مجله علمی- پژوهشی "پدافند الکترونیکی و سایبری"
سال سوم، شماره ۱، بهار ۱۳۹۴، ص ۱۴-۱

رویکردی نو در شناسایی بدافزارها با تحلیل تصویر حافظه فضای کاربر

معصومه آقایی خیرآبادی[1*]، سید محمدرضا فرشچی[2]، حسین شیرازی[3]

۱- کارشناسی ارشد، دانشکده فناوری اطلاعات ارتباطات و امنیت، دانشگاه صنعتی مالک اشتر تهران
۲- دانشجوی دکتری، مرکز فرماندهی و کنترل، آزمایشگاه شبکه‌های اجتماعی تهران
۳- استاد، دانشکده فناوری اطلاعات، ارتباطات و امنیت، دانشگاه صنعتی مالک اشتر تهران

(دریافت: ۹۳/۰۵/۰۷؛ پذیرش: ۹۴/۰۶/۱۰)

چکیده

روش‌های تشخیص بدافزار مبتنی بر تحلیل محتویات حافظه در سال‌های اخیر محبوبیت زیادی به‌دست آورده‌اند. تحقیقات انجام‌شده در این زمینه پیشرفت زیادی داشته و چهارچوب‌های تحلیل قدرتمندی نیز به‌وجود آمده است. درحالی‌که این چهارچوب‌ها امکان بررسی یک تصویر لحظه‌ای حافظه با جزئیات کامل را فراهم می‌کنند، اما تفسیر و همبسته‌سازی این جزئیات برای استخراج ناسازگاری‌ها نیاز به دانش کاملی از ساختارهای داخلی سیستم‌عامل دارد. در این پژوهش تمرکز پویش‌گر پیشنهادی ما بر استخراج اطلاعات از ساختارهای حافظه با پرداختن به ناسازگاری‌های ایجادشده توسط روش‌های دفاعی مورداستفاده بدافزارها می‌باشد. درروش ارائه‌شده، برای اولین بار با استفاده از جرم‌شناسی حافظه به بررسی عملکرد اصلی بدافزار با استخراج فراخوانی‌های آن از فضای کاربر حافظه پرداختیم؛ به‌عبارت‌دیگر در این روش با توصیف ساختارهای حافظه اثرات مؤثر مربوط به تغییرات رجیستری، دسترسی فایل‌های کتابخانه‌ای و فراخوانی‌های توابع سیستم‌عامل استخراج شدند. در انتها برای ارزیابی ویژگی‌های استخراج‌شده، نمونه‌ها را براساس ویژگی‌های انتخاب‌شده دسته‌بندی کردیم، نتایج شامل نرخ تشخیص ۹۸٪ و نرخ مثبت کاذب ۱۶٪ می‌باشند که نشان‌دهنده مؤثر بودن روش‌های تشخیص مبتنی بر تحلیل محتویات حافظه است.

واژه‌های کلیدی: تحلیل بدافزار، جرم‌شناسی حافظه، اثرات دیجیتال، حافظه فضای کاربر، داده فرار، استخراج ویژگی

۱- مقدمه

در دهه گذشته روش‌های مبهم‌سازی[1] و اختفای[2] پیشرفته در توسعه بدافزارها به‌عنوان یک راه‌کار مؤثر برای محدود کردن قدرت ضد بدافزارها به‌وجود آمدند. امروزه بدافزارها توسط متخصصان حرفه‌ای ایجاد می‌شود، آن‌ها از روش‌هایی استفاده می‌کنند تا شناسایی و حذف بدافزارها برای نرم‌افزارهای امنیتی مشکل شود. به‌عنوان مثال، بدافزار استاکس‌نت که برای ایجاد اختلال در تأسیسات مبتنی بر SCADA همانند تأسیسات هسته‌ای کشورهای مختلف به‌ویژه ایران طراحی شده بود، شامل یک پایگاه داده از ضد بدافزارهای متداول است، که بسته به نرم‌افزار امنیتی که در سیستم هدف نصب می‌باشد، اقداماتی را برای فرار از تشخیص توسط این نرم‌افزار خاص انجام می‌دهد [۱]. بدافزارها با استفاده از چندین لایه رمزنگاری و روش‌های مبهم‌سازی، کد نمونه را تغییر می‌دهند، به‌طوری‌که نسل دیگری از همان بدافزار با عملکرد رفتاری یکسان، ولی ظاهری کاملاً متفاوت ایجاد می‌شود. بدافزارهای چندریخت[3] و دگردیس[4] از این نوع هستند [۲]. روش‌های سنتی شناسایی بدافزارها عمدتاً براساس روش‌های تشخیص مبتنی بر امضای کد عمل می‌کنند. به‌طورکلی روش‌های تشخیص مبتنی بر ساختار نحوی[5] کد، به‌راحتی توسط روش‌های مبهم‌سازی فریب می‌خورند.

*رایانامه نویسنده پاسخگو: masoume_aghaei@mut.ac.ir

1- Obfuscation
2- Stealth
3- Polymorphism
4- Metamorphism
5- Syntax

روش‌های تشخیص مبتنی بر رفتار در حالت کلی به دودسته، ایستا و پویا تقسیم می‌شوند [۳]. مشکل عمده در روش ایستا، عدم تحلیل مؤثر کدهای مبهم‌سازی‌شده، است [۴]. مهم‌ترین مزیت استفاده از روش تحلیل در زمان اجرا، مقابله با روش های مبهم‌سازی و رمزنگاری کد می‌باشد. نیاز رو به افزایش سیستم‌های تشخیص بدافزار، به الگوریتم‌های تحلیل کارا برای مقابله با روش‌های ضد تحلیل پویا و زمان‌بر بودن این تحلیل‌ها چالش‌های مهمی در این حوزه می‌باشند.

هدف عمده الگوریتم‌های تحلیل بدافزار استخراج چگونگی تغییر وضعیت سیستم است. به‌طورکلی شناسایی این تغییرات به دو علت انجام می‌شود: ۱- به‌دست آوردن وضعیت‌هایی که در زمان تشخیص آلوده شدن یک سیستم، باید مورد بررسی قرار گیرند. ۲- اخذ تصمیمات مناسب در برابر مواجه‌شدن با تهدیدهای ناشی از بدافزارها. در این پژوهش، با استفاده از روش‌های تحلیل تصویر حافظه به استخراج اثرات کدهای مخرب از محتویات حافظه پرداخته می‌شود. درروش پیشنهادی علاوه بر مهم‌ترین ساختارهای داده در حافظه (فضای هسته و فضای کاربر)، روش‌های دفاعی و ضد تحلیل[1] حافظه مورد استفاده بدافزارها، نیز بررسی می‌شود. برای اولین بار تأکید ما بر استخراج اثرات با بررسی شواهد ناشی از به‌کارگیری روش‌های دفاعی می‌باشد. بر اساس آزمایش‌های دقیق و تجربی، یکی از موارد قابل‌توجه مشاهده‌شده، ثابت باقی ماندن اطلاعات در حافظه به مدت طولانی‌تری نسبت به زمان پیش‌بینی‌شده است؛ بنابراین به‌صورت همزمان علاوه بر وضعیت جاری، سیستم قادر به استخراج برخی اثرات اجرایی قبلی نیز خواهد بود.

ابزارهای تحلیل حافظه از طریق بازسازی ساختارهای داده سیستم‌عامل عمل می‌کنند. درواقع با الگو قرار دادن این ساختارها، داده‌ها مکان‌یابی شده و پردازه[2] دوباره ایجاد می‌شود. روش‌های ضد تحلیل حافظه به‌منظور شکست روش‌های تحلیل و مخفی کردن اثرات مخرب از روت‌کیت‌ها استفاده می‌کنند. این روت‌کیت‌ها اشاره‌گرها را از این ساختارهای داده حذف می‌کنند [۶-۵]. پاک کردن اشاره‌گرها، به‌ویژه از ساختارهای مدیریت داده در سطح هسته، ممکن است اجرای بدافزار یا ادامه عملکرد سیستم‌عامل را با مشکل مواجه کند. با این‌وجود، حتی درصورت پاک شدن اشاره‌گرها روش‌هایی همانند روش‌های Carving وجود دارند [۷]، که داده‌ها را بازیابی می‌کنند. بدافزار نمی‌تواند اطلاعات را از همه این ساختارها پنهان کند؛ بنابراین وجود ناسازگاری در اطلاعات جمع‌آوری‌شده از چندین ساختار، نشان‌دهنده آلوده شدن سیستم است. به‌عبارت دیگر مخفی کردن اثرات از همه ساختارها غیرممکن است و همواره اثراتی وجود دارند که ثابت باقی می‌مانند.

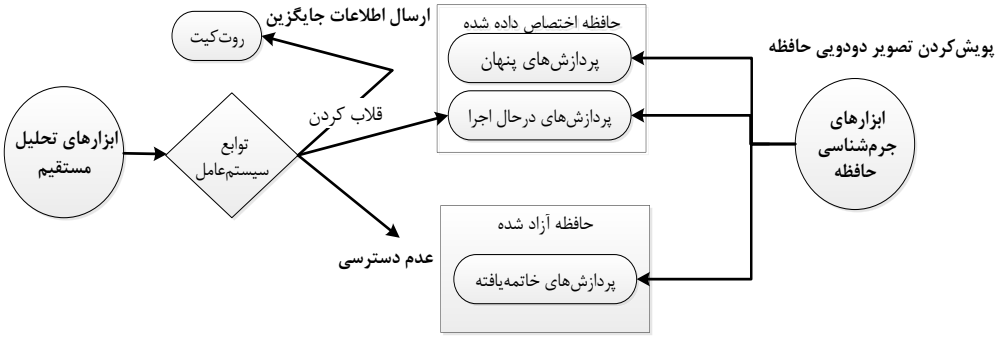
در ادامه مفاهیم مرتبط با تحلیل حافظه، به‌ویژه تحلیل فضای کاربر بیان می‌شود. در بخش ۳ مروری بر کارهای انجام‌شده درزمینه تحلیل حافظه خواهیم داشت و در بخش ۴ روش‌های ضد تحلیل حافظه مطرح‌شده در سال‌های اخیر بیان می‌گردد. در بخش ۵ پویشگر پیشنهادی معرفی می‌گردد و در بخش ۶ مراحل تحلیل و تشخیص توضیح داده می‌شود. در بخش ۷ و ۸ به ترتیب مدل پیشنهادی و فرایند دسته‌بندی را بیان نموده و در انتها (در بخش ۹ و ۱۰) نتایج ارزیابی، نتیجه‌گیری و کارهای آینده، تشریح خواهید گردید.

