وزارت علوم، تحقیقات و فناوری
دانشگاه تحصیلات تکمیلی علوم پایه
گاوازنگ - زنجان

تشخیص بدافزار به‌صورت ایستا با استفاده از رویکرد مبتنی بر رفتار

پایان‌نامهٔ کارشناسی ارشد

علیرضا عباسعلی‌پور

استاد راهنما: **دکتر ابراهیم انصاری**

شهریور ۱۳۹۶

بسم الله الرحمن الرحیم

تقدیم به پدر و مادر مهرورزم و کلیه فرهیختگانی که در راه علم‌آموزی من را با چراغ افروخته علم هدایت نموده‌اند.

تشکر و قدردانی

سپاس خدای مهربان را که اندیشه‌ام داد.

حمد و ستایش بی‌قیاس خدای را سزاست که از الطاف خود در انسان دمید و او را اشرف مخلوقات خود قرار داد. حال که به لطف او توفیق تحصیل علم و کسب دانش را پیدا نمودم، از خداوند متعال می‌خواهم که قدم‌هایم را در راه خدمت به جامعه استوار گرداند تا بتوانم از آنچه در این سال‌ها آموخته‌ام در مسیر پیشرفت و آبادانی کشور عزیزم استفاده نمایم. این هنوز نخستین گام از آموختن است و انسان با شأن والای خود هیچ‌گاه از آموختن فارغ نمی‌شود. امیدوارم که این مرحله نیز سرآغاز مبارکی باشد بر تعالی علمی و رسیدن به مقام شامخ دانش‌پژوهی.

در پایان بر خود لازم می‌دانم از مساعدت‌ها و کوشش فراوان و شایان تقدیر استاد ارجمندم جناب آقای دکتر ابراهیم انصاری که با راهنمایی‌های ارزنده خود مرا در تهیه این تحقیق یاری نمودند و نیز از دوستانی که در گردآوری این تحقیق با بنده همکاری داشته‌اند، صمیمانه سپاسگزاری نمایم.

چکیده

با افزایش روزافزون تعداد برنامه‌های مخرب و خسارت‌های ناشی از آن‌ها، نیاز به ارائه روشی خودکار و یکپارچه برای تشخیص بدافزارها به امری حیاتی تبدیل شده است. گسترش سریع نرم‌افزارهای مخرب کامپیوتری و افزایش تهدیدات ناشی از این بدافزارها در اینترنت و محیط شبکه، اهمیت توجه به مسئله امنیت کامپیوترها و شبکه‌های کامپیوتری را در دهه‌های اخیر برجسته کرده است. به منظور ایجاد امنیت در این حوزه، به‌کارگیری روش‌های ضد بدافزار، بررسی مزایا و معایب استفاده از این روش‌ها و نیز ارائه روش‌های مناسب بسیار رایج هستند. به نظر می‌رسد که روش عمومی برای مقابله با نرم‌افزارهای مخرب، استفاده از نرم‌افزارها و برنامه‌های ضد ویروس است، اما این نرم‌افزارها قادر به شناسایی بدافزارهای روز نخست و جدید نیستند. روش‌های تشخیص بدافزار مبتنی بر تحلیل ایستا با توجه به سربار کم‌تر، نیاز نداشتن به محیط اجرایی خاص و بررسی تمام مسیرهای ممکن اجرایی برنامه، نسبت به روش‌های مبتنی بر تحلیل پویا کارآمدتر هستند و برای ارائه روش‌های خودکار و به حداقل‌رساندن نقش عامل تحلیل‌گر انسانی مناسب هستند. استفاده از روش‌های مبهم‌سازی، ضد دیس‌اسمبلی از جمله مهم‌ترین عامل‌های بازدارنده در روش‌های مبتنی بر تحلیل ایستا هستند. وجود انشعابات غیرمستقیم و استخراج صحیح کدهای عملیاتی در کد برنامه بسیار پر اهمیت است. این چالش‌های پیش رو، روش‌های موجود در تحلیل ایستا را به اشتباه انداخته و کارایی بهینه را از آن‌ها می‌گیرند. این پژوهش روشی جدید براساس تحلیل ایستای کد و مدل نمایه پنهان مارکوف برای تشخیص کدهای مخرب ارائه می‌نماید. با استفاده از تکنیک‌های داده‌کاوی و براساس تحلیل ایستای فایل‌های باینری (دیس‌اسمبل و روش‌های مهندسی معکوس) یک روش تشخیص خودکار ارائه شده است و شناسایی گونه‌های خانواده‌های بدافزار که چندریخت و فراریخت هستند را در برمی‌گیرد. در روش پیشنهادی با اصلاح روند ایجاد هم‌ترازی توالی‌های چندگانه، ضمن سرعت بخشیدن به آن، با

اضافه کردن توالی اجماع از روی MSA مربوطه توانستیم مدل نمایه پنهان مارکوف را بهبود بخشیده و نتایج بسیار مطلوبی به‌دست آورده‌ایم. از آنجا که تمام قسمت‌های فایل‌های بدافزار دارای ذات مخرب نیستند، هدف ما متمایز کردن بخش‌های مخرب از بخش‌های سالم است؛ به‌طوری که در روند تشخیص بدافزار، اثر بخش‌های مخرب افزایش یابد و در مقابل قسمت‌هایی که به فایل‌های سالم شبیه هستند تاثیر کمتری بگذارند.

واژه‌های کلیدی: *امنیت کامپیوتر، تشخیص بدافزار، بدافزار چندریخت، تحلیل ایستا، تشخیص مبتنی بر رفتار*

فهرست

چکیده . . . . . پنج

۱ مروری بر پژوهش پیش رو **۱**

۱.۱ مقدمه ۱

۲.۱ ساختار و اهداف پژوهش ۷

۲ تعاریف و مفاهیم اولیه **۱۳**

۱.۲ بدافزار ۱۳

۲.۲ انواع بدافزار ۱۴

۱.۲.۲ هرزنامه ۱۴

۲.۲.۲ تروجان ۱۵

۳.۲.۲ روت‌کیت ۱۵

۴.۲.۲ ویروس ۱۵

۵.۲.۲ کرم ۱۷

۳.۲ ویروس فراریخت ۲۱

۱.۳.۲ منشاء ویروس‌ها ۲۱

۲.۳.۲ کیت‌های سازنده ۲۷

۴.۲ تکنیک‌های ایجاد ابهام در کد ۳۰
۱.۴.۲ افزودن کد زائد ۳۰
۲.۴.۲ تغییر نام ثبات‌ها ۳۱
۳.۴.۲ جایگشت زیرروال‌ها ۳۱
۴.۴.۲ مرتب‌سازی مجدد کد از طریق پرش ۳۳
۵.۴.۲ جایگزینی کد معادل ۳۴

۳ مروری بر کارهای پیشین ۳۶

۱.۳ روش‌های تحلیل بدافزار ۳۷
۱.۱.۳ تحلیل ایستا ۳۷
۲.۱.۳ تحلیل پویا ۳۸
۳.۱.۳ رویکرد ترکیبی ۳۹
۲.۳ تکنیک‌های تشخیص بدافزار ۳۹
۱.۲.۳ تکنیک مبتنی بر امضاء ۴۰
۲.۲.۳ تکنیک مبتنی بر رفتار ۴۰
۳.۲.۳ تکنیک مبتنی بر ناهنجاری ۴۱
۴.۲.۳ تکنیک مبتنی بر ساختار ۴۱
۵.۲.۳ تکنیک مبتنی بر شباهت ۴۲
۱.۵.۲.۳ تکنیک مبتنی بر گراف ۴۳
۲.۵.۲.۳ تکنیک مبتنی بر فاصله جایگزینی ساده ۴۳
۶.۲.۳ تکنیک مبتنی بر الگوی آماری ۴۴
۱.۶.۲.۳ تکنیک مبتنی بر مدل‌های پنهان مارکوف ۴۴
۲.۶.۲.۳ تکنیک مبتنی بر مدل‌های نمایه پنهان مارکوف ۴۵

۴ روش پیشنهادی ۴۶
۱.۴ تئوری مدل‌های پنهان مارکوف ۴۶
۱.۱.۴ زنجیره مارکوف ۴۶
۱.۱.۱.۴ زنجیره مارکوف مرتبه بالا ۴۷
۲.۱.۴ مدل‌های پنهان مارکوف ۴۸
۱.۲.۱.۴ الگوریتم‌هایی برای امتیازدهی به دنبالهٔ ناشناخته در برابر یک مدل شناخته‌شده ۴۹
۲.۲.۱.۴ مدل‌های نمایه پنهان مارکوف ۵۳
۳.۲.۱.۴ هم‌ترازی توالی‌های چندگانه ۵۴
۲.۴ رویکرد پیشنهادی ۵۷
۱.۲.۴ آماده‌سازی داده ۵۷
۲.۲.۴ مدل‌های پنهان مارکوف ترکیبی با ان-گرم ۵۸
۳.۲.۴ بهینه‌سازی مدل‌های نمایه پنهان مارکوف ۵۹
۴.۲.۴ مدل‌های نمایه پنهان مارکوف ترکیبی با ان-گرم ۶۰

۵ ارزیابی روش پیشنهادی ۶۱
۱.۵ ارزیابی مدل‌های پنهان مارکوف ۶۲
۱.۱.۵ ارزیابی مدل‌های پنهان مارکوف ترکیبی با ان-گرم ۶۵
۲.۵ ارزیابی مدل‌های نمایه پنهان مارکوف ۶۷
۱.۲.۵ ارزیابی مدل‌های نمایه پنهان مارکوف ترکیبی با ان-گرم ۷۰

۶ نتیجه‌گیری و کارهای آتی ۷۳
واژه‌نامه انگلیسی به فارسی ۸۷
واژه‌نامه فارسی به انگلیس ۹۵

فهرست تصاویر

۱.۲ نسخه‌برداری ویروس ساده ۲۲
۲.۲ نسخه‌برداری ویروس رمزگذاری شده ۲۲
۳.۲ نسخه‌برداری ویروس چندریخت ۲۳
۴.۲ نسخه‌برداری ویروس فراریخت ۲۴
۵.۲ نمونهٔ Regswap ۳۲
۶.۲ نمونهٔ جایگشت زیرروال‌ها ۳۳
۷.۲ نمونهٔ مرتب‌سازی مجدد کد ۳۴
۸.۲ جایگزینی کد معادل در ویروس فراریخت W۳۲.Evol ۳۵

۱.۴ زنجیره مارکوف برای توالی DNA ۴۷
۲.۴ مدل پنهان مارکوف ۴۸
۳.۴ شبه کد الگوریتم ویتربی ۵۲
۴.۴ مدل نمایه پنهان مارکوف ۵۵
۵.۴ گراف هم‌ترازی ترتیبی بخشی ۵۶

۱.۵ نمودار مدل خانواده Cygwin ۶۲
۲.۵ نمودار مدل خانواده G۲ . . . . . ۶۳

۳.۵ نمودار مدل خانواده MPCGEN ۶۴
۴.۵ نمودار مدل خانواده NGVCK ۶۴
۵.۵ نمودار مدل خانواده Cygwin برای ۲-گرم ۶۵
۶.۵ نمودار مدل خانواده G۲ برای ۲-گرم ۶۶
۷.۵ نمودار مدل خانواده MPCGEN برای ۲-گرم ۶۶
۸.۵ نمودار مدل خانواده NGVCK برای ۲-گرم ۶۷
۹.۵ نتایج کار پیشین برای خانواده NGVCK ۶۸
۱۰.۵ نتایج مدل خانواده Cygwin ۶۹
۱۱.۵ نتایج روش پیشنهادی مدل خانواده NGVCK ۶۹
۱۲.۵ نمودار مدل خانواده Cygwin برای ۲-گرم ۷۰
۱۳.۵ نمودار مدل خانواده G۲ برای ۲-گرم ۷۱
۱۴.۵ نمودار مدل خانواده MPCGEN برای ۲-گرم ۷۱
۱۵.۵ نمودار مدل خانواده NGVCK برای ۲-گرم ۷۲

۱.۶ نمودار مدل خانواده NGVCK ۷۴
۲.۶ نمودار مدل خانواده NGVCK با تقویت رویکرد پیشنهادی ۷۵

فهرست جداول

فصل اول

مروری بر پژوهش پیش رو

۱.۱ مقدمه

با توجه به این‌که امروزه، بدافزارها توانایی ایجاد خرابی زیادی را در سطح وسیعی دارند و از طرفی تولید آن‌ها به‌صورت روزافزون رو به رشد است، لذا شناسایی بدافزارها، امری بسیار مهم است. مجموعهٔ گسترده‌ای از تکنیک‌های پیشرفتهٔ تشخیص در مقالات متعدد در زمینهٔ شناسایی بدافزار مورد توجه قرار گرفته است. بعضی از تکنیک‌های تشخیصی تنها بر تحلیل ایستا[۱] [۱—۱۵] تکیه می‌کنند، یعنی ویژگی‌های مورد نظر را بدون اجرای نرم‌افزار به دست می‌آورند. علاوه بر این، تحلیل پویا[۲] [۱۶—۲۵] به‌طور موفقیت آمیز در مسئله تشخیص نرم‌افزارهای مخرب به کار گرفته شده است. به تازگی، روش‌های ترکیبی[۳] [۲۶—۲۹] مورد بررسی قرار گرفته‌اند، که در آن از ویژگی‌های ایستا و پویا استفاده می‌شود. روش‌های قدیمی با استفاده از امضای بدافزار، نرم‌افزارهای مخرب را شناسایی می‌نمایند. اگرچه این

---

[۱] static analysis

[۲] dynamic analysis

[۳] hybrid approaches

روش یک روش مؤثر است و سرعت بالایی دارد، اما مشکل آن این است که فقط بدافزارهایی را شناسایی می‌کند که امضای آن‌ها را از قبل داشته باشد [۳۰]. از طرف دیگر هنوز هم استخراج امضاء به‌طور کامل ماشینی نشده است و این کار توسط افراد متخصص انجام می‌شود. در این روش، مشکل اصلی این است که ما باید منتظر ایجاد خرابی در بخشی از سیستم‌ها شویم و پس از آن، متخصصین شروع به کار کنند و با استفاده از ابزارهایی که در اختیار دارند و با تکیه بر تخصص و تجربه خود، امضای بدافزار را استخراج کنند. در این مدت، بدافزار فرصت می‌یابد تا خود را منتشر و هر چه می‌تواند خسارت وارد کند. به همین دلیل روش‌های دیگری به وجود آمدند که مبتنی بر امضاء نیستند. این روش‌های جدید با تحلیل رفتار برنامه‌ها می‌توانند بدافزارها را تشخیص دهند. هدف اصلی ما نیز در این پژوهش تکیه کردن بر این روش‌ها است. ما از تکنیک‌های داده‌کاوی برای تحلیل رفتار برنامه‌ها استفاده می‌کنیم، در واقع سیستمی ساخته شده است که می‌تواند با استفاده از رویکردهای معناگرا، با تحلیل کد و یا تحلیل رفتار، نیت برنامه‌ها را بفهمد و بدافزار را تشخیص دهد. یکی از روش‌های مؤثری که در رویکرد تحلیل ایستا، بسیار استفاده می‌شود، بهره‌گیری از مدل‌های پنهان مارکوف و استخراج کدهای عملیاتی[۱] با تحلیل و بررسی بایت‌های خوانده‌شده از فایل به‌صورت باینری[۲] و یا استفاده از ابزار موجود برای مهندسی معکوس و دیس‌اسمبل‌کردن فایل‌ها است که در این زمینه وجود دارند [۴، ۸، ۳۱، ۳۲]. در نهایت با استفاده از این داده‌ها و ویژگی‌های استخراج‌شده، مدل کلاسه‌کننده ساخته می‌شود و با استفاده از تشخیص ترکیبی، بدافزارها و فایل‌های پاک را از هم جدا می‌نماییم. بر طبق گزارش فارل[۳] و گزارش امنیتی حملات اینترنتی سیمانتیک[۴] در سال ۲۰۱۶ با تکامل و رشد روزافزون روندهای جرائم سایبری، شاهد اوج حملات سایبری در چند دههٔ گذشته بودیم. امروزه

---

[۱] opcodes

[۲] binary

[۳] o'Farrell

[۴] Internet Security Threat Report (ISTR)

شاهد رشد روزافزون بدافزارهای[۱] پیچیده هستیم که با کمک تکنیک‌های ابهام‌سازی[۲]، رمزگذاری[۳] و فشرده‌سازی[۴] سعی در پنهان‌سازی[۵] خود و اهداف خود دارند و تحلیل‌گران را با مشکلات جدی مواجه ساخته‌اند. یکی از پیچیده‌ترین بدافزارهای مخرب جهان، استاکس‌نت[۶]، در ژوئن ۲۰۱۰ کشف شد که سیستم‌های کنترل صنعتی را هدف قرار می‌داد. این کرم با هدف حمله به تأسیسات هسته‌ای ایران ایجاد شده بود. ابتدا از طریق مایکروسافت ویندوز گسترش می‌یافت و سیستم‌های کنترل صنعتی زیمنس[۷] را هدف قرار می‌داد. هدف اصلی آن این بود که سیستم‌های کنترل صنعتی را به‌صورت مخفیانه به منظور ایجاد آسیب، دوباره برنامه‌ریزی کنند. برای این کار، ابزار و تکنیک‌هایی مانند، استثمار روز نخست[۸]، روت کیت کنترل‌کننده منطقی قابل برنامه‌ریزی[۹] ، فرار از ضد ویروس[۱۰]، تزریق فرآیند[۱۱] و روش‌های به دام‌اندازی[۱۲]، گسترش شبکه‌ای[۱۳]، به‌روزرسانی با استفاده از P۲P[۱۴] و یک رابط فرمان و کنترل مورد استفاده قرار می‌داد [۱]. همان‌طور که ذکر شد روش‌های صنعتی گذشته در مواجهه با این نسل از بدافزارها ناکارآمد هستند. امروزه با بهره‌گیری از روش‌های هوشمند تلاش می‌شود زمان

---

۱ malwares

۲ obfuscation

۳ encryption

۴ compression

۵ stealth

۶ Stuxnet

۷ Siemens

۸ zero-day exploits

۹ programmable logic controller (PLC) rootkit

۱۰ antivirus evasion

۱۱ process injection

۱۲ hooking techniques

۱۳ network spreading

۱۴ peer to peer

تحلیل برنامه‌های مشکوک کاسته شود و دقت تشخیص بالاتر رود [۳۳]. در این پایان‌نامه ما یک روش هوشمند کارا ارائه خواهیم داد که در زمانی مناسب قادر است برنامه‌هایی را که حاوی کدهای مخرب هستند یا اعمال بدخواهانه را دنبال می‌کنند، شناسایی کند. در روش پیشنهادی ما از مدل آماری به‌نام مدل نمایه پنهان مارکوف[۱] که از روی کد اسمبلی[۲] برنامهٔ موردنظر ساخته می‌شود، استفاده می‌شود. هر کد عملیاتی استخراج‌شده به‌صورت نمادهای[۳] کاراکتری یا اعداد نگاشته می‌شوند. با در کنار هم قرارگرفتن آن‌ها با همان ترتیبی که از فایل باینری استخراج می‌شوند، دنباله‌ای از نمادها را ایجاد می‌نمایند که این رشته دنباله نمایشی از فایل مورد تحلیل است. در نهایت تمام فایل‌های متعلق به یک خانواده خاص از ویروس‌های چندریختی و فراریختی به‌طور یک مجموعه در نظر گرفته می‌شود و با ایجاد هم‌ترازی توالی چندگانه[۴] برای آن، مدلی آماری ساخته می‌شود که به‌نام PHMM معروف است. با داشتن نمونه فایل ورودی، کدهای عملیاتی آن استخراج می‌شود و رشته دنباله‌ای از آپکدهای فایل ایجاد می‌گردد. این رشته به‌عنوان ورودی به مدل نمایه پنهان مارکوف داده می‌شود و امتیاز به‌دست آمده برای آن با تمام امتیازهای حاصله از مدل‌های دیگر که برای خانواده‌های دیگر بدافزار ساخته شده‌اند، مقایسه و شباهت فایل نمونه برای هر خانواده محاسبه و در نتیجه در یکی از خانواده‌های مدل‌شده کلاسه می‌شود. در نهایت با استفاده از تشخیص ترکیبی، بدافزارها و فایل‌های پاک را از هم جدا می‌نماییم. ویروس‌های کامپیوتری فراریخت[۵] (دگردیس) با تغییر ساختار خود هر بار که آن‌ها را پخش می‌کنند، جهش[۶] می‌یابند. بر خلاف سایر ویروس‌ها، از تکنیک‌های ابهام‌سازی کد

---

۱ Profile Hidden Markov Models

۲ assembly

۳ symbols

۴ Multiple Sequence Alignment

۵ metamorphic

۶ mutate

بر روی بدن ویروس استفاده می‌کنند و امضایی[۱] مشترک را ارائه نمی‌دهند. با ظهور کیت‌های سازنده[۲]، به‌راحتی گونه‌های مختلف فراریخت با صفات موروثی مشترک از یک ویروس تولید می‌شود. مدل نمایه پنهان مارکوف در بیوانفورماتیک برای یافتن توالی‌های ژنتیکی مرتبط به خانواده استفاده می‌شود. در این پژوهش، ما مشخص می‌کنیم که آیا PHMM می‌تواند برای شناسایی انواع خانواده ویروس فراریختی تولیدشده از چهار کیت سازنده استفاده شود یا خیر؟ هر کیت سازنده دارای رفتار متنوع است و از این‌رو باید مدل‌های مختلف PHMM را با دسته‌بندی صفات موروثی هر کیت سازنده تولید کنیم. به این ترتیب، مدل‌ها بر اساس احتمال رخداد تمام آپکدها[۳] در ویروس‌های مختلف محاسبه می‌شود [۳۴]. سپس ما به طبقه‌بندی فایل‌های ویروس و غیر ویروس با امتیازدهی آن‌ها در برابر این مدل‌ها با استفاده از الگوریتم ویتربی[۴] می‌پردازیم. تکامل ویروس‌های کامپیوتری نشان می‌دهد که آن‌ها روزبه‌روز هوشمندانه‌تر می‌شوند. ویروس‌های امروز وب سایت‌های اینترنتی را هدف قرار می‌دهند تا به سرعت در سراسر جهان گسترش یابند. در گذشته تولید ویروس‌ها نیاز به مهارت‌های برنامه‌نویسی زبان اسمبلی داشت، اما به تازگی به دلیل ورود کیت‌های سازنده ویروس‌های مختلف و موتورهای جهش، هر کاربر با حداقل یا بدون دانش از ساختار ویروس‌ها می‌تواند گونه‌های بسیار پیچیده و خطرناک جدیدی از ویروس‌های شناخته‌شده ایجاد کند. مشهورترین تکنیک تشخیص ویروس که امروزه استفاده می‌شود تشخیص امضا است [۵]، که به دنبال رشته‌های منحصربه‌فرد مربوط به ویروس شناخته شده است و پس از تشخیص آن، اگر امضاء آن‌ها در سیستم به‌روزشده حفظ شود، ویروس دیگر هیچ تهدیدی محسوب نمی‌شود. برای جلوگیری از تشخیص، نویسندگان ویروس به جای ایجاد ویروس‌های جدید، ویروس‌های قدیمی را تغییر دادند. این روند به ویروس‌های رمزگذاری‌شده[۱] که در

---

[۱] signature

[۲] construction kits

[۳] opcodes

[۴] Viterbi

[۱] encrypted viruses

هر بار انتشار[۲] از کلید متفاوت دیگری استفاده می‌کنند، تکامل یافته است، اما اغلب این ویروس‌ها امضاء خاصی در رمزگشای[۳] خود دارند. از سوی دیگر، ویروس‌های چندریختی[۴] از طرح‌های رمزنگاری تصادفی استفاده کردند و به استفاده از رمزگشای‌ها چندریختی ارتقاع‌یافته مبادرت نموده‌اند. اگر چه نویسندگان ویروس به‌طور چشمگیری کد ویروس را تغییر می‌دهند ولی بیشتر این ویروس‌ها همچنان می‌توانند با استفاده از تشخیص امضاء زمانی که آن‌ها رمزگشایی می‌شوند، شناسایی شوند [۱، ۳۵، ۳۶]. ویروس‌های فراریخت باعث تغییر کل کد ویروس بدون تغییر تأثیر آن می‌شوند. تکنیک‌های ابهام‌زایی کد مانند افزودن کد زائد، مرتب‌سازی مجدد کد[۵] و جایگشت‌های زیرروال[۶] برای تولید انواع مختلفی که متعلق به یک خانواده ویروس هستند، استفاده می‌شود [۱، ۱۲، ۱۳، ۳۶–۳۹]. در حال حاضر، تولید انواع جدید ویروس فراریخت با استفاده از کیت‌های سازنده آسان‌تر شده است، اما شناسایی آن‌ها یک چالش است. تشخیص امضاء موثر نیست زیرا هر نوع، دارای یک رشته پویشی[۷] متفاوت است. سایر روش‌های ضد ویروس مانند شبیه‌سازی کد[۸] و روش‌های اکتشافی[۹] می‌توانند برای تشخیص آن‌ها استفاده شوند اما کارایی زمانی ندارند. مدل‌های پنهان مارکوف در زمینه تشخیص گفتار[۱۰] بسیار شناخته شده‌اند [۷، ۴۰]. دیگر برنامه‌های کاربردی شامل مدل‌سازی پروتئین‌ها برای خانواده‌های پروتئینی و الگوهای متصل به‌هم تابیده RNA هستند [۴۱]. با استفاده از مدل‌های پنهان مارکوف برای تشخیص ویروس‌های فراریختی نتایج چشمگیری به دست آمده است [۱، ۳۶، ۳۸، ۴۲]. در این پروژه، ما تعیین

---

۲ propagate

۳ decryptors

۴ polymorphic viruses

۵ code reordering

۶ subroutine permutations

۷ scan string

۸ code emulation

۹ heuristics

۱۰ speech recognition

می‌کنیم که آیا یک نمونه ویژه از مدل‌های پنهان مارکوف، که مدل‌های نمایه پنهان مارکوف (PHHMs) نامیده می‌شود، می‌تواند در شناسایی گونه‌های فراریختی یک ویروس استفاده شود و نتایج بهینه‌ای را فراهم آورد یا خیر؟ مدل‌های پنهان مارکوف در بیوانفورماتیک برای یافتن توالی‌های مرتبط با فاصله از یک خانواده رشته پروتئین استفاده می‌شوند [۴۳، ۴۴]. هدف ما این است که با استفاده از PHMM یک خانواده ویروس فراریختی را مدل و فایل‌های ویروس و غیر ویروس را با استفاده از آن ارزیابی کنیم. یک مدل PHMM شامل مجموعه‌ای از احتمالات است و با استفاده از توالی هم‌ترازی آپکدها از انواع مختلف خانواده ویروس ایجاد می‌شود. پس از آن ما با توجه به امتیاز شباهت به مدل که با استفاده از الگوریتم ویتربی محاسبه و اندازه‌گیری می‌شود، فایل‌های ویروس و غیر ویروس را تشخیص می‌دهیم. در ادامهٔ این نوشتار به اختصار به بیان ساختار پایان‌نامه و اهداف پیش روی آن می‌پردازیم.

