DOR: 20.1001.1.2538161.1402.8.3.1.7

DOI: 10.22034/CSJ.2023.430169.1003

پاییز ۱۴۰۲، دوره ۸، شماره ۳، پیاپی ۳۰

دریافت مقاله: ۱۴۰۲/۰۷/۱۵

پذیرش مقاله: ۱۴۰۲/۰۹/۱۸

نوع مقاله: پژوهشی

استفاده از تجزیه و تحلیل گزارش در ارائه مدلی از سیستم‌های خودترمیمی در برابر حملات ایستا و پویا

امیر اصغری*
کارشناس ارشد مهندسی کامپیوتر
پست الکترونیکی: Amir.11954.asghari@gmail.com

اسلام ناظمی
دانشیار دانشکده مهندسی و علوم کامپیوتر دانشگاه شهید بهشتی
پست الکترونیکی: nazemi@sbu.ac.ir

چکیده

در این مقاله در پی ارائه مدلی از سیستم‌های خودترمیمی در برابر حملات ایستا و پویا می‌باشیم که با کمک تکنیک‌های جدید و شناخته شده همانند پردازش فرآیند و تحلیل گزارش‌ها در تشخیص زود هنگام و تعمیر خودکار استفاده گردد، تا به امروز روش‌های محافظتی متعددی ارائه شده‌اند که عملکرد خوبی در برابر حملات مختلف داشته‌اند، اما با گذشت زمان، حملات جدید، آن روش‌ها را با شکست مواجه کرده است. امروزه اینترنت اشیا (IoT) به سرعت در حال توسعه در کاربردهای متنوع و حیاتی مانند سنجش محیطی و سیستم‌های کنترل صنعتی می‌باشد دستگاه‌های اینترنت اشیا می‌توانند از نظر معماری سخت افزاری و نرم‌افزاری و ارتباطات بسیار ناهمگن باشند. بنابراین زمانی که این دستگاه‌ها بهم متصل می‌شوند یک سیستم پیچیده (مجتمع) ایجاد می‌شود و تشخیص هر گونه ناهنجاری می‌تواند چالش برانگیز باشد. در کارهای گذشته بیشتر به تحلیل حملات به صورت ایستا پرداخته شده و به تحلیل حملات در حالت پویا و در هنگام اجرای برنامه پرداخته نشده است. پیچیدگی زمانی بالا و افزایش آسیب به سیستم از دیگر مشکلات روش‌های گذشته می‌باشد. در روش پیشنهادی مدلی از سیستم‌های خودترمیمی که قابلیت تحلیل حملات در حالت پویا و در هنگام اجرای برنامه دارا می‌باشد ارائه شده است. بالاترین میزان دقت در کارهای گذشته ۹۵,٤٪ می‌باشد ولی دقت تشخیص حملات در روش پیشنهادی به ۹۸,۵٪ ارتقا یافته است.

واژه‌های کلیدی: حملات ایستا، حملات پویا، نرم‌افزارهای سالم، سیستم‌های خودترمیم، رفتار برنامه

۱- مقدمه

در حال حاضر هنگامی که سیستمی دچار حملات مختلفی اعم از ویروس، تروجان و.. می‌گردد و در صورتی که سخت افزار یا نرم‌افزار محافظی بر روی آن در حال اجرا باشد. در مقابل حملات مختلف بسته به نوع آن، از یکی از روش‌های ذیل استفاده می‌نماید :

۱. حذف کردن: هنگامی که فایلی کاملا دچار آسیب شده باشد اقدام به حذف آن می‌نماید.

۲. اصلاح کردن: قسمتی از کدهای برنامه که دچار

* نویسندهٔ مسئول

تخریب شده را تشخیص و اقدام به اصلاح آن می‌نماید.

۳. قرنطینه : در صورتی که نتواند کدهایی از برنامه که دچار تخریب شده باشد را اصلاح نماید آن را در قرنطینه قرار داده تا به امید آن که در آینده بتواند آن را اصلاح نماید.

استفاده از این روش‌ها هر کدام معایبی دارد که به شرح ذیل می‌باشد:

۱. با وجود اصلاح فایل‌های آسیب دیده، ممکن است قسمتی از آن ترمیم نگردد و منجر به بروز مشکلاتی در آینده گردد.

۲. در هنگام اصلاح فایل‌های آسیب دیده، به اشتباه یک کد دودویی تغییر نماید و منجر به بروز نتایج غیر قابل پیش بینی گردد.

۳. امکان تشخیص غلط نوع حمله و انجام عملیات اشتباه در اصلاح کدهای آسیب دیده گردد.

حال با عنایت به موارد مطروحه در اینترنت اشیا به عنوان یک پارادایم در حال ظهور که قادر به اتصال یکپارچه شخصی و دستگاه‌های الکتریکی صنعتی بین خودشان و سرویس‌های ابری آنلاین می‌باشد. این پدیده شامل اتوماسیون خانگی، محیطی حس کردن همانند اکثر صنایع همچون دستگاه‌های سنگین، اتومبیل‌ها و.. جایی که هر دستگاه به سیستم‌های نظارت متصل و فعال‌سازی شده است. اینترنت اشیا می‌تواند به صورت توزیع شده همانند یک سیستم یا سیستم‌هایی که شامل مجموعه‌ای از ملموس و ناملموس مولفه‌هایی هست که به صورت سیمی و یا بی‌سیم بهم متصل هستند. کل سیستم معمولا با اتصال به اینترنت شامل سرویس‌های ابری می‌باشد. این اجزا ممکن است طیف وسیعی از سنسورها را که می‌توانند از انواع مختلف اتصالات محلی و منطقه وسیع استفاده کنند و می‌توانند رخدادها و فعالیت‌های محیطی و فعالیت‌ها را ضبط کنند که می‌تواند به اجزای دیگر مولفه‌ها همچون مجموعه داده و پردازش اطلاعات و همبستگی رویداد سیستم‌ها منتقل شود. همه اطلاعات می‌تواند استفاده کرده باشد حال در پی ارائه مدلی از سیستم‌های خودترمیمی می‌باشیم که تا حد امکان با نظارت و بررسی برنامه‌های در حال اجرا بر روی هر کدام از این سیستم‌ها به تشخیص حملات مختلف بر روی آن شده و از ادامه اجرای آن جلوگیری نماید و تغییرات بوجود آورده شده در سیستم را اصلاح و اقدام به ترمیم خود نماید و به دلیل توزیع شدگی این سیستم ها، شفافیت خرابی کمک بسیار بزرگی در اصلاح و رفع مشکلات خواهد نمود[۱۶].

