مجله علمی پژوهشی «پدافند الکترونیکی و سایبری»
سال ششم، شماره ۴، زمستان ۱۳۹۷؛ ص ۱۳۳-۱۲۳

ارائه یک راه‌کار موثر برای تشخیص بدافزارهای آگاه به محیط مبتنی‌بر مقایسه تفاوت‌های رفتاری

سیروس قاسمی[۱]، سعید پارسا[۲]*

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 به‌واسطه 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 درنظرگیری دوگانگی رفتار چنین بدافزارهایی، در این تحقیق راه‌کاری موثر با هدف تشخیص بدافزارهای آگاه به محیط ارائه شده است. این‌روش مبتنی‌بر پایش فراخوانی‌های سیستمی نمونه‌های بدخواه و بی‌خطر تحت دو نرم‌افزار NtTracce 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 مجموعه‌داده‌ی 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، ماشین بردار پشتیبان

۱- مقدمه

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[۱]، دگرریختی[۲]، بسته‌بندی و مبهم‌سازی[۳] کد و کدخود تصحیح[۴]،که با هدف ایجاد موانع و پیچیدگی‌هایی در مسیر تحلیل و شناسایی بدافزار استفاده می‌شوند، آسیب‌پذیر هستند [۲-۱] . دسته دیگری از روش‌های تحلیلی که برتری‌هایی در مقایسه با روش‌های مبتنی‌بر امضا دارند، روش‌های تحلیلی پویای مبتنی‌بر 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[۵] استفاده می‌کنند. این‌روش‌ها شامل جستجوی کلیدهای ثبت کننده 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[۶] پیش و پس از اجرا، استفاده از اشکالات[۷] شبیه‌سازها و غیره، هستند [۵-۴]. بدافزارهایی که از روش‌های ضدتحلیلی برای دورزدن روش‌های تحلیلی مبتنی‌بر رفتار زمان اجرا، بهره می‌برند،

* رایانامه نویسنده مسئول: parsa@iust.ac.ir

1- Polymorphism
2- Metamorphism
3- Obfuscation
4- Self-modifying code
5- Anti-Analysis
6- Time Stamp Counter(TSC)
7- Bugs

اصطلاحاً بدافزارهای آگاه به‌محیط 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 اشکال‌زداها، استفاده می‌کنند. بدافزار در صورت تشخیص حضور در یک محیط تحلیلی به منظور پنهان نمودن رفتار مخرب خود یا بلافاصله به‌اجرای 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[1] یاد می‌شود [۷-۶]، قابل اندازه گیری 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[2] است. راه‌کار پیشنهادی شامل پایش فراخوانی‌های سیستمی یک نمونه‌ی مشکوک تحت دو نرم‌افزار NtTracce [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[3] و نهایتاً ارائه یک مدل با استفاده از ماشین‌های بردار پشتیبان[4] برای دسته‌بندی[5] 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[6]، یادآوری[7] و صحت[8] 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[9] 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 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 اِی‌پی‌آی‌های محلی[10] ویندوز انجام می‌شود. بعد از هر اجرا با استفاده از تصویر لحظه‌ای[11]، ماشین مجازی به وضعیت پیش از اجرای بدافزار برمی‌گردد (مرحله اول راه‌کار پیشنهادی).
- از آن‌جایی که یکی‌از روش‌های ضدتحلیلی که بدافزار برای شناسایی ابزارها و محیط‌های تحلیل شناخته‌شده و پرکاربرد، نظیر wireshark، sandboxie، virtualBox 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 این‌که 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، اِی‌پی‌آی‌های محلی ویندوز را ثبت می‌کنند. دلیل پایش فراخوانی‌های سیستمی در این سطح از سیستم‌عامل به این شرح است. اِی‌پی‌آی محلی ویندوز به‌عنوان واسطی، بین اِی‌پی‌آی‌های ویندوز و فراخوانی سیستمی قرار گرفته است و معمولاً از اِی‌پی‌آی‌های سطح بالاتر، مانند اِی‌پی‌آی‌های Win32، برای فراخوانی‌های سیستمی و انجام پیش‌پردازش‌ها و پساپردازش‌های ضروری آرگومان‌ها یا نتایج، فراخوانی می‌شود. برنامه‌های کاربردی قانونی معمولاً از طریق اِی‌پی‌آی ویندوز با سیستم‌عامل ارتباط برقرار می‌کنند. اما ممکن است کدمخرب از این لایه به‌گریزد و مستقیماً با اِی‌پی‌آی محلی تعامل کند، تا راه‌حل‌های مبتنی‌بر پایش اِی‌پی‌آی که تنها اِی‌پی‌آی‌های Win32 را به قلاب می‌اندازند[12]، بی‌نتیجه بگذارند. از این‌رو، با پایش اِی‌پی‌آی‌های 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 اِی‌پی‌آی‌های محلی استفاده می‌کنند. به‌همین دلیل چنان‌چه بدافزار یکی از این دو روش یا هر دو

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

را دور بزند، این تفاوت رفتار بدافزار در فراخوانی‌های سیستمی آن نمایان می‌شود و موجب تشخیص آن می‌شود (مرحله اول راه‌کار پیشنهادی).

