بررسی و مقایسه روش‌های کشف ویروس‌های فراریخت و چالش‌های موجود

علیرضا خلیلیان[1] و احمد براآنی[2]

[1] دانشجوی دکتری نرم‌افزار، دانشکده مهندسی کامپیوتر، دانشگاه اصفهان
khalilian@eng.ui.ac.ir
[2] دانشیار گروه مهندسی کامپیوتر، دانشکده مهندسی کامپیوتر، دانشگاه اصفهان
ahmadb@eng.ui.ac.ir

چکیده

نویسندگان ویروس‌های کامپیوتری برای جلوگیری از کشف ویروس‌ها از روش‌هایی بهره می‌برند که یکی از مؤثرترین آن‌ها، فراریختی است. برای نیل به این هدف، از روش‌های مبهم‌سازی کد استفاده می‌شود تا ضمن حفظ عملکرد کد، ساختار آن عوض شود. در سال‌های اخیر روش‌های متعددی برای شناسایی ایستای این نوع بدافزارها پیشنهاد شده‌اند، اما مداوماً در تولید ویروس‌های فراریخت روش‌های متنوع، جدید و پیچیده‌ای برای مبهم‌سازی کد به‌کار می‌رود. این مسئله شناسایی ایستای ویروس‌های فراریخت را دشوار می‌سازد. هر گونه جدید از مبهم‌سازی، هر شیوه به‌کارگیری آن و میزان و دانه‌بندی روش مبهم‌سازی به‌کار رفته در سادگی یا دشواری شناسایی ویروس فراریخت تأثیرگذار هستند. در این مقاله ۲۳ روش از آخرین روش‌های کشف ایستای ویروس‌های فراریخت مورد بررسی و مقایسه قرار می‌گیرد. برای هر روش، جزئیات آن، آزمایش‌های انجام شده و نتایج به‌دست آمده بیان شده است. سپس نقاط قوت و ضعف هر روش شناسایی شده و روش‌ها با هم مورد مقایسه قرار گرفته‌اند. این پژوهش، راهگشای تحقیقات آتی پیرامون کشف ویروس‌های فراریخت خواهد بود.

کلمات کلیدی: بدافزار، ویروس، فراریختی، مبهم‌سازی کد

۱- مقدمه

امروزه بدافزارها[1] با نرخ بالا رشد می‌کنند و هر روز چندین هزار از انواع آن‌ها پخش می‌شود [1]. از طرفی بدافزارها می‌توانند باعث خرابی سیستم‌های کامپیوتری شوند که سیستم‌های حیاتی، اقتصادی، مالی، نظامی، پزشکی، آماری، دولتی، نیروگاهی، تولیدی و غیره را کنترل می‌کنند. به این ترتیب بدافزار به‌راحتی می‌تواند خسارت‌های هنگفتی به دولت‌ها و سازمان‌ها و افراد وارد کند. برای شناسایی بدافزارها رقابت تنگاتنگی بین مهاجمین[2] سیستم‌های کامپیوتری و تولید کنندگان نرم‌افزارهای ضد بدافزار وجود دارد. برای طراحی و پیاده‌سازی ضد بدافزارها دو راه‌کار کلی پویا و ایستا بر اساس اجرا یا عدم اجرای کد مشکوک وجود دارد [1]. مشکلات موجود در روش‌های پویا باعث شده که هنوز هم روش‌های ایستا محبوبیت فراوانی داشته باشند.

یکی از انواع بدافزارها، ویروس‌ها یا کرم‌های[3] فراریخت[4] هستند [1]. این‌ها بدافزارهایی هستند که هنگام تکثیر، ساختار کد آن‌ها تغییر ریخت پیدا می‌کند به‌طوریکه با حفظ همان عملکرد قبلی، ظاهر کد متفاوتی داشته باشند. برای تغییر ریخت از روش‌های مبهم‌سازی[5] استفاده می‌شود. مبهم‌سازی روش‌های گوناگونی دارد که برخی از آن‌ها همچون استفاده از همروندی ممکن است فضای حالت بسیار بزرگی داشته باشند. از طرفی مولدهای خودکار نرم‌افزاری هم وجود دارند که از روی هر ویروس یک گونه جدید فراریخت شده تولید می‌کنند. اما در مطالعات [6] نشان داده شده که تولید ویروس‌های فراریخت مشکلاتی دارد و هنوز هم می‌توان مولدهای قوی‌تر و کاربردی‌تری برای ساخت ویروس‌های فراریخت با درجه بالاتر تولید کرد.

در سال‌های اخیر روش‌های متعددی برای شناسایی ایستای بدافزارهای فراریخت پیشنهاد شده‌اند، اما مداوماً در تولید ویروس‌های فراریخت روش‌های متنوع، جدید و پیچیده‌ای برای مبهم‌سازی کد به‌کار می‌رود. این مسأله شناسایی ایستای ویروس‌های فراریخت را دشوار می‌سازد. هر گونه جدید از مبهم‌سازی، هر شیوه به‌کارگیری آن و میزان و دانه‌بندی[6] روش مبهم‌سازی به‌کار رفته در سادگی یا دشواری شناسایی ویروس فراریخت تأثیرگذار هستند. به‌دلیل وجود چالش‌های مطرح شده، کشف کامل و قطعی همه انواع و گونه‌های ویروس‌های فراریخت با اعمال هر نوع مبهم‌سازی، هنوز نیاز به پژوهش بیشتر دارد و مسأله تحقیقاتی ارزشمندی است.

برای پژوهش در کشف ویروس‌ها و بدافزارهای فراریخت، باید عملکرد و نقاط قوت و ضعف روش‌های موجود را بشناسیم تا بتوان روش‌هایی مؤثرتر و کاراتر ارائه کرد. تا جایی که نویسندگان اطلاع دارند، تاکنون هیچ مطالعه‌ای در جهت تحلیل روش‌های موجود و مقایسه آن‌ها صورت نگرفته است. همین موضوع انگیزه و ضرورت تحقیق حاضر را روشن ساخته و ما را بر آن داشت تا در مقاله حاضر روش‌های موجود فراریختی را مورد بررسی قرار دهیم. برای

[1] Malware
[2] Attacker
[3] Worm
[4] Metamorphic
[5] Obfuscation
[6] Granularity

نیل به این هدف، در این مقاله ۲۳ مورد از آخرین روش‌ها و مطالعه‌های مطرح پیرامون کشف بدافزارها و اختصاصاً ویروس‌های فرایخت مورد بررسی قرار می‌گیرند. در مورد هر روش، ابتدا مشکل مطرح شده و روش پیشنهادی با جزئیات کافی بیان می‌شود. سپس آزمایش‌هایی که برای ارزیابی روش پیشنهادی صورت گرفته است، تشریح می‌گردند. در بخش بعدی نتایج حاصل از آزمایش‌ها بیان می‌شوند. در نهایت، در بخش جداگانه‌ای نقاط قوت و ضعف هر یک از روش‌ها جهت مقایسه ارائه می‌گردد. حاصل مطالعه‌های صورت گرفته در پژوهش حاضر، به پرسش‌های زیر پاسخ می‌دهد: (۱) نقاط ضعف ویروس‌های فرایخت فعلی چیست؟ (۲) چه نقطه ضعف‌هایی در روش‌های کنونی کشف ویروس‌های فرایخت وجود دارد؟ (۳) چگونه می‌توان روش‌های موجود را توسعه داد تا به روشی بهینه‌تر برسیم؟

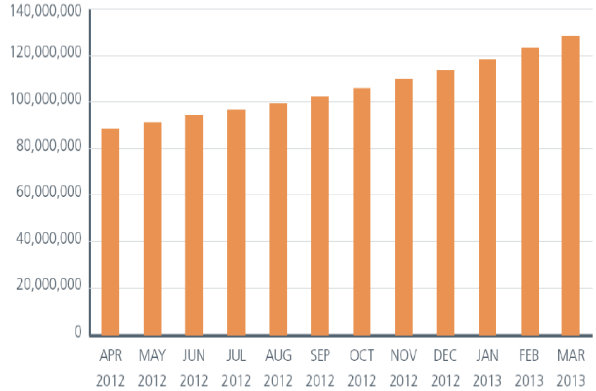
ساختار ادامه مقاله به‌شرح زیر است: در بخش دوم مقاله، انواع ویروس‌ها، نحوه کشف آن‌ها، ویروس‌های فرایخت، روش‌های مبهم‌سازی کد و ضعف فرایختی با جزئیات بیان می‌گردد. بخش سوم، ۲۳ روش اخیر را در کشف بدافزارهای فرایخت تشریح می‌نماید. در بخش چهارم، روش‌های مطرح شده با هم از نظر قوت و ضعف مقایسه می‌شوند. بخش پنجم هم به نتیجه‌گیری اختصاص دارد.

۲- مروری بر ادبیات موضوع

در این بخش تمام مفاهیم مورد نیاز به تفصیل مورد بحث قرار می‌گیرد. در ابتدا مفهوم بدافزار تشریح می‌شود. سپس ساختار و نحوه عملکرد ویروس و کرم که دو بدافزار مشهور هستند مورد بررسی قرار می‌گیرد. در ادامه پیرامون رمزگذاری ویروس‌ها و انواع نیمه چند ریختی و چند ریختی آن‌ها بحث می‌گردد. ویروس‌های فرایخت، تعریف رسمی آن‌ها، روش‌های مبهم‌سازی کد جهت تولید بدافزار فرایخت و نقطه ضعف آن‌ها در زیر بخش‌های بعدی تبیین می‌شوند. در نهایت، بخش دوم با کشف بدافزار و اختصاصاً کشف ویروس‌های فرایخت پایان می‌یابد.

۲-۱- ویروس و کرم

بدافزار برنامه کامپیوتری است که هدف آن آسیب زدن به سیستم‌های کامپیوتری است [4]. در سال‌های گذشته تعداد بدافزارها با سرعت نمایی رو به افزایش بوده است. شکل ۱ رشد بدافزارهای ثبت شده در پایگاه داده شرکت مک‌آفی را در مدت بین آوریل ۲۰۱۲ تا مارس ۲۰۱۳ نشان می‌دهد.

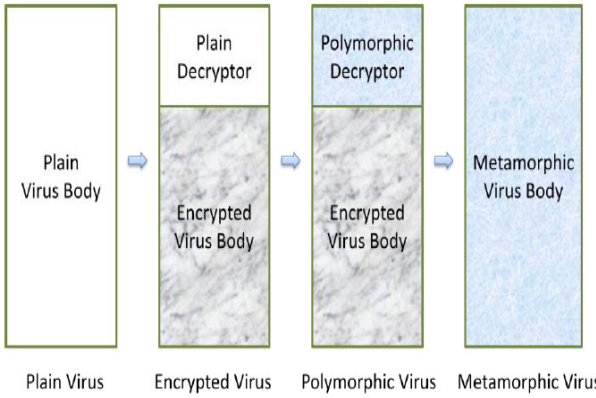


شکل ۱. تعداد نمونه‌های موجود در پایگاه داده آزمایشگاه مک‌آفی [8]

ویروس کامپیوتری بدافزاری است که خودش را با ادغام در فایل اجرایی دیگر، اجرا می‌کند. فایلی اجرایی که ویروس خودش را به آن می‌چسباند، آلوده می‌نامند [2]. وقتی برنامه آلوده شده اجرا می‌شود، کد ویروس هم به تبع آن اجرا شده و سایر فایل‌ها را نیز آلوده می‌سازد و حاصل آن ایجاد خرابی‌ها و مشکلاتی در سیستم میزبان است [4]. لذا می‌توان گفت ویروس‌ها برنامه‌های خود تکثیری[1] هستند که باعث خرابی در داخل سیستم میزبان می‌شوند [11, 14].

در مقایسه با ویروس‌ها، کرم‌ها بدافزارهای مستقلی[2] هستند و معمولاً تحت شبکه منتقل می‌شوند [4]. کرم هم یک بدافزار خود تکثیری است. هر کرم معمولاً خودش را از یک سیستم به سیستم دیگر از طریق شبکه منتقل می‌کند و برای این‌کار باید حفره امنیتی یا آسیب‌پذیری[3] در سیستم هدف وجود داشته باشد [1]. کرم‌ها نه تنها می‌توانند به سیستم میزبان خود آسیب وارد کنند، بلکه حتی سایر سیستم‌های تحت شبکه را نیز مورد حمله و تخریب قرار می‌دهند.

ویروس‌ها و کرم‌ها اغلب از روش‌های مبهم‌سازی استفاده می‌کنند تا روش‌های شناسایی بدافزار که مبتنی بر امضا[4] کار می‌کنند، نتوانند آن‌ها را شناسایی نمایند [4]. هدف از مبهم‌سازی این‌است که هر بار که بدافزار می‌خواهد منتشر شود و خودش را تکثیر نماید[5]، نسخه[6] یا شکل دیگری از خودش بسازد به‌طوریکه پویشگرهای[7] امضای ضد بدافزارها نتوانند آن‌ها را شناسایی کنند یا شناسایی آن‌ها دشوار شود. در حوزه بدافزارها، روش‌هایی مثل رمزگذاری[8]، نیمه چند ریختی[9]، چند ریختی[10] و فرایختی از جمله روش‌هایی هستند که تولیدکنندگان بدافزار برای مبهم‌سازی ساختار بدافزار مورد استفاده قرار می‌دهند. برای تکثیر و تولید نسخه‌های گوناگون بدافزار با مبهم‌سازی، دو روش موجود است [1]: یکی گذاشتن موتور تکثیر داخل خود بدافزار و دیگری داشتن یک نرم‌افزار مستقل که روی یک سیستم اجرا شود و یک‌باره چندین نسخه تکثیری از یک بدافزار بسازد. شکل ۲ تکامل ویروس‌ها را نشان می‌دهد.



شکل ۲. تکامل ویروس، به‌ترتیب از چپ به راست: ویروس معمولی، ویروس رمزگذاری شده، ویروس چندریختی، ویروس فرایختی [25]

[1] Self-replicating
[2] Standalone
[3] Vulnerability
[4] Signature-based
[5] Generation
[6] Replica
[7] Scanner
[8] Encryption
[9] Oligomorphism
[10] Polymorphism

۲-۲- ویروس‌های فراریخت

فراریختی فرایند تبدیل بخشی از نرم‌افزار به نمونه‌های منحصر به‌فرد است [3]. کپی‌های فراریخت یک نرم‌افزار از نظر عملکرد یکسان هستند ولی ساختار متفاوتی دارند. ویروس فراریخت برای حل مشکل ویروس‌های چند ریختی طراحی شده است. این ویروس‌ها از توابع رمزگذاری و رمزگشایی استفاده نمی‌کنند [2]. در عوض، ویروس هنگام تکثیر کل ساختار را تغییر می‌دهد به‌طوریکه ویروس جدیدی به‌دست آید با عملکرد همسان با ویروس قبلی و الگوی متفاوت. به این ترتیب حتی اجرای ویروس در محیط مجازی یا همان روش تقلید هم دیگر جوابگو نیست و ویروس از شناسایی شدن توسط ضد ویروس می‌گریزد [4]. به این ویروس‌ها، فراریخت یا دگردیس شده می‌گویند. بخش کلیدی هر ویروس فراریخت موتور جهش آن است [2]. وظیفه این بخش تغییر شکل بدنه ویروس است و این‌کار با اعمال تبدیلات حافظ معنی روی کد صورت می‌گیرد. موتور جهش می‌تواند مستقل از ویروس فراریخت تولیدی باشد یا به‌عنوان بخشی از بدنه آن داخل ویروس تعبیه[1] شده باشد [2]. در حالت اول درجه فراریختی بسیار بالا می‌رود زیرا خود موتور جهش نیاز به تغییر شکل ساختاری ندارد در حالیکه در حالت دوم موتور جهش هم باید در هر نسل و نسخه تکثیری تغییر شکل یابد [2]. این مسأله در ساختار موتور جهش و سطح فراریختی ویروس محدودیت‌هایی ایجاد می‌کند و درجه فراریختی کمتر خواهد شد. اگر یک ویروس با درجه بالایی فراریخت شود، حتی ممکن است نیاز به رمزگذاری هم نداشته باشد زیرا در هر نسل ساختار کد کاملاً متفاوتی دارد و استخراج یک الگوی مشترک تقریباً غیر ممکن است. به‌همین دلیل رمزگذاری فایده‌ای برای این نوع ساخت ویروس ندارد [4] و یک ویروس فراریخت شانس بالایی برای مخفی ماندن دارد.

۲-۲-۱- روش‌های مبهم‌سازی ویروس‌های فراریخت

برای اینکه ویروس‌ها را فراریخت کنیم، یا نسخه جهش یافته‌ای از کد ویروس تولید شود، راه‌کارهای متعدد مبهم‌سازی [12] پیشنهاد شده‌اند. مبهم‌سازی هم در بخش داده‌ای و هم در بخش جریان کنترلی صورت می‌گیرد [22]. مبهم‌سازی جریان کنترل با جابجایی دستورات و استفاده از دستورات پرش غیرشرطی صورت می‌گیرد. مبهم‌سازی جریان داده‌ای هم با روش‌هایی همچون تعویض دستورات، درج کد اضافی، جایگزینی دستورات معادل، تعویض ثبات‌ها و جایگشت زیرروال‌ها تحقق می‌یابد.

تعویض ثبات‌ها[2] یکی از راه‌کارهای مبهم‌سازی است. در این روش کل کد بلاتغییر می‌ماند و فقط ثبات‌ها عوض می‌شوند. برای نمونه دستور ADD EDX, 0088h را می‌توان به دستور ADD EAX, 0088h تبدیل کرد. به این ترتیب ثبات EDX با EAX جابجا می‌شود. روش تعویض ثبات به‌راحتی با استفاده از عبارت منظم قابل سوء استفاده[3] است [10].

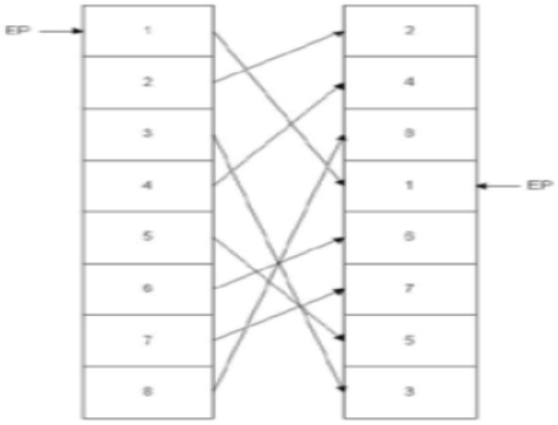
راه‌کار دیگر جانشینی دستورات[4] معادل است. در این روش یک دستور یا مجموعه‌ای از دستورات با دستور یا دستورهایی جانشین می‌شوند که همان عملکرد را داشته باشند. برای نمونه رشته دستورات PUSH R1; MOV R1, R2 را می‌توان با رشته دستورات PUSH R1; PUSH R2; POP R1; جایگزین کرد؛ عملکرد هر دو رشته دستور یکسان است [17].

گاهی اوقات برای مبهم‌سازی می‌توان ترتیب دستورات[5] را عوض کرد به‌شرطیکه هیچ وابستگی بین آن‌ها وجود نداشته باشد. این‌کار باید عملکرد دستورات را نیز عوض نکند. به این راه‌کار تعویض[6] دستورات می‌گویند؛ و ویروس‌ها را قادر می‌سازد که با روش‌های مبتنی بر امضا دیگر شناسایی نشوند زیرا ترتیب بایت‌ها دیگر با الگوی موجود مطابقت نمی‌کند [11]. جدول ۱ نمونه‌ای از این راه‌کار را نشان می‌دهد.

