مدل‌سازی رفتاری مصرف منابع در بخش رمزنگاری فایل‌ها در باج‌افزارها

مهران گرمه‌ء[۱]، رجبعلی سجادیان فر[۲]، محمد شاه‌پسندی[۲]

[۱]استادیار گروه مهندسی کامپیوتر دانشگاه بجنورد
m.garme@ub.ac.ir
[۲]دانشجوی کارشناسی ارشد مؤسسه آموزش عالی اشراق
{sajadianfar,m.shahpasandi}@ub.ac.ir

چکیده

امروزه باج‌افزارها بدترین چالش مدیران و مسئولین فناوری اطلاعات سازمان‌ها هستند. سازندگان محصولات ضدویروس و تحلیل‌گران بدافزار دشواری‌های زیادی برای شناسایی و کالبدشکافی یک بدافزار پیچیده متحمل می‌شوند. یکی از نکات کلیدی موجود در این فرآیند، سرعت تشخیص و پاسخگویی به یک حمله باج‌افزاری می باشد که این امر گاهی سرنوشت‌ساز بوده و علاوه بر مقدار داده از دست رفته و سرعت انتشار باج‌افزار، بر امکان رمزگشایی اطلاعات نیز تأثیرگذار خواهد بود. با توجه اینکه امکان دور زدن تکنیک‌های سنتی شناسایی همواره وجود دارد، استفاده از روش‌های نوین تشخیص باج‌افزار کمک شایانی به مدافعین این حوزه خواهد کرد. این پژوهش با رویکرد بررسی میزان مصرف منابع سیستم (پردازنده، حافظه و دیسک) نمونه‌هایی از پنج خانواده از باج‌افزارها با تاریخ انتشار حداکثر سه سال گذشته انجام گرفته است که در نهایت با تحلیل نتایج به‌دست آمده، رفتار باج‌افزارها نسبت به نحوه مصرف منابع، مدل‌سازی و گزارش شده است. نتایج به‌دست آمده از این تحقیق نشان می‌دهد که ساختار کد و الگوی رمزنگاری در بین باج‌افزارهای هم‌خانواده تشابهات زیادی دارد. بنابراین با استفاده از مدل به‌دست آمده خواهیم توانست سایر باج‌افزارهای یک خانواده را شناسایی کرده و دقت و سرعت تشخیص را بمراتب بالاتر ببریم.

کلمات کلیدی: Ransomware، Detection، Responding، Obfuscation.

۱ مقدمه

باج‌افزار[۱]، اصطلاحی کلی برای توصیف نوعی بدافزار است که هدف آن باج‌گیری دیجیتالی از قربانیان با ایجاد محدودیت سیستمی می‌باشد؛ این تهدید به محل جغرافیایی یا نوع سیستم محدود نیست و می‌تواند علیه هر

[1]Ransomware

دستگاهی اتفاق بیفتد. هر یک از انواع سیستم‌عامل‌ها، از اندروید گرفته تا سیستم‌های iOS و ویندوز، همه در معرض خطر این نوع سوء استفاده قرار دارند.

باج‌افزارهای امروزی با درس گرفتن از گذشته، برای اطمینان یافتن از اینکه قربانی با احتمال بیشتری به خواسته آن‌ها تن دهد، از روش‌های پیشرفته رمزنگاری و با اتکا بر کلیدهای یکتا به ازای هر حمله، بهره می‌برند. با این شگرد، مهاجمین می‌توانند اطمینان داشته باشند که نه تنها گروه امداد راهی برای یافتن کلید رمزگشایی نخواهند یافت، بلکه کلید رمزگشایی یک حمله، در سایر حملات کاربردی نخواهد داشت.

