مجله علمی پژوهشی «پدافند الکترونیکی و سایبری»
سال ششم، شماره ۴، زمستان ۱۳۹۷؛ ص ۴۴-۳۳

روشی نوین برای تشخیص تدریجی شرایط محیطی و منابع لازم برای بدافزارهای هوشمند

سعید پارسا[1*]، هادی خوش‌روی[2]

۱- دانشیار، دانشکده مهندسی کامپیوتر، دانشگاه علم صنعت ایران

۲- کارشناسی ارشد، گروه کامپیوتر، دانشکده فنی مهندسی، دانشگاه آزاد اسلامی واحد شبستر

(دریافت: ۱۳۹۶/۰۷/۲۶، پذیرش: ۱۳۹۷/۰۳/۰۶)

چکیده

بدافزارهای هوشمند دو رفتار دفاع از خود و بدخواهانه دارند. این دو نوع رفتار تحت شرایط محیطی ظاهر می‌شوند. هدف از این مقاله ارائه راهکاری جهت تشخیص شرایط محیطی برای نمایش رفتار بدخواهانه بدافزارهای هوشمند است. می‌توان با توجه به عملکرد توابع سیستمی که در لیست فهرست جدول IAT یک بدافزار موجود است و در بین این توابع آن‌هایی که در عمل در زمان اجرا فراخوانی نشده‌اند، به بدافزار مشکوک شد. با تحلیل عملکرد هر تابعی که وجود منبعی در محیط را بررسی می‌کند و با فراهم کردن منبع مورد درخواست می‌توان به‌مرور منابع و شرایط لازم برای اجرای رفتار بدخواهانه را مشخص کرد. درواقع با توجه به اینکه در یک اجرا، تابع سیستمی مورد فراخوانی وجود چه منبع و شرایط محیطی را بررسی می‌کند و با ایجاد آن منابع و شرایط می‌توان در طی اجراهای متوالی هر چه بیشتر و به‌مرور شرایط محیطی و منابع لازم برای برقراری این شرایط را مشخص نمود تا اینکه نهایتاً بعد از چند اجرا این شرایط و منابع مربوطه مشخص شوند. ارزیابی‌های انجام‌شده در یک محیط جعبه شن، کارایی روش پیشنهادی را مشخص کرده است.

کلیدواژه‌ها: جعبه شن، بدافزار، بدافزارهای هوشمند، تحلیل بدافزار، شرایط محیطی

۱- مقدمه

هدف در این مقاله ارائه روشی جهت شناسایی شرایط محیطی و منابع لازم برای فعال شدن بدافزارهای هوشمند و آگاه از محیط می‌باشد. استاکس نت ازجمله بدافزارهای هوشمندی است که صرفاً با تشخیص plc های زیمنس متصل به کامپیوتر و یا موجود در شبکه، بخش بدخواه کد خود را به اجرا درمی‌آورد. به این واسطه مسیر اجرایی حاوی کد بدخواه در بدافزارهای هوشمند را اصطلاحاً مسیرهای پنهان می‌نامند [۴-۱]. نکته اصلی در شناسایی این بدافزارها، تشخیص مسیرهای اجرایی پنهان است.

امروزه بدافزارهای هوشمند[1] به علت اینکه ضد بدافزارها هنوز قادر به ایجاد کامل شرایط محیطی برای اجرای مسیرهای پنهان بدافزار نیستند، به‌سادگی قابل‌تشخیص نمی‌باشند [۲ و ۵]. لذا، این دسته از بدافزارها در عمل عامل خطرناک‌ترین گونه از حمله‌های سایبری بوده‌اند[۷-۵].

بدافزارهای با خصوصیت هوشمند که رفتار مبهمی از خود نشان می‌دهند، معمولاً زمانی کدهای بدخواه خود را اجرا می‌نمایند که شرایط محیط را امن تشخیص دهند. برای همین منظور، منتظر اتفاق خاص مانند تشخیص وجود آسیب‌پذیری، وجود اینترنت، اتصال یک دستگاه خاص به رایانه، نصب بودن یک نرم‌افزار خاص در سیستم‌عامل، اجراشدن یک پردازه خاص و وجود یک سرویس فعال در سیستم، می‌مانند. لذا، تحلیل چنین بدافزارهایی در محیط‌های جعبه شن دشوار می‌باشد. چنین بدافزارهایی اغلب محیط جعبه شن را تشخیص داده، هنگام تحلیل در محیط جعبه شن فقط یک مسیر اجرایی از خود نشان می‌دهند، چراکه منتظر شرایط امن برای اجرا شدن کدهای مخرب خود می‌باشند. به این دلیل، تحلیل مسیرهای اجرایی مخرب، معمولاً پنهان و دور از دسترس تحلیل‌های جعبه‌های شنی قرار می‌گیرد [۶-۳]. لذا، جهت شناسایی این دسته از بدافزارها لازم است که قبل از هر کاری راه‌های شناسایی جعبه‌های شن و ماشین‌های مجازی و یا هرگونه محیط تحلیل و مهندسی معکوس بدافزار را شناسایی و مسدود نمود [۸-۷].

نکته اصلی در شناسایی بدافزارهای هوشمند تعیین شرایط محیطی لازم جهت فعال شدن رفتار بدخواهانه آن‌ها است. در این راستا با اجرای نمادین بدافزارها سعی شده مسیرهای اجرایی پنهان شناسایی شود [۴-۳]. مشکل اصلی اجرای نمادین برای کدهای بزرگ انفجار حالات[2] است. ازآنجایی‌که بدافزارها اغلب نسبتاً کوچک می‌باشند این مشکل در مورد آن‌ها مصداق ندارد.

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

1- Smart malware

2- Patch explosion

البته مبهم‌سازی کد و جاسازی بدافزار در نرم‌افزارهای نسبتاً بزرگ، دو عامل مقابله با اجرای نمادین می‌تواند باشد[۴]. بررسی بسته‌های دریافتی از شبکه جهت شناسایی فرامین برای اجرای بدافزارها راه‌کار دیگری بوده است[۹ و ۴-۳]. فرامین از مراکز فرمان و کنترل بخصوص برای بات‌ها در شبکه‌های بات نت و جاسوس‌افزارهای قابل فرماندهی از راه دور، حائز اهمیت است. مشکل، درک مفهوم این فرامین است که اغلب به‌صورت رمز شده می‌باشند.

با پایش و نظارت بر فراخوانی‌های سیستمی، می‌توان تشخیص داد که کد اجرایی مبادرت به فراخوانی‌های سیستمی جهت گرفتن اطلاعات از محیط نموده است[۴]. با بررسی این اطلاعات می‌توان تشخیص داد که کد اجرایی نیاز به چه شرایط محیطی و ورودی‌هایی از محیط اجرایی خود دارد تا مسیر پنهان خود را به اجرا درآورد. برای نمونه یک بدافزار ممکن است با اجرای چند فراخوانی سیستمی مبادرت به جستجو و باز کردن فایل خاصی بنماید. البته نیاز به فایل خاص نمی‌تواند دلیلی برای بدخواه بودن کد باشد. می‌بایست فراخوان‌های سیستمی دیگر را نیز بررسی نمود. درهرحال در این حالت وجود فایل موردنظر، شرط لازم برای تشخیص رفتار بدخواه است. تشخیص رفتار بدخواهانه مقوله‌ای جداگانه است که در اینجا مطرح نمی‌باشد.

البته بدافزار ممکن است وجود ابزار نظارت و پایش را تشخیص دهد. برای رفع این مشکل در این مقاله از مقایسه توابع سیستمی مورد دسترسی در زمان اجرا و لیست توابع مظنون به خطر سیستمی موجود در فایل کد اجرایی، مشخص می‌شود که کد اجرایی رفتار احتمالاً بدخواه خود را پنهان کرده است. منظور از توابع سیستمی مظنون به خطر، آن دسته از توابع سیستمی است که در کدهای بدخواه اغلب برای تخریب و عملیات بدخواهانه مورد استفاده قرار می‌گیرند[۱۱-۱۰]. در کد اجرایی برنامه‌های تحت ویندوز، جدول آدرس‌دهی توابع سیستمی به نام IAT[1] حاوی لیستی از کلیه DLL ها و توابع واسط سیستمی[2] است. با اجرای حمله‌های تزریق DLL و یا تزریق کد [۱۴-۱۱]. می‌توان مشخص کرد که در زمان اجرا کدام توابع سیستمی مورد دسترسی قرارگرفته‌اند. با بررسی توابع سیستمی استفاده نشده می‌توان مشخص کرد که این توابع به دنبال کدام شرایط و ورودی‌ها از محیط اجرایی هستند. با ایجاد این شرایط به‌صورت مجازی و اجرای مجدد نرم‌افزار مورد تحلیل، بعد از ایجاد شرایط می‌توان مشخص نمود که آیا برقراری شرایط در اجرای نرم‌افزار تأثیرگذار بوده است.