جدول ۱. جابجایی دستورات [4]

دستورالعمل‌های تعویض شده	دستورالعمل‌های اصلی
1: SUB R3, R4 2: SUB R1, R2	1: SUB R1, R2 2: SUB R3, R4

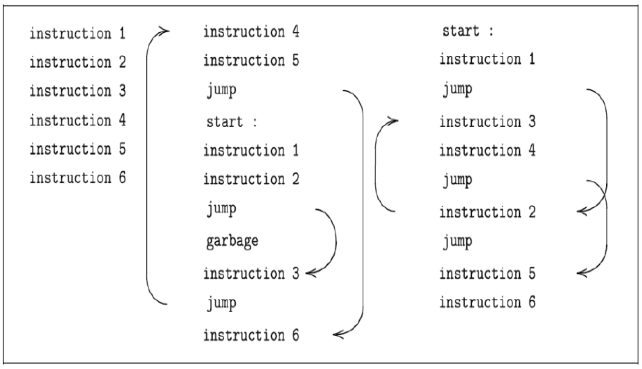
جایگشت زیرروال‌ها[7] روش دیگری است که با آن می‌توان ساختار کد ویروس را تغییر داد بدون اینکه عملکردش عوض شود. پس اگر n زیرروال داخل کد ویروس وجود داشته باشد، $n!$ گونه مختلف از ویروس می‌توان تولید کرد. برای نمونه ویروسی که ۱۰ زیرروال دارد، مثل ویروس Win32/Ghost می‌تواند ۱۰! یا ۳۶۲۸۸۰۰ گونه مختلف از خودش تولید کند [18]. اما با استفاده از عبارات منظم و الگوی رشته‌های کوتاه[8] این نوع ویروس‌ها قابل کشف هستند زیرا زیرروال‌ها تغییر نمی‌کنند و این راه‌کار به‌تنهایی مؤثر نیست [17]. نمونه‌ای از جایگشت زیرروال‌ها در شکل ۳ دیده می‌شود.



شکل ۳. جایگشت زیرروال [2]

حالت عمومی‌تر این روش جابجایی قطعات برنامه است که برنامه به چندین قطعه تقسیم می‌شود و این قطعات جابجا می‌شوند تا امضای ویروس را تغییر دهند [15]. با استفاده از دستورهای پرش مناسب در انتهای هر قطعه می‌توان ترتیب منطقی درست برنامه را حفظ کرد. در شکل ۴ نمونه‌ای از این روش نشان داده شده است. سمت چپ کدی نوشته شده است و در سمت راست دو گونه مختلف از آن با جابجایی نشان داده شده‌اند.

[1] Embed
[2] Register swap
[3] Exploit
[4] Instruction substitution
[5] Instruction reordering
[6] Transposition
[7] Subroutine permutation
[8] Short string



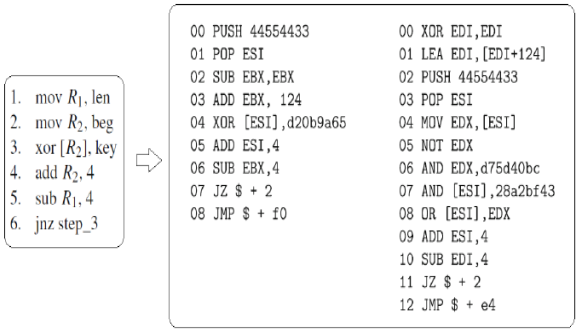
شکل ۴. جابجایی قطعات برنامه [15]

درج دستورات بیهوده، زائد[1] یا زباله روش مؤثری برای تولید ویروس‌های فرایخت محسوب می‌شود. دستورات زائد آن‌هایی هستند که در صورت اجرا، تأثیری بر عملکرد ویروس نمی‌گذارند یا اصلاً اجرا نمی‌شوند. دستورات زائد در حالت اول را کد «هیچ‌کاره[2]» و دستورات حالت دوم را «کد مرده[3]» می‌نامند. دستورات زائد امضای ویروس را تغییر می‌دهند و حتی ممکن است برای کند کردن نرم‌افزارهای مقلد یا مقابله با روش‌های شناسایی غیر مستدل هم به‌کار روند. نمونه‌ای از کد هیچ‌کاره در شکل ۵ دیده می‌شود. دستورات هیچ‌کاره به‌صورت فیزیکی بخشی از برنامه هستند و نه به‌صورت منطقی [13]. با استفاده از درج دستورات زائد، تقریباً می‌توان تعداد نامحدودی گونه‌های مختلف از یک ویروس مشخص تولید کرد. نمونه‌هایی از دستورهای هیچ‌کاره NOP و ADD EAX, 0 هستند. این‌گونه دستورها کمک شایانی به تولید ویروس‌های فرایخت می‌کنند [2]. این روش و روش قبلی استفاده بسیاری در تولید ویروس‌های فرایخت دارند. اضافه کردن کد مرده از فایل سالم داخل فایل بدافزار از روش‌های مؤثر در جلوگیری از کشف بدافزارهای فرایخت است [30].

0040102C	8BC2	MOV EAX, EDX
0040102E	90	NOP
0040102F	90	NOP
00401030	42	INC EDX
00401031	52	PUSH EDX
00401032	FE0C24	DEC BYTE PTR SS:[ESP]
00401035	4A	DEC EDX
00401036	52	PUSH EDX

شکل ۵. دستورات هیچ‌کاره (بخش وسط)

روش مطرح دیگر برای مبهم سازی، جهش گرامر رسمی[4] است. این روش در حقیقت گونه رسمی بسیاری از روش‌های تغییر شکل[5] موجود است. یک موتور تغییر شکل کلاسیک مثل یک آتوماتای غیر قطعی است زیرا گذر از هر نماد به نماد دیگر امکان‌پذیر است [19]. مجموعه نمادها همان مجموعه همه دستورهای ممکن هستند؛ به عبارت بهتر هر دستور می‌تواند بعد از هر دستور دیگر قرار گیرد. اگر روش‌های جهش را به‌صورت رسمی تعریف کنیم، از قوانین گرامرهای رسمی می‌توان استفاده کرد تا کپی‌های متنوعی از هر ویروس به‌دست آوریم، به‌طوریکه هر کپی تغییرات عمده‌ای نسبت به سایر کپی‌ها داشته باشد. شکل ۶ قالب یک رمزگشای ویروس چند ریختی را نشان می‌دهد و به‌کمک گرامر رسمی نشان داده شده در شکل ۷ دو نسخه متفاوت (جهش یافته) از آن تولید شده است. با به‌کارگیری ترکیبات مختلف گرامر رسمی شکل ۶ و اعمال روی موتور رمزگشای شکل ۷ می‌توان ۹۶۰ رمزگشای مختلف تولید کرد [19].



شکل ۶. یک رمزگشای چندریختی ساده و دو گونه مختلف آن [19]

$$
\begin{aligned}
A &\rightarrow XB \\
B &\rightarrow Y_4\varepsilon \\
X &\rightarrow X_1X_2|X_2X_1 \\
X_1 &\rightarrow GX_1|mov\ R_1, len|push\ len \oplus pop\ R_1|xor\ R_1, \\
&\quad R_1 \oplus lea\ R_1, [R_1 + len]|sub\ R_1, R_1 \oplus add\ R_1, len \\
X_2 &\rightarrow GX_2|mov\ R_2, beg|push\ beg \oplus pop\ R_2|xor\ R_2, \\
&\quad R_2 \oplus lea\ R_2, [R_2 + beg]|sub\ R_2, R_2 \oplus add\ R_2, beg \\
Y_4 &\rightarrow GY_4|W_1|S_4W_4 \\
W_1 &\rightarrow GW_1|xor\ [R_2], key\ H_1 \\
W_1 &\rightarrow not\ [R_2] \oplus xor\ [R_2], key \oplus not[R_2]\ H_1 \\
W_1 &\rightarrow mov\ R_3, [R_2] \oplus not\ R_3 \oplus and\ R_3, key \oplus and\ [R_2], \\
&\quad \neg key \oplus or\ [R_2], R_3\ H_1 \\
H_1 &\rightarrow GH_1|add\ R_2, 4\ H_2|sub\ R_2, -4\ H_2 \\
S_4 &\rightarrow GS_1|sub\ R_2, 4|add\ R_2, -4 \\
W_2 &\rightarrow GW_2|xor\ [R_1][R_2], key\ H_2 \\
W_2 &\rightarrow not\ [R_1][R_2] \oplus xor\ [R_1][R_2], key \oplus not[R_1][R_2]\ H_2 \\
W_2 &\rightarrow mov\ R_3, [R_1][R_2] \oplus not\ R_3 \oplus and\ R_3, key \oplus and \\
&\quad [R_1][R_2], \neg key \oplus or\ [R_1][R_2], R_3\ H_2 \\
H_2 &\rightarrow GH_2|sub\ R_1, 4 \oplus jnz\ xxx|sub\ R_1, 4 \oplus jz\ yyy \oplus jmp\ xxx \\
H_2 &\rightarrow add\ R_1, -4 \oplus jnz\ xxx|add\ R_1, -4 \oplus jz\ yyy \oplus jmp\ xxx \\
H_2 &\rightarrow sub\ ecx, 3 \oplus loop\ xxx \Leftrightarrow R_1 \equiv ecx
\end{aligned}
$$

شکل ۷. گرامر رسمی برای تولید نسخه جهشی رمزگشا [19]

جدول ۲ دشواری کشف برخی از روش‌های مبهم‌سازی ویروس‌های فرایختی را نشان می‌دهد. جدول ۳ فنون مبهم‌سازی که در چند ویروس فرایخت تولید شده توسط هکرها پیاده‌سازی شده‌اند را خلاصه کرده است.

[1] Garbage
[2] Do-nothing
[3] Dead-code
[4] Formal grammar mutation
[5] Morph

جدول ۲. مقایسه سطح دشواری روش‌های مبهم‌سازی [25]

روش	ساده	متوسط	سخت
جابجایی دستورات	✓		
درج کد زائد		✓	
تعویض ثبات‌ها	✓		
جانشینی دستورات			✓
تعویض دستورات			✓

جدول ۳. مقایسه روش‌های مبهم‌سازی کد در بدافزارها [23]

روش	Evol 2000	Zmist 2001	Zperm 2000	Regswap 2000	MetaPHOR 2001
جانشینی دستورات				✓	
جایگشت	✓	✓			✓
کد زائد	✓	✓			✓
جانشینی متغیر	✓	✓		✓	✓
تغییر جریان کنترل		✓	✓		✓

۲-۲-۲- نقطه ضعف ویروس‌های فرایخت

ویروس‌های فرایخت هنگام انتشار، کد خودشان را به گونه دیگری با همان عملکرد تبدیل می‌کنند تا پویشگرهای مبتنی بر امضا نتوانند آن‌ها را شناسایی نمایند و برای این‌کار از روش‌های مبهم‌سازی کد بهره می‌برند. ضمن اینکه این ویروس‌ها با تحلیلگرهای پویا مثل نماسازها هم می‌توانند مقابله کنند. وقتی که متوجه می‌شوند که در حال اجرا در یک محیط مجازی یا کنترل شده هستند، رفتار خود را تغییر می‌دهند [24]. چون سازندگان ویروس‌های فرایخت از نقطه ضعف پویشگرهای ضد ویروس‌ها با خبر هستند، کشف این نوع ویروس‌ها دشوار می‌گردد. محدودیت‌هایی که در روش‌های تحلیل ایستا و پویا وجود دارند موجب بروز محدودیت‌هایی در پویشگرهای ضد ویروس‌ها می‌گردند. نشان داده شده است [24] که پاشنه آشیل یک ویروس فرایخت، نیازش به تحلیل کد خودش است. پویشگر ضد ویروس باید بتواند ویروس فرایخت را با روشی مشابه روش به‌کار رفته توسط خود ویروس برای تحلیل خودش، تحلیل کند. در این صورت ضد ویروس نیاز به یک تغییر دهنده معکوس دارد که با اعمال قوانین تبدیل در جهت عکس، کد اصلی ویروس را بازیابی نماید. مشکل اینجاست که باید حداقل یک نمونه از ویروس در اختیار تحلیلگران باشد تا بتوانند قوانین تبدیل، مفروضات و الگوریتم‌ها را استخراج نمایند.

۲-۳- کشف ویروس‌های فرایخت

کشف ویروس‌های فرایخت دشوار است، ولی به‌همین نسبت تولید ویروس فرایخت هم برای مهاجمین دشوار است [6]. یک مشکل در تولید ویروس فرایخت این‌است که ویروس باید به اندازه کافی از نظر ساختار جهش یافته و دگردیس شود و در عین حال اندازه‌اش در حد قابل کنترلی نگه داشته شود. مشکل دیگر این است که ویروس‌های فرایخت باید تا حد امکان شبیه فایل‌های سالم باشند تا روش‌های مبتنی بر شباهت و ابتکاری قادر به کشف آن‌ها نباشند.

روش‌هایی که برای کشف ویروس‌های فرایخت پیشنهاد شده‌اند را می‌توان به دو دسته تقسیم کرد [25]: دسته اول روش‌هایی هستند که در محصولات تجاری[1] پیاده‌سازی شده‌اند و تعدادشان محدود است. دسته دوم روش‌های بسیار متعددی هستند که آزمایشگاهی[2] بوده و هنوز کاربردی نشده‌اند. روش‌های دسته اول از جهات مختلف مورد آزمایش قرار گرفته‌اند. از علت‌های تجاری شدن روش‌های دسته اول می‌توان به موفقیت آن‌ها در شناخت ویروس‌ها، کارایی زمانی و محاسباتی بالا، و نرخ پایین هشدارهای مثبت اشتباه اشاره کرد. روش‌های موجود در دسته دوم، هنوز در حد کارهای پژوهشی هستند و استفاده عمومی در نرم‌افزارهای ضد ویروس تجاری ندارند. دلایل آن یک یا چند مورد زیر است: نرخ پایین کشف موفق ویروس‌های فرایخت، نرخ بالای هشدارهای مثبت اشتباه یا غیر عملی بودن روش از نظر فضا و زمان محاسباتی.

تحقیقات [21] نشان داده است که مدل مخفی مارکوف در کشف ویروس‌های فرایخت تولید شده توسط هکرها می‌تواند مؤثر واقع شود. برای جلوگیری از کشف ویروس‌های فرایخت، نه تنها درجه فرایختی آن‌ها باید بالا باشد، بلکه باید تا حدودی شبیه فایل‌های سالم هم باشند [17]. ساخت ویروس‌هایی که هر دو راه‌کار در آن‌ها اعمال شده باشد، معمولاً دشوار است. فرایختی علاوه بر تولید بدافزار برای دفاع هم به‌کار می‌رود. مثلاً برای جلوگیری از حمله سرریز بافر، درجه کمی فرایختی، بسیار مؤثر واقع می‌شود [17].

۳- روش‌های کشف ویروس‌های فرایخت

در سال‌های گذشته روش‌های متعددی اختصاصاً برای کشف بدافزارهای فرایخت ارائه شده‌اند. در این بخش پژوهش‌های حائز اهمیت و تأثیرگذار که با استفاده از تحلیل ایستا برای کشف بدافزاهای فرایخت طراحی شده‌اند، مورد بررسی قرار می‌گیرد. هر قسمت از این بخش به یک پژوهش اختصاص دارد. در تشریح هر پژوهش، ابتدا مشکل و روش پیشنهادی ارائه می‌شود. سپس آزمایش‌های صورت گرفته برای ارزیابی روش تشریح می‌گردد. در زیر بخش بعدی نتایج حاصل از آزمایش‌ها مورد بحث قرار می‌گیرد. پژوهش‌های مورد بحث در این بخش به‌ترتیب سال ارائه تنظیم شده‌اند.

۳-۱- طبقه‌بندی تبدیلات یکتاسازی

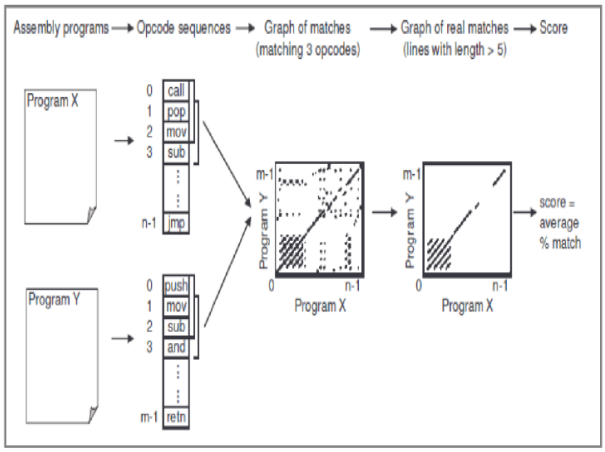
۳-۱-۱- مشکل و روش پیشنهادی

مسأله این است که برای حفاظت از حقوق معنوی نرم‌افزار و مقابله با برخی از روش‌های حمله به کد، باید نمونه‌هایی از یک نرم‌افزار تولید کرد که همگی عملکرد یکسان داشته باشند و در عین حال از نظر ساختار کد تفاوت‌هایی داشته باشند. برای این منظور می‌توان از روش‌های تبدیل کد بهره برد. اگر نمونه‌های گوناگون یک نرم‌افزار به اندازه کافی با هم متفاوت باشند، انتظار داریم که حجم کار لازم برای مهندسی معکوس N نمونه به نسبت خطی افزایش یابد. پس با افزایش تعداد نمونه‌های منحصر به‌فرد یک نرم‌افزار، پیچیدگی حمله به نرم‌افزار و به‌خطر انداختن امنیت آن هم بالا می‌رود.

[1] Commercial
[2] Experimental

روش پیشنهادی [32] با به‌کارگیری ترکیب تصادفی از فنون تبدیل کد، کاری می‌کند که یکتا سازی حتی الامکان گونه‌های متنوع و متغیری داشته باشد که مهندسی معکوس هر کد بسیار دشوار شود. تبدیلات روی کد منبع صورت می‌گیرند. زبان مورد استفاده در روش پیشنهادی کدهای C است، زیرا ذاتاً زبانی روالی است. با این وجود، این انتخاب محدود کننده نیست و تبدیلات روی هر کدی قابل استفاده هستند. اما مطلوب است که جهت رعایت ملاحظات امنیتی، تبدیلات روی کد اسمبلی قابل استفاده باشند.

برای اندازه‌گیری شباهت کدهای اسمبلی و شناخت سطح تغییر کدهای حاصل از اعمال روش پیشنهادی، نویسندگان روشی را هم برای محاسبه امتیاز شباهت ارائه کرده‌اند. هم‌ارزی برنامه‌ها یک مسأله NP سخت است. اما نویسندگان ادعا کرده‌اند که هر یک از تبدیل‌های پیشنهادی آن‌ها روی کد باعث می‌شود خروجی برنامه تغییر نکند. پس فقط کافی است که وارسی کنند خروجی‌ها یکسان می‌ماند. اعمال تبدیلات کد پیشنهادی، فقط ساختار کد را عوض می‌کند ولی منطق بلا تغییر می‌ماند. پس مسأله هم‌ارزی کد اینجا مطرح نمی‌شود. روش پیشنهادی بر مبنای تعداد کدهای متوالی یکسان که در کد اسمبلی دو برنامه مطابقت می‌کنند، صورت می‌گیرد. شکل ۸ روش کار روش پیشنهادی را نشان می‌دهد.



شکل ۸. شباهت بر اساس تحلیل *n*-gram [2]

۳-۱-۲- آزمایش‌ها

نویسندگان ابزاری به زبان C توسعه داده‌اند که شباهت را روی کدهای حاصل از تبدیل‌ها با روش پیشنهادی ارزیابی نمایند. ورودی ابزار، کد اسمبلی معادل با کد C برنامه‌های اصلی و تبدیل شده هستند. ابتدا کدهای C به کد ماشین ترجمه شده و سپس به اسمبلی برگردانده شده‌اند. کامپایلر C کد منبع را ترجمه کرده است و با دستور objdump سیستم عامل لینوکس، کدها به اسمبلی برگردانده شده‌اند. آزمایش‌های صورت گرفته، یک فایل اصلی را با خودش و با فایل‌های تبدیل شده دیگر مقایسه کرده است. همچنین فایل‌های تبدیل شده نیز با هم مقایسه شده‌اند.