۲-مقدمه‌ای بر تحلیل حافظه

جرم‌شناسی حافظه[3] به معنای، جمع‌آوری و تحلیل اطلاعات از محتوای حافظه کامپیوتر می‌باشد. به‌عبارت دیگر، این فرایند به یافتن و استخراج اثرات مخرب بدافزار از حافظه فیزیکی[4] کامپیوتر اشاره دارد. حافظه فیزیکی شامل اطلاعات ضروری در مورد وضعیت زمان اجرای برنامه‌ها در سیستم می‌باشد. با گرفتن یک نسخه کامل از محتویات حافظ و تجزیه‌وتحلیل آن روی یک سیستم مورد اعتماد، بازسازی وضعیت اصلی سیستم امکان‌پذیر است. این وضعیت شامل برنامه‌های کاربردی در حال اجرا، فایل‌های مورداستفاده برنامه‌ها، اتصالات فعال شبکه و بسیاری از اثرات دیگر می‌باشد؛ بنابراین بررسی محتویات حافظه برای به‌دست آوردن اطلاعات صحیح در مورد پردازه‌ها بسیار مهم است.

استفاده از تحلیل حافظه به فرایند تشخیص باز شدن فشردگی[5] کد، تشخیص روت‌کیت [۸] و مهندسی معکوس کمک می‌کند. جدا کردن فرایند تحلیل به‌عنوان یک ویژگی مهم در این روش مطرح است. در واقع این روش امکان تحلیل و تشخیص بدافزار، از خارج سیستم آلوده و ایجاد یک نقطه دید بی‌طرف را فراهم می‌کند. روت‌کیت‌ها با قلاب[6] شدن به توابع سیستم‌عامل و ارسال اطلاعات نادرست به ابزارهای ناظر نصب‌شده در سیستم از تشخیص اثرات مخرب جلوگیری می‌کنند. به‌عنوان مثال حضور یک فایل یا پردازه در حال اجرا را پنهان می‌کنند [۹]. در روش‌های تحلیل حافظه، این نوع روت‌کیت‌ها نمی‌توانند مانع از دسترسی به اثرات مخرب در محتوای حافظه شوند. همان‌طورکه در شکل (۱) نمایش داده شده است، ابزارهای جرم‌شناسی حافظه از فراخوانی توابع سیستم‌عامل برای

1- Anti-Forensic
2- Process
3- Memory Forensic
4- RAM
5- Packing
6- Hook



شکل (۱). مقایسه قابلیت‌های روش جرم‌شناسی حافظه با روش تحلیل مستقیم

جمع‌آوری شواهد استفاده نمی‌کنند و تصویر حافظه را به‌صورت مستقیم پویش می‌کنند. درحالی‌که ابزارهای تحلیلی که در سیستم‌عامل نصب می‌شوند فقط از طریق فراخوانی توابع سیستم‌عامل می‌توانند از وضعیت سیستم آگاه شوند. روت‌کیت‌ها با قلاب شدن به این توابع می‌توانند خروجی آن‌ها را تغییر دهند و اطلاعات اشتباه به ابزار تحلیل برگردانند. در روش‌های جرم‌شناسی وقتی پردازش خاتمه می‌یابد فضای اختصاص داده‌شده به آن تا زمانی که توسط پردازش دیگری رونویسی نشده است؛ قابل‌بررسی است. درحالی‌که سیستم‌عامل پس از خاتمه پردازش اجازه دسترسی به این فضای آزادشده را نمی‌دهد. از طرف دیگر شواهد حضور برخی بدافزارها مثل Code Red، Witty و SQL Slammer [۱۱-۱۰] را فقط می‌توان در حافظه یافت؛ بنابراین تنها راه‌حل تشخیص این نوع بدافزارها، استفاده از روش‌های تحلیل حافظه است.

۲-۱- تحلیل حافظه فضای کاربر

در ویندوز مشابه سیستم‌عامل‌های دیگر، هر فضای آدرس مجازی به دو قسمت فضای کاربر و فضای هسته تقسیم می‌شود. در فضای هسته، داده و کد مربوط به سیستم‌عامل و در فضای کاربر، داده و کد پردازه‌ها ذخیره می‌شوند. فضای کاربر برای هر پردازه منحصربه‌فرد است اما بیشتر فضای هسته بین پردازه‌ها مشترک است. استخراج اطلاعات مربوط به پردازه هدف می‌تواند براساس هرکدام از ساختارهای موجود در فضای هسته/کاربر انجام شود [۱۳-۱۲]. تاکنون تحقیقاتی درزمینه بررسی ساختارهای فضای کاربر مانند پشته و فایل انجام شده است. کاربردی‌ترین تحقیقات، که امکان تحلیل فضای کاربر را فراهم می‌کند درزمینه درخت توصیف‌کننده آدرس مجازی[1] (VAD) در [۱۴] انجام شده است. ساختار VAD برای مدیریت حافظه فضای کاربر توسط هسته استفاده می‌شود. یک اشاره‌گر در ساختار EPROCESS، آدرس شروع این ساختار را نگه‌داری می‌کند. EPROCESS ساختار داده‌ای است که برای نمایش یک پردازه توسط ویندوز استفاده می‌شود. این ساختار در فضای هسته نگه‌داری می‌شود و شامل فیلدهایی برای یافتن منابع داده مربوط به پردازه است. ردیابی (داده‌های مربوط به) پردازه می‌تواند از طریق ساختارهای EPROCESS و PEB انجام شود. ساختار PEB متشکل از سه فهرست (In Load Order Module List In Memory Order Module List, In Initialization Order List,) است که توسط سیستم‌عامل ویندوز برای هر پردازه در حال اجرا در فضای کاربر نگه‌داری می‌شود [۱۵]. این ساختار شامل همه پارامترهای حالت کاربر مختص پردازه، مانند فهرست ماژول‌های بارگذاری شده است. در حال حاضر بر اساس بررسی‌های ما بدافزارها (مثلاً اکستاکس‌نت) از روش‌های ضد تحلیل برای مخفی کردن اثرات خود از این دو ساختار بهره می‌برند. تغییر این دو ساختار تأثیری بر اطلاعات ذخیره‌شده در ساختار VAD ندارد.

جهت بهینه کردن زمان جستجو، VADها در یک درخت جستجوی دودویی خودمتوازن[2] در فضای هسته ذخیره می‌شود. وقتی به یک پردازه یک بلوک از حافظه (با آدرس پایه، اندازه ناحیه و نوع دسترسی) اختصاص می‌یابد، هسته یک گره VAD متناظر برای مدیریت این بلوک به درخت VADها اضافه می‌کند. مثلاً وقتی یک فضای حافظه به یک فایل اختصاص داده می‌شود، اطلاعات افزوده‌شده به VAD امکان مکان‌یابی جزئیات مربوط به فایل فراخوانی‌شده را فراهم می‌کند. به‌عبارت دیگر از طریق بررسی ساختارهای متداخل VAD، اطلاعات مربوط به این ناحیه حافظه استخراج می‌شود. وقتی پردازه فضای حافظه را آزاد کند، هسته گره VAD متناظر را از درخت حذف می‌کند. تغییر ساختار VAD به‌منظور جدا کردن گره‌ها[3] و اختلال در فرایند جستجوی درخت VADها به یک دسترسی سطح هسته (روش های DKOM[4]) نیاز دارد؛ بنابراین بدافزار باید بتواند در فضای کاربر و فضای هسته عمل کند و نوشتن یک بدافزار با این قابلیت مشکل خواهد بود.

1- Virtual address descriptor
2- Self-Balancing Binary Search Tree (AVL)
3- Unlink VAD Node
4- Direct Kernel Object Manipulation

مشاهده و تأیید اعتبار اطلاعات استفاده‌شده توسط بدافزار، از طریق VAD از چند جهت حائز اهمیت است. این ساختار یک دید زمان اجرا از داده‌هایی که برای یک پردازه بارگذاری‌شده را فراهم می‌کند. در بسیاری از موارد، پردازه مخرب مکان‌های این داده‌ها در فضای حافظه فرآیند را دچار ابهام می‌کند. از آن‌جایی‌که ساختار VAD، انتزاعی بر صفحات حافظه است، بدیهی است که شامل اطلاعات صحیح از زمان اجرای پردازه باشد. VAD امکان هدف قرار دادن نواحی مناسب برای تحلیل در فضای کاربر پردازه را فراهم می‌کند. وقتی فرایند باز شدن فشردگی کد پردازه به پایان رسید، ساختار VAD برای تشخیص DLLهای[1] متعلق به پردازه استفاده می‌شود. ما با انتخاب تحلیل ساختار VAD، که یک ساختار ذخیره‌شده در فضای هسته است، تحلیل فضای کاربر پردازه را انجام می‌دهیم. از اطلاعات استخراج‌شده از این ساختار به‌ویژه ردیابی DLLها و سپس ردیابی APIها[2] (رابط برنامه‌نویسی کاربردی) برای ایجاد یک الگوی رفتاری از فراخوانی‌های بدافزار و همچنین تأیید تغییرات ایجادشده در سیستم توسط بدافزار استفاده می‌کنیم.

۳- مروری بر کارهای مرتبط

استفاده از قابلیت‌های روش‌های تشخیص بدافزار مبتنی بر جرم‌شناسی حافظه در سال‌های اخیر مورد توجه بوده است. این روش برخلاف روش‌های تحلیل پویا، به‌صورت مستقیم و بدون استفاده از واسط سیستم‌عامل به پویش محتویات حافظه و ردیابی اثرات مخرب بدافزار می‌پردازد. پژوهش‌های جرم‌شناسی اخیر بر بازیابی اشیا از حافظه فضای هسته متمرکز بودند [۱۶]، این اشیا برای هر پردازه در حال اجرا بر روی سیستم وجود دارند و به‌طور مستدل بین نسخه‌های مختلف سیستم‌عامل مشابه هستند؛ بنابراین امکان توسعه سریع روش‌هایی برای به‌دست آوردن بخش‌های کلیدی اطلاعات فراهم می‌شود؛ اما این روش‌ها، استخراج اطلاعات دیگر موجود در حافظه، مانند حافظه فضای کاربر را آسان نمی‌کنند. بسیاری از پژوهش‌ها به‌منظور بررسی ساختارها، مثل پردازش‌ها، نخ‌ها و منابع شبکه انجام شده است که صرفاً به توصیف این ساختارها در حافظه پرداخته‌اند [۱۷]. همچنین روش‌هایی برای بازیابی فایل‌های نگاشته در حافظه و ارتباط اطلاعات این فایل‌ها با داده‌های پردازه‌ها در [۱۸] ارائه شده است. با استفاده از این روش‌ها می‌توان میزان اطلاعات ناشناخته در یک تصویر حافظه را کاهش داد و اطلاعات مربوط به پردازه‌ها را شناسایی کرد.