۱-۱ شرح مساله

امروزه تشخیص، جلوگیری و ترمیم حملات رخ داده برای کارشناسان امنیتی یک چالش محسوب گشته، زیرا حملات هر روز با تکنولوژی و راهکارهای جدید بروزرسانی می‌گردند و منجر به خسارات بزرگی می‌شوند. سیستم‌های تشخیص حملات که از امضاءها برای شناسایی استفاده می‌کردند دیگر نیازهای ما را تامین نکرده و امنیت را تضمین نمی‌کنند. به طور کلی ۲ نوع روش برای تجزیه و تحلیل و بررسی وجود دارد :

- روش‌های مبتنی بر امضا [1]
- روش‌های مبتنی بر رفتار[2]

روش‌های مبتنی بر امضا: در این روش تجزیه و تحلیل عبارت است از بررسی فایل اجرایی بدون مشاهده دستورالعمل‌های آن، تجزیه و تحلیل در این روش بسیار ساده و سریع است ولی در برابر کدهای پیچیده بی اثر است. تمامی نرم‌افزارهای محافظ امروزی از این روش استفاده می‌نمایند ولی از معایب آن تشخیص ضعیف آنها می‌باشد.

روش‌های مبتنی بر رفتار: در این روش با بررسی رفتارهای نرم‌افزار سعی در شناسایی حملات دارند امروزه با استفاده از الگوریتم‌های مبهم‌سازی[3] روش مبتنی بر امضا را با مشکل روبه‌رو نموده است. حال در این روش به دلیل استفاده از رفتارهای نرم‌افزار، از روش

1- Signature-based
2- Bahavior-based
3- Obfuscation

مبهم‌سازی رفتار جهت جلوگیری از تشخیص حملات استفاده می‌گردد، محققان امروزه با استفاده از راهکارهای مدل‌سازی و داده کاوی به تشخیص این‌گونه رفتارها می‌پردازند و پس از جمع‌آوری تعداد زیادی از حملات و مشاهده رفتارهای آنها، مدلی از رفتارهای مخرب آنها ساخته و از این مدل برای تشخیص حملات در حال اجرا و پویا استفاده می‌نمایند[۱۵]. سه مشکل در این زمینه وجود دارد:

- ضعف تشخیص حملات در زمان اجرای برنامه.
- عدم بهره‌گیری از برنامه‌های سالم جهت ساخت مدل.
- بالا بودن پیچیدگی زمانی مدل‌ها.

تقریبا تمامی روش‌های موجود برای تشخیص حملات، از یک محیط مجازی یا یک محیط جدا شده استفاده شده است تا حملات، آسیبی به سیستم اصلی وارد نکند. به طورکلی پس از اینکه اجرای نرم‌افزار به پایان رسید، با بررسی رفتار نرم‌افزار، مشخص می‌گردد که آیا سیستم دچار تخریب شده است یا خیر. این مسئله منجر به این می‌گردد که نتوان نرم‌افزار را در محیط واقعی و در زمان اجرا بررسی نمود.

در مدل پیشنهادی تا حد امکان به نظارت و بررسی برنامه‌های در حال اجرا می‌پردازیم و در صورت تشخیص حملات مختلف از ادامه اجرای آن جلوگیری می‌نماییم و آسیب‌های رخ داده در سیستم را ترمیم می‌نماییم. این مقاله از ۵ بخش تشکیل شده است که در بخش اول به توضیحاتی مقدماتی از شرح مسئله، در بخش دوم به کارهای انجام شده، در بخش سوم به بررسی مدل پیشنهادی، در بخش چهارم نتیجه گیری و در بخش آخر منابع معرفی می‌گردند.

۲- کارهای انجام شده

در [۱۱] با اجرای مجموعه‌ای از انواع حملات به طور خاص بدافزار و نرم‌افزار سالم در یک محیط ماشین مجازی و با نظارت رفتارهای آن‌ها، رفتار این برنامه‌ها از طریق APIهای فراخوانی شده، مقادیر آرگومان‌های آن‌ها و ... تجزیه وتحلیل می‌شود. فایل‌های باینری، همه‌ی APIها را فراخوانی نکرده‌اند اما یک زیرمجموعه از ۱۲۶ DLL که ۶ DLL می‌باشد انتخاب شده است.

جدول (۱): شرحی از DLLهای انتخاب شده [۱۱]

User32.dll	Windows management functions for message handling, timers, menus, and communications
Kernel32.dll	Low-Level operating system functions for memory management and resource handling
Advapi32.dll	Advanced API services library supporting numerous APIs including many security and registry calls
Ntdll.dll	NT Layer DLL that control NT system functions
Ws2_32.dll	Contains the windows socket API used by network and internet applications to manipulate their connections
Wininet.dll	Enables applications to access standard Internet protocols, such as FTP and HTTP

در این روش، پیشنهاد شده که استفاده از فراخوانی‌های سیستمی در روش‌های تشخیص بدافزار فوق‌العاده موثر است. در این روش برای مجموعه دوم از داده‌ها، آدرس‌های اشاره‌گر و سایر پارامترها به همراه APIهای فراخوانی شده به عنوان ویژگی‌های استخراج شده مورد استفاده قرار گرفته‌اند. در طی فرآیند استخراج ویژگی، تعدادی از نمونه‌ها که در آن‌ها ویژگی‌های مربوطه وجود دارد نیز تعیین می‌شود. در مرحله‌ی بعد نوبت به انتخاب ویژگی برای هر فایل می‌رسد تعداد فراخوانی مربوط به هر API موردنظر مشخص می‌شود. به منظور محدود کردن دامنه‌ی ویژگی‌ها در تعداد کمی از فایل‌ها وجود دارند ویژگی‌هایی را که فراوانی آنها از یک آستانه‌ی خاص کمتر است حذف می‌شوند.

