

## ارائه یک راه کار موثر برای تشخیص بدافزارهای آگاه به محیط مبتنی بر مقایسه تفاوت‌های رفتاری

سیروس قاسمی<sup>۱</sup>، سعید پارسا<sup>۲\*</sup>

۱- دانشجوی کارشناسی ارشد، دانشگاه آزاد اسلامی، واحد علوم و تحقیقات، گروه مهندسی کامپیوتر، تهران، ایران.

۲- دانشیار، دانشگاه علم و صنعت ایران، دانشکده کامپیوتر، تهران، ایران.

(دریافت: ۱۳۹۶/۱۰/۲۲، پذیرش: ۱۳۹۷/۰۳/۰۶)

### چکیده

با توجه به ناکارآمدی روش‌های تحلیل ایستا به واسطه روش‌های بدافزاری نظیر چندریختی، دگرریختی و مبهم‌سازی کد و کدخود تصحیح، روش‌های تحلیل پویا و مکاشفه‌ای که اساساً مبتنی بر تحلیل رفتار زمان اجرای بدافزار هستند، از اهمیت ویژه‌ای برخوردار شده‌اند. پیدایش بدافزارهای آگاه به محیط، که با به‌کارگیری روش‌های ضدتحلیلی پویا سعی در پنهان‌سازی رفتار بدخواهانه خود در صورت تشخیص محیط‌ها و ابزارهای تحلیل دارند، در عمل روش‌های تشخیص پویای بدافزار را با مشکل مواجه نموده است. با در نظرگیری دوگانگی رفتار چنین بدافزارهایی، در این تحقیق راه‌کاری موثر با هدف تشخیص بدافزارهای آگاه به محیط ارائه شده است. این روش مبتنی بر پایش فراخوانی‌های سیستمی نمونه‌های بدخواه و بی‌خطر تحت دو نرم‌افزار NtTrace و drstrace با روش‌های متفاوت پایش و محاسبه فاصله رفتاری حاصل، برای گردآوری داده‌ها جهت ایجاد مدلی برای شناسایی این دسته از بدافزارها است. نهایتاً یک دسته‌بند ماشین‌بردار پشتیبان، با یادگیری مجموعه داده‌ی آموزش متشکل از بدافزارهای آگاه به محیط و نرم‌افزارهای بی‌خطر، با روش اعتبارسنجی متقابل و جستجوی گرید با قابلیت تشخیص این نوع بدافزارها با میانگین دقت، یادآوری و صحت قابل توجه تا حد ۱۰۰٪، ارائه می‌شود. در حالی که ارزیابی‌های انجام شده در کار مرتبط قبلی میانگین دقت، یادآوری و صحت را به ترتیب ۹۶/۵۸٪، ۹۵/۶۸٪ و ۹۶/۱۲۵٪ نشان می‌دهد.

**واژه‌های کلیدی:** بدافزارهای آگاه به محیط، روش‌های ضدتحلیلی، فراخوانی سیستمی، فاصله رفتاری، ماشین‌بردار پشتیبان

### ۱- مقدمه

رفتار زمان اجرای بدافزارها، هستند. این روش‌ها علاوه بر مقاوم بودن نسبت به راه‌کارهای بدافزاری ذکر شده، برای شناسایی بدافزارهای از پیش دیده نشده نیز سودمند هستند [۳]. بدافزارها برای بی‌نتیجه‌گذاشتن این روش‌های تحلیلی از روش‌هایی با عنوان روش‌های ضدتحلیلی<sup>۵</sup> استفاده می‌کنند. این روش‌ها شامل جستجوی کلیدهای ثبت‌کننده یا در اصطلاح رجیستری خاص، بررسی ثبات‌های پردازنده، اجرای دستورالعمل‌های خاص، فراخوانی توابع سیستمی با آرگومان‌های معین، بررسی مدت زمان اجرای دستورالعمل‌ها با خواندن شمارنده‌های مهر زمانی<sup>۶</sup> پیش و پس از اجرا، استفاده از اشکالات<sup>۷</sup> شبیه‌سازها و غیره، هستند [۴-۵]. بدافزارهایی که از روش‌های ضدتحلیلی برای دورزدن روش‌های تحلیلی مبتنی بر رفتار زمان اجرا، بهره می‌برند،

به منظور تحلیل و شناسایی بدافزارها روش‌های تحلیلی متعددی، از سوی محققان بدافزار ارائه شده‌اند. دسته‌ای از این روش‌های تحلیلی، که مبتنی بر الگو و امضا هستند نسبت به راه‌کارهای به‌کار گرفته‌شده توسط مولفان بدافزار نظیر چندریختی<sup>۱</sup>، دگرریختی<sup>۲</sup>، بسته‌بندی و مبهم‌سازی<sup>۳</sup> کد و کدخود تصحیح<sup>۴</sup>، که با هدف ایجاد موانع و پیچیدگی‌هایی در مسیر تحلیل و شناسایی بدافزار استفاده می‌شوند، آسیب‌پذیر هستند [۱-۲]. دسته دیگری از روش‌های تحلیلی که برتری‌هایی در مقایسه با روش‌های مبتنی بر امضا دارند، روش‌های تحلیلی پویای مبتنی بر

\* رایانامه نویسنده مسئول: parsaa@iust.ac.ir

5- Anti-Analysis  
6- Time Stamp Counter(TSC)  
7- Bugs

1- Polymorphism  
2- Metamorphism  
3- Obfuscation  
4- Self-modifying code

تحقیق می‌توان به موارد زیر اشاره نمود:

- برای مشاهده‌ی رفتار بدافزار تنها نیاز به دو بار اجرای آن در محیط مجازی است. هر اجرا در یک محیط مجازی که تاکنون بدافزاری روی آن اجرا نشده، تحت یک نرم‌افزار پایش ای‌پی‌ای‌های محلی<sup>۱۰</sup> ویندوز انجام می‌شود. بعد از هر اجرا با استفاده از تصویر لحظه‌ای<sup>۱۱</sup>، ماشین مجازی به وضعیت پیش از اجرای بدافزار برمی‌گردد (مرحله اول راه‌کار پیشنهادی).
- از آنجایی که یکی از روش‌های ضدتحلیلی که بدافزار برای شناسایی ابزارها و محیط‌های تحلیل شناخته‌شده و پرکاربرد، نظیر virtualBox، sandboxie، wireshark و غیره، به کار می‌برد، جستجو در کلیدهای رجیستری و بررسی برنامه‌های در حال اجرا روی سیستم است، استفاده از دو ابزار پایش فراخوانی‌های سیستمی NtTrace و drstrace که کم‌تر رایج و شناخته شده هستند، احتمال این که بدافزار با وجود ابزارهای فراوان و متنوع رایج برای تحلیل، وجود این دو ابزار را بررسی کند، کاهش داده و در نتیجه می‌تواند این روش ضدتحلیلی را بی‌نتیجه بگذارد (مرحله اول راه‌کار پیشنهادی).
- هر دو نرم‌افزار پایش فراخوانی‌های سیستمی، ای‌پی‌ای‌های محلی ویندوز را ثبت می‌کنند. دلیل پایش فراخوانی‌های سیستمی در این سطح از سیستم‌عامل به این شرح است. ای‌پی‌ای محلی ویندوز به‌عنوان واسطی، بین ای‌پی‌ای‌های ویندوز و فراخوانی سیستمی قرار گرفته است و معمولاً از ای‌پی‌ای‌های سطح بالاتر، مانند ای‌پی‌ای‌های Win32، برای فراخوانی‌های سیستمی و انجام پیش‌پردازش‌ها و پس‌پردازش‌های ضروری آرگومان‌ها یا نتایج، فراخوانی می‌شود. برنامه‌های کاربردی قانونی معمولاً از طریق ای‌پی‌ای ویندوز با سیستم‌عامل ارتباط برقرار می‌کنند. اما ممکن است کدمخرب از این لایه به‌گریزد و مستقیماً با ای‌پی‌ای محلی تعامل کند، تا راه‌حل‌های مبتنی بر پایش ای‌پی‌ای که تنها ای‌پی‌ای‌های Win32 را به قلاب می‌اندازند<sup>۱۲</sup>، بی‌نتیجه بگذارند. از این‌رو، با پایش ای‌پی‌ای‌های محلی می‌توان چنین بدافزارهایی را نیز به دام انداخت [۱۰] (مرحله اول راه‌کار پیشنهادی).
- دو نرم‌افزار پایش استفاده شده در این تحقیق از دو روش متفاوت برای ثبت ای‌پی‌ای‌های محلی استفاده می‌کنند. به‌همین دلیل چنانچه بدافزار یکی از این دو روش یا هر دو

