abstract

Herein,the study and comparisonof malware families and malware detection methods have been performed and a new and efficient method for malware detection by static analysis is proposed based on PE Structure of executable files. Our method present some new feathers such as quantity, name and size of sections, name of system calls and their libraries in IAT and EAT table, entry point and entropy for detection and distinguishes malwares and benign files by observing and exploring PE structure and header of mentioned files very deeply. These feathers can assign positive and negative point to determine malignant or benign rate of an unknown executable file by formulas of proposed method. These features were extracted by doing data-mining on a large scale consist near 15000 malwares and 13500 benign files and uses machine learning techniques for train and learn an intelligent rule base model for malware detection. Proposed method cluster malwares in 5 and benign files in 2 families. The accuracy of proposed method in detection and clustering malware and benign files have been evaluated, indicating that the proposed method can detect and cluster malwares by more than 95 percent in accuracy and compete with other methods and get second ranked.

Keywords: Malware Detection, PE Structure, Static Analysis, Virus Header, Data Mining, Behavioral Analysis, Machine Learning.

[*] Corresponding Author E-mail: d.javaheri@iaub.ac.ir

۱. مقدمه

گسترش بدافزارها در چند سال اخیر به صورت شگفت‌انگیزی افـزایش یافته است بنا به گزارش شرکت Symantec نـرخ تولیـد بـدافزارهـا در سال ۲۰۰۸ در جهان از نرخ تولید برنامه‌های بی‌خطر بیشتر شده است و این جای بسی تأمل و تأسف دارد. علت رشد چشمگیر تولید بـدافزار در انگیزه برنامه نویسان بـدخواه، سـهولت تولیـد بـا بـه وجـود آمـدن ابزارهای تولید کد و موتورهای دگردیسی است. بنـابر گـزاش شـرکت McAfee از مرجع [۱] روزانه بیش از ۱۰۰.۰۰۰ بدافزار جدیـد تولیـد می‌شود یعنی ۶۹ بدافزار در هر دقیقه، بنابراین می‌توان ادعا کـرد کـه فرکانس تولید بدافزار امروزه تقریباً ۱ بدافزار بر ثانیه است. نکته بسیار مهم در این است که این بدافزارها هیچ شباهتی به بدافزارهای سـنتی گذشته که بـا تهیـه یـک امضـاء از آن قابـل شناسـایی باشـد ندارنـد. بدافزارهای امروزی با انواع موتـورهـای مـبهم‌سـازی، چنـد ریختـی و دگردیسی مانند VLC32 و Toward مجهز شده‌اند که شناسایی آن‌هـا توسط روش‌های مبتنی بر امضاء در عمل غیر ممکن اسـت بـه همـین دلیل روش‌های نوین تشخیص بدافزار که عمدتاً مبتنی بر تحلیل بودند ظهور کرده و شرکت‌های امنیتی نیز اقدام به تولید ابزارهای تحلیل‌گـر برای تحلیل بدافزارها با روش فوق نمودند. هم‌زمـان بـا پیشـرفت ایـن روش‌های ضدتحلیل نیز توسعه یافته و بدافزارهای نوین نیـز بیشـتر از این روش‌های برای فریب برنامه‌های امنیتی و تحلیل‌گر سود می‌برنـد. قطعاً انگیـزه توسـعه دهنـدگان روش‌هـای ضـد تحلیـل، محافظـت از بدافزارها نبوده بلکـه عمـدتاً جهـت جلـوگیری از مهندسـی معکـوس برنامه‌های مهم و حساس و یا تکثیر غیـر قـانونی بـا ایجـاد شـکاف در مجوز استفاده از نرم‌افزارها بوده است ولی بدافزارها نیز از این روش‌هـا در راستای نیل به اهداف بد خواهانه خود نیز سود می‌برند. علت دیگـر عدم کارایی روش‌های مبتنی بر امضاء حجم انبوه تولید بـدافزار اسـت که امکان تهیه امضـاء از همـه ایـن بـدافزارها و ذخیـره در پایگـاه‌داده برنامه ضدبدافزار را با چالش‌هایی مانند انفجـار بانـک اطلاعـاتی، افـت سرعت با افزایش حجم بانک اطلاعاتی و تأخیر در اعمال به‌روزرسـانی‌هـای جدید مواجهه می‌سازد بنابراین تلاش‌ها به سمت تشخیص هوشمندانه بدافزار و فارغ از امضاء تمایل پیدا کرد. در این مقالـه سـعی مـی‌شـود ضمن معرفی و مقایسه روش‌های تشخیص بدافزار و ذکر محدودیت‌هـای آن‌هـا، روشـی نـوین و کـارا بـرای تشـخیص هوشـمندانه بـدافزارها و خوشه‌بندی آن‌ها بر اساس تحلیل ایستای ساختار فایـل‌هـای اجرایـی PE ارائه شود.

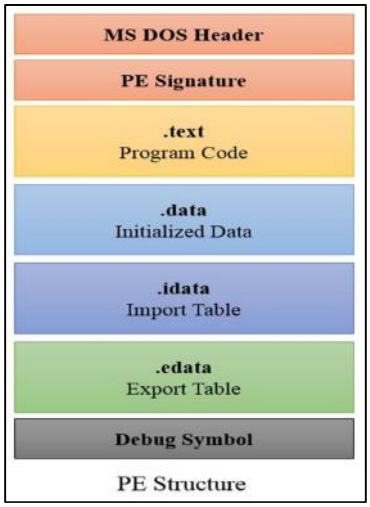
این مقاله در پنج قسمت سازمان‌دهـی شـده اسـت در قسـمت دوم روش‌های تشخیص بدافزار معرفی و مقایسه می‌شوند، در قسمت سـوم روش پیشنهادی این مقاله در تحلیل ایستا بیان مـی‌شـود، در قسـمت چهارم نحوه تشخیص و خوشه‌بندی بدافزارها و فایل‌های بی‌خطر نشان داده می‌شود و در قسمت پایـانی روش پیشـنهادی از حیـث دقـت در تشخیص و کیفیت خوشه‌بندی مـورد آزمـون قـرار گرفتـه و بـا دیگـر روش‌های موجود مقایسه می‌شود.

۲. معرفی روش‌های تشخیص بدافزار

در این قسمت روش‌های تشخیص و تحلیل بدافزارها را معرفی نموده و مزایا معایب هرکدام را بیان می‌کنیم و در پایان این بخـش مقایسـه‌ای اجمالی بین این روش‌ها خواهیم داشت. هدف از این بخـش مشـخص نمودن جایگاه روش پیشنهادی و مزایا و محدودیت‌های آن است.

۲-۱. ساختار فایل‌های اجرایی قابل حمل PE

قبل از توضیح روش‌های تشخیص بـدافزار نیـاز اسـت تـا در خصـوص ساختار فایـل‌هـای اجرایـی PE توضـیح داده شـود. PE بـر گرفتـه از Portable Executable اسـت. ایـن سـاختار، سـاختار اسـتاندارد مایکروسافت جهت اجرای فایل‌های اجرایـی از اولـین نسـخه سیسـتم عامل ویندوز NT[1] می‌باشد [۲]. تمام فایل‌های که بخواهند بر روی این سکو[2] یعنی ویندوز NT اجرا شـوند. مجبـور بـه رعایـت ایـن سـاختار هستند. در شکل زیر نمای کلی ساختار PE آورده شده است.



شکل ۱. ساختار فایل اجرایی قابل حمل PE

همان‌گونه که در شکل (۱) نشان داده شده اسـت ایـن سـاختار بـه قسمت‌هایی تحت عنوان قسمت یا Section تقسیم شده اسـت اولـین قسمت[3] MS DOS Header می‌باشد این قسمت از زمان سیستم عامل DOS باقی مانده است و سرآیند فایل‌های اجرایی تحت ایـن سیسـتم عامل را شامل می‌شود. این قسمت جهـت اجـرای برنامـه‌هـای تحـت DOS در نسخه‌های اولیه ویندوز NT قرار داده شده است. قسمت دوم این ساختار قسمت CODE است که به صورت اختصار text. یـا code. نشان داده می‌شود در این قسمت تمام کـدهـای اجرایـی برنامـه قـرار می‌گیرند. قسمت سوم تا پنجم این سـاختار بـه ترتیـب قسـمت‌هـای data، نگاه دارنده داده‌های برنامه idata، نگاه دارنـده داده‌هـای ورودی برنامه و edata که حاوی داده‌هـای خروجـی برنامـه اسـت مـی‌باشـد.

[1] New Technology
[2] Platform
[3] Section

قسمت idata یکی از مهم‌ترین نقاط برای تشخیص بدافزارها است که در روش پیشنهادی نیز از آن استفاده شده است در این قسمت جدولی به نام IAT[1] یا جدول آدرس ورودی قرار داد. این جدول در بردارنده نام و آدرس توابعی سیستمی است که برنامه مورد نظر جهت اجرا به آن ها نیاز دارد این جدول در تشخیص بدافزار بسیار کاربر دارد در قسمت edata نیز جدول با نام EAT[2] یا جدول آدرس خروجی وجود دارد. در این جدول نام و آدرس توابعی نگهداری می‌شود که توسط برنامه مورد نظر جهت استفاده توسط دیگر برنامه‌ها ارایه می‌شوند. در انتها این ساختار اطلاعات مربوط به اشکال‌زدایی برنامه شامل نمادهای اشکالزدایی نگهداری می‌شود. عمده فرمت‌های فایل‌هایی با ساختار PE شامل EXE ، SYS و DLL است [۳].

۲-۲. تشخیص مبتنی بر امضاء

سنتی‌ترین و پایه‌ای‌ترین روش تشخیص بدافزار است در این روش پس از تشخیص یک بدافزار امضایی از آن در پایگاه داده قرار داده می‌شود تا هنگام مواجه بعدی با بدافزار مذکور بتوان قبل از اجراء آن را تشخیص داد. این روش به صورت گسترده‌ای توسط برنامه‌های ضدبدافزار از دیر باز استفاده می‌شود [۴]. البته امروزه این روش به تنهایی کارایی لازم برای تشخیص بدافزارها را ندارد زیرا بدافزارهای امروزی به انواع روش‌های مبهم‌سازی و دگردیسی مسلح شده‌اند و می‌توانند به راحتی با ایجاد تغییرات در کد اجرایی خود، امضایشان را تغییر دهند تا از شناسایی مجدد توسط این روش‌ها در امان بمانند. مشکل دیگر این روش وابستگی دائمی برنامه ضدبدافزار به به‌روزرسانی پایگاه داده حاوی امضاءهای بدافزار است چون کشف این بدافزارها عمدتاً در آزمایشگاه‌های تحلیل بدافزار با استفاده از ابزارهای تحلیل‌گر نظیر ماشین‌های مجازی[3] و جعبه شن‌ها[4] صورت می‌پذیرد و تنها امضاء بدافزار کشف شده در اختیار برنامه سمت کاربر قرار می‌گیرد.

