فصلنامه پژوهش‌های حفاظتی ـ امنیتی دانشگاه جامع امام حسین(علیه‌السلام)
سال سوم، شماره‌ی ۱۳ (بهار ۱۳۹۴): صص ۱۰۹ – ۱۲۶

تشخیص بدافزار روت‌کیت با استفاده از روش شخیص ترکیبی و الگوریتم‌های یادگیری ماشین

غلامرضا نامداری[۱]
رضا نورمندی‌پور[۲]

تاریخ دریافت: ۱۳۹۳/۰۹/۲۰
تاریخ پذیرش: ۱۳۹۳/۱۲/۱۵

چکیده

امروزه استفاده از تکنیک‌های «مبهم سازی» باعث پیچیده‌سازی بدافزارها شده به‌گونه‌ای که تشخیص آن‌ها را بسیار دشوار ساخته است؛ از این‌رو تلاش برای تشخیص بدافزارهای چندریختی جدید و ناشناخته، ما را به سمت طراحی سامانه‌های پویا و ایستا برای شناسایی آن‌ها هدایت می‌کند. تعداد و تنوع بدافزارها، باعث ارائه‌ی انواع متعددی از راه‌کارهای دفاعی در مقابل آن‌ها شده است.

این پژوهش علاوه بر مروری کلی بر روی مفاهیم اساسی مانند تشخیص و ارزیابی بدافزار و تکنیک‌های یادگیری، به ارائه یک روش جدید برای تشخیص بدافزارها با تأکید بر دسته‌ای از آنها به نام روت‌کیت پرداخته است. در این روش که بر پایه‌ی توابع سیستمی فراخوانی شده است، هدف ما به‌دست آوردن الگوی دنباله‌ی ApiCall های فراخوانی شده در بدافزارها است که توانسته نرخ تشخیص آن‌ها را به ۹۷٪ برساند. این روش ترکیبی، از یک روش ایستا و یک روش پویا محسوب می‌شود که در بخش ایستای آن، برای معکوس سازی بدافزارها و استخراج نام توابع از داخل کد اسمبلی آن‌ها از نرم‌افزار IDA Pro Disassembler استفاده شده است؛ همچنین ابزار Peid به منظور باز کردن مخرب‌هایی که نویسندگان آن‌ها از تکنیک‌های «بسته بندی» برای پیچیده سازی آن‌ها استفاده کرده‌اند، به کار گرفته شده است. در بخش پویا از ابزار محیط کنترل شده‌ی API MONITORING به منظور ایجاد محیط مجازی برای اجرای بدافزارها استفاده شده و در ادامه از الگوریتم طراحی شده پیشنهادی به منظور تشخیص مخرب یا خوش‌خیم بودن فایل، کمک گرفته شده است. در نهایت از تکنیک‌های داده‌کاوی و ابزار Weka به منظور بالا بردن سرعت تشخیص استفاده گردیده است.

کلید واژه‌ها: *امنیت، بدافزار، تشخیص بدافزار، تکنیک‌های فرار، داده‌کاوی، یادگیری ماشین*

۱- دانشجوی کارشناسی ارشد مهندسی نرم‌افزار دانشگاه آزاد واحد علوم و تحقیقات سیرجان، reza768@gmail.com
۲- مدیر گروه کارشناسی ارشد دانشگاه آزاد واحد علوم و تحقیقات سیرجان، noormandi_r@iausirjan.ac.ir

۱- مقدمه

به موازات تکامل نرم‌افزارهای خوش‌خیم، توسعه دهندگان آن، اقدامات امنیتی را پیاده سازی می‌کنند تا مطمئن شوند محصولاتشان از امنیت بالایی برخوردار است. با کامل شدن این نرم‌افزارها، نویسندگان مخرب‌ها[1] نیز سعی دارند تا با استفاده از تکنیک‌های چندریختی[2]، مبهم سازی[3]، دگردیسی[4] و... بدافزارهای به‌روزتری را تولید کنند که در دام ضد بدافزارها[5] قرار نگیرد. (Sain,2012:3)

اساس و پایه‌ی روش‌های قدیمی برای تشخیص مخرب‌ها، استفاده از روش مبتنی بر امضا[6] است که در این روش‌ها، قسمتی از بدافزار به‌عنوان یک امضا برای آن در نظر گرفته می‌شود و بدافزار توسط همین امضا شناسایی و تشخیص داده می‌شود. این امضاها در داخل یک پایگاه داده ذخیره می‌شوند (Ravi ,2012:2)

به دلیل عدم موفقیت روش‌های قدیمی در تشخیص و شناسایی بدافزارهای جدید و ناشناخته، در سال‌های اخیر محققان تلاش کرده‌اند تا با استفاده از بعضی ویژگی های تغییر ناپذیر بدافزارها، روش‌های مطمئن تری را برای تشخیص آنها ارائه دهند. نتایج به دست آمده نشان می‌دهد که روش‌های جدید دارای نرخ بالاتر تشخیص نسبت به روش‌های قدیمی تر هستند.

ما در این پژوهش قصد داریم روشی را برای تحلیل و دسته بندی فایل‌های قابل اجرا ارائه دهیم. روشی که ارائه شده است، تکنیک‌های داده‌کاوی[7] را استفاده می کند.

روش ارائه شده در این کار بر این اساس است که فراخوانی های Api call[8] در فایل‌های اجرایی می‌تواند اطلاعات مفیدی را در مورد فایل‌ها در اختیار ما قرار دهد. برخلاف تحقیقاتی که تاکنون در این زمینه انجام شده است، داده های این کار دارای وسعت بیشتری است به گونه ای که انواع متفاوتی از بدافزارها مانند ویروس‌ها[9] ,تروجان ها[10] , کرم‌ها[11] , جاسوس‌افزارها[12] , روت‌کیت ها[13] و ... را در بر گرفته است که مجموعاً شامل ۳۰۱۳۱ فایل مخرب و فایل خوش‌خیم بوده که این فایل‌های خوش‌خیم مجموعه‌ای از فایل‌های سیستم‌عامل ویندوز است. نرخ تشخیص در این روش حدود ۹۷ ٪ است که این نتیجه دارای نرخ تشخیص بالاتری نسبت به روش‌هایِ تاکنون استفاده شده است.

