مجله علمی- پژوهشی «پدافند الکترونیکی و سایبری»
سال دوم، شماره ۳، پاییز ۱۳۹۳؛ ص ۷۶-۶۵

ارائه یک محیط امن برای تحلیل رفتار بدافزارها

دانیال جواهری[1*]، سعید پارسا[2]

۱- کارشناسی ارشد دانشگاه آزاد اسلامی واحد بروجرد، دانشکده تحصیلات تکمیلی، گروه کامپیوتر
۲- دانشیار دانشگاه علم و صنعت ایران، دانشکده مهندسی کامپیوتر
(دریافت: ۹۳/۰۳/۱۹، پذیرش: ۹۳/۰۶/۳۰)

چکیده

این مقاله یک محیط تحلیل‌گر مبتنی بر جعبه شن در سمت کاربر برای اجرای بی‌خطر و ایمن یک برنامه مشکوک و ناشناس به‌منظور تعیین بدخواه یا بی‌خطر بودن و همچنین اجرای کنترل شده یک بدافزار برای تشخیص نحوه عملکرد و الگوی آلودگی آن برای مدل‌سازی رفتار به‌منظور ارائه راه‌کار خنثی‌سازی و رفع آلودگی را ارئه می‌کند. اهمیت اجرا و تحلیل بدافزار به‌صورت پویا در رفع مشکلات موجود برای کشف و تحلیل بدافزارهای مبهم شده، دگردیس و چندریخت که با روش‌های مبتنی بر امضاء و تحلیل‌های رفتار ایستا قابل کشف نیستند غیرقابل چشم‌پوشی است و هدف اصلی این مقاله که فراهم کردن بستر لازم برای تحلیل رفتار پویا است را شامل می‌شود. جعبه شن پیشنهادی با نظارت، رهگیری و کنترل درخواست‌ها و تعاملات برنامه تحت تحلیل در سطح کاربر و هسته سیستم عامل امکان استخراج رفتار را فراهم می‌کند در طول فرآیند تحلیل جعبه مسئول پاسخ‌گویی به درخواست‌های برنامه تحت تحلیل است این مقاله با مکاشفه بروی تعداد ۲۱۰۰۰ نمونه بدافزار و برنامه بی‌خطر امکان دسته‌بندی درخواست‌ها را در ۸ خانواده برای پاسخ‌گویی مناسب فراهم کرده است جعبه شن پیشنهادی درخواست‌های وارده را با ۵ سیاست شامل ثبت تنها، منحرف‌سازی، محدودسازی، فریب دادن و مجازی‌سازی منابع سیستم عامل پاسخ می‌دهد و تضمین می‌کند در طی اجرا و تحلیل بدافزار به سیستم کاربر هیچ‌گونه آسیبی وارد نشود. این مقاله همچنین چالش‌های موجود بر محیط‌های تحلیل را مورد آسیب‌شناسی قرار داده و راهکارهایی برای عبور از آن‌ها بیان می‌کند این چالش‌ها عمدتاً راه‌های شناسایی و خروج از محیط تحلیل‌گر می‌باشند. این مقاله در پایان جعبه شن پیشنهادی را از حیث توانایی و قابلیت‌های رهگیری رفتار و میزان مصرف منابع سیستمی ارزیابی و با برترین نمونه‌های خارجی مقایسه می‌کند.

واژه‌های کلیدی: محیط تحلیل‌گر، جعبه شن، تحلیل بدافزار، تأیید نرم‌افزار، رهگیری توابع سیستمی، قلاب‌سازی

۱. مقدمه

امروزه نرخ تولید بدافزارها از برنامه‌های بی‌خطر بیشتر شده است [۱] بنابر گزاش شرکت McAfee از مرجع [۲] روزانه بیش از ۱۰۰.۰۰۰ بدافزار جدید تولید می‌شود یعنی ۶۹ بدافزار در هر دقیقه بنابراین می‌توان ادعا کرد که فرکانس تولید بدافزار امروزه تقریباً ۱ بدافزار بر ثانیه است که علت آن هم ابزارهای تولید کد، مبهم‌ساز و چندریختی است. همچنین بر حسب گزارش مرجع [۳] در سال ۲۰۱۳ تقریباً ۴۷٪ سازمان‌های رسمی دولتی و خصوصی پدیده نفوذ در شبکه و آلودگی سیستم‌هایشان را تجربه کرده‌اند با بررسی دقیق این بدافزارها به این نتیجه می‌رسیم که بسیاری از بدافزارهای جدید، در وقاع همان بدافزارهای قبلی بوده‌اند که با اعمال تغییراتی در کد منبع جهت دور زدن برنامه‌های ضد ویروس و تحلیل‌گر منتشر شده‌اند. امروزه بسیاری از برنامه نویسان بدخواه با استفاده از موتورهایی برای انجام خود تغییری[1] و مبهم سازی[2] اقدام به انتشار بدافزارهای جدید می‌کنند که در حقیقت نسخه جدیدی از بدافزار قبلی پایه بوده‌اند با مسلح شدن بدافزارها به این برنامه‌ها که با روش‌های مختلفی یک عملیات یکسان را انجام می‌دهند ناکارامدی برنامه‌های ضد ویروس مبتنی بر امضاء و تحلیل رفتار ایستا نمایان شد و ابزارهای ضد بدافزار ناچار برای تحلیل و تشخیص ماهیت یک برنامه به تحلیل رفتار پویا و اجرای آن روی آوردند علت این امر در آن است که در زمان ایستا و قبل از اجرای یک برنامه، می‌توان با روش‌های ذکر شده و روش‌های دیگری نظیر رمزنگاری کد و تولید کد در زمان اجرا[3] مبادرت به مخفی کردن رفتار نمود [۱] و تحلیل‌گرهای ایستا که فارغ از اجرا برنامه و فقط با بررسی کد منبع سعی در تشخیص رفتار برای تعیین بدافزار بودن یا بی‌خطر بودن برنامه را دارند به اشتباه انداخت و لیکن یک برنامه هرچند هم که از روش‌های مذکور برای مخفی کردن رفتار خود در زمان ایستا داشته باشد ناچار است در زمان اجرا جهت استفاده از منابع سیستم، رفتار خود را نمایان کند [۱] پس می‌توان در زمان اجرا با نظارت کردن بر اجرای برنامه رفتار آن را با توجه استفاده از منابع سیستم عامل و

* رایانامه نویسنده مسئول: d.javahery@iaub.ac.ir

1. Self-Modifying
2. Obfuscation
3. Dynamic Programing

عملیات‌های انجام شده مشخص نمود. چالش اساسی و مهم در این مقوله، آسیب‌پذیری‌های ناشی از اجرای یک برنامه بدخواه است [۴]. راه‌کار برای حل این مشکل اجرای برنامه ناشناس در یک محیط امن و کنترل شده برای پی بردن به ماهیت آن است لذا برخی از شرکت‌های تولید کننده ضد بدافزار مانند Comodo و Avest اقدام به طراحی و توسعه محیط‌های تحلیل‌گر برای استفاده در برنامه‌های ضد بدافزار خود نمودند. این مقاله یک محیط تحلیل‌گر امن برای اجرای کنترل شده و بی‌خطر برنامه ناشناس به‌منظور استخراج رفتار برای تعیین ماهیت بدخواه یا بی‌خطر بودن آن را پیشنهاد می‌کند. محیط تحلیل‌گر پیشنهادی این مقاله امکان اجرای ایمن بدافزارها به گونه‌ای که به سیستم عامل کاربر آسیب وارد نشود را داشته و با نظارت دقیق بر تعاملات برنامه تحت تحلیل با سیستم عامل رفتار برنامه را استخراج و بستر لازم برای تحلیل رفتار پویا را فراهم می‌کند. هدف از طراحی و توسعه این محیط تجهیز آزمایشگاه‌های تحلیل بدافزار در داخل کشور و استفاده در ضد ویروس بومی پارسا می‌باشد.

این مقاله در ادامه چنین تنظیم شده است در قسمت دوم روش‌های تحلیل بدافزار و محیط‌های تحلیل‌گر معرفی می‌شوند، در قسمت سوم روش پیشنهادی برای ایجاد یک محیط تحلیل‌گر امن تشریح می‌شود، در قسمت چهارم آسیب پذیری‌ها و چالش‌های محیط‌های امن بررسی و راه‌کارهایی برای مقابله با آن‌ها پیشنهاد می‌شود، و در قسمت پایانی محیط پیشنهادی از حیث توانایی و قابلیت‌های رهگیری ارزیابی و با نمونه‌های خارجی موجود مقایسه خواهد شد.

۲. معرفی محیط‌های امن

با آنچه که در مقدمه این مقاله گفته شد ضرورت وجود یک محیط امن برای تحلیل بدافزارهای امروزی و برنامه‌های مشکوک و ناشناخته برای تعیین ماهیت آن‌ها کاملاً مشخص شد در ادامه به معرفی و بررسی محیط‌های اجرای بی‌خطر و تحلیل‌گر می‌پردازیم.