نکته بسیار مهم در این روش محل تهیه امضاء و الگوریتم تهیه هَش می‌باشد که تأثیر بسیار زیادی دقت و سرعت این روش دارد. از الگوریتم‌های SHA-1 و MD5 به وفور برای تهیه امضاء از بدنه بدافزاها استفاده می‌شود. محل استخراج امضاء کاملاً به روش تشخیص بدافزار وابسته است در گذشته اغلب از کل فایل امضاء تهیه می‌شد ولی ناکارآمدی این روش با پیچیده شدن بدافزارها نمایان و برنامه‌های ضدبدافزار ناچار به تهیه امضاء از بخش‌های مختلف از فایل اجرای از قبیل Code Segment و گاهاً گرفتن چند امضاء از یک فایل شدند تا بتوانند با بدافزارهای جدید به ویژه بدافزارهای خودتغییر[5] مقابله نمایند.

روش‌های مبتنی بر امضاء بیشترین سرعت را در تشخیص نسبت به روش‌های دیگر دارند و به همین دلیل است که امروزه همچنان در اولین لایه تشخیص توسط برنامه‌های ضدبدافزار استفاده می‌شوند. امروزه همه کار تحلیل در آزمایشگاهی جداگانه صورت نمی‌گیرد زیرا امکان تشخیص و مقابله به موقع از برنامه‌های ضدبدافزار در مواجه با بدافزارهای جدید سلب می‌شود بنابراین امکاناتی نظیر جعبه شن‌ها در برنامه‌های ضد بدافزار قرار داده شد تا این برنامه‌ها بتوانند خود عملیات تحلیل را انجام دهند [۵]. این روش‌های تحلیل را می‌توان در دو دسته تحلیل ایستا و پویا تقسیم‌بندی کرد.

۲-۳. تشخیص بر اساس تحلیل ایستا

خصیصه اصلی این روش‌ها در تشخیص بدافزارها فارغ از اجرای آن‌ها است یعنی تمام این روش‌ها سعی در تشخیص بدافزارها بدون اجرا کردن و فقط با بررسی کد باینری و خواص آن‌ها را دارند. روش‌های زیادی در این زمینه توسعه داده شده‌اند [۴]. عمده این روش‌ها با بررسی ساختار فایل‌های اجرایی و استخراج خواص آن‌ها مانند جستجوی دنباله‌هایی از رشته‌های قابل چاپ، سعی در تشخیص بدافزارها و نوع آن‌ها دارند. استفاده از روش‌های داده‌کاوی و یادگیری ماشین جزء جداناشدنی تمام این روش‌ها است. هدف از داده‌کاوی در این روش‌ها خوشه‌بندی[6] بدافزارها بر اساس خواص آن‌ها می‌باشد تا با استفاده از روش‌های یادگیری ماشین بتوان ماهیت یک فایل ناشناس را تشخیص و بر اساس میزان نزدیکی و شباهت خواص، آن را در یکی از خوشه‌ها قرار داد و ماهیت آن فایل ناشناخته را مشخص کرد.

روش تحلیل رفتار ایستا پیشنهادی که در این دسته از روش‌های تشخیص بدافزار قرار می‌گیرد پس از تصدیق معتبر بودن ساختار PE برای یک فایل ناشناس و اطمینان از پک[7] نبودن آن اقدام به استخراج خواصی که در ادامه ذکر می‌شود، می‌نماید آنگاه براساس فرمول و قوانینی که در بخش بعدی ذکر می‌شود در خصوص بدافزار یا بی‌خطر بودن آن فایل و خوشه‌ای[8] که فایل در آن قرار می‌گیرد، تصمیم‌گیری می‌نماید.

روش‌های دیگر با دیس‌اسمبل[9] کردن کد اجرایی یک برنامه و تشکیل گراف جریان کنترلی[10] سعی در تشخیص و دسته‌بندی بدافزارها دارند. این روش الگوهایی برای گراف جریان کنترلی بدافزارها در نظر گرفته آنگاه بر اساس میزان نزدیکی گراف جریان کنترلی برنامه تحت تحلیل به آن‌ها در خصوص ماهیت برنامه تحت تحلیل تصمیم‌گیری می‌نمایند. مشکل اصلی این روش در نیازمند بودن ابزار ضدبدافزار استفاده کننده به یک دیس اسمبلر درونی و مشکلات مطرح در حوزه دیس اسمبلی و مهندسی معکوس کد باینری خواهد

[1] Import Address Table
[2] Export Address Table
[3] Virtual Machine
[4] Sandbox
[5] Metamorphic

[6] Clustering
[7] Pack
[8] Cluster
[9] Disassemble
[10] Control Flow Graph

بود [۵] زیرا عمده بدافزارها از روش‌هایی نظیر مبهم‌سازی گراف جریان کنترلی و رمز نگاری رشته‌ها برای جلوگیری از مهندسی معکوس استفاده می‌کنند.

۲-۴. تشخیص بر اساس تحلیل پویا

این روش‌ها برای پوشش کاستی‌ها و محدودیت‌های موجود در روش‌های تشخیص مبتنی بر تحلیل ایستا ارائه شدند. تمامی روش‌های این حوزه سعی در تشخیص بدافزار با اجرای فایل مشکوک به بدافزار دارند.

از جمله مشکلات موجود در روش‌های مبتنی بر تحلیل ایستا محدودیت در مواجه با تکنیک‌های نوین مبهم‌سازی است به عنوان مثال برخی از بدافزارهای نوین با انجام برنامه نویسی پویا[1] مبادرت به مخفی‌سازی فراخوانی‌های سیستمی خود می‌کنند و با این حربه روش‌های تحلیل ایستا که از بررسی فراخوانی‌های سیستمی یک فایل اجرایی سعی در تشخیص آن دارند را فریب دهند.

بسیاری از بدافزارها قبل از انتشار توسط برنامه بسته‌بندی کننده و محافظت کننده؛ بسته‌بندی و محافظت می‌شوند که این امر تشخیص آن‌ها توسط روش‌های تحلیل ایستا را بسیار سخت می‌نماید زیرا با بسته‌بندی شدن ساختار یک فایل اجرایی امکان استخراج خواص مورد نیاز از آن بسیار سخت و در پاره‌ای از موارد غیر ممکن می‌شود [۵].

۲-۵. مقایسه روش‌های تشخیص و تحلیل بدافزار

استفاده از روش‌های مبتنی بر امضاء اگر چه امروزه کارایی چندانی به ویژه در مواجه با بدافزارهای مسلح ندارند ولی به دلیل سرعت بسیار بالا در تشخیص همچنان در صف اول برای تشخیص توسط اکثر برنامه‌های ضد بدافزار مطرح استفاده می‌شوند. پیاده‌سازی این روش نسبت سایر روش‌ها ساده‌تر است ولی محدودیت‌های از جمله وابستگی دائمی برای به‌روزرسانی پایگاه داده امضاءها نقش آن‌ها را کم‌رنگ خواهد کرد. روش‌های تشخیص مبتنی بر تحلیل رفتاری اگر چه در مقابل بدافزارهای مسلح هوشمندانه‌تر عمل می‌کنند و لی پیاده‌سازی آن‌ها سخت و پیچیده است و سرعت کمتری دارند. اکثر برنامه‌های ضد بدافزار نوین و مشهور علاوه بر روش مبتنی بر امضاء از روش‌های تحلیل ایستا بیشتر از روش‌های تحلیل پویا استفاده می‌کنند علت آن هم در سرعت بیشتر تحلیل‌های رفتاری ایستا و مشکلات طراحی و پیاده‌سازی محیط‌های امن نظیر جعبه شن و ماشین مجازی برای تحلیل‌های پویا است. بر خلاف تحلیل رفتاری ایستا که رفتار برنامه فارغ از اجرای آن و در زمان ذخیره بر روی دیسک سخت استخراج و تحلیل می‌شود، در تحلیل‌های پویا بدافزار به صورت واقعی اجرا شده و با نظارت بر اجرای آن در مدت زمانی مشخص اطلاعات لازم برای تصمیم‌گیری در خصوص ماهیت فایل به دست می‌آید. این کار به ویژه بر روی کامپیوتر کاربر بسیار خطرناک است. برای انجام این مهم نیاز به محیط‌های امن تحلیل‌گر مانند محیط‌های ماشین مجازی و جعبه شن وجود دارد که در مرجع [۵] یک محیط تحلیل‌گر امن و هوشمند پیشنهاد شده و برای ساخت مدل رفتاری پیشنهادی این مقاله از آن استفاده شده است در ادامه و در شکل زیر مقایسه بین روش‌های تشخیص بدافزار نشان داده شده است.

شکل ۲. مقایسه روش‌های تشخیص بدافزار

۳. پیشینه تحقیق

در این قسمت از مقاله اقدامات مرتبط با چارچوب کلی این پژوهش یعنی تشخیص بدافزارها با استفاده از تحلیل رفتار آن‌ها به صورت ایستا بیان شده است.

روش ارائه شده توسط اسچالتز و همکاران [۶] مبتنی بر استفاده از داده‌کاوی برای تشخیص بدافزارها است که در آن خواصی از ساختار ساختار PE از جمله نام DLL های مورد استفاده برنامه، نام و تعداد توابعی که از هر Dll فراخوانی شده و جستجو رشته‌های متنی از فایل استفاده شده است. در این روش از الگوریتم Ripper برای تولید قانون، Naïve Byes برای آموزش مدل و ngram برای جستجوی رشته‌ها استفاده شده است. تعداد نمونه موجودیت‌های شرکت کننده در داده‌کاوی این روش در مجموع ۴۲۶۶ فایل شامل ۳۲۶۴ بدافزار و ۱۰۰۱ فایل بی‌خطر بوده است. دقت این روش ۹۷/۱۱ ٪ ارزیابی شده است.

روش مذکور اگر چه از دقت بالایی در تشخیص برخوردار است ولی نرخ مثبت کاذب بسیار بدی دارد و از این رو درصد قابل توجهی از فایل‌های بی‌خطر را نیز به عنوان بدافزار معرفی می‌کند.

روش ارائه شده توسط کالتر و ملوف [۷] در واقع بهبود داده شده روش قبلی است که در آن از الگوریتم‌های Naïve Bayes، SVM و Decision Tree برای یادگیری با سرپرست مدل استفاده شده و به‌جای ngram از یک روش دیگر برای جستجو رشته‌ها استفاده شده است. دقت این روش بالای ۹۰ درصد ارزیابی شده است.