اصطلاحاً بدافزارهای آگاه به محیط نامیده می‌شوند. از این بدافزارها در تحقیقات مختلف، تحت عناوین دیگری از جمله بدافزارهای حساس به محیط، مقاوم در برابر محیط و شخصیت دوگانه نیز یاد می‌شود [۳]. این نوع بدافزارها پیش از اجرای کد مخرب خود از روش‌های ضدتحلیلی با هدف بررسی محیط اجرا و پنهان نمودن رفتار مخرب، در صورت حضور در یک محیط تحلیل، مانند ماشین‌های مجازی، جعبه‌های شن یا اشکال‌زدها، استفاده می‌کنند. بدافزار در صورت تشخیص حضور در یک محیط تحلیلی به منظور پنهان نمودن رفتار مخرب خود یا بلافاصله به اجرای خود پایان می‌دهد یا رفتاری مشابه نرم‌افزارهای بی‌خطر از خود بروز می‌دهد. اما چنانچه اجرا در محیطی فاقد ابزارهای تحلیلی صورت پذیرد یا بدافزار موفق به شناسایی ابزارهای تحلیلی موجود نشود، رفتار مخرب خود را بروز می‌دهد. این دوگانگی در رفتار این بدافزارها، که از آن در تحقیقات با عنوان تفاوت یا فاصله رفتاری<sup>۱</sup> یاد می‌شود [۶-۷]، قابل اندازه‌گیری است و می‌تواند عاملی مهم در تشخیص این خانواده از بدافزارها باشد. هدف این مقاله ارائه روشی کارا و نوین جهت شناسایی بدافزارهای آگاه به محیط براساس رفتار زمان اجرای آن‌ها در قالب دنباله فراخوانی‌های سیستمی<sup>۲</sup> است. راه‌کار پیشنهادی شامل پایش فراخوانی‌های سیستمی یک نمونه‌ی مشکوک تحت دو نرم‌افزار NtTrace [۸] و drstrace [۹] با دو روش متفاوت در دو اجرای مجزا و سپس محاسبه فاصله رفتاری نمونه مشکوک در دو اجرا با استفاده از الگوریتم طولانی‌ترین زیردنباله مشترک<sup>۳</sup> و نهایتاً ارائه یک مدل با استفاده از ماشین‌های بردار پشتیبان<sup>۴</sup> برای دسته‌بندی<sup>۵</sup> نمونه بدافزارهای آگاه به محیط و نرم‌افزارهای بی‌خطر، برای شناسایی این نوع از بدافزارها است. نتایج به دست آمده از آزمون راهکار ارائه شده دقت<sup>۶</sup>، یادآوری<sup>۷</sup> و صحت<sup>۸</sup> تا حد ۱۰۰٪ در تشخیص بدافزارهای آگاه به محیط را نشان می‌دهد که در مقایسه با راهکار ارائه شده در [۳] که یک مدل تصمیم‌گیری مبتنی بر یادگیری پرسپترون چند لایه<sup>۹</sup> برای دسته‌بندی یک نمونه براساس رفتار مخرب و واکنشی‌اش به محیط، است و به ترتیب دقت، یادآوری و صحتی برابر با ۹۶/۵۶٪، ۹۵/۶۸٪ و ۹۶/۱۲۵٪ را نشان می‌دهد، بهبود یافته است. از جمله نوآوری‌ها و مزایای راه‌کار پیشنهادی در این

1- Behavioral difference/distance

2- System call

3- Longest Common Subsequence (LCS)

4- Support Vector Machines(SVM)

5- Classification

6- Precision

7- Recall

8- Accuracy

9- Multi-Layer Perceptron Learning

10- Native API

11- Snapshot

12- Hook

این نکته را نیز در نظر داشت که، بدافزار می تواند با نمایش رفتاری بی خطر از خود، به اجرایش ادامه داده و از تشخیص خود جلوگیری کند. در مرجع [۷] با این ایده که، بدافزار دارای قابلیت ضد ماشین مجازی، در محیط های مجازی و واقعی رفتاری متفاوت از خود بروز می دهد، روشی پیشنهاد شده که، اطلاعات رفتاری بدافزار را با استفاده از نرم افزار Process Monitor در دو محیط جمع آوری کرده و سپس به منظور تشخیص قابلیت ضد ماشین مجازی، با الگوریتم بهبود یافته Levenshtein تفاوت رفتار بدافزار را محاسبه می کند. به دلیل شناخته شده و رایج بودن نرم افزار استفاده شده برای پایش رفتار بدافزار، ممکن است بدافزار با بررسی برنامه های در حال اجرا روی سیستم و تشخیص آن، رفتار مخربش را پنهان کند. در مرجع [۳] یک مدل تصمیم گیری مبتنی بر یادگیری پرسپترون چند لایه با الگوریتم انتشار به عقب<sup>۲</sup>، برای دسته بندی یک نمونه براساس رفتار مخرب و واکنش اش به محیط، در یکی از چهار کلاس (infinite-running, guest-crashing, malignant, clean)، ارائه شده است. این روش چنانچه بدافزار دارای دو رفتار اجرای نامتناهی<sup>۳</sup> و شکست سیستم مهمان<sup>۴</sup> باشد، آن را به عنوان بدافزار آگاه به محیط در نظر می گیرد. باید این نکته را در نظر داشت که، بدافزار آگاه به محیط می تواند پس از شناسایی محیط تحلیل، رفتارهایی مانند پایان اجرا، یا بروز رفتاری بی خطر که لزوماً شامل تکرار در فراخوانی های سیستمی نباشد، از خود نشان دهد. در مرجع [۱۲] ابزاری با نام DISARM<sup>۵</sup>، با مقایسه رفتار بدافزار در چندین جعبه شن تحلیلی، بدافزاری که با تشخیص محیط تحلیل، از تحلیل می گریزد، شناسایی می کند. این ابزار در دو مرحله، کار می کند: مرحله پایش اجرا، با استفاده از Anubis<sup>۶</sup> و قطع فراخوانی سیستمی از داخل محیط ویندوز و مرحله مقایسه، که با مقایسه پروفایل های رفتاری بدافزار شامل تنها ویژگی هایی که مرتبط با تغییرات مانا در وضعیت سیستم هستند به علاوه ویژگی های نشان دهنده فعالیت شبکه، با استفاده از فاصله Jaccard انجام می شود. این روش نیاز به اجراهای متعدد روی چند جعبه شن مختلف دارد. از سوی دیگر، به دلیل شبیه سازی تنها یک زیرمجموعه از همه ی آی پی های ممکن ویندوز توسط ابزارهای تحلیل جعبه شن مانند Sandbox، علاوه بر آسیب پذیری نسبت به تشخیص از طریق استفاده از هر آی پی ای پیاده سازی نشده یا هر آی پی ای آی که به درستی شبیه سازی نشده، نتایج

را دور بزند، این تفاوت رفتار بدافزار در فراخوانی های سیستمی آن نمایان می شود و موجب تشخیص آن می شود (مرحله اول راه کار پیشنهادی).