1. Malware
2. Polymorphism
3. Obfuscation
4. Metamorphosis
5. Anti Malware
6. Signature
7. Data mining
8. Application Program Interface
9. Virus
10. Trojan
11. Worm
12. Spyware
13. Rootkit

نتایج به دست آمده در این پژوهش نشان می‌دهد که روش ارائه شده، روشی قابل اعتماد در تشخیص بدافزارها است.

۲- بدافزار

نام دیگر نرم‌افزارهای ویران گر، بدافزار یا نرم‌افزارهای مخرب است. این بدافزارها برای آسیب رساندن به سیستم و شبکه‌های کامپیوتری در نظر گرفته شده‌اند. اولین ویروس در حدود سال‌های ۱۹۰۰ تشخیص داده شد که آن را کرم رایانه‌ای می‌نامیدند. پس از آن، انواع بدافزارهای دیگر توسط نویسندگان آن‌ها تولید شد که به مرور زمان با استفاده از ابزارها و تکنیک‌های «مبهم سازی» پیچیده‌تر شدند به گونه ای که امروزه تشخیص آن‌ها بسیار دشوار شده است. در واقع بدافزار شامل مجموعه ای از کدهای طراحی شده برای اجرای فعالیت‌های غیر قانونی است که موجب آسیب رساندن به سیستم می‌شود و بر یکپارچگی و عملکرد آن تأثیر منفی می‌گذارد (Egele,2012:4)

هنگامی که نرم‌افزارهای مخرب، راه خود را به درون سیستم شما می‌یابند، ابتدا سیستم شما را برای سنجیدن میزان آسیب پذیر بودن، اسکن کرده و سپس اعمال ناخواسته‌ای را برای پایین آوردن عملکرد سیستم انجام می‌دهند. بدافزارها قادر به آلوده کردن فایل‌های اجرایی، فایل‌های سیستمی، داده‌ها، درایوها، ایجاد بیش از حد ترافیک در شبکه و ... هستند که همگی منجر به محدودیت‌هایی در هنگام استفاده از خدمات سیستم می‌شوند (Ravi,2012:2)

با توجه به سابقه طولانی بدافزار، انواع متعددی از آن‌ها از ابتدا تاکنون به وجود آمده که در زیر به برخی از آن‌ها اشاره شده است.

ویروس

ویروس، قطعه کدی است که خود را به برنامه‌های دیگر از جمله سیستم‌عامل اضافه می‌کند. ویروس به تنهایی قادر به اجرا نیست؛ بلکه مستلزم اجرا در برنامه میزبان است. ویروس‌ها می‌توانند علاوه بر انتشار بر روی فایل‌های محلی، بر روی سرور نیز منتشرشده و بر فایل‌های مشترک موجود در سرور تأثیر گذاشته و به این ترتیب به سایر کامپیوترها سرایت می‌کنند. (Sain,2012:3)

کرم

یک کرم برنامه‌ی مخربی است که به‌طور مستقل عمل کرده و می‌تواند به عنوان یک نسخه کامل بر روی ماشین دیگر منتشر شود. کرم در محیط‌های شبکه‌ای مانند اینترنت به شدت رایج است. مهم‌ترین

خصیصه کرم، طبیعت خودتکرار شونده آن است که منجر به اشغال حافظه و استفاده بی جا از پهنای باند شبکه می‌شود. (Herath,2009:2)

اسب تراوا

در مفاهیم رایانه‌ای، اسب تراوا است که می‌تواند خرابی‌های زیادی را به بار آورد و یا اعمالی غیر از آنچه مد نظر کاربر است را انجام دهد. این اصطلاح به تازگی به برنامه های ویران گری گفته می‌شود که بدون اجازه و آگاهی کاربر، وارد سیستم شده و موجب از بین بردن یکپارچگی سیستم می‌شود (Ravi,2012:3)

بمب منطقی[1]

بمب منطقی خود را منتشر نمی‌کند، بلکه بر روی یک سیستم نصب می‌شود و منتظر می‌ماند تا زمانی که یک اتفاق خارجی مانند ورود داده، رسیدن به یک تاریخ خاص، ایجاد یا حذف و یا حتی ویرایش یک فایل خاص رخ دهد و آنگاه به سیستم آسیب می‌رساند (Doherty,2009:2)

درب‌های پشتی[2]

معمولاً برای دسترسی به یک سیستم کامپیوتری نیاز به وارد کردن نام کاربری و رمز عبور دارید. اگرچه این سطح از امنیت گاهی اوقات سیستم‌ها را ایمن می‌سازد و تنها افراد خاصی می‌توانند با استفاده از اطلاعات صحیح وارد سیستم شوند؛ اما وجود درب‌های پشتی باعث بی اثر کردن کلیه تنظیمات امنیتی شده و اجازه دسترسی کامپیوتر شما را به دیگر کاربران، از راه دور می‌دهد. (Herath,2009:2)

جاسوس

جاسوس‌افزارها برای به دست آوردن اطلاعات شما یا داده های روی رایانه شما مورد استفاده قرار می‌گیرند. این اطلاعات شامل اطلاعات شخصی، تاریخچه مرورگر، اسامی ورود، رمزهای عبور و شماره‌های کارت اعتباری است. (Sain,2012:3)

1. Logic bomb
2. Backdoors

روت کیت

یک بدافزار است که توانایی مخفی‌سازی خود و فعالیت‌هایش را در سیستم هدف دارد. مالک روت‌کیت قادر به اجرای فایل و انجام تنظیمات بر روی سیستم قربانی است، بدون آنکه مالک سیستم متوجه آن شود. این بدافزار معمولاً خود را به فایل‌های اصلیِ هستهٔ سیستم عامل می‌چسباند و با آن‌ها اجرا می‌شوند. روت‌کیت‌ها با هدف قراردادن ساختار و برنامه‌های اصلی سیستم‌عامل و دست‌کاری محتوای آن‌ها سعی در تغییر روند عملکرد و نتیجه اجرای آن‌ها دارد. روت‌کیت‌ها به روش‌های زیر می‌توانند خود را از دید کاربر پنهان کنند:

الف– روت‌کیت، کدهای خود را با کدهای سیستم‌عامل که در سطوح پایین هستند ادغام می‌کند و با این کار می‌تواند به تمام درخواست‌های سیستم مثل خواندن فایل‌ها، پردازش‌های در حال اجرا و ... دسترسی داشته باشد.