۳-۱-۳- نتایج آزمایش‌ها

مقایسه فایل اصلی با خودش، امتیاز ۱۰۰، مقایسه فایل اصلی با گونه‌های تبدیل شده به‌ترتیب امتیازهای ۳۸، ۳۶ و ۵۷ و مقایسه فایل‌های تبدیل شده با هم به‌ترتیب درصدهای ۲۰، ۲۷ و ۳۱ را تولید کرده‌اند.نتایج آزمایش‌ها نشان داده‌اند که تبدیلات ساده اعمال شده، نمونه‌هایی تولید می‌کنند که با فایل اصلی بسیار متفاوتند. این تفاوت کار لازم توسط مهاجمین برای مهندسی معکوس را افزایش می‌دهد.

مقاله‌های [17] و [20] از روش *n*-gram برای مقایسه ویروس‌های خانواده NGVCK استفاده کرده‌اند و مقایسه‌ها روی کدهای برگردانده شده به اسمبلی صورت گرفته‌اند.

۳-۲- اعمال ترتیب روی دستورات برنامه در جهت کمک به پویشگر ضد ویروس‌ها

۳-۲-۱- مشکل و روش پیشنهادی

مسأله این‌است که ویروس‌های فرایخت با کمک روش‌های مبهم‌سازی می‌توانند خودشان را به انبوهی از گونه‌های مختلف تبدیل کنند. هر گونه جدید ویروس فرایخت امضای متفاوتی دارد و ضد ویروس‌های مبتنی بر امضا قادر به شناسایی گونه‌های جدید نیستند. برای این منظور در این مقاله تلاش شده که برنامه ویروس به شکل متعارفی تبدیل شود، به‌طوریکه دو گونه مختلف ویروس شکل متعارف یکسانی داشته باشند. در این‌صورت می‌توان امضای ویروس‌های فرایخت را از شکل متعارف آن‌ها استخراج کرد و حین پویش، هر فایل مشکوک پس از تبدیل به شکل متعارف مورد تحلیل مبتنی بر امضا قرار گیرد. برای رسیدن به شکل متعارف باید اثر تبدیلات را معکوس کرد. اما ترتیب به‌کارگیری تبدیلات معکوس برای رسیدن به یک شکل متعارف واحد دقیقاً معلوم نیست و هنوز یک مسأله تحقیقاتی باز است.

در این مقاله [33] مجموعه‌ای از روش‌های ابتکاری ارائه شده‌اند که روی دستورات برنامه‌های شبه C ترتیب‌هایی اعمال کند. با اعمال این ترتیب، اثر تبدیلات جابجایی دستورات صورت گرفته روی ویروس فرایخت خنثی خواهد شد. به این روش تبدیلات صفرسازی[1] اطلاق می‌شود. پس ضد ویروس‌ها باید امضاها را از شکل صفر ویروس‌ها استخراج نمایند.

۳-۲-۲- آزمایش‌ها

برای ارزیابی روش پیشنهادی، ابزار نمونه‌ای به اسم $C\oplus$ با استفاده از ابزار Code Surfer شرکت GrammaTech تولید شد. آزمایش روش پیشنهادی روی مجموعه‌ای از برنامه‌های واقعی C صورت گرفت؛ این برنامه‌ها عبارتند از: Bison, Cook, Computer Vision, SNNS.

۳-۲-۳- نتایج آزمایش‌ها

نتایج آزمایش‌ها نشان دادند که حدود ۵۵ درصد دستورات یک برنامه قابل جابجایی هستند. بعد از اعمال ترتیب روی دستورات با تبدیل صفرسازی، تنها شش درصد از دستورات باقی‌مانده قابل جابجایی بودند. همچنین، بعد از اعمال ترتیب روی دستورات برنامه‌های آزمایشی با روش پیشنهادی، تعداد جایشگت‌های ممکن برنامه‌های حاصل از 10^{183} به 10^{20} کاهش پیدا کرد، یعنی فضای گونه‌های برنامه به نسبت 10^{163} کم شد. در حوزه کشف ویروس‌های فرایخت، این نتیجه باعث کاهش چشمگیر امضای همه گونه‌های یک ویروس فرایخت می‌گردد.

[1] Zeroing transformations

۳-۳- کشف ویروس‌های فراریخت کامپیوتری مبتنی بر تفکیک‌پذیری با استفاده از راه‌کار کنترل افزونگی

۳-۳-۱- مشکل و روش پیشنهادی

کشف ویروس‌های فراریخت به‌دلیل اعمال فنون متنوع مبهم‌سازی هنوز دشوار است. یکی از دلایل دشواری به مسأله پنهان‌سازی نقطه ورودی به کد ویروس بر می‌گردد. این طرز مبهم‌سازی را مبهم‌سازی نقطه ورودی[1] می‌نامند. برای کشف ویروس‌های فراریخت، روش تقلید نمادین مبتنی بر تفکیک‌پذیری معرفی شده است. روش پیشنهادی [34]، کاربردی از درستی‌یابی رسمی است که با کمک روش‌های اثبات قضیه صورت می‌گیرد.

۳-۳-۲- آزمایش‌ها

آزمایش‌هایی روی ویروس‌های W32.evol، W32.simile و W32.Zmist صورت گرفته‌اند.

۳-۳-۳- نتایج آزمایش‌ها

آزمایش‌ها نشان داده‌اند که بدون راه‌کارهای پیشنهادی، تبدیل کد فراریخت به چندین عمل ساده تقریباً غیرممکن است یا حداقل از نظر محاسباتی غیر عملی است.

۳-۴- کشف بدافزار خود جهشی با استفاده از تطابق گراف جریان کنترل

۳-۴-۱- مشکل و روش پیشنهادی

برای تولید ویروس فراریخت، روش‌های متعددی جهت مبهم‌سازی و جهش کد معرفی شده‌اند که از جمله آن‌ها می‌توان به مبهم‌سازی نقطه ورود[2] اشاره کرد. این راه‌کاری است که در ویروس فراریخت Zmist مورد استفاده قرار گرفته است. این ویروس خودش را در یک فایل اجرایی سالم درج می‌کند و بدنه کدش را در بین دستورات آن پخش می‌کند. قطعات کد ویروس با دستورات پرش به‌هم وصل می‌شوند. این حقه همان مبهم‌سازی نقطه ورود است. برای حل مشکلات مبهم‌سازی، روش پیشنهادی در این مقاله [35] این‌طور کار می‌کند: برای برنامه ورودی P ابتدا کدش به اسمبلی برگردانده می‌شوند و برنامه P' به‌دست می‌آید. سپس برنامه نرمال‌سازی می‌شود تا برنامه P_N به‌دست آید. نرمال‌سازی اثر اکثر فنون جهش را کم کرده یا از بین می‌برد تا ارتباط جریانی بین کد بدخواه و سالم آشکار گردد. در حقیقت نرمال‌سازی فرایند جهش را معکوس می‌نماید. برای برنامه P_N گراف جریان کنترلی بین روالی و برچسب‌گذاری شده یا CFG_{PN} ساخته می‌شود. سپس گراف جریان کنترلی آن با گراف جریان کنترلی یک بدافزار شناخته شده مقایسه می‌شود. هدف این‌است که ببینیم آیا زیر گرافی در CFG_{PN} وجود دارد که هم‌ریخت[3] گراف جریان کنترلی بدفزار تحت بررسی باشد. بنابراین مسأله کشف بدافزار به مسأله هم‌ریختی گراف تبدیل می‌شود[4]. با این راه‌کار می‌توان در مقابل بسیاری از فنون جهش به‌کار رفته در ساخت بدافزارهای چندریخت مقابله کرد.

۳-۴-۲- آزمایش‌ها

میزان سودمندی بخش نرمال‌سازی کد در مقاله دیگری با ۱۱۵ ویروس مورد ارزیابی قرار گرفته است. در این مقاله هم به همان نتایج استناد شده است. جهت ارزیابی درستی روش پیشنهادی، چندین آزمایش روی مجموعه بزرگی از فایل‌های اجرایی سیستم صورت گرفته است. نتایج آزمایش‌ها امیدبخش بوده‌اند.

۳-۵- کشف ویروس‌های کامپیوتری فراریخت با استفاده از مشخصات جبری

۳-۵-۱- مشکل و روش پیشنهادی

در این مقاله [36] روش جدیدی برای کشف ویروس‌های فراریخت ارائه شده است. روش پیشنهادی ابتدا معنای یک زبان برنامه‌سازی را با کمک OBJ (یک نشانه گذاری رسمی برای اثبات قضیه و توصیف جبری) به‌طور رسمی مشخص می‌کند. در این مقاله زبان اسمبلی IA-32 به‌طور رسمی توصیف شده است. از این توصیف جبری برای اثبات هم‌ارزی[5] (معادل) یا نیمه هم‌ارزی[6] رشته دستورالعمل‌های زبان IA-32 استفاده می‌شود. برای اثبات هم‌ارزی یا نیمه ارزی، روش‌هایی موجود است. از این راه‌کار در جهت کشف ویروس‌های فراریخت می‌توان بهره برد.

پس از توصیف جبری دستورات زبان IA-32 با نشانه گذاری رسمی، می‌توان یک مفسر و ابزار تحلیل برنامه برای آن زبان به‌دست آورد. به این ترتیب یک قطعه کد مشکوک با توصیف OBJ زبان IA-32 قابل تفسیر است تا رفتار آن کد مورد تحلیل قرار گیرد. در نتیجه توصیف زبان IA-32 با OBJ امکان نماسازی کد بر اساس تحلیل پویا را فراهم می‌سازد. پس می‌توان از این روش در جهت تحلیل بدافزار مشکوکی که سرریز بافر دارد، بهره برد.

۳-۵-۲- آزمایش‌ها

برای نمونه اثبات هم‌ارزی و نیمه هم‌ارزی صور مختلف دو ویروس Win95/Bistro و Win9x.Zmorph.A از نظر پشته نشان داده شده‌اند. به این معنی که حالت پشته در هر دو نسل یک ویروس به‌طور یکسان تغییر می‌کند.

۳-۵-۳- نتایج آزمایش‌ها

از مطالعات منطقی است نتیجه گرفته شود که می‌توان ویروس‌های فراریختی را که کدشان از نظر گرامری متفاوت ولی از نظر معنایی یکسان باشد، تشخیص داد. برای این منظور کافیست هم‌ارزی معنایی آن‌ها را اثبات کرد.

---

[1] Entry Point Obfuscation (EPO)
[2] Entry point obfuscation
[3] Isomorphism

[4] مسأله هم‌ریختی گراف‌ها NP کامل است ولی در عمل گاهی به‌طور کارامد حل می‌شود؛ وقتی گراف مزبور خلوت باشد.

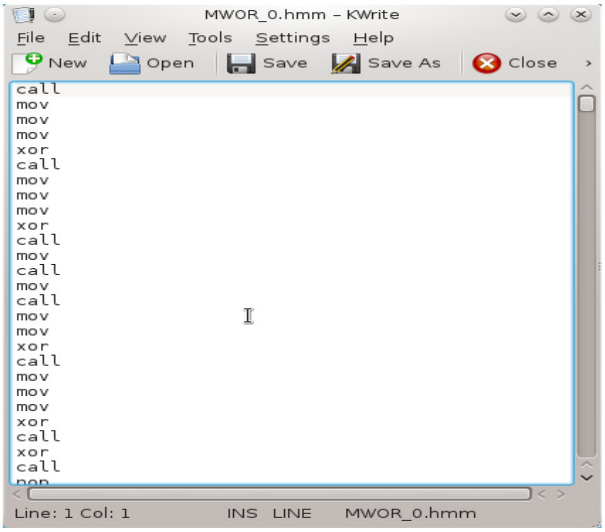
[5] Equivalence
[6] Semi-equivalence

۳-۶- شکار موتورهای مولد فراریختی

۳-۶-۱- مشکل و روش پیشنهادی

مشکل این است که معلوم نیست موتورهای فراریختی چقدر مؤثر هستند؟ گونه‌های فراریختی تولید شده چقدر با هم تفاوت دارند؟ و آیا می‌توان ویروس‌های تولیدی را کشف نمود؟ برای پاسخ به این سؤال‌ها، نویسندگان مولدهای مختلف فراریختی را مورد تحلیل قرار داده‌اند. برای این منظور روش شاخص شباهت را تعریف کرده و از آن برای سنجش درجه فراریختی هر مولد بهره برده‌اند. سپس با استفاده از مدل مخفی مارکوف، روشی برای کشف ویروس‌های فراریخت معرفی کرده‌اند. همچنین روش ساده‌ای برای کشف هم با شاخص شباهت معرفی شده تشریح شده است.

روش پیشنهادی نویسندگان [17] ابتدا رشته کدهای همه فایل‌های ویروس‌های یک خانواده را استخراج می‌کند. برای این‌کار فایل ویروس‌ها به اسمبلی برگردانده می‌شوند. از هر فایل اسمبلی برگردانده شده، فقط کدهای عملیاتی هر دستور نگه داشته می‌شود و عملوندها کنار گذاشته می‌شوند. نمونه‌ای از کدهای استخراج شده در شکل ۹ نشان داده شده است. در مرحله بعد، رشته کدهای مستخرج از همه فایل‌ها با هم الحاق می‌شوند و یک رشته کد بسیار بزرگ به‌دست می‌آید که به آن رشته مشاهدات[1] گفته می‌شود.



شکل ۹. رشته کدهای استخراج شده از یک فایل [2]

سپس یک مدل مخفی مارکوف با رشته کد حاصل آموزش داده می‌شود. مدل مخفی مارکوفی که با کدهای یک خانواده ویروس آموزش داده می‌شود، خصوصیت‌های آماری آن خانواده را نشان می‌دهد. وقتی فایل مشکوک جدیدی داده می‌شود، با کمک مدل، امتیاز LLPO برای آن محاسبه می‌شود که این امتیاز نشان می‌دهد فایل مشکوک چقدر به ویروس‌های آن خانواده نزدیک است. بر اساس معیار LLPO می‌توان سطح آستانه‌ای هم برای فایل‌های ویروس و سالم تعیین کرد. سپس یک فایل باینری جدید بر اساس امتیاز LLPO به‌عنوان فایل سالم یا ویروس طبقه‌بندی خواهد شد.

۳-۶-۲- آزمایش‌ها

آزمایش‌ها روی چهار خانواده از ویروس‌ها صورت گرفت که عبارتند از:

۱. ۱۰ فایل EXE از خانواده NGVCK
۲. ۱۰ فایل COM از خانواده G2
۳. ۱۰ فایل EXE از خانواده VCL32
۴. هفت فایل COM از خانواده MPCGEN

شباهت بین گونه‌های این چهار خانواده ویروس با روش پیشنهادی اندازه‌گیری شد. همه فایل‌های اجرایی باینری به‌کمک ابزار IDA Pro [37] به اسمبلی برگردان شدند.

۳-۶-۳- نتایج آزمایش‌ها

نتایج نشان دادند که هر یک از مولدها کارایی‌های متفاوتی دارند. بهترین مولد NGVCK است که ویروس‌هایی با بالاترین درجه فراریختی تولید می‌کنند. ویروس‌های سایر مولدها حدود ۶۰٪ با هم شباهت دارند. این درحالی است که فایل‌های سالمی که به‌طور تصادفی انتخاب شده بودند، حدود ۳۵٪ شباهت داشتند. پس معلوم شد که سه ابزار مولد فراریختی عملکرد مؤثری در تولید ویروس‌های فراریختی خوب ندارند. ویروس‌های NGVCK درجه بالای فراریختی دارند و ساختار اسمبلی آن‌ها با فایل‌های نرمال تفاوت چشمگیری دارد. هر دو روش پیشنهادی قادر بودند همه ویروس‌های NGVCK را با دقت بالا کشف کنند، هر چند که پویشگر ضد ویروس‌های تجاری موجود قادر به کشف ویروس‌های فراریخت مورد استفاده در آزمایش‌های نویسندگان نبودند.

۳-۷- استفاده از امضای موتور برای کشف بدافزارهای فراریخت

۳-۷-۱- مشکل و روش پیشنهادی

بدافزار فراریخت در هر دگردیسی امضای خود را تغییر می‌دهد و کشف آن توسط ضد ویروس‌های مبتنی بر امضا دشوار می‌گردد. در این مقاله [62] از روش امتیازدهی به موتور فراریخت استفاده می‌شود تا امضای موتورها به‌دست آید. این روش به‌دنبال قطعه کدی می‌گردد تا تعیین کند چقدر احتمال دارد که این کد بخشی از برنامه تولید شده توسط یک موتور فراریخت باشد که با روش جانشینی دستور کار می‌کند. با این روش بجای ذخیره اطلاعات گونه‌های مختلف ویروس فراریخت، فقط کافیست اطلاعاتی راجع به موتور مولد آن‌ها ذخیره نماییم.

۳-۷-۲- آزمایش‌ها

برای ارزیابی روش پیشنهادی، شبیه‌سازی نوشته شد که رفتار موتورهای فراریخت جانشین را که از جانشینی دستور بهره می‌برند، تقلید نماید. از این شبیه‌ساز در آزمایش‌هایی برای ارزیابی کارایی روش امتیازدهی به موتور فراریخت استفاده شده است. سپس از روش پیشنهادی برای ردیابی گونه‌های ویروس W32.Evol به موتور مولد آن استفاده شد. دو نوع ارزیابی توسط نویسندگان صورت گرفته است: (۱) شناسایی خروجی یک موتور فراریختی خاص و (۲) شناسایی گونه‌های مختلف یک بدافزار خاص توسط موتور.

۳-۷-۳- نتایج آزمایش‌ها

نتایج آزمایش‌های نشان داد که وقتی سطح موتور پسندی گونه اصلی ویروس بیش از ۵۰ درصد باشد، روش پیشنهادی عملکرد مناسبی روی گونه‌های

[1] Observations

ویروس داشته است. در ویروس‌هایی که سطح موتورپسندی کمی داشتند، روش پیشنهادی فقط توانست گونه‌های بعد از خروجی موتور فراریخت را شناسایی نماید.

۳-۸- کشف ویروس‌های فراریخت با استفاده از مدل مخفی مارکوف شکلی[1]

۳-۸-۱- مشکل و روش پیشنهادی

در این پروژه [40] از مدل‌های مخفی مارکوف شکلی برای مدل‌سازی خانواده ویروس‌های فراریخت استفاده شده است و با استفاده از مدل فایل‌های ویروس و غیر ویروس امتیازدهی می‌شوند. مدل مخفی مارکوف شکلی در بیوانفوماتیک برای یافتن رشته‌های مرتبط به‌هم در یک خانواده از رشته‌های پروتئین مورد استفاده قرار می‌گیرد. مدل PHMM با ترازبندی کدهای عملیاتی گونه‌های یک خانواده ویروس ساخته می‌شود و در آن گروهی از احتمالات نگهداری می‌شود. برای تفکیک فایل‌های ویروس از غیر ویروس، نزدیکی آن‌ها به مدل با الگوریتم Forward اندازه‌گیری می‌شود.

۳-۸-۲- آزمایش‌ها

هدف بررسی کارایی PHMM روی سه ابزار مختلف مولد فراریختی است که در برهه‌های زمانی مختلف ساخته شده‌اند تا پیشرفت تولید ویروس‌ها هم معلوم شود. به‌عبارت بهتر آزمایش‌ها قابلیت کشف ویروس‌های تولید شده توسط ابزارهای گوناگون را توسط PHMM مورد بررسی قرار می‌دهند. از الگوریتم Forward برای امتیازدهی فایل‌های ASM در برابر مدل PHMM استفاده شده است. داده‌های به‌کار رفته در آزمایش‌ها به‌شرح زیر است:

۱. ۱۰ گونه ویروس از ابزار VCL
۲. ۳۰ گونه ویروس از PS-MPC
۳. ۲۰۰ گونه مختلف از NGVCK
۴. ۴۰ فایل cygwin dll 1.5.19 برگردانده شده به زبان اسمبلی
۵. ۳۰ فایل dll غیرویروس دیگر از MSOffice, Adobe, IE و غیره

فایل‌های غیرویروس پیش از امتیازدهی فیلتر شدند تا فقط در آن‌ها کدهای عملیاتی وجود داشته باشد. سه ابزار VCL, PS-MPC و NGVCK (که به‌ترتیب از فنون ساده تا پیشرفته تغییر شکل استفاده می‌کنند) بسته به نوع تنظیماتشان فایل‌های اسمبلی تولید می‌کنند. برای برگرداندن به اسمبلی هم از ابزار IDA Pro Disassembler استفاده شده است. در ترازبندی، از کاراکتر جانشین * به‌جای هر کد عملیاتی که در بین ۳۶ کد عملیاتی برتر قرار ندارد، گذاشته شد. مدل PHMM از روی مشاهدات MSA به ازاء هر خانواده ویروس ساخته می‌شود. حتی برای گونه‌های مختلف هر خانواده ویروس می‌تواند چند مدل ساخته شود تا انعطاف‌پذیری بالاتر رود.