این روش اگر چه مشکل نرخ مثبت کاذب روش قبل را حدودی بهبود داده است ولی دقت به ۹۰ دصد کاهش پیدا کرده است و میزان دقت اصلاً مطلوب نیست.

[1] Dynamic Programing

روش ارائه شده توسط سیدیکی و همکاران [۸] از دنباله‌ای دستورات با طول متغیر برای تشخیص خانواده Worm استفاده کرده است. ارائه دهنده از الگوریتم‌هایDT و Random Forest Machine Learning برای آموزش مدل رفتاری استفاده کرده است. نمونه موجودیت‌های استفاده در داده‌کاوی ۲۷۷۴ عدد شامل ۱۴۴۴ بدافزار خانواده Worm و ۱۳۳۰ فایل بی‌خطر است. این روش فقط قادر به تشخیص خانواده Worm است و از این حیث با روش‌های پیشنهادی این مقاله قابل مقایسه نیست.

روش ارائه شده توسط دی گائو و همکاران [۹]مبتنی بر تحلیل ایستا است که بر اساس خواص ساختار PE، بدافزارها را تشخیص می‌دهد. این روش خوشه‌بندی را توسط نسخه‌ای بهینه شده از الگوریتم KNN انجام می‌دهد. از ۴۱۰ فایل شامل ۳۰۰ بدافزار و ۱۱۰ فایل بی‌خطر تشکیل شده است که از این تعداد ۲۸۰ نمونه برای آموزش و ۱۳۰ نمونه برای آزمون مدل استفاده شده است ۳ خوشه بدافزار شامل Trojan ، Backdoor و Worm و یک دسته فایل بی‌خطر است. دقت روش به صورت میانگین ۸۹/۹۷ درصد در تشخیص خانواده‌های فوق است.

روش مذکور برای بررسی ساختار فایل اجرایی محدود به ساختار PE32 است و نمی‌توانند ساختارهای PE32+ یا PE64 را تحلیل نماید. می‌دانیم که بدافزارهای نوین در حال طراحی و دگردیسی با ساختارهای PE32+ برای اجرای سریع‌تر در سیستم عامل‌های ۶۴ بیتی هستند. تعداد نمونه موجودیت‌های شرکت داده شده در داده‌کاوی به ویژه تعداد انتخاب شده برای آموزش مدل به صورت مضحکانه‌ای کم است! روش مذکور فقط توانایی تشخیص ۳ خانواده Trojan، Backdoor و Worm را با میزان دقت کمتر از ۹۰ درصد یعنی ۸۹/۹۷ ٪ داراست!

در روش ارائه شده توسط علیرضایی و فروزنده [۱۰]، بر اساس نام کتابخانه‌های بارگذاری شده توسط یک برنامه، بدخواه یا بی‌خطر بون آن بررسی می‌شود. در روش مذکور شناسایی بدافزار از طریق کتابخانه‌های پیوند پویا که در زمان اجراء توسط بدافزار بارگذاری می‌شود صورت می‌پذیرد. در این روش هیچ تمایزی بین توابع سیستمی موجود در فایل‌های کتابخانه‌ای پیوند پویا وجود ندارد و همچنین تنها از بدافزارها جهت مدل‌سازی رفتار مخرب استفاده شده است. در این روش با به دست آوردن معادله‌ای، مشخص می‌شود که اگر نرم‌افزاری بر اساس تعداد دسترسی به کتابخانه‌های پیوند پویا مقادیری را در یک محدوده معین به دست آورد آنگاه این نرم‌افزار دارای یک رفتار فریب‌کارانه خواهد بود. در روش مذکور معادله تشخیص بر اساس مدل رگرسیون ساده برای خوشه‌بندی به دسته بدافزار و فایل بی‌خطر ساخته شده است. دقت روش مذکور در تشخیص بدافزار ۰/۷۷۵ است.

در روش مذکور بدافزارها همگی در یک خوشه قرار گرفته‌اند! که این علت بالا بود نرخ کاذب مثبت روش مذکور به صورت غیر قابل قبول یعنی ۰/۴۳ شده است. دقت روش مذکور برای مقایسه ضعیف است.

۴. روش پیشنهادی در تحلیل ایستا

با توجه به آنچه گفته شد حوزه عملکرد روش پیشنهادی مشخص گردید هدف از روش پیشنهادی این مقاله کشف و خوشه‌بندی بدافزارها با انجام تحلیل به صورت ایستا یعنی زمانی که فایل اجرایی بر روی دیسک سخت ذخیره شده است و فارغ از امضاء می‌باشد. علت این امر در گسترش و توسعه روز افزون بدافزارها است که در عمل امکان تهیه امضاء از حجم انبوه امروزی بدافزارها به ویژه بدافزارهای چندریخت که دائماً با ایجاد تغییر در بدنه خود امضاهایشان را تغییر می‌دهند غیر ممکن شده است. در ادامه به تشریح روش پیشنهادی می‌پردازیم.

در روش پیشنهادی این مقاله، خواصی از ساختار فایل‌های اجرایی PE استخراج شده است. این خواص در تعیین ماهیت فایل‌های اجرایی و پیشگویی نوع رفتار آن‌ها در زمان اجراء بسیار مؤثر هستند در ادامه این خواص که با مکاشفه[1] در داده‌کاوی از حجم انبوهی شامل ۱۵۰۰۰ بدافزار و ۱۴۰۰۰ برنامه بی‌خطر که از مراجع [۱۴-۱۱] جمع‌آوری شده، توضیح داده می‌شوند.

۴-۱. تعداد قسمت‌ها

بر اساس استاندارد PE تعداد قسمت‌ها به طور معمول ۵ عدد و حداکثر تعداد قسمت‌های متعارف ۸ عدد می‌باشد و تعداد قسمت کمتر و یا بیشتر از این بازه شک برانگیز است که بر اساس میزان اختلاف از این بازه امتیاز منفی به فایل تحت تحلیل داده می‌شود.

۴-۲. نام قسمت‌ها

تعداد قسمت‌ها به تنهایی نمی‌تواند عاملی کافی برای نامتعارف دانستن ساختار PE فایل تحت تحلیل باشد، بنابراین نام قسمت‌ها نیز بسیار مهم است و باید این در کنار مورد قبلی در ارزیابی ملاک قرار گیرد زیرا گاهی تعداد قسمت‌های یک بدافزار در بازه متعارف است ولی قسمت‌ها برای انجام وظایفی بد خواهانه ایجاد شده‌اند بنابراین در روش پیشنهادی وجود قسمت‌ها با نام‌هایی غیر متعارف به عنوان یک امتیاز منفی در ارزیابی قرار داده می‌شود نام‌های استاندارد بر اساس ساختار PE به شرح زیر است:

.text (.code), .data, .idata, .edata, .rdata, .rsrc, .reloc, .bss

۴-۳. اندازه قسمت‌ها

در این مورد اندازه قسمت‌ها مورد بررسی قرار می‌گیرد و در صورتی که اندازه قسمت‌ها نامتعارف باشد به عنوان یک امتیاز منفی در ارزیابی تلقی می‌شود. ملاک این اندازه غیر متعارف اختلاف در اندازه

[1] Heuristic

واقعی قسمت با اندازه نوشته شده در سرآیند ساختار PE است. حد این اختلاف برای قسمت code برابر با ۱ درصد اندازه کل و برای قسمت data ، ۴ درصد اندازه کل است. برای قسمت code به ازای هر ۳ درصد اختلاف یک وزن منفی و برای قسمت data به ازای هر ۶ درصد اختلاف یک وزن منفی اضافه می‌شود.

مقایسه دیگر در اندازه سایز قسمت‌های code و data با سایز کل فایل است به عنوان مثال یک فایل با حجم بالا که قسمت کد بسیار کوچکی داشته باشد بسیار مشکوک به بدخواهانه بودن است زیرا اغلب بدافزارهایی که قصد خود تغییری دارند بخشی از کدهای اجرایی خود را در قسمت‌هایی به غیر از .Cede ذخیره می‌کنند.

۴-۴. خواص قسمت‌ها

در این مورد خواص قسمت‌های یک فایل اجرایی به ویژه قسمت code مورد ارزیابی قرار می‌گیرد. در حالت پیش فرض قسمت کد به صورت فقط خواندنی[1] بارگزاری می‌شود. در روش پیشنهادی اگر یک فایل در زمان تحلیل مشخص شود قسمت code آن خاصیت دیگری به غیر از فقط خواندنی دارد به گونه‌ای که قابلیت نوشتن در آن وجود داشته باشد. به عنوان یک امتیاز منفی با وزن بالا در ارزیابی قرار داده می‌شود دلیل این امر این است که در کنار مورد قبلی بسیاری از بدافزارهایی که قصد خود تغییری در زمان اجرا را دارند با کدهای دیگری را اغلب از قسمت‌های نامتعارف دیگر در زمان اجرا در قسمت کد قرار می‌دهند تا کدهای جدید به جای کدهای قبلی اجرا شوند.

۴-۵. اختلاف در مجموع اندازه قسمت‌ها و اندازه فایل

این مورد یکی از مهم‌ترین موارد در ارزیابی روش پیشنهادی ملاک قرار می‌گیرد بدین ترتیب که مجموع اندازه قسمت‌ها با اندازه کل فایل که در قسمت سرآیند ساختار PE نوشته شده است نباید دارای اختلاف زیادی باشد [۹]. حد مجاز برای این اختلاف در روش پیشنهادی به اندازه حداکثر ۹ درصد از حجم کل فایل می‌باشد بیش از این حد اختلاف به ازای هر ۶ درصد اختلاف یک امتیاز منفی در ارزیابی ملاک قرار داده می‌شود علت این امر هم در این است که برخی از بدافزارها بخش‌هایی از کد اجرایی خود را مخفی می‌کنند که این اختلاف گویا وجود اطلاعاتی به غیر از اطلاعات مشخص شده در قسمت‌های فایل تحت تحلیل است که اغلب برای کدهای اجرایی برای خود تغییری هستند.