تمرکز روش‌های جرم‌شناسی حافظه موجود مانند Malfind [۱۹] بر مکان‌یابی اثرات مربوط به روش‌های مشترک مورداستفاده بدافزارها، از جمله تزریق کد است. این افزونه فقط مجموعه‌ای متداول از روش‌های تزریق کد و همچنین معمول‌ترین مجوزهای صفحه که به تزریق کد مرتبط هستند را بررسی می‌کند؛ بنابراین قدرت تشخیص آن محدود است. نمونه‌ای از تزریق کد مخرب بدون استفاده از مجوزهای در نظرگرفته شده در این افزونه در منبع [۲۰] مطرح‌شده است. مجموعه‌ای از گزارش‌های تحلیل حافظه در پروژه مستندسازی volatility [۲۱] ارائه‌شده که به تحلیل نمونه بدافزارهای خاص پرداخته است. روش‌های دیگری با استفاده از روش‌های جرم‌شناسی برای شناسایی روت‌کیت ها در حوزه‌های دانشگاهی و تجاری مورد بحث قرارگرفته‌اند. همچنین تعدادی ابزار تحلیل حافظه رایگان وجود دارند که قادرند آلوده شدن سیستم به روت‌کیت را مشخص کنند. نمونه‌های مشهور عبارت‌اند از GMER [۲۲]، Revealer Rootkit [۲۳] و IceSword [۲۴]. یکی از مشهورترین روش‌های شناسایی روت‌کیت با تحلیل تصویر حافظه، تشخیص Cross-view می‌باشد. عملکرد این روش براساس جمع‌آوری اطلاعات یکسان از ساختارهای مختلف حافظه می‌باشد. در [۸] افزونه‌ای به نام Rkfinder ارائه شده است که برخی از افزونه‌های چهارچوب Volatility را که از روش Cross-view استفاده می‌کنند در یک واسط کاربری گرافیکی مجتمع می‌کند.

برخلاف روش‌های فوق در روش پیشنهادی به‌جای ردیابی شواهد روش‌هایی مانند تزریق کد، تشخیص روت‌کیت براساس ناسازگاری‌های ایجادشده در ساختارهای داده و استخراج اطلاعات خاص مربوط به پردازه‌ها مانند فایل‌های نگاشت‌شده در حافظه به دنبال ردیابی کد و سپس عملکرد اصلی آن با استخراج توابع فراخوانی‌شده، در فضای کاربر حافظه می‌باشیم. نتیجه این روش تحلیل، استخراج ویژگی‌های رفتاری از کد بدافزار می‌باشد، که عمومیت بیشتری نسبت به روش‌های قبلی دارد درنتیجه قدرت تشخیص بیشتر خواهد بود. به‌عبارت دیگر در این روش، تحلیل رفتار کد بدافزار قرار گرفته در فضای حافظه امکان‌پذیر می‌شود و عملکرد اصلی بدافزار با استخراج فراخوانی‌ها ردیابی می‌شود که در روش‌های قبلی به این روش استخراج ویژگی پرداخته نشده است. باید به این مسئله توجه داشت که بررسی محتویات حافظه پردازش برای استخراج و استنتاج رفتار، بدون استفاده از مهندسی معکوس به مدلی نیاز دارد که چگونگی عملکرد فضای کاربر را مشخص کند. روش ارائه‌شده مدلی را پیشنهاد می‌کند که با استفاده از ساختار توصیف‌کننده آدرس مجازی همه تخصیص‌های حافظه مربوط به یک پردازش را شناسایی می‌کند. هدف از بررسی این تخصیص‌ها استخراج کد قرارگرفته در حافظه توسط بدافزار و بررسی عملکرد آن می‌باشد. همچنین درروش پیشنهادی با به‌کارگیری ساختارهای دیگر، دامنه استخراج ویژگی را افزایش دادیم تا ضمن ردیابی رفتار

1- Dynamic Link Library
2- Application Programming Interface

کامل‌تر دقت تشخیص را افزایش دهیم. به‌عبارت دیگر از روش Cross-view برای افزایش دقت اطلاعات جمع‌آوری‌شده، استفاده گردیده است.

در انتها برای اثبات مؤثر بودن ویژگی‌های استخراج شده از الگوریتم‌های داده‌کاوی استفاده کردیم که خروجی این الگوریتم‌ها نرخ تشخیص قابل قبولی را ارائه می‌دهند. در روش‌های مشابه قبلی به تحلیل مجموعه‌ای از بدافزارها و سپس استفاده از روش‌های داده‌کاوی برای آزمون مجموعه ویژگی‌های استخراج‌شده در تشخیص بدافزارهای ناشناخته پرداخته نشده است.

۴- روش‌های ضد تحلیل حافظه و روش‌های مقابله با آن‌ها

جرم‌شناسی حافظه شامل: ۱- دست‌یابی به محتویات حافظه فرّار یک سیستم در حال اجرا (اثرات دیجیتال) و سپس، ۲- بررسی این محتویات ذخیره‌شده در یک سیستم مورد اعتماد است [۲۵]. هر نوع تلاش برای به خطر افتادن دسترس‌پذیری یا مؤثر بودن اثرات در فرایند تحلیل حافظه، به‌عنوان یک روش ضد تحلیل شناخته می‌شود [۲۶]. روش‌هایی وجود دارند که هدف آن‌ها شکستن فرایند تحلیل یا ارسال داده‌های اشتباه به فرایند ضبط محتویات حافظه است [۲۷ و ۲۸]. این روش‌ها در دودسته قرار می‌گیرند. دسته اول، در زمان ذخیره محتویات حافظه اختلال ایجاد می‌کنند. دسته دوم، با دست‌کاری ساختارهای حافظه تلاش می‌کنند تا فرایند تحلیل بی‌نتیجه باشد. عموماً محتویات حافظه را به دو روش می‌توانیم ذخیره کنیم. (۱) نصب یک ابزار در سیستم‌عامل برای ذخیره محتویات حافظه (۲) استفاده از یک روش یا ابزار خارج از محیط سیستم‌عامل برای ذخیره تصویر حافظه [۲۹].

در فرایند ذخیره محتویات، تضمین صحت اطلاعات جمع‌آوری‌شده نیز باید انجام شود. بسیاری از تحقیقات در این زمینه انجام‌شده و اثبات می‌کنند که اثرات جانبی اجرای یک ابزار داخلی برای جمع‌آوری محتویات حافظه اندک است [۳۰]. در تحقیقات اخیر حملاتی برای ایجاد اختلال در عملکرد این ابزارها پیشنهادشده‌اند. به‌عبارت‌دیگر وقتی پردازه مخرب حضور این ابزارها را تشخیص می‌دهد، روش‌هایی را برای متوقف کردن یا فریب آن‌ها به‌کار می‌گیرد. ازجمله روش‌های مطرح‌شده، حملات جانشینی[1] هستند. در این حملات داده‌های ایجادشده توسط بدافزار در طول فرایند ذخیره حافظه جایگزین داده‌های معتبر می‌شوند [۳۱]. حمله دیگر روت‌کیت‌ها هستند که با قلاب‌شدن به توابع سیستم عامل فرایند ضبط اطلاعات را تشخیص می‌دهند و آن را مختل می‌کنند (مثلاً قفل‌کردن سخت‌افزار[2]). درنتیجه با توجه به فرّار بودن این اثرات، امکان دوباره ذخیره کردن محتویات در شرایط یکسان وجود ندارد.

روش‌های ضد تحلیل، در فرایند تحلیل اختلال ایجاد کرده و یا باعث کاهش عملکرد فرایند ذخیره محتویات می‌شوند؛ بنابراین وجود اثراتی همچون کاهش عملکرد و ناکامل ماندن تحلیل، وقوع یک حمله را نشان می‌دهند [۱۵]. تلاش‌هایی برای آزمودن ابزارهای ذخیره محتویات حافظه انجام شده است [۳۲]. نتایج نشان می دهد که در شرایط ایده‌ال آزمایشگاهی می‌توانیم به صحت این ابزارها اعتماد کنیم. روش‌های دیگری نیز برای جمع‌آوری محتویات، بدون اجرای ابزار خاصی در سیستم هدف وجود دارد. پژوهش‌هایی درزمینه روش‌های سخت‌افزاری امن‌تر مانند دسترسی مستقیم به حافظه انجام‌شده است [۳۵-۳۳]. در روش‌های سخت‌افزاری ایجاد اختلال در فرایند ذخیره‌سازی سخت‌تر است. همچنین با توجه به گسترش استفاده از روش‌های مجازی‌سازی[3] و پیاده‌سازی سیستم‌ها در ماشین مجازی، امکان ذخیره یک تصویر حافظه از سطح ناظر ماشین مجازی[4] بدون هیچ تداخلی با سیستم هدف وجود دارد. با توجه به استفاده گسترده از مجازی‌سازی در پیاده‌سازی سیستم‌ها، استفاده از تصویر حافظه ماشین مجازی یکی از روش‌های ایده‌آل پیشنهادی است. به‌دلیل افزایش محبوبیت تحلیل حافظه، اکنون تحقیقات زیادی درزمینه توسعه روش‌های ذخیره تصویر حافظه امن انجام شده است. در [۲۹] پس از مروری بر روش‌های موجود، روش جدیدی مبتنی بر مدیریت مستقیم حافظه و بررسی سخت‌افزار PCI[5] بدون اتکا بر عملکرد سیستم‌عامل ارائه شده است. در تحلیل بدافزار با روش‌های تحلیل حافظه، می‌توان از فایل Hiberfile.sys در ویندوز نیز استفاده کرد. عموماً وقتی سیستم به‌حالت خواب زمستانی[6] می‌رود، این فایل ایجاد می‌شود. محتویات حافظه (وضعیت سیستم) در این فایل ذخیره می‌شود. به‌این‌ترتیب هر پردازه مخرب به ویژه پردازه‌هایی که در پس‌زمینه[7] در حال اجرا باشند را می‌توان در این فایل ردیابی کرد.

