کشف ویروس‌های فراریخت: دستاوردها و چالش‌ها

علیرضا خلیلیان
دانشجوی دکتری نرم‌افزار، دانشگاه اصفهان و مدرس دانشکده فنی شریعتی، دانشگاه فنی حرفه‌ای، تهران
پست الکترونیک: khalilian@eng.ui.ac.ir

نوشین وزین‌دل
دانشجوی کارشناسی ارشد نرم‌افزار، دانشگاه آزاد اسلامی واحد تهران شمال
پست الکترونیک:vazindel.nooshin@gmail.com

چکیده

فراریختی یکی از روش‌هایی است که نویسندگان ویروس‌ها و بدافزارهای کامپیوتری به‌کمک آن مانع کشف ویروس شده یا کشف ویروس را دشوار می‌سازند. فراریختی از روش‌های مبهم‌سازی کد برای تغییر ظاهر کد ویروس با حفظ عملکرد اصلی آن بهره می‌برد. برای پی بردن به نقاط قوت و ضعف روش‌های موجود در کشف فراریختی، 37 روش مطرح در کشف ویروس‌های فراریخت بین سال‌های 2003 تا 2014 مورد بررسی قرار گرفته‌اند و نتایج آن‌ها در این مقاله گزارش می‌شود. بررسی ویژگی‌های مثبت و منفی این روش‌ها زمینه‌ساز پژوهش‌های آتی در کشف ویروس‌های فراریخت خواهد بود تا بتوان روش‌های مؤثرتر و کارامدتری ابداع کرد.
واژه‌های کلیدی: بدافزار، ویروس، فراریخت، مبهم‌سازی کد.

1- مقدمه

امروزه بدافزارها با نرخ بالا رشد می‌کنند و هر روز چندین هزار از انواع آن‌ها پخش می‌شود (Aycock, 2006). 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 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 (Aycock, 2006). مشکلات موجود در روش‌های پویا باعث شده که هنوز هم روش‌های ایستا محبوبیت فراوانی داشته باشند.

یکی از انواع بدافزارها، ویروس‌ها یا کرم‌های فراریخت هستند (Aycock, 2006). 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 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 (Toderici, 2013) 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 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 تحقیقی

روش‌های موجود فراریختی را مورد بررسی قرار دهیم. برای نیل به این هدف، 37 مورد از روش‌های مطرح پیرامون کشف بدافزارها و اختصاصاً ویروس‌های فراریخت مورد مطالعه قرار گرفت. در این مقاله، نقاط قوت و ضعف هر یک از این روش‌ها ارائه می‌گردد. حاصل مطالعه‌های صورت گرفته در پژوهش حاضر، به پرسش‌های زیر پاسخ می‌دهد: (1) نقاط ضعف ویروس‌های فراریخت فعلی چیست؟ (2) چه نقطه ضعف‌هایی در روش‌های کنونی کشف ویروس‌های فراریخت وجود دارد؟

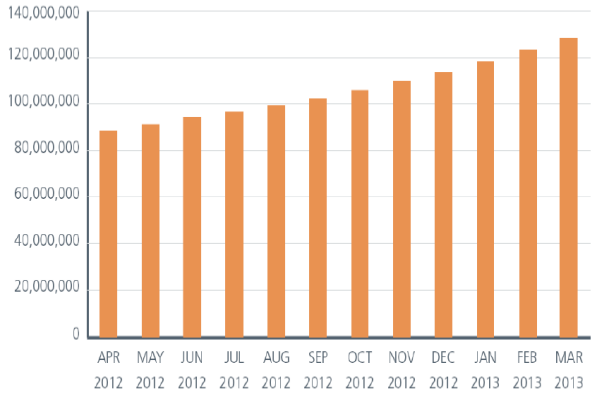
ساختار ادامه مقاله به‌شرح زیر است: در بخش دوم مقاله، ویروس و کرم، ویروس‌های فراریخت، روش‌های مبهم‌سازی کد و ضعف فراریختی با جزئیات بیان می‌گردد. بخش سوم، نقاط قوت و ضعف روش‌های مورد مطالعه را ارائه می‌نماید. بخش چهارم هم به نتیجه‌گیری اختصاص دارد.

2- مروری بر ادبیات موضوع

در این بخش تمام مفاهیم مورد نیاز به تفصیل مورد بحث قرار می‌گیرد. در ابتدا ساختار و نحوه عملکرد ویروس و کرم که دو بدافزار مشهور هستند مورد بررسی قرار می‌گیرد. در ادامه پیرامون ویروس‌های فراریخت، تعریف رسمی آن‌ها، روش‌های مبهم‌سازی کد جهت تولید بدافزار فراریخت و نقطه ضعف آن‌ها بحث می‌شود. در نهایت، بخش دوم با کشف کشف ویروس‌های فراریخت پایان می‌یابد.

2-1- ویروس و کرم

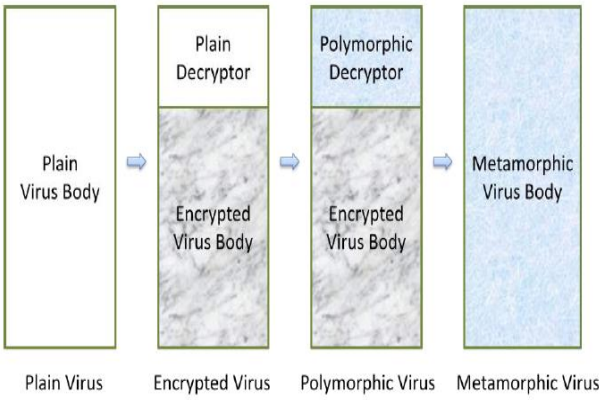
بدافزار برنامه کامپیوتری است که هدف آن آسیب زدن به سیستم‌های کامپیوتری است (Baysa, 2013). در سال‌های گذشته تعداد بدافزارها با سرعت نمایی رو به افزایش بوده است. شکل 1 رشد بدافزارهای ثبت شده در پایگاه داده شرکت مک‌آفی را در مدت بین آوریل 2012 تا مارس 2013 نشان می‌دهد.