۳-۸-۳- نتایج آزمایش‌ها

نتایج نشان دادند که PHMM در مدل‌سازی ویروس‌ها موفق عمل می‌کند. مدل ساخته شده ویروس‌های VCL و PS-MPC را با نرخ ۱۰۰٪ و بدون هشدار مثبت یا منفی اشتباه توانست کشف کند. بعد از بازآرایی زیرروال‌ها و تنظیم آستانه، مدل ساخته شده توانست ویروس‌های NGVCK را با نرخ هشدار مثبت اشتباه ۱۹/۴۳٪ و نرخ هشدار منفی اشتباه ۱٪ کشف نماید.

۳-۹- شکار ویروس‌های فراریخت غیرقابل کشف

۳-۹-۱- مشکل و روش پیشنهادی

هدف طراحی موتور فراریختی است که حتی روش مبتنی بر مدل مخفی مارکوف [38] هم نتواند آن‌را تشخیص دهد. پیش از این تحقیق، موتور فراریختی [41] طراحی شده بود که با وجود به‌کارگیری اغلب فنون مبهم‌سازی، باز هم روش مبتنی بر مدل مخفی مارکوف [38] به‌خاطر کارایی بالا در تشخیص ویروس‌هایی که درجه فراریختی آن‌ها بالاست، ویروس‌های تولیدی را تشخیص می‌داد. در این مقاله، موتور فراریختی طراحی شده است که دیگر روش HMM یاد شده قادر نباشد ویروس‌های تولیدی آن را تشخیص دهد.

برای تولید موتور فراریختی جدید، نویسندگان از همان فنون استفاده شده در [41] استفاده کرده‌اند. اما بعد از بررسی موتور قبلی، معلوم شد که مشکل موتور قبلی از فنون مبهم‌سازی به‌طور تصادفی استفاده می‌کند. این مسأله باعث می‌شود که ساختار ویروس شباهت زیادی با یک برنامه غیر ویروسی پیدا نکند. برای این منظور الگوریتمی به اسم «الگوریتم امتیازدهی پویا[2]» توسعه داده شده که بعد از هر عمل تغییر شکل ویروس، ویروس تولیدی را با یک برنامه «نرمال» مقایسه می‌نماید. نتیجه عمل تغییر شکل درصورتی روی ویروس اعمال می‌شود که آن‌را شبیه یک فایل نرمال نماید (که به‌عنوان ویروس فراریخت شناسایی نشود).

۳-۹-۲- آزمایش‌ها

آزمایش‌های بررسی شباهت مشابه [38] ترتیب یافته‌اند. بعد از بررسی موفقیت موتور تولید شده در تولید ویروس‌های غیرقابل تشخیص توسط HMM، آزمایش‌ها با پیکربندی‌های مختلف موتور تکرار شده‌اند. برای این منظور تعداد کد مرده کپی شده از فایل نرمال کاهش داده شد تا آستانه شکست و موفقیت روش HMM روی ویروس‌های تولیدی معلوم شود.

با ابزار NGVCK تعداد ۲۰۰ ویروس هم‌خانواده تولید شد. ویروس‌های تولیدی در نقش ویروس‌های اصلی مورد استفاده قرار می‌گیرند. سپس ۴۰ فایل نرمال هم از مجموعه فایل‌های cygwin ساخته شد. ویروس‌های تولیدی با روش HMM آزمایش شد که از کشف آن‌ها اطمینان حاصل شود. برای این‌منظور مدل HMM با ۱۶۰ ویروس ساخته شد و امتیاز برای ۴۰ ویروس باقی‌مانده و ۴۰ فایل نرمال با این مدل محاسبه شد. اگر امتیاز هر فایل از سطح آستانه بیشتر باشد، این فایل ویروس است و اگر کمتر باشد، فایل نرمال است. اگر امتیاز بعضی از فایل‌های نرمال از بعضی ویروس‌ها بیشتر باشد، آن‌گاه مدل HMM آستانه درستی برای تعیین نوع فایل ورودی ندارد و ویروس‌ها با روش HMM قابل شناسایی نخواهند بود.

۳-۹-۳- نتایج آزمایش‌ها

وقتی فایل ویروس‌ها شباهت بیشتری با فایل‌های نرمال پیدا کردند، کشف آن‌ها توسط روی HMM دشوارتر شد. اختصاصاً هنگامی‌که پنج درصد از زیرروال‌های فایل‌های نرمال داخل ویروس‌ها کپی شدند، روش مبتنی بر

[1] Profile Hidden Markov Model (PHMM)

[2] Dynamic Scoring Algorithm

HMM دیگر موفق عمل نکرد. وقتی ۳۰ درصد بلوک‌های کد مرده و ۳۰ درصد زیرروال‌ها کپی شدند، امتیاز اکثر ویروس‌ها و فایل‌های نرمال شبیه هم شدند.

۳-۱۰- دسته‌بندی گونه‌های ویروس فراریخت با استفاده از هیستوگرام فراوانی کدهای عملیاتی

۳-۱۰-۱- مشکل و روش پیشنهادی

روش پیشنهادی [42] از رویکرد آماری برای کشف ویروس‌های فراریخت بهره می‌برد. برای این منظور از هیستوگرام[1] فراوانی دستورها به‌عنوان خصوصیت آماری ویروس‌ها استفاده می‌کند تا بتواند تعیین کند آیا یک فایل مفروض نمونه دگردیس شده‌ای از یک ویروس است یا خیر. اساس کار روش بر این باور استوار است که با اعمال هر گونه تغییر، هنوز برخی خصوصیت‌های مشترک بین فایل ویروس‌ها باقی می‌ماند. به‌عبارت دیگر روش‌های مبهم‌سازی نمی‌توانند به طور کلی شباهت آماری دو فایل با عملکرد مشابه را از بین ببرند.

روش پیشنهادی فایل‌ها را بر اساس زیر روال‌های آن‌ها تجزیه می‌کند و هیستوگرام به ازاء هر بلوک کد یا زیر روال ساخته می‌شود. هدف این است که مقایسه دقیق‌تر صورت گیرد. سپس فاصله بین هیستوگرام‌ها با یکی از فنون داده‌کاوی مثل فاصله اقلیدوسی محاسبه می‌شود تا عدم شباهت دو بلوک به‌دست آید. هیستوگرام با بردار نمایش داده می‌شود که طول آن برابر با تعداد کل دستورهای ماشین است. عدم شباهت دو برنامه با محاسبه عدم شباهت بلوک‌های آن‌ها تعیین می‌شود.اگر مقدار عدم شباهت بین دو برنامه از سطح آستانه‌ای کمتر باشد، نتیجه گرفته می‌شود که این دو فایل گونه‌های فراریخت هم هستند. انتخاب سطح آستانه باید حسب مورد و با دقت صورت گیرد.

برای مقایسه هیستوگرام‌ها، هر یک از هیستوگرام‌های فایل اول با همه هیستوگرام‌های فایل دوم مقایسه می‌شود. هر جفت هیستوگرامی که کمترین فاصله را داشته باشند، مشابه‌ترین هیستوگرام‌ها در نظر گرفته می‌شوند و می‌توان نتیجه گرفت زیر روال‌های متناظرشان گونه‌های فراریخت هم هستند. سپس کمترین فاصله برای هر زیر روال ذخیره می‌شود. در انتها برداری با طول m داریم که در آن m کمترین فاصله برای هر یک از m زیرروال نگهداری می‌شود. متوسط این بردار به‌عنوان فاصله فایل اول از دوم در نظر گرفته می‌شود. توجه کنید که این فاصله متقارن نیست و فاصله فایل اول تا دوم با فاصله فایل دوم تا اول برابر نیست. لذا فاصله دو برنامه P_1 و P_2 با فرمول (۱) محاسبه می‌شود تا مقدار دقیق‌تر به‌دست آید:

$$d\{P_1,P_2\}=\frac{d(P_1,P_2)+d(P_2,P_1)}{2} \qquad (۱)$$

چون برنامه‌های گوناگون با هم مقایسه می‌شوند، پیش از مقایسه مقادیر هیستوگرام‌ها طبق فرمول (۲) نرمال‌سازی می‌شوند تا مقادیر عددی با هم قابل مقایسه باشند؛ که X برداری است که از n مؤلفه x_1 تا x_n تشکیل شده است:

$$X=\frac{X}{\sum_{i=1}^{m} x_i} \qquad (۲)$$

[1] نموداری است که طرز انتشار داده‌ها و فواصل آن‌ها را نشان می‌دهد.

۳-۱۰-۲- آزمایش‌ها

برای انجام آزمایش‌ها از هشت فایل سالم و در مجموع از ۱۱ گونه فراریخت از دو ویروس استفاده شده است.

۳-۱۰-۳- نتایج آزمایش‌ها

نتایج نشان دادند که برای هشت گونه از مجموع ۱۱ گونه‌ی دو ویروس، روش پیشنهادی با کمک آستانه ۰/۰۵۷ توانست به‌درستی ویروس‌ها را شناسایی کند. اما در شناسایی سه گونه از یک ویروس که در آن‌ها کد NOP به‌عنوان کد زائد درج شده بود، موفق نبوده است.

۳-۱۱- کشف ویروس‌های فراریخت غیرقابل کشف

۳-۱۱-۱- مشکل و روش پیشنهادی

در مقاله [44] ادعا شده است که کشف مطمئن و قاطع ویروس‌های فراریختی با روش‌های ایستا، که با فنون خاصی تولید شوند، یک مسأله NP کامل است. در مقاله حاضر یک مولد ویروس‌های فراریخت پیاده‌سازی شده است که ویروس‌های تولیدی آن بر اساس روش‌ها و شرایط مبهم‌سازی ارائه شده در [44] تولید می‌شوند. نویسندگان ویروس‌های تولیدی با این ابزار را مورد آزمایش قرار داده‌اند. ویروس‌های فراریخت تولید شده توسط نرم‌افزارهای ضد ویروس مبتنی بر امضای موجود کشف نشدند. اما با استفاده از روش یادگیری ماشینی مثل مدل مخفی مارکوف که در مقاله [17] ارائه شده بود، ویروس‌های فراریخت تحت آزمایش به‌راحتی کشف شدند. روش پیشنهادی [17] یک روش تحلیل ایستا محسوب می‌شود زیرا با تکیه بر اطلاعات استخراج شده از رشته کدهای عملیاتی ویروس کار می‌کند و کد ویروس اجرا یا نماسازی نمی‌شود. در [17] همچنین نشان داده شده که ویروس‌های فراریخت تولیدی توسط هکران چندان مؤثر نیستند. از بین موارد آزمایش شده در این مقاله [43]، حتی ویروس‌هایی که درجه فراریختی بالایی داشتند، به‌راحتی توسط روش‌های یادگیری ماشینی مثل مدل مخفی مارکوف کشف شدند.

۳-۱۱-۲- آزمایش‌ها

برای تولید ویروس اولیه[2] از ابزار NGVCK استفاده شده است. از روی این ویروس اولیه، ۲۰۰ ویروس فراریخت با کمک موتور فراریخت تولیدی در این مقاله ساخته شده است. این ویروس‌ها برای آموزش مدل مخفی مارکوف [17] مورد استفاده قرار گرفته‌اند از روش 5-fold cross validation (پنج دسته ۴۰ تایی که هر بار چهار تا برای آموزش و یکی برای آزمایش) استفاده شده است. در هر مورد، ۴۰ ویروس فراریخت با ۴۰ فایل نرمال گرفته شده از فایل‌های کاربردی Cygwin امتیازدهی شده‌اند. نرم‌افزارهای تجاری ضد ویروس مورد استفاده McAfee 2009 و Avast Home Edition 4.8 بوده‌اند.

۳-۱۱-۳- نتایج آزمایش‌ها

درهر مورد، ویروس اولیه توسط پویشگرهای ضد ویروس رایج شناسایی شد ولی ویروس‌های فراریخت ساخته شده توسط موتور تولیدی، شناخته نشدند. یعنی مولد فراریخت تولید شده به راحتی توانست کشف مبتنی بر امضا را دور

[2] Seed

بزند. اما موتور HMM تولید شده توانست ویروس‌های تغییر شکل یافته را از فایل‌های نرمال تمییز دهد.

کشف ویروس‌هایی که با روش‌های [44] تولید می‌شوند، حتی با روش‌های تحلیل ایستا دشوار نیست. فقط باید دقت کرد که تعریف رسمی ارائه شده برای تحلیل ایستا در [44] خیلی محدود است، حتی محدودتر از تعریف غیر رسمی تحلیل ایستا، که در خود مقاله صورت گرفته است.

نشان داده شده است [44] که از لحاظ نظری تولید ویروس‌های فرایخت و پیچیدگی‌های آن مهارنشدنی است، اما باید قبول کرد که در عمل کشف ویروس‌های فراریخت آن‌چنان پیچیده نیست. تعداد ویروس‌های فراریختی با درجه بالای فراریختی و کشف دشوار زیاد نیست [44].

۳-۱۲- ویروس‌های ویژه برای تشخیص ویروس‌های فراریخت

۳-۱۲-۱- مشکل و روش پیشنهادی

کشف ویروس‌های فراریخت بخاطر تغییر مداوم ساختار آن‌ها هنوز چالش‌های فراوانی دارد. روش ارائه شده در این مقاله بر اساس یادگیری ماشین آماری است. مدل مخفی مارکوف پیش‌از این برای تشخیص ویروس‌های فراریخت استفاده شده است. نشان داده شده است این روش برای ویروس‌های فراریختی که به نسل‌هایشان شباهت زیادی دارند خوب عمل می‌کند اما نویسندگان در مورد ویروس‌های فراریخت سخت مشهور همچون W95/Zmist، W95/Zperm و W95/Bistro چیزی عنوان نکرده‌اند. مشکل دیگر این روش در زمان برخورد با فایل‌های سالم نرخ بالای مثبت نادرست آن است.

روش این مقاله [45] بر اساس یکی از فنون مشهور تشخیص چهره به نام چهره‌های ویژه[1] کار می‌کند و بر مبنای این اصل استوار است که هر تصویر از ترکیب خطی مجموعه‌ی پایه‌ای چهره‌ها به نام تصاویر ویژه تشکیل شده است.

سیستم ابتدا تعدادی تکرار از ویروس‌های مختلف که شامل بیش از یک فایل برای هر ویروس است را دریافت می‌کند و مجموعه‌ی آموزشی مدل را تشکیل می‌دهد. سپس بردارها و محورهایی که از تفاوت‌های مهم فایل‌های تکرار مقدار می‌گیرد ساخته شده و بردار ویژه نامیده می‌شود. بردارهای ویژه فضای ویژه را می‌سازند. اگر این بردارها را با ضریب وزنی مناسب به‌صورت خطی باهم ترکیب کنیم یکی از تکرارهای ویروس اصلی ساخته می‌شود بنابراین به این بردارها ویروس‌های ویژه[2] می‌گویند. سپس مجموعه تکرارهای اصلی به این فضای برداری تصویر می‌گردند تا وزن‌های متناظر با هر تکرار یافت شود. از این پس هر تکرار ویروس با مجموع وزن‌دار ویروس‌های ویژه شناخته می‌شود. برای تشخیص این‌که ویروس جدید متعلق به یکی از مجموعه‌های اولیه است یا نه، تکرار ویروس ورودی به فضای برداری تصویر می‌گردد سپس مجموع وزن‌دار آن محاسبه می‌گردد. در گام آخر با محاسبه‌ی فاصله‌ی بردار وزن ویروس جدید با مجموعه‌ی اولیه و درنظر گرفته حدی از آستانه‌ی شباهت تعیین می‌گردد که آیا متعلق به یکی از دسته‌های ویروس می‌باشد یا خیر.

با نمایش نمونه فایل‌های ویروس بر حسب ویروس‌های ویژه، کشف ویروس به مسأله تشخیص الگو تبدیل می‌شود و با خوشه‌بندی حل می‌گردد.

[1] Eigenfaces
[2] Eigenvirus

نویسندگان از فاصله اقلیدوسی برای اندازه‌گیری فاصله بین کلاس‌ها استفاده شده است.

۳-۱۲-۲- آزمایش‌ها

در این مقاله از چهار خانواده ویروس مختلف برای آزمون استفاده شد:

۱. ابزار G2 که ابزار ساخت ویروسی است که به‌وسیله‌ی dark angel ساخته شد. ابزار G2 آلوده‌کننده‌های com و exe ۱۶ بیتی می‌سازد. در این مقاله از نسخه‌ی ۱.۰ این ویروس استفاده شده است.
۲. ابزار NGVCK که به زبان ویژوال بیسیک ساخته شده است و در هر اجرا یک کد ویروس را تولید می‌کند. در این مقاله نسخه‌ی ۳.۰ این ابزار استفاده شده و در تولید تمامی فایل‌ها پیکربندی ابزار ثابت نگه‌داشته شده است.
۳. ویروس Zperm: این ویروس در سال ۲۰۰۰ توسط Z0mbie نوشته شده است. ویروس Zperm، ۳۲ بیتی و تحت ویندوز است. این ویروس از موتور جایگشتی برای تولید نمونه‌های بعدی استفاده می‌کند که خود موتور هم دچار جایگشت می‌شود. این ویروس از فنون افزودن و حذف دستورات پرش و بی‌فایده سود می‌جوید.
۴. MetaPHOR: ویروس فراریخت سختی است که توسط Mental Driller ساخته شده است. این ویروس برای محققان ضدویروس چالش‌برانگیز است و تشخیص آن سخت می‌باشد. این ویروس دارای یک رمزگشای 4KB و بدنه‌ی 100KB است. در این مقاله به‌دلیل رمزگذاری نشدن و کوچکی، رمزگشای ویروس بررسی می‌گردد. این ویروس به‌نام‌های W32/Simile و W32/Etap نیز مشهور است.

در این مقاله دو مجموعه‌ی آموزشی و آزمایشی از ویروس‌ها تهیه گشته است. تعداد نمونه‌های لازم برای یک ویروس در مجموعه‌ی آموزشی به میزان فراریختی ویروس بستگی دارد. در این مقاله به‌ترتیب ۱، ۶، ۸ و ۱۵ نمونه‌ی G2، NGVCK، Zperm و MetaPHOR استفاده شده است که در کل مجموعه‌ی آموزشی دارای ۳۰ ویروس است. مجموعه‌ی آزمایشی هم دارای ۲۵۰ نمونه از هر ویروس است که در کل ۱۰۰۰ ویروس را شامل می‌شود.

بعد از ساخت مجموعه آموزشی، ۳۰ ویروس ویژه به‌دست می‌آید که این مقاله نشان داده با استفاده از فقط سه عدد از این ویروس‌های ویژه جواب‌های قابل قبولی مشاهده می‌گردد. به‌دست آوردن تعداد ویروس‌های ویژه که توصیف خوبی از کلاس‌های ویروس ارائه می‌دهند به روش اکتشافی انجام شده است. برای محاسبه‌ی میزان جواب‌های مثبت اشتباه از ۲۵۰ برنامه‌ی سالم Cygwin استفاده شده است.

۳-۱۲-۳- نتایج آزمایش‌ها

فضای برداری ویروس‌های G2 بسیار کوچک است؛ یعنی نمونه‌های آن تفاوت‌های زیادی با هم ندارند. ویروس‌های Zperm هم بالاترین سطح آستانه را داشته‌اند زیرا گونه‌های آن بسیار متفاوت بوده و تغییرات عمده‌ای بین نسل‌هایش دیده می‌شود.