۲.۱ ساختار و اهداف پژوهش

بدافزارها انواع مختلفی دارند که در فصل دوم، به تعریف آنها می‌پردازیم. اولین نوع بدافزاری که به وجود آمد، به نام ویروس[۱]، موسوم است [۴۵]. این نام‌گذاری بدین دلیل است که کار آن بسیار شبیه به ویروس‌های بیولوژی است. ویروس کامپیوتری کدی است که به تنهایی نمی‌تواند کاری انجام دهد، این کد پس از آن‌که به یک برنامه تزریق شود، با اجرای آن برنامه، اجرا می‌شود. به این برنامه، برنامهٔ آلوده‌شده می‌گوییم. این کد خصوصیت دیگری که دارد، این است که پس از اجرای برنامهٔ آلوده‌شده، می‌تواند خود را تکثیر نماید و دیگر برنامه‌ها را نیز آلوده نماید. پس از آن بدافزارهای قوی‌تری از جمله کرم‌ها[۲] و روت‌کیت‌ها[۳] به وجود آمدند که قدرت‌های زیادی داشتند. تاکنون برای شناسایی انواع بدافزار روش‌های متفاوتی به وجود آمده است، که در این تحقیق به برخی از آن‌ها اشاره

---

[۱] virus

[۲] worms

[۳] rootkits

شده است. روش‌های قدیمی تشخیص بدافزارها مبتنی بر تشخیص بر اساس تطابق امضای بدافزار بودند. امضای یک بدافزار چیزی شبیه به اثر انگشت است، به عبارتی دیگر یک مشخصه منحصربه‌فرد برای ویروس مورد نظر است. روش‌های قدیمی که مبتنی بر امضاء[۱] بودند، برای بدافزارهای جدید و ناشناخته، مواقعی که امضای بدافزار موجود نباشد، با شکست روبرو می‌شوند و توانایی تشخیص ندارند. از طرفی برخی از انواع بدافزارها وجود دارند که به‌ازای هر سرایت به کلی تغییر می‌کنند و استخراج امضای آن‌ها بسیار دشوار و گاهی غیر ممکن می‌شود. از طرفی روند استخراج امضای بدافزار بدین صورت است که ابتدا بایستی یک خرابی به وجود آید و به‌وجود خرابی پی ببریم و بعد از آن توسط سیستم‌های آزمایشی که مخصوص این کار هستند، فایل یا فایل‌های عامل مخرب کشف می‌شود. پس از عملیات کشف بایستی فایل‌های به‌دست‌آمده در آزمایشگاه توسط متخصصین بررسی و امضای بدافزار استخراج شود و ضد ویروس‌ها خود را به‌روز نمایند تا بتوانند بدافزار جدید را تشخیص دهند، که درطول این روند بدافزار کار خود را توسعه می‌دهد و خرابی بیشتری در سیستم‌های کامپیوتری به وجود می‌آورد. لذا به روشی که بتواند بدافزار جدید را در بدو انتشار شناسایی نماید [۴۶]، احساس نیاز می‌شود. در ادامه فصل دوم به روش‌های تشخیص موجود، به تفکیک استراتژی تشخیص و روش تحلیل، پرداخته شده است. برخی از روش‌های مبتنی بر داده‌کاوی[۲] یک نمایه[۳] از رفتار برنامه‌های معمولی را با استفاده از الگوریتم‌های یادگیری ماشین، ایجاد می‌کنند. پس از آن برای نمونه‌های آزمایشی، هر برنامه‌ای که از این نمایه به اندازه یک آستانه[۴] از قبل تعریف‌شده، فاصله داشته باشد، به‌عنوان برنامه ناهنجار[۱] گزارش می‌شود. از این استراتژی به‌عنوان استراتژی تشخیص ناهنجاری، یاد می‌شود. البته استراتژی دیگری نیز وجود دارد، و آن این است که نمایه ذکرشده را برای بدافزارها

---

[۱] signature-based

[۲] data mining

[۳] profile

[۴] threshold

[۱] anomal

ایجاد می‌کنیم و هر شباهت با این نمایه را به‌عنوان بدافزار گزارش می‌کنیم. به استراتژی مذکور، تشخیص سوء استفاده[۲] می‌گوییم.

در فصل سوم، به تحقیقات دیگر پژوهشگران که در زمینه شناسایی بدافزار با کمک داده‌کاوی، تحقیقاتی انجام داده‌اند، از دیدگاه‌های مختلفی پرداخته شده است. روش‌هایی که اغلب محققان به کار برده‌اند به چند دستهٔ کلی تقسیم می‌شوند. دسته اول روش تطابق الگو[۳] یا پویش[۴] است. در این روش به‌طور کلی در فایل مورد نظر، به دنبال یک الگوی خاص می‌گردیم. این الگو می‌تواند امضای یک بدافزار خاص یا این‌که یک الگوی پیچیده باشد. روش پویش در تمامی ضد ویروس‌های تجاری به کار می‌رود. این روش یک روش مؤثر، کارا و با سرعتی بالاتر از سایر روش‌ها است. به دلیل این‌که الگویی که به این روش جهت پویش داده می‌شود، عموماً امضای بدافزارها است و از طرفی امضای بدافزار با حساسیت خاصی استخراج می‌شود تا منحصربه‌فرد باشد. لذا این روش روشی است که خطای بسیار کمی دارد و این خطا در حد صفر است. اما این روش مشکلاتی نیز دارد که سبب به‌وجودآمدن روش‌های دیگری شده است. از زمانی که بدافزارهای چندریختی و فراریختی به وجود آمدند، رشد روش‌های دیگر سرعت بیشتری گرفت. اولین مشکل عمده‌ای که این روش دارد این است که فقط بدافزارهایی را شناسایی می‌کند که قبلاً کشف شده باشند؛ امضای آن‌ها در آزمایشگاه‌های تخصصی توسط متخصصین این امر استخراج شده باشد و نرم‌افزار ضد ویروس این امضاء را نیز داشته باشد. به عبارتی دیگر در صورتی که بدافزار جدید به وجود بیاید، این روش از شناسایی آن عاجز است، چون امضای آن را ندارد. مشکل دیگری که این روش دارد این است که نیروی انسانی و زمان و هزینهٔ زیادی صرف استخراج امضاء می‌شود. مشکل سوم آن این است که این روش اغلب در مقابل بدافزارهای فراریخت، ناتوان است. به‌دلیل این‌که بدافزارهای فراریخت می‌توانند

---

[۲] misuse detection

[۳] pattern matching

[۴] scan

کد خود را به ازای هر سرایت یا هر رخداد دیگری که نیاز باشد تغییر دهند، لذا استخراج یک الگو که بتواند تمامی این تغییرات را طوری پوشش دهد که خطای تشخیص نداشته باشیم، بسیار دشوار و اغلب نا ممکن است. به خاطر وجود این مشکلات، که مشکلات کوچکی هم نیستند، روش‌های مکاشفه‌ای به وجود آمدند. این روش‌ها بایستی بتوانند زمانی که یک بدافزار به وجود می‌آید، آن را در بدو انتشار شناسایی نماید. برای این کار روش‌هایی که پیشنهاد شده است اغلب از داده‌کاوی بهره می‌برند. این روش‌ها معمولاً بدین صورت کار می‌کنند که ابتدا یک مجموعه ویژگی از فایل‌های بدافزار را، حال با آنالیزهای مختلف، استخراج نموده، سپس با استفاده از یادگیری ماشین سعی می‌کنند رفتار یا نیت بدافزارها را یاد بگیرند. پس از آن با استفاده از کلاسه‌کننده‌ها اقدام به تفکیک بدافزار جدید از برنامه‌های پاک می‌کنند. در این پژوهش روش‌های داده‌کاوی که مبتنی بر رفتار[۱] بدافزار هستند، بررسی شده است. روش کار بدین صورت است که ابتدا مجموعه‌ای از فایل حاوی انواع خانواده از بدافزارهای چندریختی و فراریختی و برنامه‌های پاک را جمع‌آوری و روی ویژگی‌های آن‌ها کار می‌شود، سپس ویژگی‌هایی که برای تشخیص مناسب‌تر هستند، انتخاب و پس از اعمال پیش‌پردازش‌های لازم، به‌عنوان ویژگی‌های رفتاری بدافزار و برنامه‌های پاک در نظر گرفته می‌شود. در پایان با استفاده از آن ویژگی‌ها، مدل‌های کلاسه‌کننده را می‌سازیم که با استفاده از این مدل‌ها می‌توانیم بدافزارهای جدید و ناشناخته را شناسایی نماییم. پس از بررسی این موارد، روش پیشنهادی نیز بررسی می‌شود که این روش از استراتژی تشخیص مبتنی بر الگوی آماری[۲] استفاده می‌کند و تحلیل آن نیز ایستا است. روش پیشنهادی از مدل نمایه پنهان مارکوف برای مدل‌کردن تمام فایل‌های متعلق به یک خانواده از ویروس استفاده نموده است، تا بتوان برنامه‌های خوب و بد را از لحاظ معنایی[۱] از هم تفکیک نماید. این روش می‌تواند اغلب بدافزارهای چندریختی و فراریختی را شناسایی نماید. این موارد در فصل چهارم، ذکر می‌شود.

---

[۱] behavior-based

[۲] statistical-based

[۱] semantic

در فصل پنجم، روال جمع‌آوری داده، آزمایش‌های انجام‌شده و مشکلاتی که بر سر این راه قرار دارد، بررسی شده است. یکی از مشکلاتی که در زمینه تحقیق روی بدافزارها وجود دارد و آن را با دیگر کارها کمی متمایز می‌نماید، این است که یک یا چند مجموعه دادهٔ عمومی وجود ندارد تا همه محققان کار خود را روی آن آزمایش نمایند و بتوانند ادعای خود را به خوبی به اثبات برسانند. در واقع نمی‌توان چنین مجموعه‌ای داشت و علت هم این است که بدافزارها روزبه‌روز در حال تولید هستند و اغلب بدافزارهای به‌وجودآمده از نسل‌های قبلی خود تواناتر و قوی‌تر هستند. در برخی از موارد برای شناسایی، نیازمند به‌کارگیری روش جدیدی هستیم. علت اصلی این امر، این است که بدافزارها عموماً توسط اشخاصی با ضریب هوشی بالا یا سازمان‌های قدرتمند ساخته می‌شوند که در هر دو حالت پیچیده‌اند و از الگوی خاصی تبعیت نمی‌کنند. به عبارت دیگر بدافزارهای استانداردی نمی‌توان انتخاب نمود که همهٔ پژوهشگران کار خود را با آن بسنجند. از طرفی دیگر، بدافزارها فایل‌های خطرناکی هستند و نمی‌توان آن‌ها به راحتی در اختیار همه قرار داد و تنها مراکز آموزشی و پژوهشی امکان دسترسی به این مجموعه داده را دارند. اغلب محققین برای اثبات کار خود یا از معادلات ریاضی استفاده می‌کنند و یا این‌که مجموعه داده خود را با نرم‌افزارهای ضد ویروس معروف نیز آزمایش می‌کنند و نتیجهٔ کار خود را با آن‌ها مقایسه می‌نمایند.

فصل ششم، فصل پایانی این نوشتار است. در این فصل بر اساس نتایج به‌دست‌آمده از آزمایش‌ها، نتیجه‌گیری‌هایی کردیم و پیشنهادهایی نیز ذکر نموده‌ایم. به عقیده ما استفاده از روش‌های داده‌کاوی شامل دو مسئله چالش‌زا در شناسایی بدافزارها می‌شود؛ یکی این‌که خود الگوریتم داده‌کاوی را توسعه دهیم و دیگری این‌که مجموعه ویژگی بهتری تولید نماییم. در روش پیشنهادی به نحوی هر دو مسئله بررسی شده‌اند. به مسئله اول در فاز یادگیری مدل ارائه‌شده و در روند ایجاد هم‌ترازی توالی‌های چندگانه پرداخته‌ایم. در زمان ایجاد هم‌ترازی توالی‌های چندگانه، توالی‌های اجماعی برای آن ایجاد می‌نماییم که با به‌کارگیری و اضافه‌کردن آن‌ها به MSA تولید می‌شود و با ساختن مدل نمایه پنهان مارکوف از روی آن، عملکرد مدل ارائه‌شده را بهبود بخشیده‌ایم. مسئله دوم در سال‌های اخیر بیش‌تر

مورد توجه پژوهشگران در این زمینه بوده است و مهم‌تر به نظر می‌رسد، چرا که الگوریتم‌های داده‌کاوی موجود در حد خوبی کار می‌کنند و کافی است مجموعه ویژگی مناسب‌تری را به این الگوریتم‌ها داد. از این رو بیش‌تر پژوهش‌هایی که در زمینه تشخیص بدافزار با استفاده از داده‌کاوی انجام شده است روی انتخاب ویژگی‌های بهتر و مؤثرتر تمرکز داشته‌اند. در این تحقیق با توجه به این‌که بدافزارهای امروزی بسیار هوشمندتر و قوی‌تر از قبل شده‌اند، برای داشتن مجموعه ویژگی بهتر و کارآمدتر از ویژگی‌های معنایی استفاده می‌شود. بدافزارهایی که کد فراریخت دارند، به ازای هر سرایت به‌طورکلی کد خود را تغییر می‌دهند. به‌طوری‌که شناسایی این نوع از بدافزار توسط روش‌های مبتنی بر امضاء بسیار دشوار و نیازمند ساختن امضاهای پیچیده است که گاهی به کلی شناسایی ناممکن می‌شود. در این نوع بدافزار تنها چیزی که موقع سرایت از ماشینی به ماشین دیگر ثابت می‌ماند، مفهوم و هدف بدافزار است. لذا اگر بتوان از روشی استفاده کرد که مفهوم و هدف بدافزارها را دریابد، می‌توان تا حد زیادی آن را شناسایی نمود. محققان به‌تازگی از رویکردهای مختلفی برای داشتن درکی مفهومی از برنامه‌ها استفاده کرده‌اند. بهره‌گیری از روش‌های مهندسی معکوس برای استخراج کدهای عملیاتی و ایجاد دنباله و زیردنباله از آن‌ها برای ساختن مدلی آماری یکی از بهترین رویکردهایی است که می‌توان آن را برای تشخیص بدافزارها با استفاده از رفتار و عملکرد بدافزار، به‌کار برد.

فصل دوم

تعاریف و مفاهیم اولیه

در این بخش اصطلاحات مرسوم در دنیای امنیت کامپیوتر به خصوص اصطلاحات مرتبط با کرم‌ها و ویروس‌ها توضیح داده خواهد شد.

۱.۲ بدافزار

هر برنامهٔ کامپیوتری را که به قصد آسیب‌رساندن به سیستم‌ها و اطلاعات و عملیات کامپیوتری ساخته شود، بدافزار می‌نامند. بدافزارها شامل ویروس‌ها، کرم‌ها، تروجان‌ها[۱]، درب‌های پشتی[۲]،

---

[۱] trojans

[۲] backdoors

تبلیغ‌افزارها[۱]، جاسوس‌افزارها[۲]، روبات‌ها[۳]، روت‌کیت‌ها، باج‌افزارها[۴] و ... هستند. به‌طور معمول از تمامی بدافزارها در عرف با نام ویروس یاد می‌شود و اغلب ضد بدافزارهای تجاری را نیز با نام ضد ویروس می‌شناسند [۴۷]. بدافزار یا کد مخرب یک تهدید با سرعت رشد و تکامل زیاد برای دنیای کامپیوتر و وسایل هوشمند الکترونیکی است. بدافزارهای روز نخست[۵]، از نقاط ضعف و آسیب‌پذیر بهره می‌گیرند و به‌سرعت منتشر می‌شوند و گسترش پیدا می‌کنند. هرچند ابزارهای ضد ویروس به‌طور گسترده‌ای استفاده می‌شوند، اما قادر به مقابله با آن‌ها نیستند. ضد ویروس‌ها بر اساس امضاء هستند که باید برای گونه‌های جدید بدافزار محاسبه و تولید و به‌روز شوند. پس از این‌که بدافزار جدید ظاهر می‌شود قبل از پیداشدن امضاء آن، زمان کافی برای خرابکاری و آسیب‌رساندن دارد [۴۸]. بدافزارها انواع گوناگونی دارند که در زیر به آن اشاره کرده‌ایم.

۲.۲ انواع بدافزار

۱.۲.۲ هرزنامه

سوءاستفاده از سیستم پیام‌رسانی الکترونیکی برای ارسال پیام‌های ناخواسته و با هدفی خاص را ارسال هرزنامه می‌نامند. هرزنامه‌ها می‌توانند در پست الکترونیکی، وبلاگ‌ها، انجمن‌ها، پیامک‌های تلفن همراه و غیره به کار روند [۴۹].

---

[۱] adwares

[۲] spywares

[۳] botnets

[۴] ransomwares

[۵] zero-day malware

۲.۲.۲ تروجان

تروجان (یا اسب تروا) یک بدافزار است که با ظاهری شبیه به یک برنامه معمولی اهداف بدخواهانهٔ خود را عملی می‌کند. این نام از افسانه اسب تروا گرفته شده است [۴۷، ۵۰].

۳.۲.۲ روت‌کیت

مجموعه برنامه‌هایی هستند که از طریق نفوذ به مؤلفه‌های سیستم عامل به نفوذگران امکان انجام کارهای بدخواهانهٔ خود را می‌دهند. روت کیت‌ها گاهی به هستهٔ سیستم عامل نیز نفوذ می‌کنند [۴۵، ۴۷، ۵۰].

۴.۲.۲ ویروس

ویروس کد مخربی است که خود را به داخل برنامه‌های دیگر کپی می‌کند. در واقع به تنهایی نمی‌تواند کاری انجام دهد و توسط برنامه‌ای که به این ویروس آلوده شده است، اجرا می‌شود. معمولاً ویروس‌ها با دخالت کاربر اجرا می‌شوند [۴۵، ۴۷].

- دسته‌بندی ویروس‌ها بر اساس هدف

۱. ویروس بوت سکتور[۱]

master boot record تکه کدی است که هر زمان کامپیوتر راه‌اندازی (بوت) شود، اجرا می‌شود. ویروس بوت سکتور ویروسی است که کد خود را به این قسمت از دیسک تزریق می‌کند و هر زمان که کامپیوتر روشن می‌شود، این ویروس نیز اجرا می‌گردد.

۲. ویروس فایل

[۱] boot sector virus

بیشتر ویروس‌ها از این نوع هستند. این نوع ویروس فایل‌های اجرایی و سیستمی را آلوده می‌کند و با هر بار اجرای برنامهٔ آلوده‌شده، اجرا می‌شود.

۳. ویروس ماکرو[۱]

این نوع ویروس به جای این‌که فایل‌های اجرایی را آلوده نماید، اسناد و قالب‌ها را آلوده می‌کند. این ویروس به زبان ماکروی برنامه مفسر سند، از قبیل Microsoft word، VB script و HTML و غیره نوشته می‌شود و هر بار که سند باز می‌گردد، اجرا می‌شود.

- دسته‌بندی بر اساس استراتژی حفاظت از خود

استراتژی حفاظت از خود به مجموعه راهکارهایی اطلاق می‌شود که ویروس برای جلوگیری از شناسایی توسط نرم‌افزارهای ضد ویروس به کار می‌برد. بر اساس استراتژی‌های مختلف ویروس‌ها به دسته‌های زیر تقسیم می‌شوند [۱، ۱۲].

۱. بدون حفاظت

این ویروس هیچ‌گونه محافظتی از کد خود به عمل نمی‌آورد و به راحتی شناسایی می‌شود. امروزه این نمونه از ویروس‌ها بسیار کم پیدا می‌شوند.

۲. ایجاد ابهام در کد

در این روش نویسندهٔ ویروس مجموعه اعمالی را در کد انجام می‌دهد تا از شناسایی مبتنی بر امضای ایستا جلوگیری کند. این اعمال شامل افزودن دستورات بی‌مورد و پرش‌های غیرضروری و غیره می‌باشد.

۳. رمزنگاری کد

در این روش کد ویروس موقع سرایت به وسیلهٔ یک موتور رمزکننده، به‌صورت رمز در می‌آید و برای اجرا رمزگشایی می‌شود. به ازای هر سرایت با یک کلید تصادفی به رمز

---

[۱] macro

در می‌آید و خروجی‌های متفاوتی تولید می‌کند، بنابراین توسط روش‌های مبتنی بر امضاء شناسایی نمی‌شود.

۴. چندریختی

در ویروس‌هایی که از کدهای رمزشده استفاده می‌کنند، بخشی از کد که مربوط به موتور رمزکننده می‌شود، ثابت می‌ماند و از همین بخش می‌توان امضای ویروس را استخراج و آن را با روش تشخیص مبتنی بر امضاء شناسایی نمود. در ویروس‌های چندریختی موتور رمزگشای کد نیز ثابت نمی‌ماند و به ازای هر سرایت تغییر می‌کند.

۵. فراریختی

ویروس‌های فراریخت خودشان را کاملاً تغییر می‌دهند، به‌طوری که نمونه جدید هیچ شباهتی به ویروس اصلی ندارد. این نوع ویروس هیچ نوع موتور رمزکننده‌ای ندارد و به صورت خودکار هنگام سرایت، دگردیسی در کد ویروس رخ می‌دهد.

۵.۲.۲ کرم

کرم یک برنامه کامپیوتری است که به‌صورت خودکار در شبکه منتشر می‌شود و از نقاط ضعف سیستم‌ها برای مقاصد بدخواهانه استفاده می‌کند. دن الیس[۱] سال ۲۰۰۳ در مقاله خود برای کرم‌ها چرخهٔ زندگی تعریف می‌کند. با توجه به مجموعه فعالیت‌هایی که کرم در هر فاز از چرخه زندگی خود انجام می‌دهد، می‌توان کرم‌ها را به گروه‌های مختلفی دسته‌بندی نمود [۵۱].

- فعال‌شدن[۲]

فعال‌شدن اولین فاز چرخهٔ زندگی کرم است و منظور از آن شروع به کار کرم روی سیستم قربانی

---

[۱] Dan Ellis

[۲] activation

است. بر اساس روش‌هایی که کرم برای فعال‌شدن به کار می‌برد، می‌توان آن‌ها را به چند دسته تقسیم‌بندی نمود.

۱. فعال‌سازی به وسیلهٔ کاربر

در این روش کرم به وسیلهٔ کاربر اجرا می‌شود که نیز کندترین روش است.

۲. فعال‌سازی مبتنی بر فعالیت‌های کاربر

در این روش کرم مستقیم توسط کاربر فعال نمی‌شود، بلکه اجرای کرم به برخی از اعمال کاربر وابسته است و در واقع به‌طور غیرمستقیم توسط کاربر اجرا می‌شود.

۳. فعال‌سازی زمان‌بندی‌شده

این نوع فعال‌سازی با رسیدن به زمان خاص به‌طور خودکار انجام می‌شود.

۴. فعال‌سازی خودکار

این روش سریع‌ترین روش فعال‌سازی است که در آن کرم به‌صورت خودبه‌خود اجرا می‌شود. برای این کار از ضعف امنیتی برنامه‌هایی مثل پایگاه‌داده یا سرویس‌دهنده وب استفاده می‌شود که معمولاً همیشه در حال اجرا هستند.

- اَعمال پس از آلودگی[۱]

در فاز بعد، یعنی زمانی که کرم در کامپیوتر قربانی فعال شد، اعمالی را انجام می‌دهد که این اعمال به دسته‌های زیر تقسیم می‌شوند.

۱. هیچ[۲]

هیچ کار خاصی انجام نمی‌دهد. فقط با افزایش بار شبکه و بار پردازشی سیستم کاربر، سرعت و کارایی را پایین می‌آورد.

---

[۱] payload

[۲] none

۲. کنترل از راه دور از طریق اینترنت[۱]

برخی از کرم‌ها یک درب پشتی روی ماشین قربانی ایجاد می‌کنند و اجازهٔ دسترسی و کنترل از راه دور از طریق اینترنت را به نفوذگر می‌دهند.

۳. پخش‌کننده هرزنامه[۲]

این نوع کرم ماشین قربانی را به یک سرور ارسال‌کننده هرزنامه تبدیل می‌کند.

- کشف هدف[۳]

یکی از کارهایی که کرم پس از آلودگی انجام می‌دهد، یافتن هدف جدیدی برای سرایت است.

۱. کرم‌های پویشگر[۴]

این نوع کرم‌ها مجموعه آدرسی را به‌صورت ترتیبی یا تصادفی بررسی می‌کنند تا اهداف خود را پیدا نمایند.

۲. کرم‌های سریع[۵]

این نوع کرم معمولاً دارای لیست آدرسی است که از قبل تولید شده است که با استفاده از این لیست عملیات یافتن اهداف سرعت بیشتری می‌یابد.

۳. کرم‌های توپولوژی[۶]

این کرم‌ها سعی می‌کنند تا انجام جست‌وجو در یک لیست از ماشین‌های میزبان که با برنامه‌های کاربری در ارتباط هستند، توپولوژی شبکه را پیدا کنند.

---

[۱] remote control by internet

[۲] spam-relays

[۳] target discovery

[۴] scanner worms

[۵] flash worms

[۶] topological worms

۴. کرم‌های منفعل[۱]

این کرم به مداخلهٔ کاربر در انتقال به ماشین هدف اکتفا می‌کند و منتظر می‌ماند تا کاربر به‌طور ناخواسته کرم را به ماشین دیگری منتقل نماید.

۵. کشف هدف با استفاده از سرور هدایت‌کننده[۲]

این نوع کرم لیست آدرس اهداف را از یک سرور خارجی دریافت می‌کند.

- انتشار[۳]

منظور از انتشار، مجموعه سازوکارهایی است که کرم برای گسترش خود در شبکه انجام می‌دهد. بر اساس روش انتشار کرم‌ها به دسته‌های زیر تقسیم می‌شود.

۱. خودکپی[۴]

معمولاً خود کرم عمل کپی‌کردن خودش را انجام می‌دهد.

۲. دو مرحله‌ای[۵]

پس از سرایت، اتصالی از ماشین قربانی به مبدأ ایجاد و سپس بدنه اصلی کرم را از آن دریافت می‌کند.

۳. توکار[۶]

این نوع کرم خود را در یک جریان انتقال معمولی مخفی می‌کند و با آن جریان به ماشین دیگر منتقل می‌شود.

---

[۱] passive worms

[۲] metaserver worms

[۳] propagation

[۴] self-carried

[۵] second channel

[۶] embedded

۳.۲ ویروس فراریخت

۱.۳.۲ منشاء ویروس‌ها

ویروس‌ها اولین‌بار به‌عنوان برنامه‌های خودتکرارکننده[۱] در دانشگاه‌ها به منظور شوخی و اذیت‌کردن دانشجویان دیگر انتشار یافتند، اما بیش‌تر آن‌ها بی‌ضرر بودند. اگرچه ویروس‌ها در اوائل دهه ۱۹۸۰ وجود داشتند، در ۱۹۸۸ زمانی که کامپیوترهای شخصی پا به عرصه وجود گذاشتند، برای فعالیت‌های مخرب خود با ظهور کرم موریس[۲] محبوبیت پیدا کردند. کرم‌ها خودشان را منتشر می‌کنند اما ویروس‌ها برای گسترش[۳] به کمک نیاز دارند. رابرت ت. موریس جونیور[۴] ، نویسنده کرم موریس، از اینترنت برای گسترش و آلوده‌کردن سیستم‌های بسیاری استفاده کرد. این امر باعث شد تا کل اینترنت به وسیلهٔ یک حمله انکار سرویس[۵] متوقف شود که موجب وحشت گسترده و آگاهی از ویروس‌ها شد. ویروس‌های دیگر که در این زمان وجود داشتند، مانند لای[۶] ، مغز[۷] و اورشلیم[۸] ، فایل‌های بخش‌های بوتی[۹] یا برنامه‌ها را هدف قرار دادند. بعضی از ویروس‌هایی که در اواخر دهه ۱۹۸۰ و اوایل دهه ۱۹۹۰ ظهور یافتند، آثار مخرب زیادی به همراه داشتند. رفتار مخرب ویروس زمانی شروع می‌شود که شرایط مشخص تعبیه‌شده در کد آن مهیا گردد.

---

۱ self-replicating

۲ Morris worm

۳ spread

۴ Robert T. Morris, jr.

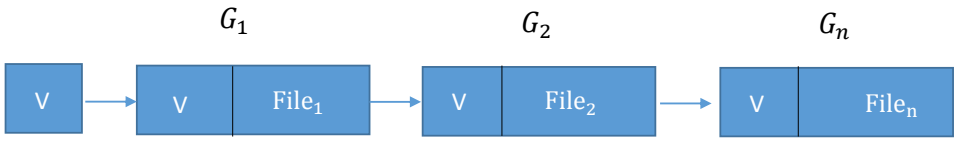
۵ denial of service attack

۶ Leigh

۷ Brain

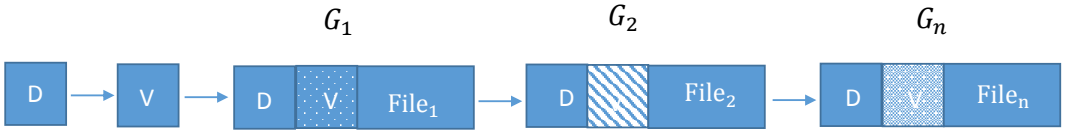
۸ Jerusalem

۹ boot sectors



شکل ۱.۲: نسخه‌برداری ویروس ساده [۱، ۱۳].

