مرکز تخصصی

آپا دانشگاه کردستان

شناسایی و تحلیل بدافزارها

هادی گلباغی

شماره سند: A96005

۱۳۹۶/۱۱/۱۶

www.cert.uok.ac.ir
apa@uok.ac.ir
087-33662932

۱- چکیده

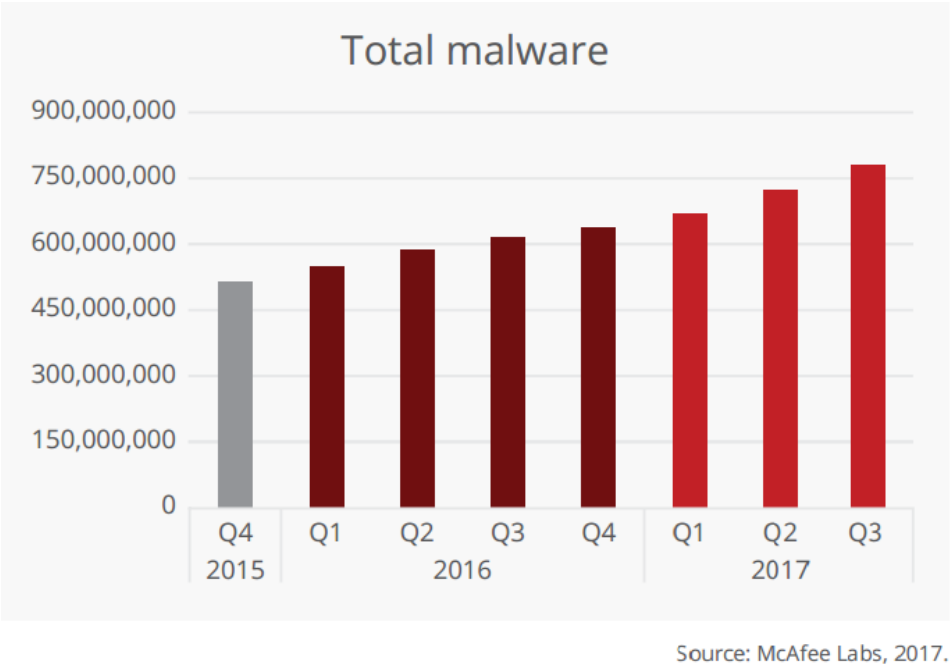
سال‌هاست که بدافزارهای گوناگون برای عملیات خرابکارانه در سیستم‌های کامپیوتری تولید می‌شوند. در سالیان اخیر شناسایی بدافزارها مسئله دشواری است، زیرا با وجود پیشرفت الگوریتم‌ها و نرم‌افزارهای شناسایی و پویش بدافزارها، نویسندگان و سازندگان بدافزارها نیز رشد کرده‌اند و از روش‌های متنوع و هوشمند برای تولید بدافزارشان بهره برده‌اند که بدافزار تولیدی آن‌ها از شناسایی فرار کند و این امر شناسایی بدافزارها را بسیار پیچیده و دشوار کرده است. برای شناسایی بدافزارها سه روش کلی وجود دارد. روش‌های مبتنی بر تحلیل ایستا، مبتنی بر تحلیل پویا و روش های ترکیبی این دو، تاکنون مدنظر بوده‌اند. روش‌های ایستا که سعی می‌کنند اطلاعات مختلفی از کد دودویی و یا کد اسمبلی بدافزار، استخراج کنند تا برای شناسایی خانواده‌های مختلف بدافزار، استفاده شود. علی‌رغم موفقیت‌های بسیار این دسته از روش‌ها، هنوز شرایطی هست که عملکرد قوی ندارند و فنونی برای شکست این روش‌ها وجود دارد که یکی از این فنون مبهم‌سازی کد و اضافه شدن مقداری کد از فایل سالم درون بدافزار جدید است. در مقابل، روش‌های مبتنی بر تحلیل پویا سعی می‌کنند در محیط‌های محافظت شده، جعبه شنی و یا مجازی بدافزارها را اجرا کنند که با مشاهده رفتار بدافزار و کنش‌های انجام گرفته، تحلیلی را صورت دهند. متأسفانه سربار محاسباتی مشکل عمده روش‌های پویا است؛ علاوه بر این بدافزار در اقدام مقابل می‌تواند روش کشف پویا را گمراه سازد. اخیراً سعی در پیشنهاد روش‌هایی مبتنی بر ترکیب روش‌های پویا و ایستا بوده است که برای بهره‌گیری از مزایای دو رویکرد ایستا و پویا و نیز کاهش مشکلات هر کدام، اقدام به طراحی روش‌های ترکیبی کرده‌اند به‌طوریکه بتوان ضمن حفظ کارایی کشف روش‌های ایستا، از اطلاعات ارزشمندی که حین اجرا از تحلیل پویا حاصل می‌شود نیز بهره‌برداری نمود.