بین اندازه مجموعه آموزشی و نتایج تشخیص دقیق موازنه‌ای وجود دارد. مجموعه آموزشی پایگاه دانش سیستم برای یادگیری ویروس‌هاست. پس هر چه نمونه‌های آموزشی بیشتر باشد، ویژگی‌های بیشتری استخراج شده و نتایج

دقیقتری هم حاصل می‌گردد. اما متقابلاً فضا و پیچیدگی محاسباتی هم بیشتر می‌شود. تعداد ویروس‌های لازم برای تولید نتایج دقیق از یک ویروس به ویروس دیگر در تغییر است. از ویروسی که نمونه‌هایش با هم شباهت بسیار دارند، تعداد اندکی برای آموزش و حصول دقت بالا کافی است.

برای محاسبه فاصله از فاصله اقلیدوسی استفاده شده است. وقتی تعداد کلاس‌های ویروس زیاد باشد، فاصله اقلیدوسی نتایج خوبی حاصل نمی‌کند و فواصل دیگری همچون ماهالانوبیس می‌تواند مفیدتر واقع شود.

برای از بین بردن خطاهای احتمالی در مجموعه‌های آموزشی بزرگ، می‌توان از روش‌های نرمال‌سازی بدافزار بهره برد؛ مثل روش برگرداندن جایگشت کد و امثال آن. این‌گونه پیش‌پردازش‌ها کارایی روش پیشنهادی را در ابزارهای تجاری مضاعف می‌سازد.

۳-۱۳- کشف بدافزار با روش مبتنی بر گراف و استفاده از تحلیل پویا

۳-۱۳-۱- مشکل و روش پیشنهادی

برای مقابله با مشکلات روش‌های مبتنی بر امضاء و کشف ویروس‌های چندریخت، روش‌هایی همچون تحلیل n-gram رد ایستا یا پویای کد بدخواه مطرح شد [50-47]. نشان داده شد که این روش‌ها در شناسایی بدافزارهای جدید شناخته نشده می‌توانند مفید واقع شوند. در روش‌های n-gram دو پارامتر باید تنظیم شود. یکی n است که طول زیررشته‌های متوالی تحلیل را مشخص می‌کند و دیگری L است که تعداد n-gram هایی که باید تحلیل شود را تعیین می‌کند. اگر n و L بزرگ باشند، فضای ویژگی‌ها بزرگ شده و احتمال تفکیک دقیق کدهای بدافزار از فایل سالم بیشتر می‌شود. اما بزرگ شدن مقدار این پارامترها، مشکل ابعاد بزرگ[1] را پدید می‌آورد: فضای ویژگی‌ها آن‌قدر بزرگ می‌شود که داده‌های کافی برای ساخت مدل وجود ندارد. کوچک کردن مقادیر n و L فضای ویژگی‌ها را کم می‌کند ولی در مقابل قدرت تفکیک‌پذیری هم بسیار کم می‌گردد. روش پیشنهادی در این مقاله نیاز به تعیین n و L را مرتفع می‌سازد و داده‌ها را به‌صورت یک زنجیره مارکوف مدل می‌کند.

در روش پیشنهادی [46]، ابتدا رشته کدهای عملیاتی[2] از فایل اجرایی استخراج می‌شود و گراف جهت‌دار و وزن‌دار کدهای عملیاتی ساخته می‌شود. سپس با استفاده از روش‌های هسته‌ای[3] گراف، امتیازی برای گراف مزبور محاسبه می‌شود. اختصاصاً یک ماتریس شباهت (هسته‌ای) بین گراف‌های زنجیره مارکوف ساخته می‌شود. در مرحله بعد از یک ماشین بردار پشتیبان[4] استفاده می‌شود تا فایل تحت بررسی با ماتریس شباهت ساخته شده دسته‌بندی گردد. به‌عبارت بهتر، SVM بر اساس داده‌های آموزشی تصمیم می‌گیرد که امتیاز داده شده مربوط به یک فایل ویروس است یا یک فایل سالم. روش پیشنهادی نویسندگان جزء روش‌های مبتنی بر شباهت کدهای عملیاتی محسوب می‌شود. این روش سپس در [51] بهبود داده شده است. روش پیشنهادی دو جزء جدید دارد: تبدیل داده‌های رد اجرای برنامه به نمایش زنجیره مارکوف و استفاده از روش‌های ماشینی هسته گراف‌ها برای ساخت ماتریس شباهت بین نمونه‌ها. برای ساخت ماتریس هسته هم از دو معیار شباهت بهره برده شده است: یکی هسته گاوس برای محاسبه شباهت محلی بین یال‌های گراف و دیگری هسته طیفی برای اندازه‌گیری شباهت سراسری بین گراف‌ها.

ساخت گراف به‌صورت زیر صورت می‌گیرد. هر کد عملیاتی که در رشته کدهای برنامه باشد، به‌عنوان یک گره مجزا در گراف در نظر گرفته می‌شود. با بررسی کل رشته کدهای عملیاتی (صرف‌نظر از آدرس‌ها، فقط خود کد دستورات)، دستور بعدی هر دستور یا کد عملیاتی استخراج می‌شود. برای هر دستور بعدی، یک یال جهت‌دار از دستور فعلی به دستور بعدی آن رسم می‌شود. وزن یال‌های گراف، احتمال گره بعدی متناظر را نشان می‌دهند. برای روشن شدن روش، رد اجرایی برنامه[5] موجود در شکل ۱۰ را در نظر بگیرید.

1	PUSH	ebp
2	MOV	ebp, esp
3	SUB	esp, 8
4	AND	esp, 0FFFFFFF0h
5	MOV	eax, ds:dword_404000
6	TEST	eax, eax
7	JZ	Short loc_401013
8	INT	3
9	FNSTCW	[ebp+var_2]
10	MOVZX	eax, [ebp+var_2]

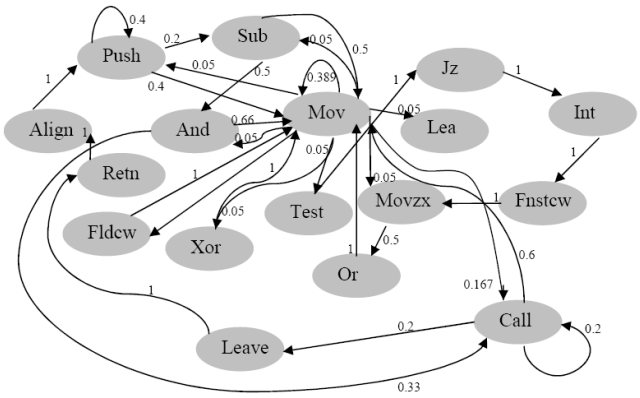
شکل ۱۰. رد دستورات به زبان اسمبلی [46]

ابتدا از برنامه موجود در شکل ۹ رشته کدهای عملیاتی استخراج می‌گردد و در سطر و ستون‌های یک ماتریس (مربعی) تعداد جفت دستورهای متوالی موجود در این رشته درج می‌گردد. به‌عبارتی نمودار فراوانی کدهای عملیاتی به‌دست می‌آید. مثلاً در برنامه نمونه، دستور MOV دقیقاً سه بار بلافاصله بعد از CALL ظاهر شده است. پس در سطر MOV و ستون CALL عدد ۳ نوشته می‌شود. سپس این ماتریس به ماتریس احتمالات (نرمال) تبدیل می‌شود. کافیست عدد هر خانه به مجموع اعداد هر سطر تقسیم شود. برای نمونه با نگاه کردن به سطر MOV معلوم می‌شود که این دستور کلاً ۱۸ بار ظاهر شده است ولی زوج دستور (MOV, CALL) فقط سه بار ظاهر شده‌اند. پس مقدار احتمال در سطر و ستون یاد شده ۳/۱۸ یا یک ششم خواهد شد. نرمال‌سازی برای این است که ماتریس‌ها مستقل از طول فایل شوند و از نظر عددی با هم قابل مقایسه باشند. مقادیر احتمال به‌دست آمده با این روش، وزن یال‌ها در گراف جهت‌دار کدهای عملیاتی خواهند شد. شکل ۱۱ گراف حاصل را نشان می‌دهد. محاسبات وزن‌ها می‌تواند در آرایه نگهداری شود. این گراف وزن‌دار همانند یک زنجیر مارکوف تلقی می‌شود. جدول ۴ هم ماتریس گراف شکل ۱۱ را نشان می‌دهد.

۳-۱۳-۲- آزمایش‌ها

نویسندگان برای ارزیابی روش خود از مجموعه بزرگی از فایل‌های سالم و ویروس استفاده کرده و نتایج خود را با چند نرم‌افزار ضد ویروس رایج نیز مقایسه نموده‌اند.

[1] Curse of dimentionality
[2] Opcode
[3] Kernel
[4] Support Vector Machine (SVM)
[5] Program trace



شکل ۱۱. گراف حاصل از تحلیل کدهای عملیاتی متوالی [46]

جدول ۴. ماتریس مجاورت گراف شکل ۱۰ [46]

	PUSH	MOV	SUB	AND	TEST	JZ	INT
PUSH	2	2	1	0	0	0	0
MOV	1	7	1	1	1	0	0
SUB	0	1	0	1	0	0	0
AND	0	2	0	0	0	0	0
TEST	0	0	0	0	0	1	0
JZ	0	0	0	0	0	0	1
INT	0	0	0	0	0	0	0

۳-۱۳-۳- نتایج آزمایش‌ها

برخی از نتایج به‌دست آمده در جدول ۵ خلاصه شده است. این نتایج از آزمایش روی ۶۱۵ نرم‌افزار سالم و ۱۶۱۵ نمونه بدافزار به‌دست آمده‌اند. روش هسته گراف به‌کار رفته به‌وضوح بهتر از مدل استاندارد *n*-gram کار می‌کند و از نظر دقت هم از همه نرم‌افزارهای ضد ویروس آزمایش شده بالاتر است. اما از نتایج می‌توان دریافت که همه شرکت‌های تولید کننده ضد ویروس محصولات خودشان را آن‌چنان طراحی کرده‌اند که به هر قیمتی از هشدارهای مثبت اشتباه اجتناب کنند، حتی تعداد هشدارهای منفی اشتباه بسیار زیاد شود. در هر صورت روش گراف دستورات عملیاتی نتایج قابل توجهی داشته است.

جدول ۵. خلاصه نتایج [46]

روش	دقت	مثبت اشتباه	منفی اشتباه
Graph kernel	96.41	47	33
n-gram	82.15	300	98
AV0	73.32	0	595
AV1	53.86	1	1028
AV2	49.60	0	1196
AV3	43.27	1	1264
AV4	42.96	1	1271

۳-۱۴- کشف ویروس‌های کامپیوتری فراریخت با روش‌های مبتنی بر مورد

۳-۱۴-۱- مشکل و روش پیشنهادی

نویسنده از روش استنتاج مبتنی بر مورد [52] جهت کشف ویروس‌های فراریخت بهره برده است. اگر امضای ویروس‌های شناخته شده که در پایگاه موردها قرار دارند، در فایلی پیدا شود، این فایل ویروس در نظر گرفته می‌شود. به این ترتیب ویروس‌های مشابه با احتمال بالا کشف می‌شوند. بروزرسانی پایگاه ویروس‌ها بدون اتصال به اینترنت و به‌طور خودکار صورت می‌گیرد. هرگاه ویروسی کشف می‌شود با نرم‌افزار طراحی شده به پایگاه اضافه می‌گردد.

۳-۱۴-۲- آزمایش‌ها و نتایج

نرم‌افزاری تهیه شده که روش پیشنهادی را پیاده‌سازی کرده است. آزمایش‌های ساده‌ای روی نرم‌افزار اجرا شده است.

۳-۱۵- آزمون شباهت برای کشف ویروس‌های فراریخت

۳-۱۵-۱- مشکل و روش پیشنهادی

در این پروژه [53] دو روش برای اندازه‌گیری شباهت بین ویروس‌های نامشابه پیشنهاد شده است. از این روش‌ها برای کشف ویروس‌های فراریخت استفاده می‌شود. روش‌های مبتنی بر شباهت یک برنامه ورودی را دسته‌بندی می‌نمایند که آیا از خانواده یکی از ویروس‌هاست یا برنامه غیرویروسی است. برای این منظور از نتایج مقایسه‌های متعدد بین چند ویروس و غیرویروس، بین چندین ویروس و بین چند برنامه غیرویروس بهره می‌گیرند تا بتوانند دسته‌بندی درستی صورت دهند. روش *n*-gram یکی از روش‌های محاسبه شباهت است. دو روش پیشنهادی در این پروژه عبارتند از فاصله تغییرات[1] و روش ترازبندی رشته‌های جفتی[2].

۳-۱۵-۲- آزمایش‌ها

به منظور ارزیابی روش‌ها، مقایسه‌ای بین ۴۰ فایل از مجموعه cygwin DLL که به‌طور تصادفی انتخاب شده‌اند و ۴۰ ویروس تولید شده با موتور فراریخت NGVCK صورت گرفته است. با کمک ابزار IDA Pro فایل ویروس‌ها و فایل‌های سالم به فایل‌های اسمبلی برگردانده شده‌اند.

جهت انجام آزمایش، امتیاز شباهت بین فایل ویروس و فایل سالم محاسبه می‌شود. اگر امتیاز از مقدار آستانه کمتر باشد، برنامه به خانواده ویروس تعلق دارد. مقدار آستانه حداقل امتیاز شباهت بین فایل‌های سالم در نظر گرفته می‌شود. امتیاز شباهت بین فایل‌های سالم، بین فایل‌های ویروس و سالم و بین فایل‌های سالم و کپی تغییریافته ویروس اصلی که طیف متنوعی از زیرروال‌ها و کد مرده در آن‌ها درج شده باشند، مقایسه شده است. اگر شباهت یک فایل ناشناخته با یک فایل غیرویروسی از حد آستانه کمتر شود، فایل ناشناخته به خانواده ویروس‌ها تعلق دارد. محاسبه شباهت در سه سری

[1] Edit distance
[2] Pairwise Sequence Alignment Method

آزمایش با روش n-gram، شباهت فاصله تغییرات و روش ترازبندی رشته‌های جفتی صورت گرفته است.

۳-۱۵-۳- نتایج آزمایش‌ها

نتایج استفاده از شباهت با فاصله تغییرات و روش ترازبندی رشته‌های جفتی نشان داد که ویروس‌های تغییر شکل یافته که درصد درج کد مرده و زیرروال در آن‌ها در محدوده ۵٪، ۱۵٪، ۲۵٪ و ۳۰٪ بوده با یک نرخ خطای معین قابل کشف هستند. روش شاخص شباهت[1] یا همان n-gram ویروس‌های تغییر شکل یافته را حداکثر تا محدوده درج ۲۵٪ کد مرده و زیرروال بدون خطا کشف می‌کند.

روش شاخص شباهت بهترین نتیجه را برای ویروس‌های تغییر شکل یافته تا محدوده درج ۲۵٪ کد مرده و زیرروال بدون خطا تولید کرده است. در ویروس‌های ۳۰٪ تغییر یافته، این روش ۶٪ و ۱۳/۳۳٪ نرخ خطا برای پنجره‌هایی به‌ترتیب با اندازه ۲۰ و ۲۵ داشته است.

در آزمایش‌ها روش فاصله تغییرات، ویروس‌های پایه را با نرخ خطای ۱/۱۶٪ تفکیک کرده است. روش شباهت ترازبندی رشته‌های جفتی نتایج بهتری نسبت به فاصله تغییرات داشته است، زیرا ویروس‌های پایه را بدون خطا کشف کرده است.

۳-۱۶- شباهت گراف کدهای عملیاتی و کشف فراریختی

۳-۱۶-۱- مشکل و روش پیشنهادی

روش پیشنهادی [51] در زمره روش‌های اندازه‌گیری شباهت بر اساس گراف کدهای عملیاتی است ولی ساده‌تر و کارامدتر از روش ارائه شده در [46] کار می‌کند. ضمن اینکه تمرکز آن بر کشف ویروس‌های فراریخت است. روش پیشنهادی این مقاله هم با دریافت یک فایل اجرایی، رشته کدهای عملیاتی را استخراج می‌کند و از روی آن گراف جهت‌دار و وزن‌دار را می‌سازد. اما بجای استفاده از هسته گراف‌ها برای محاسبه امتیازها و SVM برای دسته‌بندی، مستقیماً گراف‌های کدعملیاتی را مقایسه می‌کند.

فرض کنید که N تعداد کدهای عملیاتی متمایزی باشد که در ساخت گراف به‌کار رفته‌اند و A و B هم دو ماتریس یال-وزن متناظر با دو فایل اجرایی A و B باشند. این ماتریس‌ها $N\times N$ هستند و شماره‌گذاری کدهای عملیاتی در هر دو ماتریس یکسان است. بنابراین با فرمول (۴)، امتیازی برای مقایسه دو ماتریس محاسبه می‌گردد.

$$score(A,B)=\frac{1}{N^2}(\Sigma_{i,j=0}^{N-1}\left|a_{ij}-b_{ij}\right|)^2 \qquad (۴)$$

اگر $A=B$ باشد، امتیاز حداقل مقدار صفر را خواهد داشت. اگر $a_{ij}=1$ و $b_{ik}=1$ و $j\neq k$ باشد، حداکثر مقدار مجموع سطری را به‌دست می‌آوریم.

$$\Sigma_{j=0}^{N-1}\left|a_{ij}-b_{ij}\right|=2$$

اگر برای همه سطرها این مقدار به‌دست آید، آن‌گاه بیشترین امتیاز چهار خواهد شد و داریم:

$$0\le score(A,B)\le 4,\quad \forall A,B$$

برای به‌دست آوردن آستانه[2] امتیاز از روش زیر استفاده شده است:

۱. گراف کدهای عملیاتی برای خانواده‌ای از ویروس‌های فراریخت محاسبه می‌شود.
۲. گراف کدهای عملیاتی برای نمونه‌ای از فایل‌های سالم به‌دست می‌آید.
۳. با فرمول امتیاز پیشنهادی، برای همه جفت گراف‌های بند ۱، امتیازی محاسبه می‌شود.
۴. با فرمول امتیاز پیشنهادی، برای همه جفت‌های متشکل از یک ویروس فراریخت و یک فایل سالم، امتیاز محاسبه می‌شود.
۵. بر اساس امتیازهای بندهای ۳ و ۴، آستانه تعیین می‌گردد.

وقتی‌که سطح آستانه مشخص شد، برای تعیین وضعیت یک فایل، ابتدا یک ویروس تصادفی از خانواده فراریخت در بند ۱ را انتخاب کرده و با ساخت گراف، آن‌ها را مقایسه می‌کنیم. اگر امتیاز از آستانه کمتر بود، این فایل از خانواده فراریخت است، در غیر این صورت، سالم است. در [55] پنج گونه مختلف از تابع امتیازدهی روش پیشنهادی برای گراف‌ها ارائه شده است و همانجا ذکر شده که تغییرات کوچک بر تابع امتیازدهی می‌تواند منجر به تأثیرات قابل توجهی بر نتایج شود.

۳-۱۶-۲- آزمایش‌ها

برای مقایسه از ۲۰۰ فایل ویروس تولید شده توسط NGVKS و ۴۱ فایل سالم از مجموعه cygwin استفاده شده است. علت استفاده از این خانواده ویروس این است که مطالعات درجه بالای فراریختی آن‌ها را نشان داده‌اند. همچنین از ۲۵ فایل ویروس غیر هم خانواده هم استفاده شده است. با روش پیشنهادی حد آستانه تعیین شد و اولین مجموعه آزمایش‌ها برای مقایسه روش پیشنهادی با روش مبتنی بر HMM ترتیب یافت.