یکی از اهداف اصلی ویروس به غیر از آسیب‌رساندن، پنهان ماندن از دید برنامه‌های ضد ویروس است. تشخیص امضاء یک تکنیک ضد ویروس مشهور است که در تشخیص این ویروس‌ها استفاده می‌شود. نوشتن ویروس‌های جدید از ابتدا بسیار دشوار و وقت‌گیر است، از این رو اکثر نویسندگان ویروس تلاش می‌کنند تا ویروس‌های موجود را با رفع اشکالاتشان و ایجاد ابهام در آن‌ها افزایش دهند. این ممکن است امضای ویروس والد را تغییر ندهد، بنابراین آن‌ها هنوز تشخیص دادنی هستند. برای جلوگیری از تشخیص، نویسندگان ویروس اقدام به مخفی‌کردن و تغییر کد ویروس کردند. رمزگذاری ویروس‌ها، آن‌ها را تغییر می‌دهد اما آن‌ها در بلوک رمزگشای خود امضاء دارند.

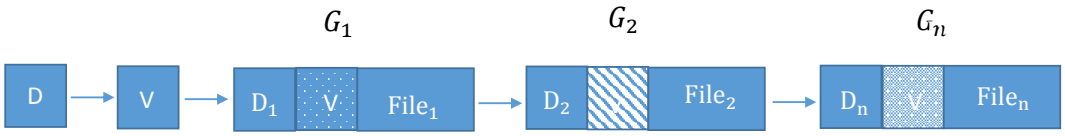


شکل ۲.۲: نسخه‌برداری ویروس رمزگذاری شده [۱، ۱۳].

اما امضاهای گرفته‌شده از رمزگشای می‌تواند منجر به علامت‌گذاری غیر ویروس‌هایی با بلوک‌های رمزگشای مشابه شود که باعث افزایش مثبت کاذب[۱] می‌شود. دیگر موارد پیچیده شامل رمزگشای

[۱] false positives

غیرخطی[۱] و حذف کد رمزگشای از ویروس است. ویروس‌های نیمه‌چندریختی[۲] با تقسیم رمزگشای خود به بخش‌های متعدد و یا با دستورالعمل مرتب‌سازی مجدد یک گام فراتر رفته‌اند. تغییرات در نسخه‌های کپی ویروس نیمه‌چندریختی زیرکانه و دقیق هستند اما هنوز هم شامل یک رشته ثابت برای جست‌وجو هستند. پس چگونه رمزگشای‌ها ظاهری بسیار متفاوت از یک‌دیگر پیدا کنند؟ پاسخ در چندریختی‌ها است. ویروس‌های چندریخت کد رمزگشای خود را با استفاده از تکنیک‌های مبهم‌سازی مانند افزودن کد زائد[۳] و ارائه کد معادل[۴] تغییر می‌دهند. مبهم‌سازی و رمزگذاری چندگانه[۵] می‌توانند میلیون‌ها نسخه تولید کنند و از این رو هر نسل جدید یک گونهٔ جدید چندریختی ایجاد می‌کند. در سال ۱۹۹۰ مارک واشبرن[۶] نخستین ویروس چندریختی شناخته‌شده را به‌نام «۱۲۶۰» نوشت که از تکنیک افزودن کد زائد برای تغییر طول رمزگشای خود استفاده می‌نمود. به نظر می‌رسد ویروس‌های چندریختی مورد توجه نویسندگان ویروس هستند، زیرا امروزه بیشتر از هرگونه ویروس دیگری وجود دارند.



شکل ۳.۲: نسخه‌برداری ویروس چندریخت [۱، ۱۳].

---

۱ non-linear decryption

۲ oligomorphic

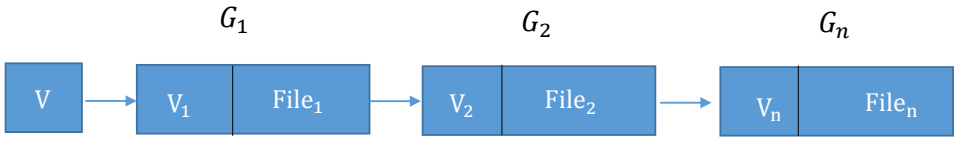
۳ garbage code insertion

۴ equivalent code submission

۵ multiplayer encryption

۶ Mark Washburn

ضعف اصلی ویروس‌های چندریختی این است که بدن ویروس تغییر پیدا نمی‌کند، با وجود پیچیدگی آن‌ها می‌توان آن‌ها را با استفاده از یک شبیه‌ساز رمزگشایی و شناسایی کرد. اگرچه شبیه‌سازی و رمزگشایی آن‌ها ممکن است سخت و خسته‌کننده باشد، ولی ناممکن نیست. امروزه بعضی ویروس‌ها از تکنیک‌های ضد شبیه‌سازی مانند محاسبات غیرضروری[۱] استفاده می‌کنند اما یک اشکال‌زدای[۲] با تجربه می‌تواند بر آن غلبه کند. آیا ویروس می‌تواند به جای جهش موتور رمزگشایش خود را تغییر دهد؟ این دقیقاً همان چیزی است که ویروس «فراریخت» انجام می‌دهد. ویروس فراریخت کل بدنهٔ ویروس را مبهم می‌کند و در نتیجه میلیون‌ها شکل متفاوت از یک ویروس یکسان می‌سازد.



شکل ۴.۲: نسخه‌برداری ویروس فراریخت [۱، ۱۳].

ویروس‌های فراریخت معمولاً از روش‌های مختلف ابهام‌زایی استفاده می‌کنند که به آن‌ها تنوع بیشتری می‌دهد. درجه جهش و تغییر آن‌ها وابسته به بخشی از کد است که تغییر شکل را انجام می‌دهد و به‌نام موتور فراریختی خوانده می‌شود. یک موتور فراریخت خوب از حداقل دو تکنیک متداول ابهام‌ساز کد[۳] استفاده می‌کند. تکنیک‌های ابهام‌ساز از تغییر نام سادهٔ ثبات‌ها تا روش‌های پیشرفته جایگزین کد[۴] هستند. برخی از روش‌ها به غیر از ابهام‌زایی، هم‌چنین از رمزگذاری برای تولید سری‌های کاملاً متفاوت از ویروس‌ها استفاده می‌کنند. موتورهای فراریخت به سختی نوشته و ایجاد می‌شوند. نویسندگان

[۱] unnecessary calculations

[۲] debugger

[۳] code-obfuscation

[۴] code-substitution

ویروس با پیچیدگی آن‌ها موافق هستند و موتورهای فراریختی را برای دانلود رایگان تولید کرده‌اند. ویروس‌های فراریخت ۳۲ بیتی سیستم‌هایی را که از سیستم عامل ۳۲ بیتی ویندوز استفاده می‌کنند، آلوده می‌نمایند و نسبت به فایل‌های هم‌خانواده مبتنی بر داس[۱] خود نظیر TMC باعث آسیب بیشتری می‌شوند. Regswap در سال ۱۹۹۸ ثبات‌ها را در انواع گونه خود جایگزین نمود، اما کد منبع اصلی تغییری نکرد و آن را خیلی فراریخت نکرد. Win۳۲.Apparition اولین ویروس فراریخت ۳۲ بیتی است که در اوایل سال ۲۰۰۰ پدیدار شد. این ویروس از تکنیک افزودن کد زائد برای تولید گونه‌های مختلف خود استفاده می‌کند. سیستم آلوده به‌طور خودکار گذرواژه‌ها را به سازنده ویروس ایمیل می‌کند و زمانی که برای از بین‌بردن ویروس تلاش می‌کنید، فایل‌های آلوده خراب می‌شوند. W۳۲.Evol در اواسط سال ۲۰۰۰ میلادی ظهور کرد؛ با یک موتور فراریخت‌کننده که می‌توانست تعداد مشخصی از انواع مختلف ترکیب دو تکنیک افزودن کد زائد و جایگزین کد معادل را ایجاد کند. برخلاف بسیاری از ویروس‌ها که همهٔ فایل‌های اجرایی[۲] را آلوده می‌کنند، Evol فقط برنامه‌های کاربردی اجرایی‌ای را هدف قرار می‌دهد که به اندازه کافی بزرگ هستند تا کد آن را جایگزین کنند. یک امضاء در پشتهٔ اجرا اما نه در کد مشاهده می‌شود که باعث سخت‌شدن شناسایی آن از طریق روش اکتشافی و پویش رشته می‌شود. در روند تولید گونه‌های جدید ویروس Evol، روال‌های مبهم‌سازی به‌طور تصادفی و مؤثر انتخاب شده‌اند. سایر ویروس‌های فراریخت پیشرفته مانند Zmist و Win۳۲.Metaphor به‌طور تصادفی چندین روش شامل رمزگذاری در لحظه[۳] و حملات بسته به ساختار فایل آلوده، انتخاب کرده‌اند. Vecna یکی از اعضای گروه نوشتن ویروس ۲۹A، در اوایل دهه ۹۰ اقدام به ایجاد ویروس کرد و در سال ۲۰۰۲ با Lexotan۳۲ مشهور شد. Lexotan۳۲ مشکل ایجاد گونه‌های جدید را با قراردادن جدولی که به برگرداندن جایگشت کد و بازتولید کد مخرب جدید مبهم‌شده با استفاده از ترکیب بسیاری از تکنیک‌های شناخته‌شده در فراریختی کمک می‌نمود، برطرف کرد. فراریختی با

---

[۱] DOS

[۲] executable

[۳] on-the-fly encryption

جایگشت متفاوت است. جایگشت با دوباره مرتب‌کردن کد، انجام می‌شود اما فراریختی ویروس‌های جایگشت شده مانند Zperm و Bistro که تلاش دارند تا دستورالعمل‌هایی برای تغییر برچسب حافظه[۱] را جایگزین کنند. جایگشت ممکن است امضایش پنهان نشود، اما هنگامی که همراه با کد فراریخت می‌شود، به تولید گونه‌های غیر مرتبط می‌پردازد. یک برنامه با دو زیرشاخه $(X_۰, Y_۰)$ و دو نوع در هر شاخه $(X_۱, X_۲, Y_۱, Y_۲)$ را در نظر بگیرید. فرض بر این است که یک امضاء در نقطه‌ای که زیرشاخه‌ها ادغام می‌شود (بنابراین ترتیبی که ظاهر می‌شود مهم است) وجود دارد، ۱۷ تغییر متفاوت وجود دارد که یک امضاء بر اساس گونه‌ای خاص از بین می‌رود. خوشبختانه نویسندگان ویروس نمی‌توانند امضاء را پیش‌بینی کنند و به استفاده از روش‌های پیچیده برای نسخه صحیح فراریخت‌شده جدید نیاز دارند. از سوی دیگر موتورهای جهش‌دهنده[۲] کمک می‌کنند تا به‌جای ایجاد کد خود مخرب، ساختار ویروس را تغییر دهند. طیف گسترده‌ای از این موتورها برای کارهایی مانند جایگشت رمزگشای، فشرده‌سازی کد، ضد اکتشافی، جایگشت کد و فراریختی وجود دارد. موتورهای جهش‌دهنده به‌عنوان جعبه‌های سیاه کار می‌کنند؛ ویروس موجود را به‌عنوان ورودی دریافت می‌کنند و خروجی یک نوع کاملاً جدید خواهد بود. اکثر آن‌ها بر روی گسترش و تغییر و خردکردن کد موجود کار می‌کنند و برای فریب دادن تکنیک‌های تشخیص امضاء بسیار کارا هستند. موتور کدگذاری زامبی[۳] (ZCME) نمونه‌ای از موتور فراریخت است که از بازگشای[۴] خود برای دریافت کد منبع استفاده می‌کند و سپس کد اصلی را با جابجایی تصادفی کد تغییر می‌دهد. مانند تغییر دستورالعمل‌های پرش و یا اضافه‌کردن دستورات NOP. دیگر موتورهای فراریخت، مانند Simile و موتورهای فراریخت MSIL، بر قابلیت موتورهای جهش‌دهنده تأکید می‌کنند. جدیدترین ویروس‌های فراریخت که اخیراً در سال ۲۰۱۲ دیده شده‌اند، نشان دهندهٔ این موضوع هستند که نویسندگان ویروس بیشتر بر گسترش آن‌ها تمرکز دارند تا این‌که آن‌ها

---

[۱] memory stamps

[۲] mutation engines

[۳] Zombie's Code Mutation Engine

[۴] disassembler

را ارتقا و توسعه دهند.

۲.۳.۲ کیت‌های سازنده

وب سایت‌هایی مانند VXHeaven کد منبع را برای ویروس‌ها و موتورهای ابهام‌ساز ارائه می‌دهند و نویسندگان تازه‌کار را قادر می‌سازند تا ویروس‌های پیشرفته را توسعه دهند. اما کاربران علاقه‌مند، به حداقل مهارت‌های برنامه‌نویسی زبان اسمبلی نیاز دارند تا آن‌ها را به یک ویروس فراریخت تبدیل کنند. کیت سازنده ترکیبی از ویژگی‌هایی مانند رمزگذاری و ضد اشکال‌زدایی با موتورهای فراریخت/چندریخت است، که اجازه می‌دهد حتی یک کاربر معمولی کامپیوتر، ویروس‌های مرگبار تولید کند. برخی از کیت‌ها قادر به تولید هزاران گونهٔ جدید هستند. کیت‌های سازنده برای ویروس‌ها، تروجان‌ها، بمب‌های منطقی و حتی کرم‌ها در دسترس‌اند. از آنجایی که آن‌ها گونه‌های مختلف ویروس را به سهولت ایجاد می‌کنند، چالش درخور توجهی برای فروشندگان ضد ویروس هستند. از چندین کیت سازنده مانند G۲ مولد نسل دوم ویروس[۱]، کتابخانه ایجاد ویروس[۲] (VCL)، Phalcon-Skism MPC و کیت خالق نسل بعدی ویروس[۳] (NGVCK) برای پروژهٔ پایان‌نامه استفاده کرده‌ایم [۱، ۷، ۵۲، ۵۳]. همان‌طور که برنامه‌نویسان مختلف این کیت‌ها را توسعه داده‌اند، به ما فرصتی می‌دهد تا عملکرد مدل‌های نمایه پنهان مارکوف را در شناسایی آن‌ها ببینیم.

در زیر شرح مختصری از هر یک از کیت‌های طراحی ویروس مورد استفاده در پروژه پایان‌نامه بیان کرده‌ایم:

- **کتابخانه ایجاد ویروس (VCL۳۲)** انواع ویروس win۳۲ را بسته به تنظیمات کاربر ایجاد می‌کند. اولین نسخه VCL که توسط گروهی از نویسندگان ویروس به نام NUKE ایجاد

---

[۱] second-generation virus generator

[۲] virus creation library

[۳] next generation virus creation kit

شده بود، در سال ۱۹۹۲ به وجود آمد و یک نسخه جدیدتر توسط گروه دیگری «۲۹A» در سال ۲۰۰۴ ظاهر شد. بر خلاف دیگر کیت‌های سازنده که از دستور فرمان[۱] برای تولید گونه‌های ویروس استفاده می‌کنند، یک رابط کاربری[۲] GUI برای انتخاب تنظیمات مختلف فراهم می‌کند. تنظیمات تغییردادنی می‌تواند شامل موارد زیر باشد: کدام بخش از میزبان، شبکه یا مسیر کنونی، داده‌های صندوق پیغام و ... آلوده شود. VCL نیز می‌تواند برای استفاده از موتور چندریختی یا موتور جهش KME-۳۲ که رمزگشای‌ها را جهش می‌دهد به‌کار رود. پس از انتخاب گزینه‌ها، VCL فایل‌های کد اسمبلی گونه‌های ویروس را ایجاد می‌کند. این فایل‌ها بعداً می‌توانند برای گرفتن فایل‌های اجرایی لینک و کامپایل شوند. گزارش شده است که کد تولیدشده توسط نسخه قبلی دارای اشکالاتی بوده و نمی‌توان آن را کامپایل کرد، اما نسخه جدید به نظر می‌رسد بر این مشکلات غلبه کرده است. بسیاری از سازندگان ویروس TASM را بر مایکروسافت اسمبلر (MASM) توصیه می‌کنند تا منابع اسمبلی‌شان را کامپایل کنند.

- **گروه Phalcon-SKISM**، یک رقیب برای گروه NUKE Phalcon-Skism (VCL) کیت PS-MPC[۳] را ایجاد کرده است. Phalcon و SKISM برای تشکیل گروه Phalcon-Skism ادغام شدند. بر خلاف نسخه اول VCL، PS-MPC در ایجاد ویروس‌هایی که قابلیت استفاده داشتند عملکرد خوبی نشان‌داد. یک فایل پیکربندی برای تغییر تنظیمات با حدود ۲۵ جایگزین که شامل پارامترهای اختیاری مانند payload است در آن استفاده می‌شود. کاربر یکی از گزینه‌های آلوده‌کردن فایل‌های COM و exe، رمزگذاری حافظه و رمزگذاری null را انتخاب می‌کند. Payload بستگی به ماه، روز و زمان تعیین‌شده در ویروس و همچنین حداقل یا حداکثر اندازه فایل‌ها برای آلوده‌کردن، تنظیم می‌گردند. PS-MPC همچنین بخش رمزگشای

---

[۱] command prompt

[۲] graphical user interface

[۳] Phalcon-Skism Mass-Produced Code

را مبهم‌سازی می‌کند، اما دیگر روش‌ها مانند تکنیک ضد اشکال‌زدایی و ضد شبیه‌سازی را اجرا نمی‌کند.

- **G2 Virus Generator**، به نظر می‌رسد که فرشته تاریکی[1] با نوشتن PS-MPC کاملاً راضی نشده بود و به همین دلیل برنامه‌ای را به‌نام G۲ Phalcon-Skism در نوامبر سال ۱۹۹۳ منتشر کرد. این نام از اعتقاد خالق آن که G۲ را مولد ویروس‌های نسل دوم می‌دانست، نشأت گرفته است. عملکرد G۲ بسیار شبیه PS-MPC است. اما تفاوت‌های چشمگیری دارند. G۲ هر بار، حتی اگر مقادیر موجود در فایل پیکربندی بدون تغییر باقی بمانند، ویروس متفاوتی ایجاد می‌کند. G۲ همچنین با یک فایل G۲.DAT عرضه می‌شود که اطلاعات واقعی برنامه را شامل می‌شود. فرشته تاریکی اعلام کرده است که او نسخه‌های به‌روزرسانی این فایل را که به‌طور کامل روش‌های عملکرد برنامه را تغییر خواهد داد، ارائه خواهد کرد.

- **Next Generation Virus Creation Kit (NGVCK)**، توسط SnakeByte ایجاد شده است و در سال ۲۰۰۱ نمایان گردید و تا حدی که اطلاع داریم پیشرفته‌ترین سازنده ویروس است. بر خلاف VCL و PS-MPC، به تنظیم تنظیمات پیکربندی نیازی نیست زیرا هر بار که استفاده می‌شود به‌طور خودکار یک گونهٔ جدید تولید می‌کند. این کیت سازنده، کد را با استفاده از تکنیک‌های مبهم‌سازی مانند جاگذاری کد بنجل[2]، مرتب‌سازی مجدد زیرروال‌ها، تعویض تصادفی ثبات‌ها و جایگزینی معادل کد، مبهم و غیر قابل تشخیص می‌کند. NGVCK به‌عنوان یک برنامه غیر ویروس با چندین نسخه و نسخه بتا توسعه یافته است. کیت NGVCK برای برآورده‌کردن نیازهای برنامه‌نویسان تازه‌کار و پیشرفته برنامه‌ریزی شده است که در آن برنامه‌نویسان پیشرفته می‌توانند نوع رمزنگاری و ضد ترفندها و مسیریابی دایرکتوری را انتخاب کنند.

---

[1] Dark Angel

[2] junk code insertion

کیت‌های سازنده و موتورهای جهش‌دهنده که در این‌جا برای استفادهٔ آسان و ایجاد ویروس‌های جدید بیان گردیدند، بسیار مرگبارند زیرا می‌توانند انواع مختلفی از ویروس‌های قدیمی را احیا کنند. چنین فراریختی از ویروس‌های قدیمی همان مشکلاتی را که برای ضد ویروس‌ها به بار می‌آوردند، دوباره تکرار می‌کند. بنابراین بسیار مهم است که از تکنیک‌های یادگیری ماشین و نوعی اتوماسیون برای شناسایی آن‌ها استفاده شود.

۴.۲ تکنیک‌های ایجاد ابهام در کد

ایجاد ابهام در کد یا همان مبهم‌سازی کد، تغییر کد و مبهم یا دشوارساختن آن برای درک است. برنامه‌نویسان نرم‌افزار از این تکنیک‌ها برای مقاوم‌کردن محصول خود در برابر مهندسی معکوس استفاده می‌کنند. نویسندگان ویروس‌های فراریختی از یک یا چند تکنیک برای ایجاد یک نسخه منحصربه‌فرد از ویروس موجود استفاده می‌کنند که باعث می‌شود آن‌ها برای پویشگرهای ویروس غیر قابل تشخیص شدند [۱، ۱۲، ۱۳، ۳۶–۳۹].

۱.۴.۲ افزودن کد زائد

کدهای بدون کارکرد یا زائد، دستورالعمل‌های برنامه‌نویسی به‌شمار می‌آیند که بخشی از برنامه فیزیکی هستند اما منطقی نیستند. آن‌ها به نتیجه برنامه مرتبط نمی‌شوند. دستورالعمل‌های بدون کارکرد مانند تغییر ثبات (XCHG) روند شبیه‌سازی کد را آهسته و کند می‌کند. دستورالعمل‌های دیگر مانند "NOP" ، " MOV ax ، ax " ، " SUB ax ، 0 " و غیره باعث می‌شود که ویروس ظاهری متفاوت پیدا کند و بنابراین احتمالاً در تحلیل اکتشافی تشخیص‌دادنی نباشد. دستورالعمل‌های زائد برای توسعهٔ کدی که هرگز اجرا نمی‌شود و یا هم‌چنین برای محاسبه متغیرها در بلوک‌های زائد دیگر می‌تواند استفاده شود. ایده اصلی این تکنیک مبهم‌ساز کد این است که دستگاه مجازی یا فرد متخصص را گیج و خسته

کند تا نتوانند ویروس را شناسایی نمایند. هرچند پویشگرهای ویروس این روزها به اندازه کافی قدرتمند هستند تا بتوانند از تکنیک کدهای بدون کارکرد گذشته جلوگیری کنند. یک روش برای تشخیص ویروس‌هایی که از این تکنیک بهره جسته‌اند این است که وقتی بیش از حد معمول دستورالعمل‌هایی بدون کارکرد در یک فایل وجود دارد، می‌توان آن را به‌عنوان یک ویروس تشخیص داد. زیرا بسیار بعید است که چنین دستورالعمل‌هایی در برنامه‌های غیر ویروس وجود داشته باشد.

۲.۴.۲ تغییر نام ثبات‌ها

تغییر نام ثبات[۱] اصطلاحی است که برای تغییرنام متغیرها و یا ثبات‌ها در یک ویروس استفاده می‌شود. هنگامی که نام ثبات‌ها تغییر می‌کنند، آن‌ها در نتیجه از کدهای عملیاتی (آپکد) مختلفی استفاده می‌نمایند که جست‌وجوی امضاء را فریب می‌دهند. Regswap یک ویروس فراریخت است که تغییر نام ثبات‌ها را برای هر گونهٔ جدید از خود انجام می‌دهد. در شکل ۵.۲ دو گونه از regswap نشان داده شده است که دارای مجموعهٔ یکسان از دستورالعمل‌ها هستند اما از ثبات‌های مختلف استفاده می‌کنند. زمانی که از تکنیک‌های مبتنی بر امضاء برای تشخیص ویروس استفاده می‌شود، ویروس موفق به دورزدن روند شناسایی نرم‌افزارهای ضد بدافزار می‌شود.

۳.۴.۲ جایگشت زیرروال‌ها

جایگشت زیرروال‌ها یک روش مبهم‌ساز ساده است که طی آن زیرروال‌ها مجدد مرتب می‌شوند. این کار اثر ویروس را تحت تأثیر قرار نمی‌دهد؛ به‌دلیل این‌که اهمیت ترتیب دیده‌شدن زیرروال‌ها در کد برای اجرای برنامه ناچیز است. بنابراین ویروس حاوی n زیرروال می‌تواند $n!$ حالت جایگشت داشته باشد. در مقایسه با روش‌های دیگر ابهام‌سازی، جایگشت زیرروال می‌تواند به راحتی توسط امضاء

---

[۱] register renaming

a.)

5A	pop	edx
BF**04000000**	mov	edi, 0004h
8BF5	mov	esi, ebp
B8**0C000000**	mov	eax, 000Ch
81C2**88000000**	add	edx, 0088h
8B1A	mov	ebx, [edx]
899C8618110000	mov	[esi+eax*4+00001118], ebx

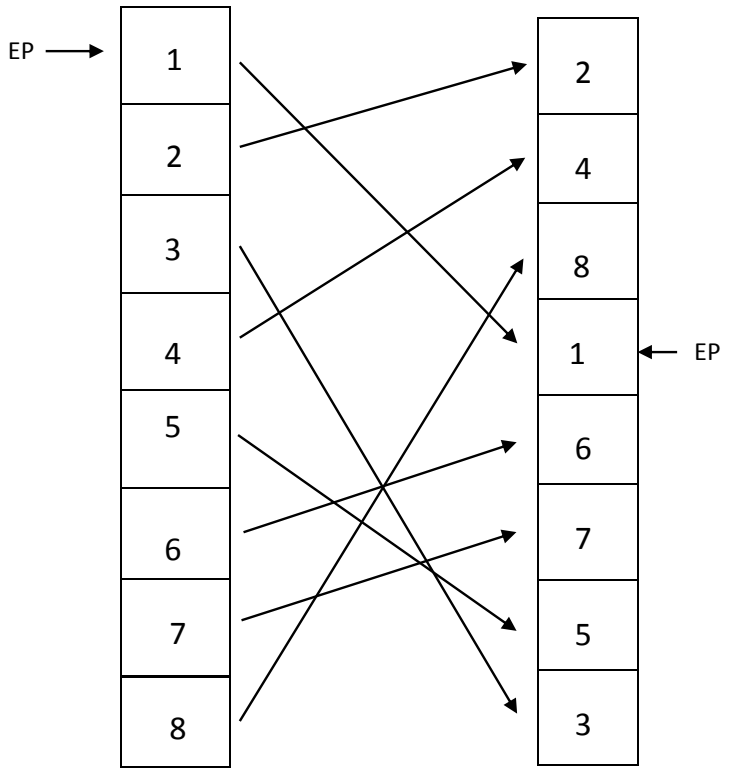
b.)

58	pop	eax
BB**04000000**	mov	edi, 0004h
8BD5	mov	edx, ebp
BF**0C000000**	mov	edi, 000Ch
81C0**88000000**	add	eax, 0088h
8B30	mov	esi, [eax]
899C8918110000	mov	[edx+edi*4+00001118], esi

شکل ۵.۲: نمونهٔ Regswap [۵۴].

شناسایی شود. ویروس‌های فراریخت مانند Win۹۵.Ghost و Win۹۵.Smash نمونه‌هایی از این رفتار هستند.

اما بازچینی زیرروال‌ها برای برخی از روش‌های تحلیل چالش‌های درخور توجهی به‌وجود می‌آورد. این کار ترتیب توالی چندگانه‌ای از یک خانواده ویروس داده‌شده را ایجاد می‌کند و با تنظیم ترتیب چندگانه بر اساس ناحیه‌های منطبق از آپکدها مدل می‌شود. اگر بر روی برنامه‌ای از تکنیک جایگشت استفاده شده باشد، اکثر ناحیه‌ها به هم منطبق نیستند و از این رو یک هم‌ترازی ضعیف خواهیم داشت و مدل ضعیف‌تری ساخته می‌شود. یک راه‌حل برای این تکنیک ابهام‌سازی این است که اثر جایگشت را بر روی هر توالی قبل از هم‌تراز کردن آن، برعکس و خنثی کنیم.