توجه به این نکته نیز ضروری است که در اغلب موارد، اقدامات تخریبی با استفاده از قابلیت‌ها و منابع سیستمی موجود در دستگاه قربانی انجام می‌شود و تنها محدود به کارایی و امکانات پردازشی همان دستگاه خواهد بود. در این میان، بسیاری از رفتارهای مخرب باج‌گیرها از الگوهای خاصی پیروی می‌کنند که قابل تشخیص هستند؛ بنابراین می‌توان از این الگوها برای شناسایی و مقابله با انواع خاصی از باج‌افزارها استفاده کرد؛ لذا شناسایی خانواده باج‌افزارها دارای اهمیت فراوان است، چرا که علاوه بر شناسایی خانواده باج‌افزارها، می‌توان از روش‌های طراحی شده برای مقابله و بازیابی از حملات نیز استفاده کرد.

اما چیزی که برای متخصصین امنیت اطلاعات ترسناک‌تر است، این است که به نظر نمی‌رسد الگویی قابل تشخیص برای حملات باج‌افزارها وجود داشته باشد. اکثر روش‌های سنتی شناسایی مبتنی بر امضا[۲] نیز توسط گروه‌های هکری مهاجم دور زده می‌شوند؛ بنابراین پیاده‌سازی روش‌های نوین شناسایی باج‌افزارها به شدت احساس می‌شود. در این پژوهش مدلی ارائه شده که سایر پژوهشگران، با استفاده از معیارهای به دست آمده در خصوص مصرف منابع و همچنین با بهره‌گیری از روش‌های ابتکاری در هوش مصنوعی یا روش‌های اکتشافی، ابزارهای موثرتری برای تشخیص باج‌افزار توسعه دهند.

۲ مروری بر پژوهش‌های انجام شده

در مطالعه و مرور پژوهش‌های صورت گرفته در حوزه شناسایی باج‌افزارها بر اساس الگوی مصرف منابع سیستمی، مقالات اندکی تاکنون به چاپ رسیده است. بیشتر پژوهش‌های مرتبط در روی نظارت بر رفتار فرآیندها و تحلیل لاگ انجام شده است. همچنین بیشتر تحقیقات صورت گرفته که مرتبط با موضوع این پژوهش می‌باشند، در خصوص مطالعه رفتار باج‌افزار بر روی هارد دیسک بوده و بررسی همزمان هارد دیسک، حافظه RAM و پردازنده تاکنون انجام نشده است. به طور کلی پژوهش‌های مرتبط با شناسایی خانواده‌های مختلف باج‌افزاری را می‌توان به صورت زیر دسته‌بندی کرد: تشخیص مبتنی بر امضا، تشخیص بر اساس تحلیل ایستا و تشخیص بر اساس تحلیل پویا.

با توجه به بررسی‌های انجام شده، در این پژوهش‌ها نرخ تشخیص ارائه شده برای شناسایی تعداد بالای باج‌افزارها دارای کاستی‌هایی چون پایین بودن نرخ دقت تشخیص، نرخ بالای مثبت کاذب و حتی بالا بودن نرخ عدد تشخیص داده شده هستند. از دیگر کاستی‌های پژوهش‌های مذکور، غفلت از تأثیر نرخ سرعت در تشخیص باج‌افزارها است؛ عدم رفع کاستی‌های مذکور در زمان پیاده‌سازی اینگونه روش‌های شناسایی،

[2] Signature

موجب متحمل شدن هزینه‌هاي زمانی و مادي زيادي، و نيز موجب كندي سيستم شناسايی و عدم دستيابی بـه خروجی صحيح و واقعی خواهد شد.

ديموو همكاران در سال ۲۰۱۹ با اندازه‌گيری و استخراج شاخص‌های متريک HDD در يک حمله باج‌افزاری برای نوع از خانواده باج‌افزار، بر اين باور قرار گرفتند كه بهترين و كارآمدترين زمان برای شناسايی باج‌افزار در شبكه در زمان اجرای پی لود آغاز حمله می‌باشد. با اندازه‌گيری عملكرد هارد ديسک در زمان اجرای باج‌افزار اين امكان را می‌دهد كه در خلال رمزنگاری رفتار سيستم سنجيده شود؛ بنابراين می‌توان سرعت دامنه رمزنگاری را مشخص و ساير سيستم‌ها را ايزوله كرد. از معايب اين روش عدم بررسی ديگر منابع مرتبط با حمله باج‌افزاری می‌باشد.