- استفاده از الگوریتم طولانی‌ترین زیردنباله مشترک به منظور محاسبه تفاوت دنباله فراخوانی‌های سیستمی (مرحله دوم راه‌کار پیشنهادی).
- استفاده از ماشین‌بردار پشتیبان برای دسته‌بندی نمونه بدافزارهای آگاه به محیط و نمونه نرم‌افزارهای بی‌خطر با نتایج به‌دست‌آمده از مقایسه تفاوت‌های رفتاری، می‌تواند روی تعداد نمونه‌های کم نیز نتایج قابل قبولی را نشان دهد [۱۱] (مرحله سوم راه‌کار پیشنهادی).
- استفاده از نمونه بدافزارهای آگاه به محیط از پیش شناخته‌شده در آموزش مدل SVM (مرحله سوم راه‌کار پیشنهادی).

در ادامه این مقاله در بخش ۲ راه‌کارهای موجود در زمینه تشخیص بدافزارهای آگاه به‌محیط و چالش‌های موجود در کارهای پیشین بیان می‌شوند. در بخش ۳، راه‌کار پیشنهادی در این تحقیق شرح داده می‌شود. در بخش ۴، ارزیابی راه‌کار پیشنهادی با استفاده از توابع ارائه‌شده توسط کتابخانه Scikit-learn روی بیش از ۱۰۰ نمونه بدافزار آگاه به‌محیط و نرم‌افزار بی‌خطر، ارائه می‌شود. نهایتاً در بخش ۵ نتیجه‌گیری و کارهای آتی بیان می‌گردند.

۲- کارهای مرتبط

به‌طور کلی کارهای انجام‌شده در زمینه تشخیص بدافزارهای آگاه به محیط، یا مبتنی‌بر تفاوت رفتاری بدافزار در اجراهای مختلف هستند، یا مبتنی‌بر جستجوی الگوها یا امضاهای از پیش یافت‌شده روش‌های شناسایی محیط و ابزارهای تحلیل، هستند. از جمله روش‌های تشخیص مبتنی‌بر تفاوت رفتاری بدافزار می‌توان به موارد زیر اشاره نمود:

در مرجع [۶] روش ارائه‌شده بدافزار دارای روش ضداشکال‌زدایی را، با به قلاب‌انداختن یک اِی‌پی‌آی ضداشکال‌زدایی مشخص و تغییر مقدار بازگشتی آن، در دو حالت تحلیلی و غیرتحلیلی اجرا می‌کند. چنان‌چه بدافزار در حالت تحلیلی، با خودداری از اجرا، عمل گریز[1] را انجام دهد، اما در حالت غیرتحلیلی رفتاری مخرب از خود در قالب اضافه نمودن یک کلید به رجیستری نشان دهد، تشخیص داده می‌شود. باید این نکته را نیز در نظر داشت که، بدافزار می‌تواند با نمایش رفتاری بی‌خطر 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 محیط‌های مجازی و واقعی رفتاری متفاوت از خود بروز می‌دهد، روشی پیشنهاد شده که، اطلاعات رفتاری بدافزار را با استفاده از نرم‌افزار Process Monitor در دو محیط جمع‌آوری کرده و سپس به‌منظور تشخیص قابلیت ضد ماشین مجازی، با الگوریتم بهبود یافته Levenshtein تفاوت رفتار بدافزار را محاسبه می‌کند. به‌دلیل شناخته‌شده و رایج بودن نرم‌افزار استفاده شده برای پایش رفتار بدافزار، ممکن است بدافزار با بررسی برنامه‌های در حال اجرا روی سیستم و تشخیص آن، رفتار مخربش را پنهان کند. در مرجع [۳] یک مدل تصمیم‌گیری مبتنی‌بر یادگیری پرسپترون چند لایه با الگوریتم انتشار به عقب[2]، برای دسته‌بندی یک نمونه براساس رفتار مخرب و واکنشی‌اش 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[3] و شکست سیستم مهمان[4] باشد، آن را به‌عنوان بدافزار آگاه به محیط در نظر می‌گیرد. باید این نکته را در نظر داشت که، بدافزار آگاه به محیط می‌تواند پس از شناسایی محیط تحلیل، رفتارهایی مانند پایان اجرا، یا بروز رفتاری بی‌خطر که لزوماً شامل تکرار در فراخوانی‌های سیستمی نباشد، از خود نشان دهد. در مرجع [۱۲] ابزاری با نام DISARM[5]، با مقایسه رفتار بدفزار در چندین جعبه شن تحلیلی، بدافزاری که با تشخیص محیط تحلیل، از تحلیل می‌گریزد، شناسایی می‌کند. این ابزار در دو مرحله، کار می‌کند: مرحله پایش اجرا، با استفاده از Anubis[6] و قطع فراخوانی سیستمی از داخل محیط ویندوز و مرحله مقایسه، که با مقایسه پروفایل‌های رفتاری بدافزار شامل تنها ویژگی‌هایی که مرتبط با تغییرات مانا در وضعیت سیستم هستند به‌علاوه ویژگی‌های نشان‌دهنده فعالیت شبکه، با استفاده از فاصله Jaccard انجام می‌شود. این روش نیاز به اجراهای متعدد روی چند جعبه‌ی شن مختلف دارد. از سوی دیگر، به دلیل شبیه‌سازی تنها یک زیرمجموعه از همه‌ی اِی‌پی‌آی‌های ممکن ویندوز توسط ابزارهای تحلیل جعبه شن مانند Sandbox، علاوه بر آسیب‌پذیری نسبت به تشخیص از طریق استفاده از هر اِی‌پی‌آی پیاده‌سازی نشده یا هر اِی‌پی‌آی‌ی که به درستی شبیه‌سازی نشده، نتایج