شکل 1: تعداد نمونه‌های موجود در پایگاه داده آزمایشگاه مک‌آفی (McAfee Labs, 2013)

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 (Sridhara, 2013). 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 (Baysa, 2013). لذا می‌توان گفت ویروس‌ها برنامه‌های خود تکثیری هستند که باعث خرابی در داخل سیستم میزبان می‌شوند (Madenur Sridhara, 2013). در مقایسه با ویروس‌ها، کرم‌ها بدافزارهای مستقلی هستند و معمولاً تحت شبکه منتقل می‌شوند (Baysa, 2013). 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 در سیستم هدف وجود داشته باشد (Aycock, 2006). 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 کار می‌کنند، نتوانند آن‌ها را شناسایی نمایند (Baysa, 2013). هدف از مبهم‌سازی این‌است که هر بار که بدافزار می‌خواهد منتشر شود و خودش را تکثیر نماید، نسخه یا شکل دیگری از خودش بسازد به‌طوریکه پویشگرهای امضای ضد بدافزارها نتوانند آن‌ها را شناسایی کنند یا شناسایی آن‌ها دشوار شود. در حوزه بدافزارها، روش‌هایی مثل رمزگذاری، نیمه چند ریختی، چند ریختی و فراریختی 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 (Aycock, 2006): 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 2 تکامل ویروس‌ها را نشان می‌دهد.



شکل 2: تکامل ویروس، به‌ترتیب از چپ به راست: ویروس معمولی، ویروس رمزگذاری شده، ویروس چندریختی، ویروس فراریختی (Saleh, 2012)

2-2- ویروس‌های فراریخت

فراریختی فرایند تبدیل بخشی از نرم‌افزار به نمونه‌های منحصر به‌فرد است (Stamp, 2011). 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 (Sridhara, 2013). 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 (Baysa, 2013). به این ویروس‌ها، فراریخت یا دگردیس شده می‌گویند. بخش کلیدی هر ویروس فراریخت موتور جهش آن است (Sridhara, 2013). 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 شده باشد (Sridhara, 2013). 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 (Sridhara, 2013). 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 (Baysa, 2013) و یک ویروس فراریخت شانس بالایی برای مخفی ماندن دارد.

2-2-1- روش‌های مبهم‌سازی ویروس‌های فراریخت

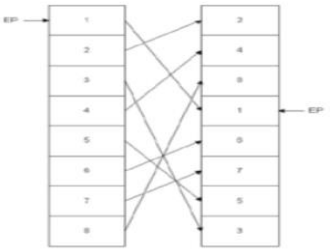
برای این‌که ویروس‌ها را فراریخت کنیم، یا نسخه جهش یافته‌ای از کد ویروس تولید شود، راه‌کارهای متعدد مبهم‌سازی (Szor, 2005) پیشنهاد شده‌اند. مبهم‌سازی هم در بخش داده‌ای و هم در بخش جریان کنترلی صورت می‌گیرد (Patel, 2011). مبهم‌سازی جریان کنترل با جابجایی دستورات و استفاده از دستورات پرش غیرشرطی صورت می‌گیرد. مبهم‌سازی جریان داده‌ای هم با روش‌هایی همچون تعویض دستورات، درج کد اضافی، جایگزینی دستورات معادل، تعویض ثبات‌ها و جایگشت زیرروال‌ها تحقق می‌یابد. تعویض ثبات‌ها 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 است (Patel, 2011). راه‌کار دیگر جانشینی دستورات 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. گاهی اوقات برای مبهم‌سازی می‌توان ترتیب دستورات را عوض کرد به‌شرطیکه هیچ وابستگی بین آن‌ها وجود نداشته باشد. این‌کار باید عملکرد دستورات را نیز عوض نکند. به این راه‌کار تعویض 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 (Madenur Sridhara, 2013). جدول 1 نمونه‌ای از این راه‌کار را نشان می‌دهد.

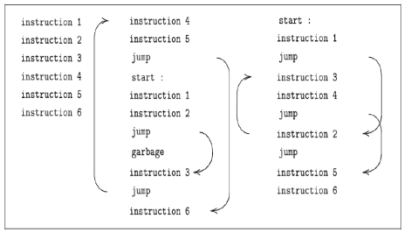
جدول 1: جابجایی دستورات (Baysa, 2013)

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

جایگشت زیرروال‌ها 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 10 زیرروال دارد، مثل ویروس Win32/Ghost می‌تواند !10 یا 3628800 گونه مختلف از خودش تولید کند (You, 2010). اما با استفاده از عبارات منظم و الگوی رشته‌های کوتاه این نوع ویروس‌ها قابل کشف هستند زیرا زیرروال‌ها تغییر نمی‌کنند و این راه‌کار به‌تنهایی مؤثر نیست. نمونه‌ای از جایگشت زیرروال‌ها در شکل 3 دیده می‌شود. 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. با استفاده از دستورهای پرش مناسب در انتهای هر قطعه می‌توان ترتیب منطقی درست برنامه را حفظ کرد. در شکل 4 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. درج دستورات بیهوده، زائد یا زباله روش مؤثری برای تولید ویروس‌های فراریخت 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» و دستورات حالت دوم را «کد مرده» 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 5 دیده می‌شود. دستورات هیچ‌کاره به‌صورت فیزیکی بخشی از برنامه هستند و نه به‌صورت منطقی. 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 فراریخت می‌کنند (Sridhara, 2013). این روش و روش قبلی استفاده بسیاری در تولید ویروس‌های فراریخت دارند. اضافه کردن کد مرده از فایل سالم داخل فایل بدافزار از روش‌های مؤثر در جلوگیری از کشف بدافزارهای فراریخت است.