دارابيان و همكاران در سال ۲۰۲۰ برای دسته‌بندی و شناسايی نمونه باج‌افزارها، از روش كاوش الگوهای متوالی استفاده نمودند. آن‌ها ويژگی‌هايی را به دست آورده، تا قابل استفاده برای الگوريتم‌های دسته‌بندی كننده يادگيری ماشين باشد. دقت ۹۹% در تشخيص نمونه‌های باج‌افزار و همين طور دقت ۹۶% در شناسايی و دسته‌بندی خانواده آن‌ها روی الگوريتم‌های متداول يادگيری ماشين، نشان از كيفيت بالای ويژگی‌های پيشنهادی دارد. اين روش دقت بالايی در شناسايی نمونه های بی‌خطر ندارد؛ از اين رو در حوزه امنيت و دفاع با درصد بالا قابل اطمينان نمی‌باشد.

جليليان و همكاران در سال ۲۰۱۶ روشی را پيشنهاد دادند كه مبتنی بر تشخيص امضا در محيط ايستا برای استخراج امضای فايل‌ها از روی آپكدهای برنامه بود كه بر اين اساس، باج‌افزارها به دو دسته سالم و مخرب تقسيم‌بندی گرديدند. اين روش دارای دقت مناسبی بوده ولی مستلزم پردازش زياد بوده و بسيار زمان‌بر می‌باشد، از اين رو در تمامی شرايط قابل استفاده نمی‌باشد.

در اين مقاله كه با رويكرد آزمايشگاهی انجام شده است، ابتدا جامعه آماری متشكل از ۵۰ نمونه باج‌افزار از ۱۰ خانواده مختلف گردآوری گرديده و پس از آماده‌سازی يک بستر امن و ايزوله، تک تک باج‌افزارها در اين محيط اجرا و آزمايش گرديده‌اند. ابزارهای اندازه‌گيری از پيش تهيه شده پارامترهای سرعت رمزنگاری، ميزان مصرف پردازنده، حافظه رم و ديسک را اندازه‌گيری می‌كنند. با توجه به فرضيه اصلی تحقيق، باج‌افزارهای هم خانواده تشابهات زيادی در پارامترهای ذكر شده خواهند داشت. اين موضوع در اين تحقيق به اثبات رسيده و در انتهای پژوهش مدلی برای اين ۱۰ خانواده باج‌افزاری ترسيم گرديده است. مدل ترسيم شده قابل بهره‌برداری برای متخصصين تحليل بدافزار و پژوهشگران و دانشجويان حوزه امنيت سايبری خواهد بود.

۳ محيط آزمايشگاهی

هرچند مطالعه رفتار باج‌افزارها را می‌توان با يک ميزبان ويندوز نيز انجام داد، اما معماری ايده‌آل بر اساس سيستم‌عامل مک يا لينوكس است؛ در صورتی كه باج‌افزار بتواند از داخل ماشين مجازی به ميزبان انتقال پيدا كند، به احتمال كمتر ميزبان را آلوده می‌كند؛ اين رخداد معمولاً از طريق آسيب‌پذيری در نرم‌افزار ماشين مجازی يا خطای تحليل‌گر رخ می‌دهد. با توجه به اينكه باج‌افزارهای ويندوزی در محيط لينوكس اجرا

نمی‌شوند، ترجیح ما استفاده از یک میزبان لینوکسی است؛ پس بنا به دلایلی که اشاره شد، سیستم‌عامل میزبان در این پژوهش نسخه دسکتاپ لینوکس اوبونتو ۲۲٫۰۴ در نظر گرفته شده است.