1- Evasion

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

به‌دست‌آمده از اجراهای یک نمونه، در چند جعبه شن، نیز می‌توانند به‌خودی خود متفاوت باشند [۴]. در مرجع [۱۳] برای مقابله با تاثیر قابل توجه عوامل نا مسلم[1] 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 می‌دهد و می‌تواند افتراق[2] آثار چندین اجرا در ماشین‌های 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 روش‌های تشخیص محیط مجازی به کار گرفته شده توسط بدافزارها را بررسی می‌کند و فقط روی چهار نمونه بدافزار آزمون شده است. در مرجع [۱۴] با هدف تشخیص بدافزار آگاه از ماشین مجازی[3]، یک تعیین‌کننده محل واگرایی جدید مبتنی‌بر بلوک‌هایی از دستورالعمل‌ها به‌عنوان واحد تحلیل، ارائه شده است. این روش هر نمونه بدافزار را در دو محیط ماشین مجازی، که یکی از آن‌ها نسبت به بررسی‌های ماشین مجازی توسط بدافزار مقاوم‌تر است، اجرا می‌کند. در طول اجرا ثبت کننده بلوک پایه در دو محیط اجرا، آثار بلوک پایه را ثبت می‌کند. از آن‌جایی که هر دو محیط اجرا، محیط ماشین مجازی هستند، ممکن است روش‌های تشخیص ماشین مجازی در دو محیط موفق به تشخیص محیط اجرا شوند. در نتیجه تفاوت در دو محیط دیده نمی‌شود. در مرجع [۱۵] راهکاری برای استخراج الگوهای رفتاری مخرب موجود در هر خانواده از بدافزار با کاوش زیرگراف‌های مهم و تعیین زیرگراف‌های تفکیک‌پذیر، ارائه شده است. این راه‌کار می‌تواند برای استخراج الگوهای مخرب بدافزارهای آگاه به محیط در محیط‌های اجرای متفاوت به‌کار گرفته شود.

از جمله روش‌های تشخیص مبتنی بر جستجوی الگوها یا امضاهای از پیش یافت‌شده روش‌های شناسایی محیط‌ها و ابزارهای تحلیل می‌توان به موارد زیر اشاره نمود:

در مرجع [۱۶] یک پلاگین IDA-Pro[4] ارائه شده است، که با اسکن امضای دو گونه اصلی، از ترفندهای anti-VMware در مراحل مختلف اجرا، برای بی‌نتیجه گذاردن تشخیص‌های VMware، به‌طور خودکار نتیجه تشخیص VMware را وصله می‌کند. در مرجع [۱۷] یک روش گریز از روش‌های ضداشکال‌زدایی با وصله‌بندی[5] قانونمند[6] در فایل‌های دودویی با تحلیل ایستای دستورالعمل‌های اسمبلی ارائه شده است. به این صورت که بعد از شناسایی بخش‌هایی از کد شامل دستورالعمل‌های ضداشکال‌زدایی، آن‌ها را با دستورالعمل‌های جدید وصله‌بندی می‌کند.