- استفاده از الگوریتم طولانی ترین زیر دنباله مشترک به منظور محاسبه تفاوت دنباله فراخوانی های سیستمی (مرحله دوم راه کار پیشنهادی).
- استفاده از ماشین بردار پشتیبان برای دسته بندی نمونه بدافزارهای آگاه به محیط و نمونه نرم افزارهای بی خطر با نتایج به دست آمده از مقایسه تفاوت های رفتاری، می تواند روی تعداد نمونه های کم نیز نتایج قابل قبولی را نشان دهد [۱۱] (مرحله سوم راه کار پیشنهادی).
- استفاده از نمونه بدافزارهای آگاه به محیط از پیش شناخته شده در آموزش مدل SVM (مرحله سوم راه کار پیشنهادی).

در ادامه این مقاله در بخش ۲ راه کارهای موجود در زمینه تشخیص بدافزارهای آگاه به محیط و چالش های موجود در کارهای پیشین بیان می شوند. در بخش ۳، راه کار پیشنهادی در این تحقیق شرح داده می شود. در بخش ۴، ارزیابی راه کار پیشنهادی با استفاده از توابع ارائه شده توسط کتابخانه Scikit-learn روی بیش از ۱۰۰ نمونه بدافزار آگاه به محیط و نرم افزار بی خطر، ارائه می شود. نهایتاً در بخش ۵ نتیجه گیری و کارهای آتی بیان می گردند.

## ۲- کارهای مرتبط

به طور کلی کارهای انجام شده در زمینه تشخیص بدافزارهای آگاه به محیط، یا مبتنی بر تفاوت رفتاری بدافزار در اجراهای مختلف هستند، یا مبتنی بر جستجوی الگوها یا امضاهای از پیش یافت شده روش های شناسایی محیط و ابزارهای تحلیل، هستند. از جمله روش های تشخیص مبتنی بر تفاوت رفتاری بدافزار می توان به موارد زیر اشاره نمود:

در مرجع [۶] روش ارائه شده بدافزار دارای روش ضد اشکال زدایی را، با به قلاب انداختن یک آی پی آی ضد اشکال زدایی مشخص و تغییر مقدار بازگشتی آن، در دو حالت تحلیلی و غیر تحلیلی اجرا می کند. چنانچه بدافزار در حالت تحلیلی، با خودداری از اجرا، عمل گریز<sup>۱</sup> را انجام دهد، اما در حالت غیر تحلیلی رفتاری مخرب از خود در قالب اضافه نمودن یک کلید به رجیستری نشان دهد، تشخیص داده می شود. باید

2- Back Propagation  
3- infinite-running  
4- guest-crashing  
5- Detecting Sandbox-AwaRe Malware  
6- Analyzing Unknown Binaries

1- Evasion

ضد اشکال زدایی با وصله بندی<sup>۵</sup> قانونمند<sup>۶</sup> در فایل های دودویی با تحلیل ایستای دستورالعمل های اسمبلی ارائه شده است. به این صورت که بعد از شناسایی بخش هایی از کد شامل دستورالعمل های ضد اشکال زدایی، آن ها را با دستورالعمل های جدید وصله بندی می کند.

چالش های موجود در روش های کنونی مبتنی بر تفاوت رفتار عبارتند از: روش هایی که مبتنی بر پایش ای پی آی های Win32 هستند، نسبت به بدافزارهای آگاه به محیطی که این ای پی آی ها را دور می زنند و مستقیماً ای پی آی های محلی را فراخوانی می کنند، آسیب پذیر هستند. روش های دیگری نیز وجود دارند، که از دنباله دستورالعمل های ماشین فراخوانی شده توسط بدافزار، چه برای تشخیص تفاوت های رفتاری در محیط های مختلف تحلیل و چه برای تشخیص الگوهای شناسایی محیط تحلیل، استفاده می کنند. چنین روش هایی به دلیل حجم بالای خروجی های حاصل از پایش زمان بر هستند. روش هایی که با استفاده از اجراهای متعدد روی چندین محیط تحلیل پویا اقدام به شناسایی بدافزارهای آگاه به محیط می کنند، نیز دارای بار کاری زیادی هستند و زمان و هزینه زیادی را صرف اجرای این بدافزارها می کنند. برخی از روش های ارائه شده در زمینه شناسایی این بدافزارها تنها به شناسایی تعداد معدودی از روش های ضد تحلیلی اکتفا می کنند و نسبت به سایر روش های ضد تحلیلی آسیب پذیر هستند. همچنین روش هایی که متمرکز بر شناسایی الگوهای تشخیص محیط تحلیل هستند، از تشخیص روش های ضد تحلیلی جدید به کار گرفته شده توسط بدافزارها ناتوان هستند. در این تحقیق سعی شده است تا به برخی از این چالش ها پاسخ داده شود.

### ۳- راه کار پیشنهادی

راه کار پیشنهادی برای تشخیص بدافزارهای آگاه به محیط، مبتنی بر مقایسه تفاوت های رفتاری بدافزار در دو اجرای مجزا تحت پایش دو نرم افزار پایش فراخوانی های سیستمی که از نظر روش پایش متفاوتند، است. از آن جایی که رفتار یک برنامه از وضعیت سیستم عامل مشخص می شود و هیچ تغییری در وضعیت سیستم عامل بدون استفاده از فراخوانی های سیستمی نمی تواند انجام شود، بنابراین، می توان فراخوانی های سیستمی یک برنامه را که واسطی غیرقابل دور زدن هستند، به عنوان نمودار رفتار آن برنامه، در نظر گرفت [۳]. بنابراین، اندازه گیری فاصله رفتاری قبل از هر چیزی مستلزم تعیین مدل رفتاری در

به دست آمده از اجراهای یک نمونه، در چند جعبه شن، نیز می توانند به خودی خود متفاوت باشند [۴]. در مرجع [۱۳] برای مقابله با تاثیر قابل توجه عوامل نا مسلم<sup>۱</sup> مانند زمان سیستم، اعداد تصادفی و غیره بر اثربخشی تشخیص بدافزار آگاه به ماشین مجازی، طرح جدیدی با نام Divergence Detector پیشنهاد شده، که با چندین بار اجرای یک نمونه روی هر ماشین مجازی، عوامل نا مسلم را در سطح دستورالعمل کاهش می دهد و می تواند افتراق<sup>۲</sup> آثار چندین اجرا در ماشین های مجازی متنوع را تشخیص دهد. سربار ناشی از این روش به دلیل نیاز به چندین اجرا در هر ماشین مجازی برای یک نمونه بدافزار، بالا است. همچنین روش پیشنهادی تنها چند نوع از روش های تشخیص محیط مجازی به کار گرفته شده توسط بدافزارها را بررسی می کند و فقط روی چهار نمونه بدافزار آزمون شده است. در مرجع [۱۴] با هدف تشخیص بدافزار آگاه از ماشین مجازی<sup>۳</sup>، یک تعیین کننده محل واگرایی جدید مبتنی بر بلوک هایی از دستورالعمل ها به عنوان واحد تحلیل، ارائه شده است. این روش هر نمونه بدافزار را در دو محیط ماشین مجازی، که یکی از آن ها نسبت به بررسی های ماشین مجازی توسط بدافزار مقاوم تر است، اجرا می کند. در طول اجرا ثبت کننده بلوک پایه در دو محیط اجرا، آثار بلوک پایه را ثبت می کند. از آن جایی که هر دو محیط اجرا، محیط ماشین مجازی هستند، ممکن است روش های تشخیص ماشین مجازی در دو محیط موفق به تشخیص محیط اجرا شوند. در نتیجه تفاوت در دو محیط دیده نمی شود. در مرجع [۱۵] راه کاری برای استخراج الگوهای رفتاری مخرب موجود در هر خانواده از بدافزار با کاوش زیرگراف های مهم و تعیین زیرگراف های تفکیک پذیر ارائه شده است. این راه کار می تواند برای استخراج الگوهای مخرب بدافزارهای آگاه به محیط در محیط های اجرای متفاوت به کار گرفته شود.