در زمینه امنیت کامپیوتر ظرف شن یک مکانیزم امنیتی جهت جداسازی و ایزوله کردن برنامه‌های در حال اجرا است از این ویژگی در اغلب موارد جهت اجرا و تحلیل کد یا برنامه بدخواهی که توسط هیچ شخص ثالث و یا کاربر یا وب سایت مطمئنی تأیید نشده باشد[۵] این مفهوم را نیز می‌توان در زمینه تحلیل بدافزار توسعه داد: هدف ما اجرای‌یک برنامه و یا فایل غیرقابل اطمینان در داخل یک محیط ایزوله جهت کسب اطلاعات درباره ماهیت و رفتار آن است جعبه شن، بدافزارها را به‌صورت پویا تحلیل می‌نمایند به عبارت دیگر بجای آنکه بدافزار به‌صورت ایستا تحلیل گردد آن را اجرا کرده و در زمان اجرا رفتار آن نظارت می‌شود این رویکرد دارای جوانب مثبت و منفی است اما مهم آن است که این روش به‌دلیل کسب اطلاعات موثر و مفید در مورد بدافزار همانند رفتارهای تحت شبکه بسیار با ارزش و مفید است البته جهت کسب اطلاعات عمیق و موثر در مورد بدافزار بهتر است که در مواجه با بدافزار هم به صورت ایستا و هم به‌صورت پویا تحلیل گردد [۶].

۲. ۱. محیط ماشین مجازی[1]

ماشین‌های مجازی محیط‌های مناسبی برای اجرای بی‌خطر یک بدافزار یا برنامه مشکوک به‌منظور تحلیل آن می‌باشد در این روش یک سیستم عامل در یک ماشین مجازی توسط یکی از ابزارهای مجازی‌سازی نظیر VM-Ware ، Hyper-V ، Xen و VirtualBox تحت عنوان ماشین میهمان بر روی یک سیستم عامل و ماشین دیگر تحت عنوان ماشین میزبان نصب و راه‌اندازی می‌شود [۷] سپس بدافزار یا برنامه مشکوک در محیط ماشین میهمان اجرا می‌شود و نتایج اجرا و خروجی آن برای تحلیل به ماشین میزبان فرستاده می‌شود این محیط به‌دلیل اینکه تمام امکانات یک سیستم عامل را در اختیار برنامه قرار می‌دهد امکان رهگیری و تشخیص دقیق تر بدافزار را نسبت به سایر روش‌ها ممکن می‌سازد و همچنین امکان تشخیص جعلی بودن محیط را برای بدافزارهایی که قصد جلوگیری از اجرا و افشای عملکرد خود در این محیط‌ها دارند را نیز سخت‌تر می‌کند این محیط‌های امن در پایان اجرا در صورت تشخیص بدافزار و آسیب وارد شدن به ماشین میهمان اقدام به نصب و راه‌اندازی مجدد سیستم عامل روی ماشین میهمان می‌نمایند و همواره تعدادی ماشین آماده برای تحلیل در اختیار خواهند داشت این روش به‌دلیل حجم منابع مورد نیاز برای راه‌اندازی امکان استفاده در سمت کاربر بسیار سخت می‌کند لذا این روش غالباً در سمت سرور ارائه می‌شود که اغلب با تعریف وب سرویس امکان استفاده را برای کاربران فراهم می‌کنند [۸] از جمله پرکاربردترین این تحلیل‌گرها می‌توان به Anubis و Cuckoo اشاره کرد، محیط تحلیل‌گر Cuckoo از یک ماشین میزبان لینوکس اوبونتو و یک ماشین میهمان ویندوز XP sp3 بهره می‌برد البته این محیط قابلیت استفاده از سیستم عامل MAC به‌عنوان میزبان و ویندوز ۷ به‌عنوان میهمان را نیز دارد [۹].

۲. ۲. محیط جعبه شن[2]

جعبه شن‌ها محیط‌های اجرایی هستند که با شبیه‌سازی، مجازی‌سازی و مخفی‌سازی امکانات سیستم عامل، محیط ایمنی را برای اجرای تحت کنترل بدافزارها فراهم می‌کنند. جعبه شن‌ها این موارد را با قلاب‌اندازی[3] به امکانات سیستم عامل و یا تزریق کتاب خانه‌های پیوند پویا[4] به برنامه‌های مورد نظر برای تحلیل انجام می‌دهند و عمدتاً در دو نسخه سمت کاربر و کارگزار ارائه می‌شوند [۵].

۲. ۳. جعبه شن سمت کاربر[5]

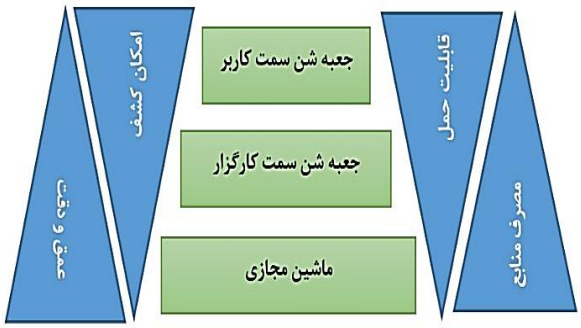
این جعبه شن‌ها دارای امکانات کمتری نسبت جعبه شن‌های سمت کارگزار هستند زیرا میزان استفاده از منابع سیستم و همچنین

1. Virtual Machine Environment
2. Sandbox Environment
3. Hooking
4. DLL Injection
5. Client Side Sandbox

قابلیت حمل برنامه بسیار حایز اهمیت است و چون قرار است برنامه در سیستم‌های مختلف با پیکربندی‌های مختلف اجرا شود بنابراین باید تا حد امکان از لحاظ میزان مصرف منابع بهینه باشند همچنین بحث امنیت و ایجاد محیط امن که در هنگام تحلیل بدافزار را تحت کنترل داشته باشد بسیار حایز اهمیت است زیرا بدافزار یا برنامه مشکوک و ناشناخته تحت سیستم‌عامل کاربر اجرا شده که در صورت از دست رفتن کنترل خسارات وارده به کاربر جبران‌ناپذیر است لذا این محیط‌ها برخی از امکانات سیستم‌عامل که می‌تواند عامل از دست رفتن کنترل شود را محدود می‌کنند.

۲. ۴. جعبه شن سمت کارگزار[1]

این جعبه شن‌ها چون در سمت کارگزار اجرا می‌شوند معمولاً با محدودیت چندانی برای استفاده از منابع مواجه نخواهند بود و مصرف بهینه منابع به‌عنوان یک گزینه جانبی ملاک قرار می‌گیرد این جعبه شن‌ها می‌توانند با شبیه‌سازی و مجازی‌سازی بسیاری از امکانات و سرویس‌های سیستم عامل امکان کشف دقیق‌تر بدافزارها را با شانس کمتر برای شناسایی شدن توسط بدافزارها را فراهم آورند [۱۰] مضاف به اینکه در صورت از دست دادن کنترل بدافزار، امکان بازیابی و نصب مجدد سیستم عامل در سمت کارگزار مشکلی چندانی نخواهد بود ولی در سمت کاربر شرایط کاملاً متفاوت است و در صورت وارد شدن خسارت به سیستم کاربر اعتبار سازنده از بین خواهد رفت و اعتماد دیگران نیز سلب خواهد شد.



شکل ۱. مقایسه بین محیط‌های تحلیل‌گر

همان‌گونه که در شکل فوق مشخص است امکان کشف محیط تحلیل‌گر در جعبه شن‌های سمت کاربر به‌دلیل محدودیت‌های ناشی از اجرا بروی سیستم کاربر از سایر روش‌ها بیشتر بوده و به همین دلیل کمترین میزان کشف و عمق تحلیل را دارند. بیشترین دقت در کشف و عمق تحلیل را ماشین‌های مجازی داشته حال آنکه جعبه شن‌های سمت کاربر بیشترین قابلیت حمل را با کمترین میزان مصرف منابع به همراه دارند در این خصوص ماشین‌های مجازی بیشترین میزان مصرف منبع و کمترین قابلیت حمل قراردارند در تمامی موارد جعبه شن‌های سمت کارگزار حالت میانی دارند.

1. Server Side Sandbox

۲. ۵. قلاب‌سازی

قلاب‌اندازی، یک مفهوم استفاده شده برای به‌دست آوردن کنترل جریان اجرایی برنامه بدون تغییر و کامپایل مجدد کد منبع آن است این کار، توسط متوقف‌سازی فراخوانی تابع و هدایت مجدد آن به کد سفارشی شده، به‌دست می‌آید با ارائه کد سفارشی، هر عملیاتی را می‌توان اجرا نمود پس از آن، قابلیت‌های اصلی تابع می‌تواند اجراشده و نتیجه می‌توانديا به سادگی برگشت داده شود و یا تغییر داده شده و برای انتقال کنترل به کدی که تابع قلاب شده را فراخوانی کرده، برگشت داده شود بنابراین، قلاب‌سازی، یک ابزار مناسب و کاملی را برای تحلیل مخرب‌ها به‌صورت پویا، فراهم می‌کند [۱۲،۱۱] قلاب‌سازی انواعی دارد، قلاب‌های درون خط به طور مستقیم، بایت‌های کد تابع را در حافظه بازنویسی می‌کنند به‌طور خاص، تنها چند دستورالعمل با یک دستورالعمل پنج بایتی پرش (jmp) به تابع قلاب جایگزین می‌شوند دستورالعمل‌های جایگزین شده در تابع trampoline که به‌عنوان نقطه ورودی جدید به فراخوانی API اصلی استفاده می‌شود، ذخیره می‌شوند در تابع قلاب، ثبت اطلاعات و اصلاح آرگومان‌ها قبل از اجرای تابع API اصلی می‌تواند انجام شود [۱۳] این نوع از قلاب‌اندازی مستلزم تغییر در فایل‌های سیستم عامل است لذا پیاده‌سازی آن بسیار سخت بوده و گاهاً سبب ایجاد ناسازگاری در سیستم عامل می‌شود.