در مرحله‌ی بعد با استفاده از الگوریتم انتخاب ویژگی، مجموعه کوچکتری از ویژگی‌ها استخراج می‌شود. پس از انتخاب بهترین ویژگی‌ها، وجود یا عدم وجود هر ویژگی در لیست رشته چک شده و با بردار فضایی نشان داده شده است. اگر ویزگی انتخاب شده در فایل ورودی قابل دسترس باشد، مقدار آن ویژگی روی ۱ و در غیر این صورت روی ۰ تنظیم می‌شود. مجموعه بردارهای فضایی ایجاد شده به عنوان ورودی‌هایی برای طبقه بندی و ساخت

مدل استفاده شده‌اند. برای این کار دقت الگوریتم‌های مختلف روی این دو مجموعه داده مورد آزمایش قرار گرفت و نتایج نشان داد که هنگامی که برای تشخیص بدافزار ترکیبی از فراخوانی‌های API و آرگومان‌ها مورد استفاده قرار گیرد میزان دقت ٦٪ افزایش خواهد یافت که در بهترین حالت ۹۵,٤٪ خواهد بود و چالش نویسنده در روش پیشنهادی خود کاهش پیچیدگی زمانی در هنگام اجرای برنامه می‌باشد.

در [۸] تجزیه و تحلیل برنامه توسط سندباکس[4]، مفهوم استفاده از سندباکس برای اولین بار توسط Wahbe در زمینه‌ی ایزوله کردن خطای نرم‌افزار معرفی شد. بسیاری از مطالعات، سندباکس‌های قدیمی را به سمت تاکیدات متفاوت بهبود بخشیده‌اند که عبارتند از اجرای سندباکس برای بررسی‌های امنیتی، سندباکس با سیاست پویا و غیره. تجزیه و تحلیل سندباکس، به صورت اجرا در یک محیط ایزوله شده انجام می‌شود و این یک تکنیک متداول برای شناسایی بدافزار است.

نمودار (۱): Sandbox

تجزیه و تحلیل سندباکسی که در دسترس عموم قرار دارد کمک می‌کند تا کاربران رهگیری‌های اجرا را بدون هیچگونه صدمه‌ای به سیستم ببینند. این سبک با جمع‌آوری بدافزارها به طراحان سند باکس در تجزیه و تحلیل کمک شایانی می‌کند. میزان دقت محاسبه شده در این روش ۷۹,٥٪ خواهد بود و چالش نویسنده در روش پیشنهادی امکان تشخیص حملات در هنگام اجرای برنامه به صورت پویا می‌باشد.

در [٦] آنوبیس[5]، ایده‌ی اصلی آنوبیس ردیابی فراخوانی‌ها به سیستم عامل در سطح CPU می‌باشد. آنوبیس به عنوان یک ماشین شبیه‌ساز اصلاح شده در نظر گرفته می‌شود. موتور پایه آن، ماشین توسعه یافته‌ی شبیه‌ساز QEMU می‌باشد. نحوه کار آن بدین‌صورت می‌باشد که زمانی اسکریپتی اجرا می‌شود در ویندوزی که به عنوان مثال در QEMU اجرا شده است کپی می‌شود و در نتیجه فایل باینری آن اجرا شده و سیستم نتایج آن را جمع‌آوری کرده و آنها را در چندین فرمت خروجی ارائه می‌دهد. آنوبیس توانایی ردیابی اجرای باینری‌های Win ۳۲[6] را دارد و از شبیه‌سازی استفاده می‌کند که دید سطح پایین‌تری از سیستم را دارد. برای هر دستوری که اجرا می‌شود. آنوبیس به دلیل استفاده از ثبات‌های CPU، دیدی کامل از CPU دارد. از مشکلات اساسی آنوبیس می‌توان به دو مورد ذیل اشاره کرد و میزان دقت محاسبه شده در این روش ۸۲٪ خواهد بود و چالش نویسنده افزایش میزان دقت، مطابقت دادن برنامه با دستورات CPU و مطابقت دادن فراخوانی DLLها با دستورات CPU می‌باشد.

در [۷] استفاده از محیط امن به منظور رهگیری رفتار و کشف بدافزار، همان طور که قبلا گفته شد به منظور تحلیل نرم‌افزار موردنظر با روش‌های مبتنی بر رفتار، برنامه‌ی هدف را در محیطی امن به اجرا درآورده و در حین اجرا عملیات تحلیل انجام می‌گیرد. در ادامه به برخی از مزایای این محیط‌های امن اشاره شده است.

- رهگیری رویدادها در سیستم عامل: همان‌طور که می‌دانید سیستم‌های عامل، از دسته نرم‌افزارهای پیچیده‌ای هستند که کوچکترین خطا یا بن بست در هر قسمت از آنها منجر به خسارات جبران‌ناپذیری خواهد شد از این‌رو از ابزارهایی که امکان جمع‌آوری بسیار سریع اطلاعات را

4- Sandbox

5- Anubis
6- Binaries

فراهم می‌کنند و بسیاری از پیچیدگی‌های تکنیکی مربوط به هسته سیستم عامل را مخفی نگه می‌دارند استفاده می‌گردد.