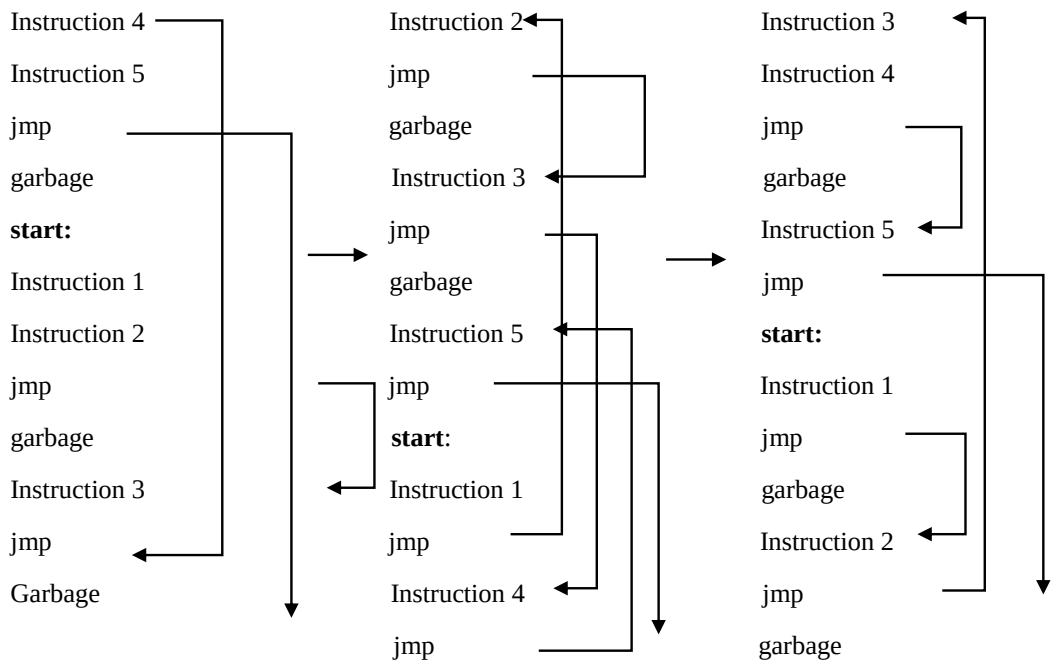
شکل ۶.۲: نمونهٔ جایگشت زیرروال‌ها [۱۴].

۴.۴.۲ مرتب‌سازی مجدد کد از طریق پرش

مرتب‌سازی مجدد کد از طریق پرش[۱] ترتیب دستورالعمل‌ها را تغییر می‌دهد، اما جریان منطقی دستورالعمل‌های اصلی را با استفاده از پرش حفظ می‌کند. مرتب‌سازی مجدد کد به‌وسیلهٔ پرش‌های بدون شرط باعث ایجاد ابهام در جریان کنترل کد می‌شود. این پرش‌های بدون شرط به‌طور تصادفی وارد می‌شوند که تشخیص بر پایه نقشه‌برداری حافظه را دچار چالش می‌نمایند.

---

[۱] code reordering through jumps



شکل ۷.۲: نمونهٔ مرتب‌سازی مجدد کد [۵۵].

شکل ۷.۲ نمونه‌ای از مرتب‌سازی مجدد کد نشان می‌دهد. این روش نسبتاً ساده، با تغییر توالی آپکدها بر تشخیص امضاء غلبه می‌کند.

۵.۴.۲ جایگزینی کد معادل

هر کاری را می‌توان با روش‌های مختلف انجام داد. به‌طور مشابه کدهای ویروسی اگرچه به نظر متفاوت هستند، می‌توانند کار یکسانی انجام دهند. جایگزینی کدهای معادل برای کدهای ویروسی از تعداد اندکی تکنیک‌های تشخیص فرار می‌کند. می‌توان از طریق بررسی رفتار آن را تشخیص داد، زیرا فرامین اجرای کد در بیشتر موارد تغییر نمی‌کند. این نوع ابهام‌سازی می‌تواند همچنین برای کاهش یا گسترش کد اصلی به وسیله جایگزینی کد با کدهای معادل کوچک‌تر یا بزرگ‌تر استفاده شود. به‌عنوان یک مثال ساده، "ADD ax ، 3" را می‌توان به "SUB ax ، -3" تبدیل کرد، همان‌طور که مشاهده می‌نمایید هر دو دستورالعمل عدد ۳ را به محتویات ثبات ax اضافه می‌کند. همچنین می‌تواند با یک

فرآیند دو مرحله‌ای مانند "MOV bx ، -3" و "SUB ax ، bx" انجام شود. W۳۲.Evol یک ویروس فراریخت است که به‌طور تصادفی کد معادل را جایگزین و گونه‌های مختلفی در هر نسل تولید می‌کند. شکل ۸.۲ چند جایگزینی که در این ویروس مشاهده شده است را نشان می‌دهد [۵۶].

	Parent	Offspring (transformed)

(a)	push eax
	mov [edi], 0x04
	jmp lable
		push eax
		push ecx
		mov ecx, 0x04
		mov [edi], ecx
		pop ecx
		jmp lable

(b)	push 0x04
	mov eax, 0x09
	jmp lable
		mov eax, 0x04
		push eax
		move ax, 0x09
		jmp lable

(c)	mov eax, 0x04
	push eax, 0x09
	jmp lable
		mov eax, 0x04
		push eax
		move ax, 0x09
		jmp lable

شکل ۸.۲: جایگزینی کد معادل در ویروس فراریخت W۳۲.Evol [۵۶].

هر قسمت از کد در فرزندان دقیقاً همانند والدش، فقط با تغییرات جزئی در کد والدین کار می‌کند. اغلب کدهای تغییریافته به اندازه کافی ساده نیستند که به وسیلهٔ جست‌وجوی رشته‌ها شناسایی شوند. با این حال، انواع نشان‌داده‌شده در مثال بالا می‌تواند با استفاده از رشته نیمه معین[۱] در امضاء شناسایی شود. یکی از تکنیک‌های تشخیص استفاده‌شده برای مقابله با چنین ابهام‌سازی پیشرفته، تبدیل کد به یک کد ساده است [۵۷].

[۱] wild string

فصل سوم

مروری بر کارهای پیشین

در یک دسته‌بندی کلی می‌توان روش‌های تحلیل بدافزار را به دو دستهٔ ایستا و پویا تقسیم کرد. روش‌های ایستا بسیار محبوب و رایج‌اند زیرا بدون اجرای بدافزار و از تحلیل ساختار کد قادرند آن‌ها را کشف نمایند. به‌طور معمول در روش ایستا برای شناسایی بدافزارها، روش‌ها و ابزار تشخیص با استفاده از روش‌های مبتنی بر امضاء به‌طور گسترده‌ای استفاده می‌شود. الگویی از ساختارهای رشته‌ای کوتاه از بایت‌ها که منحصربه‌فرد است، استخراج می‌شود، سپس امضاها در بانک‌های اطلاعاتی نگهداری می‌شوند و عملیات شناسایی مبتنی بر آن‌ها صورت می‌گیرد. در مقابل، روش‌های پویا با اجرای بدافزار می‌توانند آن‌ها را کشف کنند که این کار معمولاً به شبیه‌سازی[۱] نیاز دارد. در این فصل، برخی از نمونه کارهای پیشین را مورد بحث قرار می‌دهیم. این کارها را بر اساس اینکه روش مورد استفاده برای تحلیل بدافزار در آن‌ها متکی بر تحلیل ایستا یا تحلیل پویا است و یا در مورد تکنیک‌هایی که از یک رویکرد ترکیبی استفاده می‌کنند، دسته‌بندی و بررسی می‌شوند.

---

۱ emulation

۱.۳ روش‌های تحلیل بدافزار

۱.۱.۳ تحلیل ایستا

تحلیل ایستای برنامه بدون اجرای آن انجام می‌شود. مثال‌هایی از اطلاعاتی که می‌توان از تحلیل ایستا به دست آوریم شامل توالی کدهای عملیاتی (استخراج به وسیلهٔ بازگشایی فایل باینری) و گراف‌های جریان کنترل و غیره است. این مجموعه ویژگی‌ها می‌تواند به‌صورت جداگانه یا ترکیبی برای تشخیص بدافزار استفاده شود. در [۱۱]، نویسندگان یک تکنیک تشخیص بدافزار ارائه داده‌اند که مبتنی بر تحلیل ایستا و بر اساس گراف جریان کنترل است. رویکرد آن‌ها بر تشخیص الگوهای مبهم در نرم‌افزارهای مخرب تمرکز دارد که می‌توانند به دقت خوبی دست یابند. از تکنیک‌های یادگیری ماشین برای تشخیص بدافزار در زمینهٔ تحلیل ایستا استفاده کرده‌اند. در [۳۵] مدل پنهان مارکوف رویکردی مطلوب بر اساس توالی آپکدهای استخراج‌شده برای طبقه‌بندی نرم‌افزارهای مخرب فراریخت است. یک تحلیل مشابه در مدل نمایه پنهان مارکوف در [۷] به کار گرفته شده است، در حالی که تحلیل مولفهٔ اصلی[۱] در [۱۰] و [۱۳] استفاده شده است و ماشین‌های بردار پشتیبان برای تشخیص بدافزار در [۳] استفاده شده‌اند. مقاله [۱] بر اساس ویژگی‌های مشتق‌شده از تحلیل ایستا برای طبقه‌بندی بدافزارها خوشه‌بندی را به کار گرفته است. در [۱۲] تحلیل گراف فراخوان تابع برای استفاده از تشخیص بدافزار به کار گرفته شده است، در حالی که [۱۴] یک روش میزان شباهت مبتنی بر کدهای عملیاتی را که بر تکنیک‌های رمزنگاری جایگزین ساده متکی است، بررسی می‌کند. توالی فراخوانی رابط برنامه‌نویسی کاربردی[۲] API و توالی آپکدها هر دو در [۱۵] به کار گرفته شده است تا مشخص شود کدام یک از کدها به برخی از نرم‌افزارهای مخرب خاص، شباهت دارند. مقالات [۵۸، ۵۹] تحلیل ساختار فایل را بر

---

[۱] Principal Component Analysis

[۲] application programming interface

اساس تغییرات آنتروپی [۱] انجام داده‌اند. کار صورت‌گرفته در این مقاله‌ها توسط نمره مبتنی بر آنتروپی الهام‌گرفته‌شده در [۶۰] است.

۲.۱.۳ تحلیل پویا

تحلیل پویا اغلب به اجرای برنامه در یک محیط مجازی نیاز دارد [۶۱]. نمونه‌هایی از اطلاعاتی که می‌توان از طریق تحلیل پویا به دست آورد عبارتند از: فراخوان API، فراخوان سیستمی، ردیابی دستورالعمل‌ها، تغییر ثبات‌ها، تغییرات حافظه و غیره. در [۶۲] نویسندگان مدل‌های خوش‌ساختی می‌سازند که برای ضبط رفتار بدافزار بر اساس فراخوانی‌های سیستم طراحی شده‌اند. نتایج مدل‌های رفتاری به شکل گراف نشان داده می‌شوند؛ رأس‌ها نشان‌دهنده فراخوانی‌های سیستم و یال‌ها ارتباط بین فراخوانی‌ها را نشان می‌دهند. مقاله [۲۴] یک ابزار مانیتورینگ زمان اجرا ارائه می‌دهد که ویژگی‌های آماری را بر اساس اطلاعات مکانی و زمانی در ثبت وقایع APIها استخراج می‌کند. اطلاعات مکانی متشکل از آرگومان‌ها و مقادیر بازگشتی فراخوانی API است، در حالی که اطلاعات زمانی توالی فراخوانی API است. این اطلاعات برای ساخت مدل‌های رسمی که به الگوریتم‌های یادگیری ماشین استاندارد به‌عنوان ورودی داده می‌شود، برای استفاده در تشخیص بدافزار به کار گرفته می‌شود. در [۲۱] مجموعه‌ای از فراخوانی API برنامه استخراج و با گراف جریان کنترل ترکیب می‌شود تا مدل API-CFG به دست آید. در یک نسخه کمی اصلاح‌شده [۲۰]، روش $n-gram$ برای فراخوانی‌های API به کار رفته است.

[۱] entropy

۳.۱.۳ رویکرد ترکیبی

تکنیک‌های ترکیبی جنبه‌هایی از هر دو تحلیل ایستا و پویا را ترکیب می‌کند. در این بخش دو نمونه از کارهای اخیر از این نوع را مورد بحث قرار می‌دهیم. در [۲۹]، نویسندگان چارچوبی برای طبقه‌بندی بدافزارها با استفاده از هر دو تحلیل ایستا و پویا ارائه می‌دهند. آن‌ها ویژگی‌های نرم‌افزارهای مخرب را با استفاده از روشی که دی ان ای بدافزار (Mal-DNA) نامیدند، تعریف می‌کنند. شاکله اصلی این تکنیک مانیتور و تحلیل رفتار مبتنی بر اشکال‌زدایی است که ویژگی‌های پویا را استخراج می‌کند. در مقاله [۲۸]، نویسندگان یک ابزار را توسعه و تحلیل کرده‌اند و آن را تحلیل‌گر HDM می‌نامند. این ابزار از هر دو روش تحلیل ایستا و تحلیل پویا در مرحلهٔ یادگیری استفاده می‌کند، اما تنها در مرحله تست تحلیل ایستا اجرا می‌شود. هدف این است که از مزیت ظاهراً برتر تحلیل پویا در فاز آموزش سود برده شود، در حالی که مزیت کارایی تشخیص ایستا در مرحله امتیازدهی حفظ شود. برای مقایسه، نشان داده‌اند که تحلیل‌گر HDM نسبت به دیگر روش‌های تحلیل ایستا و پویا در [۲۸]، دقت کلی و پیچیدگی زمان بهتری دارد. تحلیل پویا در [۲۸] بر اساس توالی فراخوانی‌های API استخراج شده است.

در ادامه به مرور کارهای انجام‌شدهٔ پیشین بر اساس تکنیک‌های رایج در زمینه تشخیص بدافزار پرداخته‌ایم.

۲.۳ تکنیک‌های تشخیص بدافزار

با افزایش سریع تنوع و پیچیدگی در تکنیک‌های مبهم‌سازی که توسط نویسندگان بدافزار استفاده می‌شود، نرم‌افزارهای ضد ویروس به این‌که بتوانند سرعت خود را حفظ کنند و به‌روز باقی بمانند، به شدت نیازمند هستند. از این رو، حوزه گسترده‌ای برای تحقیق و بررسی بر تکنیک‌های تشخیص به وجود آمده است. در حال حاضر، تکنیک‌های متعددی برای شناسایی بدافزارها استفاده می‌شود که بر

پارامترهای مختلف تمرکز دارند. به برخی از این تکنیک‌ها در ادامه پرداخته‌ایم.

۱.۲.۳ تکنیک مبتنی بر امضاء

تشخیص مبتنی بر امضاء بر اساس پویش برنامه‌ها و فایل‌ها و محاسبه و ایجاد امضاء برای هر یک از آن‌ها است. تشخیص مبتنی بر امضاء شامل جست‌وجوی یک الگوی شناخته‌شده در فایل اجرایی داده‌شده به‌عنوان یک امضاء است. اکثر تولیدکنندگان ضد ویروس مخزنی بزرگ از امضای منحصربه‌فرد برای هر ویروس شناخته‌شده که به‌صورت منظم به‌روز می‌شود، دارند [۳]. هنگامی که یک نرم‌افزار ضد ویروس یک فایل اجرایی را پویش می‌کند، امضای فایل اجرایی را تولید می‌کند و در پایگاه‌داده خود به دنبال یک الگوی تطبیق‌پذیر می‌گردد. اگر یک الگوی منطبق پیدا شود، به نظر می‌رسد فایل اجرایی آلوده است. طرح تشخیص مبتنی بر امضاء در مقیاس وسیع استفاده می‌شود، زیرا که ساده و دقیق و سریع است [۱۱، ۵۰]. یکی از نقاط ضعف این تکنیک این است که برای نرم‌افزارهای مخرب جدید به به‌روز رسانی مداوم امضاء نیاز دارد. همچنین نمی‌توان آن را برای شناسایی نرم‌افزارهای مخرب که قبلاً شناسایی نشده‌اند، به کار برد و فقط می‌توانند بدافزارهایی را با امضاء شناخته‌شده شناسایی کنند. نقطه ضعف دیگر آن این است که تشخیص امضاء بسیار آسان است و از تکنیک‌های مبهم‌سازی ساده که در ویروس‌های چندریخت و فراریخت از آن‌ها استفاده می‌شود، می‌توان برای فرار از شناسایی توسط تکنیک‌های تشخیص مبتنی بر امضاء استفاده کرد [۳۰، ۳۳، ۶۳].

۲.۲.۳ تکنیک مبتنی بر رفتار

تشخیص مبتنی بر رفتار یکی دیگر از تکنیک‌های رایج است که در آن تمرکز بر روی اقداماتی است که نرم‌افزارهای مخرب در طول اجرای خود انجام می‌دهند و تلاش برای درک هدف بدافزار با استفاده از تکنیک‌های مختلف در تشخیص مبتنی بر رفتار است. رفتار هر دو گروه فایل‌های سالم و فایل‌های مخرب در مرحله اول مورد بررسی قرار می‌گیرد که عموماً به‌عنوان مرحله آموزش یا یادگیری شناخته

می‌شود و پس از آن در مرحله نظارت، از اطلاعات جمع‌آوری‌شده در مرحله آموزش برای طبقه‌بندی یک فایل اجرایی داده‌شده استفاده می‌کنیم تا به‌عنوان یک بدافزار یا فایل سالم کلاسه شود [۵، ۲۳، ۴۳].

۳.۲.۳ تکنیک مبتنی بر ناهنجاری

این تکنیک که ذاتاً به تشخیص مبتنی بر رفتار شبیه است، در واقع می‌تواند به‌عنوان یکی از تکنیک‌های تشخیص مبتنی بر رفتار با تغییراتی جزیی در نظر گرفته شود، که در آن تحلیل با مطالعه رفتار همه فایل‌ها در مرحله آموزش انجام می‌شود و در مرحله نظارت فایل‌هایی که انحراف از رفتار طبیعی نشان می‌دهند [۶۴]، بررسی و شناسایی می‌شوند، چنین فایل‌هایی طبقه‌بندی و به‌عنوان آلوده تشخیص داده می‌شوند. این تکنیک به شدت مستعدد پذیرش مثبت کاذب است. در [۶۵] یک رویکرد بی‌نظیر برای شناسایی بدافزارها با استفاده از ویژگی‌های سخت‌افزاری مبتنی بر ناهنجاری مورد بحث قرار داده‌اند.

۴.۲.۳ تکنیک مبتنی بر ساختار

در مقاله [۲۸] اسکندری و همکارانش روش جدیدی ارائه می‌دهد که با استفاده از تکنیک‌های یادگیری ماشین با بهره‌گیری از مزایای روش‌های تحلیل ترکیبی، دقت تحلیل بدافزار را در حالی که سرعت آن در حد معقول حفظ می‌شود، بهبود می‌بخشد. آن‌ها رویکرد خود را به‌عنوان تحلیل‌گر HDM نام‌گذاری کرده‌اند که برای تحلیل ترکیبی بر اساس تکنیک‌های داده کاوی است. آن‌ها از تحلیل پویا برای استخراج توالی API در زمان اجرای نمونه‌های بدافزار استفاده نموده‌اند، در عین حال از تحلیل ایستا برای استخراج گراف جریان کنترل غنی‌شده (ECFG) استفاده کرده‌اند که شامل اطلاعات مربوط به فراخوان API است. پس از استخراج ویژگی‌ها، آن‌ها از یک موتور تطبیق‌دهنده استفاده می‌کنند که ترکیبی از ویژگی‌های حاصل از تحلیل پویا با ECFG مربوطه است و پس از آن هر پرش شرطی، بر اساس اطلاعات پویایی برچسبی را دریافت می‌کند. در این مرحله یک الگوریتم یادگیری ماشین برای ساخت یک مدل آموزش دیده‌شده با گره‌های برچسب ECFG استفاده می‌شود.

این مدل یادگیری توسط HDM-Analyzer در زمان پویش و تحلیل فایل‌های اجرایی شناخته‌شده مورد استفاده قرار می‌گیرد.

اسلام و همکارانش [۲۷] یک طبقه‌بندی بر اساس ویژگی‌های استخراج‌شده از تحلیل ایستا و تحلیل پویا ارائه می‌دهند. در طی تحلیل ایستا آن‌ها بردارهای فراوانی طول تابع (FLF) و بردارهای اطلاعات رشته چاپی (PSI) را استخراج می‌کنند. ویژگی FLF بر اساس فراوانی تعداد توابع با طول‌های مختلف است. در PSI، تمام رشته‌های قابل چاپ را از نمونه‌های مخرب استخراج کرده‌اند تا یک لیست کلی از رشته‌ها ایجاد کنند. سپس آن‌ها برای هر نمونه فراوانی رشته‌های متمایز گزارش می‌کنند و پس از آن یک بردار باینری با توجه به اندیس هر رشته در لیست جهانی ایجاد می‌شود، که در آن ۱ نشان‌دهندهٔ این واقعیت است که رشته وجود دارد و ۰ نشان‌دهندهٔ آن است که رشته وجود ندارد. از سوی دیگر در طی تحلیل پویا، آن‌ها ویژگی‌های API مانند نام‌های عملکرد API و پارامترهایی از فایل‌های log را استخراج و سپس یک لیست جهانی ایجاد می‌کنند و یک بردار باینری دیگر می‌سازد که در آن ۱ نشان‌دهندهٔ تابع API در لیست جهانی است که فراخوانی شده است، در غیر این صورت ۰ را برای سلولی با اندیس مطابق با لیست جهانی در بردار مذکور قرار می‌دهند. این دو بردار برای تعیین امتیاز شباهت و طبقه‌بندی استفاده می‌شوند.

۵.۲.۳ تکنیک مبتنی بر شباهت

این نوع تشخیص مبتنی بر نوعی از اندازه‌گیری تشابه میان فایل‌های فراریخت و سالم است. این روش به دنبال پیداکردن برخی از ویژگی‌هایی است که شبیه به خانواده مخرب فراریخت داده‌شده باشند. در تحقیقات قبلی، بسیاری از راهبردهای تشخیص مبتنی بر شباهت[۱] که بر اساس تحلیل ویژگی‌های توالی آپکدها از فایل‌های مخرب است مورد بحث قرار گرفته‌اند. برخی از این تکنیک‌ها هم‌ترازی توالی‌های

[۱] similarity based detection

دوگانه[۱] [۷، ۴۳]، شباهت ان-گرم[۲] [۳۵]، شباهت کوساین[۳] [۳۲]و شباهت چی-اسگورد[۴] [۶۶] است.

۱.۵.۲.۳ تکنیک مبتنی بر گراف

تکنیک‌های مبتنی بر گراف آپکد[۵]، روش‌هایی هستند که شامل تحلیل گراف‌های تولیدشده توسط توالی آپکدها به جای تحلیل خود فایل‌ها است. برای تشخیص یک گراف جهت‌دار وزنی بر اساس تحلیل یک خانواده مخرب فراریخت‌شده، ساخته می‌شود. سپس یک گراف برای فایل مورد نظر ایجاد می‌شود. در نهایت دو گراف را می‌توان مقایسه کرد تا امتیاز شباهت برای آن محاسبه شود. این روش با محاسبه اختلاف مطلق بین عناصر مربوطه در دو گراف عمل می‌کند. اگر امتیاز محاسبه کم باشد، این بدان معنی است که دو فایل مشابه هستند و از این رو فایل به‌عنوان یک فایل مخرب طبقه‌بندی می‌شود در غیر این صورت به‌عنوان یک فایل سالم طبقه‌بندی می‌شود [۲۱، ۲۲، ۳۸].

۲.۵.۲.۳ تکنیک مبتنی بر فاصله جایگزینی ساده

فاصله جایگزینی ساده[۶] یکی دیگر از تکنیک‌های تشخیص است که بر اساس اندازه‌گیری شباهت است. این تکنیک از الگوریتم سریع جاگوبسن[۷] همان‌گونه که از آن در رمزنگاری استفاده می‌شود، بهره می‌برد. سپس امتیاز تشابه با تبدیل ماتریس تشکیل‌شده از توالی آپکدهای یک بدافزار خاص را محاسبه

---

۱ pairwise sequence alignment

۲ n-gram

۳ cosine similarity

۴ chi-squared

۵ opcode graph based

۶ simple substitution distance

۷ Jackobsen's fast algorithm

می‌کند [۱۴].

۶.۲.۳ تکنیک مبتنی بر الگوی آماری

روش‌های متعارف مانند تشخیص امضاء در مورد بدافزارهای فراریخت به دلیل استفاده از روش‌های مبهم‌سازی کد، نمی‌توانند کارآمد باشند و این بدافزارها با استفاده از تغییر کد خود از شناسایی‌شدن فرار می‌کنند. اگرچه نسخه‌های مختلف فراریخت می‌توانند تغییرات بزرگی در کد خود ایجاد کنند و از یکدیگر متمایز شوند، اما برخی از الگوهای آماری بدافزارهای مخرب فراریخت همچنان یکسان هستند. برخی تکنیک‌ها از جمله یادگیری ماشین در [۳] و تحلیل آماری در [۱، ۷، ۸، ۶۵] مورد مطالعه قرار گرفته‌اند. علاوه بر این برخی از تکنیک‌های بهبودیافته برای جلوگیری از طرح‌ها و تکنیک‌های تشخیص فراریخت در [۳۵، ۳۶] بررسی شده‌اند.

۱.۶.۲.۳ تکنیک مبتنی بر مدل‌های پنهان مارکوف

در سال‌های اخیر بسیاری از تکنیک‌های یادگیری ماشین در تشخیص نرم‌افزارهای مخرب به ویژه بدافزارهای فراریخت که با استفاده از تکنیک‌های تشخیص سنتی بدافزار قابل شناسایی نیستند، استفاده شده است. اکثر تکنیک‌های یادگیری ماشین بر اساس یک قاعدهٔ کلی به تحلیل یک خانواده خاص از ویروس برای محاسبهٔ امتیازی که به نحوی آن را می‌توان امتیاز شباهت نامید و می‌توان از آن برای تشخیص یک فایل ورودی به‌عنوان یک بدافزار متعلق به یک خانواده و یا به‌عنوان یک فایل سالم نام برد، بهره می‌برند. یکی از این روش‌ها مدل مخفی مارکوف (HMM) است که یکی از محبوب‌ترین تکنیک‌های یادگیری ماشین است که در زمینه شناسایی بدافزارها استفاده می‌شود [۲، ۳۵، ۵۳]. در این تکنیک، یک مدل مخفی مارکوف برای توالی آپکدهای استخراج‌شده از بدافزار، آموزش دیده و ساخته می‌شود [۱، ۶، ۳۵، ۳۶]. پس از پایان مرحله آموزش مدل ساخته‌شده برای امتیازدهی فایل‌های ورودی مورد استفاده قرار می‌گیرد. سپس امتیاز حاصله با آستانه از پیش‌تعیین‌شده مقایسه می‌شود، اگر بیشتر

از آستانه باشد فایل به‌عنوان یک فایل آلوده طبقه‌بندی می‌شود [۳۵، ۳۶].

۲.۶.۲.۳ تکنیک مبتنی بر مدل‌های نمایه پنهان مارکوف

یکی دیگر از مدل‌های رایج برای تشخیص بدافزار مدل نمایه پنهان مارکوف است که فرمول خاصی از روش مدل‌های پنهان مارکوف استاندارد است [۷، ۴۳، ۵۴] که در فصل ۴ به اجمال آن را معرفی و بررسی کرده‌ایم.