هدف حملاتی که با حذف کردن اشاره‌گرها از ساختارهای حافظه عمل می‌کنند، ایجاد اختلال در استخراج شواهد از محتویات جمع‌آوری شده است. روت‌کیت‌های سطح کاربر به‌راحتی ساختارهای فضای کاربر را تغییر می‌دهند. روت‌کیت‌های DKOM با دست‌کاری مستقیم اشیاء هسته تلاش می‌کنند تا فعالیت‌های مخرب را پنهان کنند [۱۶]. تغییر ساختارها با این روت‌کیت‌ها اگر به‌درستی توسط

1- Substitution

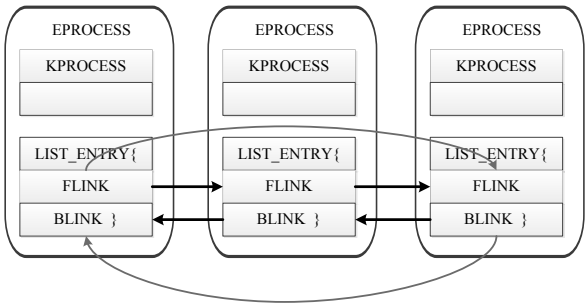
2- Hang The Hardware
3- Virtualization
4- Virtual Machine Monitor (VMM)
5- Peripheral Component Interconnect
6- Hibernation
7- Background

بدافزار به‌کار رود، مکان‌یابی داده‌ها در حافظه را مشکل می‌کند. بااین‌وجود روش‌هایی وجود دارند که با پویش[1] بلوک‌های حافظه، الگوهای خاص (به‌عنوان‌مثال، یک الگو از بایت‌ها برای یک نوع فایل منحصربه‌فرد خواهد بود) را شناسایی می‌کنند. به‌عبارت دیگر فرض کنید که اشاره‌گر به یک فایل ذخیره‌شده در دیسک از سیستم فایل حذف‌شده باشد. از طریق پویش محتویات دیسک و یافتن بخش‌هایی که با الگوی فایل موردنظر ما انطباق دارند، می‌توانیم داده‌های این فایل را بازیابی کنیم. حتی DLLهایی که برای مخفی کردن خود از ساختار VAD با روش Reflective DLL Injection (روشی برای تزریق[2] فایل‌های کتابخانه‌ای در حافظه که نگاشتی در VAD ایجاد نمی‌شود [۱۵]) تزریق می‌شوند را می‌توان با پویش بلوک‌های حافظه و سیاست‌های محافظت صفحه ردیابی کرد. بر اساس مطالعات ما، بهترین ساختار برای مکان‌یابی DLLهایی مخفی‌شده و تزریق‌شده، ساختار VAD است (ما در فرایند استخراج اثرات، تا حد امکان سعی کردیم DLLهای تزریق‌شده و مخفی‌شده در فضای پردازه‌ها را از ساختار VAD ردیابی کنیم). VAD یک گروه‌بندی منطقی از صفحات حافظه را ارائه می‌دهد. روش‌های DKOM با جدا کردن گره‌های VAD مربوط به یک پردازه، مانع از ردیابی این گره‌ها می‌شوند. ولی صفحات اختصاص داده‌شده به پردازه در جدول صفحه باقی‌مانده و قابل بررسی خواهند بود. حتی اگر فرض کنیم که بدافزار بتواند با حداقل تأثیر بر عملکرد، روش‌های ضد تحلیل را به‌طور کامل از سطح کاربر تا ساختار VAD به‌درستی پیاده‌سازی کند، از طریق بررسی ورودی‌های جدول صفحه، بازسازی درخت VADها امکان‌پذیر است.

معمول‌ترین روش مورد استفاده بدافزارها در تحلیل حافظه دست‌کاری ساختارها به‌منظور پنهان کردن شواهد است. به‌عنوان مثال به‌دست آوردن بلوک EPROCESS پردازه هدف و سپس جدا کردن آن از فهرست با تغییر دو اشاره‌گر انجام می‌شود (شکل ۲). ولی فعالیت‌های یک پردازه را می‌توان در هرجایی از سیستم ردیابی کرد و حذف همه این شواهد و حذف اشاره‌گرها از همه ساختارها غیرممکن است. به‌عبارت دیگر همیشه شناسایی اثرات و ورودی‌های پیش‌بینی‌نشده در فهرست‌های دیگر امکان‌پذیر است.

مثلاً طبق بررسی‌های ما، برای ردیابی Handleهای مربوط به یک پردازه، حداقل از ۷ ساختار (به‌عنوان مثال، جدول Handle پردازه، جدول PSPCID مربوط به نخ، جدول PSPCID مربوط به پردازه، جدول Handle مربوط به Csrss.EXE و...) می‌توان استفاده کرد. اگر ردیابی Handleها از طریق بررسی ساختار EPROCESS انجام نشود، مخفی کردن آن‌ها مشکل خواهد بود. در ادامه سه روش ضد تحلیل

1- Scanning/Carving
2- Injection



شکل (۲). مخفی کردن پردازه از ساختار EPROCES با روش DKOM

حافظه مطرح شده در سال‌های اخیر را بررسی می‌کنیم.

طبق بررسی‌های دقیق این پژوهش، تاکنون روش‌های ضد تحلیل ساختار رجیستری مطرح‌شده، به عدم انعکاس تغییرات در دیسک پرداخته‌اند. روش پیاده‌سازی‌شده در ابزار کوچک SetRegTime [36]، فقط امکان تغییر برچسب LastWriteTime کلیدهای رجیستری را فراهم می‌کند. هدف ابزار ارائه‌شده روشن کردن این واقعیت است، که تغییر این برچسب زمانی در کلیدهای رجیستری کار سختی نیست. می‌توانیم نتیجه بگیریم که اعمال روش‌های ضد تحلیل به دانش مهاجم وابسته است؛ اما به‌کارگیری این روش‌ها به‌طور جامع غیرممکن است و همیشه اثرات ناشی از به‌کارگیری آن‌ها قابل پیگیری است.

ابزار ضد تحلیل حافظه ADD[1] در [۳۷]، برای منحرف کردن مسیر تحلیل، اشیاء جعلی یا تله در حافظه ایجاد می‌کند. یکی از محدودیت‌های اشیا جعلی ایجادشده توسط ADD این است که آن‌ها مرتبط با هیچ پردازه‌ای نیستند. از طرف دیگر عملکرد این اشیا برخلاف روش‌های اختفا مورد استفاده بدافزارها است. در حال حاضر، ADD فقط می‌تواند اشیاء جعلی ضعیف در یک نسخه از ویندوز (۷- ۳۲ بیت) ایجاد کند و پشتیبانی از نسخه‌های دیگر نیازمند مطالعات و بررسی‌های بیشتری است. در روش پیشنهادی، نتیجه‌گیری ما صرفاً براساس وجود اثرات یا تغییرات ایجادشده در یک ساختار نیست. هدف ما ردیابی تغییرات یا ناهنجاری‌های ایجادشده در سیستم مرتبط با یک پردازه خاص است. ما با درگیر کردن چندین ساختار در استخراج اطلاعات، ناسازگاری‌های ناشی از به‌کارگیری روش‌های ضد تحلیل حافظه را ردیابی می‌کنیم. مزیت روش پیشنهادی، استخراج اطلاعات مؤثر در ارتباط با پردازه مخرب، ردیابی روش‌های ضد تحلیل حافظه و درنتیجه کاهش نرخ مثبت کاذب خواهد بود.

نظریه دیگری در ابزار ضد تحلیل حافظه Dementia به منظور

1- Attention-Deficit-Disorder

پنهان کردن اثرات مختلف از تصویر حافظه ضبط‌شده در سیستم‌عامل ویندوز ارائه شد [۳۸]. این ابزار با بهره‌برداری از ابزارهای ضبط محتویات حافظه داخلی، اثرات موجود در سیستم (به‌عنوان مثال، پردازه‌ها و نخ‌ها و ...) را از ابزارهای تحلیل حافظه پنهان می‌کند. Dementia با قلاب‌شدن به توابع سیستم عامل ابزارهای ضبط را تشخیص می‌دهد. پنهان نکردن کلیدهای رجیستری و مقادیر آن‌ها، اتصالات شبکه و برخی از فراخوانی‌های فایل‌های کتابخانه‌ای از محدودیت‌های این ابزار است. برای شناسایی Dementia، می‌توانیم قبل از شروع فرایند ذخیره محتویات حافظه از روش‌های تشخیص روت‌کیت استفاده کنیم. تغییر یک بایت به‌منظور شکست دادن فرایند تحلیل، روشی است که در [۳۹] پیشنهاد شده است. این ابزار با بارگذاری یک راه‌انداز دستگاه در هسته، یک بایت از ساختارهای داده را تغییر می‌دهد، تا ابزارهای تحلیل نتوانند این ساختارها را شناسایی کنند. این ابزار هم فقط ساختارها و ابزارهای تحلیل خاصی را هدف قرار می‌دهد و تغییرات ناچیزی را در حافظه هدف ایجاد می‌کند.

به‌کار بردن روش‌های ضد تحلیل حافظه برای مخفی کردن فعالیت‌های مخرب توسط بدافزار، به‌عنوان یک مانع اساسی در طول فرایند تحلیل مطرح هستند. وقتی تحلیل ما به بررسی اثرات در یک محدوده خاص بپردازد، تأثیر این روش‌ها بیشتر خواهد بود. اگر تحلیل به‌صورت جامع انجام شود و شامل درگیر بودن ترکیبی از منابع داده باشد، می‌توانیم اختلال ایجادشده توسط روش ضد تحلیل را شناسایی کنیم. واقعیت این است که بدافزار نمی‌تواند با استفاده از روش‌های ضد تحلیل به‌طور کامل اثرات اجرایی خود را پنهان کند. همواره اثرات ناشی از به‌کارگیری این روش‌ها و ناسازگاری‌های ایجادشده در سیستم قابل‌پیگیری است. مثلاً بدافزار Flame در فایل Shell32.DLL تزریق می‌شود. Shell32.DLL یکی از DLLهایی است که در اکثر نسخه‌های ویندوز وجود دارد. اگر این DLL به‌صورت معمولی (از طریق فراخوانی LoadLibrary) بارگذاری شود در فهرست‌های PEB قابل ردگیری است؛ اما Flame اثرات بارگذاری این DLL را از این فهرست‌ها مخفی می‌کند؛ بنابراین مشاهده این ناسازگاری، دلیلی برای ردیابی رفتار این فایل خواهد بود.

با توجه به مشکلات موجود، هدف ما در پویش‌گر پیشنهادی، استخراج اثرات مشکوک به‌گونه‌ای است که روش‌های ضد تحلیل بدافزارها را ردیابی کنیم. به‌عبارت‌دیگر الگوی رفتاری بدافزار را با در نظر گرفتن این قابلیت‌های دفاعی استخراج‌کنیم. ما با بررسی عملکرد روش‌های ضد تحلیل حافظه ارائه‌شده در رجیستری و روش‌های مخفی کردن فایل‌های بارگذاری‌شده توسط بدافزارها، به استخراج اثرات مربوط به این ساختارهاپرداخته‌ایم. برای تعمیم روش پیشنهادی می‌توانیم تغییرات ایجادشده در ساختارهای دیگر را نیز به همین روش بررسی کنیم. هدف ما درگیر کردن چندین ساختار داده در تحلیل به‌منظور ارزیابی و تأیید نتیجه استخراج شده است.

۵- پویش‌گر پیشنهادی: ردیابی تغییرات رجیستری، فراخوانی فایل‌ها و توابع در حافظه

یکی از روش‌هایی که بدافزار تنظیمات خود را برای مقیم‌شدن[1] در سیستم انجام می‌دهد از طریق رجیستری است. بررسی رفتارهای رجیستری بدافزار در حافظه به چند دلیل حائز اهمیت است. بدافزار می‌تواند کلیدهای رجیستری را در حافظه تغییر دهد تا به اهداف تعیین‌شده خود دست یابد و مانع از انعکاس این تغییرات در فایل‌های متناظر در دیسک شود. ارزیابی‌های عملی ما نشان می‌دهد که حملاتی که برای مخفی کردن اثرات رجیستری پیشنهادشده، مربوط به عدم انعکاس شواهد در دیسک است و این اثرات از تصویر حافظه قابل‌ردیابی است. دلیل دیگر این‌که برخی شاخه‌ها[2](یک گروه منطقی از کلیدها، زیر کلیدها و مقادیر در رجیستری) و کلیدهای رجیستری فقط در حافظه ایجاد می‌شوند و متناظر با آن‌ها هیچ فایلی در دیسک وجود ندارند. سیستم این شاخه‌ها را در حافظه ایجاد و مدیریت می‌کند. هر شاخه در حافظه با ساختار داده CMHIVE بارگذاری می‌شود. این ساختار شامل فراداده‌های[3] متعدد در مورد شاخه، مانند مسیر کامل، تعداد Handelهای باز و اشاره گر به دیگر شاخه‌های بارگذاری شده است [۴۰].