ب– در روش دیگر روت‌کیت، کدهای مخرب خود را به پردازش‌های سالم منتقل می‌کند و با انجام این کار می‌تواند از حافظه ای که در اختیار پردازش مورد نظر قرار گرفته، برای انجام فعالیت‌های مخرب خود استفاده کند. (Mathur,2013:5)

۳- تجزیه تحلیل و تشخیص بدافزار

ضد بدافزارهای موجود دائماً مورد چالش قرار می گیرند. چندین روش تجزیه و تحلیل نرم‌افزار مخرب و روش‌های تشخیص بدافزارها برای به حداقل رساندن توزیع برنامه های مخرب، پیشنهاد شده است. با این وجود نویسندگان بدافزار، تکنیک‌های جدیدی مانند مبهم سازی، تغییر رفتار برنامه و ... را به منظور ایجاد بدافزارهای جدید، بدافزارهای غیر قابل تشخیص و ... گسترش داده‌اند. (Tahan,2012:7)

ما به ارائه برخی از تکنیک‌های تشخیص بدافزارها می‌پردازیم. روش‌های تشخیص ارائه شده بر پایه‌ی راهبردهای تجزیه و تحلیل مختلفی است که در نرم‌افزارهای تجزیه و تحلیل[1]، رایج هستند و مهم‌ترین آن‌ها عبارتند از: استاتیک، پویا و ترکیبی.

استاتیک[2]

تجزیه و تحلیل استاتیک، اطلاعاتی در مورد کنترل برنامه، جریان اطلاعات و ویژگی‌های آماری بدون نیاز به اجرای برنامه به‌دست می‌آورد. روش اصلی مورد استفاده در تجزیه و تحلیل ایستا، مهندسی معکوس[3]

1. Analysis
2. Static
3. Reverse Engineering

است. یکی از مشکلات پیش روی تجزیه و تحلیل ایستا این است که کد منبع برنامه معمولاً در دسترس نیست که این امر استفاده از تکنیک‌های تجزیه و تحلیل ایستا را کاهش می‌دهد و در نتیجه منجر به تجزیه و تحلیل کدهای باینری آن‌ها می‌شود که این تجزیه و تحلیل هم به شدت پیچیده است. (Tahan,2012:9)

در روش ایستا که همراه با مهندسی معکوس است، کدهای باینری چک می‌شوند و ویروس‌ها بر اساس کدهای باینری شناسایی می‌شوند که در واقع جزو کلیدی در روش ایستا است. استخراج کدهای باینری کار نسبتاً دشواری است.

پویا[1]

روش تجزیه و تحلیل پویا نیاز به اجرای برنامه دارد تا بتواند آن را در یک محیط مجازی تحلیل کند. این روش، اطلاعاتی را در رابطه با کنترل و جریان داده به ما می‌دهد که موجب کسب نگاهی عمیق‌تر از برنامه می شود. (Gurrutxaga,2008)

CWSand-box و TTAnalyze ابزار های معروف تجزیه و تحلیل پویا هستند که برای تحلیل بدافزارها توسعه یافته‌اند.

ترکیبی

تجزیه و تحلیل ترکیبی شامل ترکیبی از روش‌ها یعنی ایستا و پویا است. در این روش ابتدا ویژگی‌های امضا، تحلیل می شود سپس آن را با پارامتر های رفتاری ترکیب کرده تا تجزیه و تحلیل را تقویت نماید.

با توجه به این روش، تحلیل ترکیبی می‌تواند باعث بهبود درک از رفتار بدافزارها شود و در نتیجه نرخ مثبت کاذب[2] را کاهش دهد؛ همچنین روش ترکیبی بر محدودیت‌های تجزیه و تحلیل پویا غلبه می‌کند زیرا یکی از اشکالات مهم در تجزیه تحلیل پویا، کند بودن این روش است.

امروزه تشخیص بدافزارها مسئله مهمی برای کاربران کامپیوتر است و همواره یکی از مسائل مهم در خصوص امنیت اطلاعات نیز به شمار می‌آید. (Damodhare,2012:4) روش‌های مختلفی برای تشخیص بدافزار وجود دارد اما با توجه به پیچیده‌تر شدن بدافزارها توسط تکنیک‌های مبهم سازی، نیاز به تکنیک‌های پیشرفته تری برای تشخیص آن‌ها است. تمامی روش‌های تشخیص بدافزار در یکی از دسته‌های زیر قرار می‌گیرند:

1. Dynamic
2. False positive

۱- مبتنی بر امضا[1]
۲- مبتنی بر رفتار[2]

تشخیص بر اساس امضا

روش مبتنی بر امضا بر اساس بررسی کدهای مشکوک و جمع آوری اطلاعات به منظور توصیف اهداف مغرضانه ی نرم‌افزارهای مخرب است. هدف اصلی این روش، استخراج توالی بایت‌های ویژه از کدها به عنوان امضا است؛ همچنین جستجو برای یک امضا در داخل فایل‌های مشکوک یکی از اهداف این روش است. بیشتر ضد بدافزارهای تجاریِ امروزی از مجموعه ای از امضاها به منظور تشخیص برنامه های مخرب استفاده می‌کنند که این کدهای مشکوک با یک توالی منحصر به فرد از ساختارهای برنامه یا بایت‌ها، مقایسه می‌شود.