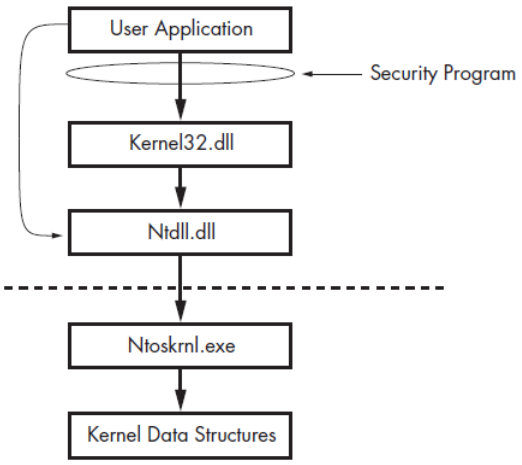
چالش‌های موجود در روش‌های کنونی مبتنی‌بر تفاوت رفتار عبارتند از: روش‌هایی که مبتنی‌بر پایش اِی‌پی‌آی‌های Win32 هستند، نسبت به بدافزارهای آگاه به محیطی که این اِی‌پی‌آی‌ها را دور می‌زنند و مستقیما اِی‌پی‌آی‌های محلی را فراخوانی می‌کنند، آسیب‌پذیر هستند. روش‌های دیگری نیز وجود دارند، که از دنباله دستورالعمل‌های ماشین فراخوانی شده توسط بدافزار، چه برای تشخیص تفاوت‌های رفتاری در محیط‌های مختلف تحلیل و چه برای تشخیص الگوهای شناسایی محیط تحلیل، استفاده می‌کنند. چنین روش‌هایی به دلیل حجم بالای خروجی‌های حاصل از پایش زمان‌بر هستند. روش‌هایی که با استفاده از اجراهای متعدد روی چندین محیط تحلیل پویا اقدام به‌شناسایی بدافزارهای آگاه به محیط می‌کنند، نیز دارای بار کاری زیادی هستند و زمان و هزینه زیادی را صرف اجرای این بدافزارها می‌کنند. برخی از روش‌های ارائه‌شده در زمینه شناسایی این بدافزارها تنها به شناسایی تعداد معدودی از روش‌های ضدتحلیلی اکتفا می‌کنند و نسبت به سایر روش‌های ضدتحلیلی آسیب‌پذیر هستند. همچنین روش‌هایی که متمرکز بر شناسایی الگوهای تشخیص محیط تحلیل هستند، از تشخیص روش‌های ضدتحلیلی جدید به‌کار گرفته‌شده توسط بدافزارها ناتوان هستند. در این تحقیق سعی شده است تا به برخی از این چالش‌ها پاسخ داده شود.

۳- راه‌کار پیشنهادی

راه‌کار پیشنهادی برای تشخیص بدافزارهای آگاه به محیط، مبتنی‌بر مقایسه تفاوت‌های رفتاری بدافزار در دو اجرای مجزا تحت پایش دو نرم‌افزار پایش فراخوانی‌های سیستمی که از نظر روش پایش متفاوتند، است. از آن‌جایی که رفتار یک برنامه از وضعیت سیستم عامل مشخص می‌شود و هیچ تغییری در وضعیت سیستم عامل بدون استفاده از فراخوانی‌های سیستمی نمی‌تواند انجام شود، بنابراین، می‌توان فراخوانی‌های سیستمی یک برنامه را که واسطی غیرقابل دور زدن هستند، به عنوان نمودار رفتار آن برنامه، در نظر گرفت [۳]. بنابراین، اندازه‌گیری فاصله رفتاری قبل از هر چیزی مستلزم تعیین مدل رفتاری در

1- Uncertain Factors
2- Divergence
3- VM-Aware
4- Interactive Disassembler Pro
5- Patching
6- Rule-Based

قالب دنباله فراخوانی‌های سیستمی در زمان اجرا است. از سوی دیگر اِی‌پی‌آی‌های محلی یک واسط سطح پایین برای تعامل با ویندوز هستند که به ندرت توسط برنامه‌های غیر مخرب استفاده می‌شوند، اما در بین مولفان بدافزار رایج هستند. فراخوانی اِی‌پی‌آی‌های محلی، اِی‌پی‌آی‌های Win32 را دور می‌زنند. فراخوانی اِی‌پی‌آی‌های محلی به طور مستقیم برای نویسندگان بدافزار، به دلیل مخفیانه‌تر بودن و این‌که به آن‌ها امکان انجام چیزهایی را می‌دهد که ممکن است به صورت دیگر امکان‌پذیر نباشند، جذاب است. بسیاری از ضدویروس‌ها و محصولات مراقبت میزبان فراخوانی‌های سیستمی انجام‌شده از کتابخانه Win32 توسط یک فرآیند را پایش می‌کنند. ازاین‌رو فراخوانی اِی‌پی‌آی‌های محلی به‌طور مستقیم ممکن است به‌تواند از یک محصول امنیتی با طراحی ضعیف مطابق شکل (۱) بگریزد [۱۰]. برای حل این مشکل، می‌توان از ابزارهایی برای ثبت‌نگاری[1] فراخوانی‌های سیستمی در سطح اِی‌پی‌آی‌های محلی استفاده نمود. در این تحقیق برای تعیین دنباله اِی‌پی‌آی‌های محلی از دو نرم‌افزار با نام‌های NtTracce و drstrace که کم‌تر توسط تحلیل‌گران بدافزار استفاده شده‌اند و فراخوانی‌های سیستمی را در سطح اِی‌پی‌آی‌های محلی ثبت می‌کنند، استفاده شده است.