بدافزارها وقتی یک سیستم را آلوده می‌کنند، برای جلوگیری از دوباره آلوده کردن سیستم، در اولین انتشار نشانه‌هایی را در سیستم قرار می‌دهند. این نشانه‌ها در یک خانواده بدافزار ثابت هستند یا بر اساس الگوریتم خاصی ایجاد می‌شوند. از این ویژگی می‌توان به‌عنوان یک روش پیشگیری هم استفاده کرد. مکان‌های مختلفی وجود دارد که بدافزار این نشانه‌ها را ذخیره می‌کند. نمونه‌های بررسی شده در [۴۱] نشان می‌دهد که بعد از Mutexها (این اشیا[4] در حافظه ایجاد می‌شوند و روش‌هایی برای استخراج آن‌ها وجود دارد) کلیدهای رجیستری متداول‌ترین نشانه‌ها هستند. به‌عنوان مثال، استاکس‌نت از یک کلید رجیستری به‌عنوان نشانه استفاده می‌کند، که در زمان انتشار وجود این کلید را بررسی می‌کند [۱].

وقتی استخراج اثرات مخرب[5] از رجیستری مدنظر باشد، ایجاد

1- Reside
2- Hive
3- MetaData
4- Object
5- Indicators of Compromise (IOC)

یک حالت پایه از وضعیت رجیستری مهم‌ترین بخش است. بعد از راه‌اندازی سیستم تحلیل، می‌توانیم وضعیت اولیه رجیستری را ذخیره کنیم. همچنین با بررسی تغییرات در رجیستری سیستم‌های مختلف، می‌توانیم تغییراتی را که معمولاً به‌طور منظم توسط سیستم‌عامل انجام می‌شوند را به سیستم آموزش بدهیم. به‌این ترتیب در زمان تحلیل، اثرات ناشی از این تغییرات حذف می‌شوند. همچنین اگر بدانیم که چه کلیدهایی را باید بررسی کنیم، حجم اطلاعاتی که باید بررسی شود کاهش می‌یابد، درنتیجه فرایند تحلیل ازنظر زمانی قابل‌قبول‌تر خواهد بود. ما کلیدهایی را که برای بررسی انتخاب کرده‌ایم در ۱۰ دسته قرار داده‌ایم، این دسته‌ها کلیدهایی هستند که معمولاً توسط بدافزارها استفاده شده‌اند. مهم‌ترین دسته‌ها در دو گروه طبقه‌بندی می‌شوند، دسته اول کلیدهایی که امکان اجراشدن یک برنامه به‌صورت خودکار را فراهم می‌کنند و دسته دوم کلیدهای مربوط به ذخیره کارهای برنامه‌ریزی‌شده [1] (کارهایی که در زمان یا تاریخ یا شرایط خاصی باید در سیستم اجرا شوند) هستند. نکته دیگری که در استخراج کلیدها در نظر گرفتیم، برچسب زمانی Lastwrite است، که ویندوز برای هر کلید در رجیستری ذخیره می‌کند. این برچسب آخرین زمانی که کلید تغییر کرده است را نمایش می‌دهد. به‌صورت کلی تغییر زیر کلیدها بر روی برچسب زمانی کلید پدر بی‌تأثیر است. به‌عبارت‌دیگر برچسب زمانی برای کلیدها وجود داشته و برای مقادیر وجود ندارد. ما با در نظر گرفتن زمان انجام تحلیل، استخراج کلیدهای موجود در حافظه و بررسی این برچسب، تغییرات را شناسایی می‌کنیم. بررسی Timelineها امکان ردیابی فعالیت‌هایی که اخیراً در سیستم رخ‌داده و چگونگی آلوده شدن سیستم را فراهم می‌کند. برخی اطلاعات زمانی قابل‌استخراج از تصویر حافظه شامل زمان سیستم، برچسب زمانی پردازه، نخ، اتصالات شبکه، گزارش‌های [2] ویندوز، کلیدهای رجیستری و برچسب زمانی قالب PE [3] است. بدافزار می‌تواند برچسب زمانی فایل‌ها را تغییر دهد یا با تغییر کلید رجیستری مربوطه مانع به‌روزرسانی برچسب زمانی فایل‌ها شود. این رفتارها با بررسی فراخوانی توابع و تغییرات رجیستری قابل‌شناسایی است.

همه بدافزارها از رجیستری برای ذخیره تنظیمات استفاده نمی‌کنند. به‌عنوان مثال بدافزار W32/Crimea به‌منظور مقیم شدن در سیستم، فایل Imm32.DLL را تغییر می‌دهد [۴۲]. بررسی اثرات مربوط به فایل‌ها نقش مهمی در تحلیل رفتار بدافزار دارد. در تحلیل حافظه تمرکز اصلی بر دودسته اصلی فایل‌ها، فایل داده [4] و فایل اجرایی [5] است. فضای آدرس پردازه عمدتاً شامل فایل‌های بارگذاری‌شده از دیسک (کد یا داده موردنیاز برای اجرای پردازه)، فایل‌های اجرایی قابل‌حمل (EXEs) و فایل‌های کتابخانه‌ای پیوند پویا (DLLs) هستند. هدف ما استخراج اثرات مرتبط با فراخوانی‌های انجام‌شده توسط فایل‌های EXE و DLL از حافظه [6] مجازی پردازه مخرب است. این EXEها و DLLها فایل‌های نگاشت‌شده در حافظه هستند که به‌وسیله ساختار FILE_OBJECT در حافظه هسته نمایش داده می‌شوند [۴۳]. این ساختار شامل چندین عضو است و بررسی آن‌ها برای استخراج رفتار پردازه‌ها حائز اهمیت است. FILE_OBJECTها را می‌توان در دو مکان مستقل مکان‌یابی کرد. ردیابی در جدول Handelهای پردازه و درخت VADها. درروش پیشنهادی، از هر دو ساختار برای جمع‌آوری اطلاعات استفاده می‌شود.

به‌منظور تغییر (خواندن/نوشتن) کلیدهای رجیستری و همچنین فایل‌ها، پردازه ابتدا یک Handle برای کلید یا فایل موردنظر با فراخوانی API مربوطه ایجاد می‌کند. مقدار این Handle به پردازه فرستاده می‌شود. سپس برای دست‌کاری محتوا، APIهای دیگری فراخوانی می‌شوند که این Handle را به‌عنوان پارامتر می‌پذیرند. این Handle تا وقتی‌که توسط پردازه بسته نشود یا پردازه خاتمه پیدا نکند، در حافظه فعال است [۱۵]. پس اگر کد بدافزار به‌درستی عمل کند، مدت‌زمان ردیابی این Handleها محدود است؛ اما APIهای فراخوانی شده مدت‌زمان بیشتری (حتی تا خاتمه پردازه) قابل‌استخراج هستند. نتایج تجربی در آزمایش‌های ما نشان می‌دهد که برخی اطلاعات موجود در حافظه در مدت‌زمان طولانی‌تری نسبت به زمان موردنظر ما باقی می‌مانند.

علاوه بر لحاظ کردن موارد فوق، تمرکز ما بر تأیید اعتبار تغییرات مخرب بر اساس ردیابی فراخوانی‌های انجام‌شده توسط بدافزار است. هر نوع بدافزار به دنباله خاصی از فراخوانی‌های توابع وابسته است. ما APIهای فراخوانی شده توسط پردازه مخرب را با بررسی فایل‌های اجرایی و کتابخانه‌های بارگذاری‌شده و همچنین ساختار VAD استخراج کردیم. VAD، ساختار مناسبی برای ردیابی APIهایی است که از طریق DLLهای تزریق‌شده و مخفی‌شده فراخوانی می‌شوند. گره‌های VADمربوط به پردازه مخرب با پیمایش درخت VADها استخراج می‌شوند. سپس گره‌ها برحسب برچسب (MMVAD_SHORT، MMVAD و MMVAD_LONG) دسته‌بندی می‌شوند. اگر گره شامل مسیر فایل اجرایی باشد، APIهای فراخوانی‌شده استخراج می‌شوند. در غیر این‌صورت با بررسی محتوای گره‌های با مجوز صفحه اجرایی، فراخوانی‌ها (ردیابی

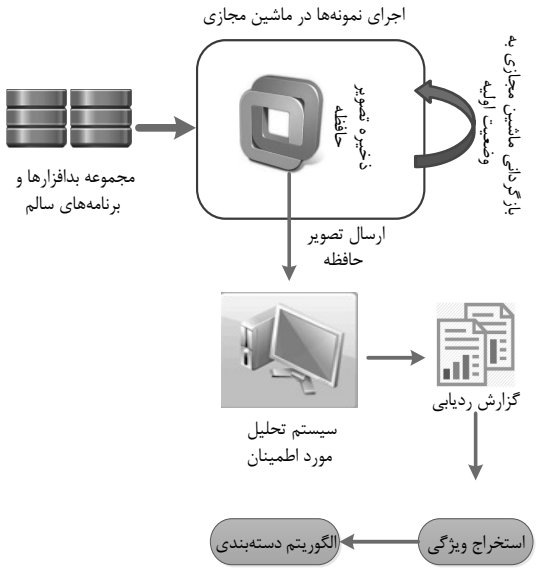
1- Scheduled Task
2- Logs
3- Portable Executable
4- Data File
5- Executable File
6- Memory Mapped File

برخی دستورات CALL و JMP) را استخراج می‌کنیم. از این طریق می‌توانیم کد بدافزار (کد اصلی بعد از اتمام فرایند رمزگشایی) را در فضای حافظه شناسایی و استخراج‌کنیم. در شکل (۳) قسمتی از گزارش استخراج‌شده از گره‌های VADیک پردازه مخرب و فراخوانی‌های متناظر با آن نمایش داده‌شده است.