اگر این امضا در داخل «پایگاه داده» وجود نداشته باشد، به این معناست که این فایل خوش‌خیم است نه مخرب؛ بنابراین یکی از محدودیت‌ها در روش تشخیص مبتنی بر امضا، نیاز به دخالت انسان در به روز رسانی پایگاه داده امضا، با امضاهای جدید است. علاوه بر این، گروهی از محققان نشان دادند که برخی از نویسندگان بدافزارهای چندریختی می‌توانند به آسانی روش مبتنی بر امضا را به وسیله تکنیک‌های مبهم سازی، شکست دهند. این امر باعث شد ما به این نتیجه برسیم که این روش تشخیص، مستعد ابتلا به منفی کاذب[3] است. همچنین هنگامی که انواعی از مخرب‌ها شناخته می‌شوند، پایگاه داده ی امضاها رشد می‌کند. (Gurrutxaga,2008)

تشخیص بر اساس رفتار

برخلاف روش ایستا که بر روی کد بدافزارها تکیه می‌کند، رفتار زمان اجرا را مورد توجه قرار می دهد. در واقع تجزیه و تحلیل یک برنامه در زمان اجرای آن را تجزیه و تحلیل پویا می‌نامند که به تجزیه و تحلیل رفتارها نیز معروف است و شامل اجرای نرم‌افزار و مشاهده رفتار آن، تعامل سیستم و اثرات آن روی سیستم میزبان است. روش تجزیه و تحلیل پویا نیاز به اجرای فایل‌های آلوده در یک محیط مجازی؛ مانند یک ماشین مجازی، یک شبیه ساز[4]، sand box و ... دارد تا بتواند آن را آنالیز کند. (Damodhare,2012:8)

1. Signature-based Detection Techniques
2. Anomaly-based Detection Techniques
3. False negative
4. Simulator

٤- کارهای مرتبط

در دهه های اخیر، روش‌های جدیدی مبتنی بر امضا به وجود آمدند؛ اما یکی از عمده‌ترین ضعف‌های این روش‌ها، ناتوانی آن‌ها در کشف و تشخیص بدافزارهایی بود که نویسندگان آن‌ها از تکنیک‌های مبهم‌سازی در نوشتن آن استفاده می‌کردند. به همین دلیل پژوهش‌های زیادی برای بهبود روش‌های مبتنی بر امضا ارائه شده که یکی از بهترین روش‌ها پروژه ای با نام save بود که تمرکز آن بر روی اندازه گیری میزان شباهت و تفاوت‌ها بین کدهای مخرب شناخته شده و کدهای مشکوک بود. در این روش امضای بدافزارها توسط دنباله ای از Api Callها تعیین می‌شود. اما این روش همچنان در شناسایی و تشخیص بدافزارهای جدید و چند شکلی، ناتوان بود. این پروژه شامل دو عیب اساسی بود:

الف- ابزار مورد استفاده در این روش یعنی W32Dasmversion8.9 ضعیف بود.

ب- این روش هنگامی که نویسنده ی مخرب از تکنیک‌های مبهم سازی مانند درج کد مرده[1] یا ... استفاده کند، دچار مشکل می‌شود؛ زیرا ترتیب apicall ها عوض می‌شود. (Patel,2008:6)

یکی دیگر از راه حل‌های ایستا که در اواخر سال۲۰۱۲ توسط GilTahan و همکارانش ارائه شد، برای تشخیص بدافزارها از امضا استفاده می‌کرد. اساس این روش تجزیه و تحلیل بخش مشترک بین بدافزارها بود. روش آنها که Mal-ID نام دارد قادر به جدا سازی فایل‌های مخرب از فایل‌های خوش‌خیم است. این الگوریتم که بر پایه الگوریتم‌های یادگیری ماشین[2] است، ابتدا فایل اجرایی را به فایل دودویی تبدیل کرده و سپس آن را به زیر بخش‌هایی از بایت‌ها تقسیم می‌کند و در نهایت هر یک از این زیر بخش‌ها به عنوان یک داده یا یک قطعه کد، دسته بندی می‌شوند که از این قطعه کدها به عنوان امضای بدافزار استفاده می‌شد و در تشخیص فایل‌های مخرب می‌توان از آن‌ها بهره برد. عیب اساسی این روش، ضعف و ناتوانی آن در تشخیص بدافزارهای جدید بود که از تکنیک‌های چند شکلی و مبهم سازی استفاده می کردند. (Tahan,2012)

بعد از چندی، فردی به نام Ramamoorthy با همکاری گروهی دیگر روشی را با نام MEDiC به وجود آوردند. در این روش برخلاف روش save که از Apicall ها به عنوان ویژگی مهم در تشخیص بدافزارها بهره برده بود، Assemblycall ها را به عنوان ویژگی مهم در نظر گرفتند. Assemblycall ها نسبت به Apicall ها برتری داشتند؛ زیرا در هنگام مقایسه‌ی فایل‌های اجرایی، به جزئیات بیشتری

1. Dead Code
2. Machine Learning

توجه می شد اما نمی‌توان گفت از Apicall ها بهتر هستند؛ زیرا دارای سرعت کمتری بودند و همین دلیل باعث شد در مواقعی که مدت زمان انجام کار مهم است Apicall ها موفق تر باشند. (Veramani,2012)

گروهی دیگر روشی را برای تشخیص بدافزارها با استفاده از Apicall ها پیشنهاد دادند. هدف آن‌ها بهبود تشخیص بدافزارها و کاهش نرخ مثبت کاذب با استفاده از تولید توالی Apicall ها با نام Apicall-Grams بود. در نهایت الگوریتم‌های یادگیری ماشین را بر روی داده های به دست آمده که همان Apicall-Grams ها بود اعمال کردند. آن‌ها در روش خود، مدل جنگل تصادفی[1] را به کار بردند زیرا نرخ هشدار غلط در این روش کمتر از بقیه بود. آن‌ها در آخرین آزمایش های خود توانستند ۹۷.۵۳٪ از بدافزارهای شناخته شده را تشخیص دهند اما همچنان در تشخیص بدافزارهای جدید و ناشناخته ناتوان بودند. (Hashemi,2011)

یکی دیگر از پژوهشگران به نام Dragos با همکاری گروهی دیگر، روشی را معرفی کردند که الگوریتم‌های یادگیری ماشین را به کار بردند. اشکال روش آن‌ها این بود که تعداد فایل‌های تمیز، چندین برابر فایل‌های مخرب بود؛ زیرا هدف اصلی این روش تشخیص نرم‌افزارهای مخرب با نرخ مثبت کاذب پائین بود. این روش که بر پایه‌ی پروسپترون های یک طرفه و پروسپترون های کرنالایز شده است، دارای یک عیب اساسی بوده که از آن می‌توان به نامتعادل بودن تعداد فایل‌های مخرب و خوش‌خیم نام برد. با آنکه در این روش، «نرخ مثبت کاذب» کاهش یافته، اما این نامتعادل بودن در زمانی که تعداد فایل‌ها، کمی افزایش یابد با مشکل رو به رو می‌شود. شایان ذکر است که روش‌های تشخیص بدافزار در قالب یادگیری ماشین نمی‌توانند جایگزین روش‌های استاندارد تشخیص بدافزارها با استفاده از محصولات ضد بدافزاری شوند اما می‌توانند برای بهبود عملکرد به آن‌ها اضافه شوند. (Dragos,2009)