روش دیگر در قلاب‌اندازی، تغییر آدرس وقفه‌های سیستمی است اما پیاده‌سازی این روش بسیار سخت و در بسیاری از مواقع غیر ممکن است زیرا به ندرت اطلاعاتی در خصوص وقفه‌های سیستمی منتشر شده است و نمی‌توان دریافت که یک وقفه با یک کد خاص برای چه رویدادی تعریف شده است. جدول توصیف‌گر این وقفه‌ها[2] در پایین‌ترین سطح در هسته سیستم عامل قرار داشته و تعداد وقفه‌های تعریف شده در آن زیاد است لذا امکان شناسایی عملکرد آن‌ها با راهکارهایی نظیر اشکال‌زدایی هسته سیستم توسط ابزارهای اشکال‌زدا مانند Windbg عملاً غیر ممکن است.

روش دیگری برای قلاب‌سازی که بیشترین کاربرد را دارد تغییر آدرس توابع سیستمی[3] است آدرس توابع سیستمی در جدول توصیف‌گر توابع سیستمی[4] (SSDT) و جدول توصیف‌گر توابع سیستمی سایه[5] (SSDT Shadow) به‌ترتیب در فایل‌های ntoskrnl.exe و user32.dll ذخیره شده‌اند که یکی از مناسب‌ترین نقاط برای قلاب‌سازی و به‌دست آوردن جریان کنترلی می‌باشند این دو جدول مشخص‌کننده آدرس توابع سیستمی سطح هسته در حافظه سیستم هستند بنابر این تمام درخواست‌های گذار از سطح کاربر برای رسیدن به سطح هسته نیاز به یافتن آدرس توابع مورد نظرشان در این جدول‌ها هستند [۱۴].

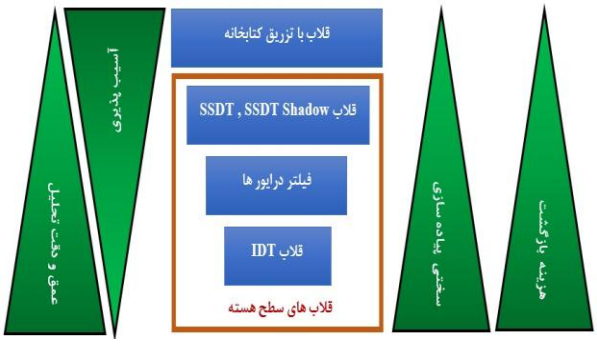
تفاوت دو جدول SSDT و SSDT Shadow در این است، جدول

2. Interrupt Descriptor Table
3. System Function Address Patching
4. System Service Descriptor Table (SSDT)
5. Shadow SSDT

SSDT نگاه دارنده آدرس توابع سیستمی برای کار با هسته سیستم عامل و جدول SSDT Shadow حاوی آدرس توابع سیستمی مرتبط با کارهای گرافیکی و پنجره‌های سیستم عامل ویندوز است [۱۵].

انجام تبدیل گذار سطح هسته به کاربر توسط دستور SysEnter برای سیستم عامل‌های ویندوز از XP به بعد و قبل از آن توسط وقفه شماره Int 2e H صورت می‌گیرد با قلاب‌اندازی به این جداول و جایگزین کردن آدرس توابع موجود در آن‌ها با آدرس توابع متناظر جعلی از پیش مهیا شده می‌توان کنترل را به‌دست گرفت و از بافرهای ورودی و پارامترهای ارسال شده به توابع مورد نظر آگاه شد سپس می‌توان مجدد تابع سیستمی اصلی را با پارامترهای درخواست شده یا تغییر داده شده فراخوانی نمود؛ تا در روند اجرایی برنامه‌ها خللی ایجاد نشود [۱۶].

در تصویر زیر روش‌های ذکر شده از نظر میزان آسیب‌پذیری در مقابل روش‌های ضد تحلیل، عمق و دقت تحلیل، سختی پیاده‌سازی و هزینه یافتن نام توابع سیستمی معادل که در ادامه توضیح داده شده با یک دیگر مقایسه شده‌اند.



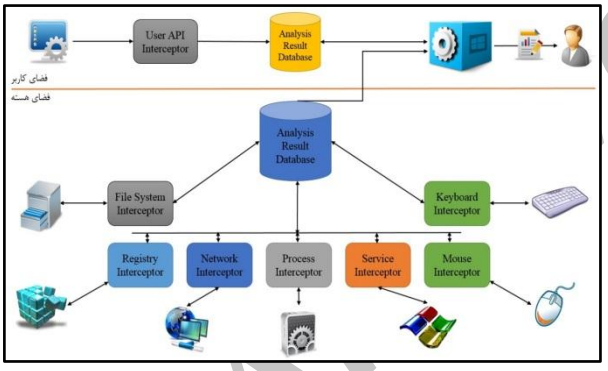
شکل ۲. مقایسه روش‌های قلاب‌سازی برای رهگیری رفتار

رهگیری در سطح کاربر با تزریق کتابخانه پیوند پویا کم ترین عمق رهگیری را دارد به همین دلیل در مقابل راهکارهای ضد تحلیل پویا بیشترین آسیب‌پذیری را دارد ولی در عین حال پیاده‌سازی آن از همه روش‌ها نیز ساده‌تر است هر چقدر به سخت‌افزار نزدیک شویم عمق رهگیری بیشتری خواهیم داشت ولی در عین حال پیاده‌سازی نیز سخت‌تر می‌شود. رهگیری وقفه‌ها در سطح هسته سیستم عامل بیشترین عمق رهگیری را دارد بنابراین کمترین آسیب‌پذیری در مقابل روش‌های ضد تحلیل را دارد ولی پیاده‌سازی آن بسیار سخت است با توجه به اینکه روش پیشنهادی تحلیل رفتاری را بر اساس نام توابع سیستمی یا WinAPI انجام می‌دهد [۱]. به غیر از روش رهگیری در سطح کاربر توسط تزریق کتابخانه در سایر روش‌های سطح هسته دارای هزینه یافتن تابع سیستمی یا WinAPI معادل، وجود دارد این هزینه با نزدیک شدن به سخت‌افزار بیشتر می‌شود در تمام موارد ذکر شده، رهگیری‌های توابع در سطح هسته بر پایه قلاب‌اندازی به جداول SSDT و SSDT Shadow و فیلتر درایورها حالت میانی دارند.

۳. معماری محیط تحلیل‌گر پیشنهادی

در این قسمت معماری محیط پیشنهادی که یک محیط جعبه شن سمت کاربر می‌باشد نشان داده شده در روش پیشنهادی درخواست‌های برنامه یا پردازه[1] تحت تحلیل برای استفاده از توابع سیستمی در سطح کاربر و هسته سیستم عامل رهگیری می‌گردد. هدف از این رهگیری نظارت بر عملکرد برنامه تحت تحلیل، محدودسازی و مجازی‌سازی امکانات حیاتی سیستم عامل و منحرف‌سازی یا فریب درخواست‌های برنامه تحت تحلیل با ثبت و تغییر در پارامترها و مقادیر بازگشتی توابع سیستمی است. موارد فوق در کنار خصوصیات دیگر برنامه شامل معماری فایل و اطلاعاتی از ساختار PE[2] برنامه مورد تحلیل از جمله تعداد، اندازه، نام و خواص قسمت‌ها[3] [۱۷] در قالب تعریف شده، خروجی لازم برای تعیین رفتار و کشف ماهیت بدخواهانه یا بی‌خطر بودن را فراهم می‌کنند.

در روش پیشنهادی رهگیری توابع سیستمی در سطح هسته با قلاب‌اندازی به امکانات سیستم عامل به خصوص جدول توصیف‌گر سرویس‌های سیستم عامل[4] و جدول توصیف‌گر سرویس‌های سیستم عامل سایه و رهگیری توابع سطح کاربر با تزریق کتابخانه‌های پیوند پویا به حافظه برنامه مورد نظر صورت می‌گیرد در شکل زیر معماری کلان محیط پیشنهادی آورده شده است.



شکل ۳. طرح معماری کلان محیط تحلیل‌گر پیشنهادی

همان‌طور که در شکل فوق نشان داده شده است ۷ درایور برای رهگیری، محدودسازی و استخراج رفتارهای مبتنی بر تعامل با فایل سیستم، رجیستری، شبکه، پردازه، سرویس‌های سیستم عامل، موشواره و صفحه کلید در سطح هسته و ۳ کتابخانه تزریق شونده در سطح کاربر برای رهگیری توابع سیستمی سطح کاربر که در سطح هسته قابل رهگیری نیستند یا رهگیری آن‌ها هزینه زیادی خصوصاً هزینه برگشت به عقب دارند طراحی و پیاده‌سازی شده است. نتایج حاصل از رهگیری درایورهای سطح هسته در یک مخرن و نتایج

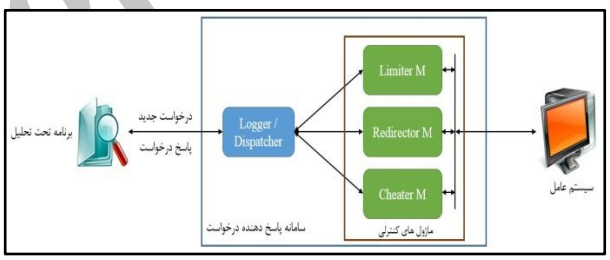
1. Process
2. Portable Executable
3. Sections
4. System Service Descriptor Table

حاصل از رهگیری توسط قلاب‌های تزریق شونده نیز در یک مخزن جداگانه در سطح کاربر ذخیره می‌شوند آنگاه محتویات این دو مخزن به‌صورت بلادرنگ توسط ماژول همگام‌ساز، بر اساس زمان فراخوانی همگام شده و آماده تهیه گزارش عملکرد، بصری‌سازی و نهایتاً تحویل به کاربر می‌شود. ارتباط بین درایورها و برنامه‌های سطح کاربر از طریق تعریف خط لوله نامدار و ایمن شده با مکانیسم احراز هویت مبتنی بر تعریف کلید تصادفی در مبداء و مقصد برقرار شده است. طرح پیشنهادی در سطح کاربر توسط زبان C++ Native و در سطح هسته توسط درایورهای با زبان C پیاده‌سازی شده است.