واژه‌های کلیدی: بدافزار، مبهم‌سازی کد، موتورهای تکثیر، تحلیل ایستا

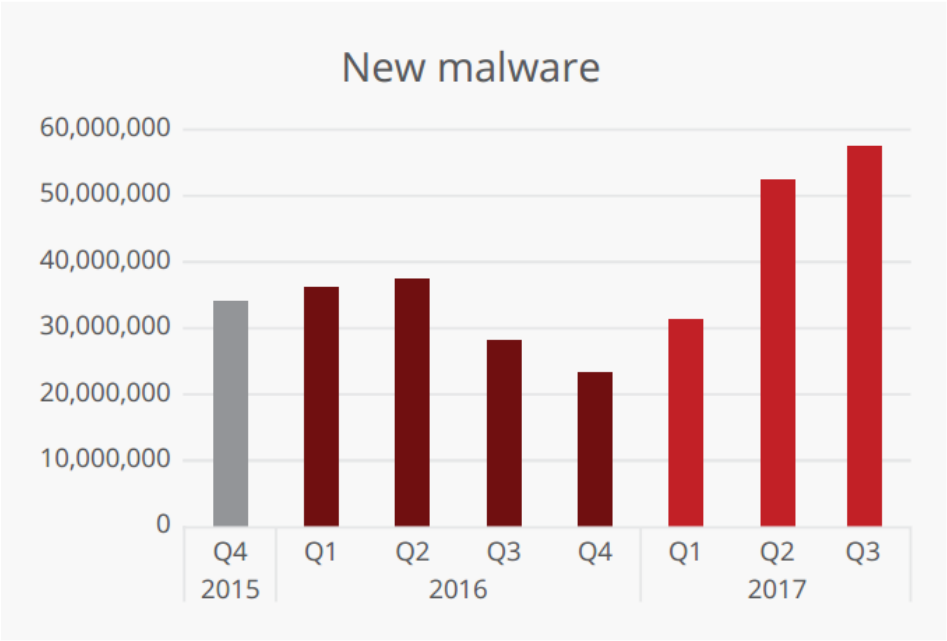
۲- مقدمه

با گسترش روز افزون فناوری‌های نوین و راه پیدا کردن اینترنت در زندگی انسان‌ها در تمامی حیطه‌ها، زمینه برای نفوذ و آلوده شدن سیستم‌های مختلف با نیت‌های متفاوت، فراهم شده است. بدافزارها[1] برنامه‌های کامپیوتری هستند که هدف آن‌ها آسیب‌رسانی به سیستم‌های کامپیوتری است [۱]. بدافزارها علاوه بر اینکه با عملیات خرابکارانه به منابع و اطلاعات سیستم‌ها دسترسی پیدا کرده و به سرویس‌دهی و فعالیت‌های سیستم‌ها آسیب وارد می‌کنند، با سرقت غیرمجاز اطلاعات شخصی افراد هزینه‌های روحی و روانی زیادی در کنار ضررهای مالی، در زمینه افشای حریم خصوصی وارد می‌کنند [۲]. به همین منظور مسئله شناسایی بدافزارها برای دو دهه اخیر مورد توجه شرکت‌های ضد بدافزار و محققان بوده است. بر مبنای گزارش‌های سالیان اخیر بدافزارها به طور روزافزونی تهاجمی‌تر شده‌اند [۳]. در شکل ۱ آمار مجموع تعداد بدافزارها از سال ۲۰۱۵ تا سه سوم سال ۲۰۱۷ و رشد آن‌ها نشان داده شده است. تعداد نمونه‌های ثبت شده در سال ۲۰۱۷ نسبت به زمان مشابه در سال ۲۰۱۵ رشد بالایی داشته است. همچنین طبق همین گزارش، تعداد بدافزارهای جدید تولید شده از سال ۲۰۱۵ تا سه سال ۲۰۱۷، در هر دقیقه ۳۸۷ بدافزار جدید بوده است که این یعنی در هر ثانیه در جهان شش بدافزار جدید تولید می‌شود و آمار آن در شکل ۲ نمایش داده شده است.

1 Malware



شکل ۱- مجموع بدافزارها در پایگاه داده آزمایشگاه مک‌آفی [۴].