در روشی دیگر، محققی به نام Shultzl به همراه گروهی دیگر روشی را معرفی کردند که در آن از تکنیک‌های داده‌کاوی در جهت تشخیص بدافزارهای جدید و ناشناخته استفاده شده است. آنها ابتدا فهرست توابع Dll[2] فراخوانی شده و تعدادی از توابع سیستمی فراخوانی شده در داخل هر Dll را استخراج کردند، سپس از یک الگوریتم که به Ripper معروف است برای یافتن الگوها از داخل اطلاعات Dll ها استفاده کردند. بالاترین دقت عملکرد در این روش ۸۳.۱۱٪ است. نویسنده، روش خود را با روش‌های قدیمی بر پایه‌ی امضا مقایسه کرده است و ادعا می‌کند که روش‌های تشخیص مبتنی بر داده‌کاوی دارای دقتی دو برابر بالاتر از روش‌های ساده‌ی تشخیص مبتنی بر امضا هستند. گرچه این روش خوب است اما هنوز هم دارای دقت بالایی نیست. (Baldangombo,2013)

1. Random Forest
2. Dynamic Link Library

در کاری که اخیراً توسط YE انجام شده است، سیستمی با نام IMDS تولید شده که در آن از الگـوی فراخوانی‌های سیستمی استفاده شده است. که بر روی این الگوها عمل داده‌کاوی صورت گرفته اسـت. ایـن پژوهش شامل ۱۲۲۱۴ فایل سالم و ۱۷۳۶۶ فایل مخرب است که آن‌هـا تنهـا از ۲۰۰۰ فایـل بـرای تسـت سیستم استفاده کرده‌اند. اگرچه نرخ یادگیری و دقت در این روش تقریباً خوب است؛ اما دو مشـکل اساسـی نیز وجود دارد که عبارتند از:

الف– تعداد فایل‌های دخیل در این تحقیق کم است در حالی که بـرای ایـن تحقیـق در حـدود ۳۰۱۳۱ فایل در نظر گرفته شده است.

ب– مشکل دوم، نامتعادل بودن داده های تست در مقابـل متعـادل بـودن داده هـای یـادگیری اسـت. (Damodhare,2012:5)

۵– روش انجام تحقیق

معرفی روش:

در تحقیق حاضر برخلاف تحقیقات موجود، اقدامات زیر به منظور افـزایش صـحت و کـارایی الگـوریتم پیشنهادی و رفع نقایص روش‌های بالا صورت گرفته است که عبارتند از:

الف– استفاده از نام توابع سیستمی فراخوانی شده در کد اجرایی بـدافزارها بـه عنـوان ویژگـی مهـم در هنگام شناسایی آن‌ها

ب– استفاده از ابزار های مهندسی معکوس به منظور تحلیل ایستای بدافزارها و اسـتخراج نـام توابـع از داخل کد اسمبلی آن‌ها

ج– استفاده از ابزار محیط کنترل شده[1] به منظور ایجاد محیط مجازی برای اجرای بدافزارها

د– استفاده از ابزار Peid به منظور باز کردن مخرب‌هایی که نویسـندگان آن‌هـا از تکنیک‌هـای بسـته بندی به منظور پیچیده سازی آن‌ها استفاده کرده بودند

ه– استفاده از تکنیک‌های داده‌کاوی به منظور بالا بردن سرعت تشخیص

کاری که ما در این پژوهش انجام می‌دهیم شامل یـک مجموعـه داده بسـیار وسـیع اسـت کـه انـواع گوناگونی از نرم‌افزارهای خوش‌خیم و مخرب را در بر می‌گیرد که در کل تعداد فراخوانی‌های استخراج شـده از آن در حدود ۴۴۰۰۰ نام متفاوت از ۳۰۱۳۱ فایل مختلف از ۸۹۰ کتابخانه است که شامل انواع متفاوتی از

1. Controlled Environment

مخرب‌ها از خانواده هـای Spoofer, Sniffer, Flooder, Exploit, Worm, Backdoor, Trojan و virus است.

نمای کلی سیستم:

چارچوب ارائه شده در این تحقیق بر روی فایل‌های اجرایی قابل حمل[1] متمرکز شده اسـت کـه نمـای کلی آن به شرح زیر است:

۱- جمع آوری داده‌ها، مجموعه ای از فایل‌های اجرایی که شامل فایل‌های مخرب و خوش‌خیم هستند را جمع آوری می‌کنیم.

۲- پردازش داده‌ها، از ابزار دیس اسمبلر[2] برای تحلیل در بخش ایستا و ابزار مانیتورینگ بـرای تحلیـل در بخش پویا استفاده شده است. این فایل‌های اجرایی جمع آوری شده را در قالـب فایـل ورودی، بـه ابـزار دیس اسمبلر داده تا کد اسمبلی این فایل‌ها را به دست آورد و در انتها لیست توابع سیستمی فراخوانی شـده در آن‌ها بر گردانده می شود؛ همچنین آن‌ها را در داخل ابزار پویای خود نیز اجرا می‌کنیم تا هماننـد تحلیـل بخش ایستا، فهرست توابع سیستمی را به دست آوریم.

۳- اعمال الگوریتم طراحی شده برای تشخیص بدافزارها: الگوریتمی کـه بـرای تشـخیص مخـرب یـا خوش‌خیم بودن فایل در نظر گرفته شده است را بر روی این فایل ذخیره شده در قالب Xml[3] اعمال کـرده و اطلاعات آن را در بانک اطلاعاتی خود ذخیره می‌کنیم.

۴- ابزارهای داده‌کاوی: با استفاده از ابزار های داده‌کـاوی و بانـک اطلاعـاتی بـه دسـت آمـده توسـط الگوریتمی که طراحی کرده‌ایم، نرخ موفقیت در تشخیص بدافزارها را به دست می‌آوریم.

این پژوهش با انجام چند گام اصلی حاصل شده است:

- جمع‌آوری داده‌ها
- پردازش داده‌ها
- تحلیل نتایج

در ادامه، به بررسی هر یک از این مراحل می‌پردازیم.