جامعه‌ی آماری مورد استفاده در این پژوهش مربوط به باج‌افزارهای ویندوزی می‌باشد؛ لذا برای پیاده‌سازی محیط آزمایشگاه از سیستم‌عامل خانواده ویندوز استفاده شده است. در انتخاب نوع سیستم‌عامل مهمان دو گزینه ویندوز ۷ و ویندوز ۱۰، با توجه به محبوبیتی که دارند پیش رو قرار داشت. با توجه به اینکه دیتاست مورد استفاده در این پژوهش مربوط به ۳ سال گذشته می‌باشد، در انتخاب نوع سیستم‌عامل مهمان نیز سعی شده تا از آخرین نسخه‌های سیستم‌عامل ویندوز ۱۰ استفاده گردد که مشکلی به لحاظ سازگاری به وجود نیاید.

در این پژوهش، سیستم‌عامل ویندوز ۱۰ با مشخصات زیر برای ساخت آزمایشگاه در نظر گرفته شد:

OS Name: Microsoft Windows 10 Enterprise LTSC (X64)
OS Version: 10.0.17763 N/A Build 17763
CPU: Intel Corei7-7700 3.60GHz (4 Core)
RAM: 8.00 GB
Disk: ADATA SU800 256GB NVME

برخی از باج‌افزارها برای اجرا نیاز به اتصال به اینترنت دارند تا با سرور فرمان و کنترل خود ارتباط بگیرند. با توجه به این که ماشین مجازی تحلیل، یک محیط ایزوله است و دسترسی به اینترنت ندارد، از طرفی به خاطر به حداقل رساندن امکان گسترش باج‌افزار به سیستم‌عامل میزبان، امکان فعال کردن اینترنت بر روی آن وجود ندارد. راهکاری که برای این موضوع در نظر گرفته شد، استفاده از یک شبیه‌ساز پروتکل‌های اینترنت می‌باشد. نرم‌افزار INetSim و BurpSuite برای این منظور در محیط لینوکسی REMnux پیکربندی شدند و با هدایت ترافیک شبکه ماشین تحلیل به سمت INetSim این مشکل مرتفع گردید. REMnux در واقع یک سیستم‌عامل لینوکسی بر پایه اوبونتو می‌باشد که مختص آنالیز بدافزار و مهندسی معکوس ساخته شده است.

به‌دلیل یکسان‌سازی شرایط محیط آزمایشگاه تست باج‌افزار، تمامی موارد مرتبط با سیستم عامل مهمان از قبیل Windows Defender، Windows Update غیرفعال گردیدند. برای تست و برداشت اطلاعات مورد نیاز نیز از نرم‌افزارهای HashOptionRightClick.reg، Process Explorer، FreeCommanderXE، Dummy File Generator استفاده گردید؛ درنهایت، تعدادی فایل طعمه بر اساس جدول ۱ با ویژگی‌های زیر با استفاده از ابزار اسکریپتی ایجاد گردید، تا در فرآیند آزمایش مورد استفاده قرار گیرند. برای اینکه فایل‌های طعمه جزء لیست سفید[3] باج‌افزارها نباشند و صد در صد رمزگذاری شوند، پسوند (doc.) برای آن‌ها در نظر گرفته شده است. همچنین برای سهولت در رمزنگاری و افزایش سرعت، مسیر فایل‌های طعمه در مسیر ریشه[4] سیستم‌عامل در نظر گرفته شده است.