پس از شناسایی و ایجاد مجازی و بعضاً واقعی شرایط محیطی برای ۱۶۰۰ بدافزار متعلق به ۱۶ خانواده مختلف، مبادرت به اصلاح جعبه شنی خود به نام جعبه شن پارسا نموده‌ایم. جعبه شنی پارسا برای تحلیل دقیق بدافزارهای ۳۲ و ۶۴ بیتی توسعه داده شده است. با افزودن نرم‌افزار مولد شناسایی شرایط محیطی توانستیم بدافزارهای هوشمندی که عملاً توسط جعبه شن‌های شناخته‌شده مثل کوکوسندباکس [۱۵] و ژئوسندباکس [۱۶] قابل بررسی نبودند را شناسایی کنیم. معمولاً محصولات ضد بدافزاری بعد از مظنون شدن به یک بدافزار جهت تحلیل بیشتر، آن‌ها را برای جعبه‌شن محلی و یا مرکزی خود ارسال می‌کنند. حاصل تحلیل توسط ضد بدافزارها بررسی می‌شود. ازآنجایی‌که جعبه‌های شنی مسلح به ابزار شناسایی و ایجاد شرایط محیطی خاص بدافزار نیستند، محصولات ضد بدافزاری قادر به شناسایی آن‌ها نبوده به‌عنوان بدترین نوع بدافزار شناسایی شده‌اند [۶-۳ و ۱۸-۱۷]

در ادامه این مقاله، در بخش دوم کارهای مرتبط موردبررسی و کنکاش قرارگرفته و نهایتاً چالش‌ها و مشکلات روش‌های کنونی مشخص‌شده است. در بخش سوم روش پیشنهادی ما برای شناسایی شرایط محیطی مطرح‌شده است. در بخش چهارم روش پیشنهادی در عمل مورد ارزیابی قرارگرفته است. نهایتاً نتیجه‌گیری و پیشنهاد کارهای آتی در بخش پنجم مطرح‌شده است.

۲- کارهای مرتبط

علیرغم تأکید و تحقیقات زیاد در مورد شناسایی بدافزارهای هوشمند و آگاه از محیط، هنوز محیط جعبه شن مناسب جهت ایجاد شرایط محیطی لازم برای شناسایی این دسته از بدافزارها ارائه نشده است. تشخیص رفتار پنهان بدافزار و کشف تونل‌های مخفی اجرایی بدافزار همواره یکی از چالش‌های متخصصین امنیت بوده است. در این راستا کارهای تحقیقاتی متعددی ارائه شده است [۲۰-۱۹].

اغلب تحقیقات کنونی متمرکز بر شناسایی مجموعه محدود و مشخصی از شرایط محیطی و منابع لازم برای شناسایی رفتار بدافزارهای هوشمند بوده‌اند. برای نمونه MineSweeper [۱۹] ابزاری برای تشخیص نیازها و شرایط لازم برای درگاه‌های شبکه، صفحه‌کلید و شرایط زمانی جهت نمایان شدن رفتار بدخواه بدافزار است. یا روشی جهت تشخیص فایل‌های موردنیاز، ورودی‌ها از درگاه‌های شبکه و بالاخره شرایط زمانی لازم برای فعال شدن بدافزار آگاه از محیط ارائه شده است[۲۱]. ابزار PyTrigger [۲۲] تأکید بر تشخیص فرامین و ورودی‌های کاربر از طریق صفحه‌کلید دارد که موجب فعال شدن بدافزار هوشمند

1- Import Address Table
2- API

می‌گردد. اصولاً، محققین در زمینه بدافزار [۲۲ و ۲۳-۲۱] به این نتیجه رسیده‌اند که در بسیاری از موارد مشاهده می‌شود که بدافزار منتظر فعالیت خاصی توسط کاربر از طریق صفحه‌کلید یا ماس است تا رفتار بدخواهانه خود را ظاهر نماید. در این راستا ابزار PyTrigger [۲۲] با ایجاد مجازی شرایط محیطی مبتنی بر فعالیت‌های کاربر سعی به شناسایی بدافزارهای هوشمند نموده است. این مجموعه فعالیت‌ها در ارتباط با فعالیت‌های کاربر از طریق صفحه‌کلید و ماس هستند. ازجمله این فعالیت‌ها استفاده از مرورگر، ارسال ایمیل، دسترسی به حساب‌های بانکی، عنوان صفحات بازشده توسط کاربر و آدرس‌های مورد دسترسی را می‌توان ذکر نمود.

مطالعات بر روی رفتار بدافزارها در سیستم‌های موبایل نشان داده که موقعیت مکانی، پروفایل کاربر و وجود برنامه‌های کاربردی خاص بر روی سیستم‌های موبایل ازجمله مواردی است که موجب فعال شدن بدافزارهای هوشمند در این‌گونه سیستم‌ها بوده است [۲۰].

تعیین شرایط محیطی لازم برای اجرای بدافزارهای هوشمند مسلماً تأثیرگذار بر تعیین مدل رفتاری این نوع بدافزارها می‌باشد. در این راستا استفاده از ابزارهای نظارتی یا در اصطلاح مانیتورنگ بخصوص جهت شناسایی فراخوانی‌های سیستمی مطرح می‌باشد. یک روش پیاده‌سازی ابزار مانیتورینگ قلاب‌اندازی به جدول‌های SSDT[1] و Shadow SSDT است [۱۲ و ۱۴-۱۳] که در سیستم‌های عامل بعد از ویندوز XP عملاً به‌واسطه KPP[2] امکان‌پذیر نمی‌باشد. لذا، استفاده از جعبه‌های شنی مبتنی بر فیلتر درایورها برای نظارت بر فراخوانی‌های سیستمی مطرح شده است. مشکل عمده در این راستا شناسایی محیط جعبه شن توسط بدافزارها می‌باشد. علاوه بر این، جعبه‌های شنی مبتنی بر فیلتر درایور امکان تشخیص فراخوانی‌های سیستمی در سطح کاربر را ندارند. می‌بایست خاطرنشان نمود که هر فراخوانی در سطح کاربر ممکن است به چندین فراخوانی درسطح سیستم نیاز داشته باشد. لذا، درک اینکه درمجموع فراخوانی‌های سیستمی چه فراخوانی در سطح کاربر را تشکیل می‌دهند، مشکل می‌باشد. همین مشکل تحلیل رفتار را پیچیده می‌نماید. در ادامه در بخش (۲-۱) روش اکتشاف مسیرهای چندگانه برای بررسی مسیرهای اجرایی جهت شناسایی فراخوانی‌های مظنون سیستمی ارائه شده است. در این فرامین ورودی برای اجرای بات‌ها در شبکه‌های بات‌نت و جاسوس افزارهای کنترل راه دور، به‌عنوان شرایط محیطی تشخیص داده می‌شود.

1- System Service Descriptor Table
2- Kernel patch protection

اجرای نمادین[۴-۳]، اجرای کانکالیک [۲۵-۲۴] و تفسیر انتزاعی سه روش برای تحلیل پویای بدافزارها بوده‌اند. اجرای نمادین جهت شناسایی شرایط محیطی موضوع بخش ۲-۲ است.

۲-۱- روش‌های کشف پنهان مسیرهای اجرایی بدافزار

کشف رفتارهای پنهان بدافزار یکی از چالش‌های متخصصین امنیت می‌باشد. مزایا و مشکلات این روش‌ها در ادامه توضیح داده شده است.

۲-۱-۱- اکتشاف مسیرهای چندگانه

ایده اصلی اکتشاف مسیرهای اجرایی چندگانه [۲]، مبتنی بر نحوه استفاده کد از مقادیر ورودی خاص است. به عبارت دیگر، مقادیر ورودی‌های مشخص خوانده‌شده توسط برنامه، مثل زمان جاری سیستم، بررسی فایل، مقادیر خوانده‌شده از شبکه و یا نتیجه بررسی اتصال اینترنت، به‌صورت پویا ردگیری‌شده تا نقاطی که از این مقادیر برای تصمیم‌گیری جریان کنترل برنامه استفاده می‌کنند، شناسایی شوند. وقتی یک چنین نقاطی شناسایی شدند، ابتدا یک تصویر لحظه‌ای از وضعیت فعلی اجرای برنامه گرفته‌شده، سپس برنامه، وابسته به مقدارهای واقعی خوانده‌شده یکی از مسیرهای اجرایی را ادامه می‌دهد. بعد از اتمام این مسیر، به تصویر لحظه‌ای گرفته‌شده بازگشته و مقادیر ورودی به‌گونه‌ای بازنویسی می‌شوند تا مسیر دیگری طی شود. در این صورت می‌توان مسیرهای مختلف برنامه را با توجه به ورودی‌ها بررسی نمود [۲].

برای مثال نرم‌افزاری را در نظر بگیرید که وجود اتصال به شبکه محلی را بررسی می‌کند. با رهگیری فراخوانی‌های سیستمی زمان اجرای نرم‌افزار، هنگامی‌که نتیجه فراخوانی سیستمی مربوط به وجود اتصال به شبکه محلی در یک پرش شرطی مورداستفاده قرار می‌گیرد، یک تصویر لحظه‌ای از وضعیت فعلی اجرای نرم‌افزار گرفته می‌شود. با فرض اینکه نتیجه حاصل، عدم وجود اتصال به شبکه محلی باشد، اجرای نرم‌افزار خاتمه می‌یابد. برای اکتشاف مسیر اجرایی با شرط وجود شبکه محلی، با استفاده از تصویر لحظه‌ای گرفته‌شده، برنامه را به وضعیت قبلی برگردانده و نتیجه به‌گونه‌ای بازنویسی می‌شود که این بار وجود اتصال به شبکه محلی، حاصل شود. درنتیجه می‌توان عملکرد نرم‌افزار را در صورت وجود اتصال به شبکه محلی هم کشف نمود.