ابزارهایی که برای استخراج اثرات از تصویر حافظه وجود دارند، امکان استخراج اثرات یا ساختارهای سطح پایین را، مستقل از یکدیگر فراهم می‌کنند [۴۵ -۴۴]. هدف همبسته‌سازی[1] این اثرات برای تأیید اعتبار تغییرات و ایجاد یک الگو از رفتارهای بدافزار است. عملکرد روش ارائه‌شده در ابزار SetRegTime درروش پیشنهادی ما قابل‌شناسایی است. درواقع در زمان تحلیل با ردیابی فراخوانی های توابع استفاده‌شده توسط این ابزار (APIهای محلی در Ntdll.DLL) و تغییرات کلیدهای رجیستری موجود در حافظه این فعالیت قابل‌شناسایی است. APIهای سرویس‌های محلی ویندوز، مجموعه توابعی هستند که برای اجرا در حالت هسته پیاده‌سازی شده‌اند. نام این توابع با پیشوند Nt یا Zw شروع می‌شود. درایورهای سطح هسته می‌توانند این توابع را مستقیماً فراخوانی کنند و برنامه‌های سطح کاربر با استفاده از فراخوانی‌های سیستمی به این توابع دسترسی دارند. برخی از APIهای محلی در حالت کاربر به‌صورت مستقیم در Ntdll.DLL پیاده‌سازی شده‌اند؛ اما اکثر APIهای محلی در Ntoskrnl.EXE هستند و از طریق Ntdll.DLL در حالت کاربر مورداستفاده قرار می‌گیرند [۴۷ -۴۶]. درروش پیشنهادی فراخوانی‌های بدافزار تا سطح Ntdll.DLL ردیابی شده‌اند.

۶- مراحل تحلیل و تشخیص

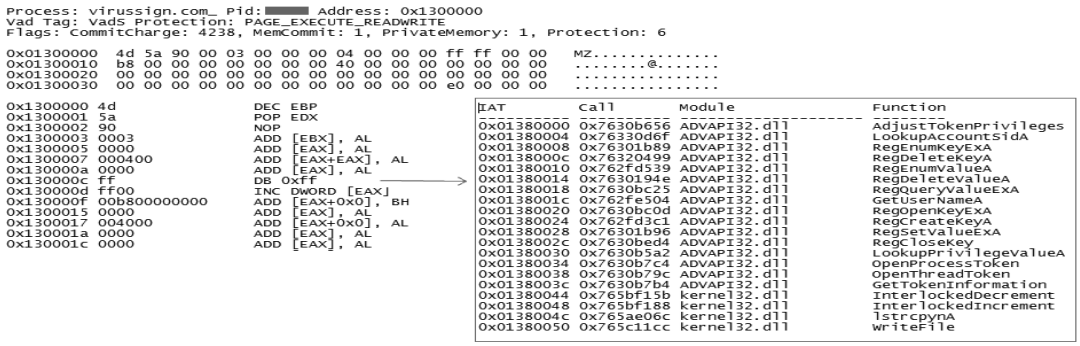
در این پژوهش، با استفاده از روش تحلیل محتوای حافظه، فراخوانی‌های API، دسترسی به فایل‌های کتابخانه‌ای و تغییرات رجیستری در حافظه ردیابی شده است. محیط تحلیل، شامل یک ماشین مجازی VMware9 با سیستم‌عامل ویندوز ۷ و قابلیت اتصال به اینترنت می‌باشد (شکل ۴). پس از اجرای هر نمونه، تصویر حافظه با فاصله زمانی حداکثر ۵ دقیقه‌ای ذخیره می‌شود. در بررسی برخی نمونه‌ها به‌منظور مشاهده رفتار کامل‌تر لازم شد فرایند ذخیره محتویات حافظه با یک‌فاصله زمانی، چندین بار تکرار شود. در جدول (۱) مراحل تحلیل و تشخیص بیان شده است.



شکل (۴). محیط اجرای نمونه‌ها و جریان استخراج ویژگی‌ها

جدول (۱). الگوریتم اجراشده برای جمع‌آوری و ردیابی اثرات

گام (۱): دستیابی به محتویات حافظه	
۱	ذخیره یک تصویر حافظه از وضعیت اولیه ماشین مجازی
۲	بازگردانی ماشین مجازی به وضعیت اولیه (محیط تحلیل برای همه نمونه‌ها یکسان است)
۳	انتقال نمونه به ماشین مجازی (فرایند می‌تواند خودکار انجام شود)
۴	اجرای نمونه
۵	متوقف کردن ماشین مجازی
۶	ذخیره تصویر حافظه ماشین مجازی
۷	تکرار مراحل ۲ تا ۶ برای همه نمونه‌ها
گام (۲): فرایند استخراج	
۸	استخراج اثرات مربوط به پردازه موردنظر (سالم یا مخرب)
۹	استخراج ویژگی‌ها بر اساس تغییرات رجیستری، بارگذاری فایل‌های کتابخانه‌ای و فراخوانی‌های API از پردازه‌ها
۱۰	انتخاب ویژگی و استفاده از آن‌ها در الگوریتم یادگیری و دسته‌بندی نمونه‌های جدید



شکل (۳). گره VAD شامل قطعه کد اجرایی معتبر و فراخوانی‌های متناظر با آن

1- Correlation

۷- مدل پیشنهادی برای تحلیل تصویر حافظه و انتخاب ویژگی

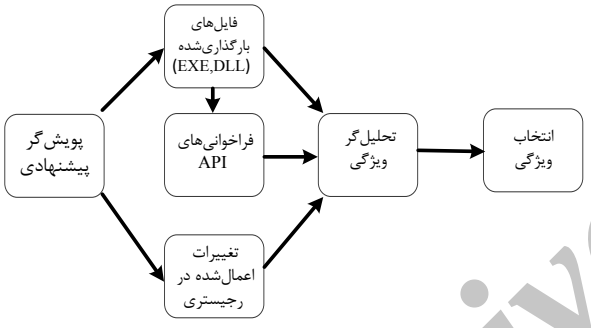
تأکید ما بر استخراج اثرات به روشی است که بدافزارهای باقابلیت‌های دفاعی پیشرفته قابل‌تشخیص باشند. همان‌طورکه در بخش‌های قبل توضیح داده شد، هدف ما دسته‌بندی بدافزارها بر اساس استخراج اثرات با پرداختن به ناسازگاری‌های ناشی از کاربرد روش‌های دفاعی (ضد تحلیل و مبهم‌سازی) است. مجموعه نمونه‌ها شامل دودسته، فایل‌های اجرایی سالم و بدافزارها می‌باشند. تعداد ۶۵۰ نمونه از بدافزارهای موجود در سایت virussign.com [۴۸] و نسل‌های مختلف یک بدافزار در سایت vxheaven.org [۴۹] انتخاب شدند. فایل‌های سالم نیز شامل ۳۵۰ نمونه، که از فایل‌های اجرایی نسخه‌های مختلف سیستم‌عامل ویندوز و برنامه‌های کاربردی جمع‌آوری شدند. درروش پیشنهادی ابتدا از میان مجموعه گزارش‌های رفتاری جمع‌آوری‌شده، تغییرات انجام‌شده در رجیستری و فراخوانی‌های API و فایل‌های کتابخانه‌ای از ساختارهای مختلف استخراج می‌شوند. این اطلاعات به‌دست‌آمده باهم مقایسه می‌شوند تا ضمن ردیابی ناسازگاری‌ها، تأیید اعتبار اطلاعات تضمین شود. سپس ویژگی‌ها بر اساس تعداد دسترسی و اندازه هر فایل کتابخانه‌ای، تعداد دسترسی‌ها و تغییر در هر کلید رجیستری و فراخوانی‌های API بر اساس تعداد انجام عمل مذکور از مجموعه کل اطلاعات انتخاب می‌شوند. درنهایت برخی ویژگی‌ها نیز بر اساس عملکرد، انتخاب می‌شوند (شکل ۵). با دسته‌بندی ویژگی‌های استخراج‌شده (سالم و مخرب)، رفتارهای مشترک بین برنامه‌های سالم و مخرب را حذف می‌کنیم. درواقع اگر تعداد تکرار ویژگی استخراج‌شده بین بدافزارها و سالم‌ها نزدیک بود، این ویژگی را بی‌تأثیر در نظر می‌گیریم. برای محاسبه نرخ مشاهده ویژگی‌ها از روابط (۱) و (۲) استفاده می‌کنیم.

$$M(p) = \frac{\text{تعداد ویژگی مشاهده‌شده در بدافزارها}}{\text{تعداد کل بدافزارها}} \quad (۱)$$

$$B(p) = \frac{\text{تعداد ویژگی مشاهده‌شده در نمونه‌های سالم}}{\text{تعداد کل سالم‌ها}} \quad (۲)$$

در جدول (۲) نمونه‌هایی از ویژگی‌های استخراج‌شده از نسل‌های بدافزار Trojan-Banker.Win۳۲، بر اساس دسترسی‌های انجام شده در نسل‌های مختلف و بسامد[1] آن‌ها در یک مجموعه از توالی‌ها را مشاهده می‌کنید. برای نمایش کلیدهای رجیستری از نمادها اختصاری استفاده کردیم.

ارزیابی آماری ما بر روی فراخوانی‌های API مربوط به اعمال تغییرات در کلیدهای رجیستری و فایل‌ها توسط بدافزارها در شکل (۶) نمایش داده‌شده است. با توجه به مشاهده این رفتارها در درصد بالایی از نمونه‌ها، نتیجه می‌گیریم که بررسی این دسته از رفتارها بسیار حائز اهمیت است. همان‌طورکه بیان شد، بعد از استخراج ویژگی از مجموعه بدافزارها و فایل‌های سالم، دو گونه ویژگی را برای دسته‌بندی انتخاب کردیم. نوع اول بر اساس بسامد این رفتار در مجموعه آزمایشی و نوع دوم، ویژگی‌هایی که به‌طور معنایی می‌توانند بیانگر رفتار مخرب باشد؛ مثلاً ایجاد کلید در شاخه Trusted Publisher\Certificates در رجیستری به‌عنوان یک ویژگی در نظر گرفته می‌شود. بدافزار گواهینامه[2] خود را به‌عنوان یک برنامه قابل‌اطمینان در این شاخه درج می‌کند. یا بدافزار با ایجاد تغییر در شاخه Firewall Policy\Standard Profile \Authorized Applications\List پیکربندی دیوار آتش[3] را تغییر می‌دهد. همچنین توابعی مانند Create Mutex ، Exit Process ازنظر عملکرد حائز اهمیت هستند و توسط تعداد قابل‌توجهی از بدافزارها استفاده شده‌اند.