۱ – ۵: جمع‌آوری داده‌ها

به منظور گردآوری فایل‌های اجرایی بدافزارها، مجموعه ای از ۳۰۱۳۱ فایل اجرایی مخرب را از سـایت http://vxheaven.org جمع آوری کرده‌ایم که بخش عمده ای از آن شامل روت‌کیت ها است. به دلیل

1. Portable
2. Disassembler
3. eXtensible Markup Language

اینکه هدف ما در این مقاله نرخ تشخیص بدافزارها به ویژه روت‌کیت ها است، این مجموعه بدافزار را انتخاب کردیم.

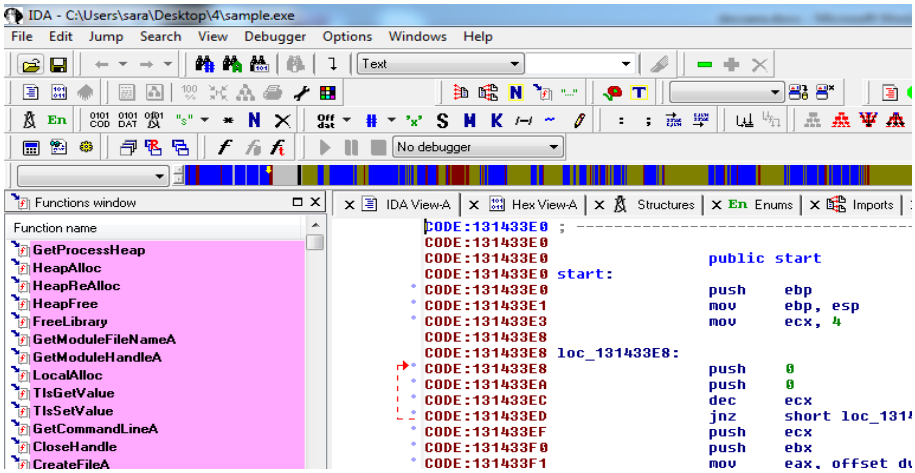
۲– ۵: پردازش و آماده‌سازی داده‌ها

همان طور که در بالا هم ذکر شد نویسندگان مخرب از تکنیک‌های مبهم سازی، چند شکلی، دگردیسی و ... استفاده کردند تا بتوانند اعمالی را بر روی مخرب‌ها پیاده سازی کنند تا آن‌ها توسط ضد بدافزارهای کنونی قابل تشخیص نباشند. برخی از این تکنیک‌ها به گونه ای است که بدافزار را در هنگام تحلیل حمایت می‌کنند و برخی دیگر برعکس، بدافزار را در هنگام تشخیص، به صورت پویا محافظت می‌کنند به گونه ای که قابل شناسایی نباشند.

البته گاهی اوقات این تکنیک‌ها به گونه ای هستند که بدافزار می‌تواند هم در برابر تشخیص به کمک تحلیل ایستا و هم تحلیل پویا مقابله کند برای همین ما از دو ابزار معروف و کارآمد دیس اسمبلر IDA PRO و API MONITORING و همچنین ابزار ضد بسته‌بندی Peid به پردازش داده‌ها می‌پردازیم.

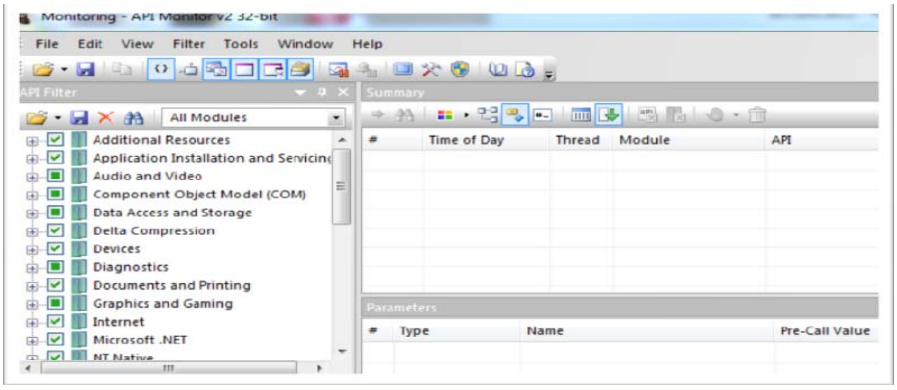
ابتدا باید با ابزار Peid داده که همان فایل اجرایی بدافزار است را پردازش کرده البته با این شرط که آن فایل توسط ابزار بسته بندی Packing شده باشد در غیر این صورت نیازی به اعمال این ابزار بر روی آن نیست. در واقع با عمل Unpacking اگر بر روی فایل، عمل بسته بندی انجام شده باشد، حذف می‌شود؛ چون در غیر این صورت این فایل توسط ابزارهای مهندسی معکوس در تحلیل ایستا و ابزار های تحلیل پویا قابل اجرا نیست و در نتیجه قادر به دسترسی توابع سیستمی فراخوانی شده در آن نخواهیم بود.

سپس تک تک فایل‌ها را به عنوان ورودی به دیس اسمبلر ذکر شده در بالا داده تا ابتدا کد اسمبلی فایل اجرایی بدافزار را به دست آوریم و در نهایت به وسیله‌ی این کدها توابع سیستمی فراخوانی شده در فایل را به دست آوریم. تجزیه و تحلیل با زبان‌های سطح پایین نسبت به زبان‌های سطح بالا، اطلاعات بیشتری را به ما نشان می‌دهد؛ همچنین با این ابزار به راحتی می‌توانیم گراف جریان کنترل بدافزارها را به دست آوریم. در این مرحله از کار با استفاده از زبان اسمبلی اطلاعات مفید هر بدافزار را به دست می‌آوریم به گونه ای که می‌توانیم درک خوبی از رفتار های آن داشته باشیم.



شکل ۱ـ نمای کلی از نرم‌افزار Ida Pro Disassembler

اکنون فایل را به عنوان ورودی به ابزار API MONITORING داده و لیست توابع سیستمی فراخوانی شده در فایل که توسط این ابزار تشخیص داده شده‌اند را به دست می‌آوریم.