در این روش برای تغییر مسیرهای اجرایی می‌بایست ورودی‌ها که درواقع فرامین مرکز کنترل به بدافزار هستند، تغییر داده شوند. تغییر تصادفی ورودی‌ها برای تعیین مسیرهای اجرایی کاری بسیار زمان‌گیر است. البته می‌توان در صورت مبهم نبودن کد، در طی یک فرایند تکاملی داده‌های ورودی برای تشخیص مسیرهای مختلف اجرایی در یک برنامه را، ایجاد نمود. به هرحال

تغییر تصادفی ورودی‌ها برای تعیین مسیرهای اجرایی کاری بسیار زمان‌گیر می باشد.

۲-۱-۲- اجرای نمادین[1]

اجرای نمادین، با جایگزینی رابطه محاسبه کننده مقدار متغیرها با خود متغیرها روابط بزرگی صرفاً بر اساس ورودی‌های برنامه می‌سازد. برای هر مسیر اکتشافی، یک فرمول ایجاد می‌شود. شرایط اجرای هر مسیر نیز بر اساس ورودی‌ها مشخص می‌شود. با استفاده از ابزاری به نام solver معادلات مربوط به شرایط اجرایی مسیرها حل می‌شوند تا مقدار ورودی‌ها برای اجرای آن مسیرها مشخص گردد [۲].

برای نمونه روشی مبتنی بر اجرای نمادین جهت تشخیص زمان مقرر، ورودی‌ها از شبکه و فایل‌های موردنیاز برای اجرای بدافزارهای هوشمند ارائه شده است [۲۱]. در پژوهشی دیگر جهت ارتقاع روش مبتنی بر اجرای نمادین سعی شده بعد از تشخیص شرایط محیطی، شرایط موردنیاز جهت تحلیل رفتار بدافزار ایجاد شود [۱۹]. جهت مقابله یا تحلیل نمادین از روش‌های مبهم‌سازی کد استفاده می‌شود. در مرجع جهت تقویت روش‌های اجرای نمادین برای مقابله با مشکلات ناشی از مبهم سازی کد، روشی کارا ارائه شده است [۲۵].

مشکل عمده روش‌های اجرای نمادین مشکل شناخته‌شده انفجار حالت است [۴-۳] برای رفع مشکل انفجار حالت از روش دیگری تحت عنوان اجرای کانکالیک استفاده می‌شود. درهرحال همگی این روش‌ها ایستا و برای مقابله با آن‌ها نویسنده‌های بدافزار از راه‌کارهای مبهم‌سازی کد اجرایی استفاده می‌کنند.

۳- فرآیند طرح پیشنهادی

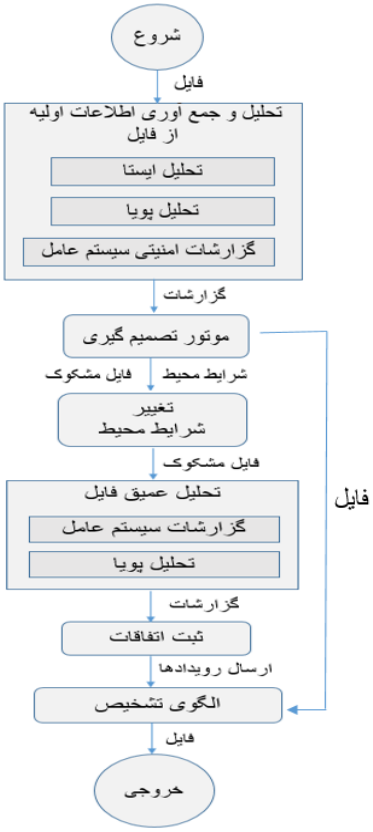
فرآیند طرح پیشنهادی در شکل (۱) نشان داده شده است. به‌طورکلی، این فرآیند شامل سه مؤلفه اصلی جمع‌آوری اطلاعات اولیه، موتور تصمیم‌گیری و تغییر شرایط محیط می‌باشد.

مؤلفه تحلیل و جمع‌آوری اطلاعات اولیه، با استفاده از جعبه شنی ۶۴ بیتی پارسا، فایل موردنظر را دریافت و موردبررسی و تحلیل قرار می‌دهد. تحلیل به دو صورت ایستا بر اساس ساختار و امضای فایل به‌صورت و پویا بر اساس سابقه یا گزارش‌های اجرایی فایل داده شده انجام می‌شود. علاوه بر این، این مؤلفه با استفاده از سابقه اجرایی تحت عنوان os log در سیستم‌عامل ویندوز اطلاعات امنیتی مثل ارسال بسته‌ها در سطح شبکه و هرگونه دسترسی و تغییرات در سیستم فایل را دریافت می‌کند. توضیح بیشتر در بخش ۳-۱، ارائه شده است.

مؤلفه دوم موتور تصمیم‌گیری است که بر اساس گزارش‌های گردآوری‌شده توسط مؤلفه اول تصمیم، به بررسی وجود مسیر پنهان در برنامه می‌پردازد. درواقع بدافزار هوشمند که تحت شرایط خاص رفتار مخرب خود را نشان می‌دهد، حاوی اصطلاحاً مسیرهای پنهانی است که در شرایط عادی اجرا نمی‌شوند.

در مؤلفه سوم شرایط محیطی که موتور تصمیم‌گیری مشخص کرده برای اجرای بدافزار ایجاد می‌شود و بدافزار تحت این شرایط در داخل محیط جعبه شنی اجرا می‌شود.

مؤلفه چهارم تحت شرایط ایجادشده به تحلیل فایل اجرایی می‌پردازد. در این مؤلفه واحد ثبت اتفاقات با مقایسه سوابق اجرایی قبل از اعمال شرایط محیطی و بعدازآن، مسیرهای پنهان را مشخص می‌کند. در ادامه این چهار مؤلفه توضیح داده شده‌اند.



شکل (۱): فرآیند روش پیشنهادی

۳-۱- تحلیل و جمع‌آوری اطلاعات اولیه

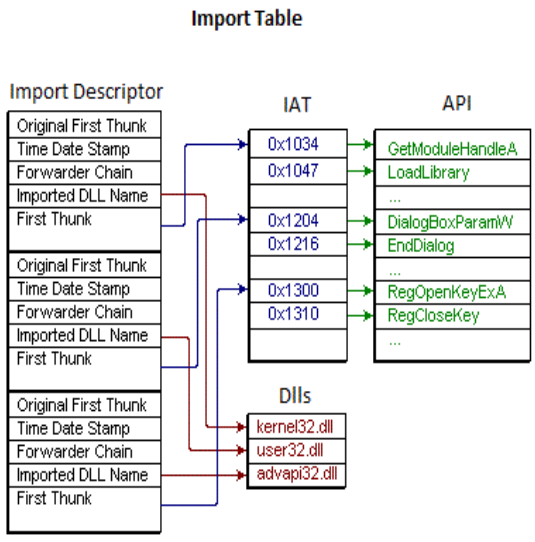
مؤلفه تحلیل و جمع‌آوری اطلاعات اولیه، شامل سه بخش تحلیل ایستا، تحلیل پویا و گزارش‌های سیستم‌عامل است. در این مرحله علاوه بر اینکه فایل مظنون تحلیل می‌شود، اطلاعات اولیه از فایل جهت تحلیل در مراحل بعد نیز گردآوری می‌شود.

اطلاعات اولیه از فایل مظنون، جهت تشخیص بدافزارهای هوشمند توسط مؤلفه موتور تصمیم‌گیری، گردآوری می‌شود. این اطلاعات اولیه شامل، لیست توابع API، و مسیرهای پیمایش شده در طول اجرا است.

1- concrete and symbolic execution

جدول آدرس‌دهی موجود در ساختار فایل‌های اجرایی در سیستم‌عامل ویندوز تحت عنوان IAT شناخته‌شده است. یکی از راه‌های شناخته‌شده برای محصولات ضد بدافزاری، استفاده از جدول IAT است. جدول IAT درواقع آدرس توابع واسط سیستمی را نگهداری می‌نماید. با روش قلاب اندازی[1] به این جدول، تحلیل‌گر می‌تواند لیست توابع واسط سیستمی یا در اصطلاح APIهای فایل اجرایی را به‌دست آورد. قلاب‌اندازی روشی برای تحلیل رفتار کد بداندیش است. این راه‌کار توسط ابزارهای تحلیل از قبیل جعبه شنی و ضد بدافزارها برای نظارت بر عملکرد و تعیین رفتار کد بداندیش استفاده می‌شود. روش مناسب برای به‌دست آوردن لیست توابع سیستمی، استفاده از سابقه اجرایی بدافزار می‌باشد چراکه، امکان دارد بدافزار بعد از اجرا برخی توابع واسط سیستمی دیگر را نیز به جدول آدرس‌دهی خود اضافه نماید [۲۶].