[3] Whitelist
[4] Root

جدول ۱: مشخصات فایل‌های طعمه

حجم فایل طعمه	1 KB	100 KB	1 MB	100 MB	1 GB
تعداد	۱۰۰۰	۱۰۰۰	۱۰۰۰	۱۰۰۰	۱۰۰۰

۴ آماده‌سازی دیتاست پژوهش

برای آماده‌سازی دیتاست، ابتدا ۵ نمونه از باج‌افزارهای خانواده‌های HiddenTear، Dharma، Phobos، STOP/Djvu، Ryuk، Maze، Revil، Makop، LockBit و Conti از سامانه مرجع VirusTotal[۵] انتخاب و دانلود گردید. خانواده‌های باج‌افزاری به گونه‌ای انتخاب شده‌اند که بر اساس آمارهای جهانی، قربانیان زیادی را به خود اختصاص داده و در مناطق جغرافیایی متعددی منتشر شده باشند. همچنین سعی بر آن بوده که از بین باج‌افزارهای مطرح که قربانیانی را نیز در داخل کشور داشته‌اند، نمونه‌های تصادفی انتخاب گردند. تاریخ انتشار نمونه‌های انتخاب شده حداکثر مربوط به سه سال گذشته می‌باشد؛ دلیل این امر این است که بسیاری از باج‌افزارهای قدیمی به دلیل از کار افتادن سرورهای فرمان و کنترل و باگ‌های نرم‌افزاری یا سازگاری با سیستم‌عامل‌های جدید، هرگز اجرا نمی‌شوند. لذا برای حل این مشکل، بازه زمانی انتشار باج‌افزارها، حداکثر سه سال گذشته در نظر گرفته شده است. ملاک شناسایی نمونه‌های باج‌افزاری بر اساس الگوریتم هش[۶] SHA256 می‌باشد.

اجرای باج‌افزارها در محیط آزمایشگاهی شامل سه مرحله شروع، رمزنگاری و پایان می‌باشد. مدت زمان فرآیند رمزنگاری بسته به نوع باج‌افزار متفاوت است؛ این بازه معمولاً بین ۱ تا ۲۰ دقیقه ممکن است به طول بینجامد و این رفتار، محاسبه دقیق سرعت رمزگذاری فایل‌ها را دچار مشکل می‌کند. برای حل این مسأله، نتایج آزمایش به صورت میانگین بازه‌ای از زمان که شامل Peakهای متعدد می‌باشد، در نظر گرفته شده است.

در حین اجرای باج‌افزارها، میزان منابع مصرف شده توسط ابزار Process Explorer ثبت گردید. پس اتمام فرآیند رمزنگاری نیز، کلیه فایل‌هایی که پسوندشان تغییر کرده بود را به صورت صعودی مرتب کرده و تاریخ آخرین تغییر انجام شده بر روی فایل‌ها را توسط ابزار FreeCommanderXE ثبت نمودیم.

۵ تحلیل دست‌آوردهای فنی و ترسیم مسیر آینده پژوهش

تحلیل نتایج: بر اساس مشاهدات صورت گرفته از اجرای باج‌افزارها در محیط آزمایشگاهی و شاخص‌های به دست آمده، نتایج مطابق جدول ۲ حاصل گردیده است.

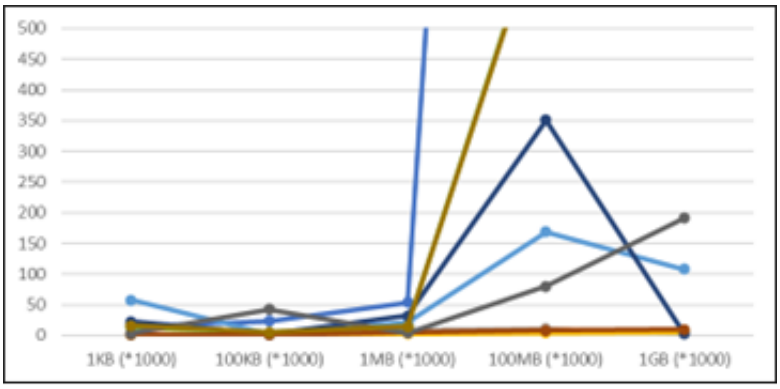
نتیجه اول: بر اساس شکل ۱ سرعت رمزنگاری فایل‌های با حجم مشخص توسط باج‌افزارهای هم‌خانواده، تقریباً یکسان است. به‌عنوان مثال، باج‌افزار خانواده Ryuk تعداد ۱۰۰۰ عدد فایل با حجم ۱۰۰ کیلوبایت