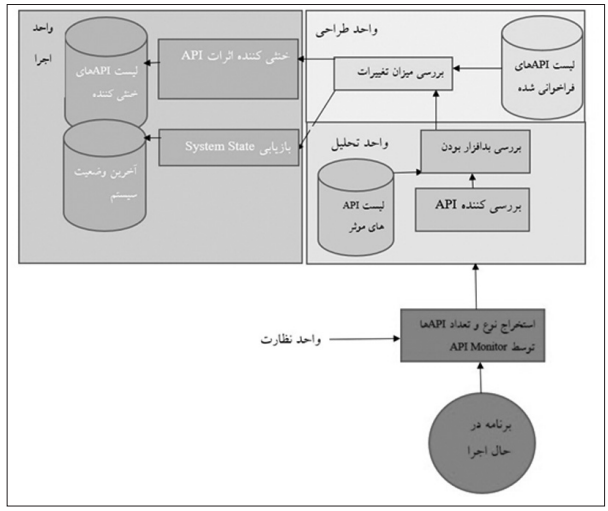
ماشین مجازی به عنوان محیط امن: ماشین مجازی یک نوع برنامه‌ی کامپیوتری است که برای ایجاد یک یا چند محیط اجرایی مورد استفاده قرار می‌گیرد که این محیط‌ها از نظر فیزیکی وجود دارند.

- ماشین مجازی در تکنولوژی تجزیه و تحلیل اکتشافی ایستا: هنگام اعمال تشخیص بدافزار، تجزیه و تحلیل اکتشافی برای تشخیص یک فایل مخرب بالقوه از یک رویکرد مبتنی بر امضا استفاده می‌کند. تکنولوژی تجزیه و تحلیل ایستا به وسیله‌ی اسکن کردن کد یک فایل برای ویژگی‌های مشکوک بدافزار تعیین می‌کند که این فایل، بدافزار است یا خیر.
- ماشین مجازی در تکنولوژی تجزیه و تحلیل اکتشافی پویا: تکنولوژی تجزیه و تحلیل از طریق اجرا کردن بدافزار در یک محیط محدود شده مثلا ماشین مجازی رفتار بدافزار را مشاهده، تجزیه و تحلیل می‌کند. به طور سنتی رفتار بدافزار می‌تواند مستقیما از طریق API در سخت افزار محدود به نوع ماشین مجازی دستگیر شود[۱۲].
- در کارهای گذشته به بررسی حملات به صورت ایستا پرداخته شده است و به بررسی حملات در حین اجرای برنامه و به صورت پویا نمی‌پردازد، در صورتی که عمده حملات رخ داده امروزه به صورت پویا و در حال اجرای برنامه می‌باشد و نیاز به سیستمی که به صورت پویا بتواند حملات را تشخیص دهد و از ادامه اجرای برنامه جلوگیری نماید و سپس با استفاده از مراحل خودترمیمی اقدام به ترمیم و بازگشت به حالت سالم خود گردد احساس می‌گردد. در [۱۶] تجزیه و تحلیل لاگ‌ها و پردازش برای شناسایی تکنیک تشخیص آنومالی استفاده کرد. در این بخش به ناهنجاری‌های رفتاری مثلا امنیت، خطاهای نرم‌افزار یا سخت افزار با استفاده از تکنیک‌های استخراج شده از پردازش فرآیندها می‌پردازیم. در ابتدا با چگونگی رخدادها شروع می‌کنیم و می‌توانیم لاگ‌ها در عملکرد اینترنت اشیا ثبت کرد. سپس ما می‌توانیم به استخراج فعالیت‌ها توجه کنیم و مدل‌ها و رفتارهای ناهنجار از لاگ رخدادها را کشف کنیم، در صورت امکان کد برنامه می‌تواند به صورت صریح قانون‌های لاگ کردن به صورت صریح بیان کند این قوانین لاگ کردن باعث شده وقتی که رویدادهای خاص رخ می‌دهد انجام گردد. در سیستم‌های اینترنت اشیا گرچه بیشتر زمان لاگ کردن داده‌ها در سطح پایین داده‌های حساس جمع‌آوری شده توسط دستگاه‌های اینترنت اشیا است. در این کار ما علاقمند به تجزیه و تحلیل فعالیت‌های انجام شده توسط ارتباط دستگاه‌های اینترنت اشیا هستیم. در ابتدا ما به استخراج چارچوبی برای استخراج از فعالیت داده حساس سطح پایین می‌پردازیم. پس از آن با فرض اینکه به صراحت لاگ رویدادها از برنامه ها، ما تکنیک‌هایی برای ابزارسازی کد از لاگ رخدادها تولید می‌کنیم.

۳- راهکار پیشنهادی

در کارهای گذشته تا فاز تحلیل به بررسی موضوع پرداخته شده بود و در مدل پیشنهادی به بهبود این واحد پرداخته شده است و همچنین بخش طراحی و اجرا که معرفی مدلی از خودترمیمی می‌باشد اضافه شده است. در این روش یک دستورالعمل به دو فرآیند برای تشخیص و مقابله با حملات استفاده می‌گردد ۱- فرآیند ناظر (M-Process) ۲- فرآیند برنامه اصلی (P-Process) که به نظارت بر جریان کنترل فرآیند برنامه اصلی می‌پردازد. در مرحله کامپایل، برنامه به دو فرآیند تقسیم می‌گردد: که فرآیند ناظر شامل شرایط سازگاری جریان کنترل برای فرآیند اصلی می‌باشد. در صورتی که حملاتی توسط فرآیند ناظر تشخیص داده شود یک عکس‌المعل با توجه به نوع حمله رخ داده توسط فرآیند اصلی اجرا می‌گردد.