در طول مدت تحلیل جعبه شن مسئول پاسخ گویی به تمام درخواست‌های برنامه تحت تحلیل است. تصمیم‌گیری در خصوص نحوه پاسخ به این درخواست‌ها یکی از مهم ترین جنبه‌های طراحی و ساخت یک محیط تحلیل‌گر است که در ادامه توضیح داده می‌شود.

۳. ۱. سیاست پاسخ به درخواست‌ها

در طول زمان تحلیل پاسخ به درخواست‌های برنامه تحت تحلیل به عهده جعبه شن می‌باشد. روش پیشنهادی ۵ نوع سیاست را در پاسخ به درخواست‌های برنامه تحت تحلیل مد نظر دارد این سیاست‌ها در ادامه ذکر شده‌اند.



شکل ۴. سیاست‌های پاسخ به درخواست‌های برنامه تحت تحلیل توسط ماژول‌های کنترلی جعبه شن

- **فقط ثبت:** این وظیف بر عهده ماژول ثبت‌کننده یا Logger Module از سامانه رهگیری محیط تحلیل‌گر است.
- **محدودسازی:** این وظیفه بر عهده ماژول محدودساز یا Limiter Module از سامانه رهگیری محیط تحلیل‌گر است.
- **منحرف‌سازی:** این وظیفه بر عهده ماژول منحرف‌ساز یا Redirector Module از سامانه رهگیری محیط تحلیل‌گر است.
- **فریب دادن:** این وظیفه بر عهده ماجول فریب‌دهنده یا Cheater Module از سامانه رهگیری محیط تحلیل‌گر است.
- **مجازی‌سازی:** این مهم بر عهده ماژول شبیه‌ساز یا Emulator module از سامانه رهگیری محیط تحلیل‌گر است.

برای تصمیم‌گیری در خصوص ارائه پاسخ مناسب به درخواست‌های وارده لازم است نوع درخواست تشخیص داده شود که در ادامه نحوه دسته‌بندی درخواست‌ها برای انجام واکنش صحیح از جعبه شن تشریح می‌شود.

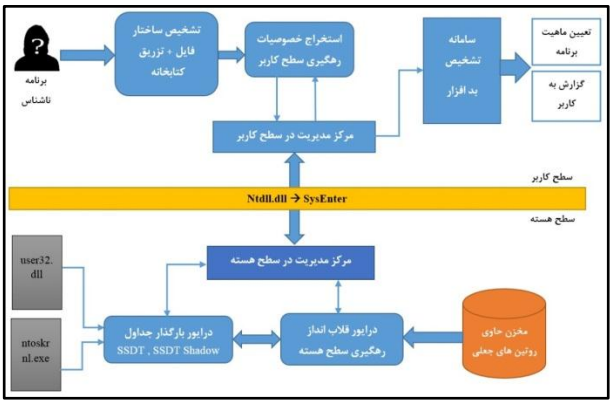
۳. ۲. دسته‌بندی خواست‌ها

با مکاشفه[1] انجام شده در تحلیل بیش از ۹۰۰۰ بدافزار و ۱۲۰۰۰ برنامه بی‌خطر از مراجع [۱۹،۱۸] امکان دسته‌بندی و تصمیم‌گیری در خصوص نحوه پاسخ به درخواست‌های یک برنامه ناشناس برای استخراج رفتار و مدل‌سازی آن به‌منظور کشف بدخواه بودن یا بی‌خطر بودن فراهم شده است این دسته‌بندی به شرح زیر می‌باشد.

۳. ۲. ۱. درخواست‌های مبتنی بر تعامل با فایل سیستم

تمامی درخواست‌های مبتنی بر کار با فایل‌ها در سیستم پیشنهادی شامل ایجاد، حذف و تغییر فایل به مسیری تحت نظارت برنامه که کاملاً محافظت شده می‌باشد منحرف می‌شود این مهم در سیستم پیشنهادی با قلاب‌اندازی به توابع مرتبط با فایل سیستم از جدول SSDT و تغییر در پارامترهای درخواست‌های وارده صورت می‌گیرد به‌عنوان مثال درخواستی با هدف ایجاد فایل در مسیر C:\Windows\file.exe به مسیر Sandbox\c\windows\file.exe تغییر جهت داده می‌شود تا بدین ترتیب از ایجاد فایل در محیطی به غیر از محیط تحت کنترل و نظارت جعبه شن جلوگیری به عمل آید. درخواست حذف فایل: برای این نوع درخواست مقداری جعلی به‌عنوان مقدار بازگشتی تابع سیستمی فراخوانی شده، در اینجا NtDeleteFile برگردانده می‌شود در این حالت بدافزار فریب خورده و گمان می‌کند که فایل مورد نظرش حذف شده در صورتی که در واقعیت چنین نیست. درخواست‌های مبتنی بر تغییر خواص یک فایل: پس از ثبت توسط جعبه شن با توجه به مقدار پارامترهای ورودی اجازه انجام داده می‌شود و یا مقدار جعلی برگردانده می‌شود ملاک این تصمیم گیری برای اجازه انجام عملیات، قرار داشتن فایل مورد نظر در مسیر تحت نظارت جعبه شن می‌باشد یعنی بدافزار می‌تواند فایل‌های خودش را تغییر خصیصه دهد ولی فایل‌های دیگر در سیستم با برگرداندن مقدار جعلی از سوی جعبه شن موجب فریب برنامه تحت تحلیل می‌شود. درخواست‌های جستجوی فایل پس از کنترل سابقه اجرایی برنامه در صورتی که فایل قبلاً توسط برنامه تحت تحلیل حذف شده باشد با برگرداندن مقدار جعلی عدم وجود فایل را به برنامه تحت تحلیل نشان داده می‌شود زیرا برنامه تحت تحلیل می‌خواهد از صحت عملکرد حذف فایل اطمینان پیدا کند در غیر این صورت به استثناء فایل‌های متعلق به برنامه تحت تحلیل برای سایر فایل‌های سیاست مجازی‌سازی با تحویل یک رونوشت از فایل مورد درخواست در نظر گرفته می‌شود برنامه تحت تحلیل می‌تواند عملیات حذف، تغییر و تکثیر را در نسخه انجام دهد هدف از این کار پی بردن به اقدامات مخرب بدافزارها خصوصاً خانواده بدافزارهای چسبنده، بروی فایل‌های دیگر است.

1. Heuristic



شکل ۵. طرح معماری جعبه شن پیشنهادی برای رهگیری رفتار در سطح کاربر و هسته

همان‌گونه که در شکل فوق مشخص است محیط پیشنهادی برای رهگیری رفتار به‌صورت ترکیبی در سطح کاربر و سطح هسته عمل نموده، در سطح هسته یک درایور مسئول بارگذاری جداول SSDT و SSDT Shadow از فایل‌های مالک یعنی User32.dll و ntoskrnl.exe است پس از بارگذاری جداول فوق روتین‌های جعلی از مخزنی در سطح هسته توسط درایور قلاب انداز جایگزین روتین‌های اصلی سیستمی عامل می‌شوند این جایگزینی با تغییر آدرس تابع سیستمی با آدرس تابع جعلی متناظر برای رهگیری رفتار در سطح هسته صورت می‌پذیرد در سطح کاربر یک فایل پیش از اجرا برای تحلیل پویا برای استخراج خواص رفتار ایستا بر اساس ساختار PE مورد بررسی قرار گرفته و کتابخانه حاوی روتین‌های جعلی به حافظه تخصیص یافته به برنامه تحت تحلیل برای رهگیری در سطح کاربر تزریق می‌شود در پایان نتایج حاصل از تحلیل رفتار به‌صورت ایستا و پویا برای تحویل به کاربر یا برنامه تشخیص بدافزار آماده می‌شود. هم گام‌سازی نتایج تحلیل توسط مراکز مدیریت در سطح کاربر و هسته با ثبت زمان مهر[1] صورت می‌پذیرد.

۳. ۲. ۲. درخواسته‌های مبتنی بر تعامل با رجیستری

تمام درخواست‌های تعامل با رجیستری از قبیل ایجاد، حذف و تغییر مقدار کلیدهای رجیستری نیز مانند درخواست‌های تعامل با فایل سیستم پاسخ داده می‌شوند یعنی درخواست‌های ایجاد به مسیری کنترل شده و تحت نظارت منحرف شده، درخواست‌های تغییر مقدار و حذف کلید پس از مشخص شدن مقصد آن، برای کلیدهای متعلق به برنامه تحت تحلیل بلامانع و برای کلیدهای دیگر سیاست مجازی‌سازی در پیش گرفته می‌شود.