فصل چهارم

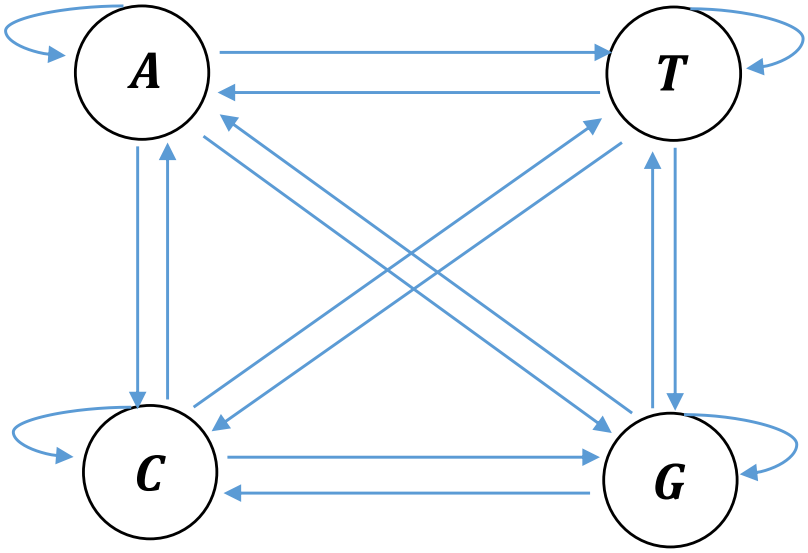
روش پیشنهادی

۱.۴ تئوری مدل‌های پنهان مارکوف

۱.۱.۴ زنجیره مارکوف

زنجیره مارکوف مجموعه وضعیت‌هایی همراه با احتمال مربوط به هر انتقال بین وضعیت‌ها است. در زنجیره مارکوف اگر احتمالات انتقال فقط به وضعیت فعلی و نه به وضعیت‌های قبلی بستگی داشته باشد، در نتیجه آن یک زنجیرهٔ مارکوف بدون حافظه است. به عبارت دقیق‌تر، در زنجیره مارکوف مرتبه اول، احتمال انتقال تنها به وضعیت فعلی بستگی دارد، در حالی که در زنجیره مارکوف مرتبه nام احتمالات انتقال به وضعیت فعلی و وضعیت‌های $n-1$ قبلی بستگی دارد. در هر صورت زنجیره مارکوف حافظه متناهی دارد. زنجیرهٔ مارکوفی برای توالی DNA در شکل ۱.۴ نشان داده‌ایم؛ کد شیمیایی توالی DNA با یک حرف از چهار نماد (پایه) که با نام‌های A (آدنین)، C (سیتوزین)، G (گوانین) و T (تیمین) مشخص شده است [۴۱، ۶۷]. هر یال در شکل نشان‌دهندهٔ احتمال انتقال یک پایه به دنبال پایه دیگر است. احتمالات انتقال را می‌توان پس از مشاهده چندین توالی DNA

محاسبه کرد. ماتریس احتمال انتقال متناظر با ترکیب‌کردن این احتمالات انتقال به دست می‌آید. این مدل مارکوف برای توالی DNA یک مدل مارکوف مرتبه اول است زیرا هر رویداد فقط به رویداد قبلی بستگی دارد.



شکل ۱.۴: زنجیره مارکوف برای توالی DNA [۷،۶۷].

احتمال انتقال از یک وضعیت با مشاهده نماد X به وضعیت دیگر با مشاهده نماد Y را احتمال انتقال از X به Y می‌گویند. باید توجه داشت که مجموع احتمالات انتقال از هر وضعیت برابر یک است، چرا که این انتقال‌ها نشان‌دهنده توزیع آماری هستند. زنجیره مارکوف گاهی اوقات مدل مشاهده‌شدهٔ مارکوف[۱] نیز نامیده می‌شود.

۱.۱.۱.۴ زنجیره مارکوف مرتبه بالا

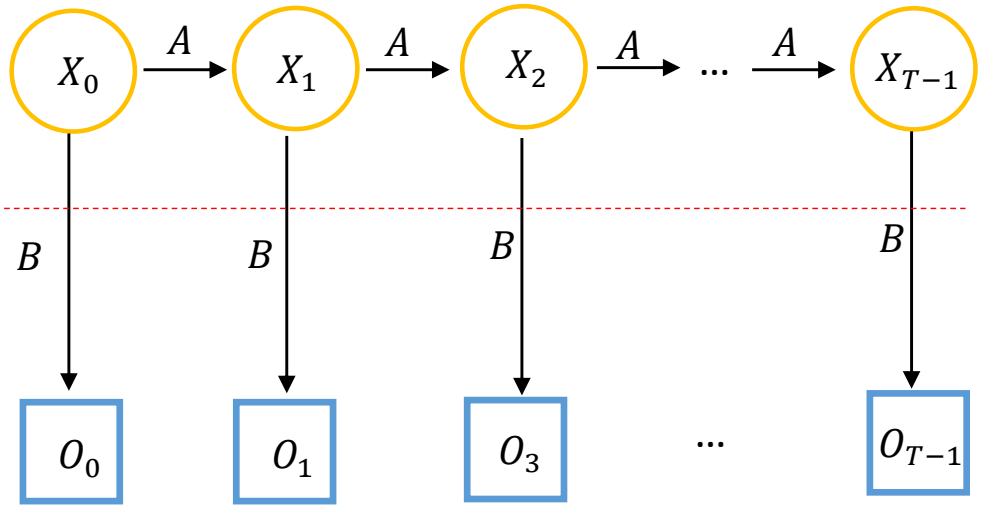
همان‌طور که در بالا ذکر شد زنجیرهٔ مارکوف مرتبه بالاتر، زنجیره‌ای است که در آن رویداد فعلی به بیش از یک رویداد قبلی بستگی دارد. فرآیند مارکوف مرتبه n بر روی الفبایی به تعداد m نماد می‌تواند

[۱] observed Markov model

به‌عنوان زنجیره مارکوف مرتبه اول با کاراکترهایی به تعداد $m \times n$ نمایش داده شود. برای مثال، الفبای دو نماد A و B را در نظر بگیرید. سپس توالی ABAAB را می‌توان به‌صورت جفت‌های متوالی (AB ، AA ، BA ، AB) که می‌تواند توسط مدل مارکوف مرتبه اول چهار وضعیتی با وضعیت‌های AB و AA ، BA ، BB و یا به‌عنوان مدل مارکوف مرتبه دوم نشان داد.

۲.۱.۴ مدل‌های پنهان مارکوف

مدل پنهان مارکوف یک مدل توالی[۱] است. مدل توالی یا کلاسه‌کننده توالی[۲] کارش این است که برچسب یا کلاسی را به هر واحد در یک دنباله اختصاص دهد. بنابراین توالی‌ای از مشاهدات را به دنباله‌ای از برچسب‌ها نگاشت می‌کنیم. HMM یک مدل توالی احتمالاتی است: با توجه به توالی واحد (کلمات، حروف، نمادها، شناسه‌ها و ...)، توزیع احتمال در توالی‌های احتمالی برچسب‌ها محاسبه می‌شود و بهترین ترتیب برچسب را انتخاب می‌کنند.



شکل ۲.۴: مدل پنهان مارکوف [۲۶].

---

۱ sequence model

۲ sequence classifier

$$P(O_t|X_0, X_1, \ldots, X_T, O_0, \ldots, O_{t-1}, O_{t+1}, \ldots, O_T) = P(O_t|X_t)$$

مدل‌های پنهان مارکوف به‌طورکلی برای تحلیل الگوهای آماری استفاده می‌شود. آن‌ها را می‌توان در تشخیص گفتار [۴۰] و تشخیص بدافزار و تجزیه و تحلیل توالی زیستی [۶۷] استفاده کرد. امروزه شاهد استفادهٔ گسترده از این مدل‌ها در حوزه تشخیص بدافزار نیز هستیم.

۱.۲.۱.۴ الگوریتم‌هایی برای امتیازدهی به دنبالهٔ ناشناخته در برابر یک مدل شناخته‌شده

در مدل‌های پنهان مارکوف سه مسئله اساسی مطرح است [۶۸،۲۶].

مسئله ۱: ارزیابی. با داشتن مدل پنهان مارکوف M و دنباله مشاهدات O، مقدار درست‌نمایی دنباله مشاهدات O با داشتن M تعیین می‌شود. به عبارت دیگر، می توانیم به توالی مشاهدات داده‌شده بر اساس یک مدل داده‌شده نمره دهیم؛ که با چه احتمالی دنباله داده‌شده با مدل داده‌شده مطابقت دارد.

$$P(O_1|M), P(O_1, O_2|M), \ldots, P(O_1, \ldots, O_T|M)$$

مسئله ۲: رمزگشایی. با توجه به مدل پنهان مارکوف و دنباله مشاهدات O، توالی مطلوبی را از انتقال‌های وضعیت پنهان X مرتبط با O تعیین می‌کنیم. به این ترتیب، می‌توانیم "بهترین" توالی وضعیت پنهان را کشف کنیم. در اینجا "بهترین" در معنای حداکثرسازی تعداد مورد انتظار وضعیت‌های پنهان صحیح است که بهترین مسیر ممکن را نشان می‌دهد.

$$P(X_1, \ldots, X_T|O_1, \ldots, O_T, M) = \frac{P(X_1, \ldots, X_T, O_1, \ldots, O_T|M)}{P(O_1, \ldots, O_T|M)}.$$

مسئله ۳: آموزش. با توجه به دنبالهٔ مشاهده‌شده O، تمام پارامترهای مدل پنهان مارکوف تنظیم گردد

تا احتمال رخداد دنباله O توسط مدل حداکثر شود. به عبارت دیگر، می‌توانیم یک مدل برای دنباله‌ای از مشاهدات O بسازیم.

- **الگوریتم به‌جلو**

با الگوریتم به‌جلو[۱] مسئله اول را می‌توان حل کرد. البته این مسئله با درنظرگرفتن تمام حالات ممکن حل‌شدنی است که همان راه‌حل بی‌رحمانه[۲] و البته ناکارآمد برای آن است. با استفاده از الگوریتم به‌جلو، به‌جای اینکه تمام حالات ممکن رخداد برای دنباله‌ای از مشاهدات محاسبه شود، فقط به‌صورت بازگشتی از مجموع امتیاز محاسبه‌شده توسط این الگوریتم برای زیردنباله مشاهدات رخداده در یک گام قبل‌تر استفاده می‌کند. با استفاده از آن امتیازی برای دنباله مشاهده داده‌شده برای مدل داده‌شده محاسبه می‌شود.

$$\alpha_t(j) = P(O_1, \ldots, O_t, X_t = j|M)$$

با استفاده از فرمول زیر $\alpha_1(j)$ را می‌توان محاسبه کرد.

$$\alpha_1(j) = P(O_1, X_1 = j|M) = P(X_1 = j|M)P(O_1|X_1 = j, M) = a_{Ij}b_j(O_1)$$

با فرض این‌که $|X| = m$ تعداد وضعیت‌های پنهان مدل باشد،

$$\alpha_t(j) = \sum_{i=1}^{m} P(O_1, \ldots, O_t, X_t = j, X_{t-1} = i|M) =$$
$$\sum_{i=1}^{m} P(O_1, \ldots, O_{t-1}, X_{t-1} = i|M) \times P(X_t = j|O_1, \ldots, O_{t-1}, X_{t-1} = i, M)$$
$$\times P(O_t|X_t = j, O_1, \ldots, O_{t-1}, X_{t-1} = i, M) = \sum_{i=1}^{m} \alpha_{t-1}(i)a_{ij}b_j(O_t)$$

- **الگوریتم ویتربی**

---

[۱] forward algorithm

[۲] brute force

برای حل مسئله دوم می‌توان از الگوریتم ویتربی استفاده نمود. این الگوریتم نیز مشابه الگوریتم به‌جلو یک نوع برنامه‌نویسی پویا است، با این تفاوت که در هر مرحله به‌جای مجموع احتمالات ممکن فقط بیشینهٔ مقدار احتمال ممکن برگردانده می‌شود. با این الگوریتم می‌توان بهترین مسیر از وضعیت‌های پنهان $X_1 \cdots X_T$ را برای دنباله رخدادهای $O_1 \cdots O_T$ برآورد کرد. در شکل ۳.۴ شبه کد الگویتم ویتربی را نشان داده‌ایم.

- **باوم ولش**

مسئله سوم را که فاز آموزش نیز به حساب می‌آید می‌توان با الگوریتم باوم ولش[1] حل نمود. این الگوریتم می‌تواند درست‌نمایی بیشینه[2] را برای پارامترهای انتقال و رخداد یک مدل پنهان مارکوف محاسبه کند. این الگوریتم جزو الگوریتم‌های یادگیری ماشین دسته‌بندی می‌شود. یعنی یک مجموعه داده از مشاهدات به‌عنوان داده‌های آموزشی در دسترس است و الگوریتم از روی این داده‌ها، پارامترهای مدل را تخمین می‌زند. توجه داشته باشید که این الگوریتم به داده‌هایی که توسط الگوریتم‌های Forward و Backward تولید می‌شوند نیاز دارد.

$$\alpha_t(j) = P(O_1, \ldots, O_t, X_t = j|M), \qquad \beta_t(i) = P(O_{t+1}, \ldots, O_T|X_t = i, M)$$

$$\beta_t(i) = P(O_{t+1}, \ldots, O_T|X_t = i) = \sum_{j=1}^{m} P(O_{t+1}, \ldots, O_T, X_{t+1} = j|X_t = i) =$$

$$\sum_{j=1}^{m} P(X_{t+1} = j|X_t = i)P(O_{t+1}|X_{t+1} = j, X_t = i)$$

$$P(O_{t+2}, \ldots, O_T|O_{t+1}, X_{t+1} = j, X_t = i) =$$

$$\sum_{j=1}^{m} a_{ij} b_j(O_{t+1})\beta_{t+1}(j)$$

---

[1] Baum-Welch algorithm

[2] Maximum likelihood

Initialize:

for each x

$\nu_{\text{next}}(x) = \log\left(P_{O(۱),X(۱)}\left(o(۱), x\right)\right)$

Iterate:

for t from ۲ to $T - ۱$

Swap $\nu_{\text{next}} \leftrightarrow \nu_{\text{old}}$

for each x_{next}

for each x_{old}

$\omega(x_{\text{old}}, x_{\text{next}}) = \nu(x_{\text{old}}, t) + \log\left(P(x_{\text{next}}|x_{\text{old}})\right)$

$+ \log\left(P(o(t+۱)|x_{\text{next}})\right)$

Find best predecessor

$B(x_{\text{next}}, t+۱) = \text{argmax}_{x_{\text{old}}}\, \omega(x_{\text{old}}, x_{\text{next}})$

Update ν

$\nu_{\text{next}}(x_{\text{next}}) = \omega(B(x_{\text{next}}, t+۱), x_{\text{next}})$

Backtrack:

$\bar{x} = \text{argmax}_x\, \nu_{\text{next}}(x)$

$\hat{x}(T) = \bar{x}$

for t from $T - ۱$ to ۱

$\bar{x} = B(\bar{x}, t+۱)$

$\hat{x}(t) = \bar{x}$

شکل ۳.۴: شبه کد الگوریتم ویتربی

۲.۲.۱.۴ مدل‌های نمایه پنهان مارکوف

در زمینه تشخیص بدافزار و دسته‌بندی آن‌ها شماری از تکنیک‌های خوشه‌بندی و طبقه‌بندی در یادگیری ماشین و داده‌کاوی برای طبقه‌بندی خانوادگی بدافزارها و شناسایی خانواده‌های جدید بدافزار، با روش‌های مبتنی بر رفتار استفاده می‌شوند. از مدل نمایه پنهان مارکوف می‌توان برای طبقه‌بندی فایل‌های مخرب به خانواده‌ها یا گروه‌ها بر اساس رفتار آن‌ها در سیستم میزبان استفاده کرد. PHMM در زمینه بیوانفورماتیک به‌طور گسترده‌ای استفاده شده است تا دنباله‌های پروتئینی و DNA مشابه را در یک پایگاه‌دادهٔ بزرگ جست‌وجو کند. استفاده از این مدل خاص به ما کمک می‌کند تا بر موانع مطرح‌شده توسط بدافزارهای چندریختی و فراریختی که در مجموعه بدافزارهای امروزی بسیار رایج است، غلبه کنیم. این مدل دقت طبقه‌بندی زیاد و قیاس‌برداری با سایر روش‌های پیشرفتهٔ مطرح دیگر در این حوزه دارد. مدل‌های نمایه پنهان مارکوف فرمول خاصی از روش مدل‌های پنهان مارکوف استاندارد است که برای مقابله با مشکلات اساسی در بیوانفورماتیک طراحی شده است [۶۷، ۶۹]. یکی از تفاوت‌های مهم بین HMMs و PHMMs این است که PHMMs آشکارا از اطلاعات مکانی (یا هم‌ترازی) موجود در توالی مشاهدات استفاده می‌کنند، در حالی که HMMs استاندارد این‌گونه نیست. تفاوت دیگر این است که بر خلاف HMMs استاندارد، PHMMs اجازه انتقال تهی[۱] را دارند، به‌طوری که برای تطابق توالی می‌تواند از درج و یا حذف استفاده کنند. در مورد DNA این تفاوت‌ها به‌طور طبیعی در طول تکامل[۲] رخ می‌دهد. ویروس فراریخت اغلب در هنگام فرآیند ایجاد تا حدودی مشابه فرآیند تغییر تکاملی[۳] است، بنابراین دلیلی برای این باور که PHMMs در این زمینه ممکن است مؤثر باشد وجود دارد. در تعیین توالی DNA می‌توانیم توالی‌های متعدد از ژن‌های شناخته‌شده را با برخی از روابط بیولوژیکی مورد نظر تراز کنیم. از نتایج هم‌ترازی توالی‌های چندگانه (MSA) می‌توان برای

---

[۱] null

[۲] evolution

[۳] evolutionary change

تعیین این‌که آیا یک دنبالهٔ ناشناخته ممکن است به MSA مربوط باشد، استفاده کنیم. هدف ما تعیین این نکته است که دنباله آپکد داده‌شده ممکن است به یک خانواده از ویروس که برای آن MSA ساخته شده است، تعلق داشته باشد. ساختار یک PHMMs را در شکل ۴.۴ نشان داده‌ایم. در شکل وضعیت حذف[۱] به شکل دایره (انتقال پوچ) است، وضعیت درج[۲] به شکل لوزی (فاصله[۳] برای هم‌ترازکردن توالی) و وضعیت تطابق[۴] به شکل مستطیل (وضعیت در HMMs استاندارد) است. توجه داشته باشید که احتمالات رخداد—که به ماتریس B در یک HMMs استاندارد مرتبط است— برخلاف یک HMMs استاندارد وابسته به موقعیت است. علاوه بر این، احتمالات رخداد از MSA مشتق شده است و بنابراین ایجاد MSA اساساً معادل با مرحله آموزش در یک HMMs استاندارد است. در نهایت وضعیت حذف اجازه می‌دهد تا مدل از گپ‌ها که همیشه در MSA وجود دارد برای رسیدن به دیگر وضعیت‌های رخداد، عبور کند. وجود چنین گپ‌هایی لازم است تا مانع از بروز بیش‌آموزشی[۵] شود. توجه به این مسئله نیز حائز اهمیت است که میزان فاصله‌ها به‌کار رفته در MSA بر روی دقت مدل آموزش‌دیده تاثیرگذار است و میزان گپ زیاد باعث کاهش دقت مدل می‌شود.

۳.۲.۱.۴ هم‌ترازی توالی‌های چندگانه

یک مفهوم اصلی در ساخت مدل نمایه پنهان مارکوف، MSA است. در توالی DNA هم‌ترازی می‌تواند استفاده شود تا تعیین گردد که آیا دنباله‌های ژنی از اجداد مشترک هستند یا خیر. در شناسایی بدافزار نیز برای فایل ناشناخته ورودی که به‌صورت دنباله‌ای از نمادها در آمده است، MSA از یک نمایه، می‌تواند برای تعیین این‌که آیا دنباله مربوط به آن است یا نه، مورد استفاده قرار گیرد. هم‌ترازی

---

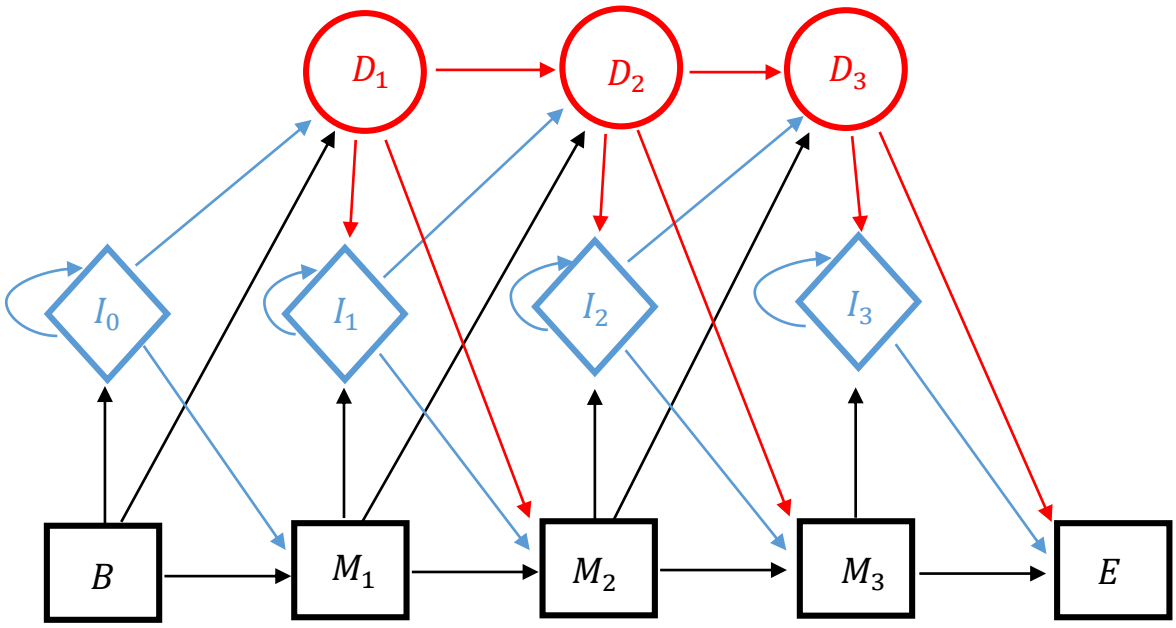
۱ delete

۲ insert

۳ gap

۴ match

۵ over-fitting

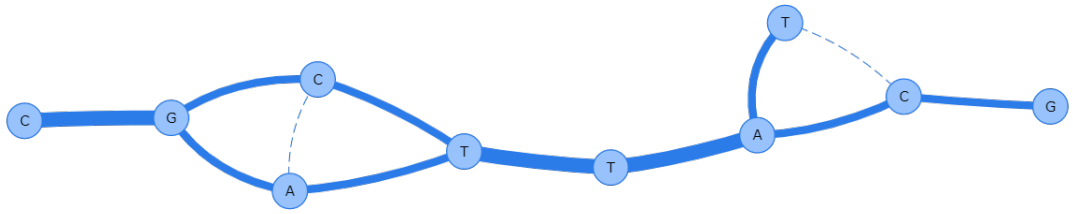


شکل ۴.۴: مدل نمایه پنهان مارکوف [۴۴، ۶۹].

دوگانهٔ دو توالی، یک جفت توالی با طول یکسان تولید می‌کند که تفاوت بین دو توالی اصلی را با واردکردن '-' یا فاصله‌ها تنظیم و طول هر دو توالی را یکسان می‌کند. هم‌ترازی کلی، هم‌ترازی که تطابق‌ها را به حداکثر و درج‌ها / حذف‌ها را به حداقل می‌رساند. مسئلهٔ هم‌ترازی محلی، تلاش می‌کند تا طولانی‌ترین زیرتوالی را از هر دنباله که مشابه هستند زیر یکدیگر قرار دهد. این کار را می‌توان برای هم‌ترازی چندین توالی نیز بسط داد. MSA خانواده‌ای از توالی‌های مشابه را نشان می‌دهد که زیرتوالی‌های مشابه در همهٔ آن‌ها وجود دارند. در حالی‌که راهکارهای مبتنی بر برنامه‌نویسی پویا برای هم‌ترازی دوگانه کارآمد هستند، اما هم‌ترازی چندگانه دارای پیچیدگی زمانی و حافظه از مرتبه $O(n^r)$ است و این باعث می‌شود که هزینهٔ اجرای آن بسیار زیاد باشد.

POA برنامه‌ای سریع برای هم‌ترازی توالی چندگانه در بیوانفورماتیک است. POA هم‌ترازی مطلوب برای هر توالی جدید، در مقایسه با هر توالی موجود در MSA که در نظر گرفته شود را تضمین می‌کند. مزایای POA عبارتند از: سرعت، مقیاس‌پذیری، حساسیت و توانایی برتر در مدیریت انشعاب /

ایندل‌ها[۱] در هم‌ترازی. این الگوریتم ساخت و ایجاد MSA برای ساختارهای عظیم و پیچیده را امکان‌پذیر می‌سازد [۷۰–۷۲]. هم‌ترازی ترتیبی بخشی[۲] (POA) الگوریتمی است که مبتنی بر گراف است و بدین صورت عمل می‌کند که همانند روش برنامه‌نویسی پویا، دو توالی به وسیله ماتریس امتیازها را بر اساس وضعیت‌های تطابق، عدم تطابق و فاصله در خود ذخیر می‌کند. در نهایت با تکنیک عقبگرد به ساختن توالی‌های هم‌ترازشده که بیشترین امتیاز را دارند، مبادرت می‌کند. تنها تفاوتی که الگوریتم POA با برنامه‌نویسی پویا دارد این است که به‌جای دو رشته توالی ارائه‌شده برای ماتریس مذکور از یک گراف و یک رشته (که به‌عنوان رشته‌ای که قرار است به MSA الحاق شود) استفاده می‌شود. به ازای هر بار اجرای برنامه‌نویسی پویا، گراف ساخته‌شده اصلاح و به‌روز می‌شود. نتیجه حاصل از این الگوریتم گرافی است که برخی از یال‌های آن ضخیم‌تر (دارای وزن بیشتر) و برخی با وزن کمتر هستند. در شکل ۵.۴ (برگرفته از[۳]) نمونه‌ای از گراف POA نشان داده شده است. این گراف برای تولید توالی اجماع[۴] استفاده می‌شود. ما از POA برای تولید MSA استفاده کرده‌ایم. مرحله ایجاد MSA عمدتاً به‌عنوان



شکل ۵.۴: گراف هم‌ترازی ترتیبی بخشی برگرفته از[۱]

یک مرحله آموزشی عمل می‌کند که در آن دنباله‌های بدافزار متعلق به یک خانواده را قرار می‌دهیم و در رویکرد خود برای امتیازدهی با استفاده از مدل PHMM در تشخیص بدافزار از الگوریتم ویتربی استفاده کرده‌ایم.

---

۱ handle branching/indels

۲ Partial Order Alignment

۳ http://simpsonlab.github.io/2015/05/01/understanding-poa/

۴ consensus sequence

۲.۴ رویکرد پیشنهادی

با توجه به فراگیر شدن استفاده از کامپیوترها و فضای مجازی، آسیب‌ها و مشکلات امنیتی در این حوزه بسیار جدی و پراهمیت است. یکی از مسائل درخور توجه در حیطه امنیت سیستم‌های هوشمند به‌ویژه کامپیوترها حملات سایبری و برنامه‌های مخرب هستند؛ تشخیص و شناسایی به‌هنگام بدافزارها با دقت و سرعت مناسب یکی از مهم‌ترین اهداف پژوهشگران در این حوزه است. برای شناسایی کدهای مخرب رویکرد پیشنهادی ما مبتنی بر رفتار ایستا و براساس تشخیص الگوی آماری است. در این روش با بهره‌گیری از مدل نمایه پنهان مارکوف به تشخیص خانواده‌های بدافزارهای چندریخت و فراریخت پرداخته‌ایم و با تغییر و اصلاح روند ایجاد MSA که گام یادگیری در این مدل محسوب می‌شود توانسته‌ایم نتایج بسیار مطلوبی کسب نمائیم.