نفوذ در فضای آدرس پردازه برای اهداف مختلفی مثل اشکال‌زدایی، نظارت و تحلیل رفتار نرم‌افزارها و یا با نیت بدخواهانه مثل نوشتن ابزارهای جاسوسی، باشد. ضد بدافزارها با استفاده از روش‌های قلاب اندازی مثل IAT Hooking، Inline Hooking و با تزریق کتابخانه به فضای اجرایی نرم‌افزارها شروع به تحلیل رفتار برنامه‌ها می‌نمایند. شکل (۲)، نمونه‌ای از مراحل استخراج لیست توابع واسط سیستمی متعلق به سیستم‌عامل ویندوز را نشان می‌دهد. در واقع تحلیل‌گر، با روش قلاب اندازی، ره‌گیری و نظارت بر عملکرد فایل اجرایی را انجام می‌دهد.



شکل (۲): استخراج جدول آدرس‌دهی و لیست توابع سیستمی

۳-۱-۱- تحلیل ایستا

بدافزارها اغلب به‌منظور شناخته نشدن فایل‌های سیستمی مورد استفاده‌شان، در زمان اجرا مبادرت به بار کردن کتابخانه‌های موردنظر خود می‌نمایند. برای این منظور به‌طورمعمول از تابعی مثل loadlibrary استفاده می‌شود. لذا، علاوه بر تحلیل ایستا نیاز به تحلیل پویا جهت شناسایی این کتابخانه‌ها است. با تحلیل ایستا می‌توان لیست توابع سیستمی مورداستفاده را از جدول IAT استخراج کرد. این جدول در داخل فایل اجرایی که در اصطلاح به آن PE گفته می‌شود، نگهداری می‌گردد و همان‌گونه در شکل یک در بالا مشخص‌شده، حاوی اسامی کتابخانه‌ها و توابع واسط سیستمی مورد دسترسی است. با مقایسه این لیست با لیست کتابخانه‌ها و توابع مورد دسترسی در زمان اجرا می‌توان متوجه شد که کدام توابع و کتابخانه در زمان اجرا مورد دسترسی قرارگرفته است. این تفاوت عاملی برای شک به برنامه مربوطه به‌عنوان یک بدافزار است.

البته ممکن است بخش مخرب یک بدافزار به‌صورت کد پوسته یا در اصطلاح shell-code در داخل یک تصویر، logo، word و یا هر نوع فایل غیر اجرایی دیگر ذخیره شده باشد. برای همین عمل بررسی در داخل جعبه شن نبایستی صرفاً به کد اجرایی محدود گردد. می‌بایست هر فایل جدیدی موردبررسی قرار گیرد تا هرگونه کدپوسته ممکن در داخل فایل شناسایی گردد. این‌گونه کد معمولاً در مکان قابل پیش‌بینی در داخل فایل‌ها ذخیره‌سازی می‌شود. برای نمونه انتهای فایل عکس و بخش text در داخل PE، معمولاً کد پوسته ذخیره‌سازی می‌شود.

تزریق کد پوسته مخرب در سایر فایل‌ها یکی از راه‌کارهای بدافزار نویسان برای گریز از تشخیص بدافزار در هنگام اجرای آن می‌باشد. همان‌گونه که در بالا توضیح داده شد توابع سیستمی مورداستفاده کد اجرای، نیز در مرحله تحلیل ایستا استخراج می‌شوند. دلیل استخراج توابع واسط سیستمی یا در اصطلاح API های مورداستفاده در یک فایل اجرایی، تشخیص وجود مسیرهای اجرایی پنهان می‌باشد. که در مؤلفه موتور تصمیم‌گیری در شکل (۱) تشخیص داده خواهد شد.

۳-۱-۲- تحلیل پویا

در مرحله تحلیل پویا با اجرای کد ارائه‌شده در محیط جعبه شنی، سابقه اجرایی ثبت و موردبررسی و تحلیل قرار می‌گیرد. سابقه موردبررسی درواقع دنباله فراخوانی‌های سیستمی است، که در ضمن اجرا مشاهده شده است. علاوه بر نگهداری لیست فراخوانی‌های مشکوک سیستمی، جدول IAT در زمان اجرا از داخل ساختار EPROCESS برای فرایند اجرایی بررسی می‌شود.

1- Hooking

با استفاده از جدول IAT می‌توان لیست توابع سیستمی که مورد دسترسی یک برنامه است را مشخص نمود. برای این منظور از قطعه کد ذیل می‌توان استفاده نمود:

```
   BYTE* lpImportNameBuffer = new
BYTE[BUFFER_SIZE];
   BOOL bSuccess = ReadProcessMemory
   (hProcess,  (LPCVOID)((unsigned long
   long)lpImageBaseAddress + (unsigned
long long)pImageThunk->u1.AddressOfData),
   lpImportNameBuffer,
   BUFFER_SIZE,       0);
```

در قطعه کد فوق تابع سیستمی ReadProcessMemory لیست کتابخانه‌ها و توابع سیستمی هر کتابخانه جدول IAT را در lpImportNameBuffe ذخیره می‌نماید.

باید توجه شود که جدول IAT در زمان اجرا می‌تواند ساختار متفاوتی از آنچه در داخل فایل PE به‌عنوان جدول IAT مشخص شده داشته باشد. در این مرحله درواقع با توجه به جدول IAT کتابخانه‌ها و توابع مورد دسترسی در عمل مشخص می‌شوند. نکته دیگر این است که کد اجرایی بدافزارها اغلب به‌صورت بسته‌بندی شده یا در اصطلاح pack شده و یا بعضاً رمز می‌باشد. .در هنگام اجرا، کد بسته‌بندی شده بازگشایی و یا در اصطلاح unpack می‌شود. در داخل کد بسته‌بندی و یا رمز شده نمی‌توان جدول IAT را به‌درستی مشخص نمود. لذا، لازم است که در زمان اجرا هم این عمل انجام شود.

به این ترتیب ابهام در کد بدافزار به‌واسطه تغییر شکل جدول IAT مشخص می‌شود. علت استخراج تمامی توابع سیستمی مورداستفاده بدافزار، پی بردن به وجود مسیرهای اجرایی پنهان می‌باشد که در مؤلفه موتور تصمیم‌گیری تشخیص داده می‌شود. درصورتی‌که بدافزار توابع سیستمی خاصی را در جدول آدرس‌دهی توابع خود قرار داده ولی زمان اجرا استفاده نکند، موتور تصمیم‌گیری پی به وجود مسیرهای اجرایی پنهان اکتشاف نشده خواهد برد.

۳-۱-۳- گزارش‌های امنیتی سیستم‌عامل

گزارش‌های امنیتی که خود سیستم‌عامل ثبت می‌کند، کار تشخیص بدافزار را برای سیستم تحلیل ما راحت‌تر خواهد کرد. بررسی گزارش‌ها امنیتی سیستم‌عامل بعد از اجرای فایل مورد تحلیل انجام می‌شود. بررسی گزارش‌های امنیتی که توسط برنامه‌ای مثل Event viewer توسط سیستم‌عامل ویندوز تهیه می‌شود، می‌تواند اطلاعات نسبتاً عمیق از عملکرد بدافزار در اختیار قرار دهد. به‌این‌ترتیب فایل‌های مشکوک تشخیص داده شده و جهت بررسی بیشتر در محیط آزمایشگاهی اجرا می‌شوند و اجازه دسترسی به تمام منابع موجود سیستم‌عامل به آن‌ها داده می‌شود. باوجود مجوز، دسترسی، بدافزار به تمام منابع موجود سیستم عامل دسترسی نموده، رفتار خود را کاملاً مشخص می‌نماید.

۳-۲- موتور تصمیم‌گیری و امکان‌سنجی

وظیفه موتور تصمیم‌گیری تشخیص وجود رفتار پنهان بدافزارهای هوشمند می‌باشد. موتور تصمیم‌گیری پیشنهادی ما، با مشاهده توابع API حساس و مورداستفاده اغلب بدافزارها [۱۳][۲۷][۲۸]، در جدول IAT متعلق به فایل اجرایی به آن فایل مشکوک می‌شود. اگر یکی از این توابع در زمان اجرا فراخوانی نشوند موتور تصمیم‌گیری مشکوک به کداجرایی در قالب یک بدافزار هوشمند می‌شود. لذا، درصورتی‌که نام یک تابع API حساس در جدول IAT درون فایل کد اجرایی موجود باشد، اما، در زمان اجرا این تابع API عملاً مورد فراخوانی قرار نگیرد، موتور تصمیم‌گیری برای ردگیری مسیرهای پنهان دستور تغییر شرایط محیط را می‌دهد. در اینجا منظور از مسیر اجرایی پنهان درواقع مسیری است که با برقراری شرایط محیطی بدافزار اجرا می‌شود. نکته قابل‌توجه در اینجا است، که با بررسی توابع API فراخوانی نشده موجود در جدول IAT می‌توان مشخص کرد که بدافزار هوشمند دنبال کدام شرایط و ورودی‌ها از محیط اجرایی می‌باشد. برای نمونه با توجه به وجود توابع Send، Socket، WSAStartup و Recv در جدول IAT و عدم فراخوانی این توابع در زمان اجرا، می‌توان متوجه شد که می‌بایست برای نمونه اینترنت و با شبکه محلی و یا سراسری قابل‌دسترسی گردد. بنابراین، بر این اساس شرایط محیطی دسترسی به شبکه باید ایجاد شود.