شکل 3: جایگشت زیرروال (Sridhara, 2013)

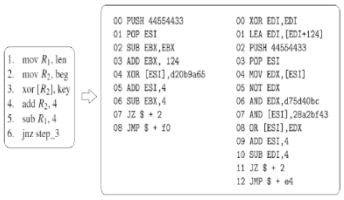


شکل 4: جابجایی قطعات برنامه (Deshpande, 2012)

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 5: دستورات هیچ‌کاره (بخش وسط)

روش مطرح دیگر برای مبهم سازی، جهش گرامر رسمی است. این روش در حقیقت گونه رسمی بسیاری از روش‌های تغییر شکل موجود است. یک موتور تغییر شکل کلاسیک مثل یک آتوماتای غیر قطعی است زیرا گذر از هر نماد به نماد دیگر امکان‌پذیر است (Zbitskiy, 2009). 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 6 قالب یک رمزگشای ویروس چند ریختی را نشان می‌دهد و به‌کمک گرامر رسمی نشان داده شده در شکل 7 دو نسخه متفاوت (جهش یافته) از آن تولید شده است. با به‌کارگیری ترکیبات مختلف گرامر رسمی شکل 6 و اعمال روی موتور رمزگشای شکل 7 می‌توان 960 رمزگشای مختلف تولید کرد (Zbitskiy, 2009).



شکل 6: یک رمزگشای چندریختی ساده و دو گونه مختلف آن (Zbitskiy, 2009)

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

شکل 7: گرامر رسمی برای تولید نسخه جهشی رمزگشا (Zbitskiy, 2009)

جدول 2 دشواری کشف برخی از روش‌های مبهم‌سازی ویروس‌های فراریختی را نشان می‌دهد. جدول 3 فنون مبهم‌سازی کـه در چنـد ویروس فراریخت تولید شده توسط هکرها پیاده‌سازی شده‌اند را خلاصه کرده است.

جدول 2: مقایسه سطح دشواری روش‌های مبهم‌سازی (Saleh, 2012)

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

جدول 3: مقایسه روش‌های مبهم‌سازی کد در بدافزارها (Venkatachalam, 2011)

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

2-2-2- نقطه ضعف ویروس‌های فراریخت

ویروس‌های فراریخت هنگام انتشار، کد خودشان را به گونه دیگری با همان عملکرد تبدیل می‌کننـد تـا پویشـگرهای مبتنـی بـر امضـا نتوانند آن‌ها را شناسایی نمایند و برای این‌کار از روش‌های مبهم‌سازی کد بهره می‌برند. ضمن اینکه این ویروس‌ها با تحلیلگرهـای پویـا مثل نماسازها هم می‌توانند مقابله کنند. وقتی که متوجه می‌شوند که در حال اجرا در یک محیط مجازی یا کنترل شده هستند، رفتـار خود را تغییر می‌دهند. چون سازندگان ویروس‌های فراریخت از نقطه ضعف پویشگرهای ضد ویروس‌ها با خبر هستند، کشـف ایـن نـوع ویروس‌ها دشوار می‌گردد. محدودیت‌هایی که در روش‌های تحلیل ایستا و پویا وجود دارند موجب بروز محـدودیت‌هایی در پویشـگرهای ضد ویروس‌ها می‌گردند. نشان داده شده است (Konstantinou, 2008) که پاشنه آشیل یک ویروس فراریخت، نیازش به تحلیـل کـد خودش است. پویشگر ضد ویروس باید بتواند ویروس فراریخت را با روشی مشابه روش به‌کار رفتـه توسـط خـود ویـروس بـرای تحلیـل خودش، تحلیل کند. در این صورت ضد ویروس نیاز به یک تغییر دهنده معکوس دارد که با اعمال قوانین تبدیل در جهـت عکـس، کـد اصلی ویروس را بازیابی نماید. مشکل اینجاست که باید حداقل یک نمونه از ویروس در اختیار تحلیلگران باشد تا بتوانند قوانین تبـدیل، مفروضات و الگوریتم‌ها را استخراج نمایند.

2-3- کشف ویروس‌های فراریخت

کشف ویروس‌های فراریخت دشوار است، ولی به‌همین نسبت تولید ویروس فراریخت هـم بـرای مهـاجمین دشـوار اسـت (Toderici, 2013). یک مشکل در تولید ویروس فرایخت این‌است که ویروس باید به اندازه کافی از نظر ساختار جهش یافته و دگردیس شـود و در عین حال اندازه‌اش در حد قابل کنترلی نگه داشته شود. مشکل دیگر این است کـه ویروس‌هـای فراریخـت بایـد تـا حـد امکـان شـبیه فایل‌های سالم باشند تا روش‌های مبتنی بر شباهت و ابتکاری قادر به کشف آن‌ها نباشند.