از جمله روش های تشخیص مبتنی بر جستجوی الگوها یا امضای از پیش یافت شده روش های شناسایی محیط ها و ابزارهای تحلیل می توان به موارد زیر اشاره نمود:

در مرجع [۱۶] یک پلاگین IDA-Pro<sup>۴</sup> ارائه شده است، که با اسکن امضای دو گونه اصلی، از ترفندهای anti-VMware در مراحل مختلف اجرا، برای بی نتیجه گذاردن تشخیص های VMware، به طور خودکار نتیجه تشخیص VMWare را وصله می کند. در مرجع [۱۷] یک روش گریز از روش های

1- Uncertain Factors

2- Divergence

3- VM-Aware

4- Interactive Disassembler Pro

5- Patching

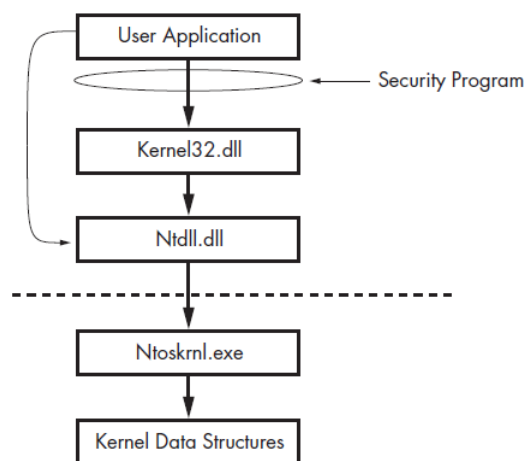
6- Rule-Based

فراخوانی شده در هر اجرا استخراج می شود. در مرحله دوم فاصله رفتاری هر نمونه، در دو اجراء، براساس بزرگترین زیردنباله مشترک از فراخوانی های سیستمی با استفاده از الگوریتم طولانی ترین زیردنباله مشترک محاسبه می شود. همچنین سایر ویژگی های رفتاری از دنباله های ای پی ای های فراخوانی شده استخراج می شوند. نهایتاً در مرحله سوم با استفاده از ماشین بردار پشتیبان، مدلی برای دسته بندی نمونه های مورد بررسی ساخته می شود. از این مدل برای تشخیص نمونه بدافزارهای جدید که ویژگی های رفتاری آن ها در دو مرحله قبل محاسبه شده، استفاده می گردد. در ادامه این بخش، هر کدام از مراحل به تفصیل شرح داده می شوند.

### ۳-۱ - مرحله اول راه کار پیشنهادی

در این مرحله فایل اجرایی نمونه بدافزار آگاه به محیط یا نرم افزار بی خطر با شرایط یکسان تحت دو نرم افزار پایش فراخوانی های سیستمی drstrace و NtTrace اجرا می شود. دلیل استفاده از این دو نرم افزار این است که اولاً این دو نرم افزار کم تر توسط تحلیل گران بدافزار استفاده شده اند، که باعث می شود یکی از چالش های موجود در راه کارهای پیشین یعنی بررسی فرآیندهای در حال اجرا در سیستم توسط بدافزار و شناسایی ابزارهای ثبت نگاری مرتفع شود، ثانیاً این دو نرم افزار از دو روش کاملاً متفاوت برای ثبت نگاری استفاده می کنند. NtTracce از حالت اشکال زدایی ویندوز برای ثبت ای پی ای های محلی فراخوانی شده استفاده می کند [۸]. drstrace ابزاری در چارچوب کاری Dr. Memory [۱۸] است، که آن نیز بر روی سکوی DynamoRIO ساخته شده است. DynamoRIO با قرارگیری بین نمونه مورد بررسی و سیستم عامل، به عنوان یک پردازشگر ماشین مجازی<sup>۲</sup> بر روی فرآیند یا پردازنده مورد نظر عمل می کند [۱۹]. این تفاوت در روش های ثبت نگاری سبب می شود، در صورتی که بدافزار یکی از این دو روش یا هر دو را دور بزند، این تفاوت رفتار بدافزار در دنباله فراخوانی های سیستمی آن آشکار گردد و تشخیص آن میسر شود. از سوی دیگر، به دلیل این که هر دو نرم افزار اقدام به ثبت فراخوانی های سیستمی یک نمونه در سطح ای پی ای های محلی ویندوز می کنند، امکان دور زدن این برای بدافزار بسیار دشوار می گردد. به منظور اجرای خودکار فایل اجرایی نمونه مورد بررسی از واسط خط فرمان ارائه شده برای دو نرم افزار، استفاده می شود. در شکل (۲) نمای کلی از مرحله اول راه کار پیشنهادی آورده شده است. پس از اجرای نمونه مورد

قالب دنباله فراخوانی های سیستمی در زمان اجرا است. از سوی دیگر ای پی ای های محلی یک واسط سطح پایین برای تعامل با ویندوز هستند که به ندرت توسط برنامه های غیر مخرب استفاده می شوند، اما در بین مولفان بدافزار رایج هستند. فراخوانی ای پی ای های محلی، ای پی ای های Win32 را دور می زند. فراخوانی ای پی ای های محلی به طور مستقیم برای نویسندگان بدافزار، به دلیل مخفیانه تر بودن و این که به آن ها امکان انجام چیزهایی را می دهد که ممکن است به صورت دیگر امکان پذیر نباشند، جذاب است. بسیاری از ضدویروس ها و محصولات مراقبت میزبان فراخوانی های سیستمی انجام شده از کتابخانه Win32 توسط یک فرآیند را پایش می کنند. از این رو فراخوانی ای پی ای های محلی به طور مستقیم ممکن است به تواند از یک محصول امنیتی با طراحی ضعیف مطابق شکل (۱) بگریزد [۱۰]. برای حل این مشکل، می توان از ابزارهایی برای ثبت نگاری<sup>۱</sup> فراخوانی های سیستمی در سطح ای پی ای های محلی استفاده نمود. در این تحقیق برای تعیین دنباله ای پی ای های محلی از دو نرم افزار با نام های NtTracce و drstrace که کم تر توسط تحلیل گران بدافزار استفاده شده اند و فراخوانی های سیستمی را در سطح ای پی ای های محلی ثبت می کنند، استفاده شده است.