شکل (۵). بخش‌های مختلف مدل پیشنهادی برای تحلیل تصویر حافظه و انتخاب ویژگی

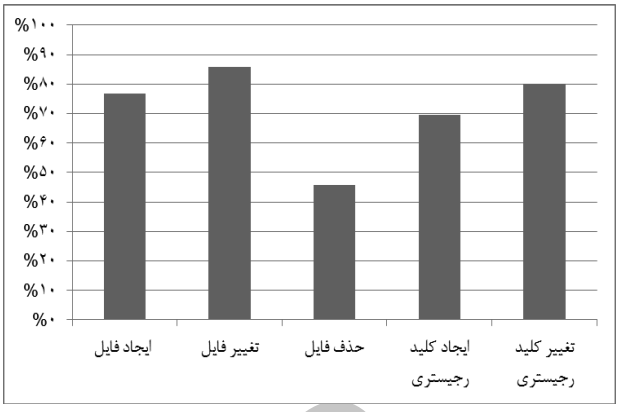
جدول (۲). خلاصه‌ای از ویژگی‌های استخراج‌شده از بدافزار Trojan-Banker.Win32

Family API Call	…GetProcAddress-VirtualAlloc-LoadLibraryA-GetModuleHandleA–CloseHandle-RegCreateKeyExA-CreateThread- Sleep- FreeLibrary- Create-FileA - CreateMutexA-…
Family Key Access	… \ CONTROL\SESSION MANAGER-\ SORTING\VERSIONS-\ PARAMETERS\PROTOCOL-\ USER\ S-1-5-21-83171654 \ Config\SECURITY\CMI-…
Family DLL Access	… User32. DLL-Ole32. DLL-Comctl32. DLL- Uxtheme. DLL-Olepro32. DLL-WS2_32.DLL- Kernel32. DLL- Ntdll. DLL- Advapi32. DLL- Oleaut32. DLL-Version. DLL-…

1- Frequency

2- Certificate

3- Firewall



شکل (۶). رفتارهای مشاهده‌شده در نمونه‌ها

۸- فرایند دسته‌بندی

پس از انتخاب مجموعه ویژگی‌ها، بردارهای ویژگی نهایی برای هر نمونه مقداردهی می‌شود. بر اساس اینکه کدام ویژگی انتخابی در هر نمونه ظاهرشده است به آن مقدار عددی مناسب تخصیص داده می‌شود. این مقدار بر اساس حضور یا عدم حضور کلید رجیستری (۰ یا ۱)، نسبت تعداد هر فراخوانی به تعداد کل فراخوانی‌های انتخاب‌شده و حاصل‌ضرب تعداد دسترسی به هر فایل کتابخانه‌ای در اندازه آن برحسب مگابایت برای هر نمونه محاسبه می‌شود. سپس با توجه به مقادیر محاسبه‌شده در کل نمونه‌ها یک پیش‌پردازش انجام‌شده و همه مقادیر به ۰ و ۱ تبدیل می‌شوند.

در این مرحله ما از کتابخانه Weka [۵۰] استفاده کردیم. این کتابخانه مجموعه‌ای از ابزارهای داده‌کاوی است که برای کلاس‌بندی و خوشه‌بندی استفاده می‌شوند. در تمام آزمایش‌ها پس از انتخاب الگوریتم کلاس‌بندی برای ایجاد، آموزش و ارزیابی دسته‌بند روش Cross validation را انتخاب کرده و تعداد Foldها را ۱۰ در نظر گرفتیم. در این روش ابتدا مجموعه داده به بخش‌های مساوی به تعداد Foldها تقسیم می‌شود، سپس به‌صورت متوالی یکی از بخش‌ها به‌عنوان مجموعه آزمون و سایر بخش‌ها به‌عنوان آموزش در نظر گرفته می‌شود، تا تمام بخش‌ها به‌عنوان آزمون استفاده شوند. به‌این‌ترتیب در هر بار تکرار بعد از آموزش، مدل با نمونه‌های جدید آزمون می‌شود. در انتها میانگین نتایج این اجراها به‌عنوان خروجی نهایی انتخاب می‌شود. ما ۴ دسته‌بند پایه از Weka شامل SMO، Random Forest، Decision Tree، Nave-Bays و Instance-Base را آزمایش کردیم. ورودی این الگوریتم‌های دسته‌بندی ۱۳۰ ویژگی استخراج‌شده از طریق پویش‌گر پیشنهادی می‌باشد.

۹- نتایج ارزیابی

روش مبتنی بر تحلیل تصویر حافظه در سال‌های اخیر مورد استفاده وسیع محققان قرارگرفته است. بررسی‌های ما نشان می‌دهد که این روش مؤثری برای تشخیص بدافزار در سیستم‌های مشکوک به آلودگی است. ارزیابی شامل اعتبارسنجی نتایج الگوریتم‌های دسته‌بندی مختلف است، که توسط پارامترهای محاسباتی همچون نرخ تشخیص و دقت انجام می‌شود. جهت ارزیابی خروجی و مقایسه طرح پیشنهادی با سایر مدل ها از پارامترهای جدول (۳) و روابط (۶-۳)، استفاده می‌کنیم.

جدول (۳). پارامترهای استفاده‌شده در ارزیابی

نرخ مثبت درست[1] (TPR)	نسبت برنامه‌های مخربی که به‌عنوان بدافزار شناسایی می‌شوند.
نرخ مثبت کاذب[2] (FPR)	نسبت برنامه‌های سالمی که به‌عنوان بدافزار شناسایی می‌شوند.
نرخ منفی صحیح[3] (TNR)	نسبت برنامه‌های سالمی که به‌عنوان بی‌خطر شناسایی می‌شوند.
نرخ منفی کاذب[4] (FNR)	نسبت برنامه‌های سالمی که بدافزار شناسایی می‌شوند.
مجموع TP+ FN	کل تعداد بدافزارها و مجموع FP+TN برابر کل نمونه‌های سالم می‌باشد.
ROC[5]	یک طرح گرافیکی که کارایی یک سیستم دسته‌بندی دودویی را نشان می‌دهد.
Recall	نسبت کل نمونه‌های درست دسته‌بندی‌شده به نسبت کل نمونه‌های مرتبط را بیان می‌کند.

$$\text{نرخ تشخیص} = \frac{TP}{TP + FN} \quad (۳)$$

$$\text{نرخ مثبت کاذب} = \frac{FP}{FP + TN} \quad (۴)$$

$$\text{دقت تشخیص} = \frac{TP + TN}{TP + TN + FP + FN} \quad (۵)$$

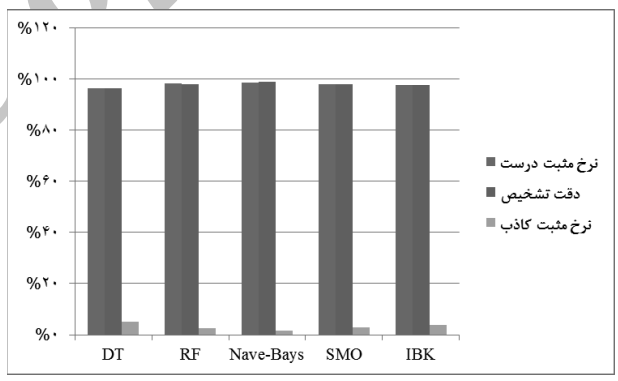
$$\text{معیار} - F = 2 * \frac{Recall * Precision}{Recall + Precision} \quad (۶)$$

1- True Positive Rate
2- False Positive Rate
3- True Negative Rate
4- False Negative Rate
5- Receiver Operating Characteristic

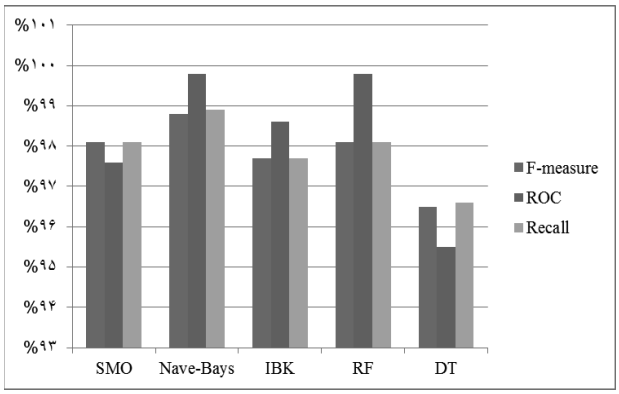
جدول (۴) نتایج حاصل از ۵ الگوریتم بررسی‌شده بر روی ۱۳۰ ویژگی انتخاب‌شده را نمایش می‌دهد (شکل ۷ الی ۱۰). در برخی آزمایش‌ها برای بهبود نتایج از الگوریتم کاهش بعد مبتنی بر انتخاب ویژگی نیز استفاده شده است. با توجه به تأکید روش پیشنهادی بر استخراج ویژگی با پرداختن به روش‌های دفاعی و مبهم‌سازی، نرخ مثبت صحیح برای تمام نسل‌های یک بدافزار برابر ۱۰۰٪ می‌باشد.

جدول (۴). نتایج دسته‌بندی ویژگی‌های استخراج‌شده با پویش‌گر پیشنهادی

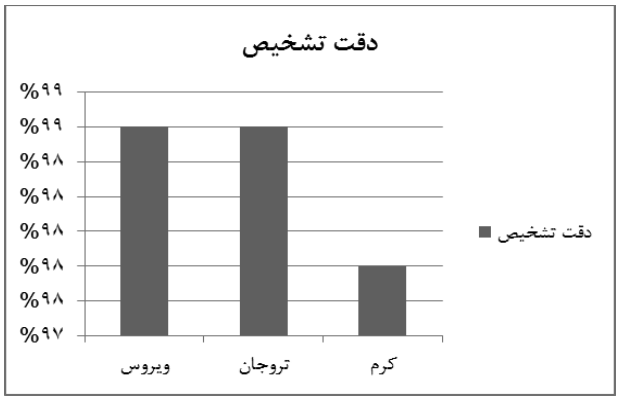
الگوریتم‌ها	نرخ مثبت درست	نرخ مثبت کاذب	دقت تشخیص	معیار F-	ROC Area
DT	۰/۹۶۶	۰/۰۵۱	۰/۹۶۶	۰/۹۶۵	۰/۹۵۵
RF	۰/۹۸۴	۰/۰۲۵	۰/۹۸۱	۰/۹۸۱	۰/۹۹۸
Nave-Bays	۰/۹۸۸	۰/۰۱۶	۰/۹۸۹	۰/۹۸۸	۰/۹۹۸
SMO	۰/۹۸۱	۰/۰۲۹	۰/۹۸۱	۰/۹۸۱	۰/۹۷۶
IBK	۰/۹۷۷	۰/۰۳۸	۰/۹۷۸	۰/۹۷۷	۰/۹۸۶



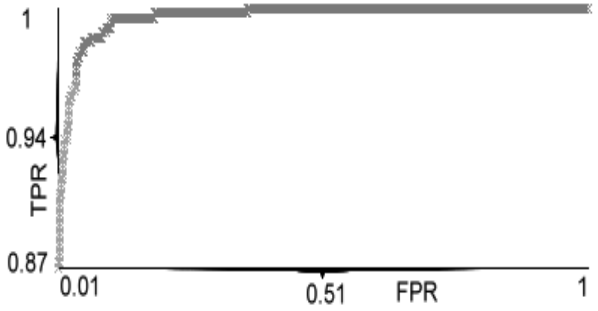
شکل (۷). نتایج الگوریتم‌های دسته‌بندی



شکل (۸). معیار ارزیابی‌شده در الگوریتم‌های دسته‌بندی



شکل (۹). دقت تشخیص در سه دسته اصلی



شکل (۱۰). نمودار ROC الگوریتم Nave-Bays در Weka

۱۰- نتیجه‌گیری

در این پژوهش، روشی برای استخراج، همبسته‌سازی و فیلترکردن اطلاعات از برخی ساختارهای حافظه با هدف تحلیل بدافزارها با قابلیت‌های دفاعی بررسی شد. این روش شامل یک پویش‌گر پیشنهادی برای استخراج فایل‌های کتابخانه‌ای بارگذاری شده، تغییرات اعمال‌شده در رجیستری و فراخوانی‌های توابع از ساختارهای مدیریت حافظه می‌باشد. برای اولین بار با ردگیری کد بدافزار در حافظه و سپس ردیابی فراخوانی‌های آن، به بررسی عملکرد اصلی بدافزار پرداختیم که روش عمومی‌تری نسبت به روش‌های قبلی می‌باشد. پس از استخراج ویژگی‌ها، بخش تحلیل‌گر ویژگی بر اساس بسامد هر ویژگی در مجموعه آزمایشی و تحلیل عملکرد آن، ویژگی‌ها را دسته‌بندی می‌کند. در بخش انتخاب ویژگی، خصیصه‌های نهایی برای الگوریتم دسته‌بندی انتخاب می‌شوند. سپس ویژگی‌ها با الگوریتم‌های کلاس‌بندی ارزیابی می‌شوند. بهترین نتایج شامل نرخ تشخیص ۹۸٪ و نرخ مثبت کاذب ۱۶٪ می‌باشند که نشان‌دهنده مؤثر بودن روش تحت مطالعه در شناسایی بدافزارها می‌باشد. در آینده ما درصدد بررسی ساختارهای دیگر حافظه برای استخراج پارامترهای مربوط به هر تابع و دنباله وابستگی فراخوانی‌های مرتبط هستیم.