علاوه‌بر این با توجه به فراخوانی‌های سیستمی انجام‌شده می‌توان مشخص کرد که شرایط محیطی لازم برای اجرای مسیرهای پنهان چه می‌تواند باشد. برای نمونه، با اجرای فراخوانی InternetGetConnectedState در سطح کاربر می‌توان متوجه شد که برنامه نیاز به اینترنت دارد. بنابراین، برای تحلیل رفتار پنهان نرم‌افزار مربوطه می‌بایست اینترنت در دسترس قرار گیرد. به‌عنوان نمونه دیگر خیلی از بدافزارها به دنبال وجود سرویس‌های آسیب‌پذیر فعال هستند. برای این منظور تابع سیستمی openservice به‌صورت ذیل فراخوانی می‌شود:

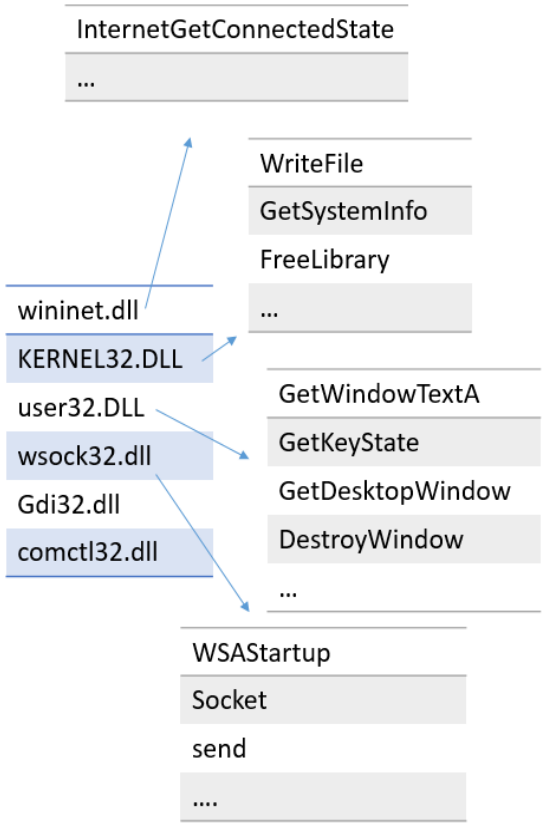
```
OpenService(
      schSCManager, //SCM DB.
    szSvcName,    //name    of
                          service
              SERVICE_STOP |
       SERVICE_QUERY_STATUS |

   SERVICE_ENUMERATE_DEPENDENTS);
```

با قرار دادن پارامتر szzSvcName مساوی با "LanmanServer" می‌توان وجود سرویس پروتکل SMB را بررسی کرد. آسیب‌پذیری قبلی موجود در این پروتکل عامل حملاتی مثل WannaCry بود. در این صورت می‌بایست سرویس SMB فعال شود تا کد مخرب مسیر پنهان خود را اجرا نماید.

به‌طورکلی با ایجاد شرایط محیطی به‌صورت مجازی و اجرای مجدد نرم‌افزار مورد تحلیل، می‌توان مشخص نمود که آیا برقراری شرایط در کشف مسیر پنهان تأثیرگذار بوده است. کار موتور تصمیم‌گیری درواقع پی بردن به رفتار گریز از تشخیص بدافزارهای هوشمند می‌باشد. طبق روال توضیح داده‌شده درصورتی‌که موتور تصمیم‌گیری مشکوک به رفتار فایل اجرایی شود، فایل را به مرحله تغییر شرایط محیط جهت کشف پنهان مسیر می‌فرستد.

برای درک بهتر حل مسئله، جاسوس‌افزار هوشمند Diablo را مورد بررسی قرار می‌دهیم. جاسوس‌افزار Diablo ازجمله بدافزارهای غیرقابل تشخیص توسط محصولات ضدبدافزاری زمان خود بود، که به بدافزار خاموش یا خوش خیم معروف بود. Diablo برای گریز از تشخیص توسط محصولات ضدبدافزاری شرایط محیط را بررسی کرده و سپس کدهای بدخواه خود را اجرا می‌نماید. یکی از شرایط لازم برای Diablo، وجود اینترنت می‌باشد. عدم وجود اینترنت باعث پنهان ماندن مسیرهای اجرایی مخرب از دیدگاه جعبه شنی در هنگام بررسی و اجرای Diablo می‌باشد.

موتور تصمیم‌گیری پیشنهادی ما با خواندن توابع API موجود در جدول IAT بدافزار Diablo، به این نتیجه می‌رسد که توابعی مثل Send، Socket، WSAStartup در جدول IAT موجود هست که هنگام تحلیل پویا فراخوانی نشده‌اند. درنتیجه مشکوک به وجود مسیر پنهان بدافزار شده و درخواست تحلیل فایل مشکوک در شرایط محیطی که امکان وجود شبکه و اینترنت می‌باشد را صادر می‌نماید. انتخاب شرایط محیطی برای تحلیل بدافزار Diablo بر اساس مفهوم و عملکرد توابع Send، Socket و WSAStartup می‌باشد. درواقع این سه تابع سیستمی ذکرشده در سیستم‌عامل ویندوز برای ارسال و دریافت اطلاعات در سطح شبکه مورداستفاده قرار می‌گیرند. همچنین موتور تصمیم‌گیری با فراخوانی تابع InternetGetConnectedState فراخوانی شده می‌فهمد که جاسوس‌افزار وجود اینترنت را جهت ارسال اطلاعات جاسوسی بررسی می‌نماید. در شکل (۳) جدول IAT مورداستفاده Diablo در زمان اجرا مشخص شده است. در این جدول FreeLibrary نمایانگر کتابخانه‌ای برای آزادسازی هر کتابخانه بارشده در زمان اجرا است.



شکل (۳): جدول IAT بدافزار Diablo

به‌طور خلاصه مراحل شناسایی شرایط محیطی توسط موتور تصمیم‌گیری به شرح ذیل می‌باشد:

۱- جعبه شن مسیرهای اجرایی پیمایش شده بدافزار را که توابع سیستمی فراخوانی شده می‌باشد را ثبت می‌نماید.
۲- لیست توابع موجود در جدول IAT خوانده می‌شود.
۳- توابع سیستمی به‌دست‌آمده از مرحله یک را با مرحله دو مقایسه نموده و درنتیجه به توابعی می‌رسد که زمان اجرا توسط فایل مشکوک استفاده نشده است.
۴- پی بردن به وجود مسیرهای اجرا نشده.
۵- طبق عملکرد توابع موجود در جدول IAT که زمان اجرا فراخوانی نشده‌اند. تغییر شرایط محیط انجام می‌شود.
۶- فرمان آزمون دوباره فایل در شرایط محیطی انتخاب شده.

تحلیل‌های که توسط شرکت‌های امنیتی معروف مثل Symantec، Norton و خود شرکت مایکروسافت بر روی بدافزار Stuxnet انجام شد. مشخص گردید که Stuxnet خاصیت هوشمند و رفتار مبهم را دارد. این بدافزار مخرب زمانی حملات خود را انجام می‌داد که دستگاه PLC متصل به سیستم‌عامل ویندوز بود و یا نرم‌افزار SCADA بر روی سیستم‌عامل نصب باشد. بررسی محیط موجود و تحریک در زمان خاص از ویژگی‌های بدافزارهای

هوشمند می‌باشد. تشخیص و آگاهی از شرایط محیط از ویژگی‌های بدافزارهای هوشمند می‌باشد که مراحل تشخیص را دشوارتر می‌نماید.

۳-۳- تغییر شرایط محیط

در این مرحله ما از یک خدمت‌گذار برای تغییر شرایط محیط استفاده کرده‌ایم. طبق ورودی دریافت شده از موتور تصمیم‌گیری شرایط محیط تحلیل به شرایط خواسته‌شده، توسط خدمت گذار برده می‌شود. شرایط محیطی تصمیم گرفته‌شده توسط موتور تصمیم‌گیری می‌تواند شامل امکان اتصال سیستم‌عامل به شبکه، امکان اتصال به اینترنت، باز بودن مرورگر، غیرفعال بودن دیوار آتشین، اتصال دستگاه خاص به سیستم‌عامل و یا هر خواسته دیگر توسط موتور تصمیم‌گیری باشد. فایل مشکوک ارسال‌شده طبق شرایط خواسته‌شده توسط موتور تصمیم‌گیری با مدیریت جعبه شن تحلیل خواهد شد. با دادن امکانات خواسته‌شده مثل وجود اینترنت، شبکه و غیره توسط موتور تصمیم‌گیری، خواسته‌های بدافزار هوشمند برآورده شده و زمان تحلیل و اجرا رفتار بدخواه خود را نشان خواهد داد.