نمودار فرآیند مدل پیشنهادی به شرح ذیل می‌باشد :



نمودار (۲): فرآیند مدل پیشنهادی

شرحی اجمالی از قسمت‌های مدل پیشنهادی:

- نظارت: با اجرای نرم‌افزار، نرم‌افزار API Monitor اقدام بـــه جمع‌آوری، تجمیع، مدیریت، پالایش و گزارش جزییات دریافت شده از فراخوانی‌های سیستمی، می‌پردازد[۱۳].
- تحلیل: با توجه به خروجی‌های واحد نظارت، با بررسی نوع و تعداد فراخوانی‌های سیستمی و مقایسه با الگوهای نشـــان‌دهنده بدافـــزار، اقـــدام به تحلیـــل وضعیت موجود می‌نماید.
- طراحی: در این قســـمت با توجه به خروجی که از واحد تحلیل به دســـت آمده اقدام بـــه طراحی مکانیزم‌هایی برای عملیاتی نمودن در راســـتای اهداف سیستم می‌نماید و با توجه به میزان آسیب دیدگی اقدام به ترمیم خود می‌نماید.
- اجـــرا: در این قســـمت با توجه به خروجـــی که از واحد طراحی به دســـت آمـــده به اجـــرا مکانیزم‌هـــای مربوطه می‌پردازد.
- پایگاه داده: این قسمت در مرکز چرخه خود ترمیمی قرار دارد اطلاعـــات، دیتاها و مدل‌های مورد نیاز برای هر کدام از چهار قسمت نامبرده در این قسمت ذخیره شده است.
- قابل ذکر اســـت که در این روش نیاز به اجرای همزمان نرم‌افزارهای نظارت و کشف می‌باشد. با تعریف یک زمان تعیین شده برای ثبت فایل سابقه‌ای توسط برنامه‌ی نظارت و ارسال آن به برنامه‌ی کشف و با تعیین یک زمان خاص

برای این منظور با توجه به میزان اهمیت موضوع امنیت، با شـــروع اجرای برنامه‌ی هدف، فایل سابقه‌ای در زمان‌های تعیین شده، به برنامه‌ی کشف ارسال شده و مورد بررسی قرار می‌گیرد. اگر براساس این سری از فایل‌های سابقه‌ای نشـــان از وجود بدافزار باشد از ادامه اجرای آن جلوگیری به عمل می‌آید. مهمترین موضوع در شناســـایی و کشـــف بدافزار ســـاخت یک الگوی رفتاری، از بدافزار اســـت که با مقایسه‌ی رفتار یک برنامه‌ی ناشناس با این الگو، پر خطر و بی‌خطر بودن آن قابل تشـــخیص باشـــد. در این روش رفتار برنامه با تعداد فرخوانی‌های APIهای موثر مشخص می‌گـــردد. در این مدل مهمترین موضوع در شناســـایی و کشف بدافزار تشخیص الگوی رفتار یک بدافزار است که با مقایســـه‌ی رفتار یک برنامه‌ی ناشناس با این الگو، پرخطر و بی خطر بودن آن قابل تشخیص می‌باشد. در این روش رفتار برنامه با تعداد فراخوانی‌های APIهای موثر مشخص می‌گردد. به طورکلی استفاده از تمام اطلاعات ذخیره شده در زمان اجرا یعنی استفاده از تمام اطلاعات موجود در فایل سابقه‌ای به دست آمده از نرم‌افزار نظارت بسیار زمان‌بر و از طرفی ممکن اســـت منجر به آگاهی بدافزار نسبت به این نظارت خواهد گردید. با تشخیص بدافزار از ادامه‌ی اجرای آن جلوگیری به عمل آمده و لیست APIهای فرخوانی شده مورد بررسی قرار گرفته و با بررسی میزان تغییرات اعمال شده بر روی فایل‌ها، فرآیندها، شبکه، پنجره‌ها، رجیستری و سرویس‌های ویندوز بررسی می‌گردد.

تصمیم گرفته می‌شـــود کـــه از APIهـــای خنثی‌کننده استفاده گردد و این مرحله مستقل از مرحله تشخیص بوده و بایـــد کارایی لازم را حتی برای روش‌های تشـــخیص نه چندان قوی نیز داشـــته باشـــد. برای مثال روش تشخیص نباید اجـــازه دهد که یک بدافزار فایلـــی را ایجاد کند اما با این وجود، در مرحلـــه بازیابی، فراخوانی‌های API مربوط به فایل در نظر گرفته می‌شود و در هر صورت اگر در این مرحله، تشـــخیص داده نشود و فایلی ایجاد شود، آن فایل بایـــد در مرحله بازیابی حذف گردد. در صورتی که از این

روش نتوان ترمیم را انجام داد با توجه به آخرین تصویری که از دیتاها، وضعیت سیستم[۷] که در زمان‌بندی‌های مشخص گرفته شده است و در فایلی که بر روی آن کلمه عبوری قرار گرفته شده است تا از حملات ویروس‌های خودکار جلوگیری به عمل آید استفاده می‌گردد و عملیات بازیابی در سطح کل سیستم صورت گرفته تا سیستم به وضعیت سالم مشخص شده برگردد.

عملیات بازیابی: تقریبا تمامی حملات امنیتی از چند مرحله تشکیل شده است. در واقع دنباله‌ای از عملیات یک حمله را تشکیل می‌دهند حال آن که ممکن است هر کدام از این عملیات خود به خودی نشان‌دهنده یک حمله نباشد. با توجه به اینکه در این روش بررسی بدافزار بودن یک برنامه، بر روی سیستم واقعی صورت می‌گیرد، نیاز است تا از ورود آسیب به آن جلوگیری گردد. این مدل به طورکلی از دو مرحله ذیل تشکیل شده است. ۱- تشخیص بدافزار و جلوگیری از ادامه اجرای آن ۲- خنثی کردن تاثیرات تا قبل از اتمام اجرا. در حقیقت این دو مرحله در محافظت از سیستم، مکمل یکدیگر هستند، بنابراین اگر روش تشخیص بدافزار، بدافزار را سریع تشخیص دهد، نیاز به خنثی کردن تاثیرات آن کمتر می‌باشد. با توجه به مدل ارائه شده می‌توان ادعا کرد در بخش تشخیص به خوبی عمل کرده و بدافزار را در مراحل اولیه شناسایی کرده ولی این ادعا را نمی‌توان داشت که هیچ تاثیری را روی سیستم نمی‌گذارد. به همین علت نیاز است تا عملیاتی را جهت خنثی کردن تاثیرات بدافزار روی سیستم انجام داد که عملیات بازیابی نامیده می‌شود.