شکل ۲ـ نمای کلی از نرم‌افزار مانیتورینگ

پس از آنکه این عمل بر روی تمامی فایل‌ها انجام شد، دارای یک لیست کامل از API CALL های فراخوانی شده در تمامی فایل‌ها هستیم که توسط ابزار ایستا و پویای ذکر شده، تشخیص داده شده است. با انجام این کار مطمئن می‌شویم که اگر فایل در مقابل تشخیص ایستا و یا تحلیل پویا مبهم شده باشد باز هم قابل تشخیص است و می‌توانیم لیست توابع سیستمی آن را به دست آوریم.

پس از استخراج کد اسمبلی، برنامه ها و لیست APIcall های فراخوانی شده در آن‌ها، لیست توابع آن که در فرمت txt است را با استفاده از ابزار Stylus Studio Now به فرمت xml تبدیل می‌کنیم؛ زیرا این فرمت برای ذخیره‌سازی داده‌های نیمه ساختار یافته، بهترین گزینه است. پس از اینکه مراحل بالا را بر روی تمامی فایل‌های بانک بدافزار خود اعمال کردیم، دارای یک مجموعه xml هستیم که هر کدام حاوی نام یک سری APIcall است. حال ابتدا نام تمامی APIcall ها را از تمام فایل‌ها به دست آورده و در یک جدول از بانک اطلاعاتی خود ذخیره می‌کنیم.

اکنون باید با استفاده از الگوریتمی که نوشته ایم چک کنیم که در هر یک از فایل‌هایxml – که گزارشی از هر فایل بدافزار است – کدام یک ازapicall ها فراخوانی شده، و در جدولی که با هدف این کار تهیه گشته، برای نام آنapicall فراخوانی شده، ارزش ۱ و برای apicall فراخوانی نشده ارزش ۰ را درج کنیم. ما در بانک اطلاعاتی خود برای این کار یک جدول در نظر گرفته‌ایم. ساختار جدولی که برای این کار در نظر گرفته شده است دارای ۵۰۰۲ فیلد بوده که فیلد اول به شماره فایل xml –که منظور نام بدافزار است– فیلد دوم به نوع بدافزار و ۵۰۰۰ فیلد دیگر به نام apicall ها اشاره دارد. این الگوریتم با زبان php که یک زبان منبع باز[1] است و پایگاه داده‌ی MySql تهیه شده است، الگوریتم را به صورت یک نرم‌افزار کوچک تهیه کرده تا کلاینت بتواند به راحتی از آن استفاده نماید.

به منظور ارزیابی الگوریتم و روش پیشنهادی، از تکنیک‌های داده‌کاوی و با اعمال آن‌ها بر روی نتایج به دست آمده در مرحله قبل، تشخیص می دهیم که آیا فایل بدافزار، یک روت‌کیت است یا خیر و در انتها با استفاده از همین ابزار داده‌کاوی (ابزار داده‌کاوی Weka) درصد موفقیت خود را در تشخیص بدافزارها به دست آوریم. بدین منظور از جداول پایگاه داده‌ی خود خروجی xlsx گرفته تا از آن‌ها به عنوان ورودی برای فرایند داده‌کاوی استفاده نماییم.

۳– ۵ : نتایج

بدافزارهای موجود در یک دسته، عموماً دارای الگوهای عمومی مشترکی هستند، مثلاً یک تعداد از اسامی توابع سیستمی در تمام اعضای این خانواده مشترک می‌باشند.

هدف ما تجزیه و تحلیل و تشخیص بدافزار روت‌کیت، به وسیله‌ی بررسی این الگوهای مشترک با استفاده از تکنیک‌های یادگیری ماشین در میان بدافزارها است.

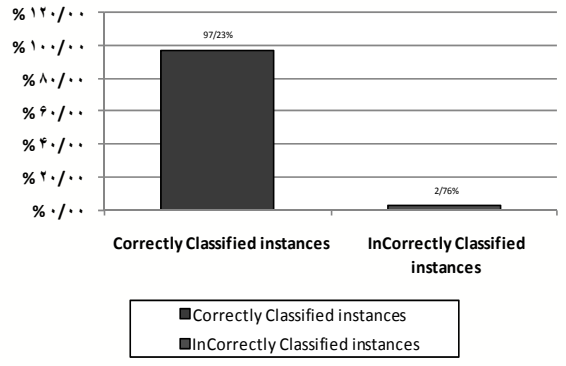
1. Open Source

در واقع هدف ما این است که برای رفع محدودیت روش های مبتنی بر امضـاهای سـنتی و مقابلـه بـا تکنیک‌هایی که نویسندگان مخرب‌ها از آن‌ها استفاده می‌کنند و همچنـین بـالا رفـتن نـرخ کشـف بـدافزار روت‌کیت از ApiCall های فراخوانی شده در بدافزارها استفاده کنیم.

این روش که بر اساس توابع سیستمی فراخوانی شده در کد اجرایی بدافزار است، برای به دسـت آوردن لیست توابع سیستمی از ابزار مهندسی معکوس به منظور تحلیل ایستا و ابزار مانیتورینگ به منظـور تحلیـل پویا استفاده می‌کند. یعنی با دیس اسمبل کردن کد بدافزارها، ابتدا کد اسـمبلی آن‌هـا را بـه دسـت آورده و سپس توابع سیستمی فراخوانی شده در آن را استخراج کرده و همچنین با استفاده از مانیتورینگ فایـل نیـز لیست API CALL های موجود در فایل اجرایی بدافزار را به دست می‌آورد. در نهایت با توجه بـه دنبالـه ای از این توابع – که در بدافزار روت‌کیت، عمومیت داشته – به تشخیص مخرب‌های روت‌کیت می‌پردازیم.

از فواید این روش می‌توان به درصد موفقیت بالای آن در تشخیص بدافزارها اشـاره کـرد؛ زیـرا بـه‌طور مستقیم با کدهای باینری بدافزارها در ارتباط است و همچنین نیاز به اجرای آن‌ها نیست و تنها با استفاده از کد آن‌ها و به دست آوردن دنباله‌ی مشترک توابع سیستمی فراخوانی شده، روت‌کیت بودن یا نبودن بـدافزار را تشخیص می دهد. بعد از ورود فایل log بـدافزارها، بـر روی الگـوریتم تهیـه شـده، پایگـاه داده خـود را تشکیل می دهیم، سپس اطلاعات این پایگاه را به یک ابـزار داده‌کـاوی –کـه مـا از ابـزار weka اسـتفاده کرده‌ایم– داده تا درصد موفقیت عمل تشخیص را به دست آوریم.