شکل (۱): استفاده از اِی‌پی‌آی‌های محلی برای اجتناب از شناسایی [۱۰]

راه‌کار پیشنهادی برای تشخیص بدافزارهای آگاه به محیط به سه مرحله تقسیم می‌شود. در مرحله اول فایل اجرایی نمونه بدافزارهای آگاه به محیط و نرم‌افزارهای بی خطر تحت دو نرم‌افزار پایش در شرایط یکسان اجرا و سپس دنباله اِی‌پی‌آی‌های فراخوانی شده در هر اجرا استخراج می‌شود. در مرحله دوم فاصله رفتاری هر نمونه، در دو اجرا، براساس بزرگترین زیردنباله مشترک از فراخوانی‌های سیستمی با استفاده از الگوریتم طولانی‌ترین زیردنباله مشترک محاسبه می‌شود. همچنین سایر ویژگی‌های رفتاری از دنباله‌های اِی‌پی‌آی‌های فراخوانی شده استخراج می‌شوند. نهایتاً در مرحله سوم با استفاده از ماشین‌بردار پشتیبان، مدلی برای دسته‌بندی نمونه‌های مورد بررسی ساخته می‌شود. از این مدل برای تشخیص نمونه بدافزارهای جدید که ویژگی‌های رفتاری آن‌ها در دو مرحله قبل محاسبه شده، استفاده می‌گردد. در ادامه این بخش، هر کدام از مراحل به تفصیل شرح داده می‌شوند.

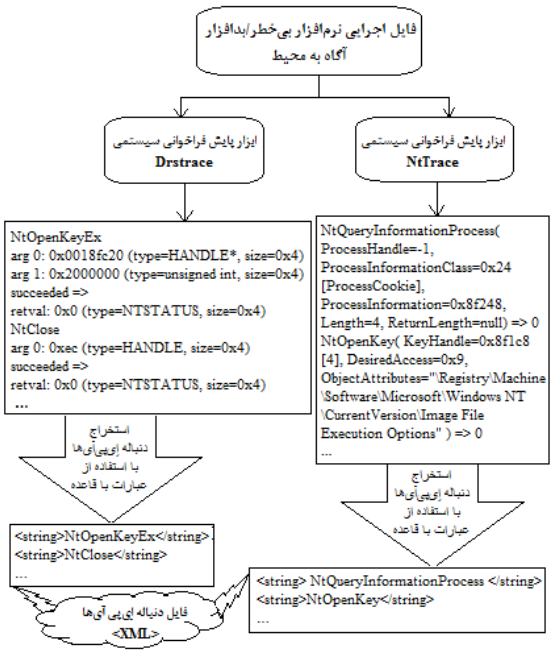
۳-۱ - مرحله اول راه‌کار پیشنهادی

در این مرحله فایل اجرایی نمونه بدافزار آگاه به محیط یا نرم‌افزار بی‌خطر با شرایط یکسان تحت دو نرم‌افزار پایش فراخوانی‌های سیستمی drstrace و NtTrace اجرا می‌شود. دلیل استفاده از این دو نرم‌افزار این است که اولاً این دو نرم‌افزار کم‌تر توسط تحلیل‌گران بدافزار استفاده شده‌اند، که باعث می‌شود یکی از چالش‌های موجود در راه‌کارهای پیشین یعنی بررسی فرآیندهای در حال اجرا در سیستم توسط بدافزار و شناسایی ابزارهای ثبت‌نگاری مرتفع شود، ثانیاً این دو نرم‌افزار از دو روش کاملاً متفاوت برای ثبت‌نگاری استفاده می‌کنند. NtTracce از حالت اشکال‌زدایی ویندوز برای ثبت اِی‌پی‌آی‌های محلی فراخوانی‌شده استفاده می‌کند [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[2] بر روی فرآیند یا پردازه مورد نظر عمل می‌کند [۱۹]. این تفاوت در روش‌های ثبت‌نگاری سبب می‌شود، در صورتی که بدافزار یکی از این دو روش یا هر دو را دور بزند، این تفاوت رفتار بدافزار در دنباله فراخوانی‌های سیستمی آن آشکار گردد و تشخیص آن میسّر شود. از سوی دیگر، به‌دلیل این‌که هر دو نرم‌افزار اقدام به ثبت فراخوانی‌های سیستمی یک نمونه در سطح اِی‌پی‌آی‌های محلی ویندوز می‌کنند، امکان دور زدن این برای بدافزار بسیار دشوار می‌گردد. به منظور اجرای خودکار فایل اجرایی نمونه مورد بررسی از واسط خط فرمان ارائه‌شده برای دو نرم‌افزار، استفاده می‌شود. در شکل (۲) نمای کلی از مرحله اول راه‌کار پیشنهادی آورده شده است. پس از اجرای نمونه مورد