۳. ۲. ۳. درخواست‌های مبتنی بر فرآیندها و نخ[2]ها

درخواست‌های خواندن و نوشتن از / در حافظه فرآیندهای دیگر، توسط جعبه شن ثبت شده و با توجه به فرآیند هدف که از پارامترهای درون توابع سیستمی فراخوانی شده قابل استخراج است پاسخ داده می‌شود توابع مربوطه شامل WriteProcessMemory، ReadProcessMemory و SetThreadContext می‌باشند در صورتی که پردازه مقصد، پردازه محیط تحلیل‌گر یا هر پردازه تحت محافظت محیط تحلیل‌گر باشد سیاست محدودسازی و در صورتی که پردازه مقصد، پردازه دیگری باشد سیاست فقط ثبت برای پاسخ گویی اعمال خواهد شد. رهگیری دقیق این دسته رفتار بسیار مهم است زیرا درخواست‌های خواندن و نوشتن در حافظه پردازه‌های دیگر اغلب حکایت از وجود بدافزارهای چسبنده[3] دارد[۲۰].

۳. ۲. ۴. درخواست‌های استفاده از امکانات اشکال‌زدایی

برای این درخواست‌ها جعبه شن پیشنهادی سیاست محدودسازی با ثبت مقداری جعلی به‌عنوان مقدار بازگشتی توابع مربوطه در پیش می‌گیرد علت این امر مخفی کردن وجود هرگونه برنامه تحلیل‌گر یا اشکال‌زدا در سیستم عامل از دید برنامه تحت تحلیل است بسیاری از بدافزارها از تابع سیستمی فوق برای تشخیص فعال بودن یک برنامه اشکال‌زدا در سیستم عامل به‌منظور جلوگیری از اجرای خود استفاده می‌کنند علت این امر ممانعت از افشای عملکرد مخرب بدافزار با اجرا در محیط اشکال‌زدا است که توسط شرکت‌های ضد ویروس برای تحلیل بدافزارها به‌صورت گسترده استفاده می‌شود رهگیری این دسته از درخواست‌ها با تزریق کتابخانه در سطح کاربر انجام می‌شود.

۳. ۲. ۵. درخواست‌های مبتنی بر کار با سرویس‌ها

تمام درخواست‌های کار با سرویس‌ها شامل ایجاد، کنترل و حذف سرویس برای برنامه مورد تحلیل پس از ثبت توسط جعبه شن محدود می‌شوند زیرا این درخواست امکان بارگذاری درایور برای برنامه‌ها را فراهم می‌کنند که در صورت وقوع این اتفاق امکان کنترل کردن برنامه به‌دلیل بارگذاری درایور و ورود برنامه تحت تحلیل به فضای هسته سیستم عامل توسط جعبه شن کم شده و امکان از دست رفتن و شکست جعبه شن بوجود می‌آید لذا برای جلوگیری از ورود خسارت به سیستم عامل تمام این درخواست‌ها برای جعبه شن سمت کاربر محدود می‌شود اگر چه عمق تحلیل را کاهش می‌دهد این دسته از درخواست‌ها در سطح هسته توسط درایور قلاب‌انداز رهگیری می‌شوند.

۳. ۲. ۶. درخواست‌های مبتنی بر کار با پنجره‌ها

تمامی توابع مربوط به کار با پنجره‌های سیستم عامل ویندوز شامل ایجاد، بستن و تغییر مکان پنجره با سیاست فقط ثبت توسط جعبه شن پاسخ داده می‌شود درخواست‌های استفاده از توابع سیستمی FindWindow و GetWindow که برای جستجوی وجود پنجره فعالی با نام مشخص استفاده می‌شوند توسط جعبه شن

1. Time Stamp
2. Thread
3. Binder

رهگیری و در برخی موارد سیاست فریب با برگرداندن مقادیر جعلی ملاک قرار می‌گیرد علت این است که بسیاری از بدافزارها برای جلوگیری از تحلیل شدنشان توسط برنامه‌های تحلیل‌گر با استفاده از توابع سیستمی مذکور اقدام به جستجوی وجود پنجره فعالی با نام برنامه‌های تحلیل‌گر خصوصاً برنامه‌های تحلیل‌گر معروف می‌کنند برای حل این مشکل جعبه شن پیشنهادی از روش نام‌گذاری تصادفی برای پنجره رابط کاربری خود استفاده می‌کند [۲۱].

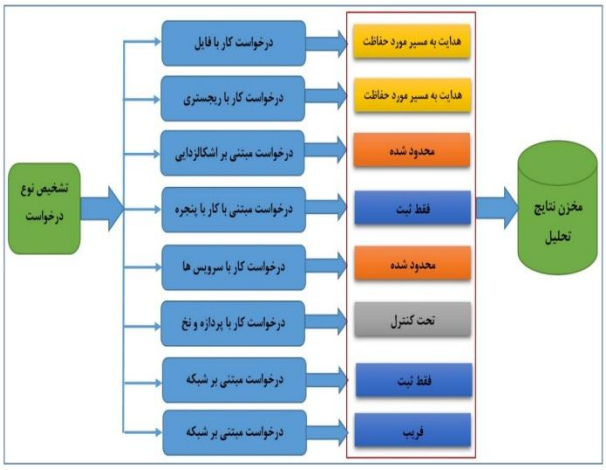
۳. ۲. ۷. درخواست‌های مبتنی بر کار با زمان سیستم

توابع مربوط به خواندن زمان‌ها از سیستم عامل مانند زمان ایجاد یک فایل توسط جعبه شن با سیاست فقط ثبت پاسخ داده می‌شوند ولی برای توابع مربوط به تغییر این مقادیر سیاست فریب با بازگردانی مقدار جعلی به برنامه هدف در نظر گرفته می‌شود هدف از این کار فریب برنامه تحت تحلیل برای نشان دادن موفق بودن عملیات مربوطه (به‌عنوان مثال تغییر زمان ایجاد فایل) است علت این امر در آن است که دسته از بدافزارها تحت عنوان بدافزارهای زمان محدود که در زمان‌های خاصی رفتار بدخواهانه خود را آشکار می‌کنند را بتوان تحلیل نمود جعبه شن پیشنهادی با قلاب‌اندازی به توابع سیستمی مربوط به کار با زمان و تاریخ سیستم شامل GetSystemTime و GetLocalTime اقدام به تحویل زمان‌های جعلی به برنامه تحت تحلیل نموده تا بتوان رفتار آن را در زمان‌های آینده برای تأیید بی‌خطر بودن بررسی نمود.

۳. ۲. ۸. درخواست‌های تعامل با شبکه

بسیاری از بدافزارها خصوصاً جاسوس افزارها که قصد سرقت اطلاعات را دارند برای ارسال اطلاعات نیاز دارند تا یک سوکت به مقصدشان ایجاد کنند این سوکت از ترکیب آدرس IP و Port و نوع پروتکل مورد نظر به‌دست می‌آید. توابع سیستمی Socket, WSASocket امکان تعریف یک سوکت را فراهم می‌کند که بهترین محل برای رهگیری جهت کشف مقصد ارسالی می‌باشد با استفاده از توابع سیستمی Connect, WSAConnect می‌توان به سوکت ایجاد شده متصل شد و با توابع سیستمی Send ، SendTo و WSASend به سوکت‌های منعقد شده داده ارسال نمود. جعبه شن پیشنهادی تمام درخواست‌های مرتبط با شبکه برای پردازه‌های مختلف را با تزریق کتابخانه پیوند پویا رهگیری و ثبت می‌کند اطلاعات ثبت شده شامل شماره شناسه و نام پردازه، نوع بسته ارسالی، آدرس IP و پورت مقصد و زمان ارسال است علت ثبت تمام اطلاعات کار با شبکه در این است که بعد از کشف یک جاسوس افزار بتوان تعاملاتی که در گذشته با شبکه وجود داشته، مقصدهای مورد نظر و اطلاعات سرقت شده را شناسایی نمود جعبه شن می‌تواند مقصد مورد نظر جاسوس افزار را با دقت کشور، شهر و فراهم‌کننده خدمات اینترنت[1] رهگیری کند این رهگیری به‌صورت ترکیبی با تزریق کتابخانه در سطح کاربر و قلاب‌اندازی به توابع سیستمی در سطح هسته صورت می‌پذیرد [۲۱].

با توجه به آنچه تا کنون گفته شد نحوه پاسخ‌گویی مناسب به درخواست‌های برنامه تحت تحلیل با دسته‌بندی درخواست‌های وارده و استفاده از سیاست‌های تنظیم شده محرض و تأثیر پاسخ صحیح در کشف بدافزارها و حفظ کنترل جعبه شن نمایان گردید در شکل زیر نمایی کلی از نحوه پاسخ‌گویی برنامه جعبه شن به درخواست‌ها در دسته‌های مختلف آورده شده است.



شکل ۶. دسته‌بندی درخواست‌های برنامه تحت تحلیل و سیاست مناسب برای پاسخ به آنها

۳. ۳. سیاست کشف بدافزار و خوشه‌بندی آن‌ها

بر خلاف اغلب محیط‌های تحلیل‌گر که برای خوشه‌بندی بدافزارهای کشف شده از واسط‌های کاربری آماده به خصوص واسط Yara استفاده می‌کنند روش پیشنهادی از الگریتم‌های اختصاصی ارائه شده در مرجع [۱۷] برای کشف و خوشه‌بندی فایل‌های بی‌خطر و بدافزارها سود می‌برد.