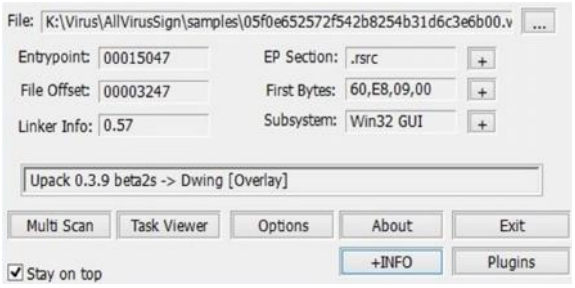
۴-۶. اختلاف در اندازه خام و مجازی فایل

این مورد یکی از مهم‌ترین و پر وزن‌ترین ملاک در روش پیشنهادی است. در این ارزیابی اختلاف بین اندازه خام فایل Raw Size و اندازه مجازی فایل Virtual Size است. Raw Size نشان دهنده اندازه فایل بر روی دیسک سخت و Virtual Size نشان دهنده اندازه فایل در حافظه اصلی می‌باشد که از سرآیند فایل در ساختار PE قابل استخراج هستند [۹]. این اختلاف در صورتی که بیشتر از ۱ درصد حجم کل فایل باشد به عنوان یک امتیاز منفی در روش پیشنهادی مد نظر قرار می‌گیرد. آنگاه بر اساس هر ۳ درصد اختلاف به آن یک وزن افزوده می‌شود.

۴-۷. محل نقطه شروع فایل اجرایی OEP

در این مورد آدرس اصلی محل شروع فایل اجرایی OEP[2] با استفاده از سرآیند ساختار PE محاسبه می‌گردد آنگاه آدرس شروع و انتها قسمت کد نیز محاسبه می‌شود. سپس بررسی می‌شود که آدرس نقطه شروع یعنی OEP در داخل Code Section است یا خیر؟ در حالت عادی برای فایل‌های بی‌خطر نقطه شروع در محدوده قسمت کد است و معمولاً در ابتدای آن است حال اگر این نقطه در آدرسی خارج از محدوده قسمت کد باشد بسیار به بدخواه بودن مشکوک است. بسیاری از بدافزارها توسط ابزارهای محافظت کننده و بسته‌بندی کننده، محافظت و بسته‌بندی شده‌اند. یکی از اصلی‌ترین مشخصه‌های آن همین آدرس نقطه شروع یا OEP می‌باشد. اغلب ابزارهای بسته‌بندی کننده با رمز کردن کدهای اجرایی ناحیه Code Section بر اساس معادلات برگشت‌پذیر ریاضی، اقدام به ایجاد قسمت‌های جدید برای ذخیره کدهایی جهت رمزگشایی قسمت کد می‌نمایند آنگاه ناچارند نقطه شروع را به ابتدای قسمت ایجاد شده انتقال دهند تا به هنگام اجرای فایل ابتدا برنامه رمزگشا اجرا شده و ناحیه کد را رمزگشایی نماید بنابراین یک فایل با ساختار PE می‌تواند چندین نقطه شروع یا Entery Point داشته باشد ولی فقط یکی از آن‌ها نقطه شروع اصلی یا OEP می‌باشد که کدهای واقعی برنامه از آنجا شروع به اجرا می‌کنند.

مضاف بر مطالب ذکر شده بدافزارهای چسبنده به فایل نیز که از روال تزریق کد در فایل هدف خود استفاده می‌کنند. عموماً آدرس نقطه شروع یا OEP را بر روی ابتدای آدرس کدهای تزریق شده خود تنظیم می‌کنند تا در هر بار اجراء کدهای آلوده تزریق شده اجرا شود.



شکل ۳. EP یک نمونه بدافزار از خانواده Trojan که در قسمت rsrc. قرار دارد.

در روش پیشنهادی عدم وجود آدرس نقطه شروع یا OEP در محدوده آدرس قسمت کد به عنوان یک امتیاز منفی در نظر گرفته شده و با توجه به میزان فاصله آدرس OEP از آدرس ابتدای قسمت کد و به ازای هر ۱۰ درصد نسبت کل فضای آدرس‌دهی فایل تحت

[1] Read Only

[2] Original Entry Point

تحلیل وزن به امتیاز منفی داده می‌شود. و در صورتی که نتوان این میزان اختلاف را محاسبه کرد برابر نبودن نقطه‌ای که برنامه از آنجا شروع شده با نقطه OEP که در سرآیند فایل مشخص شده است برای کسب امتیاز منفی کافی است. برای محاسبه EP در یک فایل اجرای از فرمول زیر استفاده می‌شود.

محاسبه نقطه شروع برنامه:

$$EP = ImageBase + C \qquad (۱)$$

در فرمول فوق، EP نقطه شروع برنامه یا Entry Point است که از حاصل جمع محل شروع کد در حافظه یا Image Base با یک مقدار ثابت به دست آمده است. مقدار Image Base تا نسخه ویندوز Vista همواره ثابت و برابر 400000 بود بنابراین محاسبه EP بسیار ساده بود. پس از آن برای مقابله با حملات سرریز بافر و Heap و حملات تزریق کد فناوری ASLR[1] توسط مایکروسافت در نسخه‌های جدید ویندوز مورد استفاده قرار گرفت. این فناوری ناحیه‌های تخصیص یافته به فضاهای Stack, Heap و قسمت‌هایی دیگر را به صورت تصادفی در هر بار اجراء تغییر می‌دهد بنابراین آدرس Image Base نیز در هر بار اجراء تغییر می‌کند. این فناوری همچنین امکان بارگذاری تصادفی کتابخانه‌های مورد استفاده یک برنامه را نیز دارد این فناوری به صورت گسترده توسط دیگر سیستم عامل‌ها نیز مورد استفاده قرار گرفته است به کارگیری فناوری فوق باعث کاهش قابل توجه حملات تزریق کد، شل کدنویسی و سرریز بافر شد گرچه کار محاسبه EP را سخت نموده و باعث افزایش آنتروپی فایل می‌شود. در جدول زیر نام برخی از سیستم عامل‌های استفاده کننده از فناوری ASLR آورده شده است اگرچه همچنان برخی سیستم عامل‌های نظیر اسکادا ترجیح می‌دهند از این فناوری استفاده نکنند.

جدول ۱. نام و نسخه برخی سیستم‌عامل‌های استفاده کننده از فناوری ASLR

ردیف	سیستم عامل	نسخه
۱	لینوکس	هسته نسخه ۲/۶ به بعد
۲	اندروید	نسخه ۴/۰ به بعد
۳	IOS	نسخه ۴/۳ به بعد
۴	سلاریس	نسخه ۱۱/۱ به بعد

۴-۸. میزان آنتروپی

یکی از ویژگی‌های اصلی برای تشخیص در روش پیشنهادی محاسبه آنتروپی[2] است. منظور از آنتروپی میزان تنوع بایت‌ها می‌باشد این تنوع برای قسمت کد محاسبه می‌شود. بایت‌های موجود در ناحیه کد برای یک فایل با ساختار PE بر اساس تعداد دستورات ماشین اینتل ۳۲ یا ۶۴ بیتی تا یک حد مشخص می‌تواند تنوع داشته باشد ولی بیشتر از آن نشان دهنده بسته‌بندی بون کدهای آن ناحیه است علت آن هم رمزنگاری رشته‌ها توسط ابزار بسته‌بندی کننده است. در روش پیشنهادی برای هر فایل که آنتروپی آن بیشتر از ۷ باشد به ازای هر واحد اختلاف یک امتیاز منفی در نظر گرفته می‌شود. میزان آنتروپی از طریق فرمول زیر محاسبه می‌شود.

محاسبه میزان آنتروپی برای یک فایل با ساختار PE:

$$E_T = E_s - A_s + E_m - A_m + E_x - A_x + E_h - A_h \qquad (۲)$$

در فرمول فوق، $\boldsymbol{E_T}$ نشان دهنده مجموع آنتروپی بر حسب bit/Byte است که از جمع آنتروپی در قسمت‌های مختلف به دست می‌آید در جدول (۲) سایر اقلام فرمول توضیح داده شده است.

جدول ۲. اقلام محاسبه میزان آنتروپی

E_s	میزان تنوع در بایت‌های Stack Top
E_m	میزان تنوع در بایت‌های Memory Map
E_x	میزان تنوع در بایت‌های Executable Base
E_h	میزان تنوع در بایت‌های Heap Base
A_s	Attacked bits of Stack Entropy
A_m	Attacked bits Per Attempt of mmap() base Entropy
A_x	Attacked bits per attempt of main Executable base Entropy
A_h	Attacked bits per attempt of heap base Entropy

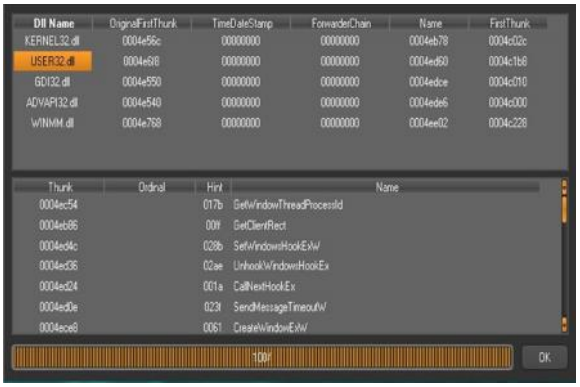
با استفاده از فرمول فوق میزان آنتروپی برای قسمت‌های مختلف ساختار PE یک نمونه بدافزار محاسبه و در تصویر زیر نمودار آن آورده شده است.

همان‌گونه که در شکل فوق مشاهده می‌کنید میزان آنتروپی قسمت کد یک نمونه بدافزار در تحلیل ایستا توسط روش پیشنهادی بسیار بالا و در کنار خواص دیگری از جمله، بسته‌بندی بودن قسمت rdata و برنامه‌نویسی پویا امتیاز منفی بسیار بالا و بسیار مشکوک به بدافزار دریافت کرده است.

نکته حائز اهمیت در امتیازدهی به آنتروپی برای یک فایل تحت تحلیل قابلیت‌های سیستم عامل می‌باشد در صورتی که سیستم عامل از قابلیت ASLR پشتیبانی کند آنگاه حد آستانه ثبت امتیاز منفی برای برنامه تحت تحلیل نیز به میزان یک واحد افزایش می‌یابد علت آن هم اضافه شدن آنتروپی با تصادفی‌سازی‌های اعمال شده توسط ASLR است.