۱.۲.۴ آماده‌سازی داده

یکی از مشکلاتی که در زمینه تحقیق روی بدافزارها وجود دارد و آن را با دیگر کارها کمی متمایز می‌نماید، این است که یک یا چند مجموعه دادهٔ عمومی وجود ندارد تا همه محققان کار خود را روی آن آزمایش نمایند و بتوانند ادعای خود را به خوبی به اثبات برسانند. در واقع نمی‌توان چنین مجموعه‌ای داشت و علت هم این است که بدافزارها روزبه‌روز در حال تولید هستند و اغلب بدافزارهای به‌وجودآمده از نسل‌های قبلی خود تواناتر و قوی‌تر هستند. از طرفی دیگر، بدافزارها فایل‌های خطرناکی هستند و نمی‌توان آن‌ها به راحتی در اختیار همه قرار داد و تنها مراکز آموزشی و پژوهشی امکان دسترسی به این مجموعه داده را دارند. در این پروژه از مجموعه داده شامل ۴ خانواده بدافزار فراریخت و یک مجموعه از فایل‌های سالم استفاده کرده‌ایم که توسط نرم‌افزار PE Explorer دیس‌اسمبل و کدهای عملیاتی مربوط به آن‌ها استخراج گردیده‌است. توضیحات کامل درباره خانواده‌های بدافزار در فصل ۲ بخش کیت‌های سازنده بیان شده است. برای هر خانواده تعداد ۵۰ نمونه از کیت‌های سازنده موجود در وب‌سایت VX

Heaven ایجاد شده است به استثناء خانواده NGVCK که تعداد ۲۰۰ نمونه دارد که جزء پیچیده‌ترین بدافزارهای فراریخت موجود است .مجموعه داده مذکور در مقالات [۹، ۳۵، ۵۲، ۷۳—۷۶] نیز به‌کار گرفته شده‌اند.

۲.۲.۴ مدل‌های پنهان مارکوف ترکیبی با ان- گرم

با توجه به مباحث مطرح‌شده در ابتدا به پیاده‌سازی مدل پنهان مارکوف که در تشخیص بدافزارهای فراریخت و چندریخت موفق ظاهر شده بودند، پرداختیم و برای استخراج ویژگی‌های بهتر و مؤثرتر برای اندازه‌گیری میزان شباهت از یکی از روشی‌هایی که در بازیابی اطلاعات برای استخراج الگو و البته در روش‌های تشخیص مبتنی بر امضاء بسیار مورد توجه قرار گرفته است نیز بهره بردیم. این روش ان- گرم نام دارد و بدین صورت کار می‌کند که در هر لحظه تعداد n بایت یا آپکد را در کنار هم به‌عنوان یک ویژگی در نظر می‌گیرد.

پیاده‌سازی مدل پنهان مارکوف و $n-gram$: پیاده‌سازی به زبان پایتون انجام شده است. در این روش ابتدا فایل‌های مخرب را به‌صورت خانواده‌های متمایز از هم تفکیک کردیم و از فایل‌های متعلق به هر خانواده، آپکدهای موجود در هر فایل را با استفاده از نرم‌افزارهایی که بدین منظور موجود هستند مانند PE Explorer استخراج کرده‌ایم. با توجه به فراوانی آپکدها در تمامی فایل‌ها در اینجا ۳۶ آپکد پرتکرار اول را به یک شماره شناسه نگاشت کرده‌ایم و شناسه ۳۷ را برای بقیه آپکدها درنظر گرفته‌ایم. بدین ترتیب برای هر فایل یک رشته دنباله از شناسه‌ها ایجاد می‌شود که آن‌ها را به‌عنوان ورودی برای ایجاد مدل پنهان مارکوف استفاده می‌کنیم و در نهایت فایل مدنظر را برای شناسایی نیز به‌صورت رشته مذکور تبدیل و به تمام مدل‌های ساخته‌شده برای هر خانواده ارائه می‌دهیم و امتیاز آن را براساس این‌که فایل موردنظر با احتمال بیشتر متعلق به کدام خانواده است، محاسبه و مشخص می‌نماییم. این روش در کارهای پیشین مربوط استفاده شده است و از نتایج آن برای مقایسه با رویکرد مدل پنهان مارکوف ترکیبی با ان- گرم بهره برده‌ایم. در ادامه از مفهوم $n-gram$ برای استخراج آپکدها استفاده کردیم و

به ازاء هر $n-gram$ با توجه به فراوانی تکرار آن‌ها در تمام فایل‌ها (در این‌جا ۱۰۰ان-گرم پرتکرار اول را در نظر گرفته‌ایم) دنباله شناسه را ایجاد نموده‌ایم و با آن‌ها مدل پنهان مارکوف را ایجاد کردیم که این امر به‌گونه‌ای محاسبه مدل پنهان مارکوف با مرتبه بالاتر را موجب می‌شود. نتایج حاصل از این پیاده‌سازی برای هر دو روش مناسب است ولی با استفاده از $n-gram$ فاصله امتیازهای به‌دست آمده بیشتر خواهد بود که به موجب آن می‌توان امتیاز آستانه بهتری تعیین و میزان مثبت کاذب کمتری داشت. نتایج مذکور را به‌طور کامل در فصل ۵ ارائه داده‌ایم.

۳.۲.۴ بهینه‌سازی مدل‌های نمایه پنهان مارکوف

همان‌طور که به آن اشاره شد یک مفهوم اصلی در ساخت مدل نمایه پنهان مارکوف، MSA است. در حقیقت مرحلهٔ یادگیری برای مدل محسوب می‌شود؛ در روش پیشنهادی به روند ایجاد MSA و تغییر و بهبود آن پرداخته‌ایم، ابتدا با تغییر الگوریتم به کار رفته برای ساخت MSA که از الگوریتم‌های مترقی هم‌ترازی چندگانه[۱] است و استفاده از POA به‌جای آن، توانستیم سرعت ایجاد MSA را به‌طور مطلوبی همان‌طور که در بخش هم‌ترازی توالی‌های چندگانه نیز به آن اشاره شد، کاهش دهیم. ایده اصلی برای بهبود مدل نمایه پنهان مارکوف، ایجاد MSAای که در آن بخش‌های مشترک در خانواده مدل‌شده پررنگ و قسمت‌های دیگر از اهمیت کمتری برخوردار شود. این امر با ایجاد و اضافه کردن توالی اجماع به MSA و ساخت مدل نمایه پنهان مارکوف براساس آن صورت گرفته است.

در روند ارزیابی و امتیازدهی براساس مدل ساخته شده، طول توالی ایجاد شده از فایل‌های ورودی بر امتیاز محاسبه‌شده اثر زیادی دارد و به‌همین دلیل در روند استخراج آپکدها و ایجاد توالی مربوط به هر یک از فایل‌ها از برخی آپکدها مانند NOP صرف‌نظر کردیم که باعث شد تاثیر تکنیک افزودن کد زائد در مبهم‌سازی کد کاهش یابد و نتایج بهتری حاصل شود.

---

[۱] progressive multiple alignment

۴.۲.۴ مدل‌های نمایه پنهان مارکوف ترکیبی با ان-گرم

پیاده‌سازی مدل نمایه پنهان مارکوف و $n-gram$: در روش مدل‌های پنهان مارکوف ترکیبی با ان-گرم به دلیل اینکه مدل ایجادشده بر اساس میزان رخداد هر آپکد است و موقعیت رخداد آن‌ها در نظر گرفته نمی‌شود، بر میزان دقت روش‌های مبتنی بر تحلیل رفتاری مورد استفاده که برای تشخیص خانواده فایل‌های مخرب به کارگرفته می‌شود، تاثیرگذار خواهد بود. لذا از روش دیگری که در تحلیل و شناسایی رشته‌های پروتئینی در محاسبات زیستی مرسوم است، استفاده می‌نماییم. در این روش همانند روش قبلی آپکدهای فایل‌های متعلق به یک خانواده را استخراج و یک رشته از شناسه‌ها را با آن می‌سازیم. از این رشته‌ها در ساختن MSA استفاده می‌نماییم که مرحله آموزش مدل نیز به حساب می‌آید. سپس از آن برای ساختن مدل نمایه پنهان مارکوف بهره می‌گیریم. در فاز بعدی از $n-gram$ برای ایجاد دنباله شناسه‌ها استفاده کرده‌ایم و در نهایت یک مدل نمایه پنهان مارکوف بهینه‌شده با استفاده از تغییر در MSA مذکور ایجاد کرده‌ایم که نتایج بسیار بهتری به‌همراه داشته است. در آخر پیاده‌سازی فوق را بر روی فایل‌های باینری و استخراج بایت به‌جای آپکدها انجام داده‌ایم و همچنین در روشی دیگر از ابزار موجود برای بازگشایی فایل‌ها و استخراج آپکدها به‌صورت مستقیم در برنامه پایتون استفاده کرده‌ایم. همچنین پیش‌پردازشی برای کاهش اثر منفی طول متفاوت فایل‌ها در یک خانواده انجام داده‌ایم و نتایج بهتری حاصل گردید. نتایج تمام روش‌های ذکرشده به همراه نمودارها و تحلیل آن‌ها را در فصل ۵ بیان کرده‌ایم.

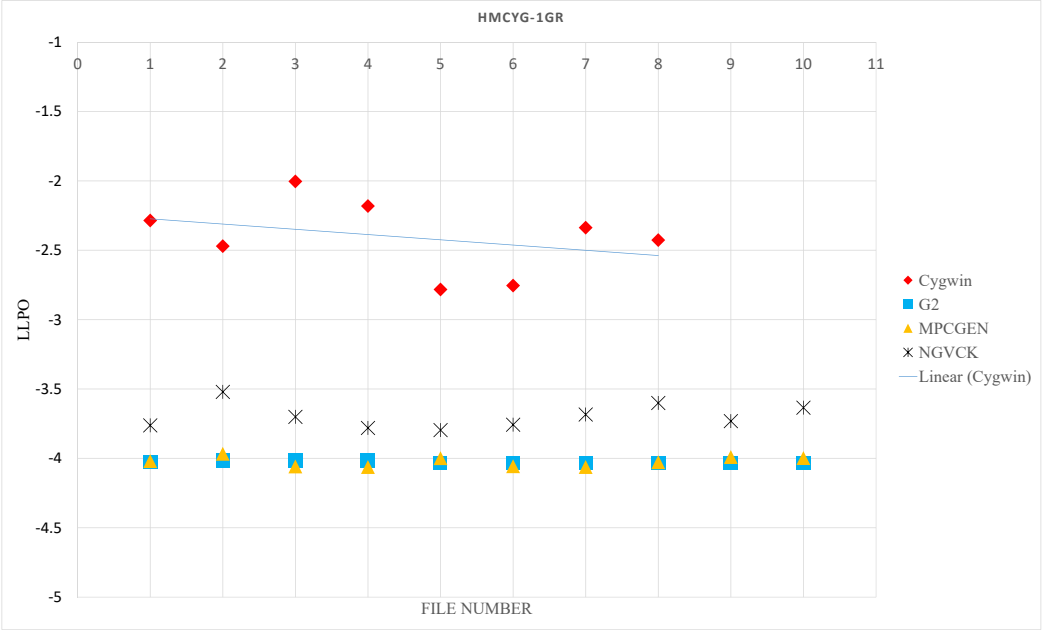
فصل پنجم

ارزیابی روش پیشنهادی

به منظور ارزیابی روش پیشنهادی، آزمایش‌هایی بر روی ۴۰۰ فایل متشکل از فایل‌های سالم و بدافزار فراریخت از ویروس‌های خانواده‌های VCL۳۲، NGVCK، MPCGEN، G۲ و فایل‌های سالم خانواده Cygwin صورت گرفته است. تمام این بدافزارها از وب سایت heaven VX دانلود شده‌اند. مجموعه فایل ذکرشده در ارزیابی‌های انجام‌شده برای روش‌های گوناگونی در تشخیص بدافزار که در مقالات [۹، ۳۵، ۵۲، ۷۳–۷۶] به آن‌ها پرداخته شده است، مورد استفاده قرار گرفته‌اند. نتایج آزمایش‌ها نشان می‌دهند در کنار دقت شناسایی ۱۰۰ درصدی، سادگی و سربار کم محاسباتی از مزایای روش پیشنهادی هستند. گام مهم و اساسی در ایجاد مدل نمایه پنهان مارکوف ایجاد MSA از توالی‌های حاصله از ویژگی‌های استخراج شده است. برای ایجاد MSA در این پژوهش از نرم‌افزار و ابزاری که در محاسبات زیستی از آن بسیار استفاده می‌شود و سرعت بسیار مناسب (مرتبه زمانی چندجمله‌ای) و مقیاس‌پذیری و حساسیت مناسبی دارد، استفاده کرده‌ایم.

۱.۵ ارزیابی مدل‌های پنهان مارکوف

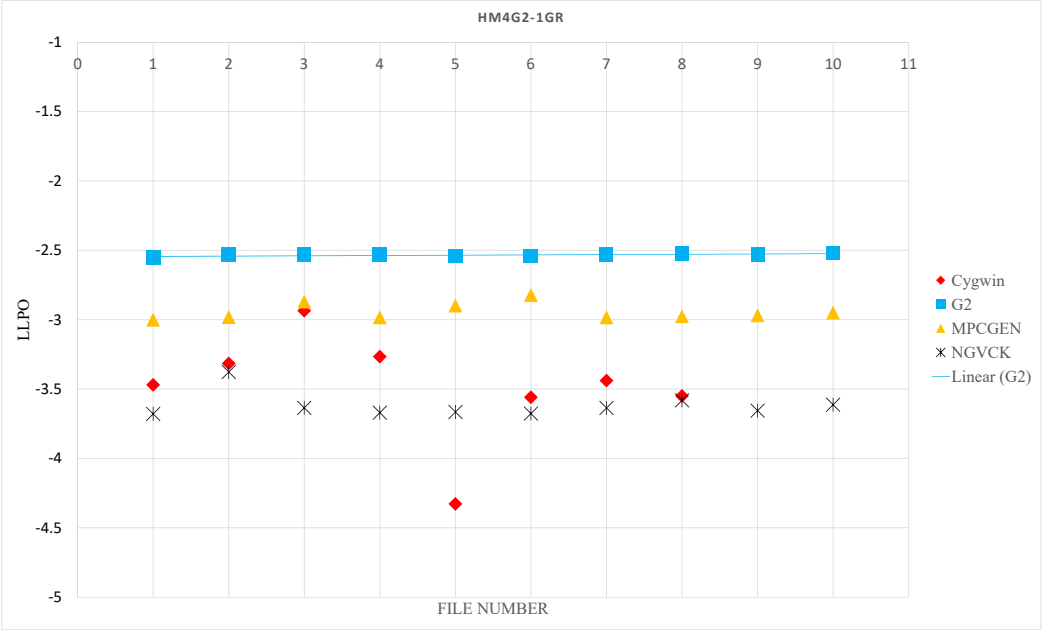
در رویکرد تشخیص بدافزار مبتنی بر رفتار و محتوا برای بدافزارهای فراریخت و چندریخت که در هر بار انتشار و تولید نسل بعدی کد خود را کاملاً تغییر می‌دهند، پیشنهاد می‌شود که از الگوهای آماری استفاده شود. چرا که می‌تواند با ایجاد یک مدل آماری و توزیع آماری مرتبط با آن، تحلیل آماری دقیقی مبتنی بر رفتار این‌گونه از بدافزارها داشته باشند. یکی از مدل‌های آماری رایج و پرکاربرد که در علوم رایانه از آن استفاده می‌شود، مدل‌های پنهان مارکوف است. امروزه در حل مسائلی در زمینه محاسبات زیستی، تشخیص گفتار، پردازش تصویر و تشخیص بدافزار و ... این روش به‌کار گرفته می‌شود. در ارزیابی‌های صورت‌گرفته برای روش مذکور نتایج به‌دست‌آمده در قالب نمودار به شرح زیر است. در



شکل ۱.۵: نمودار مدل خانواده Cygwin

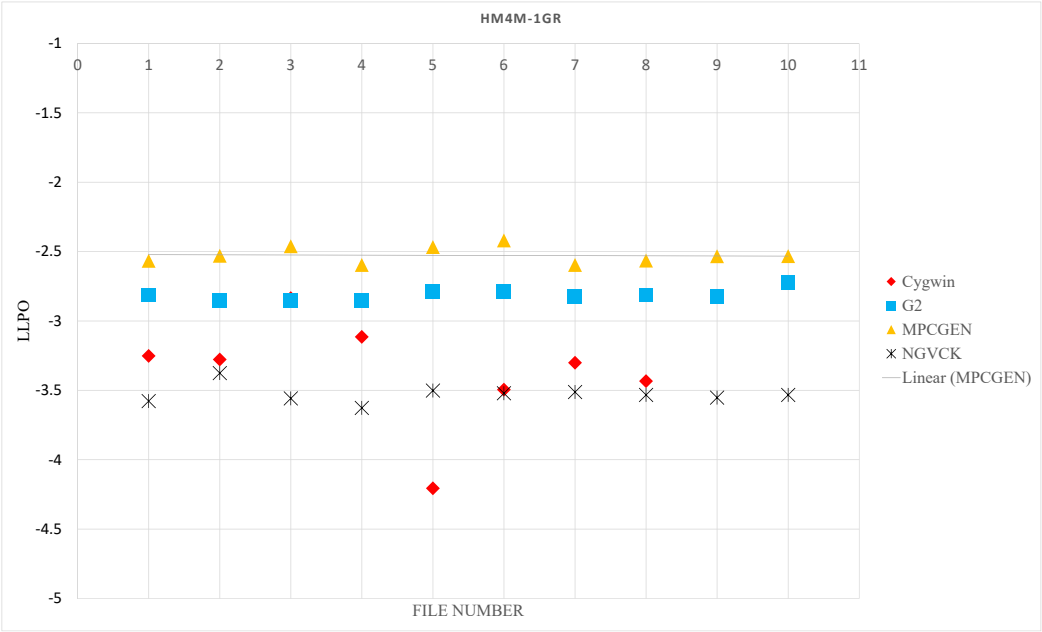
مدلی که بر اساس بدافزارهای متعلق به خانواده Cygwin ایجاد شده است، همان‌طور که مشاهده می‌نمایید به دلیل اینکه کیت‌های مولد بدافزارها بر این تلاش هستند تا بدافزارهای تولیدی خود را شبیه

به فایل‌های سالم بسازند، در نتیجه در ساخت کد خود از بخش‌هایی از فایل‌های سالم بهره می‌گیرند که باعث می‌گردد تا بدافزارهای تولیدی به فایل‌های سالم شبیه‌تر شوند. این مسئله در نمودار مدل خانواده NGVCK نیز به چشم می‌خورد تا جایی که امتیازهای محاسبه‌شده بسیار نزدیک به امتیازات نمونه‌های متعلق به خانواده Cygwin است. با توجه به نتایج ارائه‌شده در نمودار مدل خانواده G۲

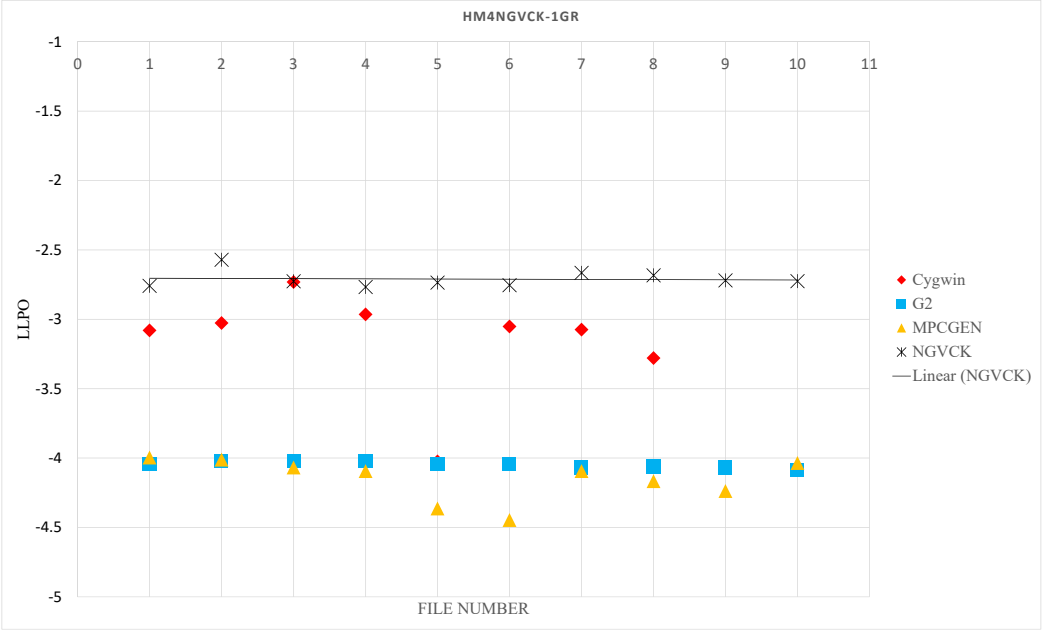


شکل ۲.۵: نمودار مدل خانواده G۲

و با درنظرگرفتن این که فایل‌های متعلق به این خانواده بسیار به هم شبیه هستند، می‌توان بیان کرد که مدل ساخته‌شده بسیار دقیق است و قوی عمل کرده است و همچنین تمامی نمونه‌های خانواده G۲ را به‌درستی از بقیه نمونه‌ها متمایز کرده است. همان‌طور که در فصل ۳ و در بخش معرفی کیت‌های سازندهٔ خانواده بدافزارها به آن اشاره کردیم، خانواده MPCGEN و خانواده G۲ متعلق به یک گروه سازنده برنامه‌های مخرب هستند و به همین علت تقریباً ساختار یکسانی نسبت به هم دارند که نمودار مدل خانواده G۲ و نمودار مدل خانواده MPCGEN گواهی بر این امر هستند.



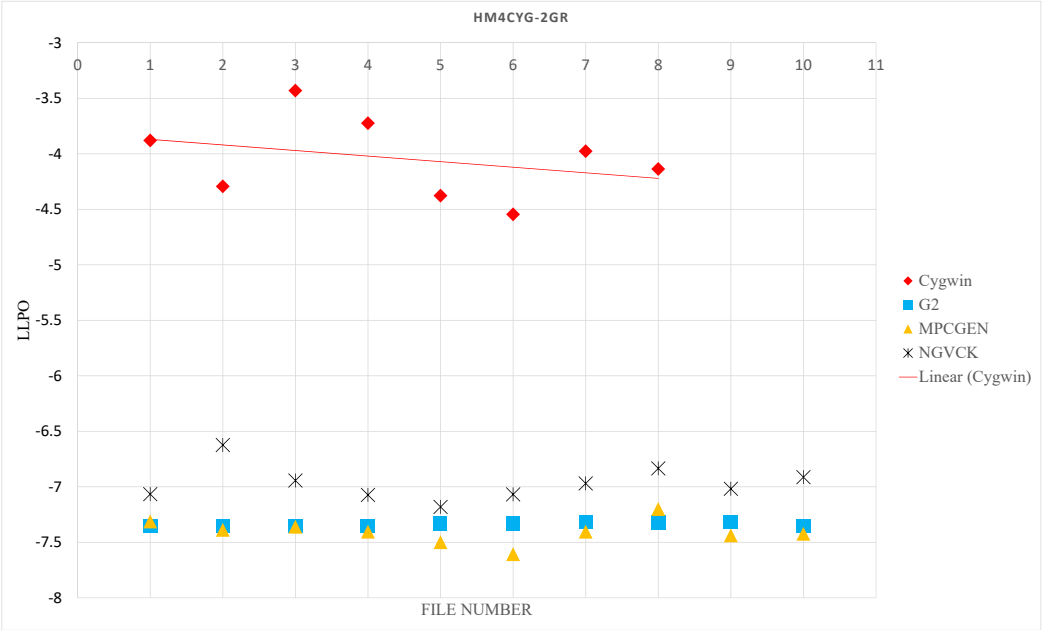
شکل ۳.۵: نمودار مدل خانواده MPCGEN



شکل ۴.۵: نمودار مدل خانواده NGVCK

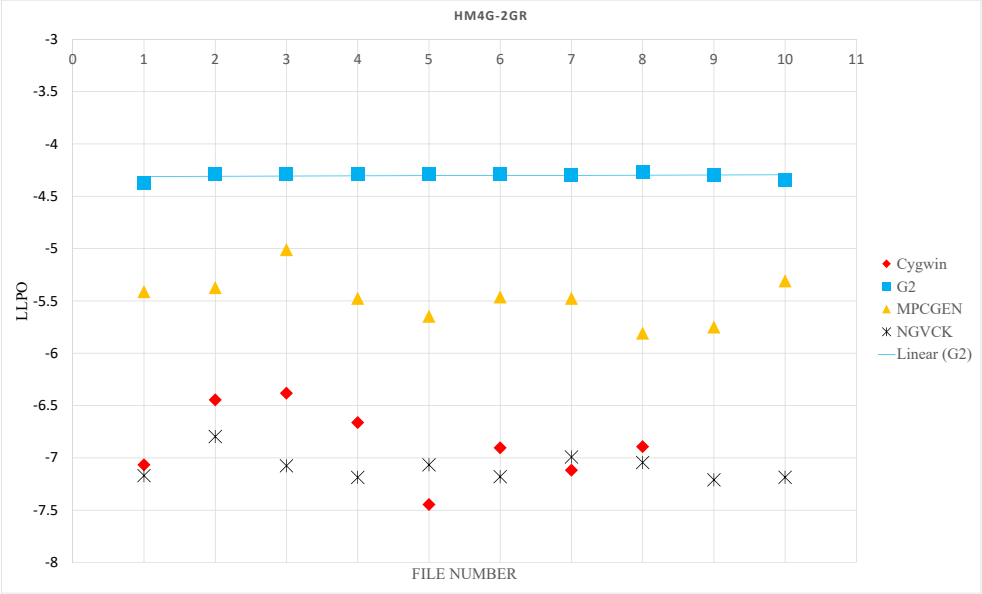
۱.۱.۵ ارزیابی مدل‌های پنهان مارکوف ترکیبی با ان-گرم

در ارزیابی‌های صورت‌گرفته برای روش مذکور نتایج مطلوبی به دست آمده است که در نمودارهای ۵.۵ تا ۸.۵ نشان داده‌ایم. با مقایسه دو نمودار متعلق به خانواده Cygwin و بررسی آن می‌توان نتیجه



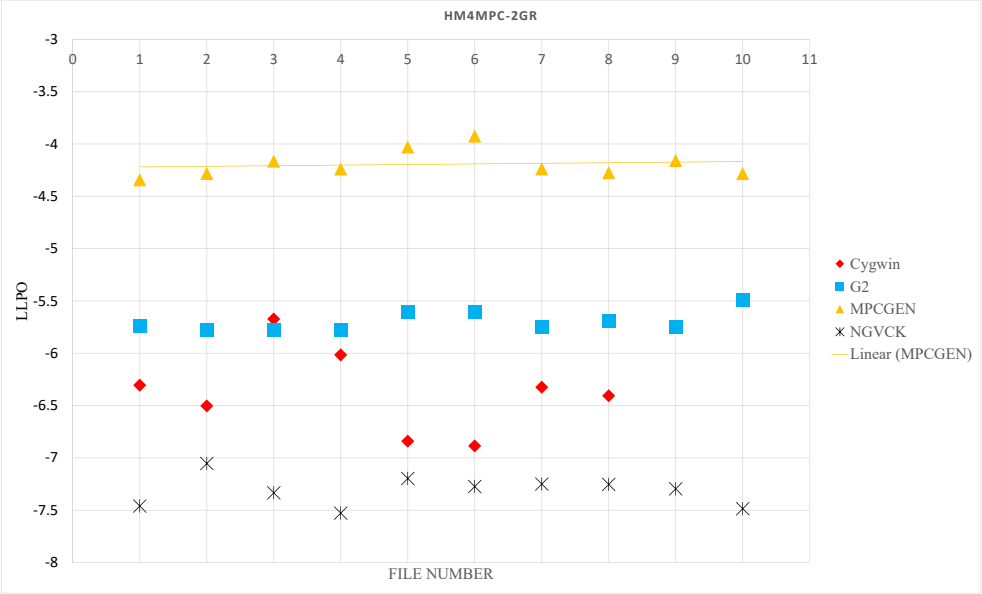
شکل ۵.۵: نمودار مدل خانواده Cygwin برای ۲-گرم

گرفت که استفاده از روش موسوم به ان-گرم با توجه به توضیحات بیان‌شده در فصل ۴ و در قسمت زنجیره مارکوف مرتبه بالا، برای استخراج ویژگی‌ها و بهره‌گیری از آن‌ها به‌عنوان ورودی برای ایجاد مدل پنهان مارکوف باعث ایجاد مدل پنهان مارکوف با مرتبه بالا (بیشتر از یک) خواهد شد و این موضوع باعث بهبود نتایج حاصل از مدل ساخته‌شده می‌شود. استفاده از مدل پنهان مارکوف با مرتبه بالا باعث ایجاد اختلاف و فاصله بین امتیازهای محاسبه‌شده برای نمونه‌های متعلق به خانواده‌های مختلف می‌شود. این موضوع می‌تواند برای تعیین یک حد آستانه مناسب برای هر یک از این خانواده‌ها مفید باشد و در نتیجه نرخ کاذب مثبت را کاهش دهد که باعث ایجاد یک مدل کارا برای تشخیص

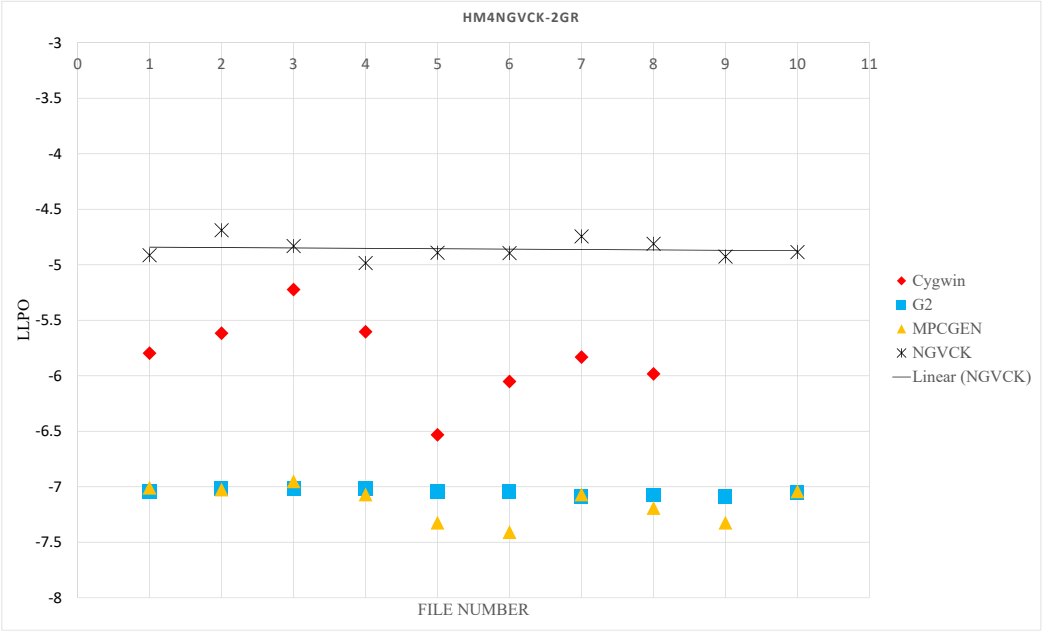


شکل ۶.۵: نمودار مدل خانواده G۲ برای ۲-گرم

بدافزارها به‌صورت کلاسه‌کننده آن‌ها در خانواده‌های مختلف شود.