روش‌هایی که برای کشف ویروس‌های فراریخت پیشـنهاد شـده‌اند را می‌تـوان بـه دو دسـته تقسـیم کـرد (Saleh, 2012): دسـته اول روش‌هایی هستند که در محصولات تجاری پیاده‌سازی شده‌اند و تعدادشان محدود است. دسته دوم روش‌های بسـیار متعـددی هسـتند که آزمایشگاهی بوده و هنوز کاربردی نشده‌اند. روش‌های دسته اول از جهات مختلف مورد آزمایش قرار گرفته‌انـد. از علت‌هـای تجـاری

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 فراریخت، نرخ بالای هشدارهای مثبت اشتباه یا غیر عملی بودن روش از نظر فضا و زمان محاسباتی.

تحقیقات (Wong, 2006) نشان داده است که مدل مخفی مارکوف در کشف ویروس‌های فراریخت تولید شده توسط هکرها می‌تواند مؤثر واقع شود. برای جلوگیری از کشف ویروس‌های فراریخت، نه تنها درجه فراریختی آن‌ها باید بالا باشد، بلکه باید تا حدودی شبیه فایل‌های سالم هم باشند. ساخت ویروس‌هایی که هر دو راه‌کار در آن‌ها اعمال شده باشد، معمولاً دشوار است. فراریختی علاوه بر تولید بدافزار برای دفاع هم به‌کار می‌رود. مثلاً برای جلوگیری از حمله سرریز بافر، درجه کمی فراریختی، بسیار مؤثر واقع می‌شود.

3- نقاط قوت و ضعف روش‌های مورد مطالعه

در این بخش نقاط قوت و ضعف روش‌های مورد مطالعه 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 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 4 الی 9 به‌ترتیب نتایج سال‌های 2003 تا 2005، سال 2006 و 2007، سال 2008 و 2009، سال 2010، سال 2011 و 2012 و سال 2013 و 2014 را نشان می‌دهند.

جدول 4: مقالات سال‌های 2003 تا 2005

مشخصه رویکرد	نقاط قوت و ضعف
طبقه‌بندی تبدیلات یکتاسازی (Mishra, 2003)	➤ تبدیل‌های کد ساده و تولید پیشنهادی نمونه‌هایی با ساختار بسیار متفاوت ➤ روش ساده و در عین حال مؤثر ✖ امکان افزایش هشدار مثبت اشتباه به دلیل تاثیرتعیین مقدار آستانه در مقایسه فایل‌های سالم و ویروسی بر دقت روش ✖ شکست روش در صورت استفاده از قطعه‌های با طول کوتاه در روش جابجایی قطعات یا استفاده از روش جانشینی دستورات ✖ نیاز به برگردان به اسمبلی برای کشف ویروس‌های فراریخت
اعمال ترتیب روی دستورات برنامه در جهت کمک به پویشگر ضد ویروس‌ها (Lakhotia, 2004)	➤ کاهش حجم امضاهای مورد نیاز برای نگهداری باعث کاهش چشمگیر فضای حالت گونه‌های یک ویروس فراریخت ✖ برگرداندن کد ویروس به یک زبان سطح بالا برای استفاده از روش پیشنهادی ✖ نیاز به توسعه روش پیشنهادی برای کار روی فایل باینری ویروس‌ها ✖ نیاز به تحلیل کامپایلری با سربار محاسباتی بالا روی کد سطح بالای ویروس ✖ تبدیل نشدن همیشگی تمام گونه‌های یک ویروس به یک شکل متعارف به دلیل ماهیت ابتکاری و اکتشافی روش ✖ آزمایش نکردن روی ویروس‌های واقعی ✖ بررسی نکردن همه تبدیلات جهش کد
کشف ویروس‌های فراریخت کامپیوتری مبتنی بر تفکیک‌پذیری با استفاده از راه‌کار کنترل افزونگی (ANDO, 2005)	➤ در مقایسه با روش مبتنی بر نماسازی، موثر بودن روش پیشنهادی مبتنی بر درستی‌یابی رسمی در تشخیص ویروس‌های فراریختی که از فنون ضد ابتکاری همچون جانشینی ثبات استفاده می‌کنند . ✖ غیر عملی به‌خاطر پیچیدگی تحلیل
نرم افزار فراریخت برای کاهش سرریز بافر (Gao,2005)	➤ موثر در کاهش شدت حمله سرریز بافر ➤ روش استفاده از نرم افزار فراریخت مورد بحث در این مقاله باعث بهبود امنیت نرم افزار می شود. ✖ این روش مشکل است چون نیاز به فراریختی‌هایی دارد که دارای کارکرد مشابه باشند و هر نسخه دارای یک ساختار داخلی متفاوت باشد .