[۵] https://www.virustotal.com/

[۶] هش تابعی است که ورودی از حروف و اعداد را به یک خروجی رمزگذاری شده با طولی ثابت تبدیل می‌کند. توابع hash در سرتاسر اینترنت به منظور ذخیره ایمن کلمه عبور، یافتن سوابق تکراری، ذخیره سریع و بازیابی اطلاعات و موارد این‌چنین به‌کاربرده می‌شوند.

جدول ۲: میانگین سرعت رمزنگاری فایل‌ها

Ransomware Family	1KB (*1000)	100KB (*1000)	1MB (*1000)	100MB (*1000)	1GB (*1000)
Tear Hidden	15.8	23.2	54	3418	5940
Dharma	2.8	1.2	8	10.8	4.8
Phobos	1	1.4	9.2	11	3.4
STOP/Djvu	1	2.4	2.6	4.2	6
Ryuk	58.2	2.4	21	168.8	108
Maze	14	4.8	15	708	6370
Revil	22.6	2.2	32.2	351	3.6
Makop	1	1.4	5.4	8.2	10.2
LockBit	3.2	42.6	4	80	190.8
Conti	16.2	5.4	14.8	704	1757



شکل ۱: میانگین سرعت رمزنگاری فایل‌های طعمه توسط باج‌افزارها

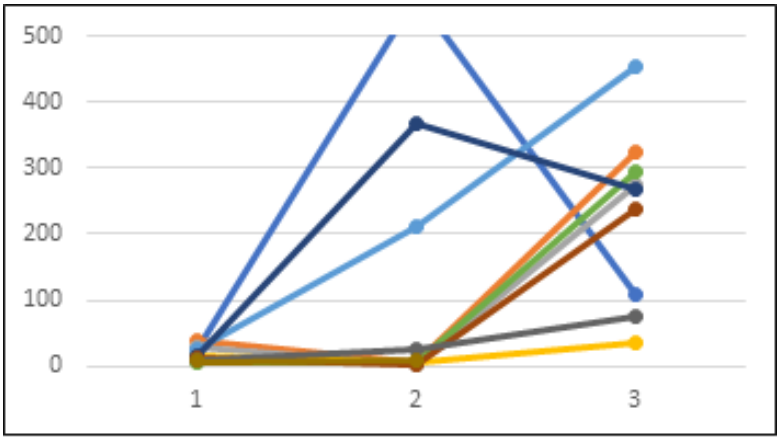
را در بازه زمانی ۱ تا ۴ ثانیه رمزگذاری می‌کند. این رفتار در بین تمام نمونه‌های باج‌افزاری در این خانواده مشابه است. دلیل این موضوع این است که ساختار کد و الگوی رمزنگاری در بین باج‌افزارهای یک خانواده تشابهات زیادی دارد.

نتیجه دوم: بر اساس جدول ۳، میزان مصرف منابع سیستم هنگام رمزنگاری فایل‌های با حجم مشخص توسط باج‌افزارهای هم خانواده، تقریباً یکسان است؛ به عنوان مثال در مورد باج‌افزار Makop میزان اشغال حافظه RAM حین اجرای باج‌افزار بین بازه ۲٫۲ تا ۲٫۸ مگابایت قرار دارد. با توجه به شکل ۲، این رفتار در بین تمام نمونه‌های باج‌افزاری در این خانواده مشابه است.