در زیر نموداری از نتایج عملیات داده‌کاوی با استفاده از ابزار Weka بر روی پایگاه داده به دست آمده نشان داده شده است.



شکل ۳ـ درصد موفقیت عمل تشخیص روت‌کیت

همان‌طور که در بالا مشاهده می‌کنید نرخ موفقیت این روش در تشخیص روت‌کیت بیش از ۹۷٪ است که نرخ چشمگیری است.

۴-۵ : مقایسه نتایج با سایر تحقیقات

در این بخش سعی بر این است که الگوریتم پیشنهادی از نظر تکنیکی با سایر الگوریتم‌های پیشنهادی تحقیقات دیگر، مقایسه شود که شامل موارد زیر است:

الف- در این روش از فراخوانی های سیستمی (ApiCall ها) استفاده شده است، شبیه روش هایSave و YE؛ با این تفاوت که در این روش ها اگر نویسنده ی مخرب از تکنیک‌های مبهم سازی مانند درج کد مرده یا ... استفاده کند، دچار مشکل می‌شود؛ زیرا ترتیب ApiCall ها عوض می‌شود اما در روش پیشنهادی ما، برای حل این مشکل، از گراف جریان کنترلی بدافزارها استفاده شده است که با استخراج این گراف، لیست توابع سیستمی فراخوانی شده، به دست آمده است. گراف کنترل جریان، یک نمایش از برنامه با استفاده از نمادهای گراف است که تمام مسیرهای ممکن را که در طول اجرای برنامه ممکن است پیمایش شود، نمایش می‌دهد.

ب- ابزار مهندسی معکوس استفاده شده در روش ما، IDAProDisassembler است که نسبت به ابزارهای HDasm و W32DSM89 در روش های مشابه ذکر شده، دارای قدرت بیشتری برای درک برنامه ها و شناخت ApiCall های فراخوانی شده است.

ج- نکته قابل تأمل روش پیشنهادی ما که در پژوهش های آتی جای توسعه و رشد دارد، این است که روشی به منظور معکوس سازی بدافزارها به صورت کاملاً خودکار -که بتواند تمام جزئیات بدافزار را در اختیار ما قرار دهد- ارائه گردد؛ زیرا ابزار IDAProDisassembler که در روش ما استفاده شده و ابزارهای دیگر شبیه HDasm و W32DSM89 که در روش های دیگر به کار رفته، ابزارهای دستی هستند.

٦- نتیجه‌گیری

امروزه فناوری تشخیص بدافزار، یک حوزه جدید محسوب شده و همواره یکی از مسائل مهم در خصوص امنیت اطلاعات به شمار می‌آید. در اینجا روش‌های پیشرفته تری به منظور تشخیص هوشمندی بدافزارها مورد توجه قرار گرفته است. این پژوهش ضمن مرور اجمالی بر مفاهیم اساسی مانند تشخیص و ارزیابی بدافزار و تکنیک‌های یادگیری، با مقایسه چند روش مختلف تشخیص و ارزیابی خودکار بدافزار، به استخراج ویژگی‌های مشترک روش‌های مذکور می‌پردازد.

چندین روش تشخیص بدافزار وجود دارد که از تکنیک‌های ایستا، پویا، الگوریتم‌های خوشه بندی[1] و یادگیری استفاده می‌کنند با این حال این الگوریتم‌ها معمولاً دارای نرخ پایینی در تشخیص هستند و یا دقت آن‌ها پایین است.

در این پژوهش ما پایگاه داده ای را بر اساس کاوش فراخوانی توابع سیستمی و با هدف تشخیص روت‌کیت ها ساخته‌ایم که شامل تحلیل فایل‌های اجرایی و شناسایی توابع سیستمی فراخوانی شده در این فایل‌ها است و توسط آن قادر به تشخیص روت‌کیت بودن یا نبودن یک بدافزار خواهیم بود.

مجموعه داده بزرگی که برای این کار در نظر گرفته شده است شامل انواع برنامه های مخرب دسته بندی شده و فایل‌های خوش‌خیم است که در آن، دامنه وسیع‌تری از فراخوانی‌های Api را نسبت به کارهای مرتبط دیگر مد نظر قرار دادیم و نرخ تشخیص بدافزار روت‌کیت را در میان دیگر فایل‌ها از جمله فایل‌های خوش‌خیم و یا انواع بدافزارهای دیگر به‌طور چشمگیری بالا برده‌ایم.

1. Clustering

کتابنامه

C. Ravi, R. M, Malware Detection using Windows Api Sequence and Machine Learning. International Journal of Computer Applications, (2012).

M. Sain, A.P,H.L. Survey on malware evasion techniques: state of the art and challenges, (2012).

Egele, M. S,A Survey on Automated Dynamic Malware-Analysis. ACM Comput er, 42, (2012).

Herath, H. M. P. S., & Wijayanayake, W. M. J. I. (2009). Computer Misuse in the Workplace. Journal

of Business Continuity & Emergency Planning, Vol.3, No.3, P.P 259–270.

K. Mathur, S.H. ,A Survey on Techniques in Detection and Analyzing Malware Executables, Advanced Research in Computer Science and Software Engineering, (2013).

Doherty, N. F., Anastasakis, L., & Fulford, H. (2009). The Information Security Policy Unpacked: A Critical Study of the Content of University Policies. International Journal of Information Management, Vol.29, No.6, P.P 449–457.

G. Tahan,L.R.Y. Automatic Malware Detection Using Common Segment Analysis and Meta-Features. Journal of Machine Learning Research 13 (2012) 949-979, (2012).

I. Gurrutxaga,O.A.J.P,J.M,J.M,I.P, Evaluation of Malware clustering based on its dynamic behaviour. Seventh Australasian Data Mining, (2008). Conference.

S. Damodhare, P.G, INTELLIGENT MALWARE DETECTION SYSTEM, International Journal of Engineering Research and Applications (IJERA), (2012).

Patel, S. C., Graham, J. H., & Ralston, P. A. (2008). Qualitatively Assessing the Vulnerability of Critical Information Systems: A New Method for Evaluating Security Eenhancements. International Journal of Information Management, Vol.28, No.6. P.P 483–491.