شکل (۱): استفاده از ای پی ای های محلی برای اجتناب از شناسایی [۱۰]

راه کار پیشنهادی برای تشخیص بدافزارهای آگاه به محیط به سه مرحله تقسیم می شود. در مرحله اول فایل اجرایی نمونه بدافزارهای آگاه به محیط و نرم افزارهای بی خطر تحت دو نرم افزار پایش در شرایط یکسان اجرا و سپس دنباله ای پی ای های

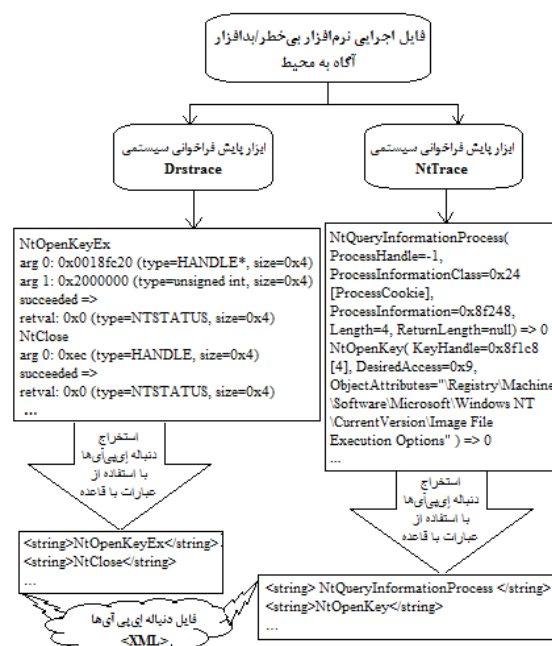
بردار پشتیبان استفاده می‌شود. با ساخت LCS طولانی‌ترین زیر دنباله مشترک از فراخوانی‌های سیستمی را می‌سازیم. ما از این زیر دنباله تعداد ای‌پی‌ای‌های محلی منحصر به فرد فراخوانی شده در هر اجرا را به عنوان ویژگی دیگری برای آموزش مدل SVM استفاده می‌کنیم. سایر ویژگی‌هایی که برای آموزش مدل SVM از دنباله فراخوانی‌های سیستمی اجراهای یک نمونه مورد بررسی، استخراج می‌کنیم، در زیر آورده شده‌اند:

- تعداد ای‌پی‌ای‌هایی که توسط هر یک از دو نرم‌افزار در دو اجرا ثبت شده‌اند.
- این ویژگی می‌تواند برای تشخیص بدافزارهای آگاه به محیطی که بلافاصله بعد از تشخیص محیط تحلیل به اجرای خود پایان می‌دهند، استفاده شود.
- تعداد ای‌پی‌ای‌های منحصر به فردی که توسط هر یک از دو ابزار در دو اجرا ثبت شده‌اند.
- این ویژگی می‌تواند برای تشخیص بدافزارهای آگاه به محیطی که بعد از تشخیص محیط تحلیل با دستورات تکراری بی‌خطر در حلقه‌های تکرار سعی در پنهان نمودن رفتار مخرب خود دارند، استفاده شود.
- تعداد ای‌پی‌ای‌های مشترک از نظر نوع که در هر دو دنباله ای‌پی‌ای موجود هستند.
- این ویژگی در واقع میزان مشابهت اجراهای یک نمونه در نوع فراخوانی‌های سیستمی را بیان می‌کند.

بعد از محاسبه ویژگی‌های رفتاری هر نمونه مورد بررسی، نتایج حاصل به همراه نوع نمونه مورد بررسی در یک فایل با نام آن نمونه در قالب xml ذخیره می‌شود. همچنین نتایج نمونه‌هایی که نوع آن‌ها مشخص است، در یک فایل مجزا ذخیره می‌گردند، تا در مرحله بعد برای آموزش مدل SVM استفاده شوند. شکل (۳) نمای کلی از مرحله دوم راه‌کار پیشنهادی را نشان می‌دهد.

نوع نمونه مورد بررسی یکی از موارد بدافزار آگاه به محیط یا نرم‌افزار غیرآگاه به محیط یا ناشناخته خواهد بود. نرم‌افزارهای بی‌خطر و بدافزارهای فاقد روش‌های ضدتحلیلی در دسته نرم‌افزارهای غیرآگاه به محیط ذخیره می‌شوند. دلیل این کار این است که هدف ما در این تحقیق تشخیص بدافزارهای آگاه به محیط از سایر بدافزارها و نرم‌افزارها است، تا به توانیم از تشخیص نادرست آن‌ها در تحلیل پویا جلوگیری کنیم. برای تشخیص سایر بدافزارها می‌توان از روش‌های بسیاری که برای تشخیص انواع بدافزار ارائه شده‌اند، استفاده نمود که مربوط به هدف این تحقیق نیست. برای نمونه‌هایی که نوع آن‌ها مشخص نیست، نوع ناشناخته ذخیره می‌گردد، تا بعداً با استفاده از مدل

بررسی، فایل‌های رخداد<sup>۱</sup> ثبت شده خروجی دو ابزار به طور خودکار خوانده می‌شوند و سپس با توجه به قالب فایل رخداد خروجی هر ابزار و با استفاده از عبارت‌های با قاعده<sup>۲</sup>، دنباله ای‌پی‌ای‌های فراخوانی شده، استخراج می‌شوند. نهایتاً دنباله ای‌پی‌ای‌های استخراج شده، در یک فایل با نام نمونه مورد بررسی در قالب xml برای پردازش‌های بعدی ذخیره می‌شود.



شکل (۲): نمای کلی مرحله اول راه‌کار پیشنهادی

### ۳-۲- مرحله دوم راه‌کار پیشنهادی

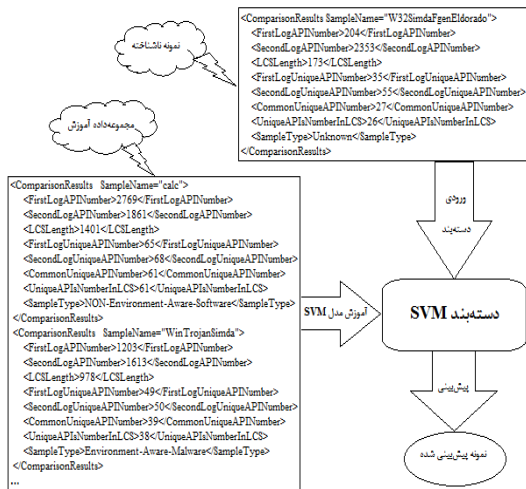
برای محاسبه فاصله رفتاری بین دو اجرای یک نمونه، می‌توانیم میزان شباهت دنباله فراخوانی‌های سیستمی دو اجرا را با استفاده از الگوریتم‌های ارائه شده برای یافتن شباهت متن<sup>۳</sup> مانند LCS، Jaro، Damerau-Levenshtein، N-gram، و غیره بیابیم [۲۰]. ما در این تحقیق از الگوریتم طولانی‌ترین زیردنباله مشترک (LCS) [۲۱] برای محاسبه تفاوت رفتاری در دو اجرای یک نمونه استفاده کرده‌ایم. برای محاسبه فاصله رفتاری یک نمونه در دو اجرا فایل‌های حاوی دنباله فراخوانی‌های سیستمی برای آن نمونه را می‌خوانیم و آن‌ها را به عنوان ورودی به الگوریتم طولانی‌ترین زیردنباله مشترک می‌دهیم. این الگوریتم طول طولانی‌ترین زیردنباله مشترک از فراخوانی‌های سیستمی را به ما می‌دهد. این مقدار، میزان مشابهت دو دنباله فراخوانی‌های سیستمی اجراها را بیان می‌کند و به عنوان یکی از ویژگی‌ها برای آموزش مدل ماشین

1- Log File  
2- Regular Expression  
3- Text Similarity

SVM ارائه شده نوع آن‌ها پیش‌بینی شود.