جدول 5: مقالات سال 2006 و 2007

مشخصه رویکرد	نقاط قوت و ضعف
کشف بدافزار خودجهشی با استفاده از تطابق گراف جریان کنترل (Bruschi, 2006)	✖ کار روی کد اسمبلی و نیاز به برگرداندن و ساخت گراف جریان کنترل ✖ نیاز به بهبودهایی برای راه‌کار ارائه شده ✖ نیاز به ارزیابی روش در نمونه‌های واقعی بیشتر
کشف ویروس‌های کامپیوتری فراریخت با استفاده از مشخصه‌های جبری (Webster, 2006)	➤ دقت بالای روش به دلیل مبنای اثبات ریاضی ✖ افزایش سربار اجرایی به دلیل پیچیدگی روش
شکار موتورهای مولد فراریختی (Wong, 2006)	➤ حدود 10٪ نرخ تشخیص مثبت اشتباه کشف با مدل مخفی مارکوف ✖ می‌توان روش کشف مبتنی بر مدل مخفی مارکوف را شکست داد؛ مثل درج کد زائد از فایل‌های سالم داخل کد ویروس. این‌کار باعث شباهت فایل ویروس از نظر آماری به فایل‌های سالم ✖ شف مبتنی بر مدل مخفی مارکوف دچار شکست می‌شود.اگر بجای درج تصادفی کدهای زائد، بلوک‌های بزرگ و متوالی از کد مثل یک زیرروال از فایل سالم داخل ویروس کپی شود.
استفاده از امضای موتور برای کشف بدافزارهای فراریخت (Chouchane, 2006)	✖ نیاز به ذخیره حجم انبوهی از امضای بدافزارهای فراریخت ✖ احتمال شناسایی کم روش پیشنهادی با افزایش طول فایل و درج کد زائد، ✖ تمرکز فقط روی موتورهای فراریختی است که از روش مبهم‌سازی جانشینی کد بهره می‌برند ✖ روش پیشنهادی دیگر درست کار نمی‌کند اگر سطح موتور پسندی بدافزار تولید شده کم باشد
کشف ویروس‌های فراریخت با استفاده از مدل مخفی مارکوف شکلی (Attaluri, 2007)	➤ با توجه به نتایج و کارایی، کشف ویروس‌های فراریخت با روش پیشنهادی این مقاله 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.
ترازبندی جفتی ویروس‌های کامپیوتری فراریخت (2007 McGhee,)	➤ روشی خوب برای شناسایی ویروس ✖ روش پیشنهادی در این پروژه تنها در تشخیص برخی از انواع ویروس‌ها موفقیت آمیز است . ✖ از این روش فقط برای شناسایی ویروس‌های فراریخت استفاده شده است و در مورد ویروس‌های پیشرفته تر فراریخت کاری انجام نشده است. ✖ این روش تنها به صورت نظری کاربرد دارد و می تواند ویروس را شناسایی کند.

جدول 6: مقالات سال 2008 و 2009

مشخصه رویکرد	نقاط قوت و ضعف
فنون مبهم سازی کد برای ویروس‌های فراریخت (2008 Borello,)	➤ طراحی این روش کارآمد است و به منظور جلوگیری از تجزیه و تحلیل ایستا طراحی شده است تا بر ویروس فراریخت اعمال شود. ✖ روش مبهم سازی ارائه شده برای ویروس‌های موجود نمی تواند استفاده شود. ✖ تمرکز این پروژه تنها بر روی مبهم سازی ویروس‌های فراریخت است.
رسمی سازی توسعه یافته مبتنی بر بازگشت برای جهش ویروس (Beaucamps, 2009)	➤ بررسی ویروس‌ها در چارچوب نظریه بازگشت و استفاده از دستور زبان رسمی باعث شناسایی دقیق تر مکانیسم‌های تولید ویروس می شود. ➤ موثرتر در جهش ویروس ✖ روش پیشنهادی بار محاسباتی بالایی داشته و برای عمومی سازی چندان مناسب نیست.
تشخیص عملی ویروس‌های کامپیوتری فراریخت (Govindaraj, 2008)	➤ استفاده از این روش باعث کاهش زمان کل اجرا ، بهبود بازده کلی و حذف فرایند دستی در فاز جداسازی قطعات می شود. ✖ این روش فقط برای استخراج قطعه کد اجرایی است و قطعه داده را شامل نمی شود. ✖ در این روش بر روی فایل‌های اسمبلی کاری انجام نشده است.
مبهم سازی کد و تشخیص ویروس (Venkatesan, 2008)	➤ این روش، ویروس‌هایی را که غیر قابل کشف بودند با استفاده از اسکن مبتنی بر امضا کشف می کند. ✖ این روش در طراحی یک موتور ویروس فراریختی که با هر دو روش تشخیص مبتنی بر امضاء و تشخیص بر اساس مدل مخفی مارکوف قابل تشخیص نباشد، کارا نیست.

تشخیص بدافزار فراریخت و مبتنی بر مجازی‌سازی با استفاده از مشخصات جبری (Webster, 2009)	➤ ویروس‌های کامپیوتری فراریخت می توانند کد نیمه هم ارزی جایگزین را به منظور تولید انواع شکل برای فرار از تشخیص مبتنی بر امضاء استفاده کنند. ➤ برای توسعه تشخیص بدافزار فراریخت و مبتنی بر مجازی سازی از روش‌های تجزیه و تحلیل، که به طور رسمی قابل اثبات است استفاده می شود. ✖ اجرای برنامه‌ها به منظور تشخیص هر وقوع قرص قرمز توسط تجزیه و تحلیل پویا می تواند از نظر محاسباتی سنگین باشد، و سربار اضافه کند. ✖ تشخیص مبتنی بر مجازی سازی شده همیشه تضمین شده نیست.
شکار ویروس‌های فراریخت غیرقابل کشف (Lin, 2009)	➤ شبیه کردن فایل ویروس فراریخت به فایل نرمال برای جلوگیری از کشف توسط روش مدل مخفی مارکوف ✖ 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. ✖ تولید ویروس‌های فراریخت با درجه بالا با شباهت کم ویروس‌های هم‌خانواده
مدل مخفی مارکوف شکلی و تشخیص ویروس‌های فراریخت (Attaluri, 2009)	➤ مدل‌های این مقاله با استفاده از کل توالی کدهای عملیاتی هر ویروس آموزش داده شده است. این می تواند به مدل هر زیرروال به طور مستقل اصلاح شود که ممکن است باعث تشخیص بهتر ویروس‌های فراریختی شود . ✖ زمان پیش پردازش داده‌ها گرفته شده است(به عنوان مثال، پیاده کردن کد)، که باعث می شود این رویکرد تا حدودی غیر ممکن باشد.