[1] Address Space Layout Randomization
[2] Entropy

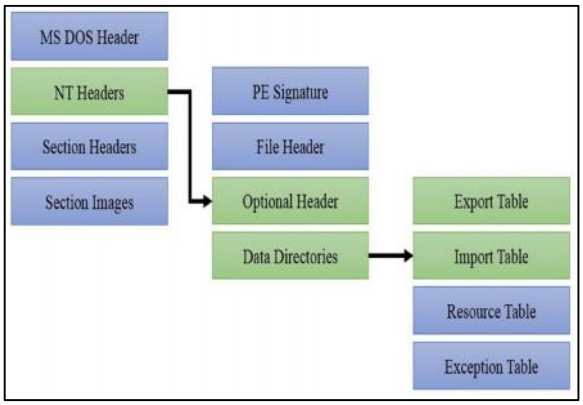
شکل ۴. محاسبه میزان آنتروپی یک نمونه بدافزار



شکل ۶. جدول IAT یک نمونه بدافزار از خانواده Infector

۴-۹. توابع سیستمی موجود در IAT

این مورد مهم‌ترین مورد در روش پیشنهادی در خصوص تصمیم‌گیری درباره ماهیت فایل را دارد در این روش پس از بررسی معتبر بودن IAT نام توابعی که توسط برنامه مورد تحلیل استفاده خواهد شد و ماژول‌های حاوی آن‌ها استخراج می‌شود. آنگاه بر اساس لیست تهیه شده از توابع خطرناک سیستم عامل یعنی توابعی که در حالت معمول کمتر مورد استفاده قرار می‌گیرند [۱۵] با داده‌کاوی روی حجم انبوهی از بدافزارهای و فایل‌های بی‌خطر گردآوری شده از منابع [۱۴-۱۱] قوانینی در خصوص تصمیم‌گیری در مورد بدخواه یا بی‌خطر بودن فایل استخراج شده است. این قوانین پس از فیلتر شدن در خصوص دسته‌بندی بدافزارها کاربرد بسیاری دارند. در ادامه محل قرارگیری جدول‌های IAT و EAT در ساختار PE آورده شده است.



شکل ۵. نحوه محاسبه آدرس جدول IAT در ساختار PE

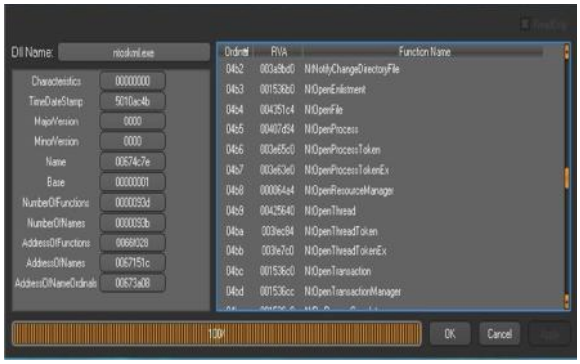
بعد از محاسبه آدرس جدول IAT در ساختار PE اطلاعات موجود در این جدول از جمله نام توابع مورد استفاده برنامه به همراه ماژول‌های مالک آن‌ها قابل استخراج است، این موارد در شکل (۶) نشان داده شده است.

۴-۱۰. وجود قسمت منابع

قسمت منابع یا Resource که به صورت rsrc در ساختار PE نام‌گذاری می‌شود حاوی اطلاعات مربوط به منابع رابط کاربری سیستم از جمله تصاویر و نمایه‌ها می‌باشد. اکثر بدافزارها فاقد رابط کاربری هستند و این در حالی است که درصد قابل توجهی از فایل‌های بی‌خطر دارای رابط کاربری هستند بنابراین وجود قسمت معتبر rsrc. به عنوان یک امتیاز مثبت در ارزیابی روش پیشنهادی قرار می‌گیرد. نکته حایز اهمیت در خصوص وجود این قسمت در معتبر بودن آن است زیرا برخی از بدافزارها جهت فریب دادن برنامه‌های ضدبدافزار قسمت‌های rsrc. جعلی درست می‌کنند ملاک اعتبار این قسمت در قابل دسترس بودن و صحیح بودن آدرس‌ها موجود در قسمت مذکور است.

۴-۱۱. وجود جدول آدرس توابع خروجی

جدول آدرس توابع خروجی در قسمت edata است. این جدول نام و آدرس توابع خروجی توسط برنامه را نشان می‌دهد. این توابع می‌توانند توسط برنامه دیگری که فایل جاری را Import کند مورد استفاده قرار گیرد. بسیار نادر است که بدافزار تابعی را برای استفاده دیگر برنامه‌ها ارائه دهد و در مقابل اغلب برنامه‌های بی‌خطر به ویژه کتابخانه‌های پیوند پویا در مجموعه‌های نرم‌افزاری ارائه دهنده این توابع هستند. بنابراین در روش پیشنهادی وجود جدول EAT معتبر یک امتیاز مثبت برای بی‌خطر بودن فایل در حال تحلیل می‌باشد با توجه به تعداد توابع خروجی وزن این امتیاز مثبت افزایش می‌یابد. همانند مورد قبل نکته مهم در معتبر بودن این جدول است زیرا برخی از بدافزارها برای فریب، جدول‌های EAT جعلی ایجاد می‌کنند. ملاک صحت جدول‌های EAT در روش پیشنهادی صحیح بودن آدرس‌های توابع خروجی و قابل فراخوانی بودن آن‌ها است.



شکل ۷. جدول EAT یک نمونه برنامه بی‌خطر

شکل شماره ۸. جدول EAT یک نمونه برنامه بدافزار از خانواده Rootkit

۴-۱۲. وجود پوشه اشکال‌زدایی

پوشه اشکال‌زدایی در بخش آخر از ساختار PE قرار دارد که حاوی اطلاعات مربوط به اشکال‌زدایی برنامه نظیر نمادهای اشکال‌زدایی است. وجود پوشه اشکال‌زدایی و معتبر بودن آن حکایت از کامپایل شدن برنامه در حالت Debug دارد. گاهی برنامه‌های معمولی به ویژه در نسخه‌های بتا برای آزمون و اشکال‌زدایی توسط کاربران در نسخه Debug منتشر می‌شوند ولی غیر ممکن است ویروسی در این حالت کامپایل شود. در روش پیشنهادی وجود این پوشه و کامپایل در حالت Deubug یک امتیاز مثبت برای بی‌خطر بودن فایل است.

۴-۱۳. نرخ خوش‌خیمی و بدخیمی

در این قسمت فرمول به دست آمده از داده‌کاوی برای تعیین میزان مشکوک بودن یک فایل ناشناس بر اساس خواص فوق به بدافزار را تعیین می‌کند.

محاسبه نرخ میزان مشکوک بودن فایل به بدافزار:

$$Suspicious\ Rate = \frac{\sum W \times P_i + \sum W \times N_j}{\sum Wi + Wj} \qquad (۳)$$

در فرمول فوق، $\sum W \times P_i$ حاصل جمع امتیازهای مثبت داده شده به فایل تحت تحلیل با در نظر گرفتن وزن‌های آن است و $\sum W \times N_j$ حاصل جمع امتیازهای منفی داده شده به همراه وزن‌های آن‌ها است. $\sum Wi + Wj$ نشان دهنده مجموع امتیازهای مثبت و منفی قابل اعمال به فایل تحت تحلیل با در نظر گرفتن وزن هر یک از آن‌ها است.

اگر نتیجه فرمول عددی مثبت شود بنابراین فایل تحت تحلیل به بی‌خطر بودن نزدیک‌تر است و اگر عددی منفی حاصل شود فایل تحت تحلیل به بدافزار بودن متمایل است. این مورد به صورت قانون زیر قابل بیان است:

IF Suspicious Rate = Negative Then Suspicious to Malware
Else Not Malware

محاسبه نرخ‌های خوش‌خیم / بدخیم:

$$Malware\ Rate = \frac{\sum W \times N_i}{\sum Wj} \qquad (۴)$$

$$Benign\ Rate = \frac{\sum W \times P_i}{\sum Wi} \qquad (۵)$$

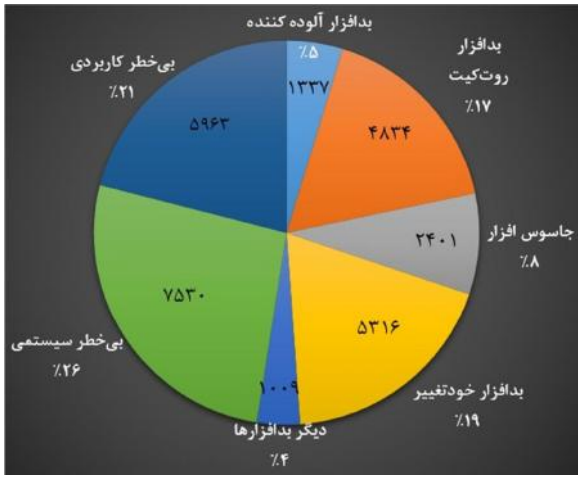
فرمول‌های فوق، میزان بدخیمی و خوش خیمی فایل اجرایی را به صورت جداگانه اندازه‌گیری می‌کنند. با توجه به فرمول‌های فوق نرخ بی‌خطر بودن یا بدافزار بودن برای یک برنامه ناشناس از روی مجموع خواص قابل استخراج آن محاسبه می‌شود. حال هر یک از این نرخ‌ها بیشتر باشد آنگاه برنامه به قرارگیری در آن دسته متمایل شده با استفاده از فرمول‌های استخراج شده به طور دقیق دسته‌بندی می‌شود. مورد ذکر شده در قالب قانون زیر تعریف می‌شود:

IF MalwareRate > BenignRate Then Suspicious to Malware
Else No Malware

۵. تشخیص و خوشه‌بندی

در این قسمت با داده‌کاوی انجام شده بر حجم انبوهی از بدافزارها و فایل‌های بی‌خطر آن‌ها را با توجه به میزان شباهت و نزدیکی خواص ذکر شده در قسمت قبل این مقاله خوشه‌بندی نموده تا به هنگام مواجه با یک فایل ناشناخته علاوه بر تشخیص بد خواه بودن یا بی‌خطر بودن، بتوانیم آن فایل را در یکی از دسته‌های موجود قرار داده و اطلاعات دقیق‌تری از ماهیت آن به دست آوریم.

بسیار حائز اهمیت است که تعداد نمونه موجودیت‌های شرکت کننده در داده‌کاوی عددی مناسب باشد. تعداد کم نمونه موجودیت باعث تولید قوانین ناکارآمد و در نتیجه خوشه‌بندی غلط شود همچنین نباید اختلاف تعداد بین نمونه موجودیت‌ها در خوشه‌های مختلف زیاد باشد چرا که باعث افزایش نرخ کاذب مثبت خواهد شد به عنوان نمونه می‌توان اشاره کرد که به دلیل تعداد بسیار کم نمونه موجودیت‌ها از دسته بدافزارهای تبلیغاتی یا Adwareها که تنها ۱۴۹ نمونه در مراجع [۱۴-۱۱] بود از شرکت در داده‌کاوی کنار گذاشته شدند.