نتیجه سوم: در این آزمایش، هنگامی که حجم فایل‌ها رفته رفته افزایش می‌یابد، خانواده‌های مختلف باج‌افزاری دو رفتار متفاوت از خود نشان دادند:

جدول ۳: میانگین منابع اشغال شده رمزنگاری فایل‌ها

Ransomware Family	(%) CPU	(MB) RAM	(MB/s) Disk
Tear Hidden	24.26	561.44	109.1
Dharma	40.46	6.88	326
Phobos	28.44	3.94	274.8
STOP/Djvu	15.06	6.84	34.48
Ryuk	24.6	210.1	452.9
Maze	7.06	6.98	293.98
Revil	16.66	368.12	267.94
Makop	12.64	2.44	237.98
LockBit	8.8	26.9	75.84
Conti	7.56	8.94	351.06



شکل ۲: منابع اشغال شده توسط Process مربوط به باج‌افزار در حین رمزنگاری

- حجم فایل با سرعت رمزنگاری و مصرف منابع رابطه مستقیم دارد؛ یعنی با افزایش حجم فایل‌ها، سرعت رمزنگاری و مصرف منابع سیستم نیز افزایش می‌یابد. باج‌افزارهای Hidden، Tear Maze و Conti این رفتار را از خود نشان دادند. بررسی ساختار فایل‌های رمزگذاری شده نشان داد که این باج‌افزارها در مواجهه با فایل‌های با حجم متفاوت، تمام ساختار فایل را رمزگذاری می‌کنند و حجم فایل هیچ تاثیری در الگوی رمزنگاری ندارد؛ به همین دلیل فایل‌های با حجم پایین، سریع‌تر و فایل‌های با حجم بالا، کندتر رمزگذاری می‌شوند. شکل شماره ۱ نیز گویای این مطلب است.

- حجم فایل با سرعت رمزنگاری و مصرف منابع رابطه عکس دارد؛ یعنی با افزایش حجم فایل‌ها، سرعت رمزنگاری و میزان مصرف منابع سیستم کاهش می‌یابد یا تغییر چندانی نمی‌کند؛ باج‌افزارهای Dharma، Phobos، STOP/Djvu، Ryuk، Revil، Makop و LockBit این رفتار را از خود نشان دادند. بررسی ساختار فایل‌های رمزگذاری شده نشان داد که این باج‌افزارها در مواجهه با فایل‌های با حجم پایین (حدوداً ۱۰۰ مگابایت)، تمام ساختار فایل را رمزگذاری می‌کنند؛ اما هنگامی که باج‌افزار با فایل‌های بیشتر از این حجم مواجه می‌شود، الگوی رمزنگاری تغییر کرده و تنها بخشی از ساختار فایل (شامل Header فایل و بخش‌هایی از بدنه یا انتهای فایل) رمزگذاری می‌گردد؛ به همین علت، سرعت رمزنگاری بسیار بالاتر از گروه قبلی است. دلیل این موضوع این است که باج‌افزارهای ساخت یافته، با کاهش مدت زمان رمزنگاری، دیتای بیشتری را در فاصله زمانی کوتاه رمزگذاری می‌کنند و بدین ترتیب، خسارت بیشتری به بار می‌آورند. این به حداقل رساندن سربار عملکرد باج‌افزار، نه تنها به کاهش احتمال شناسایی شدن توسط نرم‌افزارهای امنیتی نظارت بر فرایندها کمک می‌کند، بلکه به طور موثری نیز از منابع پردازشی سیستم آلوده استفاده می‌کند تا تعداد و حجم بیش‌تری از فایل‌ها را رمزگذاری نماید؛ از سوی دیگر هرچه زمان رمزنگاری کوتاه‌تر باشد، قدرت مهار حمله یا محدودسازی دامنه خسارت توسط مدیر سیستم نیز کاهش می‌یابد. بنابراین باج‌افزارهای این گروه جزو خطرناکترین باج‌افزارهای دنیا محسوب می‌گردند.

- با استفاده از نتایج حاصل از پژوهش، با بررسی رفتار مشترک بین خانواده‌های باج‌افزاری و الگوبرداری از میزان مصرف منابع مختلف سیستم، با ایجاد نمودن یک دسته‌بندی و به کارگیری تکنیک‌های هوش مصنوعی می‌توان نتایج مناسب‌تری حاصل نمود.