جدول 7: مقالات سال 2010

مشخصه رویکرد	نقاط قوت و ضعف
دسته‌بندی گونه‌های ویروس فراریخت با استفاده از هیستوگرام فراوانی کدهای عملیاتی (Bashari Rad, 2010)	➤ هر گونه ویروس از فنون گوناگون مبهم‌سازی استفاده می‌کند و موفقیت روش تحت تاثیرانتخاب سطح آستانه مناسب ✖ آزمایش‌های صورت گرفته بسیار محدود جهت ارزیابی ✖ برای بعضی از روش‌های مبهم‌سازی عملکرد مؤثری دارد؛ روش پیشنهادی دچار شکست می شود اگر مبهم‌سازی از طریق درج کد زائد و جانشینی دستورات باشد ✖ فقط در مواردی قابل استفاده است که روش مبهم‌سازی بر فراوانی دستورها بی تأثیر باشد یا تأثیر جزئی داشته باشد.
از طراحی یک موتور فراریخت عمومی تا دسته بندی روش‌های کشف ضد ویروس‌ها به صورت جعبه سیاه (Borello, 2010)	➤ کارامد در تشخیص پویای نوعی از بدافزارها ✖ در موارد عملی امکان تشخیص خودکار چنین تهدیدهای فراریختی مشکل است . ✖ شناسایی پیچیده خود تکرار مانند فراریخت‌ها نیست، منجر به نتایج 100٪ منفی کاذب می شود زمانی که با کرم‌های فراریخت مواجه می شود.
ویروس فراریخت با سرریز بافر داخلی (توکار) (Shah, 2010)	➤ کارامد در تولید ویروس‌های فراریخت غیر قابل کشف ➤ سر بار کمتر، چون به جای کد اسمبلی، روی کد باینری فایل ویروس کار می کند. ✖ روش پیشنهادی روی سیستم عامل‌هایXP و ویستا و 7 کار می کند هر چند که می توان آن را بر روی بعضی سیتم عامل‌ها مانند سیستم عامل Mac OS X توسعه داد در اینصورت نیاز به آزمایش‌هایی برای ارزیابی کارایی دارد.
مدل سازی فراریختی با تفسیر انتزاعی (Dalla Preda, 2010)	➤ کار در استخراج امضای فراریختی ✖ بخش روش تغییر کد به عنوان مثال، اثر دستورالعملی که باعث تغییر دستورالعمل‌های دیگر می شود مشکل است . ✖ تعیین میزان جهش کد که برای ایجاد تحولات فراریختی انجام می شود دشوار است.

جدول 8: مقالات سال 2011 و 2012

مشخصه رویکرد	نقاط قوت و ضعف
کشف ویروس‌های فراریخت غیرقابل کشف (Venkatachalam, 2011)	➤ نشان دهنده کشف ویرس‌های فراریختی با شرایط خاص با روش مدل مخفی مارکوف ✖ از نظر (Deshpande, 2013) مدل مخفی مارکوف تحلیل ایستا نیست، تعریف آن‌ها از تحلیل ایستا خیلی محدود است ولی در هر حال مدل مخفی مارکوف هم روش تحلیل ایستا محسوب می‌شود. ✖ این مقاله مستقیماً نظر (Deshpande, 2013) را رد نمی‌کند ولی فقط روش روش مدل مخفی مارکوف را برای بررسی صحت ادعای آن‌ها (مثال نقض) به‌کار برده است، که از تعریف محدود آن‌ها روش مدل مخفی مارکوف 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 (Saleh, 2011)	➤ روش پیشنهادی از روش‌های آماری مفید در تشخیص بدافزارهایی است که سایر روش‌های آماری در تشخیص آن‌ها موفق نبوده‌اند. ➤ نرخ تشخیص بالا، میزان مثبت اشتباه کم و همچنین می‌تواند الگوهای جدید ویروسها را جهت تشخیص بعدی آموزش ببیند. ✖ برای رسیدن به دقت بالا باید ویروس‌های بسیاری برای آموزش استفاده شوند که زمان و فضای محاسباتی را بالا می‌برد. ✖ برای تشخیص با دقت زیاد بدافزارهایی با درجه فراریختی بالا نیاز به نمونه‌های بسیار دارد.
کشف بدافزار با روش مبتنی بر گراف و استفاده از تحلیل پویا (Anderson, 2011)	➤ در شناسایی ویروس‌ها از فایل‌های سالم بهتر از روش‌های مبتنی بر امضا و n-gram کار می‌کند. ➤ 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 ✖ تمرکز روی شناسایی ویروس‌های چند ریخت است و ویروس‌های فراریخت بررسی نشده‌اند ✖ پیچیدگی محاسبات هسته‌ای در کاربردهای واقعی محدودیت ایجاد می‌کند، هرچند که راه‌کارهایی برای تسریع محاسبات ارائه شده‌اند. ✖ توسعه روش n-gram به gram-2 برای تولید احتمالات گذر در زنجیره مارکوف
کشف ویروس‌های کامپیوتری فراریخت با روش‌های مبتنی بر مورد (Berkat, 2011)	✖ اعتبار سنجی روش پیشنهادی صورت نگرفته است و روش پیشنهادی نقاط ضعف بسیار دارد.
آزمون شباهت برای کشف ویروس‌های فراریخت (Patel, 2011)	➤ در مقایسه با روش شاخص شباهت برای کشف ویروس‌های تغییر شکل یافته، روش مذکور ویروس‌هایی را که تا 25٪ کد درج شده دارند، کشف می‌کند. ✖ در مطالعات (ANDO,2005) نشان داده شده است که اگر ویروس را به فایل‌های نرمال نزدیک‌تر کنیم، مدل مبتنی بر روش مدل مخفی مارکوف در کشف ویروس‌های تغییر شکل یافته‌ای که 5٪ کد مرده و زیرروال از فایل‌های نرمال در آن‌ها درج شده، موفق نیستند. ✖ اگر همه ویروس‌های تغییر شکل یافته پیش‌پردازش شود، روش شاخص شباهت کارایی بهتری در مقایسه با سایر روش‌های مبتنی بر شباهت خواهد داشت.
تشخیص فراریختی از طریق شبیه سازی (Priyadarshi, 2011)	➤ قابل پیاده سازی ✖ برای کشف همه ی ویروس‌ها کاربردی نیست. ✖ از نظر امنیتی کامل نیست زیرا امکان حمله و تغییر شبیه ساز کد وجود دارد. ✖ شبیه ساز کد در این مقاله شبیه سازی مراقبت از دستگاه I/O را ندارد.
شباهت گراف کدهای عملیاتی و کشف فراریختی (Runwal, 2012)	➤ محاسبه امتیاز ساده و کارامد است. ✖ اگر از روش جایگذاری دستورهای مشابه در فراریختی استفاده شود، یا دستورات با دقت جابجا شوند، تشخیص فراریختی دشوار می‌شود