۳-۱۶-۳- نتایج آزمایش‌ها

تحت شرایطی، روش پیشنهادی کارایی بهتری نسبت به روش قبلی مبتنی بر HMM داشته است. به‌علاوه، نتایج نشان دادند که معیار شباهت پیشنهادی فقط منحصر به کشف فراریختی نیست. نویسندگان دو حمله را نیز تشریح کرده‌اند و نشان داده‌اند که روش پیشنهادی آن‌ها در مقابل حمله حذف کدهای عملیاتی نادر مقاوم است. حمله دوم، اصلاح موتور تغییر شکل است، که در دو حالت بررسی شده است. یکی تغییر شکل بلوکی[3] است که در آن کدی از یک فایل سالم به‌عنوان یک بلوک واحد کد مرده داخل کد ویروس درج می‌شود. حالت دوم تغییر شکل تصادفی[4] است که کد برداشته شده از فایل سالم تقریباً به‌طور یکنواخت در داخل ویروس تغییر شکل یافته به‌عنوان کد مرده درج می‌شود. در حالت اول، هرچه کد مرده بیشتری درج شود، امتیاز مقایسه فایل‌ها کمتر می‌شود و کشف دشوارتر می‌گردد. در این حالت روش پیشنهادی چند دسته‌بندی اشتباه دارد. در حالت دوم، نتایج آزمایش‌ها نشان داده‌اند که تغییر شکل تصادفی منجر به ساده‌تر شدن کشف می‌شود نه دشواری آن.

---

[1] Similarity index
[2] Threshold
[3] Block morphing
[4] Random morphing

۳-۱۷- کشف ویروس‌های فراریخت با استفاده از فاصله کای دو

۳-۱۷-۱- مشکل و روش پیشنهادی

هر چند که مدل مخفی مارکوف در شناسایی ویروس‌های فراریخت مؤثر است، ولی با درج مقداری کد از یک فایل سالم داخل فایل ویروس می‌توان از کشف ویروس توسط HMM جلوگیری کرد. آن‌قدر از فایل سالم در فایل ویروس کد درج می‌شود که در نقطه‌ای دیگر دسته‌بند HMM قادر به کشف ویروس فراریخت نخواهد بود یا نمی‌توان به دسته‌بند برای تفکیک ویروس از فایل سالم اعتماد کرد. یکی از مطالعات پیشین [20] ابزاری تولید کرده است که از این نقطه ضعف سوء استفاده می‌کند و ویروس‌های فراریختی با درجه بالا می‌سازد. اگر قطعه‌های کوچک کد از فایل سالم داخل فایل ویروس درج شود، یا تغییرات اعمال شده به کد ویروس تصادفی باشند، باز هم روش مبتنی بر HMM مقاوم است و خوب کار می‌کند. اما هنگامی که کد کپی شده از فایل‌های سالم بلوک‌های پیوسته کد باشد، مثلاً کل یک زیرروال، روش مبتنی بر HMM دیگر موفق عمل نمی‌کند. هدف از این مقاله بهبود این نقطه ضعف است.

در این مقاله [6] از فاصله آماری کای دو[1] برای بهبود روش‌های موجود کشف ویروس‌های فراریخت استفاده شده است. مبانی نظری این روش در [56] پیشنهاد شده و مورد مطالعه قرار گرفته است. روش پیشنهادی برای یک فایل اجرایی داده شده، ابتدا طیفش[2] را محاسبه می‌کند، یعنی فراوانی دستورات موجود در این فایل را به‌دست می‌آورد. سپس با کمک آزمون فرض آماری مشخص می‌کند که آیا این فراوانی مربوط به فایل سالم است (یعنی آیا با فراوانی مورد انتظار یک فایل سالم مطابقت دارد) یا فایل ویروس و یکی از دو فرض صفر یا فرض مقابل را رد می‌کند و دیگری را می‌پذیرد. به‌عبارت دقیق‌تر، اگر فراوانی‌ها یکسان بودند، فایل مشکوک به یک خانواده از ویروس تعلق دارد و فرض صفر پذیرفته می‌شود. اما اگر فراوانی‌ها تفاوت قابل ملاحظه‌ای داشته باشند، فرض مقابل پذیرفته می‌شود و فایل مشکوک به‌عنوان یک فایل سالم در نظر گرفته می‌شود.

۳-۱۷-۲- آزمایش‌ها

برای آزمایش‌ها از تخمین چرخشی یا همان اعتبارسنجی متقاطع[3] پنج تایی استفاده شده است. برای ارزیابی هم از متریک‌های دقت[4] و منحنی ROC استفاده شده است. مجموعه داده‌های آزمایشی ۲۰۰ ویروس فراریخت هم‌خانواده از NGVCK و ۴۰ فایل سالم از مجموعه فایل‌های cygwin انتخاب شده است. هر فایل ویروس با ابزار IDA Pro به اسمبلی برگردان شده است.

۳-۱۷-۳- نتایج آزمایش‌ها

برای کشف مبتنی بر CSD، تفاوت زیادی ندارد که کد سالم داخل فایل ویروس پخش شود (به‌صورت کد مرده) یا به‌صورت بلوک‌های پیوسته (مثل کد زیرروال) داخل ویروس درج گردد. اما این موضوع برای HMM صادق نیست زیرا وقتی کد درج شده یک بلوک پیوسته باشد، کشف کننده مبتنی بر HMM موفق عمل نمی‌کند. پس معلوم می‌شود که دو روش CSD و HMM کمیت‌های آماری کاملاً متفاوتی تولید می‌کنند. نتایج آزمایش‌ها نشان دادند که آماره کای دو که از روی فراوانی کدهای عملیاتی محاسبه شده باشد قادر است به کشف ویروس‌های فراریخت کمک کرده و به‌طور چشمگیری آن‌را بهبود دهد. میزان کد کپی شده از فایل سالم، چه کم باشد و چه بلوک بزرگ و پیوسته‌ای از کد، تأثیر چندانی بر آماره کای دو ندارد. ترکیب آماره کای دو با روش مبتنی بر HMM، نتایج بهتری نسبت به استفاده مجزای هر یک داشته است.

۳-۱۸- کشف گونه‌های بدافزار تغییرشکل‌یافته از طریق تخصیص موتور

۳-۱۸-۱- مشکل و روش پیشنهادی

مبهم‌سازی با فراریختی، کشف ویروس‌ها را برای ضد ویروس‌ها دشوارتر می‌سازد. اگر روش کشف ایستا مبتنی بر ساختار کد ویروس کار کند، شانس موفقیت آن بسیار پایین می‌آید. از این‌رو برخی از روش‌های پیشنهادی از فنون پیشرفته تحلیل معنای برنامه بهره می‌برند. این‌گونه فنون از روش‌های توصیف رسمی بهره می‌برند که بتوانند در مورد عملکرد بدخواهانه کد قضاوت نمایند. متأسفانه اکثر این روش‌ها به‌دلیل پیچیدگی بالای فرایند تحلیل یا قیود و محدودیت‌های خاص روی محیط در عمل قابل پیاده‌سازی نیستند.

روش پیشنهادی [57] در این مقاله برای کشف بدافزار فراریخت، از تخصیص نویسندگی[5] استفاده می‌کند. تخصیص نویسندگی شیوه‌ای در علوم اجتماعی است که هدف آن تعیین نویسنده یک سند است. برای این‌منظور از آثار قبلی یک نویسنده، خصوصیت‌ها و الگوهای نویسندگی متنی او استخراج می‌شود و به‌کمک آن‌ها مشخص می‌شود که آیا یک سند جدید توسط فلان نویسنده نوشته شده است یا خیر. به این خصوصیت‌ها متمایزکننده سبکی[6] گفته می‌شود که ضمن اینکه در آثار یک نویسنده خاص ثابت هستند، در عین حال در بین آثار سایر نویسندگان متفاوت خواهد بود. فنون تخصیص نویسندگی سال‌هاست در کشف سرقت ادبی، تصدیق نویسنده و امروزه در بیومتریک‌ها و تحلیل نویسندگی کد منبع مورد استفاده قرار می‌گیرد.

۳-۱۸-۲- آزمایش‌ها

آزمایش‌ها روی هفت موتور تغییر شکل صورت گرفته‌اند: ADMmutate, CLET, NGVCK, VCL, Shikata Na Gai, Fnstenv Mov و Call4dWord. با هر موتور ۱۰۰ نمونه بدافزار تولید شدند که خروجی چهار موتور اول کد باینری و خروجی سه موتور آخر به زبان اسمبلی هستند.

۳-۱۸-۳- نتایج آزمایش‌ها

دقت کشف در آزمایش‌های هر یک از سه روش پیشنهادی حداقل ۹۶٪ بوده است، درحالیکه اندازه امضای موتورها بین یک عدد حقیقی تا ماتریس مربعی ۱۷×۱۷ از اعداد حقیقی بوده است.

---

[1] Chi-squared distance (CSD)
[2] Spectrum
[3] Cross-validation
[4] Accuracy
[5] Authorship attribution
[6] Stylistic discriminator

۳-۱۹- فاصله جانشینی ساده و کشف ویروس‌های فراریخت

۳-۱۹-۱- مشکل و روش پیشنهادی

هدف شناسایی بدافزارهای فراریخت با کمک روش تحلیل رمز[1] سیستم رمزنگار[2] جانشینی[3] ساده است و از معیار شباهت مبتنی بر کدعملیاتی[4] استفاده شده است. در سیستم رمزنگار جانشینی ساده هر حرف متن اصلی[5] به یک حرف متن رمزی[6] نگاشت می‌یابد. یعنی نگاشت یک به یک بین حروف متن اصلی و متن رمزی وجود دارد. اگر سیستم رمزنگار روی حروف الفبای انگلیسی طراحی شده باشد، با داشتن آمار حروف انگلیسی و متنی با طول مناسب، می‌توان آن‌را رمزگشایی نمود. اما اگر با همین آمار، رمزگشایی روی متنی از کلمات فرانسوی به‌کار رود، نتیجه خوبی به‌دست نمی‌آید.

روش پیشنهادی [29] سعی می‌کند رشته کدهای عملیاتی یک فایل مشکوک را بر اساس آمار کدهای عملیاتی حاصل از یک خانواده از بدافزارهای فراریخت به اصطلاح «رمزگشایی» کند. سپس با محاسبه امتیازی بررسی می‌کند که فایل رمزگشایی شده چقدر شبیه آماره مستخرج از ویروس‌های هم‌خانواده است. در این‌صورت فایل به‌عنوان بدافزار شناخته می‌شود. توجه کنید که فایل بدافزار واقعاً رمزگشایی نمی‌شود، زیرا رمزگذاری نشده بوده است. به‌عبارت بهتر، فرایند محاسبه امتیازها در هر مرحله به‌نوعی ابهام‌زدایی[7] فایل بدافزار محسوب می‌گردد. این فرایند مشابه بازیابی متن اصلی از یک متن رمزی بر اساس آمار حروف الفبای انگلیسی است. ولی چنانچه فایل مشکوک، بدافزار نباشد و فرایند ابهام‌زدایی و محاسبه امتیاز شباهت روی آن به‌کار رود، مثل این‌است که از آمار حروف انگلیسی برای رمزگشایی متنی از کلمات فرانسوی استفاده شده است که نتایج ضعیفی خواهد داشت.

۳-۱۹-۲- آزمایش‌ها

آزمایش‌های صورت گرفته ابتدا پارامترهای مختلف را مورد آزمون قرار داده‌اند تا مقدار مناسب برای اندازه ماتریس کد عملیاتی، فرمول محاسبه امتیاز مناسب، روش بهینه نرمال‌سازی و راه‌کار مناسب جابجایی تعیین شوند. سپس سه خانواده ویروس NGVCK، G2 و کرم‌های MWOR برای آزمایش‌ها مورد استفاده قرار گرفتند. برای نمایش نتایج آزمایش‌ها از نمودار ROC استفاده شده است و ارزیابی با روش متقاطع پنج تایی صورت گرفته است.

۳-۱۹-۳- نتایج آزمایش‌ها

روش پیشنهادی قادر است به‌راحتی ویروس‌های خانواده G2 و NGVCK را تشخیص دهد. اما روش پیشنهادی در کشف کرم‌های MWOR کارایی قوی ندارد به‌خصوص وقتی نسبت کد افزوده سالم داخل کرم‌ها افزایش می‌یابد. کرم‌های MWOR طوری طراحی شده‌اند که روش‌های کشف بدافزارهای فراریخت به آسانی نتوانند آن‌ها را کشف نمایند. ولی نتایج کشف کرم‌ها با روش پیشنهادی از نتایج کشف کرم‌ها توسط ترکیبی مدل مخفی مارکوف (که نتایجش به‌عنوان محکی جهت مقایسه سایر روش‌ها به‌کار می‌رود) بهتر بوده است.

۳-۲۰- تحلیل ایستا برای کشف ویروس‌های کامپیوتری فراریخت با روش‌های ابتکاری شمارش دستورات تکراری

۳-۲۰-۱- مشکل و روش پیشنهادی

روش‌های کشف موجود فقط در مقابل ویروس‌های شناخته شده مؤثر هستند، زمانگیر هستند و پایگاه امضای آن‌ها مرتباً باید بروزرسانی شود. روش پیشنهادی در این مقاله با تحلیل ایستای فرکانس تکرار دستورات کد ویروس حاصل از برگرداندن کد به اسمبلی کار می‌کند. سپس از یک دسته‌بند[8] روی فرکانس دستورات استفاده می‌شود تا تعیین کند برنامه مشکوک ویروس است یا خیر.

روش پیشنهادی [30] از عوارض جانبی روش‌های تولید بدافزارهای فراریخت بهره می‌برد. یکی از این عوارض این‌است که به‌کارگیری ترکیبی از دو یا چند روش مبهم‌سازی منجر به پخش دستورات یکسان و تکراری در ویروس‌های فراریخت می‌شود؛ مثل دو ویروس Lexotan و Puron.

۳-۲۰-۲- آزمایش‌ها

آزمایش روی ۱۰۰۰ برنامه شامل ۵۰۰ برنامه غیربدافزار، ۲۵۰ بدافزار فراریخت و ۲۵۰ بدافزار غیر فراریخت صورت گرفته است. همچنین برای ساختن دسته‌بندها از پنج الگوریتم یادگیری ماشینی استفاده شده است. آزمایش دوم روی ویروس‌های فراریختی انجام شده که مقداری کد مرده از فایل‌های سالم به آن‌ها افزوده شده است، زیرا افزودن کد مرده یکی از فنون مؤثر در ساخت بدافزارهای فراریختی با درجه بالاست.

فایل‌های ویروس‌ها از مجموعه G2, MPCGEN, NGVCK, NRLG و SMEG گرفته شده‌اند. در آزمایش‌ها از روش آزمون فرض آماری برای بررسی اینکه آیا بین تعداد کدهای عملیاتی فایل‌های مختلف (بدافزار، غیربدافزار، فراریخت، غیر فراریخت) تفاوت آماری وجود دارد یا خیر، بهره برده شده است. برای دسته‌بندی فایل‌ها هم از ابزار weka برای ساخت پنج نوع درخت تصمیم استفاده شده است که عبارتند از: J48, LADTREE, NBTREE, RANDOMFOREST و REPTREE. پیش از انجام آزمایش‌ها، ماتریس‌ها نرمال‌سازی می‌شوند: هر عنصر بر مجموع تعداد کل درایه‌ها تقسیم می‌شود، مانند جدول ۶.

برای مقایسه داده‌های ویروس‌ها و غیر بدافزارها، از فرمول (۱۱) استفاده شده است:

$$PD(op_i) = \frac{IOM_N^{NM}(op_i) - IOM_N^{M}(op_i)}{\max(IOM_N^{NM}(op_i), IOM_N^{M}(op_i))} \qquad (۱۱)$$

[1] Cryptanalysis
[2] Cipher
[3] Substitution
[4] Opcode
[5] Plaintext
[6] Ciphertext
[7] De-obfuscate
[8] Classifier

جدول ۶. داده‌های نرمال شده

تعداد دستورالعمل‌های منحصر به‌فرد که بیشتر از یک بار ظاهر شده‌اند: IOM	کدهای عملیاتی ۸۰۸۶
۰	AAA
۰/۰۰۷۶۹	AAD
۰/۰۰۴۶۱	AAM
۰/۰۰۳۰۷	AAS
۰	ADC
۰/۰۲۳۰۷	ADD
...	...

۳-۲۰-۳- نتایج آزمایش‌ها

نتایج نشان می‌دهند که روش پیشنهادی در کشف موارد زیر مؤثر است:

۱. ویروس‌های فراریخت از برنامه‌های غیر بدافزار
۲. ویروس‌های فراریخت از بدافزار غیر فراریخت
۳. غیر بدافزار از بدافزار غیر فراریخت
۴. غیر بدافزار از هر نوع بدافزار مورد استفاده از آزمایش

نرخ هشدارهای مثبت اشتباه هم در آزمایش‌ها گواه دقت بالای روش بوده است. روش پیشنهادی دقت بالایی در کشف بدافزارهای فراریخت دارد.

۳-۲۱- کرم فراریختی که موتور فراریختی خودش را با خود دارد

۳-۲۱-۱- مشکل و روش پیشنهادی

کرم‌ها بر خلاف ویروس‌ها خود تکثیر هستند. به‌همین دلیل کرم فراریخت نیاز به حمل موتور فراریختی‌اش با خود دارد. با توجه به اینکه موتور فراریختی خودش یک امضاء محسوب می‌شود، کرمی که موتور فراریختی با خودش دارد، مجبور است هنگام تکثیر موتور تغییر شکلش را نیز تغییر دهد. این مسأله مشکل و محدودیت‌هایی در ساختار موتور تغییر شکل به‌وجود می‌آورد. نشان داده شده [17] که HMM در کشف ویروس‌های فراریختی با درجه بالا بسیار کارامد است. نتایج کشف با HMM در [21] نرخ ۹۰ درصد کشف و نرخ هشدارهای مثبت اشتباه کمتر از ۱۰ درصد را نشان داده است. روش‌هایی برای جلوگیری از کشف توسط HMM هم وجود دارد [39, 20]. یکی از مطالعات [20] از فایل‌های سالم کدی به‌عنوان کد مرده در ویروس‌ها درج می‌کند. همچنین معلوم شده که [39] افزایش میزان کد مرده باعث شباهت بیشتر ویروس فراریخت به فایل سالم شده و از کشف آن جلوگیری می‌کند. همچنین نتایج [39] بررسی‌ها نشان داده‌اند که درج رشته‌های طولانی کد مثل زیرروال‌ها برای جلوگیری از کشف توسط HMM، مؤثرتر از درج بلوک‌های تصادفی کد است. روش پیشنهادی این مقاله [2] حین درج کد مرده از این نتیجه بهره برده است.

۳-۲۱-۲- آزمایش‌ها

آزمایش‌های صورت گرفته فقدان وجود شباهت بین نسل‌های یک ویروس و قابلیت عدم کشف آن را مورد بررسی قرار داده‌اند. برای این منظور از روش شباهت n-gram [32]، شباهت با روش گراف [51] و کشف مبتنی بر HMM [17] استفاده شده است. یک راه کارا برای جلوگیری از کشف بر اساس HMM این است که بلوک‌های بزرگی از رشته دستورات از فایل‌های اجرایی سالم انتخاب شود و در فایل ویروس درج شود تا فایل ویروس از نظر آماری شبیه فایل سالم گردد.

برای هر آزمایش ۱۰۰ کرم تولید شد و ۲۰ فایل سالم هم انتخاب شد. از هر ۱۰۰ کرم، ۸۰ مورد برای آموزش HMM و ۲۰ تای باقی‌مانده همراه ۲۰ فایل سالم با HMM آموزش دیده مورد آزمایش قرار گرفتند. مقایسه شباهت دو به دو بین ۲۰ کرم تولیدی و نیز دو به دو بین ۲۰ کرم و ۲۰ فایل سالم صورت گرفته است.

۳-۲۱-۳- نتایج آزمایش‌ها

نتایج آزمایش‌ها و نمودار ROC نشان می‌دهند که وقتی نسبت لایی‌گذاری از ۲/۵ بیشتر شود، مدل HMM در کشف دچار اشتباه می‌شود و دیگر خوب کار نمی‌کند. نتایج محاسبه شباهت با روش n-gram هم نشان داد که بدنه کرم‌ها آن‌قدر متفاوت شده‌اند که روش‌های مبتنی بر امضاء نتواند آن‌ها را کشف کند. در محاسبه شباهت با گراف هم کرم‌ها را نتوانستند بر اساس آستانه شباهت از هم تفکیک نمایند.