شکل ۷.۵: نمودار مدل خانواده MPCGEN برای ۲-گرم

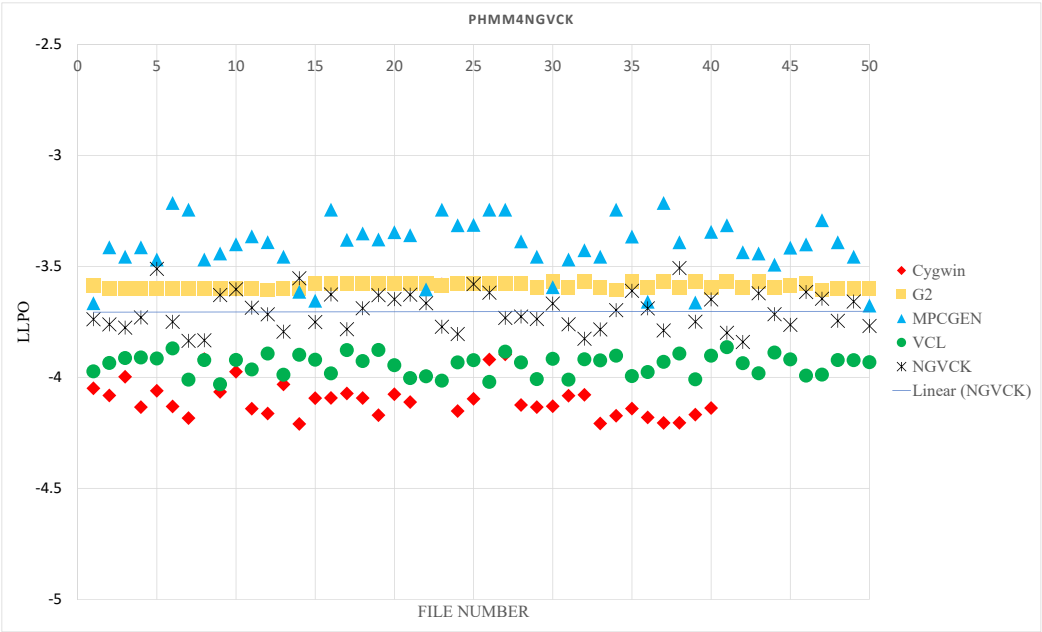


شکل ۸.۵: نمودار مدل خانواده NGVCK برای ۲-گرم

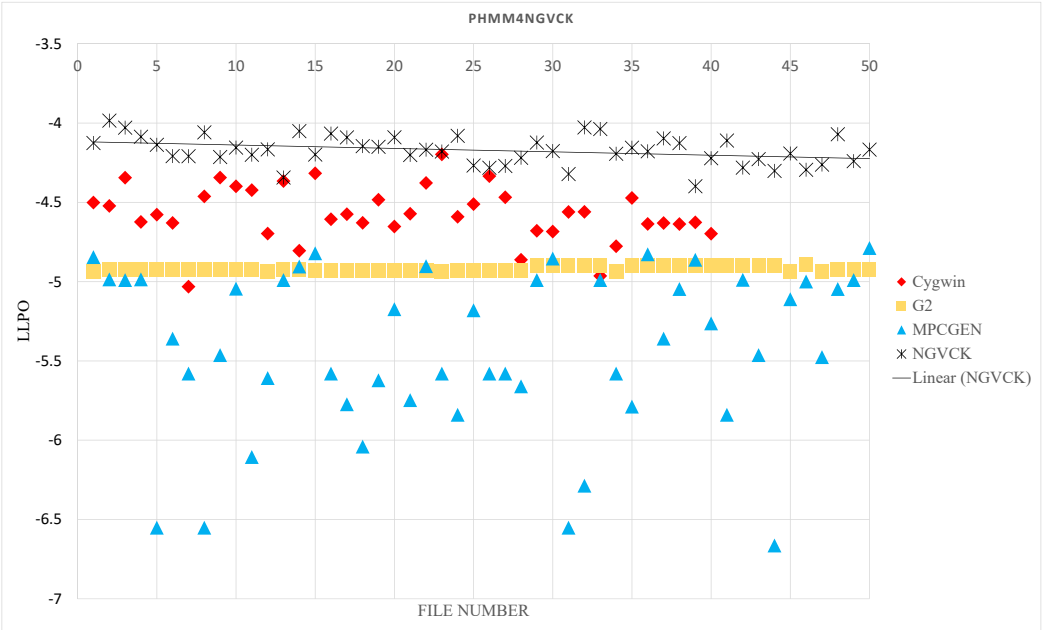
۲.۵ ارزیابی مدل‌های نمایه پنهان مارکوف

در مقاله [۷] اتالوری و همکارانش از مدل آماری موسوم به مدل‌های نمایه پنهان مارکوف که پیش از این در مسائل علوم زیستی از آن استفاده شده بود و نتایج حاصل از آن مطلوب و درخور توجه بود، برای تشخیص بدافزارها به ویژه بدافزارهای چندریخت و فراریخت استفاده کرده است. همان‌طور که قبلاً نیز به آن اشاره شد به دلیل ساختار و روند مشابه‌ای که تولید بدافزارهای فراریخت با مرحله تکامل ژن‌ها و رشته‌های پروتئینی دارند، استفاده از تکنیک‌ها و روش‌های رایج در زمینه محاسبات زیستی برای تشخیص این‌گونه از برنامه‌های مخرب در علوم رایانه مرسوم و معمول است. بخشی از مجموعه داده معرفی‌شده و به‌کاررفته در این مقاله نیز در این پروژه استفاده شده است. خانواده‌های VCL۳۲، MPCGEN و NGVCK در این مقاله مورد بررسی قرار گرفتند که نتایج آن بدین شرح است: مدل

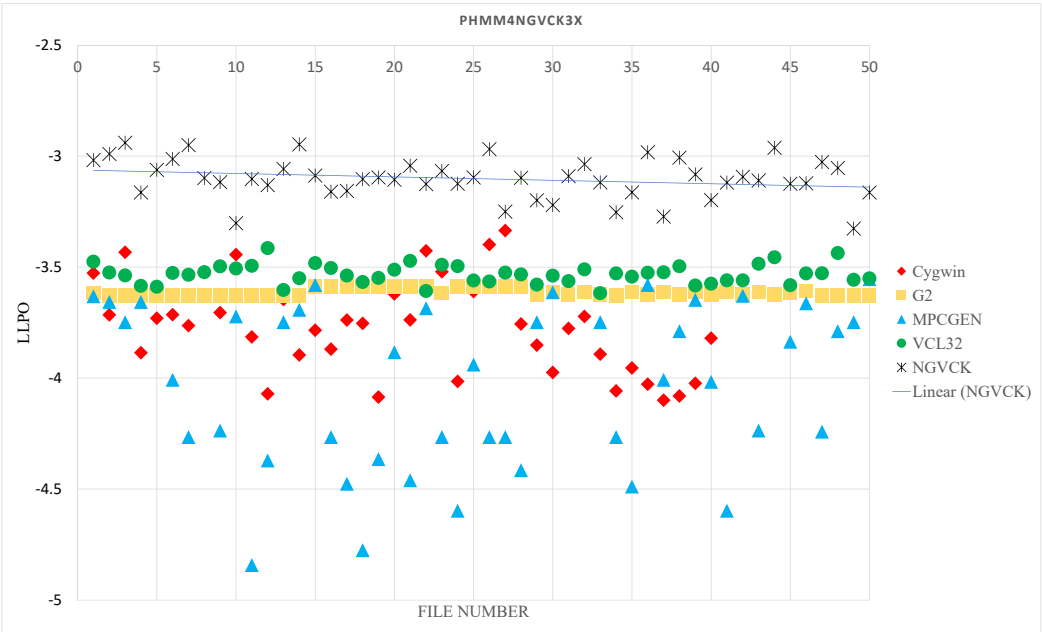
ایجادشده بر اساس خانواده VCL۳۲ توانسته است تمام نمونه‌های موجود را به‌درستی کلاسه کند. مدل آموزش دیده‌شده برای خانواده MPCGEN نیز توانسته با دقت ۱۰۰ درصد تمامی نمونه‌ها را تشخیص بدهد و شناسایی کند ولی برای نمونه‌های متعلق به NGVCK شکل ۹.۵ نتیجه مطلوبی برای اجراهای متفاوتی که توسط نویسنده انجام گرفته است، گزارش نشده است. در ارزیابی‌های صورت‌گرفته برای روش مذکور نتایج پذیرفتنی‌ای به دست آمد که به‌صورت نمودار ۱۱.۵ ارائه داده‌ایم.



شکل ۹.۵: نتایج کار پیشین برای خانواده NGVCK



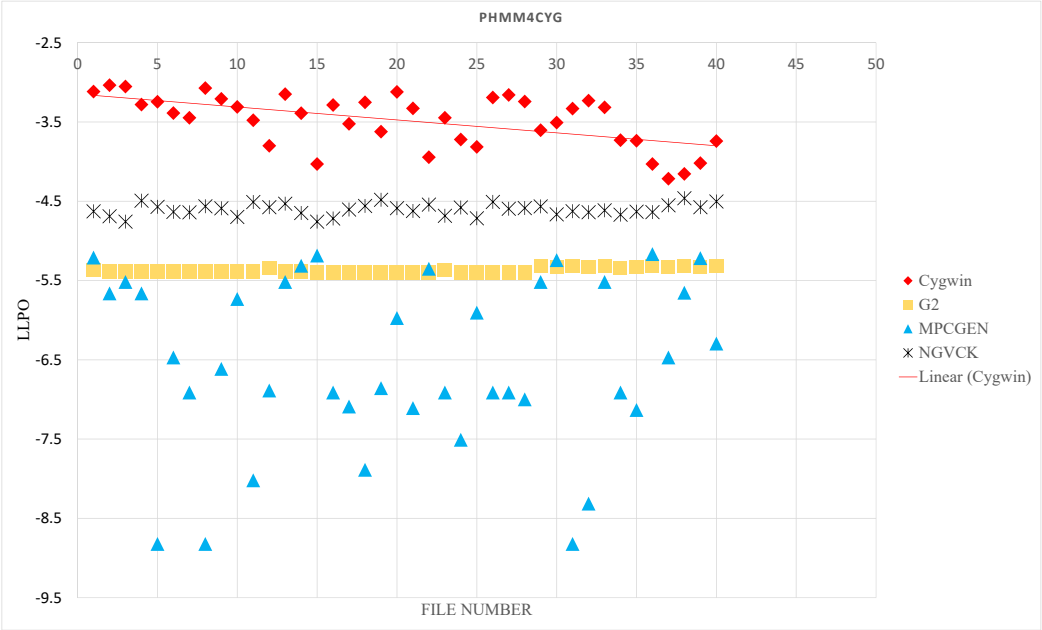
شکل ۱۰.۵: نتایج مدل خانواده Cygwin



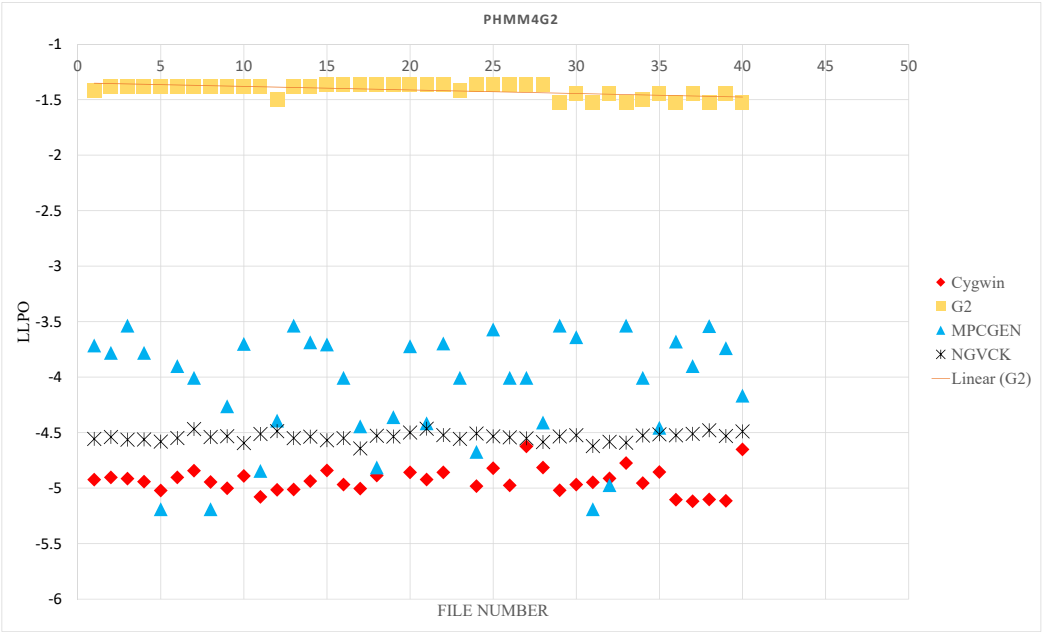
شکل ۱۱.۵: نتایج روش پیشنهادی مدل خانواده NGVCK

۱.۲.۵ ارزیابی مدل‌های نمایه پنهان مارکوف ترکیبی با ان-گرم

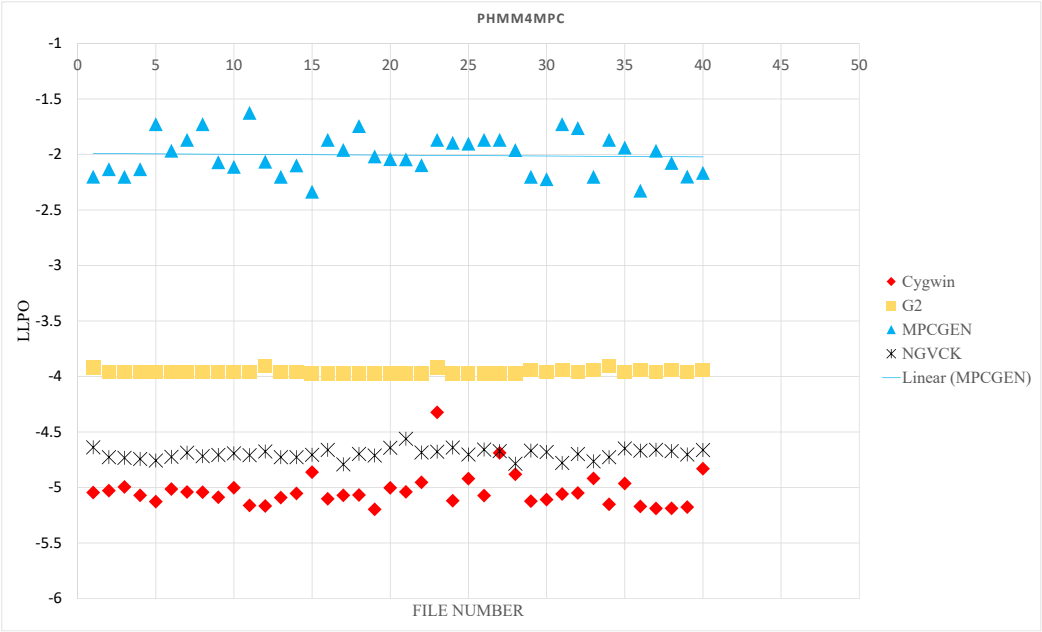
همانند مدل پنهان مارکوف، مدل نمایه پنهان مارکوف را نیز با روش ان-گرم به‌صورت ترکیبی استفاده کرده‌ایم که نتایج آن را به‌صورت نمودار ارائه می‌دهیم. با بررسی نتایج حاصل می‌توان نتیجه گرفت که همانند HMMs که با ان-گرم مدلی با مرتبه بالاتر ایجاد می‌کند و نتایج مطلوبی به همراه دارد؛ ولی باید توجه داشت که این امر برای PHMMs به دلیل تفاوت فاز یادگیری آن با HMMs که همان ایجاد MSA می‌باشد، می‌تواند مفید نباشد. استفاده از ان-گرم بر شمایل توالی‌های ساخته‌شده از ویژگی‌های استخراج‌شده از فایل‌ها تاثیر بسیاری دارد که این موضوع باعث افزایش نرخ فاصله یا همان گپ در MSA می‌شود. میزان نرخ بالای فاصله در MSA باعث کاهش دقت و کارایی مدل ایجادشده از MSA برای کلاسه‌کردن بدافزارها می‌شود. در نمودارهای ارائه‌شده در این بخش می‌توان آثار استفاده از روش ان-گرم را مشاهده و با هم مقایسه کرد.



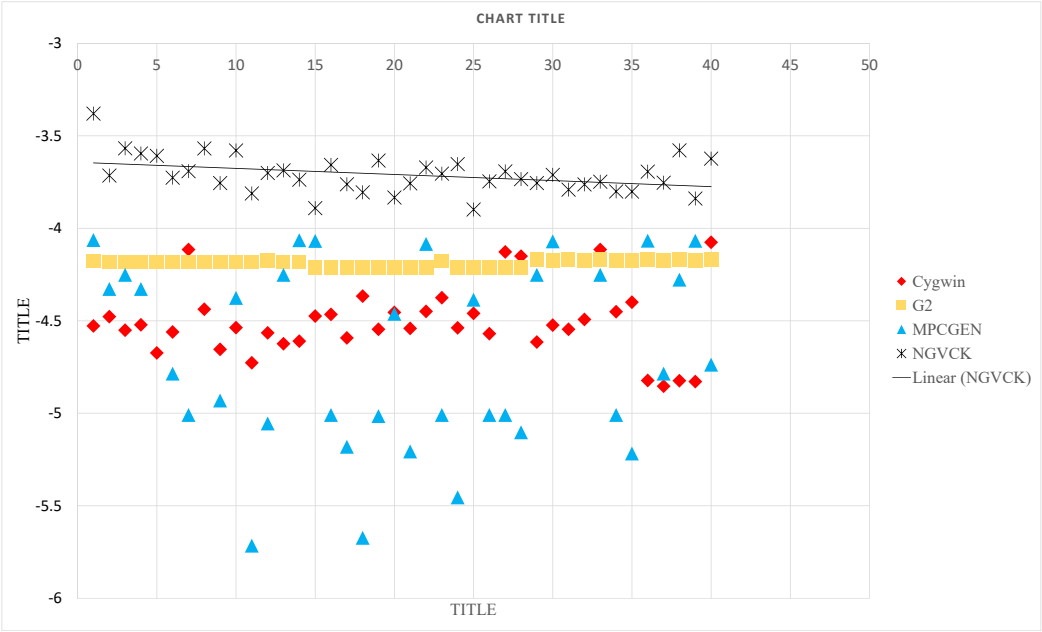
شکل ۱۲.۵: نمودار مدل خانواده Cygwin برای ۲-گرم



شکل ۱۳.۵: نمودار مدل خانواده G۲ برای ۲-گرم



شکل ۱۴.۵: نمودار مدل خانواده MPCGEN برای ۲-گرم



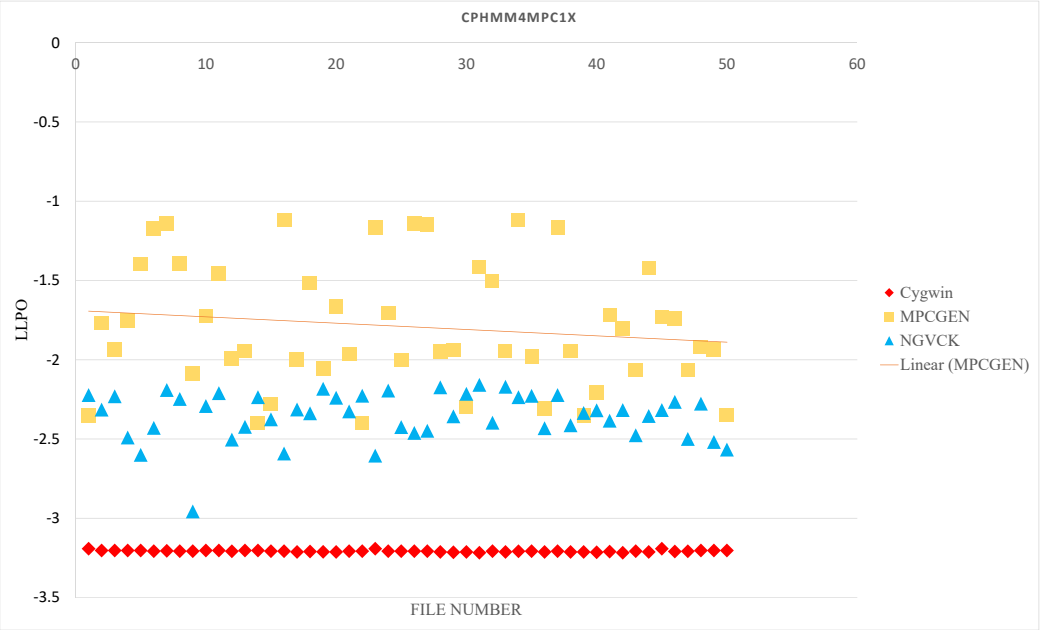
شکل ۱۵.۵: نمودار مدل خانواده NGVCK برای ۲-گرم

فصل ششم

نتیجه‌گیری و کارهای آتی

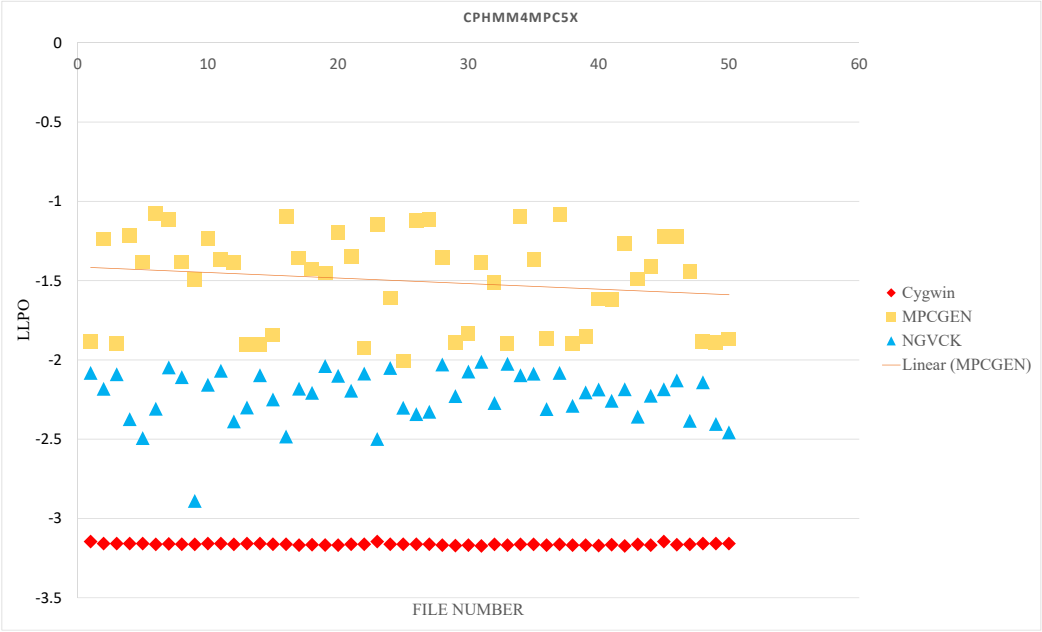
مدل‌های پنهان مارکوف توسط ریاضیدانان در اواخر دهه ۱۹۶۰ توسعه یافتند و از دهه ۱۹۸۰ میلادی مدل‌های پنهان مارکوف به‌طور مؤثر برای حل مشکلات بسیاری در علوم رایانه‌ای به کار رفته‌اند. در حقیقت، امروزه HMMها تکنیک یادگیری ماشین است. اخیراً متخصصان علوم زیستی مدل‌های پنهان مارکوف را برای برطرف‌کردن برخی از سخت‌ترین مشکلات در بیوانفورماتیک توسعه داده‌اند. PHMMها را می‌توان به‌عنوان فرمی بسیار اختصاصی از HMMهایی که برای رسیدگی به فرآیندهای تکاملی طراحی شده‌اند، در نظر گرفت. در این پژوهش، رویکرد استفاده از مدل نمایه پنهان مارکوف را برای مسئله تشخیص و طبقه‌بندی بدافزارهای مبتنی بر رفتار با تحلیل ایستای بدافزار، مورد بررسی قرار دادیم. این رویکرد به دلیل شباهت ذاتی تولید بدافزارهای فراریخت به روند تکامل و جهش در توالی ژن یا پروتئین در نظر گرفته شده است. نتایج کاملاً نشان‌دهنده کلاسه دقیق و عملکرد خوشه‌بندی خوب است، حتی زمانی که تعداد کمی از نمونه‌های آموزشی برای ساخت مدل‌ها وجود داشته باشند. در رویکرد ارائه‌شده در این پژوهش از تکنیک‌های مهندسی معکوس و دیس‌اسمبل برای استخراج ویژگی‌های مورد نیاز برای ایجاد مدل آماری، بهره گرفته‌ایم. این روش‌ها که برای پیش‌پردازش و آماده‌سازی داده‌ها به کار گرفته می‌شوند، سرعت انجام کار را کاهش می‌دهند و روند اجرای رویکرد

پیشنهادی را از حالت خودکار خارج می‌کنند. لذا در حال حاضر بر روی روشی که بر اساس داده‌های خام به‌دست آمده از خواندن فایل‌ها به‌صورت باینری است، تحقیق و بررسی می‌کنیم. که برای فاز اول فایل‌ها را بدون هیچ‌گونه فرآیند پیش‌پردازش، به‌عنوان ورودی به برنامه‌ای که به زبان پایتون پیاده‌سازی شده است، می‌دهیم و با خواندن آن‌ها به‌صورت بایت به بایت و به فرمت هگزادسیمال، ویژگی‌های مدنظر را با انجام پیش‌پردازشی که به وسیلهٔ برنامه مذکور انجام می‌شود، استخراج می‌نماییم. این روند کاملاً به‌طور خودکار انجام می‌شود و باعث می‌شود تا نیاز به دخالت نیروی انسانی به حداقل برسد و سرعت پردازش برای تحلیل بر روی بدافزارها در مقیاس‌های بزرگ افزایش پیدا می‌کند. نتایج به‌دست آمده در فاز اول در نمودارهای ۱.۶ و ۲.۶ آورده شده است. در فاز دوم به بررسی و ارائه روش‌هایی بر



شکل ۱.۶: نمودار مدل خانواده NGVCK

روی استخراج ابر ویژگی‌هایی برای خانواده بدافزارها می‌پردازیم. این روند با ایجاد فیلترهایی در روند پالایش داده‌های خام به‌دست‌آمده از فایل‌ها (بایت‌ها به فرمت هگزادسیمال) صورت خواهد گرفت. مراحل پیاده‌سازی و بررسی نتایج در حال انجام است. همچنین علاقه‌مند هستیم تا این رویکرد را برای



شکل ۲.۶: نمودار مدل خانواده NGVCK با تقویت رویکرد پیشنهادی

شناسایی خانواده‌های جدید بدافزار (مانند کرم‌های فراریخت و تروجان‌ها) و تشخیص فایل‌های سالم در کار آیندهٔ خود گسترش دهیم که ما را قادر به طراحی یک سیستم ضد بدافزار کامل می‌کند. همچنین روش‌هایی برای سرعت‌بخشیدن به الگوریتم‌ها در مدل‌های نمایه پنهان مارکوف و روش‌های مبتنی بر تحلیل پویا، برای استفاده در تحلیل نرم‌افزارهای مخرب در مقیاس بزرگ، می‌تواند مورد بررسی قرار گیرد.