۳-۳- مرحله سوم راه کار پیشنهادی

به منظور دسته‌بندی نمونه نرم‌افزارهای بی‌خطر و بدافزارهای آگاه به محیط مورد آزمون، پارامترهای استخراج شده از پایش فراخوانی‌های سیستمی آن‌ها را به عنوان داده‌های آموزش به ماشین بردار پشتیبان می‌دهیم. با ارائه یک مدل، نوع نمونه نرم‌افزار از پیش دیده نشده را پس از انجام مرحله یک و دو برای آن نمونه، پیش‌بینی می‌کنیم. در شکل (۵) نمای کلی مرحله سوم راه کار پیشنهادی آورده شده است.



شکل (۵). نمای کلی مرحله سوم راه کار پیشنهادی

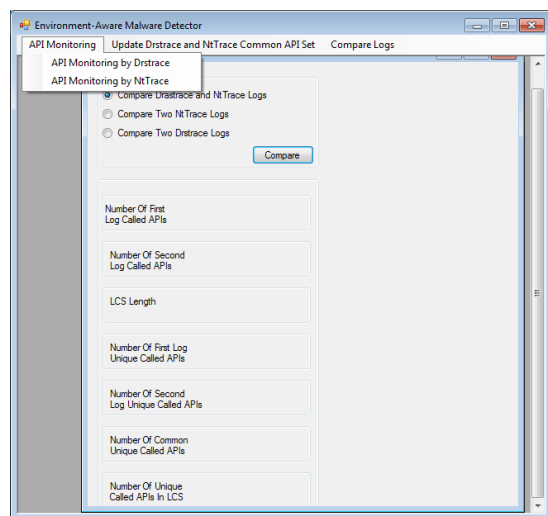
به منظور ساخت مدل ماشین بردار پشتیبان از فایل حاوی نتایج محاسبات فاصله رفتاری اجراها، برای آموزش این مدل استفاده می‌کنیم. برای ساخت دسته‌بندی ماشین بردار پشتیبان از کتابخانه Scikit-Learn در پایتون که شامل ابزارهای کارا و ساده برای داده‌کاوی و تحلیل داده‌ها است، استفاده می‌کنیم. در این تحقیق از نسخه 0.19.0 کتابخانه Scikit-Learn که آخرین نسخه ارائه‌شده این کتابخانه تا زمان نگارش این تحقیق است. همچنین از پایتون نسخه 2.7 برای به‌کارگیری این کتابخانه استفاده می‌شود. ما در این تحقیق از کلاس SVC ارائه‌شده در ماژول SVM که در آن دسته‌بندی چند کلاسه براساس شمای یک-در برابر-یک انجام می‌شود، استفاده می‌کنیم. پارامترهای مهم این کلاس عبارتند از: پارامتر C، کرنل، گاما. همانند سایر دسته‌بندها، SVC نیز دو آرایه به‌عنوان ورودی می‌گیرد: یک آرایه X با اندازه [n-ویژگی، n-نمونه] که نتایج محاسبات فاصله‌های رفتاری اجراها برای نمونه‌های بررسی‌شده را به‌عنوان نمونه‌های آموزشی نگه می‌دارد و یک آرایه y که شامل نوع هر نمونه

- 1- Predict
- 2- Classifier
- 3- One-VS-One



شکل (۳): نمای کلی مرحله دوم راه کار پیشنهادی

مرحله اول و دوم راه کار ارائه شده برای تشخیص بدافزارهای آگاه به محیط، در نرم‌افزاری به نام EnvironmentAwareMalwareDetector در سکوی NET 4.6.1 Framework، با زبان برنامه‌نویسی #C توسعه داده شده است که در شکل (۴) نمایی از واسط کاربر آن آورده شده است.



شکل (۴): واسط کاربر نرم‌افزار EnvironmentAwareMalwareDetector



مرتبط در این زمینه را بررسی کرده و نمونه بدافزارهای آگاه به محیط که در آن مقالات معرفی یا استفاده شده بودند، را جمع‌آوری کردیم. در مرحله بعد با جستجوی این بدافزارها در سایت‌های ارائه‌کننده فایل بدافزارها، مانند openmalware.org، DasMalwerk و VirusShare، thezoo.morirt.com، amalwr.com فایل اجرایی آن‌ها را دانلود کردیم. در جدول (۲) برخی از خانواده‌های بدافزار آگاه به محیط استفاده شده در این تحقیق آورده شده‌اند. بدافزارهای آماده شده همگی توسط شرکت‌های آنتی ویروس مشهور تحلیل و نوع آن‌ها مشخص شده‌اند. پس از آماده‌سازی مجموعه داده، از آن برای آموزش و آزمون دسته‌بند ماشین بردار پشتیبان استفاده کردیم.

**جدول (۲):** خانواده بدافزارهای آگاه به محیط استفاده شده در ارزیابی

خانواده بدافزار آگاه به محیط	نوع بدافزار
Simda	Trojan
Agobot	Backdoor, Worm
Autoit	Trojan, Worm
Carberp	Trojan
Kryptik	Trojan
Reptile	Trojan

#### ۴-۲- آموزش مدل ماشین بردار پشتیبان

همان‌طور که در مرحله سوم راه‌کار پیشنهادی بیان شد، در این تحقیق ما از کلاس SVC در کتابخانه Scikit-Learn برای ساخت یک دسته‌بند ماشین بردار پشتیبان استفاده می‌کنیم. به منظور بهینه‌سازی مدل ماشین بردار پشتیبان، از رویه ارائه شده در مرجع [۲۵] استفاده می‌کنیم. این رویه شامل گام‌های زیر است:

- تبدیل داده‌ها به قالب یک بسته ماشین بردار پشتیبان: کلاس SVC دو آرایه به عنوان ورودی می‌گیرد، یک آرایه  $X$  با اندازه  $[n\text{-sample}, n\text{-feature}]$  که نمونه‌های آموزشی را نگه می‌دارد و یک آرایه  $y$  که دربردارنده برچسب‌های کلاس با اندازه  $[n\text{-sample}]$ .
- انجام مقیاس‌گذاری ساده روی داده‌ها: به منظور مقیاس‌گذاری داده‌ها، از کلاس `sklearn.preprocessing.StandardScaler` در `scikit-learn` که ویژگی‌ها را با حذف میانگین و مقیاس‌گذاری به واریانس واحد استانداردسازی می‌کند، برای مقیاس‌گذاری هم داده‌های آموزش و هم داده‌های آزمون استفاده می‌کنیم [۲۶].

بررسی شده به‌عنوان برچسب‌های کلاس (رشته یا اعداد صحیح) با اندازه  $[n\text{-نمونه}]$  است. بعد از برآزش<sup>۱</sup> مدل، می‌توان برای پیش‌بینی مقادیر جدید از آن استفاده کرد [۲۳-۲۲].

#### ۴-ارزیابی

به‌منظور ارزیابی راه‌کار پیشنهادی مجموعه داده‌ای متشکل از فایل‌های اجرایی بدافزارهای آگاه به محیط و نرم‌افزارهای بی‌خطر را برای آموزش و آزمون مدل ماشین بردار پشتیبان فراهم می‌کنیم. سپس داده‌ها را با استفاده از برنامه کمکی `train_test_split` [۲۴]، به یک مجموعه توسعه<sup>۲</sup> که به نمونه کلاس `GridSearchCV` داده می‌شود و یک مجموعه ارزیابی برای محاسبه متریک‌های کارایی، تقسیم می‌کنیم. ساخت مدل ماشین بردار پشتیبان با پارامترهای بهینه با استفاده از روش اعتبارسنجی متقابل و جستجوی گرید روی مجموعه توسعه که شامل ۷۰٪ نمونه‌ها است، انجام می‌شود. در نهایت، مدل ماشین بردار پشتیبان روی مجموعه ارزیابی که شامل ۳۰٪ باقی‌مانده نمونه‌های موجود در مجموعه داده‌ها که در فرآیند آموزش مدل ماشین بردار پشتیبان شرکت داده نشده‌اند، ارزیابی می‌شود.