شکل ۹. تعداد و درصد نمونه موجودیت‌های شرکت کننده در داده‌کاوی در روش پیشنهادی به تفکیک خوشه‌ها

شکل ۱۰. تعداد و درصد مجموع نمونه موجودیت‌های شرکت کننده در داده کاوی برای آموزش مدل رفتاری روش پیشنهادی

لازم به ذکر است در روش پیشنهادی، مدل‌سازی در یک مقیاس مناسب از تعداد نمونه بدافزارها و فایل‌های بی‌خطر شرکت کننده در داده‌کاوی انجام شده است و از حیث مقیاس‌پذیری قابل تعمیم به مقیاس‌های بزرگ‌تر می‌باشد و همچنین در روش پیشنهادی نمونه بدافزارها از زمان‌های مختلف از مراجع [۱۱ و ۱۲] جمع‌آوری شده‌اند علت این امر در متفاوت بودن رفتار برحسب فناوری‌های زمان ساخت است در ادامه خانواده‌های بدافزار حاصل از داده‌کاوی تشریح می‌شود.

در روش پیشنهادی نتایج از داده‌کاوی و آزمون توسط ابزار Weka در حالت یادگیری نیمه مدیریت شده[1] با استفاده از الگوریتم‌های موجود در آن استفاده شده است در داده‌کاوی از الگوریتم Jrip برای تولید قانون و الگوریتم J48 برای ایجاد درخت تصمیم جهت تعبیه در سامانه اسکن برنامه ضدویروس استفاده شده است در ادامه یک نمونه قانون برای تشخیص بدافزار از دسته آلوده کننده آورده شده است.

IF ((CodeSectionAttributes = WritableReadableExecutable) and (DifOfSectionsSizeAndFileSize <= 0.000006) and (SubOfVirtualAndRawSizeOfodeSection >= 0.000001) and (NumberOfUnknownNamedSections <= 0) and (ResourceDirectorySize >= 0.000436) and (SubOfVirtualAndRawSizeOfodeSection <= 0.000045)) Then IsMalware = Virus.Infector (۱.۰/۴۵.۰)

قانون فوق با استفاده از خواص ذکر شده مانند خاصیت ناحیه کد، اختلاف بین اندازه مجازی و اندازه خام، تعداد قسمت‌های ناشناخته، وجود قسمت منابع، بدافزار بودن را تأیید و آن را از نوع آلوده کننده تشخیص می‌دهد.

۵-۱. جاسوس‌افزار

در این دسته تمام بدافزارهایی که به نوعی قصد سرقت اطلاعات را با هرگونه روشی اعم از نصب شنودگر، دسترسی و تکثیر غیر مجاز اطلاعات دارند، قرار داده می‌شود. برای دسته‌بندی و تمیز دادن این دسته از بدافزارها نیاز مند بررسی و شناخت عملکرد درونی آن‌ها می‌باشیم تا با تولید قوانینی از عملکرد آن‌ها امکان دسته‌بندی آن‌ها را در فرآیند داده‌کاوی فراهم نماییم.

نحوه عملکر این ابزار ها تا حد بسیار زیادی شبیه یک دیگر است از این حیث که همگی برای رسیدن به اهداف خود و سرقت اطلاعات نیازمند قلاب اندازی به امکانات سیستم عامل به ویژه توابع سیستمی هستند. هدف از قلاب اندازی این است تا با واسط قرار گرفتن بین منابع سیستم عامل و درخواست‌های وارده شده از سوی کاربر یا برنامه‌های کاربردی نصب شده روی سیستم قربانی اقدام به ثبت و سرقت اطلاعات لازم نمایند. بسیاری از این بدافزارها می‌توانند اطلاعات سرقت شده را در زمان‌های تصادفی با استفاده از اتصال شبکه برای مقصدی خاص ارسال کنند [۲۳]. برای مطالعه بیشتر در خصوص نحوه تشخیص جاسوس‌افزارها به مرجع مذکور مراجعه شود.

۵-۲. بدافزار آلوده کننده (چسبنده)

بدافزارهایی که در این دسته قرار می‌گیرند، بدافزارهایی هستند که اقدام به آلوده کردن فایل‌های دیگر به ویژه فایل‌های اجرایی قابل حمل یعنی PE می‌کنند. اکثریت این بدافزارها برای آلوده کردن فایل‌های دیگر نیازمند تزریق کدهای آلوده و یا کتابخانه‌های پیوند پویا آلوده در فایل‌های قربانی دارند [۲۲] و بر اساس قوانینی که وجود یک برنامه کوچک[2] تزریق کننده را در یک فایل اثبات می‌کند در آن-است. شناسایی و دسته‌بندی می‌شوند. شناسایی برنامه تزریق کننده ملزم آشنایی با روال تزریق کد یا کتابخانه پیوند پویا دارد [۲۱].

[1] Semi Supervised

[2] Stub

۱. ایجاد یک فرآیند در حالت معلق CreateProcess
۲. درخواست تخصیص حافظه به نام فرآیند مذکور VirtualAllocEx
۳. نوشتن کدهای مخرب در حافظه تخصیص یافته WriteProcessMemory
۴. بارگذاری فایل کتابخانه پیوند پویا در حافظه تخصیص یافته LoadLibrary
۵. اجرای کدهای مخرب با ایجاد نخ راه دور CreateRemoteThread
۶. انتظار برای اتمام فرآیند تزریق WaitforSignalObject
۷. اجرای نخ اصلی ResumeThread

شکل ۱۱. الگوریتم تزریق کد و کتابخانه پیوند پویا [۲ و ۱۹]

وجود نام توابع مذکور در جدول IAT یک برنامه، در کنار توالی آن‌ها نمایانگر وجود یک برنامه تزریق کننده است و در صورتی که طبق فرمول ذکر شده بدخواه بودن این فایل اثبات شده باشد با اثبات وجود توابع فوق به ترتیب ذکر شده، بر اساس قانون مذکور می‌توان آن فایل را در دسته بدافزارهای آلوده کننده یا Infector قرار داد. بسیاری از بدافزارها برای سهولت کار، از برنامه‌های تزریق کننده آماده استفاده می‌کنند از جمله پر کاربردترین این برنامه‌ها می‌توان به EasyHook و Hook Tool SDK برای زبان دات نت و Marshal SDK برای زبان دلفی اشاره کرد. روش پیشنهادی در صورت مشاهده هر کدام از استاب‌های فوق در یک برنامه آن را به عنوان یک بدافزار آلوده کننده خواهد شناخت. این تشخیص با جستجوی زیر دنباله‌هایی یکتا از استاب‌های فوق با استفاده از n-gram صورت می‌پذیرد.

۵-۳. بدافزار روت‌کیت سطح هسته

بدافزارهایی که در این دسته قرار می‌گیرند، بدافزارهایی هستند که قصد ورود به فضای هسته سیستم عامل و بارگذاری هر گونه درایور برای مخفی‌سازی دیگر برنامه‌ها را داشته باشند [۱۶]. این دسته از بدافزارها خطرناک‌ترین دسته محسوب شده زیرا با ورود آن‌ها به فضای هسته سیستم عامل و برخورداری از عدم اعمال محدودیت‌های سطح کاربر توسط سیستم، به ویژه محدودیت‌های اعمالی توسط UAC[1] سیستم عامل‌های ویندوز ۷ و ۸ امکان کنترل و رفع آلودگی ناشی از آن‌ها بسیار سخت می‌شود. امروزه برخی از برنامه‌های ضدبدافزار برای ترمیم سیستم عامل قربانی به این دسته از بدافزارها امکاناتی نظیر دیسک‌های نجات را گسترش داده‌اند. این امکان به ویژه در مواردی که بدافزار پس از نفوذ به هسته سیستم عامل اجازه نصب هر گونه برنامه ضد بدافزار را محدود می‌کند کاربرد دارد در این ضد بدافزارها یک برنامه اسکنر تحت یک سیستم عامل قابل حمل که اغلب توزیع‌های سفارشی شده از هسته لینوکس می‌باشند به صورت Bootable از روی دیسک نوری اجرا و درایورهای سیستمی را به منظور کشف و رفع آلودگی اسکن می‌کند.

۹۰ درصد بدافزارهای روت کیت نیاز به اجرا در سطح هسته سیستم عامل دارند [۲۴] با علم به این موضوع در روش پیشنهادی ملاک تشخیص و دسته‌بندی این بدافزارها وجود نام توابع سیستمی مرتبط با ثبت و راه اندازی سرویس در جدول IAT فایل‌های اجرایی آن‌ها می‌باشد. در روش پیشنهادی قانون دسته‌بندی برای این دسته از بدافزار پس از تشخیص بد خواه بودن وجود نام توابع مربوط به کار با سرویس‌های سیستم عامل شامل توابع زیر است.

Create Service, StartService, ControlService

این توابع توسط بدافزارهای مذکور عمدتاً برای بارگزاری درایور در سیستم استفاده می‌شوند [۲۵].

همچنین می‌توان با استفاده از تکنیک‌های Mempry Farensic این گونه بدافزارها را با نظارت بر دسترسی‌های غیر مجاز به حافظه سیستم تشخیص داد در مرجع [۲۴] راهکاری برای این منظور پیشنهاد شده است.

۵-۴. بدافزار خود تغییر[2]

این دسته در بردارنده بدافزارهایی است که مبادرت به اعمال هرگونه تغییر در کدهای اجرایی خود کنند. تشخیص این دسته از بدافزارها به علت تغییرات دائمی که در خود ایجاد می‌کنند بسیار سخت است به گونه‌ای که تقریباً امکان کشف آن‌ها با روش‌های مبتنی بر امضاء تقریباً غیر ممکن و با روش‌های تحلیل ایستا نیز بسیار سخت است. در حقیقت عمل تغییر در کدهای اجرایی یک برنامه توسط برنامه دیگری صورت می‌پذیرد که اطلاعات لازم در خصوص برنامه اول را در اختیار دارد. بنابراین این بدافزارها برای انجام عمل خود تغییری نیاز مند برنامه دیگر هستند و اغلب این برنامه کوچک را درون محتویات فایل‌های اجرایی خود پنهان می‌سازند [۱۷ و ۱۸].

تنها راه تشخیص این بدافزارها در تحلیل ایستا تشخیص، اثبات وجود یک برنامه Stub خود تغییری در محتویات فایل اجرایی است. در روش پیشنهادی پس از تعیین بد خواه بودن یک فایل اجرایی در صورت مشاهده نام توابعی زیر وجود Stub خود تغییری محتمل می‌شود. بدافزارهای تصدیق کننده این قانون در دسته بدافزارهای خود تغییر قرار داده می‌شوند از جمله توابعی که برای این‌گونه رفتار استفاده می‌شود ReadProcessMemory و WriteProcessMemory با دستگیره‌ای به پردازه خود برای آرگمان اول است.