1-Logging

2- Process virtual machine

بررسی، فایل‌های رخداد[1] ثبت‌شده 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[2]، دنباله اِی‌پی‌آی‌های فراخوانی‌شده، استخراج می‌شوند. نهایتاً دنباله اِی‌پی‌آی‌های 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 ارائه‌شده برای یافتن شباهت متن[3] مانند LCS، N-gram، Damerau-Levenshtein، Jaro 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 به‌عنوان یکی از ویژگی‌ها برای آموزش مدل ماشین بردار پشتیبان استفاده می‌شود. با ساخت LCS طولانی‌ترین زیر دنباله مشترک از فراخوانی‌های سیستمی را می‌سازیم. ما از این زیر دنباله تعداد اِی‌پی‌آی‌های محلی منحصربه‌فرد 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 اِی‌پی‌آی‌هایی که توسط هر یک از دو نرم‌افزار در دو اجرا ثبت شده‌اند.
  این ویژگی می‌تواند برای تشخیص بدافزارهای آگاه به‌محیطی که بلافاصله بعد از تشخیص محیط تحلیل به اجرای خود پایان می‌دهند، استفاده شود.
- تعداد اِی‌پی‌آی‌های منحصربه‌فردی 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 اِی‌پی‌آی‌های مشترک از نظر نوع که در هر دو دنباله اِی‌پی‌آی 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.

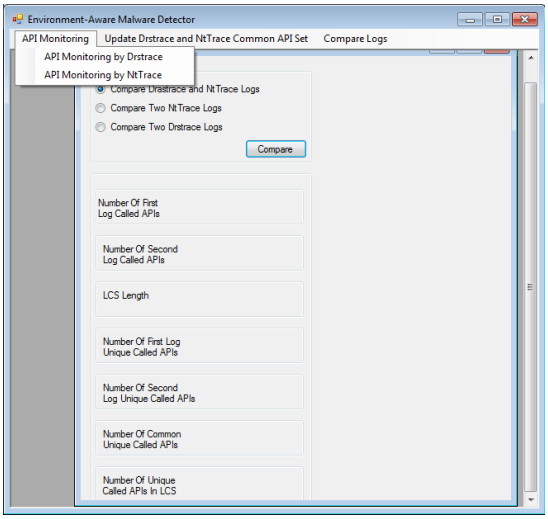
نوع نمونه مورد بررسی یکی از موارد بدافزار آگاه به محیط یا نرم‌افزار غیرآگاه به محیط یا ناشناخته خواهد بود. نرم‌افزارهای بی‌خطر و بدافزارهای فاقد روش‌های ضدتحلیلی در دسته نرم‌افزارهای غیرآگاه به محیط ذخیره می‌شوند. دلیل این‌کار این است که هدف ما در این تحقیق تشخیص بدافزارهای آگاه به‌محیط از سایر بدافزارها و نرم‌افزارها است، تا به‌توانیم 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

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

SVM ارائه شده نوع آن‌ها پیش‌بینی شود.

شکل (۳): نمای کلی مرحله دوم راه‌کار پیشنهادی

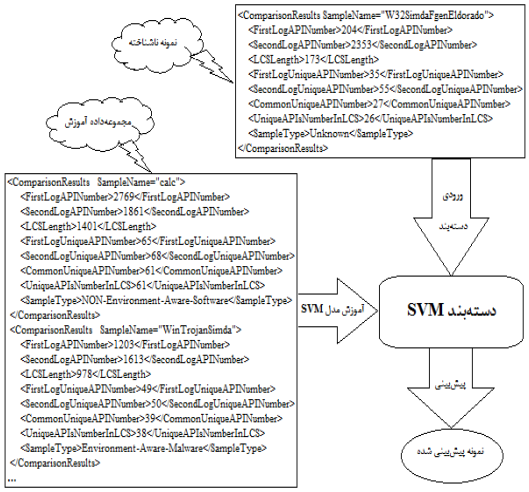
مرحله اول و دوم راه‌کار ارائه شده برای تشخیص بدافزارهای آگاه به محیط، در نرم‌افزاری به نام EnvironmentAwareMalwareDetector در سکوی .NET 4.6.1 Framework، با زبان برنامه‌نویسی C# توسعه داده شده است که در شکل (۴) نمایی از واسط کاربر آن آورده شده است.