برای نمونه با توجه به جدول (۱)، تحلیل بدافزار Diablo نشان می‌دهد با توجه به وجود توابع Send، Socket و WSAStartup در جدول IAT و عدم فراخوانی این توابع در زمان اجرا، می‌توان متوجه شد که بدافزار باید در شرایط محیطی که قابلیت دسترسی به شبکه و اینترنت هست، مورد تحلیل قرار گیرد. بنابراین، بدافزار Diablo رفتار پنهان خود که درواقع ارسال اطلاعات جاسوسی می‌باشد را نشان می‌دهد. انتخاب شرایط محیطی برای آزمون مجدد بدافزار Diablo بر اساس مفهوم و عملکرد توابع Send، Socket و WSAStartup می‌باشد. درواقع این سه تابع سیستمی ذکرشده در سیستم‌عامل ویندوز برای ارسال و دریافت اطلاعات در سطح شبکه مورداستفاده قرار می‌گیرند.

جدول (۱): شرایط لازم برای اجرای رفتار پنهان بدافزار

بدافزار	توابع موجود در جدول IAT	توابع فراخوانی نشده از جدول IAT	توابع بررسی شرایط محیط	شرایط موردنیاز برای اجرای رفتار پنهان
Diablo	WSAStartup Socket Send GetWindowsTextA LoadLibraryA InternetGetConnectedState GetwindowsDirectoryA GetSystemInfo GetWindowsDC GetKeyState FindResourceA LoadkeyboardLayoutA KillTimer …	WSAStartup Socket Send GetKeyState GetWindowsTextA …	InternetGetConnectedState GetSystemInfo …	Internet Access LAN Network Access Browse Directory Browser Execution
NetSky	LoadLibraryA GetProcAddress GetNetWorkParams InternetGetConnectedState VirtualProtect …	GetProcAddress VirtualProtect ..	InternetGetConnectedState GetNetWorkParams	Internet Access LAN Network Access
Wannacry	openServiceA openSCManagerA GetFileAttributesW GetFileSize CopyFileA GetComputernameW CryptReleaseContext GetwindowsDirectoryw LoadLibraryA FreeLibraryA Ws2_32.11 Ws2_32.4 …	Ws2_32.11 Ws2_32.4 ..	openSCManagerA openServiceA	SMB Services Access Network Access Internet Access

۳-۴- تحلیل عمیق فایل

فایل مشکوک در شرایط محیط خواسته‌شده مورد تحلیل و بررسی قرار می‌گیرد. این تحلیل مثل مؤلفه تحلیل و جمع‌آوری اطلاعات اولیه شامل تحلیل ایستا و پویا می‌باشد. نتیجه این مؤلفه وجود رخداد جدید نسبت به تحلیل قبلی خواهد بود. رخداد جدید سرنخی برای کشف مسیر اجرایی پنهان یا در حالت کلی شناسایی رفتار جدید از فایل مشکوک خواهد بود.

۳-۵- ثبت اتفاقات

اتفاقات جدید رخ داده بعدازاین تحلیل ثبت می‌گردد. وجود اتفاقات جدید رخداده را می‌توان از مقایسه خروجی‌های مؤلفه تحلیل و جمع‌آوری اطلاعات اولیه از فایل با مؤلفه تحلیل عمیق فایل فهمید. اتفاقات رخ داده جدید هدف و خواسته ما برای آشکارسازی و تشخیص رفتار بدخواه بدافزار می‌باشد.

۳-۶- الگوی تشخیص

برای تشخیص بدافزار از روی تحلیل‌های انجام‌شده توسط جعبه شن، محققین امنیت معمولاً از روش‌هایی مثل یادگیری ماشین، تابع رگرسیون، شبکه‌های پتری نت، ژنتیک و غیره، الگوهای تشخیص رفتار بدافزارها را مشخص می‌نمایند.

۴- ارزیابی روش پیشنهادی

روش پیشنهادی تشخیص گام‌به‌گام و تدریجی بر اساس فراخوانی‌های سیستمی که شرایط و وجود منابع خاص محیطی را بررسی می‌کنند بوده است. بر این اساس تعداد زیادی بدافزار مورد تحلیل و بررسی قرار داده شده‌اند که در این مقاله به سه بدافزار مطرح اکتفا شده است.

هدف ارائه روشی جهت برای شناسایی رفتار پنهان بدافزارهای هوشمند بوده است. در این راستا به بررسی ۷۵ نمونه از بدافزارهای شناخته‌شده مثل NetSky،Diablo، Stuxnet، SpyEye، WannaCry، Bancos و Zeus توسط ابزار حاصل از این تحقیق پرداختیم. کار ابزار ما همان‌گونه که در بالا توضیح داده شد تعیین و ایجاد، گام‌به‌گام شرایط محیطی است. با ایجاد این شرایط طبق جدول (۲)، برای بدافزارهای Diablo، NetSky و نهایتاً بدافزار WannaCry، توابع سیستمی حساس را با ایجاد تدریجی شرایط محیطی توانستیم شناسایی کنیم. این توابع زمان تحلیل پویای بدافزار توسط هیچ‌کدام از جعبه شن‌های شناخته‌شده Coco، Jeo و Anubis با اجرای این سه بدافزار قابل‌تشخیص نمی‌باشند.

طبق گزارش تحیل‌های انجام‌شده با استفاده از روش پیشنهادی ما که در جدول (۲) نشان داده‌شده است، بدافزار Diablo از توابع WsaStartup، Socket و Send برای ارسال اطلاعات جاسوسی در شبکه و اینترنت استفاده می‌نماید. تابع GetWindowsTextA برای گرفتن عنوان پنجره‌ها و GetKeyState جهت گرفتن کلیدهای فشرده شونده توسط کاربر مورداستفاده قرار گرفته است. این توابع به دلیل استفاده نسبتاً زیاد توسط بدافزارها، جزء لیست توابع حساس می‌باشند. در اولین اجرا دو تابع سیستمی برای گرفتن شرایط محیطی که در ستون چهارم جدول مشخص‌شده اجرا شدند. اولین تابع InternetGetConnectedState نشانگر نیاز بدافزار Diablo به برقراری اتصال با اینترنت است. لذا، در اولین گام ارتباط با اینترنت را برقرار نمودیم.

تابع GetSystemInfo سطح دسترسی به سیستم‌عامل را کنترل می‌کند که با غیرفعال یا در اصطلاح disable نمودن UAC گرفتن مجوز مدیر سیستم برای بدافزار امکان‌پذیر می‌شود. با برقراری این شرایط، توابع حساس مشخص‌شده در ستون سوم جدول (۲) به اجرا درآمده، مشخص می‌شود که شرایط محیطی برای بدافزار Diablo طبق آنچه در ستون پنجم جدول (۲) مشخص‌شده است، شامل امکان دسترسی به اینترنت و داشتن مجوز مدیر است. با ایجاد شرایط محیطی توانستیم در داخل جعبه شن خود لیستی از توابع سیستمی مندرج در ستون سوم جدول (۲) را مشخص کنیم. درحالی‌که هیچ‌کدام از جعبه شن‌های شناخته‌شده کنونی مثل Cuckoo، Jeo و Anubis قادر به شناسایی این دنباله فراخوانی‌های سیستمی نیستند. علت این است که هیچ‌یک از جعبه شن‌های شناخته‌شده کنونی امکان شناسایی تدریجی رفتار پنهان بدافزارهای هوشمند را ندارند.

درواقع ارزیابی ما از محیط پیشنهادی برای جعبه‌های شنی، ایجاد امکانی مطابق با روش پیشنهادی ما برای تشخیص وجود رفتار پنهان بدافزارها می‌باشد. مشکوک شدن منطقی، گام اول در کشف کدهای بدخواه می باشد. جهت ارزیابی روش پیشنهادی، در عمل سه بدافزار شناخته‌شده WannaCry، Diablo و NetSky در جدول (۲) مشخص‌شده است. همان‌گونه که در جدول (۲) مشخص شده است، تفاوت توابع API مورد دسترسی با توابع فراخوانی شده در زمان اجرا می‌تواند عاملی برای مشکوک شدن به کد اجرایی به‌عنوان یک بدافزار باشد. با توجه به این‌که تابع فراخوانی شده چه شرایط محیطی را بررسی می‌کند و به دنبال چه منابعی برای ادامه کار خود هست، می‌توان شرایط لازم برای اجرا و نمایش رفتار بدخواهانه را مشخص نمود.

به‌عنوان نمونه‌ای دیگر در کداجرایی کرم باج‌گیر WannaCry، طبق ستون سوم جدول (۲)، ازآنجایی‌که از توابع کتابخانه ws2_32.dll به‌عنوان مثال Ws2_32.11 و Ws2_32.4 برای ارسال

اطلاعات در شبکه و اینترنت مورد استفاده می‌شود، می‌توان متوجه شد که این تابع نیاز به برقراری ارتباط با شبکه را دارد. با دادن امکان شرایط دسترسی به شبکه و اینترنت رفتار پنهان بدافزار، درواقع انتشار خود در سطح شبکه کشف می‌شود. با توجه به این‌که تابع OpenServiceA به دنبال وجود سرویس فعال می‌باشد و با توجه به پارامترهای ارسالی برای این تابع نیز متوجه شد که این تابع به دنبال سرویس SMB فعال می‌باشد. در صورت فعال بودن سرویس SMB این بدافزار از طریق آسیب‌پذیری شناخته‌شده در آن امکان حمله و نفوذ در سطح شبکه را در اختیار می‌گیرد.