با توجه به موارد گفته شده تشخیص این دسته از بدافزار بسیار سخت‌تر از سایرین می‌باشد و این امر سبب کاهش نرخ کشف مثبت واقعی و افزایش نرخ کشف کاذب مثبت در این دسته نسبت به میانگین کل دسته‌ها می‌باشد. روش‌های تقلید رفتار یا Emulation نیز برای تشخیص این دسته از بدافزاها نیز محبوب است [۲۶]. برخی از شرکت‌های تولید کننده ضدبدافزار مانند KasperSky از این روش با

[1] User Access Control

[2] Self-Modifying

استفاده از ابزار Bouch سود می‌برند در مرجع [۲۷] راهکاری برای انجام این نوع تشخیص پیشنهاد شده است.

در روش پیشنهادی راه حل دقیق‌تر برای تشخیص این دسته از بدافزارها در زمان اجراء و به صورت پویا ارائه می‌شود تا اگر به هر دلیلی در زمان ایستا توسط روش پیشنهادی کشف نشدند در زمان اجرا شناسایی شوند. در راه حل پیشنهادی برنامه در یک بازه زمانی مشخص از قسمت کد پردازه‌های فعال در سیستم عامل امضاء تهیه می‌کند سپس برای هر پردازه با بازکردن ماژول اصلی مسیر فایل اجرایی بر روی دیسک سخت را مشخص کرده و از قسمت کد در صورتی که فایل مورد نظر بسته‌بندی نباشد امضاء گرفته می‌شود آنگاه این دو امضاء با یکدیگر مقایسه می‌شوند در صورتی که یکسان نباشند قطعاً برنامه خودتغییری داشته است.

۵-۵. دیگر بدافزارها

سایر فایل‌های اجرایی که بر اساس خواص و فرمول ذکر شده در قسمت قبل این مقاله بدافزار تشخیص داده شده‌اند ولی بر اساس قوانین تولید شده در هیچ کدام از این دسته‌بندهای ذکر شده تاکنون قرار نگرفته‌اند در این دسته قرار خواهند گرفت. نکته مهم در خصوص دسته‌بندی بدافزارها این است که با توجه به میزان شناخت سازوکارهای عملکرد بدافزارها و تعداد نمونه موجودیت‌های شرکت کننده در داده‌کاوی امکان تولید قوانین بیشتر و دقیق‌تر برای دسته‌بندی بدافزارها وجود دارد ولی در صورت کم بودن تعداد قوانین افزایش تعداد دسته‌ها موجب کاهش نرخ کشف و افزایش نرخ کاذب مثبت خواهد شد بنابراین با توجه به جامعه آماری که از بدافزارها و فایل‌های بی‌خطر در اختیار نویسندگان این مقاله قرار داشت، بر اساس قوانین تولید شده بهینه‌ترین تعداد برای دسته‌بندی همین تعداد ذکر شده می‌باشد به عنوان مثال با تولید چند قانون برای ایجاد دسته‌ای جدید از بدافزارها تحت عنوان ابزاهای تبلیغی یا Adware ها کاهش ۲۰ درصدی در نرخ کشف به وجود آمد. زیرا نمونه موجودیت‌های مربوط به Adwareها و بر حسب آن، قوانین تولید شده کافی نبود. در مرجع [۵] راهکاری برای حل این مشکل پیشنهاد شده است.

مسئله دیگر در خوشه‌بندی بدافزارها این است که برخی از بدافزارها رفتار چند خوشه به صورت هم‌زمان را دارند به عنوان مثال روت کیت‌های که جاسوس‌افزار هم هستند روش پیشنهادی در خصوص این بدافزارها رفتاری که زودتر کشف نماید را ملاک قرار می‌دهد.

۶-۵. بی‌خطر سیستمی

حجم قابل توجهی از فایل‌های بی‌خطر که برای شرکت در داده‌کاوی جمع‌آوری شده است فایل‌های بی‌خطر سیستمی می‌باشد این فایل‌ها، فایل‌های اجرایی و کتابخانه‌های پیوند پویا با قالب فایل اجرایی قابل حمل از سیستم عامل‌های ویندوز XP، Vista، ۷ و ۸ در نسخه‌های ۳۲ و ۶۴ بیتی با حذف فایل‌های تکراری است.

۷-۵. بی‌خطر کاربردی

سایر فایل‌های بی‌خطری که در داده‌کاوی شرکت داده شده اند فایل‌ها و کتابخانه‌های پیوند پویا با ساختار PE از نرم افزارهای کاربری قابل نصب بر روی نسخه‌های مختلف سیستم عامل ویندوز عمدتاً از مجموعه نرم‌افزارهای Lord و King بوده است بنابراین تمام فایل‌های بی‌خطر غیر سیستمی در این دسته قرار می‌گیرند.

۶. ارزیابی و مقایسه روش پیشنهادی در تحلیل ایستا

در این قسمت میزان دقت روش پیشنهادی را برای کشف بدافزارها و دسته‌بندی آن‌ها نشان می‌دهیم. میزان تشابه در ساختار بدافزارها و برنامه‌های بی‌خطر حداقل ۱۴ و حداکثر ۹۳ درصد و در حالت معمول ۳۵ درصد است [۲۸] بنابراین انتخاب دقیق خواص ذکر شده و تعداد نمونه موجودیت شرکت کننده و تعادل بین خوشه‌های مختلف عواملی بسیار تأثیرگذار در دقت مدل هستند در ادامه دقت روش پیشنهادی را برازش می‌کنیم.

میزان تشخیص مثبت واقعی True Positive و مثبت کاذب False Positive در روش پیشنهادی به تفکیک خوشه‌ها و همچنین میزان دقت Accuracy بر اساس فرمول زیر در روش پیشنهادی محاسبه می‌شود.

محاسبه میزان دقت یا Accuracy:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (۴)$$

رابطه بین میزان $FP^{۱}$ ، $FN^{۲}$ ، $TP^{۳}$ و $TN^{۴}$:

$$TN + FP = 1 \quad (۵)$$

$$TP + FN = 1 \quad (۶)$$

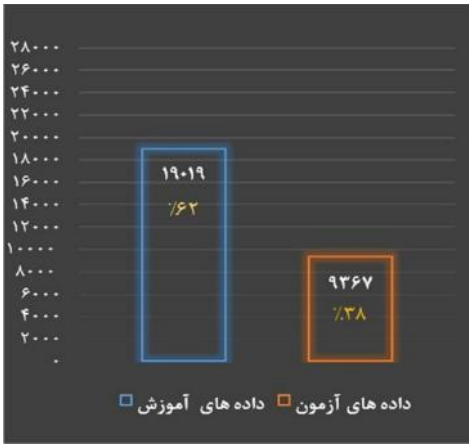
همان‌طور که در جدول (۳) ملاحظه می‌فرمایید بیشترین دقت مربوط به دسته بی‌خطر سیستمی و کمترین دقت مربوط به بدافزارهای خود تغییر است. در تمام مراحل داده‌کاوی میزان تقسیم داده‌ها به مجموعه‌های آزمون و ساخت مدل بر اساس شکل (۱۲) بوده است.

با توجه به آنچه گفته شد میزان دقت روش پیشنهادی در تشخیص و دسته‌بندی بدافزارها و فایل‌های بی‌خطر مشخص گردید.

[1] False Positive
[2] False Negative
[3] True Positive
[4] True Negative

جدول ۳. نرخ تشخیص مثبت واقعی True Positive و مثبت کاذب False Positive در روش پیشنهادی به تفکیک خوشه‌ها

ردیف	خوشه	TP	FN	FP	TN	ACC
۱	بدافزار آلوده کننده	۹۸۰.۰	۰۲.۰	۱۸۹.۰	۸۱۱.۰	۸۹۵.۰
۲	بدافزار روت کیت	۰.۹۹۶	۰.۰۰۴	۰.۰۳۰	۰.۹۷	۰.۹۸۳
۳	بدافزار خود تغییر	۰.۹۵۳	۰.۰۵	۰.۲۴۱	۰.۷۵۹	۰.۸۷۴
۴	جاسوس‌افزار	۰.۹۸۱	۰.۰۱۹	۰.۰۳۷	۰.۹۶۳	۰.۹۷۲
۵	دیگر بدافزارها	۰.۸۹۰	۰.۱۱	۰.۰۶۸	۰.۹۳۲	۰.۹۱۱
۶	بی‌خطر سیستمی	۰.۹۹۸	۰.۰۰۲	۰.۰۲۲	۰.۹۷۸	۰.۹۸۸
۷	بی‌خطر کاربردی	۰.۹۹۴	۰.۰۰۶	۰.۰۳۱	۰.۹۶۹	۰.۹۸۱



شکل ۱۲. میزان تقسیم‌بندی داده‌ها برای ساخت و آزمون مدل

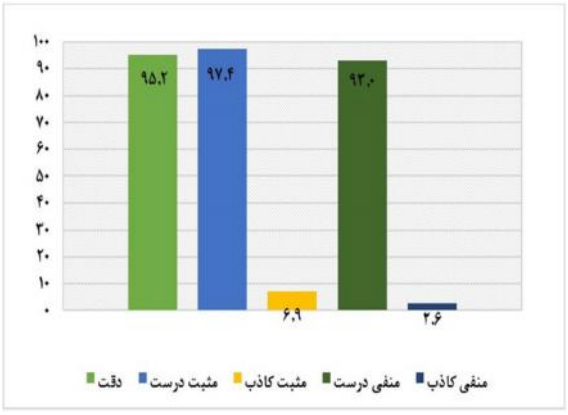
۶-۱. ارزیابی و مقایسه دقت روش پیشنهادی

در این قسمت روش پیشنهادی را از نظر دقت در تشخیص با روش موجود مقایسه می‌کنیم البته لازم به ذکر است که مقایسه با روش‌های دیگر از نظر میزان دقت تشخیص مشروط به یکسان بودن محیط و داده‌آزمون[1] است

شرایط داده‌آزمون: برای ارزیابی و برازش مدل‌های رفتاری باید از داده‌آزمون با شرایط زیر استفاده نمود.