جدول 9: مقالات سال 2013 و 2014

مشخصه رویکرد	نقاط قوت و ضعف
کشف ویروس‌های فراریخت با استفاده از فاصله کای دو (Toderici, 2013)	➤ روش فاصله آماری کای دو ساده‌تر است و اغلب اوقات عملکردش مشابه روش ترکیبی است. پس در عمل ارجحیت دارد. ✖ روش پیشنهادی سربار محاسباتی بیشتری دارد.
تشخیص فراریختی با استفاده از تجزیه و تحلیل گراف فراخوانی تابع (Deshpande, 2013)	➤ در تشخیص ویروس‌های فراریخت بهتر از مدل مخفی مارکوف عمل می کند. ➤ می توان این روش تشخیص را در انواع دیگر بدافزارها مانند ردگم کن، درب پشتی، گسترش داد. ✖ این روش تنها قادر به کشف ویروسهای فراریخت است.
تشخیص فراریختی با استفاده از تجزیه مقادیر تکین (Kumar Jidigam, 2013)	➤ روشی بسیار سریع و کارآمد ➤ در این پروژه کارهای قبلی انجام شده در تشخیص بدافزار با استفاده از روش تجزیه و تحلیل مقادیر ویژه (Bruschi, 2006) بهبود یافته است. ✖ این روش تنها بر روی سه نوع بدافزار آزمایش شده است و بر روی دیگر خانواده بدافزارها آزمایش نشده است.
کشف گونه‌های بدافزار تغییرشکل‌یافته از طریق تخصیص موتور (Chouchane, 2013)	➤ این روش روی کد اسمبلی کار می‌کند، می‌توان آن‌را برای تحلیل کد باینری هم توسعه داد که در این‌صورت نیاز به آزمایش‌هایی برای ارزیابی کارایی دارید. ➤ سربار استخراج، نگهداری و توزیع امضای تک تک بدافزارها (با این حجم انبوه) را کم می‌کند. ✖ روش‌های پیشنهادی در کشف ویروس‌های رمزگذاری شده قابل به‌کارگیری نیستند هر چند در آزمایش‌ها چند بدافزار چند ریخت هم وجود داشته است. ✖ نرم‌افزار کشف کننده باید قادر به برگرداندن به اسمبلی باشد. ✖ نیاز به تحلیل رفتاری همچون تحلیل جریان اجرایی یا نماسازی توسط نرم‌افزار کشف کننده را مرتفع می‌سازد.
فاصله جانشینی ساده و کشف ویروس‌های فراریخت (Shanmugam, 2013)	➤ 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 فراریخت با روش‌های ابتکاری شمارش دستورات تکراری (Canfora, 2014)	➤ 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.
کرم فراریختی که موتور فراریختی خودش را با خود دارد (Sridhara, 2013)	➤ استفاده از روش درج کد زائد که برای جلوگیری از کشف توسط مدل مخفی مارکوف 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 (Baysa, 2013)	➤ 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 (Deshpande, 2014)	➤ این روش برای کدهایی که درجه فراریختی آن‌ها بالاست و از روشهای تشخیص آماری گریخته‌اند مفید است. ✖ روش فراریختی که بتواند با حفظ آمار دستورات، شکل کد را عوض کند، روش پیشنهادی را دچار شکست می‌نماید.

4- نتیجه‌گیری

در این مقاله 37 مفاهیم ویروس و کرم به‌عنوان دو بدافزار رایج معرفی شد. سپس فراریختی که از روش‌های مؤثر برای جلوگیری از کشف ویروس است مورد بررسی قرار گرفت و روش‌های کشف ویروس‌های فراریخت بیان شد. در نهایت، خلاصه مطالعه‌ی 37 روش

مطرح در کشف ویروس‌های فراریخت با ارائه نقاط قوت و ضعف آن‌ها ارائه شد. بررسی این نقاط قـوت و ضـعف بـرای پـژوهش آتـی در زمینه کشف فراریختی مفید خواهد بود.

منابع

[1] 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.

[2] 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.

[3] S. Attaluri, “Detecting metamorphic viruses using profile hidden markov models,” Diss. San Jose State University, 2007.