بررسی و تحقیقات ما ابتدا روی ۷۵ نمونه از بدافزارهای شناخته‌شده مثل NetSky، Diablo، Stuxnet، SpyEye، WannaCry، Bancos و Zeus انجام شد. کسب اطلاعات از شرایط محیط توسط بدافزارهای اشاره‌شده، یکی از عوامل شروع فعالیت ما برای درک مفهوم شرایط محیط و ره‌گیری آن می‌باشد.

همچنین با بررسی توابع API فراخوانی نشده موجود در جدول IAT بدافزار مورد تحلیل، نمایشی از وجود مسیر اجرایی پنهان مشخص می‌شود. در این قسمت مؤلفه موتور تصمیم‌گیری که در فرایند ارائه‌شده در شکل (۱) مشخص شده است، شرایط موردنیاز بدافزار را با توجه به جدول (۲) تشخیص داده و موردبررسی قرار می‌دهد.

با پیاده‌سازی فرایند پیشنهادی در شکل (۱) توانستیم در جعبه شن پارسا امکان بررسی شرایط لازم برای به اجرا درآوردن رفتار بدخواهانه بدافزارها را فراهم نماییم. در جدول (۱)، مقایسه امکانات این جعبه شن با سایر جعبه شن‌های موجود ارائه شده است.

جدول (۲): مقایسه جعبه شن پیشنهادی با نمونه‌های دیگر

محیط تحلیل	تغیر شرایط محیط	موتور تصمیم‌گیری	تحلیل پویا	روش کشف رفتار پنهان
Cuckoo sandbox	ندارد	ندارد	دارد	تحلیل استاتیک فایل و الگوهای تشخیص رفتاری
Jeo Sandbox	ندارد	ندارد	دارد	تولید ورودی‌های تصادفی و تحلیل پویا و استاتیک فایل
Anubis sandbox	ندارد	ندارد	دارد	تولید ورودی‌های تصادفی
Parsa sandbox	دارد	دارد	دارد	با استفاده از توابع فراخوانی نشده از جدول IAT و تغییر شرایط محیط همچنین تولید ورودی‌های تصادفی

نکته قابل‌توجه در تشخیص شرایط محیطی لازم برای اجرای مسیرهای پنهان این است که این شرایط به‌مرور مشخص می‌شوند. بدین معنا که با تشخیص یک شرط محیطی و ایجاد منابع برای برقراری آن شرط، در اجرای بعدی ممکن است شرایط محیطی جدیدتر و درواقع نیازهای بیشتری مشخص شوند. در شکل ذیل برای نمونه تعداد مراحلی که برای تشخیص شرایط در شکل ۴ نرخ کشف رفتارهای مخرب بدافزار Diablo در واکنش به شرایط محیطی ارائه شده است. همان‌گونه که در این شکل مشخص است، در ابتدا ۴۰٪ از رفتار بدخواهانه با برقراری شرایط اولیه تشخیص داده شده است. نهایتاً بعد از ده مرحله اجرا توانستیم درمجموع حدوداً ۸۰٪ از رفتارهای بدخواهانه بدافزار کشف شد. با ادامه تغییر شرایط محیطی رفتارهای پنهان دیگر جاسوس‌افزار اکتشاف شد.

در مجموع رفتار بدخواه این جاسوس‌افزار که شامل نوشتن خود در بوت سیستم‌عامل، اخذ کلیدهای فشرده‌شده توسط کاربر، عکس‌برداری از صفحه‌نمایش، خواندن عنوان پنجره‌ها، خواندن اطلاعات سیستمی، ارسال اطلاعات جاسوسی و رفتارهای مخرب دیگر می‌باشد، تشخیص داده شد.

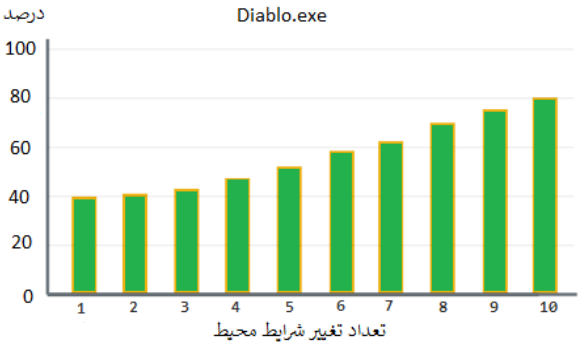
شرایط محیط آزمون، توسط مؤلفه موتور تصمیم‌گیری درخواست می‌شود. نتیجه و روند مراحل آزمون جاسوس‌افزار Diablo به‌اختصار شامل موارد ذیل می‌باشد.

مرحله یک آزمون در شرایط اولیه بدون تغییر شرایط محیط انجام شد. مرحله دوم با امکان مجوز دسترسی بدافزار، برای تغییر ثبت کننده یا در اصطلاح رجیستری سیستم‌عامل ویندوز انجام شد. چراکه بدافزار با اجرای تابع سیستمی GetSystemInfo شرایط محیطی برای اجراشدن در سطح مدیر را بررسی می‌نماید. لذا، موتور تصمیم‌گیری با تشخیص این فعالیت، دستور تحلیل بدافزار را در شرایط محیطی که مجوز مدیر را داشته باشد صادر می‌نماید. بنابراین، بدافزار قادر به تغییر رجیستری سیستم‌عامل ویندوز شده و رفتار پنهان خودش را که درواقع اجرای کد مخرب زمان راه‌اندازی مجدد سیستم‌عامل می‌باشد اعمال می‌نماید.

مرحله سوم با تولید کلیدهای فشرده‌شده مجازی، رفتار پنهان اخذ کلیدهای فشرده توسط جاسوس‌افزار کشف شد.

چراکه موتور تصمیم‌گیری با تشخیص وجود تابع حساس GetkeyState در لیست فهرست جدول IAT و عدم اجرای آن زمان تحلیل پویا توسط بدافزار، به وجود رفتار پنهان بدافزار مشکوک می‌شود. با تشخیص این فعالیت بدافزار، دستور تحلیل بدافزار را در شرایط محیطی که کلیدهای فشرده‌شده مجازی تولید شوند را صادر می‌نماید. بنابراین، بدافزار تحریک‌شده و شروع به گرفتن کلیدهای فشرده می‌نماید. بنابراین، رفتار پنهان بدافزار که درواقع اخذ کلیدهای فشرده‌شده کاربر می‌باشد؛ تشخیص داده می‌شود.

درنهایت، در مرحله ده با دادن امکان دسترسی به اینترنت، رفتار پنهان ارسال اطلاعات جاسوسی تشخیص داده شده است.



شکل (۴). نرخ کشف رفتارهای مخرب جاسوس‌افزار Diablo

به تعداد دفعات تغییر شرایط محیط طبق دستور و درخواست صادره از موتور تصمیم‌گیری آزمون اکتشاف مسیرهای اجرایی پنهان فایل مشکوک انجام می‌شود. در هر مرحله، آزمون و تحلیل بدافزار در شرایط محیطی خواسته‌شده توسط موتور تصمیم‌گیری انجام می‌شود. لذا، رفتارهای پنهان بدافزار Diablo در هر مرحله از آزمون مجدد آشکار می‌گردد.

۵- نتیجه‌گیری

جهت بررسی و تحلیل رفتاری بدافزارهای هوشمند، می‌توان با نظارت بر فراخوانی‌های سیستمی، به‌مرور منابع و شرایط لازم برای اجرای بدخواهانه بدافزار هوشمند را مشخص و مهیا نمود. هر بار با فراهم کردن شرایط محیطی و منابع موردنیاز، بدافزار به مسیر خود ادامه داده و با اجرای توابع سیستمی بعدی می‌توان از منابع و شرایط بعدی به‌مرور آگاهی پیدا کرد تا نهایتاً به‌طور کامل شرایط و منابع محیطی برای اجرای بدخواهانه بدافزار مشخص شوند. با فراهم نمودن شرایط محیطی، در محیط جعبه‌های شنی، می‌توان شاهد رفتار بدخواهانه یک بدافزار هوشمند بود.

جعبه شن‌های کنونی جهت تشخیص بدافزارهای هوشمند نیاز به تحلیل عمیق فایل اجرایی در شرایط مختلف سیستم‌عامل دارند. چراکه بدافزارهای هوشمند در صورت برآورده شدن برخی شرایط خواص رفتار مخرب خود را نشان می‌دهند. بدافزارها با خصوصیت هوشمند حملات سایبری جدی و مخرب را انجام داده‌اند. درحالی‌که تحقیقات انجام‌شده از طرف شرکت‌های امنیتی روی چنین بدافزارهای همچنان کم می‌باشد.

مؤلفه موتور تصمیم‌گیری که پی به شناخت وجود مسیرهای اجرانشده فایل مشکوک می‌نماید یکی از نوع آوری‌های ما در این مقاله هست. همچنین تغییر شرایط محیط برای برآورده شدن خواسته‌های بدافزار جهت تحریک و نشان دادن رفتار مخرب خود یکی دیگر از نوآوری‌های تحقیقات انجام‌شده می‌باشد.