۱. جامع باشد، یعنی دربردارنده تمام خانواده بدافزارها و فایل‌های بی‌خطر خوشه‌بندی شده باشد.
۲. از داده‌های شرکت کننده در یادگیری مدل نباشد، داده‌های انتخاب شده برای آزمون مدل حتماً باید داده‌هایی، متفاوت از داده‌های شرکت کننده در یادگیری باشد.
۳. توزیع زمانی مناسب داشته باشد، یعنی نمونه‌های انتخاب شده برای آزمون متعلق به یک زمان خاص نباشد. علت این امر در این است بدافزارها برحسب فناوری‌های موجود در زمان ساختشان می‌توانند رفتارهای متفاوتی داشته باشند بنابراین اگر نمونه بدافزارها برای ساخت مدل محدود به یک زمان خاص باشند آنگاه تنوع و تفاوت در رفتار بدافزارها در مدل ساخته شده کمرنگ شده و بدافزارها با رفتارهای قدیمی‌تر قابل کشف نخواهند بود البته اگر در نمونه‌برداری از بدافزارها با زمان انتشار بسیار دور استفاده شود دقت مدل به دلیل ورود داده‌های نامرتبط در داده‌کاوی کاهش خواهد یافت بنابراین نمونه‌برداری باید در یک بازه زمانی متناسب از زمان انتشار بدافزارها صورت پذیرد.

در ارزیابی روش پیشنهادی از داده‌آزمون‌هایی متفاوت با داده‌های شرکت کننده در داده‌کاوی و در برگیرنده همه خوشه‌های مذکور استفاده شده است این داده‌آزمون در مراجع [۲۹] قرار داده خواهد شد. نتایج این داده‌کاوی برای ارزیابی دقت روش پیشنهادی بر اساس نرخ‌های مثبت‌درست و کاذب‌مثبت به شرح زیر است.

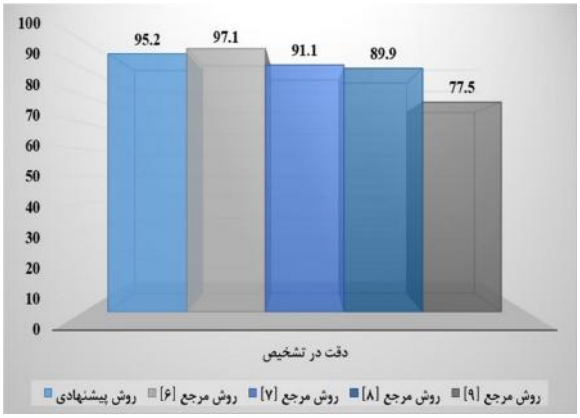


شکل ۱۳. نتایج حاصل از ارزیابی روش پیشنهادی

نتایج فوق میانگین نرخ مثبت‌درست، مثبت‌کاذب، منفی‌درست، منفی کاذب و دقت روش پیشنهادی را که براساس فرمول‌های شماره (۶-۴) و داده‌آزمون‌ها با فراوانی نشان داده شده در شکل (۱۲)، ارزیابی شده است را نشان می‌دهد.

در ادامه و در شکل زیر، روش پیشنهادی از نظر میزان دقت، در تشخیص و خوشه‌بندی با روش‌های موجود مقایسه شده است.

[1] Test Data



شکل ۱۴. مقایسه دقت روش پیشنهادی با سایر روش‌های موجود

بر حسب گزارش مرجع [۲۰] تا نیمه سال ۲۰۱۴ در دنیا فقط ۹ روش تا زمان ارائه مرجع مذکور می‌باشد و روش پیشنهادی این مقاله در مقایسه انجام شده با روش‌های دیگر که میزان دقت آن‌ها توسط ارائه دهندگان ارزیابی و در مقالات علمی منتشر شده است، در جایگاه دوم قرار می‌گیرد.

۶-۲. چالش‌ها و روش‌های ضد تحلیل

یکی از چالش‌های اصلی تحلیل ایستا مواجهه با بدافزرهای بسته‌بندی شده و مبهم شده و بدافزارهای هوشمندی است که با روش‌های برنامه نویسی پویا و تولید کد در زمان اجراء سعی در پنهان‌سازی رفتار خود دارند، می‌باشد که در آن صورت امکان استخراج همه خواص ذکر شده در این مقاله وجود نخواهد بنابراین باید تا حد امکان این روش‌ها که از آن‌ها با عنوان روش‌های ضدتحلیل یاد می‌شود مقابله نمود. در مقاله دیگری ضمن تشریح آن‌ها راهکارهایی پیشنهادی خود را برای مقابله مطرح خواهیم نمود.

۷. نتیجه‌گیری

با توجه به مطالب گفته شده، در این مقاله نشان داده شد که چگونه با بررسی دقیق ساختار PE می‌توان خواصی را برای تحلیل ایستا فایل‌های اجرایی استخراج و مدل نمود. این خواص‌ها شامل تعداد، نام، اندازه قسمت‌ها در ساختار PE، بررسی جداول توابع ورودی و خروجی و مواردی از این دست بوده که به منظور تشخیص و دسته‌بندی بدافزارها و فایل‌های بی‌خطر استفاده شد. در روش پیشنهادی با استفاده از تکنیک‌های یادگیری ماشین مدل رفتاری برای سنجش میزان بدخیم یا خوش خیم بودن یک فایل اجرایی با انتصاب امتیازات مثبت و منفی بر اساس فرمول‌های تعریف شده ایجاد و با استفاده از قوانین تولیده شده در داده‌کاوی بر حجم انبوهی از بدافزارها و فایل‌های بی‌خطر برای تشخیص و دسته‌بندی هوشمند بدافزارها و فایل‌های بی‌خطر آموزش داده شد.

در روش پیشنهادی با به کارگیری قوانین استخراج شده امکان دسته‌بندی دقیق بدافزارها در ۵ و فایل‌های بی‌خطر در ۲ دسته فراهم شد. در پایان میزان کارا بودن روش پیشنهادی، بر اساس میزان دقتش در تشخیص و دسته‌بندی بدافزارها و فایل‌های بی‌خطر ارزیابی و با روش‌های موجود مقایسه شد. این ارزیابی و مقایسه حکایت از دقت ۹۵/۲ درصدی روش پیشنهادی در تشخیص و خوشه‌بندی داشته که روش پیشنهادی را در جایگاه دوم نسبت به دیگر روش‌ها قرار دارد.

۸. مراجع

[1] Infographic: The State of Malware, http://www.mcafee.com/in/security-awareness/articles/state-of-malware-2013.aspx., 2013.

[2] Russinovich, M.; Solomon, D.; Ionescu, A. "Windows Internals Part 1"; 6th, 2012.

[3] Petzold, C. "Programming Windows"; 6th, 2013.

[4] Idika, N.; Mathur, A. P. "A Survey of Malware Detection Techniques"; Purdue Univ. 2007, 48.

[5] Javaheri, D. "Design and Implementation a Secure and Intelligent Environment for Malware Analysis"; M.Sc. Thesis, Islamic Azad University Borujerd Banch, 2014 (In Persian).

[6] Schultz, M.; Eskin, E.; Zadok, F.; Stolfo, S. "Data Mining Methods for Detection of New Malicious Executables"; In Proc. of 2001 IEEE Symposium on Security and Privacy, Oakland, 14-16 May 2001, 38-49.

[7] Kolter, J.; Maloof, M. "Learning to Detect Malicious Executables in the Wild"; In Proc. of the 10th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining 2004, 470-478.

[8] Siddiqui, M.; Wang, M.C.; Lee, J. "Detecting Internet Worms Using Data Mining Techniques"; J. of Systemics, Cybernetics and Informatics 2009, 6, 48-53.

[9] Gao, D.; Yin, G.; Dong, Y.; Kou, L. "A Research on the Heuristic Signature Virus Detection Based on the PE Structurerd"; In Proc. Int. Conf. on Electric and Electronics (EEIC), 2013.

[10] Alirezaei E. "Behavioral Analysis of Malicious Code"; M.Sc. Thesis, Kish Paradise Univ. of Tehran, Kish, 2011. (In Persian)

[11] "Virus Sign Malware Data Base"; http://www.virussign.com/, 2014.

[12] "CWSandbox Data"; http://pi1.informatik.uni-mannheim.de/malheur/, 2014.

[13] "Malware Data Base."; http://borax.poluxhosting.com/madchat/vxdevl/vxsrc, 2008.

[14] "Malware Data Base"; http://www.vx.netlux.org; 2007.

[15] Fu, W.; Pang, J.; Zhao, R.; Zhang, Y.; Wei, B. "Static Detection of API-Calling Behavior from Malicious Binary Executables"; In Proc. Int. Conf. on Computer and Electrical Eng.(ICCEE), 2008, 388-392

[16] Hoglund, G.; Butler, J. "Rootkits: Subverting the Windows Kernel"; 1st, 2005.

[17] Balachandran, V.; Emmanuel, S. "Potent and Stealthy Control Flow Obfuscation by Stack Based Self-Modifying Code"; IEEE Trans. on Information Forensics and Security 2013, 8, 669 – 681.

[18] Mavrogiannopoulos, N.; Kisserli, N.; Preneel, B. "A taxonomy of Self-Modifying Code for Obfuscation"; Computer & Security Katholieke Universiteit Leuven, Belgium, 2010, 679-691.

[19] Berdajs J.; Bosnić, Z. "Extending Applications Using an Advanced Approach to Dll Injection and Api Hooking"; Software: Practice and Experience 2010, 40, 567-584.

[20] Gandotra, E.; Bansal, D.; Sofat, S., "Malware Analysis and Classification: A Survey."; J. of Information Security 56-64, 5, 2014.

[21] Sikorski, M.; Honig, A. "Practical Malware Analysis"; 1st, 2012.

[22] Salmani Balu, A.; Lazemi, S.; Parsa, S. "Disinfect Infector Viruses with PE Header"; In Proc. of Sofware Security, Shiraz Univ., Iran, 2014, 97-102. (In Persian)

[23] Javaheri D.; Parsa S. "Protection of Operation System against Spywares and Their Diversion"; J. Passive Defence Sci. Tech., 2014, 5, 171-181. (In Persian)

[24] Chen C.M.; Wu M.E.; He B.Z.; Zheng X.; Hsing C.; Sun H.M. "A Methodology for Hook-Based Kernel Level Rootkits."; Shenzhen Graduate School, Shenzhen, China, 2014.

[25] Blunden, A. "The Rootkit Arsenal"; 2nd, 2012.

[26] Priyadarshi, S. "Metamorphic Detection via Emulation"; Master's Thesis, Jose State Univ. 2011.

[27] Gouran Orimi, A. "Provide an Optimal and Transparent Framework for Automatic Analysis of Malware"; dsds, M.Sc. Thesis, Iran Univ. of Sci. and Tech., Tehran, 2014 (In Persian).

[28] Desai, P. "A Highly Metamorphic Virus Generator"; Int. J. Multimedia Intelligence and Security 2010, 1, 402-427.

[29] "Daniel Javaheri Personal Website"; http://djavaheri.ir/downlads/td4.rar.