مراحل اجرای بازیابی سیستم: در فاصله‌های زمانی مشخص از آخرین وضعیت سیستم، برنامه‌ها و داده‌های آن یک تصویر گرفته می‌شود و در یک فایل ذخیره می‌گردد و بر روی آن یک کلمه عبور نسبت داده می‌شود تا از حمله توسط ویروس‌های خودکار جلوگیری گردد. با توجه به اینکه در این روش، بررسی بدافزار بودن یک برنامه، بر روی یک سیستم واقعی صورت می‌گیرد، نیاز است تا از ورود آسیب به آن جلوگیری شود. حال در صورتی که تاثیرات اجرای بدافزار قابل بازگشت نباشد نیاز به اجرای فاز بازیابی می‌باشد.

پس از بررسی APIهای مختلف APIهایی که در هر گروه قرار می‌گیرند به دست آمده و برای هر کدام در صورت امکان، API خنثی‌کننده آن در نظر گرفته می‌شود. حال در صورت عدم تشخیص و جلوگیری از عدم تاثیر اجرای بدافزار چرخه خودترمیمی اقدام به بازیابی کامل خود می‌نماید بدین‌صورت که از آخرین وضعیت سیستم و دیتاهای آن که در بخشی از ذخیره‌ساز ذخیره شده است اقدام به بازیابی خود می‌نماید.

جدول (۲): نمونه‌ای از APIهای قابل ترمیم

دسته	نام APIها
کپی یا حذف کردن	CloseHandle, CopyFile, CopyFileEx, CopyFileTransacted, CreateFile, CreateFileTransacted, CreateHardLink, Transacted, CreateSymbolicLink, LinkTransacted, DeleteFile, DeleteFileTransacted
نوشتن روی فایل	OpenFile, OpenFileById, ReOpenFile, ReplaceFile, WriteFile, CreateFile
تغییر خصوصیات فایل	SetFileToANSI, SetFileAPIToOEM, SetFileAttributes, SetFileAttributesTransacted, SetFileBandwidthReservation, SetFileInformationByHandle, SetFileShortName, SetFileValidData
جابه جا کردن	MoveFile, MoveFileEx, MoveFileTransacted, MoveFileWithProgress

حال در صورتی که در این مدل عدم سالم بودن نرم‌افزار شناخته نگردد و سیستم دچار آسیب گشته باشد نیاز است خود را بازیابی نماید. ایده استفاده شده در این قسمت بدین‌صورت است که در فاصله‌های زمانی مشخص از آخرین وضعیت سیستم، برنامه‌ها و داده‌های آن یک تصویر گرفته می‌شود و در یک فایل ذخیره می‌گردد و بر روی آن یک کلمه عبور نسبت داده می‌شود تا از حمله توسط ویروس‌های خودکار[۸] جلوگیری گردد. در صورتی که نیاز به بازیابی خود داشته باشد با استفاده از کلمه عبور به فایل مربوطه وارد شده و اقدام به بازیابی خود می‌نماید.

برای سنجش میزان دقت[۹] روش پیشنهادی به بررسی میزان دقت واحد تحلیل می‌پردازیم که نیاز به تعریف معیارهایی ذیل می‌باشد که در ادامه به تعریف آن‌ها

7- System State

8- Autoran

9- Accuracy

می‌پردازیم.

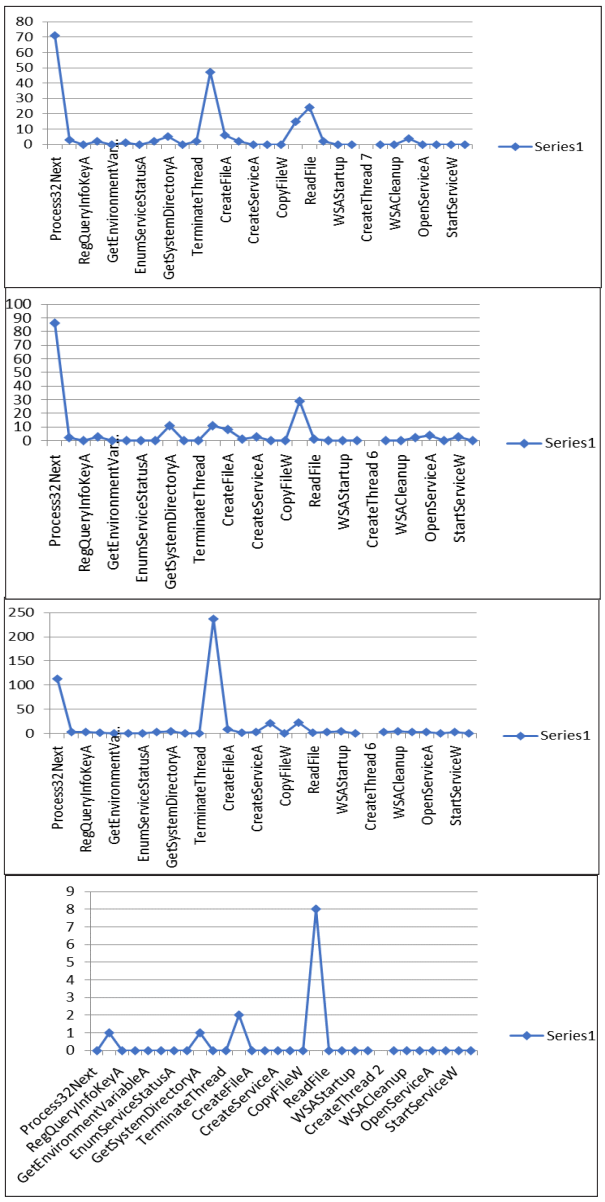
این معیارها عبارتند از :

- نرخ مثبت درست[10]: این معیار درصد بدافزارهایی را که به عنوان بدافزار شناسایی شده‌اند نشان می‌دهد.
- نرخ مثبت کاذب[11]: این معیار درصد برنامه‌های سالمی را که به عنوان بدافزار شناسایی شده‌اند نشان می‌دهد.
- نرخ منفی درست[12]: این معیار درصد برنامه‌های سالمی را که به عنوان برنامه‌ی سالم شناسایی شده‌اند نشان می‌دهد.
- نرخ منفی کاذب[13]: این معیار درصد بدافزارهایی را که به عنوان برنامه‌ی سالم تشخیص داده شده‌اند نشان می‌دهد.