۴. چالش‌ها و آسیب‌پذیری‌های محیط‌های تحلیل‌گر

در قسمت‌های قبل این مقاله ضمن معرفی محیط‌های تحلیل‌گر جهت اجرای ایمن برنامه‌های مشکوک و ناشناخته به‌منظور تحلیل و استخراج مدل رفتاری آن‌ها، نحوه ایجاد یک محیط امن مبتنی بر جعبه شن در سمت کاربر مطرح شد در این قسمت به بیان آسیب شناسی و بررسی روش‌های تشخیص و گریز از محیط‌های تحلیل‌گر پرداخته می‌شود این آسیب‌شناسی یکی از نکات بسیار مهم در طراحی و پیاده‌سازی یک محیط تحلیل‌گر می‌باشد زیرا برنامه‌های حساس امنیتی و بدافزارها خصوصاً بدافزارهای نوین تا حد امکان قبل از اجرا و نشان دادن رفتار خود سعی در شناسایی این‌گونه محیط‌ها دارند تا در صورت کشف وجود آن‌ها از اجرای خود برای حفظ محرمانگی مکانیسم عملکردشان جلوگیری کنند همچنین برخی از بدافزارها به روش‌ها و امکاناتی برای دور زدن و خروج از محیط‌های تحلیل‌گر و آسیب زدن به سیستم عامل میزبان مجهز هستند لذا

1. Internet Service Provider

جعبه شن باید برای مواجهه با این گونه بدافزارها مکانیسم‌های ایمنی و دفاع از خود مناسبی داشته باشد در ادامه عمده چالش‌های موجود بیان و تا حد امکان راهکارهایی برای مقابله با آن‌ها بیان می‌شود.

۴. ۱. کنترل سرعت اجرای برنامه

در این روش بدافزارها بعد از شناسایی نوع پیکربندی سیستم شامل نوع، تعداد و فرکانس پردازنده، میزان حافظه آزاد و دیگر پارامترهای تأثیرگذار اقدام به اجرای Benchmark برای سنجش میزان سرعت سیستم بر اساس تعداد کلاک‌های و زمان صرف شده برای اجرا می‌کنند (توجه شود که این کدها کاملاً بی‌خطر بوده و برای آزمایش اجرا می‌شوند) سپس مقادیر ارزیابی شده با مقادیری که از قبل با اجرا بروی یک سیستم واقعی مورد اطمینان در پایگاه داده خود میزان سنخیت زمان اجرای به‌دست آمده با پیکربندی سیستم را ارزیابی می‌کند [۲۲] علت آن است که در ماشین‌های مجازی امکانات سیستم عامل مجازی‌سازی شده و از منابع سیستمی به‌صورت اشتراکی استفاده می‌شود لذا زمان اجرای برنامه‌ها قدری بیشتر شده و سیستم نسبت ماشین واقعی کندتر عمل می‌کند این مشکل بیشتر برای محیط های تحلیل‌گر مبتی بر ماشین های مجازی مطرح است راه حل پیشنهادی برای این چالش، نشان دادن اطلاعات غلط از پیکربندی سیستم به برنامه درخواست‌کننده و در عین حال سبک نگه‌داشتن ماشین‌های میزبان از هرگونه برنامه و پردازه غیر ضروری برای جلوگیری اتلاف منابع سیستمی به‌ویژه وقت پردازنده است.

تابع سیستمی GetSystemInfo امکان مشاهده اطلاعات سیستم را در قالب یک ساختار SystemInfo فراهم می‌کند در این ساختار پارامترهای NumberOfProcessors و ProcessorType نشان‌دهنده تعداد و نوع پردازنده سیستم است [۲۳] بدافزارها با استفاده از تابع سیستمی مذکور سعی در تخمین تعداد کلاک‌ها و زمان لازم برای اجرای یک تکه کد بی‌خطر به‌عنوان آزمون بر اساس آن پیکربندی در حالت عادی را دارند، روش پیشنهادی با قلاب‌اندازی به تابع مذکور درخواست های وارد شده به آن با سیاست فریب و تحویل اطلاعات غلط از پیکره سیستم جهت فعال‌سازی بدافزار داده می‌شود.

۴. ۲. بررسی کلیدهای رجیستری[1]

کلیدهای رجیستری محل خوبی برای شناسایی محیط ماشین مجازی و جعبه شن‌ها است اکثر ماشین‌های مجازی اقدام به ایجاد کلیدهایی در رجیستری سیستم برای انجام فعالیت‌ها، ارتباطات و پیکربندی خود می‌کنند لذا با تحلیل و جستجوی کلیدهایی خاص در رجیستری سیستم عامل امکان شناسایی این دسته از ماشین‌های مجازی وجود دارد جعبه شن پیشنهادی برای حل کردن این موضوع درخواست‌های برنامه تحت تحلیل برای جستجوی کلیدهای رجیستری توسط توابع سیستمی نظیر RegOpenKey, NtCreateKey, NtSetValueKey را بررسی نموده تا در صورتی که بخواهند برای جستجوی وجود کلیدهای متلق به جعبه شن استفاده شوند با فریب برنامه تحت تحلیل وجود این کلیدهای را از دید آن‌ها پنهان کند.

۴. ۳. بررسی پردازه‌های در حال اجرا

یکی از ساده‌ترین روش‌های شناسایی محیط ماشین مجازی و جعبه شن‌ها جستجوی در لیست پردازه‌های فعال و پس زمینه سیستم عامل با هدف پیدا کردن پردازه‌ای در رابطه با آن‌ها است به‌عنوان مثال همواره پردازه‌ای تحت عنوان VM-Ware Tray در ماشین میهمان ایجاد شده توسط برنامه مجازی ساز VM-Ware وجود دارد که می‌تواند نمایانگر وجود این محیط ماشین مجازی باشد، طرح پیشنهادی برای جلوگیری از کشف وجود جعبه شن توسط بدافزار، با شنود درخواست‌ها برای دریافت لیست پردازه‌های سیستم عامل و یا جستجو میان آن‌ها توسط توابع سیستمی Process32First و Process32Next وجود پردازه مربوط به جعبه شن را از دید برنامه تحت تحلیل مخفی می‌کند این کار با تغییر پارامترهای بازگشتی از توابع فوق صورت می‌گیرد.

۴. ۴. بررسی درگاه‌های ورودی/خروجی

سیستم عامل نصب‌شونده تحت ماشین مجازی (سیستم عامل میهمان) نیاز به برقراری ارتباط و تعاملاتی با ماشین میزبانش دارد این ارتباط‌ها معمولاً از طریق ایجاد سوکت[2] و خط لوله‌های نام دار انجام می‌شود لذا می‌توان با بررسی پورت‌های وروی و خروجی سیستم پورت‌های خاص که توسط برنامه مجازی ساز برای تعامل ماشین‌های میزبان و میهمان ایجاد شده‌اند را کشف و وجود محیط ماشین مجازی را تأیید نمود این مشکل بیشتر مربوط به ماشین‌های مجازی است و برای جعبه شن‌ها در سمت کاربر مطرح نیست.

۴. ۵. بررسی لیست برنامه‌ها و سرویس‌های سیستم

هدف از این روش جستجو در لیست برنامه‌های نصب شده در سیستم عامل و پوشه‌های درایو سیستمی در جهت کشف نام و نشانی از ماشین مجازی یا جعبه شن است تا بتوان ردپایی از وجود برنامه‌های فوق را به‌دست آورد طرح پیشنهادی با این روش همانند دو روش جستجو در پردازه‌ها و جستجو در کلیدهای رجیستری برخورد نموده و با کنترل درخواست‌های وارده از سوی برنامه تحت تحلیل برای جستجو فایل و سرویس توسط توابع مربوطه یعنی FindFirstFile و FindNextFile [۱۴] با اتخاذ سیاست فریب وجود فایل‌های خود را از دید برنامه تحت تحلیل مخفی می‌سازد.

۴. ۶. درخواست سرویس‌های سطح بالا و محدودشده

در محیط‌های جعبه شن خصوصاً جعبه شن‌های سمت کاربر به

1. Registry Key

2. Socket

دلایل امنیتی ذکر شده مجبور به محدودسازی برخی از سرویس‌ها و امکانات حیاتی سیستم عامل جهت حفظ کنترل بر برنامه تحت تحلیل مانند جلوگیری از بارگذاری درایور و یا ثبت سرویس هستیم و برخی بدافزارها با علم به این موضوع قبل از اجرا سرویس‌های سطح بالایی از سیستم مانند ثبت یک بوت– استارت درایور را درخواست می‌کنند که در صورت عدم دریافت پاسخ به اجرا تحت محیط محدودکننده مشکوک می‌شوند همچنین راهکارهای مبتنی بر بررسی حلقه‌های دسترسی[1] سیستم عامل در مرجع [۲۲] برای تشخیص محیط ماشین مجازی پیشنهاد شده است.

۴. ۷. امکانات اشکال‌زدایی سیستم عامل

در این روش برنامه بدخواه با استفاده از توابع مربوط به اشکال‌زدایی سیستم عامل سعی در شناسایی محیط اشکال‌زدا دارند به‌عنوان مثال می‌توان با فراخوانی تابع IsDebuggerPresent از کتابخانه Kernel32.dll در صورت بازگشت مقدار True از فعال بودن یک برنامه اشکال‌زدا در سیستم عامل مطمئن [۱۵] و به اجرا در تحت یک اشکال‌زدا مشکوک شد طرح پیشنهادی با کنترل تمام درخواست‌های واصله به توابع سیستمی مرتبط با اشکال‌زدایی از طریق تزریق کتابخانه [۲۴] و پاسخ با سیاست فریب وجود محیط تحلیل‌گر را از دید برنامه تحت تحلیل مخفی می‌کند.