۳-۲۲- بدافزار فراریخت و بی‌نظمی ساختاری

۳-۲۲-۱- مشکل و روش پیشنهادی

روش پیشنهادی زیرمجموعه روش‌های آماری محسوب شده و مبتنی بر شباهت میان دو فایل کار می‌کند. این روش از بی‌نظمی ساختاری برای تحلیل تغییرات داخل داده‌های یک فایل بهره می‌برد

روش پیشنهادی [4] برای تشخیص اینکه آیا یک فایل مفروض به خانواده بدافزار فراریخت شناخته شده‌ای تعلق دارد یا خیر، به‌کار رفته است. روش پیشنهادی دو فایل را مقایسه کرده و مقدار شباهت را تولید می‌کند. برای این منظور، هر فایل را به رشته‌ای از قطعات با سطوح آنتروپی متفاوت تقسیم می‌کند و برای تقسیم‌بندی از تحلیل موجک بهره می‌برد. سپس دو رشته را ترازبندی می‌نماید. برای ترازبندی فاصله تغییرات را بین سگمنت‌های رشته محاسبه می‌نماید. حاصل این محاسبه شباهت بین دو فایل است که بر حسب درصد بیان می‌شود.

۳-۲۲-۲- آزمایش‌ها

برای ارزیابی روش پیشنهادی، مسأله کشف ویروس‌های فراریخت مورد بررسی قرار گرفته است و آزمایش‌هایی مشابه [51] با اندکی تغییر ترتیب داده شده است. تنها تفاوت تعداد مقایسه‌ها یا جفت فایل‌های مورد مقایسه بوده است. در این مقاله همه جفت فایل‌های ممکن با هم مقایسه شده‌اند و آزمایش کاملتری صورت گرفته است. هدف تفکیک درست فایل‌های ویروس از فایل‌های سالم است. برای این منظور باید آستانه شباهتی تعریف کرد که فایل‌های بدخواه را از فایل‌های سالم تفکیک نماید.

۳-۲۲-۳- نتایج آزمایش‌ها

نتایج نشان داده‌اند که روش پیشنهادی قادر است نمونه‌های بدافزار تولیدی با مولدهای فراریخت G2 و MWOR را با نرخ ۱۰۰٪ شناسایی کند. حتی وقتی میزان کد افزوده شده از فایل‌های سالم به ویروس‌های MWOR زیاده بوده

است، روش پیشنهادی موفق عمل کرده است، درحالیکه روش‌های دیگر مبتنی بر کد عملیاتی در این شرایط عملکرد خوبی نداشته‌اند. همچنین، میانگین شباهت بین فایل‌های MWOR با افزایش نرخ لایی‌گذاری به آرامی کاهش یافته است.

نتایج آزمایش‌ها روی خانواده ویروس NGVCK نشان دادند که روش پیشنهادی قادر است ویروس‌ها را از فایل‌های سالم با نرخ ۱/۵٪ تطابق اشتباه در مورد فایل‌های چهار کیلوبایتی تفکیک کند. در کل، نتایج حاکیست که معیار شباهت با بی‌نظمی ساختاری برای دسته‌بندی بدافزارهایی با درجه بالای فراریختی مفید واقع می‌شود.

۳-۲۳- تحلیل مقادیر ویژه برای کشف فراریختی

۳-۲۳-۱- مشکل و روش پیشنهادی

در این مقاله [58] روش پیشنهادی قبلی [45] مبتنی بر بردار برای تشخیص فراریختی تحلیل می‌گردد. ضمن اینکه تغییراتی برای بهبود عملکرد روش قبلی داده شده است. روش قبلی که مبتنی بر فنون تشخیص چهره کار می‌کند و آن‌ها را برای تشخیص بدافزارهای فراریخت توسعه داده است. در این روش، بردارهای ویژه حاصل از بایت‌های خام فایل‌های اجرایی متعلق به خانواده‌ای از ویروس‌های فراریخت تهیه می‌گردد. سپس از این بردارهای ویژه جهت امتیازدهی به فایل‌هایی متشکل از خانواده از ویروس‌ها و نیز فایل‌های سالم استفاده می‌گردد.

۳-۲۳-۲- آزمایش‌ها

از آزمون‌های فراگیر برای سنجش تاثیرگذاری این روش دسته‌بندی استفاده شده است. آزمایش‌ها روی ویروس‌های MPCGEN, G2 و NGVCK و نیز روی کرم‌های MWOR صورت گرفته است. بجز کرم‌های MWOR، بقیه ویروس‌ها در محیط ویندوز اجرا می‌شوند. کرم‌های MWOR در محیط لینوکس اجرا می‌شوند و به‌همین جهت با فایل‌های سالم ویندوز مقایسه شده‌اند. همچنین از ۳۳ فایل مجموعه cygwin و ۱۱ فایل لینوکس به‌عنوان فایل‌های سالم در آزمایش‌ها استفاده شد. در آزمایش‌ها از روش ارزیابی متقاطع پنج‌تایی بهره برده شده است.

۳-۲۳-۳- نتایج آزمایش‌ها

نتایج نشان دادند که روش مورد آزمایش قادر است کرم‌های MWOR که درجه فراریختی آن‌ها بالاست را نیز تشخیص دهد. این کرم‌ها، به‌خصوص آن‌هایی که نرخ کدهای افزوده شده بالایی دارند، توسط سایر روش‌های آماری موجود قابل تشخیص نبودند یا با دقت پایین کشف می‌شدند.

نتایج همچنین نشان دادند که استفاده از این روش روی کدهای عملیاتی به‌جای بایت‌های خام، نتایج بهتری ندارد و گاهی نتایج بدتر هم می‌شود. از قسمت دیگری از نتایج هم معلوم شد درصد بسیار کمی از مهمترین بردارهای ویژه، حامل اطلاعات مفید کد ویروس هستند.

۴- مقایسه روش‌ها و نقاط قوت و ضعف آن‌ها

در این بخش نقاط قوت و ضعف روش‌های مورد مطالعه در بخش سوم ارائه می‌شود. استخراج این نقاط قوت و ضعف از نوآوری‌های این مقاله به‌شمار می‌آید. به‌منظور فراهم شدن بستری برای مرور سریع روش‌ها و ویژگی‌هایشان و مقایسه آن‌ها، این اطلاعات در قالب جدول نشان داده شده‌اند. در این جدول، یک ستون به نام مقاله، ستون بعدی به نویسندگان و سال انتشار مقاله و ستون سوم به ارائه نقاط قوت و ضعف روش اختصاص پیدا کرده است. مقاله‌ها به‌ترتیب سال از گذشته تا امروز مرتب شده‌اند.

با مرور ستون اول می‌توان سیر نوآوری‌ها و پژوهش‌های مؤثر در زمینه روش‌های کشف ویروس‌های فراریخت و مطالعه‌هایی که روش‌ها را آزموده و ارزیابی کرده‌اند را بررسی کرد. ستون دوم، آشنایی از پژوهشگران مطرح و پرکار در این حوزه و فراوانی تحقیقات در هر سال را فراهم می‌سازد. ستون سوم هم زمینه تحقیقات آتی پیرامون کشف بدافزار فراریخت را آماده می‌کند؛ چراکه برای خلق روش جدیدتر یا بهبود روش‌های موجود، ضمن تسلط بر دانش موجود، باید مشکلات و خلأ تحقیقاتی در راه‌کارهای کنونی را نیز بشناسیم.

جدول‌های ۷ الی ۱۱ به‌ترتیب نتایج سال‌های ۲۰۰۴ و ۲۰۰۵، سال ۲۰۰۶، سال‌های ۲۰۰۷ تا ۲۰۱۰، سال ۲۰۱۱ و ۲۰۱۲ و سال ۲۰۱۳ و ۲۰۱۴ را نشان می‌دهند.

۵- نتیجه‌گیری

مقاله حاضر مروری داشت بر تحقیق پیرامون کشف ویروس‌های فراریختی. ابتدا در بخش دوم ادبیات موضوع و تمام واژگان ضروری، اصول و مفاهیم پایه‌ای با جزئیات کافی مورد بحث قرار گرفتند. در بخش سوم، ۲۳ روش اخیر و با اهمیت در کشف بدافزارهای فراریختی به تفصیل ارائه شد. برای هر روش جزئیات روش پیشنهادی، شرح آزمایش‌های انجام شده برای ارزیابی و نتایج حاصل از آزمایش‌ها بیان شدند. در بخش چهارم نقاط قوت و ضعف هر یک از روش‌ها ارائه شد تا مبنایی برای مقایسه روش‌ها و امکانی برای بهبود آن‌ها شود. با تجمیع نقاط قوت و ضعف روش‌های بررسی شده، می‌توان روش‌های جدیدتری طراحی کرد که این کاستی‌ها را برطرف نمایند. همچنین، برای هر پژوهشگری که قصد دارد در این حوزه تحقیق نماید، بخش‌های دوم و سوم می‌توانند راه‌گشا بوده و مقدمه‌ی مطالعه قرار گیرند؛ علی‌الخصوص بخش سوم که آخرین نوآوری‌های مطرح را مرور می‌کند و شیوه انجام آزمایش و تحلیل نتایج را نشان می‌دهد. به‌عنوان کار آتی می‌توان چندین روش را پیاده سازی کرد و آن‌ها را روی مجموعه ویروس‌های مختلف آزمایش نمود. از این طریق می‌توان سرعت اجرای روش‌ها، کارایی آن‌ها و دقتشان را با هم مقایسه نمود.

جدول ۷. نتایج سال ۲۰۰۴ و ۲۰۰۵

نام مقاله	نویسنده و سال انتشار	نقاط قوت و ضعف
طبقه‌بندی تبدیلات یکتاسازی [32]	میشرا[1]، ۲۰۰۳	✓ تبدیل‌های کد ساده پیشنهادی نمونه‌هایی با ساختار بسیار متفاوت تولید می‌کنند. ✓ روش ساده و در عین حال مؤثر است. × تعیین مقدار آستانه در مقایسه فایل‌های سالم و ویروس دقت روش را تحت تأثیر قرار می‌دهد و ممکن است باعث افزایش هشدار مثبت اشتباه شود. × اگر از روش جابجایی قطعات استفاده شود و طول قطعه‌ها کوتاه باشند یا از روش جانشینی دستورات استفاده شود، روش دچار شکست می‌شود. × برای کشف ویروس‌های فراریخت، نیاز به برگردان به اسمبلی دارد.
اعمال ترتیب روی دستورات برنامه در جهت کمک به پویشگر ضد ویروس‌ها [33]	لاخوتیا[2] و محمد[3] ۲۰۰۴	✓ کاهش چشمگیر فضای حالت گونه‌های یک ویروس فراریخت، حجم امضاهای مورد نیاز برای نگهداری را بسیار کم می‌کند. × برای استفاده از روش پیشنهادی، کد ویروس باید به یک زبان سطح بالا برگردانده شود. × روش پیشنهادی باید برای کار روی فایل باینری ویروس‌ها توسعه داده شود. × روش پیشنهادی نیاز به تحلیل کامپایلری روی کد سطح بالای ویروس دارد. این تحلیل ممکن است سربار محاسباتی بالا داشته باشد. × بخاطر ماهیت ابتکاری و مکاشفه‌ای روش پیشنهادی، همیشه همه گونه‌های یک ویروس به یک شکل متعارف تبدیل نمی‌شود. × آزمایش روی ویروس‌های واقعی صورت نگرفته است. × همه تبدیلات جهش کد نیز بررسی نشده است.
کشف ویروس‌های فراریخت کامپیوتری مبتنی بر تفکیک‌پذیری با استفاده از راه‌کار کنترل افزونگی [34]	آندو[4] و همکاران، ۲۰۰۵	✓ در مقایسه با روش مبتنی بر نماسازی، روش پیشنهادی مبتنی بر درستی‌یابی رسمی در تشخیص ویروس‌های فراریختی که از فنون ضد ابتکاری همچون جانشینی ثبات استفاده می‌کنند، مؤثر است. × به‌خاطر پیچیدگی تحلیل غیر عملی است.

جدول ۸. نتایج سال ۲۰۰۶

نام مقاله	نویسنده و سال انتشار	نقاط قوت و ضعف
کشف بدافزار خودجهشی با استفاده از تطابق گراف جریان کنترل [35]	بروشی[5] و همکاران ۲۰۰۶	× راه‌کار ارائه شده نیاز به بهبودهایی دارد. × روی کد اسمبلی کار می‌کند و نیاز به برگرداندن و ساخت گراف جریان کنترل هم دارد. × نیاز به ارزیابی روش در نمونه‌های واقعی بیشتر دارد.
کشف ویروس‌های کامپیوتری فراریخت با استفاده از مشخصه‌های جبری [36]	وبستر[6] و گرنت[7]، ۲۰۰۶	✓ مبنای اثبات ریاضی باعث دقت بالای روش می‌شود. × در عین حال پیچیدگی روش باعث افزایش سربار اجرایی آن می‌گردد.
شکار موتورهای مولد فراریختی [17]	ونگ[8] و استمپ[9]، ۲۰۰۶	✓ آزمایش‌های انجام شده در [38] نشان داده است که کشف با مدل مخفی مارکوف حدود ۱۰٪ نرخ تشخیص مثبت اشتباه داشته است. × تحقیقاتی انجام شده است [20] که نشان داده‌اند می‌توان روش کشف مبتنی بر مدل مخفی مارکوف را شکست داد؛ مثل درج کد زائد از فایل‌های سالم داخل کد ویروس. این‌کار باعث می‌شود فایل ویروس از نظر آماری شبیه فایل‌های سالم شود. × همچنین نتایج حاصل از تحقیقات [39] نشان داده‌اند اگر بجای درج تصادفی کدهای زائد، بلوک‌های بزرگ و متوالی از کد مثل یک زیرروال از فایل سالم داخل ویروس کپی شود، کشف مبتنی بر مدل مخفی مارکوف دچار شکست می‌شود.

[1] Mishra
[2] Lakhotia
[3] Mohammed
[4] Ando
[5] Bruschi
[6] Webster
[7] Malcolm
[8] Wong
[9] Stamp

استفاده از امضای موتور برای کشف بدافزارهای فراریخت [62]	چوچن[1] و لاخوتیا[2] ۲۰۰۶	✓ × × ×	نیاز به ذخیره حجم انبوهی از امضای بدافزارهای فراریخت را مرتفع می‌سازد. به‌نظر می‌آید افزایش طول فایل و درج کد زائد، احتمال شناسایی توسط روش پیشنهادی را کم می‌کند. تمرکز فقط روی موتورهای فراریختی است که از روش مبهم‌سازی جانشینی کد بهره می‌برند. اگر سطح موتور پسندی بدافزار تولید شده کم باشد، روش پیشنهادی دیگر درست کار نمی‌کند.

جدول ۹. نتایج سال ۲۰۰۷ تا ۲۰۱۰

نام مقاله	نویسنده و سال انتشار	نقاط قوت و ضعف	
کشف ویروس‌های فراریخت با استفاده از مدل مخفی مارکوف شکلی [40]	آتالوری[3]، ۲۰۰۷	✓ ×	با توجه به نتایج و کارایی، کشف ویروس‌های فراریخت با PHMM کاملاً عملی است. آموزش و محاسبه امتیاز سریعتر از فنون ابتکاری انجام می‌شود ولی زمان مورد نیاز برای فیلتر کردن داده‌ها و برگرداندن به اسمبلی روی کارایی تأثیر منفی می‌گذارد.
شکار ویروس‌های فراریخت غیرقابل کشف [39]	لین، ۲۰۰۹	✓ ✓ ×	شبیه کردن فایل ویروس فراریخت به فایل نرمال برای جلوگیری از کشف توسط روش HMM. تولید ویروس‌های فراریخت با درجه بالا که شباهت ویروس‌های هم‌خانواده کم است. روش پیشنهادی برای درج کد مرده از دستور پرش غیرشرطی استفاده می‌کند که این کد اجرا نشود. ولی پویشگرهای ضد ویروس‌ها این‌گونه کدهای مرده را به‌راحتی کشف کرده و پیش از پویش حذف می‌کنند.
دسته‌بندی گونه‌های ویروس فراریخت با استفاده از هیستوگرام فراوانی کدهای عملیاتی [42]	راد و مَصروم[4]، ۲۰۱۰	✓ × × × ×	برای بعضی از روش‌های مبهم‌سازی عملکرد مؤثری دارد. مبهم‌سازی از طریق درج کد زائد و جانشینی دستورات روش پیشنهادی را دچار شکست می‌نمایند. آزمایش‌های صورت گرفته جهت ارزیابی بسیار محدود بوده‌اند. فقط در مواردی قابل استفاده است که روش مبهم‌سازی بر فراوانی دستورها بی تأثیر باشد یا تأثیر جزئی داشته باشد. هر گونه ویروس از فنون گوناگون مبهم‌سازی استفاده می‌کند و انتخاب سطح آستانه مناسب چالش برانگیز است و به‌شدت بر موفقیت روش تأثیر می‌گذارد.

جدول ۱۰. نتایج سال ۲۰۱۱ و ۲۰۱۲

نام مقاله	نویسنده و سال انتشار	نقاط قوت و ضعف	
کشف ویروس‌های فراریخت غیرقابل کشف [43]	ونکاتاچالام[5] و استمپ، ۲۰۱۱	✓ ×	نشان می‌دهد که با روش HMM می‌توان بالاخره ویروس‌های فراریخت تولیدی با شرایط خاص را هم کشف کرد. از نظر [44] مدل مخفی مارکوف تحلیل ایستا نیست، تعریف آن‌ها از تحلیل ایستا خیلی محدود است ولی در هر حال HMM هم روش تحلیل ایستا محسوب می‌شود. این مقاله مستقیماً نظر [44] را رد نمی‌کند ولی فقط روش HMM را برای بررسی صحت ادعای آن‌ها (مثال نقض) به‌کار برده است، که از تعریف محدود آن‌ها HMM تحلیل ایستا نیست. اما این سؤال به ذهن متبادر می‌شود که آیا تکیه به چنین تعریف محدودی، درست است؟
ویروس‌های ویژه برای تشخیص ویروس‌های فراریخت [45]	صالح[6] و همکاران، ۲۰۱۱	✓ ✓ × ×	روش پیشنهادی از روش‌های آماری مفید در تشخیص بدافزارهایی است که سایر روش‌های آماری در تشخیص آن‌ها موفق نبوده‌اند. نرخ تشخیص بالایی دارد و میزان مثبت اشتباه آن کم است. همچنین می‌تواند الگوهای جدید ویروس‌ها را جهت تشخیص بعدی بیاموزد. برای رسیدن به دقت بالا باید ویروس‌های بسیاری برای آموزش استفاده شوند که زمان و فضای محاسباتی را بالا می‌برد. اگر بخواهیم بدافزارهایی با درجه فراریختی بالا را با دقت زیاد تشخیص دهیم، نیاز به نمونه‌های بسیار داریم.
کشف بدافزار با روش مبتنی بر گراف و استفاده از	آندرسون[7] و همکاران، ۲۰۱۱	✓	این روش، در شناسایی ویروس‌ها از فایل‌های سالم بهتر از روش‌های مبتنی بر امضا و n-gram کار می‌کند.