در این مقاله ارزیابی روش پیشنهادی توسط دو معیار نرخ مثبت درست و نرخ مثبت کاذب انجام شده است. محاسبه‌ی دقت با استفاده از رابطه‌ی زیر قابل محاسبه است :

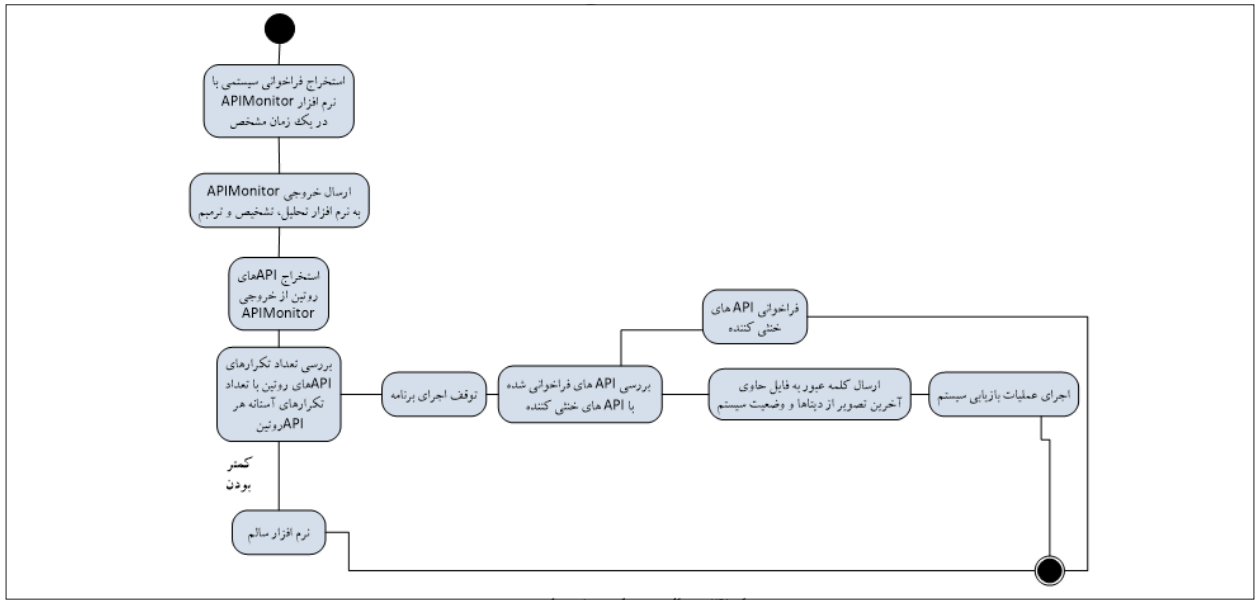
$$\text{دقت} = \frac{TP + TN}{TP + TN + FP + FN}$$

مجموعه داده‌ای که برای آزمون روش پیشنهادی استفاده شده است شامل ۸۰ نوع حمله که به طور خاص بدافزار و ۱۰۰ نرم‌افزار سالم می‌باشد که بدافزارها شامل انواع ویروس، کرم‌ها، تروجان و غیره می‌باشد و نرم‌افزارهای سالم شامل فایل‌های اجرایی ویندوز و برنامه‌های کاربردی نصب شده بر روی یک ماشین مجازی با منابع، یک پردازنده ٤ هسته‌ای و ۸ گیگابایت حافظه می‌باشد. پس از مرحله‌ی رهگیری رفتار برنامه‌ی ناشناس و به دست آمدن ویژگی‌های رفتاری آن نوبت به این می‌رسد که با استفاده از این ویژگی‌ها الگوی رفتاری مخرب ساخته شود و از طریق آن مخرب بودن برنامه‌ی ناشناس تعیین شود. در این روش‌ها از تغییراتی که بدافزار ایجاد می‌کند و با DLLها و APIها و دیگر رشته‌هایی که به صورت پویا استخراج می‌شوند به عنوان ویژگی‌هایی برای شناسایی بدافزارها به کار می‌رود. اطلاعات زمانی و مکانی که در زمان اجرای برنامه قابل دستیابی است برای ساخت مدل و تشخیص و تمایز بدافزار از نرم‌افزارهای سالم به کار می‌روند. در این گونه روش‌ها ویژگی‌های رفتاری برنامه‌ها (بدافزارو نرم‌افزار سالم) استخراج و از این طریق دستیابی به رفتار برنامه امکان‌پذیر می‌شود. با بررسی این رفتار می‌توان نوع برنامه را تشخیص داد. لازم به ذکر است که در روش‌های گوناگون رفتار بدافزار براساس فاکتورهای مختلفی به عنوان ویژگی‌های رفتاری، ترسیم شده است، که با توجه به آن‌ها تمایز میان بدافزار و نرم‌افزار سالم مشخص می‌شود. که در ذیل نمونه‌هایی از تعداد فراخوانی‌های آنها قابل مشاهده است.