۴. ۸. قرص‌های قرمز[2] و محافظت کننده‌ها

قرص‌های قرمز برنامه‌های آماده‌ای هستند که با شناختی که از محیط‌های ماشین‌های مجازی و جعبه شن‌های موجود دارند امکان شناسایی آن‌ها را فراهم می‌کنند. قرص‌های قرمز اغلب با استفاده از یک یا چند آسیب‌پذیری ذکر شده شناخت لازم از محیط‌های تحلیل‌گر را به‌دست می‌آورند به‌عنوان مثال ابزار بسته‌بندی[3] و محافظ‌کننده[4] Themida اجازه اجرا به برنامه‌های تحت حافظت خود را در محیط ماشین مجازی VM-Ware نمی‌دهد برخی از قرص‌های قرمز امکان شکست و خروج از محیط‌های جعبه شن که کنترل دقیق و صحیح بر برنامه تحت تحلیل ندارند را فراهم می‌کنند این مورد برای جعبه شن‌های سمت کاربر بیشتر نمایان است ولی در خصوص ماشین‌های مجازی تحلیل‌گر این آسیب پذیری وجود ندارد.

۵. استخراج رفتار و کشف بدخواه بودن آن

در بخش‌های پیشین نحوه طراحی و ایجاد یک محیط امن توضیح داده شد و با آسیب‌شناسی انجام شده روش‌های شناسایی و گریز از محیط‌های امن نیز ذکر شد و راه کارهایی برای مقابله با آن‌ها ارائه گردید در این قسمت نحوه استخراج اطلاعات اجرایی و تحلیل آن‌ها توضیح داده می‌شود. طرح پیشنهادی پس از اتمام اجرای پردازه یا مدت زمانی خاص که کاربر تعیین می‌کند تمام وقایع سیستمی که توسط جعبه شن ثبت شده شامل نام توابع فراخوانی شده، مقدار پارامترها و مقادیر بازگشتی از توابع به همراه نوع واکنش جعبه شن که براساس زمان وقوع با ثبت زمان مهر مرتب شده‌اند و دیگر خصوصیات برنامه تحت تحلیل شامل نوع معماری، امضاء از قسمت‌های مختلف فایل و اطلاعات مفید دیگر از ساختار PE شامل تعداد، نام، اندازه و خواص قسمت‌ها در قالب XML و PDF در اختیار کاربر یا برنامه تشخیص بدافزار قرار می‌گید علت انتخاب قالب XML برای درختی بودن آن برای نشان دادن تعاملات بین یک پردازه با پردازه‌های فرزند و نخهایش می‌باشد [۴].

۶. ارزیابی و مقایسه

در این قسمت جعبه شن پیشنهادی را با چند جعبه شن مطرح در دنیا از حیث توانایی‌ها و قابلیت‌های رهگیری رفتار و میزان مصرف منابع سیستمی مقایسه می‌کنیم.

۶. ۱. ارزیابی و مقایسه از جهت توانایی‌ها و قابلیت‌های رهگیری رفتار

عمق و توانایی رهگیری رفتار تأثیر مستقیم و بسزایی در دقت و سهولت روش‌های تحلیل پویا دارد و از این حیث بسیار حایز اهمیت است که بستر رهگیری بتواند فرآیند استخراج رفتار را به‌صورت صحیح و کامل انجام دهد به‌عنوان مثال رهگیر تمام رهگیرهایی که فرآیند رهگیری را فقط در سطح کاربر انجام می‌دهند از تحلیل و تشخیص بدافزارهای سطح هسته از جمله اکثر روت کیت‌ها ناتوان خواهند بود [۲۵] در ادامه توانایی‌های رهگیری جعبه شن پیشنهادی ارزیابی و با نمونه‌های خارجی مقایسه شده است.

همان‌طور که در جدول بالا مشخص است جعبه شن پیشنهادی نسبت به سایر جعبه شن‌ها تعداد بهینه‌ای از توابع را رهگیری می‌کند رهگیری زیاد توابع موجب لختی و افت سرعت سیستم عامل خواهد شد همچنین در جعبه شن پیشنهادی برخی از توابع در سطح هسته و برخی دیگر در سطح کاربر رهگیری شده‌اند یکی از دلایل این امر کاهش حجم کتابخانه تزریق شونده در فرآیندهای هدف است بسیاری از جعبه شن‌های ذکر شده نتوانستند در سطح هسته اقدام به رهگیری نمایند که علت آن هم پیچیدگی بسیار زیاد پیاده‌سازی و برنامه نویسی هسته است ولی جعبه شن پیشنهادی این مهم را انجام داده است مضاف بر آنکه برخی از جعبه شن‌های پیشنهادی برای رهگیری در سطح کاربر نیز از ابزارهای تزریق‌کننده آماده برای تزریق کتابخانه پیوند پویا استفاده کرده‌اند که این خود عاملی برای وابستگی به خارج است.

1. Privilege Rings
2. Red Pill
3. Packer
4. Protector

جدول ۱. مقایسه جعبه شن پیشنهادی با نمونه‌های مطرح خارجی از حیث توانایی و قابلیت‌های رهگیری رفتار

محیط امن	سکوی اجرا		دسترسی به متن برنامه	زبان برنامه نویسی	تعداد توابع رهگیری شده سطح کاربر	تعداد توابع رهگیری شده سطح هسته	قلاب انداز مجزا
جعبه شن PyBox	لینوکس	ویندوز	متن باز	Python	۲۵	۰	بلی
جعبه شن Comodo	ویندوز		متن بسته	?	۳۰	۲۴	بلی
جعبه شن Avast	ویندوز		متن بسته	?	۵	۲۰	بلی
جعبه شن Quemu	لینوکس	ویندوز	متن بسته	?	۴	۰	خیر
جعبه شن SandboxIE	ویندوز		متن بسته	?	۹۸	۰	بلی
جعبه شن پیشنهادی	ویندوز		متن بسته	C/C++	۳۰	۲۱	بلی

۶. ۲. ارزیابی و مقایسه از جهت میزان مصرف منابع سیستمی

یکی از نقاط ضعف تمام ابزارهای تحلیل‌گر در ایجاد لختی و افت سرعت سیستم عامل میزبان به‌دلیل نصب مکانیسم‌های رهگیری از جمله قلاب‌های تزریق شونده در سطح کاربر و درایورهای قلاب‌انداز در سطح هسته می‌باشد. همواره بین عمق رهگیری و ایجاد لختی و افت سرعت سیستم عامل رابطه معکوس برقرار است و از این حیث جعبه شن پیشنهادی فرآیند رهگیری را به‌صورت بهینه انجام می‌دهد تا در عین رهگیری دقیق با عمق بالا کمترین میزان لختی و مصرف منابع سیستمی را داشته باشد میزان مصرف منابع سیستمی برای جعبه شن پیشنهادی و چند نمونه خارجی تحت شرایط یکسان ارزیابی و مقایسه شده است.

میزان مصرف منابع سیستمی شامل پردازنده و حافظه سیستم با معدل گیری از تحلیل ۶ نمونه بدافزار و ۴ نمونه فایل بی‌خطر با حجم‌های مختف توسط جعبه شن پیشنهادی و نمونه‌های خارجی اندازه‌گیری شده که نتایج آن به شرح زیر است.

شکل ۷. مقایسه میانگین درصد مصرف منابع سیستمی شامل پردازنده و حافظه

همان‌گونه که در جدول فوق مشخص است میزان مصرف منابع سیستمی شامل پردازنده و حافظه در محیط پیشنهادی نسبت به اکثر محیط‌های دیگر که همگی در سمت کاربر ارزیابی شده‌اند کمتر است که عامل اصلی آن هم هوشمندی جعبه شن و توانایی تغییر پیکربندی خود برای حذف قلاب‌های اضافی و کاهش حجم کتابخانه تزریق شوند است این در حالی است که جعبه شن پیشنهادی در سطح هسته نیز مبادرت به رهگیری توابع سیستمی می‌کند که خود عاملی برای مصرف مضاعف منابع سیستمی است در حالی که اغلب جعبه شن‌های دیگر فقط در سطح کاربر عمل رهگیری را انجام می‌دهند همچنین کمینه بودن میزان مصرف منابع سیستمی جعبه شن Avest به‌دلیل کم بودن عمق رهگیری آن است که با کاهش تعداد قلاب‌های رهگیری توانسته مصرف منابع سیستمی را کاهش دهد ولی در عین حال عمق رهگیری آن نیز کاهش یافته است.

۷. نتیجه‌گیری

این مقاله ضمن معرفی و مقایسه محیط‌های تحلیل‌گر یک محیط تحلیل‌گر مبتنی بر جعبه شن در سمت کاربر نیز پیشنهاد شد جعبه شن پیشنهادی توانایی رهگیری و استخراج رفتار مخرب‌ها را در سطح کاربر و سطح هته سیستم عامل دارد این امکان در سطح کاربر با تزریق کتابخانه پیوند پویا به حافظه برنامه هدف و درایورهای قلاب انداز به جداول توصیف‌گر سرویس‌های سیستمی در سطح هسته فراهم شده است. جعبه شن پیشنهادی این مقاله درخواست‌های برنامه تحت تحلیل را در ۸ خانواده دسته‌بندی و با ۵ سیاست پاسخ می‌دهد. جعبه شن پیشنهادی با استفاده از سیاست‌های ذکر شده برخی امکانات حیاتی سیستم عامل را به‌منظور حفظ کنترل بر برنامه تحت تحلیل محدود می‌نماید. این مقاله چالش‌های موجود بر محیط‌های تحلیل‌گر را آسیب شناسی و با ارائه راهکارهای کارا برای آن‌ها ایمنی سیستم عامل میزبان را در طی فرآیند تحلیل تضمین می‌نماید در پایان جعبه شن پیشنهادی را از حیث توانایی و قابلیت‌های رهگیری و میزان مصرف منابع سیستمی ارزیابی و با نمونه‌های مطرح خارجی مقایسه نمود. ابزار پیاده‌سازی شده از روش پیشنهادی این مقاله می‌تواند برای تأیید ایمنی یک نرم افزار و یا تحلیل بدافزار در آزمایشگاه‌ها وهمچنین تعبیه در برنامه‌های ضد ویروس استفاده شود.