مراجع

[۱] آلن لیسکا، تیموتی گالو (۲۰۱۹). «باج‌افزار: روش‌های دفاع در برابر باج‌گیری دیجیتال»، (ترجمه دکتر مهران گرمه‌ء، میلاد حضرتی، سارا رحیمی دوین، ۱۳۹۶)، انتشارات نشر گسترش علوم پایه.

[۲] حمید دارابیان، ستار هاشمی، سجاد همایون، کرم‌الله باقری‌فرد (۱۴۰۰). «شناسایی باج‌افزارها و خانواده آن‌ها با بهره‌گیری از روش کاوش الگوهای متوالی در تحلیل پویا».

[۳] آزاده جلیلیان، ابراهیم انصاری (۱۳۹۶). «شناسایی بدافزار براساس تشخیص امضای ایستا کد دستور و فایل باینری».

[4] Manaar Alam, Sayan Sinha, Sarani Bhattacharya, Swastika Dutta, Debdeep Mukhopadhyay, Anupam Chattopadhyay (2018). Ransomware Prevention via Performance Counters.

[5] Dimo Dimov, Yuliyan Tsonev (2020). Observing, Measuring and Collecting HDD Performance Metrics on a Physical Machine During Ransomware Attack, DOI:10.11610/isij.4723.

[6] Hesham A. Hefny, Nagy Ramadan, Hesham Alshaikh (2020)., Ransomware Prevention and Mitigation Techniques, International Journal of Computer Applications 117(40):31-39.

[7] Nathanael Paul, Sudhanva Gurumurthi, David Evans (2005). Towards Disk-Level Malware Detection.

[8] Jelena Milosevic, Miroslaw Malek, Alberto Ferrante (2016). A Friend or a Foe? Detecting Malware using Memory and CPU Features, International Conference on Security and Cryptography, 10.5220/0005964200730084.

[9] Juan A. Herrera-Silva, Lorena Barona, Leonardo Valdivieso, Myriam Hernandez Alvarez (2019). A Survey on Situational Awareness of Ransomware Attacks—Detection and Prevention Parameters, Remote Sensing 11(10):1168.

[10] Patrick Düssel, Thorsten Holz, Pavel Laskov, Konrad Rieck (2009). Learning and Classification of Malware Behavior, 10.17877/DE290R-2041.

[11] Abdullahi Arabo, Remi Dijoux, Timothee Poulain, Gregoire Chevalier (2020). Detecting Ransomware Using Process Behavior Analysis, Procedia Computer Science 168:289-296, 10.1016/j.procs.2020.02.249.

[12] Daniel Gonzalez, Thaier Hayajneh (2017). Detection and Prevention of Crypto-Ransomware, Conference: 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), 10.1109/UEMCON.2017.8249052.

[13] Patrick Luckett, J. Todd Mcdonald (2018). Identifying stealth malware using CPU power consumption and learning algorithms, Journal of Computer Security 26(10):1-25, 10.3233/JCS-171060.

[14] Robert Bridges, Jarilyn Hernandez Jimenez (2018). Towards malware detection via CPU power consumption: Data collection design and analytics, Project: Power consumption analysis for malware detection.

[15] Hernandez Jimenez, J., & Goseva-Popstojanova, K. (2019). Malware Detection Using Power Consumption and Network Traffic Data. 2019 2nd International Conference on Data Intelligence and Security (ICDIS).

[16] Kuruvila AP, Kundu S, Basu K. (2020). Analyzing the efficiency of machine learning classifiers in hardware-based malware detectors.

[17] Ramesh G, Menen A. (2020). Automated dynamic approach for detecting ransomware using finite-state machine. Decision Support Systems 138:113400.

[18] Tanana, D., & Tanana, G. (2020). Advanced Behavior-Based Technique for Cryptojacking Malware Detection. 2020 14th International Conference on Signal Processing and Communication Systems (ICSPCS), 7.