مراجع

[1] Annachhatre, Chinmayee, Austin, Thomas H, and Stamp, Mark. Hidden markov models for malware classification. *Journal of Computer Virology and Hacking Techniques*, 11(2):59–73, 2015.

[2] Gharacheh, Mina, Derhami, Vali, Hashemi, Sattar, and Fard, Seyed Mehdi Hazrati. Detection of metamorphic malware based on hmm: A hierarchical approach. *International Journal of Intelligent Systems and Applications*, 8(4):18, 2016.

[3] Singh, Tanuvir. Support vector machines and metamorphic malware detection. 2015.

[4] Wang, Tingting and Xu, Ning. Malware variants detection based on opcode image recognition in small training set. in *Cloud Computing and Big Data Analysis (ICCCBDA), 2017 IEEE 2nd International Conference on*, pp. 328–332. IEEE, 2017.

[5] Galal, Hisham Shehata, Mahdy, Yousef Bassyouni, and Atiea, Mohammed Ali. Behavior-based features model for malware detection. *Journal of Computer*

Virology and Hacking Techniques, 12(2):59–67, 2016.

[6] Gharacheh, Mina, Derhami, Vali, Hashemi, Sattar, and Fard, Seyed Mehdi Hazrati. Proposing an hmm-based approach to detect metamorphic malware. in *Fuzzy and Intelligent Systems (CFIS), 2015 4th Iranian Joint Congress on*, pp. 1–5. IEEE, 2015.

[7] Attaluri, Srilatha, McGhee, Scott, and Stamp, Mark. Profile hidden markov models and metamorphic virus detection. *Journal in computer virology*, 5(2):151–169, 2009.

[8] Rezaei, Saeid, Afraz, Ali, Rezaei, Fereidoon, and Shamani, Mohammad Reza. Malware detection using opcodes statistical features. in *Telecommunications (IST), 2016 8th International Symposium on*, pp. 151–155. IEEE, 2016.

[9] Rezaei, Fereidoon, Hamedi-Hamzehkolaie, M, Rezaei, Saeid, and Payandeh, Ali. Metamorphic viruses detection by hidden markov models. in *Telecommunications (IST), 2014 7th International Symposium on*, pp. 821–826. IEEE, 2014.

[10] Jidigam, Ranjith Kumar, Austin, Thomas H, and Stamp, Mark. Singular value decomposition and metamorphic detection. *Journal of Computer Virology and Hacking Techniques*, 11(4):203–216, 2015.

[11] Christodorescu, Mihai and Jha, Somesh. Static analysis of executables to detect malicious patterns. tech. rep., Wisconsin Univ-Madison Dept of Computer Sciences, 2006.

[12] Deshpande, Prasad. Metamorphic detection using function call graph analysis. 2013.

[13] Deshpande, Sayali, Park, Younghee, and Stamp, Mark. Eigenvalue analysis for metamorphic detection. *Journal of computer virology and hacking techniques*, 10(1):53–65, 2014.

[14] Shanmugam, Gayathri, Low, Richard M, and Stamp, Mark. Simple substitution distance and metamorphic detection. *Journal of Computer Virology and Hacking Techniques*, 9(3):159–170, 2013.

[15] Shankarapani, Madhu K, Ramamoorthy, Subbu, Movva, Ram S, and Mukkamala, Srinivas. Malware detection using assembly and api call sequences. *Journal in computer virology*, 7(2):107–119, 2011.

[16] Vemparala, Swapna, Di Troia, Fabio, Corrado, Visaggio Aaron, Austin, Thomas H, and Stamo, Mark. Malware detection using dynamic birthmarks. in *Proceedings of the 2016 ACM on International Workshop on Security And Privacy Analytics*, pp. 41–46. ACM, 2016.

[17] Vemparala, Swapna. Malware detection using dynamic analysis. 2015.

[18] Park, Younghee, Reeves, Douglas S, and Stamp, Mark. Deriving common malware behavior through graph clustering. *computers & security*, 39:419–430, 2013.

[19] Qiao, Yong, He, Jie, Yang, Yuexiang, and Ji, Lin. Analyzing malware by abstracting the frequent itemsets in api call sequences. in *Trust, Security and Pri-*

vacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on, pp. 265–270. IEEE, 2013.

[20] Eskandari, Mojtaba, Khorshidpur, Zeinab, and Hashemi, Sattar. To incorporate sequential dynamic features in malware detection engines. in *Intelligence and Security Informatics Conference (EISIC), 2012 European*, pp. 46–52. IEEE, 2012.

[21] Eskandari, Mojtaba and Hashemi, Sattar. A graph mining approach for detecting unknown malwares. *Journal of Visual Languages & Computing*, 23(3):154–162, 2012.

[22] Anderson, Blake, Quist, Daniel, Neil, Joshua, Storlie, Curtis, and Lane, Terran. Graph-based malware detection using dynamic analysis. *Journal in computer Virology*, 7(4):247–258, 2011.

[23] Park, Younghee, Reeves, Douglas, Mulukutla, Vikram, and Sundaravel, Balaji. Fast malware classification by automated behavioral graph matching. in *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*, p. 45. ACM, 2010.

[24] Ahmed, Faraz, Hameed, Haider, Shafiq, M Zubair, and Farooq, Muddassar. Using spatio-temporal information in api calls with machine learning algorithms for malware detection. in *Proceedings of the 2nd ACM workshop on Security and artificial intelligence*, pp. 55–62. ACM, 2009.

[25] Dai, Jianyong, Guha, Ratan K, and Lee, Joohan. Efficient virus detection using

dynamic instruction sequences. *JCP*, 4(5):405–414, 2009.

[26] Damodaran, Anusha, Di Troia, Fabio, Visaggio, Corrado Aaron, Austin, Thomas H, and Stamp, Mark. A comparison of static, dynamic, and hybrid analysis for malware detection. *Journal of Computer Virology and Hacking Techniques*, 13(1):1–12, 2017.

[27] Islam, Rafiqul, Tian, Ronghua, Batten, Lynn M, and Versteeg, Steve. Classification of malware based on integrated static and dynamic features. *Journal of Network and Computer Applications*, 36(2):646–656, 2013.

[28] Eskandari, Mojtaba, Khorshidpour, Zeinab, and Hashemi, Sattar. Hdm-analyser: A hybrid analysis approach based on data mining techniques for malware detection. 9, 05 2013.

[29] Choi, Young Han, Han, Byoung Jin, Bae, Byung Chul, Oh, Hyung Geun, and Sohn, Ki Wook. Toward extracting malware features for classification using static and dynamic analysis. in *Computing and Networking Technology (ICCNT), 2012 8th International Conference on*, pp. 126–129. IEEE, 2012.

[30] Fan, Yujie, Ye, Yanfang, and Chen, Lifei. Malicious sequential pattern mining for automatic malware detection. *Expert Systems with Applications*, 52:16–25, 2016.

[31] Kazi, Shabana and Stamp, Mark. Hidden markov models for software piracy detection. *Information Security Journal: A Global Perspective*, 22(3):140–149, 2013.

[32] Santos, Igor, Brezo, Felix, Nieves, Javier, Penya, Yoseba K, Sanz, Borja, Laorden, Carlos, and Bringas, Pablo G. Idea: Opcode-sequence-based malware detection. in *International Symposium on Engineering Secure Software and Systems*, pp. 35–43. Springer, 2010.

[33] Tahan, Gil, Glezer, Chanan, Elovici, Yuval, and Rokach, Lior. Auto-sign: an automatic signature generator for high-speed malware filtering devices. *Journal in computer virology*, 6(2):91–103, 2010.

[34] Bilar, Daniel. Opcodes as predictor for malware. *International Journal of Electronic Security and Digital Forensics*, 1(2):156–168, 2007.

[35] Lin, Da and Stamp, Mark. Hunting for undetectable metamorphic viruses. *Journal in computer virology*, 7(3):201–214, 2011.

[36] Sridhara, Sudarshan Madenur and Stamp, Mark. Metamorphic worm that carries its own morphing engine. *Journal of Computer Virology and Hacking Techniques*, 9(2):49–58, 2013.

[37] Rad, Babak Bashari, Masrom, Maslin, and Ibrahim, Suhaimi. Evolution of computer virus concealment and anti-virus techniques: a short survey. *arXiv preprint arXiv:1104.1070*, 2011.

[38] Runwal, Neha, Low, Richard M, and Stamp, Mark. Opcode graph similarity and metamorphic detection. *Journal in Computer Virology*, 8(1-2):37–52, 2012.

[39] Desai, Priti. Towards an undetectable computer virus. 2008.

[40] Rabiner, Lawrence R. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[41] Gollery, Martin. Bioinformatics: Sequence and genome analysis. *Clinical Chemistry*, 51(11):2219–2219, 2005.

[42] Wong, Wing. Analysis and detection of metamorphic computer viruses. 2006.

[43] Ravi, Saradha, Balakrishnan, N, and Venkatesh, Bharath. Behavior-based malware analysis using profile hidden markov models. in *Security and Cryptography (SECRYPT), 2013 International Conference on*, pp. 1–12. IEEE, 2013.

[44] Huang, Lin and Stamp, Mark. Masquerade detection using profile hidden markov models. *computers & security*, 30(8):732–747, 2011.

[45] Cohen, Fred. Computer viruses: theory and experiments. *Computers & security*, 6(1):22–35, 1987.

[46] Schultz, Matthew G, Eskin, Eleazar, Zadok, Erez, and Stolfo, Salvatore J. Data mining methods for detection of new malicious executables. in *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on*, pp. 38–49. IEEE, 2001.

[47] Idika, Nwokedi and Mathur, Aditya P. A survey of malware detection techniques. *Purdue University*, 48, 2007.

[48] Vasudevan, Amit. Maltrak: tracking and eliminating unknown malware. in *Computer Security Applications Conference, 2008. ACSAC 2008. Annual*, pp.

311–321. IEEE, 2008.

[49] Gyongyi, Zoltan and Garcia-Molina, Hector. Web spam taxonomy. in *First international workshop on adversarial information retrieval on the web (AIRWeb 2005)*, 2005.

[50] Siddiqui, Muazzam, Wang, Morgan C, and Lee, Joohan. Data mining methods for malware detection using instruction sequences. *Proceedings of Artificial Intelligence and Applications, AIA 2008*, 2008.

[51] Ellis, Dan. Worm anatomy and model. in *Proceedings of the 2003 ACM workshop on Rapid malcode*, pp. 42–50. ACM, 2003.

[52] Canfora, Gerardo, Iannaccone, Antonio Niccolò, and Visaggio, Corrado Aaron. Static analysis for the detection of metamorphic computer viruses using repeated-instructions counting heuristics. *Journal of Computer Virology and Hacking Techniques*, 10(1):11–27, 2014.

[53] Austin, Thomas H, Filiol, Eric, Josse, Sebastien, and Stamp, Mark. Exploring hidden markov models for virus analysis: a semantic approach. in *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, pp. 5039–5048. IEEE, 2013.

[54] Attaluri, Srilatha. Detecting metamorphic viruses using profile hidden markov models. *Diss. San Jose State University*, 2007.

[55] Venkatesan, Ashwini. Code obfuscation and virus detection. 2008.

[56] Walenstein, Andrew, Mathur, Rachit, Chouchane, Mohamed R, and Lakhotia, Arun. Normalizing metamorphic malware using term rewriting. in *Source Code Analysis and Manipulation, 2006. SCAM'06. Sixth IEEE International Workshop on*, pp. 75–84. IEEE, 2006.

[57] Finones, Rodelio G and Fernandez, R. Solving the metamorphic puzzle. *Virus Bulletin*, pp. 14–19, 2006.

[58] Baysa, Donabelle, Low, Richard M, and Stamp, Mark. Structural entropy and metamorphic malware. *Journal of computer virology and hacking techniques*, 9(4):179–192, 2013.

[59] Lee, Jared, Austin, Thomas H, and Stamp, Mark. Compression-based analysis of metamorphic malware. *International Journal of Security and Networks*, 10(2):124–136, 2015.

[60] Sorokin, Ivan. Comparing files using structural entropy. *Journal in computer virology*, 7(4):259–265, 2011.

[61] Egele, Manuel, Scholte, Theodoor, Kirda, Engin, and Kruegel, Christopher. A survey on automated dynamic malware-analysis techniques and tools. *ACM computing surveys (CSUR)*, 44(2):6, 2012.

[62] Kolbitsch, Clemens, Comparetti, Paolo Milani, Kruegel, Christopher, Kirda, Engin, Zhou, Xiao-yong, and Wang, XiaoFeng. Effective and efficient malware detection at the end host. in *USENIX security symposium*, pp. 351–366, 2009.

[63] Priyadarshi, Sushant. Metamorphic detection via emulation. 2011.

[64] Alazab, Manoun, Layton, Robert, Venkataraman, Sitalakshmi, and Watters, Paul. Malware detection based on structural and behavioural features of api calls. 2010.

[65] Tang, Adrian, Sethumadhavan, Simha, and Stolfo, Salvatore J. Unsupervised anomaly-based malware detection using hardware features. in *International Workshop on Recent Advances in Intrusion Detection*, pp. 109–129. Springer, 2014.

[66] Hii, A. Chi-squared distance and metamorphic detection. *Master's report, Department of Computer Science, San Jose State University*, 2011.

[67] Durbin, Richard, Eddy, Sean R, Krogh, Anders, and Mitchison, Graeme. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press, 1998.

[68] Stamp, Mark. A revealing introduction to hidden markov models. *Department of Computer Science San Jose State University*, 2004.

[69] Eddy, Sean R. Profile hidden markov models. *Bioinformatics (Oxford, England)*, 14(9):755–763, 1998.

[70] Lee, Christopher, Grasso, Catherine, and Sharlow, Mark F. Multiple sequence alignment using partial order graphs. *Bioinformatics*, 18(3):452–464, 2002.

[71] Lee, Christopher. Generating consensus sequences from partial order multiple sequence alignment graphs. *Bioinformatics*, 19(8):999–1008, 2003.

[72] Grasso, Catherine and Lee, Christopher. Combining partial order alignment and progressive multiple sequence alignment increases alignment speed and scalability to very large alignment problems. *Bioinformatics*, 20(10):1546–1556, 2004.

[73] Govindaraju, Aditya. Exhaustive statistical analysis for detection of metamorphic malware. 2010.

[74] Fazlali, Mahmood, Khodamoradi, Peyman, Mardukhi, Farhad, Nosrati, Masoud, and Dehshibi, Mohammad Mahdi. Metamorphic malware detection using opcode frequency rate and decision tree. *International Journal of Information Security and Privacy (IJISP)*, 10(3):67–86, 2016.

[75] Khodamoradi, Peyman, Fazlali, Mahmood, Mardukhi, Farhad, and Nosrati, Masoud. Heuristic metamorphic malware detection based on statistics of assembly instructions using classification algorithms. in *Computer Architecture and Digital Systems (CADS), 2015 18th CSI International Symposium on*, pp. 1–6. IEEE, 2015.

[76] Wong, Wing and Stamp, Mark. Hunting for metamorphic engines. *Journal in Computer Virology*, 2(3):211–229, 2006.

واژه‌نامه انگلیسی به فارسی

ؤ

الف

آدرس مجازی نسبی ..... Relative Virtual Address (RVA)

آزمایشگاه تحلیل بدافزار ..... Mallab

ابهام‌زدا ..... Deobfuscator

اجرایی ..... Executable

اسمبلی ..... Assembly

اشکال‌زدا ..... Debugger

اَعمال بعد از آلودگی ..... Payload

افزودن ..... Insertion

اکتشافی ..... Heuristic

انتشار ..... Propagation

ان-گرم ..... N-gram

امضاء ..... Signature

ب

باج‌افزار ..... Ransomware

بازگشای کد ..... Disassembler

باینری ..... Binary

بدافزار ..... Malware

بدافزار درب پشتی ..... Backdoor

پ

پاک‌افزار ..... Cleanware

پخش‌کننده هرزنامه ..... Spam-relays

پنهان‌سازی ..... Stealth

پویش ..... Scan

ت

تانک، ریزتابع ..... Thunk

تبلیغ‌افزار ..... Adware

تحلیل ایستا ..... Static Analysis

تحلیل پویا ..... Dynamic Analysis

تحلیل رفتاری ..... Behavioral Analysis

تحلیل زمان‌اجرا ..... Runtime Analysis

تحلیل کد ماشین ..... Machine-code Analysis

تروجان ..... Trojan

تزریق کد ..... Code Injection

تشخیص سوء استفاده ..... Misuse Detection

تشخیص عمومی ..... Generic Detection

تشخیص گفتار ..... Speech Recognition

تطابق الگو ..... Pattern Matching

توالی چندگانه ..... Multiple Sequence

توکار ..... Embedded

ث

ثبت‌کننده ورودی صفحه کلید ..... Keylogger

ج

جاسوس‌افزار ..... Spyware

جایگشت ..... Permutations

جستار ..... Query

جعبه شنی ..... Sandbox

جهش ..... Mutate

چ

چندریخت ..... Polymorphic

چندریختی ..... Polymorphism

ح

حد آستانه ..... Threshold

حمله همه‌جانبه ..... Blended Threat

خ

خوب‌افزار ..... Goodware

خودتکرارکننده ..... Self-replicating

خودکپی ..... Self-carried

د

داده‌کاوی ..... Data Mining

ر

رمزگذاری ..... Encryption

رمزگشا ..... Decryptor

رمزگشای غیرخطی ..... Non-linear Decryption

روبات ..... Botnet

روت‌کیت ..... Rootkit

روز نخست ..... Zero-day

ز

زائد ..... Garbage

زیرروال ..... Subroutine

س

سریع ..... Flash

ش

شبیه‌سازی ..... Enumeration

ض

ضد اشکال‌زدا ..... Anti-debugger

ضد سپر امنیتی ..... Shellcode

ضد مهندسی معکوس ..... Anti-reversing

ضد ویروس ..... Anti-virus

ف

فایل ناشناخته ..... Unknown File

فشرده‌سازی ..... Compression

فرار ..... Evasion

فراریخت ..... Metamorphic

فراریختی ..... Metamorphism

ک

کد مخرب ..... Malcode

کدهای عملیاتی ..... Opcodes

کرم ..... Worm

کرم منفعل ..... Passive Worm

کنترل از راه دور ..... Remote Control

کیت‌ها سازنده ..... Construction Kits

گ

گسترش ..... Spread

گیرانداختن ..... Hooking

ل

لیست سیاه ..... Blacklist

م

مبتنی بر الگوی آماری ..... Statistical-based

مبتنی بر امضاء ..... Signature-based

مبتنی بر رفتار ..... Behavior-based

مبتنی بر ناهنجاری ..... Anomaly-based

مبهم‌سازی ..... Obfuscation

مثبت کاذب ..... False Positive

مرتب‌سازی مجدد ..... Reordering

معنایی ..... Semantic

مهندسی معکوس ..... Reverse Engineering

ن

نرم‌افزار پیش‌نصب شده ناخواسته ..... Bloatware

نرم‌افزار پیش‌نصب شده ناخواسته ..... Crapware

نرم‌افزار پیش‌نصب شده ناخواسته ..... Junkware

نرم‌افزار خاکستری ..... Grayware

نرم‌افزار مشکوک ..... Greyware

نگاشت حافظه ..... Memory Mapping

نماد ..... Symbol

نمایه ..... Profile

نیمه چندریخت ..... Oligomorphic

و

ویروس الحاق شونده ..... Appender

ویروس بوت سکتور ..... Boot Sector Virus

ویروس فراریخت ..... Metamorphic Virus

ویروس ماکرو ..... Macro Virus

ه

هدف ..... Target

هم‌ترازی ..... Alignment

هم‌زمان، در لحظه ..... On-the-fly

هیچ ..... None

هیچ ..... Null

ی

یکپارچه ..... Integrated

واژه‌نامه فارسی به انگلیسی

A

تبلیغ‌افزار ..... Adware

هم‌ترازی ..... Alignment

مبتنی بر ناهنجاری ..... Anomaly-based

ضد اشکال‌زدا ..... Anti-debugger

ضد مهندسی معکوس ..... Anti-reversing

ضد ویروس ..... Anti-virus

ویروس الحاق شونده ..... Appender

اسمبلی ..... Assembly

B

بدافزار درب پشتی ..... Backdoor

تحلیل رفتاری ..... Behavioral Analysis

مبتنی بر رفتار ..... Behavior-based

باینری ..... Binary

لیست سیاه ..... Blacklist

حمله همه‌جانبه ..... Blended Threat

نرم‌افزار پیش‌نصب شده ناخواسته ..... Bloatware

Boot sector virus ..... ویروس بوت سکتور

Botnet ..... روبات

C

Cleanware ..... پاک‌افزار

Code Injection ..... تزریق کد

Compression ..... فشرده‌سازی

Construction Kits ..... کیت‌ها سازنده

Crapware ..... نرم‌افزار پیش‌نصب شده ناخواسته

D

Data Mining ..... داده‌کاوی

Debugger ..... اشکال‌زدا

Decryptor ..... رمزگشا

Deobfuscator ..... ابهام‌زدا

Disassembler ..... بازگشای کد

Dynamic Analysis ..... تحلیل پویا

E

Embedded ..... توکار

Encryption ..... رمزگذاری

Enumeration ..... شبیه‌سازی

Evasion ..... فرار

Executable ..... اجرایی

F

False Positive ..... مثبت کاذب

Flash ..... سریع

G

Garbage ..... زائد

Generic Detection ..... تشخیص عمومی

Goodware ..... خوب‌افزار

Grayware ..... نرم‌افزار خاکستری

Greyware ..... نرم‌افزار مشکوک

H

Heuristic ..... اکتشافی

Hooking ..... گیرانداختن

Hybrid Approaches ..... روش‌های ترکیبی

I

Insertion ..... افزودن

Integrated ..... یکپارچه

J

Junkware ..... نرم‌افزار پیش‌نصب شده ناخواسته

K

Keylogger ..... ثبت‌کننده ورودی صفحه کلید

M

Machine-code Analysis ..... تحلیل کد ماشین

Macro Virus ..... ویروس ماکرو

Malcode ..... کد مخرب

Mallab ..... آزمایشگاه تحلیل بدافزار

Malware ..... بدافزار

Memory Mapping ..... نگاشت حافظه

Metamorphic ..... فراریخت

Metamorphic Virus ..... ویروس فراریخت

Metamorphism ..... فراریختی

Misuse Detection ..... تشخیص سوء استفاده

Multiple Sequence ..... توالی چندگانه

Mutate ..... جهش

N

N-gram ..... ان-گرم

None ..... هیچ

Non-linear Decryption ..... رمزگشای غیرخطی

Null ..... هیچ

O

Obfuscation ..... مبهم‌سازی

Oligomorphic ..... نیمه چندریخت

On-the-fly ..... هم‌زمان، در لحظه

Opcodes ..... کدهای عملیاتی

P

Passive Worm ..... کرم منفعل

Pattern Matching ..... تطابق الگو

Payload ..... اَعمال بعد از آلودگی

Permutations ..... جایگشت

Polymorphic ..... چندریخت

Polymorphism ..... چندریختی

Profile ..... نمایه

Propagation ..... انتشار

Q

Query ..... جستار

R

Ransomware ..... باج‌افزار

Relative Virtual Address (RVA) ..... آدرس مجازی نسبی

Remote Control ..... کنترل از راه دور

Reordering ..... مرتب‌سازی مجدد

Reverse Engineering ..... مهندسی معکوس

Rootkit ..... روت‌کیت

Runtime Analysis ..... تحلیل زمان‌اجرا

S

Sandbox ..... جعبه شنی

Scan ..... پویش

Self-carried ..... خودکپی

Self-replicating ..... خودتکرارکننده

Semantic ..... معنایی

Shellcode ..... ضد سپر امنیتی

Signature ..... امضاء

Signature-based ..... مبتنی بر امضاء

Spam-relays ..... پخش‌کننده هرزنامه

Speech Recognition ..... تشخیص گفتار

Spread ..... گسترش

Spyware ..... جاسوس‌افزار

Static Analysis ..... تحلیل ایستا

Statistical-based ..... مبتنی بر الگوی آماری

Stealth ..... پنهان‌سازی

Subroutine ..... زیرروال

Symbol ..... نماد

T

Target ..... هدف

Threshold ..... حد آستانه

Thunk ..... تانک، ریزتابع

Trojan ..... تروجان

U

فایل ناشناخته ..... Unknown File

W

کرم ..... Worm

Z

روز نخست ..... Zero-day

Abstract

With the rapid growth of computer malicious software (also known as malware) over recent decades, and the increasing number of threats posed by these malwares on the network environments (e.g. the Internet and intelligent environments), digital devices and computer systems have become more prone to being compromised, making it crucial to pay attention to the issue of the security of computer networks in information security and digital forensics. In order to provide security in this area, anti-malware techniques are very common. However, these techniques can have their own challenges such as the need to have an appropriate scheme and their side effects on the host systems. It seems that the general way to deal with malware is to use anti-virus software, but these software programs mostly use signature detection techniques and are not capable of identifying new malwares and zero-day threats. The method presented in this study is to identify species of malware families which are more sophisticated, obfuscated, and structurally diverse. This has made it more difficult for them to be detected compared to the malwares of the early stages. These generations of malware (a.k.a polymorphic and metamorphic malware) use methods that are called obfuscation techniques. We represent an approach that is based on data mining methods to preprocess binary and raw data files (i.e. reverse engineering) and a machine learning technique on malware families' behavioral characteristics. Reviews of recent work indicate that Hidden Markov Models are an attractive and acceptable option for malware detection. However, some of the malicious software can evade detection

techniques based on this method. Using sequences obtained by extracting important features of malware families, this study will walk through the process of modeling and training optimal Profile Hidden Markov Models. Since all parts of malware file are not of malicious essence, our aim is to distinguish the malicious portions from the benign portions, in order for the impact of the malicious portions on the process of malware detection to increase and for the benign parts not to have much effect.

Keywords: *Malware Detection, Metamorphic, Static Analysis, Profile Hid-den Markov Models.*

Institute for Advanced Studies
in Basic Sciences
Gava Zang, Zanjan, Iran

Static Malware Detection Using a Behavior-based Approach

Master's Thesis

Alireza Abbas Alipour

Supervisor: Dr. Ebrahim Ansari

September 2017