شکل (۴): واسط کاربر نرم‌افزار EnvironmentAwareMalwareDetector

۳-۳- مرحله سوم راه‌کار پیشنهادی

به منظور دسته‌بندی نمونه نرم‌افزارهای بی‌خطر و بدافزارهای آگاه به‌محیط مورد آزمون، پارامترهای استخراج شده از پایش فراخوانی‌های سیستمی آن‌ها را به عنوان داده‌های آموزش به ماشین‌بردار 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[1] می‌کنیم. در شکل (۵) نمای کلی مرحله سوم راه‌کار پیشنهادی آورده شده است.



شکل (۵). نمای کلی مرحله سوم راه‌کار پیشنهادی

به منظور ساخت مدل ماشین بردار پشتیبان از فایل حاوی نتایج محاسبات فاصله رفتاری اجراها، برای آموزش این مدل استفاده می‌کنیم. برای ساخت دسته‌بند[2] 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 کتابخانه‌ی 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[3] 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-نمونه, n-ویژگی] 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

بررسی‌شده به‌عنوان برچسب‌های کلاس (رشته یا اعداد صحیح) با اندازه [n-نمونه] است. بعد از برازش[1] 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[2] که به نمونه کلاس GridSearchCV داده می‌شود و یک مجموعه ارزیابی برای محاسبه متریک‌های کارایی، تقسیم می‌کنیم. ساخت مدل ماشین‌بردار 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 مجموعه‌داده‌ها

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 استفاده‌شده 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 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، malwr.com، thezoo.morirt.com، VirusShare و DasMalwerk 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 مجموعه‌داده، از آن برای آموزش و آزمون دسته‌بند ماشین بردار پشتیبان استفاده کردیم.

جدول (۲): خانواده بدافزارهای آگاه به محیط استفاده‌شده در ارزیابی

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

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 ماشین‌بردار پشتیبان استفاده می‌کنیم. به‌منظور بهینه‌سازی مدل ماشین‌بردار پشتیبان، از رویه ارائه‌شده در مرجع [۲۵] استفاده می‌کنیم. این رویه شامل گام‌های زیر است:

- تبدیل داده‌ها به قالب یک بسته ماشین‌بردار پشتیبان: کلاس SVC دو آرایه به عنوان ورودی می‌گیرد، یک آرایه X با اندازه‌ی [n-sample, n-feature] که نمونه‌های آموزشی را نگه می‌دارد و یک آرایه y که دربردارنده برچسب‌های کلاس با اندازه [n-sample].
- انجام مقیاس‌گذاری ساده روی داده‌ها: به‌منظور 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].

1 -Fitting
2- Development Set

- انتخاب مدل: کرنل خطی و RBF، را در نظر می‌گیریم.
- به‌کارگیری روش اعتبارسنجی متقابل برای یافتن بهترین پارامتر C و گاما: از کلاس sklearn.model-selection.GridSearchCV که یک جستجوی جامع[1] روی مقادیر تعیین شده پارامتر برای یک برآوردگر[2] انجام می‌دهد، استفاده می‌کنیم[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 دسته‌بند است، در این‌که یک نمونه‌ی منفی را به عنوان مثبت برچسب‌گذاری نکند. بهترین مقدار دقت یک و بدترین مقدار آن صفر است. برای محاسبه‌ی دقت از رابطه (۱) که در آن، tp تعداد صحیح مثبت[3] و fp تعداد غلط مثبت[4] است، استفاده می‌شود[۲۹] .

$$\text{Precision}=\frac{tp}{tp+fp} \quad (۱)$$

- یادآوری: یادآوری، توانایی دسته‌بند در یافتن همه نمونه‌های مثبت را بیان می‌کند. بهترین مقدار یادآوری یک و بدترین مقدار آن صفر است. برای محاسبه یادآوری از رابطه (۲) که در آن، tp تعداد درست‌های مثبت و fn تعداد غلط‌های منفی[5] است، استفاده می‌شود [۳۰] .

$$\text{Recall}=\frac{tp}{tp+fn} \quad (۲)$$

- صحت: کسر نمونه‌هایی که به درستی دسته‌بندی شده‌اند، را بیان می‌کند. بهترین مقدار صحت یک و بدترین مقدار آن صفر است [۳۱].

مدل حاصل روی ۳۰٪ از نمونه‌ها که در فرآیند آموزش مدل کنار گذاشته شده‌اند، مورد آزمون قرارگرفته و میانگین دقت، میانگین یادآوری و میانگین صحت آن، هر کدام تا حد ۱۰۰٪ است. مقایسه راه‌کار پیشنهادی با راه‌کار ارائه‌شده در مرجع [۳]، که به ترتیب میانگین دقت، میانگین یادآوری و میانگین صحت را ۹۶/۵۶٪، ۹۵/۶۸٪ و ۹۶/۱۲۵٪ برای مدل تصمیم‌گیری ساخته شده براساس پرسپترون چندلایه نشان می‌دهد، بیانگر بهبود پارامترهای ارزیابی در راه‌کار پیشنهادی است.