[1] Chouchane
[2] Lakhotia
[3] Attaluri
[4] Masrom
[5] Venkatachalam
[6] Saleh
[7] Anderson

تحلیل پویا [46]		✓ کارامد در کشف بسیاری از بدافزارها از جمله چندریختی ✓ نرخ هشدارهای مثبت اشتباه کم است. مناسب شناسایی بدافزار از فایل سالم و نیز تفکیک بین انواع مختلف بدافزار ✓ توسعه روش n-gram به 2-gram برای تولید احتمالات گذر در زنجیره مارکوف ✗ تمرکز روی شناسایی ویروس‌های چند ریخت است و ویروس‌های فاریخت بررسی نشده‌اند. ✗ پیچیدگی محاسبات هسته‌ای در کاربردهای واقعی محدودیت ایجاد می‌کند، هرچند که راه‌کارهایی برای تسریع محاسبات ارائه شده‌اند.
کشف ویروس‌های کامپیوتری فاریخت با روش‌های مبتنی بر مورد [52]	برکات[1]، ۲۰۱۱	✗ اعتبار سنجی روش پیشنهادی صورت نگرفته است. ✗ روش پیشنهادی نقاط ضعف بسیار دارد.
آزمون شباهت برای کشف ویروس‌های فاریخت [53]	پاتل[2]، ۲۰۱۱	✓ در مطالعات [20] نشان داده شده است که اگر ویروس را به فایل‌های نرمال نزدیک‌تر کنیم، مدل مبتنی بر HMM در کشف ویروس‌های تغییر شکل یافته‌ای که ۵٪ کد مرده و زیروال از فایل‌های نرمال در آن‌ها درج شده، موفق نیستند. در مقایسه با روش شاخص شباهت برای کشف ویروس‌های تغییر شکل یافته، روش مذکور ویروس‌هایی را که تا ۲۵٪ کد درج شده دارند، کشف می‌کند. ✓ اگر همه ویروس‌های تغییر شکل یافته را پیش‌پردازش کنیم، روش شاخص شباهت که بهترین نتایج را نشان داده، کارایی بهتری در مقایسه با سایر روش‌های مبتنی بر شباهت خواهد داشت.
شباهت گراف کدهای عملیاتی و کشف فاریختی [51]	ران وال و همکاران، ۲۰۱۲	✓ محاسبه امتیاز ساده و کارامد است. ✗ اگر از روش جایگذاری دستورهای مشابه در فاریختی استفاده شود، یا دستورات با دقت جابجا شوند، تشخیص فاریختی دشوار می‌شود.

جدول ۱۱. نتایج سال ۲۰۱۳ و ۲۰۱۴

نام مقاله	نویسنده و سال انتشار	نقاط قوت و ضعف
کشف ویروس‌های فاریخت با استفاده از فاصله کای دو [6]	تودریچی[3] و استمپ، ۲۰۱۳	✓ روش CSD ساده‌تر است و اغلب اوقات عملکردش مشابه روش ترکیبی است. پس در عمل ارجحیت دارد. ✗ روش پیشنهادی سربار محاسباتی بیشتری دارد.
کشف گونه‌های بدافزار تغییرشکل‌یافته از طریق تخصیص موتور [57]	چوچان[4] و همکاران، ۲۰۱۳	✓ روش پیشنهادی سربار استخراج، نگهداری و توزیع امضای تک تک بدافزارها (با این حجم انبوه) را کم می‌کند. ✓ نیاز به تحلیل رفتاری همچون تحلیل جریان اجرایی یا نماسازی توسط نرم‌افزار کشف کننده را مرتفع می‌سازد. ✗ روش‌های پیشنهادی در کشف ویروس‌های رمزگذاری شده قابل به‌کارگیری نیستند هر چند در آزمایش‌ها چند بدافزار چند ریخت هم وجود داشته است. ✗ روش پیشنهادی روی کد اسمبلی کار می‌کند، هر چند که می‌توان آن‌را برای تحلیل کد باینری هم توسعه داد. در این‌صورت نیاز به آزمایش‌هایی برای ارزیابی کارایی داریم. ✗ نرم‌افزار کشف کننده باید قادر به برگرداندن به اسمبلی باشد. ✗ کارکردن روی کدهای نمادین باعث آگاهی روش‌های پیشنهادی از معنای برنامه‌ها می‌شود. در این‌صورت با لحاظ کردن دستورات یا الگوی آن‌ها می‌توان آگاهی از معنا را گسترش هم داد. ولی این‌کار هزینه محاسباتی بیشتری می‌طلبد.
فاصله جانشینی ساده و کشف ویروس‌های فاریخت [29]	شانموگام[5] و همکاران، ۲۰۱۳	✓ بخش محاسبه امتیاز، مقایسه‌ها و جابجایی‌های الگوریتم پیشنهادی کاملاً از نظر محاسباتی کارامد است. ✗ استخراج کدهای عملیاتی هزینه محاسباتی بالا دارد. ✗ آزمایش‌ها روی کدهای اسمبلی برگردان شده از باینری صورت گرفته است.
تحلیل ایستا برای کشف ویروس‌های کامپیوتری فاریخت با روش‌های ابتکاری شمارش دستورات	کانفورا[6] و همکاران، ۲۰۱۳	✓ این روش به‌سادگی در هر ضد بدافزاری قابل پیاده‌سازی است. نیاز به اجرای برنامه ندارد. ✗ روی کد باینری کار نمی‌کند و باید به اسمبلی برگردان شود. ✗ اگر از روش جانشینی کد استفاده شود، توزیع دستورات تکراری تغییر می‌کند و روش پیشنهادی دچار شکست می‌گردد.

[1] Berkat
[2] Patel
[3] Toderici
[4] Chouchane
[5] Shanmugam
[6] Canfora

تکراری [30]		
کرم فراریختنی که موتور فراریختی خودش را با خود دارد [2]	ریدهارا[1] و استمپ، ۲۰۱۳	✓ استفاده از روش درج کد زائد که برای جلوگیری از کشف توسط HMM بسیار مؤثر واقع می‌شود. ✗ درج کدهای زائد به‌طور تصادفی در محل‌های ممکن صورت گرفته و با ابزارهای جست‌وجوی کد مرده پیدا می‌شوند. ✗ خود کرم از روش‌های ساده تغییر شکل بهره می‌برد.
بدافزار فراریخت و بی‌نظمی ساختاری [4]	بیسا[2] و همکاران، ۲۰۱۳	✓ چون به‌جای کد اسمبلی، روی کد باینری فایل ویروس کار می‌کند، سربار کمتر و کارایی بیشتر دارد. ✓ بر اساس تحلیل ایستا کار می‌کند و نیاز به برگردان به اسمبلی ندارد. ✗ برای شکست روش پیشنهادی، مولد فراریخت بجای تغییر شکل ساده دستورات، باید ساختار برنامه را به‌شکل معنی‌دار تغییر دهد. ✗ اگر محدوده آستانه‌های تعیین شده در دو مرحله با هم تداخل داشته باشد، این روش ممکن است دیگر درست کار نکند. ✗ اگر مولد فراریخت، فایل‌هایی با طول‌های بسیار متغیر و متفاوت تولید کند، روش پیشنهادی ممکن است دچار شکست شود.
تحلیل مقادیر ویژه برای کشف فراریختی [58]	دشپاند[3] و همکاران، ۲۰۱۴	✓ این روش برای کدهایی که درجه فراریختی آن‌ها بالاست و از روش‌های تشخیص آماری گریخته‌اند مفید است. ✗ روش‌های فراریختی هم موجودند که این روش آماری در مقابل آن‌ها شکست می‌خورد. روش فراریختنی که بتواند با حفظ آمار دستورات، شکل کد را عوض کند، روش پیشنهادی را دچار شکست می‌نماید.

[1] Sridhara
[2] Baysa
[3] Deshpande

مراجع

[1] J. Aycock, *Computer Viruses and Malware,* Advances in Information Security, Vol. 22, Springer-Verlag, 2006.

[2] S. M. Sridhara and M. Stamp, “Metamorphic worm that carries its own morphing engine,” *J. Comput. Virol.* Vol. 9, No. 2, pp. 49-58, 2013.

[3] M. Stamp, *Information Security: Principles and Practice*. Wiley, New York, 2011.

[4] D. Baysa, R. M. Low, M. Stamp, “Structural entropy and metamorphic malware,” *J. Comput. Virol.* Vol. 9, pp. 179-192, 2013.

[5] J. Han, M. Kamber, J. Pei, *Data Mining: Concepts and Techniques* (3rd ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2011.

[6] A. H. Toderici and M. Stamp, “Chi-squared distance and metamorphic virus detection,” .*J. Comput. Virol.* Vol. 9, No. 1, pp. 1-14, 2013.

[7] N. Idika, A. P. Mathur. *A survey of malware detection techniques*. Purdue University, pp. 48, 2007.

[8] McAfee Labs. Mcafee threats report: First quarter 2013. Technical report, McAfee, 2013.

[9] Search Security. Metamorphic and polymorphic malware, [Online], Available: http://searchsecurity.techtarget.com/definition/ metamorphic-and-polymorphic-malware, 2010.

[10] M. Patel, Similarity tests for metamorphic virus detection, Master's report, Department of Computer Science, San Jose State University, http://scholarworks.sjsu.edu/etd_projects/175/., 2011.

[11] S. Madenur Sridhara, M. Stamp. “Metamorphic worm that carries its own morphing engine,” *J. Comput. Virol.* Vol. 9, No. 2, pp. 49-58, 2013.

[12] P. Szor. The Art of Computer Virus Research and Defense. Addison-Wesley Professional, 2005.

[13] K. Kaushal, P. Swadas, N. Prajapati, “Metamorphic Malware Detection Using Statistical Analysis,” In *International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307 (Online),* Vol. 2, No. 3, pp. 49-53, 2012.

[14] Symantec. Viruses, worms, and trojans, [Online], Available:http://service1.symantec.com/support/nav.nsf/docid/1999041209131106, 2011.

[15] S. Deshpande, Eigenvalue Analysis for Metamorphic Detection, Master’s Thesis, San Jose State University, 2012.

[16] G. Wagener, R. State, A. Dulaunoy. “Malware behavior analysis,” *J. Comput. Virol.* Vol. 4, No. 4, pp. 279-287, 2008.

[17] W.Wong, M. Stamp, “Hunting for metamorphic engines,” *J. Comput. Virol.* Vol. 2, No. 3, pp. 211-229, 2006.

[18] I. You, K. Yim, “Malware obfuscation techniques: a brief survey,” Broadband, Wireless Computing, Communication and Applications (BWCCA), 2010 International Conference on, pp. 297-300, 2010.

[19] P. Zbitskiy. “Code mutation techniques by means of formal grammars and automatons,” *J. Comput. Virol.* Vol. 5, pp. 199-207, 2009.

[20] D. Lin, M. Stamp, “Hunting for undetectable metamorphic viruses,” *J. Comput. Virol.* Vol. 7, No.3, pp. 201-214, 2011.

[21] W. Wong, “Analysis and detection of metamorphic computer viruses,” Department of Computer Science, San Jose State University, May, Master’s Thesis, 2006.

[22] M. Patel, Similarity tests for metamorphic virus detection. Master’s report, Department of Computer Science, San Jose State University, 2011.

[23] S. Venkatachalam, M. Stamp, “Detecting undetectable metamorphic viruses,” In *Proceedings of 2011 International Conference on Security & Management (SAM’11),* pp. 340-345, 2011.

[24] E. Konstantinou, Metamorphic virus: Analysis and detection. Technical Report, *Royal Holloway University of London*, 2008.

[25] M. Saleh, "Towards Metamorphic Virus Recognition Using Eigenviruses," *arXiv preprint arXiv,* pp. 1206.5871, 2012.

[26] D. N. Kumar, et al. "THE NEW AGE OF COMPUTER VIRUS AND THEIR DETECTION," *International Journal of Network Security & Its Applications* Vol. 4, No. 3, 2012.

[27] F. Cohen, "Computer viruses-Theory and experiments," *Computers and Security,* Vol. 6, No. 1, pp: 22-35, 1984.

[28] D. M. Chess, S. R. White, "An undetectable computer virus," In *Virus* Bulletin Conference, 2000.

[29] G. Shanmugam, R. M. Low, M. Stamp, “Simple substitution distance and metamorphic detection,” *J. Comput. Virol.* Vol. 9, No. 3, pp. 159-170, 2013.

[30] G. Canfora, A. N, Lannaccone, C. A. Visaggio, “Static analysis for the detection of metamorphic computer viruses using repeated-instructions counting heuristics,” *J. Comput. Virol.* Vol. 10, No. 1, pp. 11-27, 2014.

[31] B. Bashari Rad, M. Masrom, “Metamorphic virus variants classification using opcode frequency histogram,” In *Proceedings of the 14th WSEAS international conference on Computers: part of the 14th WSEAS CSCC multiconference - Volume I* (ICCOMP'10), Nikos E. Mastorakis, Valeri Mladenov, and Zoran Bojkovic (Eds.), Vol. I. World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, pp. 147-155, 2010.

[32] P. Mishra, “Taxonomy of uniqueness transformations,” http://www.cs.sjsu.edu/faculty/stamp/students/FinalReport.doc, 2003.

[33] A. Lakhotia, M. Mohammed, “Imposing Order on Program Statements to Assist Anti-Virus Scanners,” In *Proceedings of the 11th Working Conference on Reverse Engineering* (WCRE '04). IEEE Computer Society, Washington, DC, USA, pp. 161-170, 2004.

[34] R. ANDO, A. Q. NGUYEN, T. YOSHIYASU, "Resolution based computer metamorphic virus detection using redundancy control strategy," Proceedings of the 4th WSEAS Int. Conf. on Information Security, Communications and Computers, Tenerife, Spain, December 16-18, pp. 265-270, 2005.

[35] D. Bruschi, L. Martignoni, M. Monga, “Detecting self-mutating malware using control-flow graph matching,” In *Proceedings of the Third international conference on Detection of Intrusions and Malware & Vulnerability Assessment* (DIMVA'06), Roland Büschkes and Pavel Laskov (Eds.). Springer-Verlag, Berlin, Heidelberg, pp. 129-143, 2006.

[36] M. Webster, M. Grant, "Detection of metamorphic computer viruses using algebraic specification," *Journal in Computer Virology* Vol. 2, No. 3, pp. 149-161, 2006.

[37] IDA Pro Disassembler, [Online], Available: www.datarescue.com/idabase/.

[38] W. Wong. “Analysis and detection of metamorphic computer viruses,” Master's Projects. Paper 153. http://scholarworks.sjsu.edu/etd_projects/153, 2006.

via engine attribution," *J. Comput. Virol.* Vol. 9, No. 3, pp. 137-157, 2013.
[58] S. Deshpande, Y. Park, M. Stamp. "Eigenvalue analysis for metamorphic detection," *Journal of computer virology and hacking techniques,* Vol. 10, No. 1, pp. 53-65, 2014.
[59] E. Daoud, I. Jebril, "Computer virus strategies and detection methods," Int. J. Open Probl. Comput. Sci. Math. Vol. 1, No. 2, pp. 29–36, 2006.
[60] I. Sorokin. "Comparing files using structural entropy," *Journal in Computer Virology*, Vol. 7, No. 4, pp. 259-265, 2011.
[61] Mark Stamp Website in San Jose State University, [Online], http://cs.sjsu.edu/~stamp/viruses/
[62] M. R. Chouchane, A. Lakhotia, "Using engine signature to detect metamorphic malware," In *Proceedings of the 4th ACM workshop on Recurring malcode* (WORM '06). ACM, New York, NY, USA, pp. 73-78, 2006.

علیرضا خلیلیان دانشجوی دکتری نرم‌افزار ورودی سال ۱۳۹۱ در دانشگاه اصفهان است. وی کارشناسی ارشد خود را در گرایش نرم‌افزار از دانشگاه علم و صنعت ایران در سال ۱۳۸۸ دریافت کرده است. زمینه‌های تحقیقاتی او آزمون و اشکال‌زدایی نرم‌افزار، مهندسی نرم‌افزار مدل رانده، امنیت سیستم‌های نرم‌افزاری و داده کاوی است.

احمد براآنی دکتری خود را در رشته مهندسی کامپیوتر از دانشگاه ولنگنگ استرالیا در سال ۱۳۷۵ اخذ کرده است. او هم اکنون دانشیار مهندسی کامپیوتر در دانشگاه اصفهان است. زمینه‌های تحقیقاتی ایشان داده کاوی، امنیت سیستم‌ها و آزمون نرم‌افزار است.

[39] D. Lin. Hunting for undetectable metamorphic viruses, Master's Projects. Paper 18. http://scholarworks.sjsu.edu/etd_projects/18, 2009.
[40] S. Attaluri, "Detecting metamorphic viruses using profile hidden markov models," Diss. San Jose State University, 2007.
[41] P. Desai, "Towards an Undetectable Computer Virus," Master's thesis, San Jose State University, http://www.cs.sjsu.edu/faculty/stamp/students/Desai_Priti.pdf, 2008.
[42] B. Bashari Rad, M. Masrom, "Metamorphic virus variants classification using opcode frequency histogram," In *Proceedings of the 14th WSEAS international conference on Computers: part of the 14th WSEAS CSCC multiconference - Volume I* (ICCOMP'10), Nikos E. Mastorakis, Valeri Mladenov, and Zoran Bojkovic (Eds.), Vol. I. World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, pp. 147-155, 2010.
[43] S. Venkatachalam, M. Stamp, "Detecting undetectable metamorphic viruses," *Proceedings of 2011 International Conference on Security & Management (SAM'11)*. pp. 340-345, 2011.
[44] J. Borello, L. Me, "Code obfuscation techniques for metamorphic viruses," *Journal in Computer Virology*, Vol. 4, No. 3, pp. 211–220, 2008.
[45] M. E. Saleh, A. B. Mohamed, A. A. Nabi, "Eigenviruses for metamorphic virus recognition," *IET information security* Vol. 5, No. 4, pp. 191-198, 2011.
[46] B. Anderson, D. Quist, J. Neil, C. Storlie, T. Lane, "Graph-based malware detection using dynamic analysis," *J. Comput. Virol.* Vol. 7, No. 4, pp. 247-258, 2011.
[47] J. Dai, R. Guha, J. Lee, "Efficient virus detection using dynamic instruction sequences," *J. Comput*. Vol. 4, No. 5, pp. 405–414, 2009.
[48] D. Reddy, S. Dash, A. Pujari, "New malicious code detection using variable length *n*-grams," In: Information Systems Security. Lecture Notes in Computer Science, Vol. 4332, pp. 276–288, 2006.
[49] D. Reddy, A. Pujari, "N-gram analysis for computer virus detection," *J. Comput. Virol.* Vol. 2, pp. 231–239, 2006.
[50] S. Stolfo, K. Wang, W. J. Li, "Towards stealthy malware detection. In: Malware Detection," Advances in Information Security, Vol. 27, pp. 231–249, 2007.
[51] N. Runwal, R. M. Low, M Stamp, "Opcode graph similarity and metamorphic detection," *J. Comput. Virol.* Vol. 8, No. 1-2, pp. 37-52, 2012.
[52] A. Berkat, "Metamorphic computer virus detection by Case-Based Reasoning (CBR) methods," *International Journal of Software Engineering & Applications* Vol. 2, No. 4, 2011.
[53] M. Patel, "Similarity Tests for Metamorphic Virus Detection," Diss. San Jose State University, 2011.
[54] S. McGhee, "Pairwise Alignment of Metamorphic Computer Viruses," Master' s Thesis, San Jose State University, 2007.
[55] N. Runwal, "Graph technique for metamorphic virus detection," Master's report, Department of Computer Science, San JoseState University. http://www.cs.sjsu.edu/faculty/stamp/students/runwal_neha.pdf, 2011.
[56] E. Filiol, S. Josse, "A statistical model for undecidable viral detection," *J. Comput. Virol.* Vol. 3, No. 1, pp. 65–74, 2007.
[57] R. Chouchane, N. Stakhanova, A. Walenstein, A. Lakhotia, "Detecting machine-morphed malware variants

An Investigation and Comparison of Metamorphic Virus Detection and Current Challenges

Alireza Khalilian and Ahmad Baraani
School of Computer Engineering, University of Isfahan
Isfahan, Iran
{khalilian, ahmadb}@eng.ui.ac.ir

***Abstract*—**Virus writers employ different approaches to evade from detection, with metamorphism to be one of the most effective ones. To achieve this, code obfuscation techniques are used to change the structure of the code while retaining its current functionality. In recent years, various techniques have been proposed in the literature to statically detect the metamorphic viruses. Meanwhile, virus writers employ more complex and newer methods of code obfuscation to overcome the existing methods which in turn make the detection of metamorphic malwares a challenging task. Moreover, any new method of code obfuscation, the different ways they are applied, and the level of granularity have major impacts on the detection of metamorphic viruses. This paper investigates 23 of the latest studies for static detection of metamorphic malwares. For each method, the proposed approach, the experiments and the obtained results have been discussed in detail. Then, the strengths and weaknesses of each method were presented and compared. The results of this research work make room for further research in the context of metamorphic detection.

***Keywords*-**Malware, Virus, Metamorphism, Code Obfuscation.