۱۱- مراجع

[1] L. O. Murchu and E. Chien, "W32.Stuxnet dossier," Symantec Security Response, Tech. Rep., Oct. 2010.

[2] P. O'Kane, S. Sezer, and K. Mclaughlin, "Obfuscation: The Hidden Malware," in Security & Privacy, IEEE, Sept-Oct. 2011.

[3] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, "A survey on Automated Dynamic Malware-Analysis Techniques and Tools," ACM Computing Surveys (CSUR), February 2012.

[4] M. Sharif, A. Lanzi, J. Giffin, and W. Lee, "Automatic Reverse Engineering of Malware Emulators," in Security and Privacy, 2009 30th IEEE Symposium on, 17-20 May 2009.

[5] C. Ries, "Inside Windows Rootkits," in Vigilant Minds Inc., 4736, May 2006.

[6] J. Butler and P. Silberman, "Raide: Rootkit analysis identi-fication elimination," in Black Hat USA, vol. 47, 2006.

[7] A. Kristine, "Techniques and Tools for Recovering and Analyzing Data from Volatile Memory," 2009. [Online]. Available:http://www.sans.org/?utm_source=web&utm_medium=text-ad&utm_content=generic_rr_pdf_(c)_text1&utm_campaign= Reading_Room&ref=36914.

[8] S. Vomel and H. Lenz, "Visualizing Indicators of Rootkit Infections in Memory Forensics," In IT Security Incident Management and IT Forensics (IMF), 2013 Seventh International Conference on IEEE, pp. 122-139, March 2013.

[9] "Windows Rootkit Overview," Symantec Corporation, 2010.

[10] A. Aljaedi, D. Lindskog, P. Zavarsky, R. Ruhl, and F. Almari, "Comparative Analysis of Volatile Memory Forensics: Live Response vs. Memory Imaging," in Privacy, Security, Risk and Trust (passat), International Conference on and 2011 IEEE third, International Conference on Social Computing (socialcom), 9-11 Oct. 2011.

[11] "SQL Slammer Worm Propagation," 2003. [Online]. Available: http://xforce.iss.net/xforce/xfdb/11153.

[12] A. White, B. Schatz, and E. Foo, "Surveying the User Space Through User Allocations," in Digital Investigation 9, August 2012.

[13] M. E. Russinovich and D. A. Solomon, "Windows Internals," 4th ed., Redmond: Microsoft, 2005.

[14] B. Dolan-Gavitt, "The VAD Tree: A Process-eye View of Physical Memory," in Digital Investigation, September 2007.

[15] M. Ligh, S. Adair, B. Hartstein, and M. Richard, "Malware Analyst's Cookbook and DVD: Tools and Techniques for Fighting Malicious Code," Wiley, 2010.

[16] M. Ligh, S. Adair, B. Hartstein, and M. Richard, "Malware Analyst's Cookbook and DVD: Tools and Techniques for Fighting Malicious Code," Wiley, 2010.

[17] A. Schuster, "Searching for Processes and Threads in Microsoft Windows Memory Dumps," Digital Investigation 3, pp. 10-16, 2006.

[18] A. Tevanian and e. al, "A UNIX Interface for Shared Memory and Memory Mapped Files Under Mach," in USENIX Summer, 1987.

[19] M. Ligh, "Malfind Volatility Plugin," [Online]. Available: http://mnin.blogspot.com, 2009.

[20] T. C. Keong, "Dynamic Forking of Win32 EXE," [Online]. Available: http://www.security.org.sg/ code/loadexe.html, 2004.

[21] A. Walters and B. Dolan-Gavitt, "Volatility: an advanced memory forensics framework," 2007.

[22] "GMER - Rootkit Detector and Remover," [Online]. Available: http://www.gmer.net/, 2012.

[23] B. Cogswell and M. Russinovich, "Rootkit Revealer," [Online]. Available: www. sysinternals. com/ ntw2k/freeware/rootkitreveal. shtml , 2006.

[24] J. Pan, "Ice Sword," [Online]. Available: http:// www.xfocus.net /tools/200509 /1085.html, 2005.

[25] G. Palmer, "A Roadmap for Digital Forensic Research," First Digital Forensic Research Workshop (DFRWS), 2001.

[26] R. Harris, "Examining how to define and control the anti-forensics problem," Proceedings of the 6th Annual Digital Forensic Research Workshop (DFRWS '06), Digital Investigation 2006, 3(Suppl. 0), 2006.

[27] T. Haruyama and H. Suzuki, "One-byte Modifications for Breaking Memory Forensic Analysis," In Proceedings of Blackhat Europe, 2012.

[28] L. Milkovic, "Defeating Windows Memory Forensics," In Proceedings of the 29th Chaos Communications Conference, 2012.

[29] J. Stüttgen and C. M, "Anti-forensic Resilient Memory Acquisition," In The Proceedings of the Thirteenth Annual DFRWS Conference, August 2013.

[30] H. Inoue, F. Adelstein, and R. Joyce, "Visualization in Testing a Volatile Memory Forensic Tool," In Digital Investigation, 2011.

[31] D. Bilby, "Low down and Dirty: Anti-forensic Rootkits," In: Proceedings of Black Hat, Japan, 2006.

[32] S. Vömel and F. Freiling, "Correctness, atomicity, and integrity: defining criteria for forensically-sound memory acquisition," In Digital Investigation, November 2012.

[33] B. D. Carrier and J. Grand, "A hardware-based Memory Acquisition Procedure for Digital Investigations," in Digital Investigation, February 2004.

[34] A. Boileau, "Hit by a Bus: Physical Access Attacks with Firewire," In Ruxcon Computer Security Conference, 2006.

[35] J. Wang, F. Zhang, K. Sun, and A. Stavrou, "Firmware-assisted Memory Acquisition and Analysis Tools for Digital Forensics," Systematic Approaches to Digital Forensic Engineering (SADFE),IEEE Sixth International Workshop on. IEEE, 2011.

[36] C. Tilbury, August 2012. [Online]. Available: https://code.google.com/p/mft2csv/wiki/SetRegTime.

[37] J. Williams and A. Torres, 2014. [Online]. Available: http://code.google.com/p/attention-deficit-disorder/.

[38] L. Milković, 28 December Communication Congress in Hamburg 2012. [Online]. Available: http://code.google.com/p/dementia-forensics/downloads/detail?name=Defeating Windows memory forensics.pdf.

[39] T. Haruyama and H. Suzuki, 16 March 2012. [Online]. Available: https://media.blackhat.com/bh-eu-12/Haruyama/bh-eu-12-Haruyama-Memory_Forensic-Slides.pdf.

[40] D. Brendan, "Forensic Analysis of the Windows Registry in Memory," in Digital Investigation, September 2008.

[41] A. Wichmann and E. Gerhards-Padilla, "Using Infection Markers as a Vaccine Against Malware Attacks," In Green Computing and Communications (GreenCom), International Conference on, 20-23 Nov. 2012.

[42] [Online]. Available: http://home.mcafee.com/virusinfo/virusprofile.aspx?key=142626.

[43] R. B. Van Baar, W. Alink, and A. R. Van Ballegooij, "Forensic Memory Analysis: Files Mapped in Memory," In Digital Investigation, 2008.

[44] "Volatility Labs," Black Hat USA & DFRWS 2014, July 2014. [Online]. Available: http://volatility-labs.blogspot.ae/.

[45] S. Almarri and P. Sant, "Optimised Malware Detection in Digital Forensics," International Journal of Network Security & Its Applications 6.1, 2014.

[46] "ntoskrnl.exe," [Online]. Available: http://en.wikipedia.org/wiki/Ntoskrnl. [Accessed 2014].

[47] V. Zwanger and F. C. Freiling, "Kernel Mode API Spectroscopy for Incident Response and Digital Forensics," Proceedings of the 2nd ACM SIGPLAN Program Protection and Reverse Engineering Workshop. ACM, 2013.

[48] "Malware Research & Data Center," [Online]. Available: http://www.virussign.com/.

[49] "Computer Virus Collection," [Online]. Available: http://vxheaven.org/vl.php. [Accessed 2014].

[50] Melville, "WEKA Tutorial," [Online]. Available: http://www.cs.utexas.edu/users/ml/tutorials/Weka-tut/. [Accessed 2014].

A new Architecture for Impact Projection of Cyber-Attacks Based on High Level Information Fusion in Cyber Command and Control

M. Aghaei Kheirabady*, S. M. R. Farshchi, H. Shirazi

*Malek Ashtar University of Technology, Tehran, Iran

(Received: 29/07/2014, Accepted: 01/09/2015)

ABSTRACT

Detection methods based on analysis of memory contents have achieved great popularity in recent years. Researches in this area have great progress and powerful analysis frameworks has been innovated. Although these frameworks provide detailed examination of a memory snapshot, interpretation and correlation of these details to extract inconsistencies require a comprehensive knowledge of the internal structure of the operating system. In this paper, our proposed scanner focus on extracting information from the memory structure along with addressing the inconsistencies created by defense techniques used by malwares. In the proposed method, memory forensics is used, for the first time, to investigate the main functionality of malware by extracting function calls from the user space memory. In other words, in this method memory structures are described to extract the effective indicators related to registry changes, access to library files and operating system function calls. At last to evaluate the extracted features, Samples have been classified based on the selected feature. Best result include detection rate of 98% and false positive rate of 16%, which demonstrates the effectiveness of the memory contents.

Keywords: Malware Analysis, Memory Forensic, Digital Artifacts, Userspace Memory, Volatile Data, Feature Extraction

* Corresponding Author Email: masoume_aghaei@mut.ac.ir