#### ۴-۱- آماده‌سازی مجموعه داده‌ها

ما در مجموع، از بیش از صد فایل اجرایی بدافزارهای آگاه به محیط و نرم‌افزارهای بی‌خطر برای ارزیابی راه‌کار پیشنهادی استفاده کرده‌ایم. فایل‌های اجرایی نرم‌افزارهای بی‌خطر را، از دایرکتوری `C:\Windows\System32` و نرم‌افزارهای شناخته شده جمع‌آوری کرده‌ایم. در جدول (۱) تعدادی از این نرم‌افزارها آورده شده‌اند.

**جدول (۱):** نرم‌افزارهای بی‌خطر استفاده شده در ارزیابی

SystemPropertiesComputerName	eventvwr	msinfo32
SystemPropertiesPerformance	taskmgr	NetProj
UserAccountControlSettings	magnify	Compact
SystemPropertiesProtection	mspaint	Colorcpl
SystemPropertiesHardware	charmap	regedt32
SystemPropertiesRemote	notepad	ETDMag
ComputerDefaults	getmac	SndVol
SnippingTool	chkdsk	Resmon
PrintBrmUi	printui	Cttune
powershell	control	Osk
NAPSTAT	msra	Calc
StikyNot	comp	Cmd

به منظور جمع‌آوری بدافزارهای آگاه به محیط، ابتدا مقالات

1 - Fitting  
2 - Development Set



شده براساس پرسپترون چندلایه نشان می دهد، بیانگر بهبود پارامترهای ارزیابی در راه کار پیشنهادی است.

## ۵- نتیجه گیری

با در نظرگیری آسیب پذیری روش های تحلیل ایستا در برابر بدافزارهایی که بدون تغییر در منطق کدشان، تغییر شکل می دهند، به کارگیری راه کارهایی برای تحلیل بدافزار که روی رفتار زمان اجرای بدافزار تمرکز می کنند، نظیر روش های تحلیل پویا و مکاشفه ای، امری اجتناب ناپذیر است. با این حال اساساً چنین روش هایی نیز در برابر بدافزارهایی با عنوان بدافزارهای آگاه به محیط که از روش های ضدتحلیلی پویا برای پنهان سازی رفتار مخرب خود در صورت تشخیص محیطها و ابزارهای تحلیل استفاده می کنند، آسیب پذیر هستند. از این رو با هدف رفع این آسیب پذیری و تشخیص بدافزارهای آگاه به محیط و باتوجه به ماهیت رفتار دوگانه چنین بدافزارهایی، در این تحقیق، راه کاری موثر مبتنی بر پایش فراخوانی های سیستمی، با استفاده از دو نرم افزار پایش فراخوانی های سیستمی با روش های متفاوت پایش به نام های NtTracce و drstrace و محاسبه فاصله رفتاری حاصل از دو اجرای یک نمونه ارائه شده است. این روش با پایش فراخوانی های سیستمی نمونه های بدخواه و بی خطر و محاسبه فاصله رفتاری حاصل از دو اجرای هر نمونه، یک دسته بند ماشین بردار پشتیبان را با استفاده از روش اعتبارسنجی متقابل و جستجوی گرید با آموزش روی ۷۰٪ داده های حاصل، ارائه می کند. در نهایت این دسته بند، با ارزیابی روی ۳۰٪ باقی مانده داده ها میانگین دقت، یادآوری و صحت تا حد ۱۰۰٪، را نشان می دهد، که در مقایسه با کار مرتبط در این زمینه با میانگین دقت، یادآوری و صحت به ترتیب ۹۶/۵۸٪، ۹۵/۶۸٪ و ۹۶/۱۲۵٪، بهبود را نشان می دهد.

به عنوان کارهای آتی پیشنهاد می شود که از ابزارهای پایش توابع Win32 نظیر WinAPIOverride [۳۲]، Process Monitor و غیره استفاده نمود. می توان با نداشت بین توابع Win32 و ای پی آی های محلی تفاوت رفتاری بدافزارهای آگاه به محیط را، در قالب فراخوانی های سیستمی آنها که از دو لایه متفاوت از سیستم عامل صورت می گیرد، آشکار نمود. همچنین پیشنهاد می گردد که با داده کاوی نتایج حاصل از پایش این گروه از بدافزارها و استخراج الگوهای استفاده شده که در بدافزارهای مختلف تکرار می شود، روش های جدید به کارگرفته شده برای تشخیص محیط تحلیل را شناسایی و در ادامه محیط های تحلیل را نسبت به آنها مقاوم نمود.

- انتخاب مدل: کرنل خطی و RBF، را در نظر می گیریم.
- به کارگیری روش اعتبارسنجی متقابل برای یافتن بهترین پارامتر C و گاما: از کلاس sklearn.model-selection.GridSearchCV که یک جستجوی جامع<sup>۱</sup> روی مقادیر تعیین شده پارامتر برای یک برآوردگر<sup>۲</sup> انجام می دهد، استفاده می کنیم [۲۸-۲۷].
- آزمون مدل

## ۴-۳- آزمون مدل ماشین بردار پشتیبان

پارامترهای امتیازدهی استفاده شده برای ارزیابی مدل ارائه شده عبارتند از:

- دقت: دقت بیانگر توانایی دسته بند است، در این که یک نمونه منفی را به عنوان مثبت برچسب گذاری نکند. بهترین مقدار دقت یک و بدترین مقدار آن صفر است. برای محاسبه دقت از رابطه (۱) که در آن،  $tp$  تعداد صحیح مثبت<sup>۳</sup> و  $fp$  تعداد غلط مثبت<sup>۴</sup> است، استفاده می شود [۲۹].

$$\text{Precision} = \frac{tp}{tp+fp} \quad (1)$$

- یادآوری: یادآوری، توانایی دسته بند در یافتن همه نمونه های مثبت را بیان می کند. بهترین مقدار یادآوری یک و بدترین مقدار آن صفر است. برای محاسبه یادآوری از رابطه (۲) که در آن،  $tp$  تعداد درست های مثبت و  $fn$  تعداد غلط های منفی<sup>۵</sup> است، استفاده می شود [۳۰].

$$\text{Recall} = \frac{tp}{tp+fn} \quad (2)$$

- صحت: کسر نمونه هایی که به درستی دسته بندی شده اند، را بیان می کند. بهترین مقدار صحت یک و بدترین مقدار آن صفر است [۳۱].