۸. تشکر و قدردانی

از همکاران عزیزمان در طرح ضد ویروس بومی، آقایان امیر گوران اوریمی و امیر محمدزاده لاجوردی که در پیاده‌سازی روش پیشنهادی، نویسندگان این مقاله را یاری نمودند تشکر و قدردانی می‌شود.

۹. مراجع

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

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

[3] The Need for Speed: Incident Response Survey, FireEye. http://www.inforisktoday.in/surveys/2013-incident-response-survey-s-18, 2013.

[4] Mohammadzadeh Lajevardi, A. "Design and Implementation of a Behavior-Based Method for Malware Detection."; M.Sc. Thesis, Iran University of Science and Technology, Tehran, 2013. (In Persian)

[5] "Applications (Confining the Wily Hacker)."; In Proc. of the 6th USENIX UNIX Security Symposium, 2011.

[6] Hoglund, G.; Butler, J. "Rootkits: Subverting the windows kernel."; 1st, 2005.

[7] Silberschatz, A.; Galvin, P.B.; Gagne, G. "Operating System Concepts."; 9th, 2012.

[8] Sanabria, A. "Malware Analysis: Environment Design and Architecture."; SANS Institute InfoSec Reading Room, 2007.

[9] "Cuckoo Sandbox Book."; http://docs.cuckoosandbox.org/en/latest/, 2013.

[10] Gooran Ourimi, A. "Design and Implementation a File Analyzer Based on Virtual Machine Hypervisor."; M.Sc. Thesis, Iran University of Science and Technology, Tehran, 2014. (In Persian)

[11] Schönbein, C. "PyBox - A Python Sandbox"; Diploma Thesis, May 2011.

[12] Engelberth, M.; Göbel, J.; Schönbein, C.; Freiling, C. "PyBox A Python Sandbox."; In Proc. of Make Available to a Broad Public Recent Findings in Informatics of Computer Science and Information Systems, pp. 137-138, 2011.

[13] Plohmann, D.; Leder, F. "GI Graduate Workshop on Reactive Security for PyBox."; University of Bonn, Germany, 2010.

[14] Russinovich, M.; Solomon, D.; Ionescu, A. "Windows Internals Part1."; 6th, 2012.

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

[16] Parsa, S.; Mohammadzadeh Lajevardi, A.; Amiri, M. J. "Propose a Method for Attack to Malware Detector Tools with Hiding System Calls."; In Proc. of 18th Iran Computer Conference, Sharif University of Technology, 2013. (In Persian)

[17] Javaheri D.; Parsa S. "Protection of Operation System against Spywares."; Advanced Defense Science and Technology, vol. 5, no. 2, pp. 171-181, 2014. (In Persian)

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

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

[20] Salmani Balu, A.; Lazemi, S.; Parsa, S. "Disinfect Infector Viruses with PE Header."; In Proc. of 2nd Sofware Security, Shiraz University, Shiraz, pp. 97-102, 2014. (In Persian)

[21] Javaheri D.; Parsa S. "A Malware Detection Method Based on Static Analysis of PE Structure"; Passive Defense Science and Technology, 2014, vol. 5. (In Persian)

[22] Sharifi, M.; Salimi, H.; Saberi, A.; Gharibshah, J. "VMM Detection Using Privilege Rings and Benchmark Execution Times."; Int. J. Communication Networks and Distributed Systems, 2014.

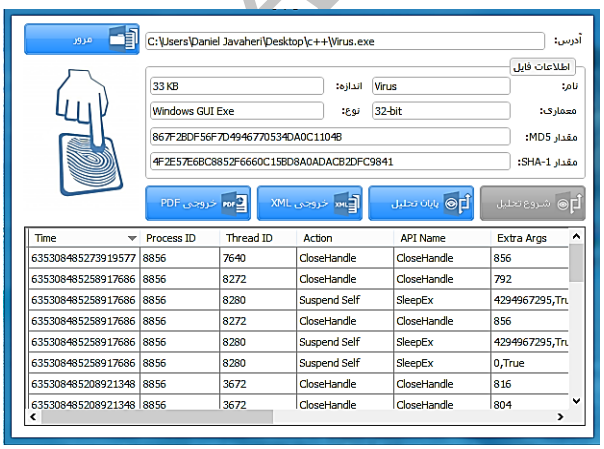
[23] Petzold, C., "Programming Windows."; 6th, 2013.

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

[25] Martin Arnold, T.; "A Comparative Analysis of Rootkit Detection Techniques."; M.S Thesis, The University of Houston Clear Lake, May 2011.

ضمیمه ۱

نرم افزار جعبه شن پیاده‌سازی شده بر اساس روش پیشنهادی این مقاله امکان استخراج و مدل کردن رفتار برنامه‌ها را با رهگیری بیش از ۵۰ فراخوانی سیستمی به همراه پارامترها و مقادیر بازگشتی آن‌ها در سطح کاربر و هسته سیستم عامل ویندوز را داشته و در کنار خواص دیگر از جمله اطلاعات معماری، خصوصیات ساختار PE شامل تعداد، نام، اندازه قسمت‌ها و هش‌های MD5 و SHA-1 گزارش رفتار برنامه تحت تحلیل را در قالب‌های XML و PDF برای تحویل به برنامه کشف بدافزار و کاربر فراهم می‌کند این ابزار برای تحلیل بدافزارها در آزمایشگاه و همچنین تعبیه در ضد ویروس بومی پارسا مورد بهره برداری قرار گرفته است.



شکل ۸. تصویری از محیط برنامه جعبه شن پیشنهادی در زمان تحلیل بدافزار

ضمیمه ۲

نام و شرح عملکرد توابع سیستمی یاد شده در طول مقاله

جدول ۲. توابع مبتنی بر تعامل با رجیستری سیستم

شرح عملکرد	نام تابع
ایجاد یک کلید جدید در رجیستری یـا خوانـدن یک کلید موجود	NtCreateKey
بازکردن یک کلید از رجیستری	NtOpenKey
تعیین یک مقدار برای یک کلید در رجیستری	NtSetValueKey
خواندن مقدار از یک کلید در رجیستری	NtQueryValueKey
حذف یک کلید از رجیستری	NtDeleteKey
حذف مقدار یک کلید از رجیستری	NtDeleteValueKey

جدول ۳. توابع مبتنی بر تعامل فایل سیستم

شرح	نام تابع	
باز کردن فایل	ZwOpenFile	NtOpenFile
خواندن از فایل باز شده	ZwReadFile	NtReadFile
نوشتن در فایل باز شده	ZwWriteFile	NtWriteFile
ایجاد فایل	ZwCreateFile	NtCreateFile
حذف فایل	ZwDeleteFile	NtDeleteFile

جدول ۴. توابع مرتبط با سرویس‌های سیستم

شرح	نام تابع
ایجاد یک سرویس جدید	CreateService
حذف یک سرویس	DeleteServie
شروع کار یک سرویس ایجاد شده	StartService

A Secure Environment for Behavioral Malware Analysis

D. Javaheri[1*], S. Parsa[2]

1- M. Sc. Graduate, College of Post-Graduate Education, Islamic Azad University, Borujerd Branch
2- Associated Professor at Department of Computer Engineering, Iran University of Science and Technology

(Reccive: 2014/04/28, Accept: 2014/08/23)

Abstract

In this article we propose a file analyzer based on sandbox in the client side. This analyzer environment is used for safe execution of a suspicious application to find its behavior and determine if it is safe or not. This sandbox can also be used for behavioral modeling of a malware by in hand execution for understanding distractive and infecting pattern of malwares for creating disinfection and a cleaner method. The advantages of proposed method is in reducing problems with malware detection specifically in detection of obfuscated and metamorphic malwares that can't be detected by signature and static base analysis methods. So this contains the main goal of this article for providing platform of dynamic analysis. Proposed sandbox can monitor and track incoming requests of an application in both user and kernel mode of operating system. This article clusters incoming requests in 8 families with performing data mining on 21000 samples of malwares and benign files and replying them with 5 policies y including logging, redirection, rejecting, cheating and emulating of system resources. Our sandbox guarantees health of operation system during execution and analysis of malwares. In addition this article discusses challenges on dynamic analysis and analyzer environment and gives solutions for them. Most of the challenges focus on methods of detecting and bypassing analyzer environments. At last, this article evaluates the proposed sandbox based on the potentiality and capabilities of behavioral tracking and usage of system resources and compares it with some top famous analyzers in the word.

Keywords:

Analyzer Environment, Sandbox, Malware Analysis, Software Verification, System Call Tracking, Hooking

*** Corresponding Author Email:** d.javahery@iaub.ac.ir