۵- نتیجه‌گیری

با در نظرگیری آسیب‌پذیری روش‌های تحلیل ایستا در برابر بدافزارهایی که بدون تغییر در منطق کدشان، تغییر شکل می‌دهند، به‌کارگیری راه‌کارهایی برای تحلیل بدافزار که روی رفتار زمان اجرای بدافزار تمرکز می‌کنند، نظیر روش‌های تحلیل پویا و مکاشفه‌ای، امری اجتناب‌ناپذیر است. با این حال اساساً چنین روش‌هایی نیز در برابر بدافزارهایی با عنوان بدافزارهای آگاه به‌محیط که از روش‌های ضدتحلیلی پویا برای پنهان‌سازی رفتار مخرب خود در صورت تشخیص محیط‌ها و ابزارهای تحلیل استفاده می‌کنند، آسیب‌پذیر هستند. از این‌رو با هدف رفع این آسیب‌پذیری و تشخیص بدافزارهای آگاه به محیط و باتوجه به ماهیت رفتار دوگانه چنین بدافزارهایی، در این تحقیق، راه‌کاری موثر مبتنی‌بر پایش فراخوانی‌های سیستمی، با استفاده از دو نرم‌افزار پایش فراخوانی‌های سیستمی با روش‌های متفاوت پایش به نام‌های NtTracce و drstrace و محاسبه فاصله رفتاری حاصل از دو اجرای یک نمونه ارائه شده است. این روش با پایش فراخوانی‌های سیستمی نمونه‌های بدخواه و بی‌خطر و محاسبه فاصله رفتاری حاصل از دو اجرای هر نمونه، یک دسته‌بند ماشین‌بردار پشتیبان را با استفاده از روش اعتبارسنجی متقابل و جستجوی گرید با آموزش روی ۷۰٪ داده‌های حاصل، ارائه می‌کند. در نهایت این دسته‌بند، با ارزیابی روی ۳۰٪ باقی‌مانده داده‌ها میانگین دقت، یادآوری و صحت تا حد ۱۰۰٪، را نشان می‌دهد، که در مقایسه با کار مرتبط در این زمینه با میانگین دقت، یادآوری و صحت به‌ترتیب ۹۶/۵۸٪ ،۹۵/۶۸٪ و ۹۶/۱۲۵٪، بهبود را نشان می‌دهد.

به‌عنوان کارهای آتی پیشنهاد می‌شود که از ابزارهای پایش توابع Win32 نظیر WinAPIOverrride [۳۲]، Process Monitor [۳۳] و غیره استفاده نمود. می‌توان با نگاشت بین توابع Win32 و اِی‌پی‌آی‌های محلی تفاوت رفتاری بدافزارهای آگاه به‌محیط را، در قالب فراخوانی‌های سیستمی آن‌ها که از دو لایه متفاوت از سیستم عامل صورت می‌گیرد، آشکار نمود. همچنین پیشنهاد می‌گردد که با داده‌کاوی نتایج حاصل از پایش این گروه از بدافزارها و استخراج الگوهای استفاده‌شده که در بدافزارهای مختلف تکرار می‌شود، روش‌های جدید به‌کارگرفته شده برای تشخیص محیط تحلیل را شناسایی و در ادامه محیط‌های تحلیل را نسبت به آن‌ها مقاوم نمود.

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

۶- منابع

[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–Reactive 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, 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," [On.line] Available:www.howzatt.demon.co.uk/NtTrace, 2017.

[9] "System Call Tracer for Windows," [Online] Available: 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 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, 3rd 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] Avai-lable: http://scikit-learn.org/stable/modules/generated/skl-earn.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] Availa-ble: http://scikit-learn.org/stable/modules/generated/skl-earn.preprocessing.StandardScaler.html, 2017.

[27] "sklearn.model_selection.GridSearchCV," [Online] Avai-lable: http://scikit-learn.org/stable/modules/generated/skl-earn.model_selection.GridSearchCV.html, 2017.

[28] "Tuning the hyper-parameters of an estimator," [Online] Available: http://scikit-learn.org/stable/modules/grid_sea-rch.html, 2017.

[29] "sklearn.model_selection.GridSearchCV," [Online] Avai-lable: http://scikit-learn.org/stable/modules/generated/skl-earn.metrics.precision_score.html, 2017.

[30] "sklearn.model_selection.GridSearchCV," [Online] Avai-lable: http://scikit-learn.org/stable/modules/generated/skl-earn.metrics.recall_score.html, 2017.

[31] "sklearn.model_selection.GridSearchCV," [Online] Avai-lable: http://scikit-learn.org/stable/modules/generated/skl-earn. .metrics.accuracy_score.html, 2017.

[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.

Journal of Electronical & Cyber Defence
Vol. 6, No. 4, 2019, Serial No. 24

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

* Corresponding Author Email: parsa@iust.ac.ir