۶- منابع

[1] V. S. Subrahmanian, et al, “Types of Malware and Malware Distribution Strategies,” The Global Cyber-Vulnerability Report, Springer International Publishing, pp. 33-46, 2015.

[2] A. Moser, K. Christopher, and K. Engin, “Exploring multiple execution paths for malware analysis,” Security and Privacy, 2007. SP’07. IEEE Symposium on. IEEE, 2007.

[3] D. Brumley, et al, “Automatically identifying trigger-based behavior in malware,” Botnet Detection, pp. 65-88, 2008.

[4] B. Kang, J. YANG, J. So, and C. Y. Kim, “Detecting Trigger-based Behaviors in Botnet Malware,” In Proceedings of the 2015 Conference on research in adaptive and convergent systems, ACM, 2015.

[5] S. Bahtiyar, “Anatomy of targeted attacks with smart malware,” Security and Communication Networks 9.18, pp. 6215-6226, 2016.

[6] G .Hăjmăşan, M. Alexandra, and C. Octavian, “Dynamic behavior evaluation for malware detection,” Digital Forensic and Security (ISDFS), 2017 5th International Symposium on. IEEE, 2017.

[7] R. de Tangil and S.Guillermo, “Mining structural and behavioral patterns in smart malware,” Diss. Universidad Carlos III de Madrid, 2014.

[8] C. Matthew, T. Liston, and E. Skoudis, “Hiding virtualization from attackers and malware,” IEEE Security & Privacy, vol. 5, no. 3, 2007.

[9] M. Mehra and P. Dhawal, “Event triggered malware: A new challenge to sandboxing,” India Conference (INDICON), 2015 Annual IEEE. IEEE, 2015.

[10] S. N. Alsagoff, “Malware self protection mechanism,” Information Technology, 2008. ITSim 2008. International Symposium on. vol. 3, IEEE, 2008.

[11] K. Navroop, A. K. Bindal, and A. PhD, “A Complete Dynamic Malware Analysis,” International Journal of Computer Applications, vol. 135, no. 4, pp. 20-25, 2016.

[12] D. Keragala, “Detecting malware and sandbox evasion techniques,” SANS Institute InfoSec Reading Room, vol. 16, 2016.

[13] C. Ravi and R. Manoharan, "Malware detection using windows api sequence and machine learning," International Journal of Computer Applications, vol. 43, no. 17, pp. 12-16, 2012.

[14] L. Desmond, P. Watters, and X. Wu, "Rbacs: Rootkit behavioral analysis and classification system," Knowledge Discovery and Data Mining, WKDD'10. Third International Conference on, 2010.

[15] D. Vidyarthi, S. P. Choudhary, S. Rakshit, and C. Kumar, "Malware Detection by Static Checking and Dynamic Analysis of Executables," International Journal of Information Security and Privacy, 2017.

[16] P. Xie, et al., "An automatic approach to detect anti-debugging in malware analysis," International Conference on Trustworthy Computing and Services, Springer, Berlin, Heidelberg, 2012.

[17] B. Kang, J. Yang, J. So, and C. Y. Kim, "Detecting Trigger-based Behaviors in Botnet Malware," In Proceedings of the 2015 Conference on research in adaptive and convergent systems, 2015.

[18] M. Lindorfer, C. Kolbitsch, and P. M. Comparetti, "Detecting environment-sensitive malware," In International Workshop on Recent Advances in Intrusion Detection, 2011.

[19] D. Brumley, C. Hartwig, Z. Liang, J. Newsome, D. Song, and H. Yin, "Automatically Identifying Trigger-based Behavior in Malware," In Botnet Detection, Springer, pp. 65-88, 2008.

[20] Suarez-Tangil, M. Conti, J. E. Tapiador, and P. Peris-Lopez, "Detecting targeted smartphone malware with behavior-triggering stochastic models," In In European Symposium on Research in Computer Security, 2014.

[21] A. Moser, C. Kruegel, and E. Kirda, "Exploring multiple execution paths for malware analysis," in Proceedings- IEEE Symposium on Security and Privacy, 2007.

[22] D. Fleck, A. Tokhtabayev, A. Alarif, A. Stavrou, and T. Nykodym, "PyTrigger: A system to trigger & extract user-activated malware behavior," in International Conference on Availability, Reliability and Security, 2013.

[23] S. Ranu and A. K. Singh, "GraphSig: a scalable approach to mining significant subgraphs in large graph databases," In IEEE 25th International Conference on Data Engineering, 2009.

[24] R. Majumdar and S. Koushik, "Hybrid concolic testing," Software Engineering, 2007. ICSE 2007. 29th International Conference on. IEEE, 2007.

[25] X. Xu, et al., "Software backdoor analysis based on sensitive flow tracking and concolic execution," Wuhan University Journal of Natural Sciences vol. 21, no. 5, pp. 421-427, 2016.

[26] H. Yin and S. Dawn, "Hooking Behavior Analysis," Automatic Malware Analysis, Springer, New York, pp. 43-58, 2013.

[27] K. Youngjoon, E. Kim, and H. Kang Kim, "A novel approach to detect malware based on API call sequence analysis," International Journal of Distributed Sensor Networks, vol. 11, no. 6, 2015.

[28] J. Berdajs and Z. Bosnić, "Extending applications using an advanced approach to dll injection and api hooking," Software: Practice and Experience, vol. 40, no. 7, pp. 567-584, 2010.

[29] J. M. Ceron, C. B. Margi, and L. Zambenedetti, "MARS: An SDN-based malware analysis solution," In IEEE Symposium on Computers and Communication (ISCC), 2016.

[30] D. Oktavianto and M. Iqbal, "Cuckoo Malware Analysis," Packt Publishing Ltd, 2013.

[31] C. Annachhatre, T. H. Austin, and M. Stamp, "Hidden Markov models for malware classification," Journal of Computer Virology and Hacking Techniques 11.2, pp. 59-73, 2015.

[32] N. Nissim, et al., "Novel active learning methods for enhanced PC malware detection in windows OS," Expert Systems with Applications 41.13, pp. 5843-5857, 2014.

[33] I. Rafiqul, et al., "Classification of malware based on integrated static and dynamic features," Journal of Network and Computer Applications, vol. 36, no. 2, pp. 646-656, 2013.

[34] I. Santos, et al., "Opem: A static-dynamic approach for machine-learning-based malware detection," International Joint Conference CISIS'12-ICEUTE 12-SOCO 12 Special Sessions, Springer, Berlin, Heidelberg, 2013.

[35] S. Silnov and T. O. Vladimirovich, "Analysis of Modern Attacks on Antiviruses," Journal of Theoretical & Applied Information Technology, vol. 76, no. 1, 2015.

[36] M. Lindorfer, K.Clemens, and P. Milani Comparetti, "Detecting environment-sensitive malware," Recent Advances in Intrusion Detection. Springer Berlin/Heidelberg, 2011.

[37] S.T. King and P. M. Chen, "implementing malware with virtual machines," Security and Privacy, IEEE, 2006.

[38] D. Keragala, "Detecting Malware and Sandbox Evasion Techniques," SANS Institute InfoSec Reading Room, 2016.

[39] A. Lakhani, "Malware Sandbox and Breach Detection Evasion Techniques," Doctor Chaos, 18 February 2016. [Online]. Available: http://www.drchaos.com/malware-sandbox-and-breach-detection-evasion-techniques/. [Accessed 2016].

[40] A. B. Cesar and D. Andrade, "Malware Automatic Analysis," Computational Intelligence and 11th Brazilian Congress on Computational Intelligence, 2013.

[41] U. Bayer, K. Christopher, and K. Engin, "TTAnalyze: A tool for analyzing malware," na, 2006.

[42] T. Smith and M. Waterman, "Identification of common molecular subsequences," Journal of molecular biology, pp. 195-197, 1987.

[43] B. Yadegari and S. Debray, "Symbolic Execution of Obfuscated Code," In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015.

[44] X. Chen, et al., "Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware." Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on. IEEE, 2008.

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

A New Method For Gradual Detection of Environmental Conditions and Resources Required by Smart Malware

S. Parsa*, H. khoshruy
Iran University of Science and Technology (IUST)
(Received: 18/10/2017, Accepted: 27/05/2018)

ABSTRACT

Smart malware samples have two different types of behaviors, namely defensive and aggrasive which they exhibit according to environmental conditions. This article offers a new method for detection of environmental conditions suitable for exhibition of aggrasive behaviors. Considering the list of system functions, apparant in the IAT table of a malware, those APIs which are not invoked at runtime could be identified as grounds for suspecting the executable file as a malware. Analyzing the functionality and task of these APIs and the ones invoked at runtime, the conditions and resources required for the malware to reveal its malicious behavior, could be determined. In fact, supplying all the required conditions and resources requested through one or more API calls, at a run, the malware could be prepared for asking for the next possible resource in the next run. This process could be repeated as far as no more conditions or resources are looked for. In order to evaluate the suggested method, three known malware samples are analysed in our sandboxing environment, Parsa.

Keywords: Sandbox, Malware, Smart Malware, Malware Analysis, Conditions Environment

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