مدل حاصل روی ۳۰٪ از نمونه ها که در فرآیند آموزش مدل کنار گذاشته شده اند، مورد آزمون قرار گرفته و میانگین دقت، میانگین یادآوری و میانگین صحت آن، هر کدام تا حد ۱۰۰٪ است. مقایسه راه کار پیشنهادی با راه کار ارائه شده در مرجع [۳]، که به ترتیب میانگین دقت، میانگین یادآوری و میانگین صحت را ۹۶/۵۶٪، ۹۵/۶۸٪ و ۹۶/۱۲۵٪ برای مدل تصمیم گیری ساخته

1- Exhaustive Search  
2- Estimator  
3- True Positive  
4- False Positive  
5- False Negative

## ۶- منابع

- Points with Basic Block Comparison,” in Software Security and Reliability, Eighth International Conference, pp. 196-205, 2014.
- [15] S. Parsa, H. Saifi, M. H. Alaeian, “Providing a New Approach to Discovering Malware Behavioral Pattern Based on the Dependency Graph Between System Calls,” in Journal Of Electronical & Cyber Defence, vol. 4, no. 3, 2016. (In Persian)
- [16] L. Sun, T. Ebringer, and S. Boztas, “An automatic anti-anti-VMware technique applicable for multi-stage packed malware,” in Malicious and Unwanted Software. MALWARE, 3rd International Conference on, pp. 17-23, 2008.
- [17] J. Lee, B. Kang, and E. G. Im, “Evading anti-debugging techniques with binary substitution,” International Journal of Security & its Applications, vol. 8, no.1, pp.183-192, 2014.
- [18] “Dr. Memory,” [Online] Available: <http://drmemory.org>, 2017.
- [19] D. Bruening, “Efficient, Transparent, and Comprehensive Runtime Code Manipulation,” Ph.D. Thesis, MIT, September 2004.
- [20] W. H. Gomaa and A. A. Fahmy, “A survey of text similarity approaches,” International Journal of Computer Applications, vol. 68, pp. 13-18, 2013.
- [21] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, “Introduction to algorithms,” MIT press, 3<sup>rd</sup> Edition, pp. 390-396, 2009.
- [22] L. Buitinck, et al., “API design for machine learning software: experiences from the scikit-learn project,” ECML PKDD Workshop: Languages for Data Mining and Machine Learning, pp. 108-122, 2013.
- [23] F. Pedregosa, et al., “Scikit-learn: Machine Learning in Python,” Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.
- [24] “sklearn.model\_selection.GridSearchCV,” [Online] Available: [http://scikit-learn.org/stable/modules/generated/sklearn.model\\_selection.train\\_test\\_split.html](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html), 2017.
- [25] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, “A practical guide to support vector classification,” [Online] Available: <https://www.csie.ntu.edu.tw/~cjlin>, 2016.
- [26] “sklearn.preprocessing.StandardScaler,” [Online] Available: <http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>, 2017.
- [27] “sklearn.model\_selection.GridSearchCV,” [Online] Available: [http://scikit-learn.org/stable/modules/generated/sklearn.model\\_selection.GridSearchCV.html](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html), 2017.
- [28] “Tuning the hyper-parameters of an estimator,” [Online] Available: [http://scikit-learn.org/stable/modules/grid\\_search.html](http://scikit-learn.org/stable/modules/grid_search.html), 2017.
- [29] “sklearn.model\_selection.GridSearchCV,” [Online] Available: [http://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision\\_score.html](http://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html), 2017.
- [1] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, “A survey on automated dynamic malware-analysis techniques and tools,” ACM computing surveys (CSUR), vol. 44, p. 6, 2012.
- [2] A. Jadhav, D. Vidyarthi, and M. Hemavathy, “Evolution of evasive malwares: A survey,” in International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT), 2016
- [3] S. Naval, V. Laxmi, M. S. Gaur, S. Raja, M. Rajarajan, and M. Conti, “Environment-Responsive Malware Behavior: Detection and Categorization,” in Data Privacy Management, Autonomous Spontaneous Security, and Security Assurance, ed: Springer, pp. 167-182, 2015.
- [4] P. Ferrie, “Attacks on Virtual Machine Emulators,” [Online] Available: [https://www.symantec.com/avcenter/reference/Virtual\\_Machine\\_Threats.pdf](https://www.symantec.com/avcenter/reference/Virtual_Machine_Threats.pdf), Symantec Advanced Threat Research, 2007.
- [5] N. Falliere, “Windows anti-debug reference,” [Online] Available: <https://www.symantec.com/connect/articles/windows-anti-debug-reference>, 2007.
- [6] K. Yoshizaki and T. Yamauchi, “Malware detection method focusing on anti-debugging functions,” in Computing and Networking (CANDAR), Second International Symposium on, pp. 563-566, 2014.
- [7] M.-K. Sun, M.-J. Lin, M. Chang, C.-S. Laih, and H.-T. Lin, “Malware virtualization-resistant behavior detection,” in Parallel and Distributed Systems (ICPADS), IEEE 17th International Conference on, pp. 912-917, 2011.
- [8] “NtTrace - Native API tracing for Windows,” [Online] Available: [www.howzatt.demon.co.uk/NtTrace](http://www.howzatt.demon.co.uk/NtTrace), 2017.
- [9] “System Call Tracer for Windows,” [Online] Available: [http://drmemory.org/docs/page\\_drstrace.html](http://drmemory.org/docs/page_drstrace.html), 2017.
- [10] M. Sikorski and A. Honig, “Practical Malware Analysis,” no starch press, pp.159-160, 2012.
- [11] “An introduction to machine learning with scikit-learn,” [Online] Available: <http://scikit-learn.org/stable/tutorial/basic/tutorial.html>, 2017.
- [12] M. Lindorfer, C. Kolbitsch, and P. MilaniComparetti, “Detecting environment-sensitive malware,” in Recent Advances in Intrusion Detection, pp. 338-357, 2011.
- [13] C.-W. Hsu and S. W. Shieh, “Divergence detector: A fine-grained approach to detecting vm-awareness malware,” in Software Security and Reliability (SERE) IEEE 7th International Conference on, pp. 80-89, 2013.
- [14] Y. J. Liu, C. K. Chen, M. C. Y. Cho, and S. Shieh, “Fast Discovery of VM-Sensitive Divergence

- [32] "winapioverride32," [Online] Available: <http://jacquelin.potier.free.fr/winapioverride32/>, 2017.
- [33] M. Russinovich, "Process Monitor v3.40," [Online] Available: <https://docs.microsoft.com/en-us/sysinternals/downloads/procmon>, 2017.
- [30] "sklearn.model\_selection.GridSearchCV," [Online] Available: [http://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall\\_score.html](http://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html), 2017.
- [31] "sklearn.model\_selection.GridSearchCV," [Online] Available: [http://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy\\_score.html](http://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html), 2017.

## An Effective Method to Detect Environment-Aware Malware Based on the Behavioral Distances Comparison

S. Ghasemi, S. Parsa\*

Iran University of Science and Technology (IUST)

(Received: 12/01/2018, Accepted: 27/05/2018)

### ABSTRACT

*Given the inefficiency of static analysis methods due to malware techniques such as code polymorphism, metamorphism, and obfuscation, and self-modifying code, leveraging dynamic and heuristic analysis methods that are based on the analysis of runtime behavior of malwares, have become particularly important. Environment-aware malware that attempts to conceal its malicious behavior through dynamic anti-analysis methods has caused problems for dynamic analysis detection methods in practice. The purpose of this study is to present an effective method for environment-aware malware detection. Regarding to split-personality of such malware behaviors, this research has proposed an effective way to detect environment-aware malware. This method is based on system call monitoring of malicious and benign samples under the two NtTrace and drstrace softwares with different monitoring techniques and calculating behavioral distances as training data to create a Support Vector Machine model. Finally, the resulted support vector machine classifier is used to detect this type of malware with an average precision, recall and accuracy up to 100%, whereas the evaluation of previous related work shows an average precision, recall and accuracy 96.85%, 95.68% and 96.12%, respectively.*

**Keywords:** Environment-aware malware, Anti-analysis techniques, System call, Behavioral Distance, Support Vector Machines