نمودار (۳): نمونه ۴ عدد از تعداد فراخوانی‌های APIهای موثر در بدافزارها

10- Tru Positive Rate(TP_{Rate})
11- False Positive Rate(FP_{Rate})
12- True Negative Rate(TN_{Rate})
13- False Negative Rate(FN_{Rate})



نمودار(۴): حالت مدل پیشنهادی

با توجه به اینکه مدل پیشنهادی با استفاده از خروجی نرم‌افزار API Monitor پیاده‌سازی شده است، لذا برای روشن شدن شیوه کارکرد آن (روش پیشنهادی) نمودارحالت توالی آن در ذیل نشان داده شده است.

جدول (۳): مقایسه دقت روش پیشنهادی با روش‌های پیشین

روش	دقت
تجزیه و تحلیل به کمک نرم‌افزار سندباکس	٪۷۹,۵
تجزیه و تحلیل به کمک نرم‌افزار آنوبیس	٪۸۲
اجرای مجموعه‌ای از انواع بدافزار و نرم‌افزار سالم در یک محیط ماشین مجازی و نظارت بر رفتارهای آن و استخراج ۶ DLL و تمرکز بر نحوه فراخوانی‌آنها	٪۹۵,۴
روش پیشنهادی	٪۹۸,۵

۴- نتیجه گیری

هدف از انجام این مقاله، ارائه مدلی جدید در زمینه خودترمیمی در هنگام رخ داد آسیب هنگام حملات ویروس می‌باشد که مدل به دست آمده قابل استفاده در مقابل حملات بدافزارها می‌باشد. همان طور که گفته شد سه مشکل در زمینه تشخیص و ترمیم حملات عنوان شد که عبارتند از :

۱. عدم بررسی برنامه در زمان اجرا.

۲. عدم استفاده از برنامه‌های سالم جهت ساخت مدل تشخیص بدافزار.

۳. پیچیدگی زمانی بالای روش‌های موجود.

مدل پیشنهادی با استفاده از بررسی رفتار برنامه مشکلات مطرح شده در بالا را حل نموده است. جهت به دست آوردن این الگوی رفتاری با بررسی تعداد فراخوانی‌های APIهای موثر صورت گرفت و برای به دست آوردن APIهای موثر، ۸۰ بدافزار و ۱۰۰ برنامه‌ی سالم مورد بررسی قرار گرفتند و در نهایت ۳۵ عدد API موثر بدست آمد. برای هر کدام از این APIها نیز مقدار مرزی‌ای به دست آمد که نشان‌دهنده وضعیت بدافزار یا سالم بودن برنامه فراخواننده می‌باشد.

۵- مراجع

[1] Y. Ye, D. Wang, T. Li, D. Ye and Q. Jiang, “An intelligent PE-malware detection system based on association mining”, Springer journal in computer virology, 2008, France, pp.323-334.

[2] U. Bayer, P. Milani Comparetti, C. Hlauschek, C. Kruegel and E. Kirda, “Scalable, Behavior-Based Malware Clustering”. 16th Annual Network and Distributed System Security Symposium (NDSS) San Diego, CA, 2009.

[3] J. Wilander and P. Fak, “Pattern Matching Security Properties of Code using Dependence Graphs”, First International Workshop on Code Based Software Security Assessments (CoBaSSA 2005), Pittsburgh, Pennsylvania, USA, pp. 5 - 8.

[4] Matthias Tichy “A Master Level Course on Modeling Self-Adaptive Systems with Graph Transformations”Organic Computing, Department of Computer Science University of Augs-

burg, Augsburg, Germany 2010.
[5] M. Alazab, S. Venkataraman and P. Watters, “Towards Understanding Malware Behaviour by the Extraction of API Calls”, Second Cybercrime and Trustworthy Computing Workshop, 2010, Ballarat, Victoria, Australia, pp.52-59.
[6] Omer Sezgin U gurlu “STEALTH SANDBOX ANALYSIS OF MALWARE” a thesis submitted to the department of computer engineering and the institute of engineering and science of bilkent university
in partial fulfillment of the requirements for the degree of master of science, 2009.
[7] Kang, M.g., Yin, H., Hanna, S., McCamant, S., Song, D., “Emulating Emulation-Resistant Malware”, In Proceedings of the Workshop on Virtual Machine Security (VMSec), 2009
[8] Oyama, Y., Onoue, K., Yonezawa, A., “Speculative Security Checks in Sandboxing Systems”, in Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, April 2005
[9] Sun, M-K., Lin, M-J., Liah, Ch-S., Lin, H-T., “Malware Virtialization-Resistant Behavior Dtection”, IEEE, 17th International Conference on Parallel and Distributed Systems, 2011.
[10] Ferrie, P., “Attacks on virtual machine emulators”, Symantec Security Response, December, 2006.
[11] Salehi, Z., Ghiasi, M., Sami, A., “A Miner for Malware Detection Based on API Function Calls and Their Arguments”, IEEE 16th CSI International Symposium on Artificial Intelligence and Signal Processing, 2012.
[12]Tao Ma, Shaukat Ali, Tao Yue “Modeling foundations for executable model-based testing of self healing
Cyber-physical systems”,Springer Verlag Gmbh Germany 2018.
[13]Fatma Kachi, Chafia Bouanaka “A hybrid model for effivient decision making in self adaptive systems” Elsevier Information and Software Technology 2023.
[۱۴]علیرضایی، ا.، فروزیده ، ن.، ”تحلیل رفتاری کدهای بداندیش“، پایاننامه کارشناسی ارشد، دانشکده مهندسی کامپیوتر، پردیس کیش دانشگاه تهران، ۱۳۹۰.
[۱۵]محمدمهدی عمادی کوچک ، بیژن ساعدی، مهدی نجف زاده ، حمید محسنی «ارائه معماری نوین شبکه‌های دفاعی گسترده مستحکم در برابر حملات هدفمند»، فصلنامه علمی- پژوهشی فرماندهی و کنترل.. ۱۴۰۰
[16]Prasannjeet Singh, Mehdi Saman Azari,Francesco Vitale,Francesco Flammini “Using log analytics and process mining to enable self-healing in the Internet of Things” Springer 2022