[4] S. Attaluri, S. McGhee, M. Stamp, “Profile hidden Markov models and metamorphic virus detection,” *J. Comput. Virol.* Vol. 5, No. 3, pp. 151-169, 2009.

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

[6] 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.

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

[8] P. Beaucamps, “Extended recursion-based formalization of virus mutation,” *J. Comput. Virol.* Vol. 5, No. 3, pp. 209-219, 2009.

[9] A. Berkat, "Metamorphic computer virus detection by Case-Based Reasoning (CBR) methods," *International Journal of Software Engineering & Applications* Vol. 2, No. 4, 2011.

[10] J-M. Borello, É. Filiol, L. Mé, “From the design of a generic metamorphic engine to a black-box classification of antivirus detection techniques,” *J. Comput. Virol.* Vol. 6, No. 3, pp. 277-287, 2010.

[11] J-M. Borello, L. Mé, “Code obfuscation techniques for metamorphic viruses,” *J. Comput. Virol.* Vol. 4, No. 3, pp. 211-220, 2008.

[12] 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.

[13] 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.

[14] 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.

[15] R. Chouchane, N. Stakhanova, A. Walenstein, A. Lakhotia, “Detecting machine-morphed malware variants via engine attribution,” *J. Comput. Virol.* Vol. 9, No. 3, pp. 137-157, 2013.

[16] M. Dalla Preda, R. Giacobazzi, S. Debray, K. Coogan, G. Townsend, “Modelling Metamorphism by Abstract Interpretation,” In *Proceedings of the 17th international conference on Static analysis*(SAS'10). Springer-Verlag, Berlin, Heidelberg. Vol. 6337, pp. 218-235, 2010.

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

[18] P.Deshpande, Metamorphic Detection Using Function Call Graph Analysis. Master’s report, Department of Computer Science, San Jose State University, 2013.

[19] 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.

[20] X. Gao, Metamorphic software for buffer overflow mitigation. Master’s report, Department of Computer Science, San Jose State University, 2005.

[21] S. Govindaraj, Practical Detection of Metamorphic Computer Viruses. Master’s report, Department of Computer Science, San Jose State University, 2008.

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

[23] R.Kumar Jidigam, Metamorphic Detection Using Singular Value Decomposition. Master’s report, Department of Computer Science, San Jose State University, 2013.

[24] 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.
[25] D. Lin. Hunting for undetectable metamorphic viruses, Master's Projects. Paper 18. http://scholarworks.sjsu.edu/etd_projects/18, 2009.
[26] S. Madenur Sridhara, M. Stamp. “Metamorphic worm that carries its own morphing engine,” *J. Comput. Virol.* Vol. 9, No. 2, pp. 49-58, 2013.
[27] McAfee Labs. Mcafee threats report: First quarter 2013. Technical report, McAfee, 2013.
[28] S. McGhee, Pairwise Alignment of Metamorphic Computer Viruses. Master’s report, Department of Computer Science, San Jose State University, 2007.
[29] P. Mishra, “Taxonomy of uniqueness transformations,” http://www.cs.sjsu.edu/faculty/stamp/students/FinalReport.doc, 2003.
[30] M. Patel, Similarity tests for metamorphic virus detection. Master’s report, Department of Computer Science, San Jose State University, 2011.
[31] 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.
[32] S. Priyadarshi, Metamorphic Detection via Emulation. Master’s report, Department of Computer Science, San Jose State University, 2011.
[33] 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.
[34] 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.
[35] R. Shah, Metamorphic Viruses with Built-In Buffer Overflow. Master’s report, Department of Computer Science, San Jose State University, 2010.
[36] M. Saleh, "Towards Metamorphic Virus Recognition Using Eigenviruses," *arXiv preprint arXiv,* pp. 1206.5871, 2012.
[37] G. Shanmugam, R. M. Low, M. Stamp, “Simple substitution distance and metamorphic detection,” *J. Comput. Virol.* Vol. 9, No. 3, pp. 159-170, 2013.
[38] 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.
[39] M. Stamp, *Information Security: Principles and Practice*. Wiley, New York, 2011.
[40] P. Szor. The Art of Computer Virus Research and Defense. Addison-Wesley Professional, 2005.
[41] A. H. Toderici and M. Stamp, “Chi-squared distance and metamorphic virus detection,” .*J. Comput. Virol.* Vol. 9, No. 1, pp. 1-14, 2013.
[42] A. Venkatesan, Code Obfuscation and Virus Detection. Master’s report, Department of Computer Science, San Jose State University, 2008.
[43] S. Venkatachalam, M. Stamp, “Detecting undetectable metamorphic viruses,” In *Proceedings of 2011 International Conference on Security & Management (SAM’11),* pp. 340-345, 2011.
[44] M. Webster, M. Grant, "Detection of metamorphic computer viruses using algebraic specification," *Journal in Computer Virology* Vol. 2, No. 3, pp. 149-161, 2006.
[45] M.Webster, G. Malcolm, “Detection of metamorphic and virtualization-based malware using algebraic specification,” *J. Comput. Virol.* Vol. 5, No. 3, pp. 221-245, 2009.
[46] W. Wong, “Analysis and detection of metamorphic computer viruses,” Department of Computer Science, San Jose State University, May, Master’s Thesis, 2006.
[47] W.Wong, M. Stamp, “Hunting for metamorphic engines,” *J. Comput. Virol.* Vol. 2, No. 3, pp. 211-229, 2006.
[48] 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.
[49] P. Zbitskiy. “Code mutation techniques by means of formal grammars and automatons,” *J. Comput. Virol.* Vol. 5, pp. 199-207, 2009.