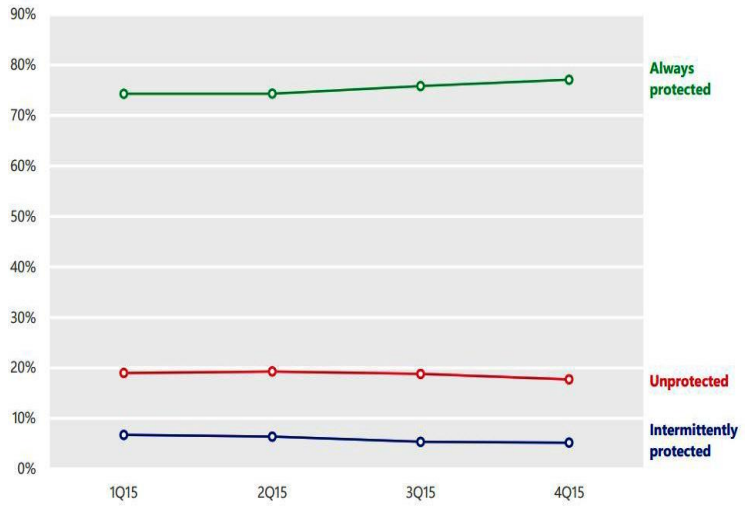
شکل ۲- بدافزارهای جدید تولید شده در پایگاه داده آزمایشگاه مک‌آفی [۴].

تحلیلگران شرکت سمانتیک[۲] نشان داده‌اند که بدافزاری از خانواده بدافزارهای فراریختی حدود ۵۰۰ هزار کامپیوتر را در عرض حدود ۱۸ روز آلوده کرده است [۳]. به طور متوسط در هر حادثه جرایم سایبری ۱۹۷ دلار از دست می‌رود [۳]. در سال ۲۰۱۳ طبق تخمین‌ها حدود ۵۵۶ میلیون کاربر در سرتاسر جهان یک نمونه از جرایم سایبری را تجربه کرده‌اند [۳]. در سال ۲۰۱۱ مشتری‌ها بیشتر از ۴۹ هزار گزارش مختلف از تهدیدات خانواده‌های مختلف بدافزارها را به مرکز حفاظت از بدافزارهای مایکروسافت [۵] داده‌اند. تعدادی از این گزارش‌ها

2 Semantic

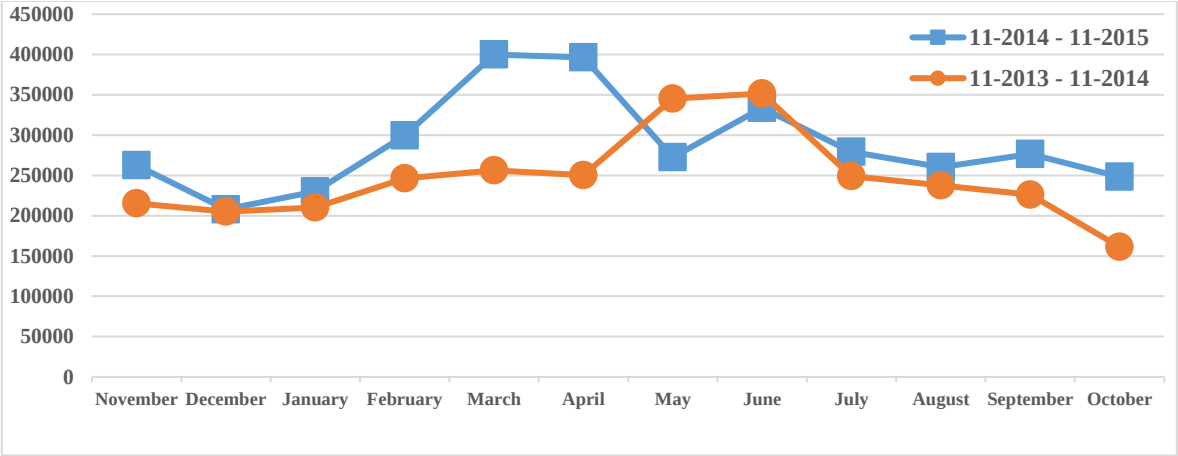
تهدیداتی از جنس خانواده‌های بدافزارهای چندریختی[3] و فراریختی[4] بوده است [۳]. طبق گزارش سال ۲۰۱۶ مرکز امنیتی مایکروسافت، حدود ۳۰ درصد سیستم‌های کامپیوتری موجود در جهان یا سیستم امنیتی ندارند و یا از نظر امنیتی در سطح ابتدایی قرار دارند که آمار آن در شکل ۳ نشان داده شده است [۶].

بر طبق گزارش آزمایشگاه کاسپرسکای[5]، در سال ۲۰۱۵ تعداد سیستم‌هایی که می‌خواسته‌اند با استفاده از اجرای بدافزار دسترسی‌های غیرمجاز به منابع مالی آنلاین بانک‌ها داشته باشند ۱,۹۶۶,۳۲۴ مورد بوده است که نسبت به سال ۲۰۱۴ دارای رشد ۲/۸ درصدی می‌باشد.



شکل ۳- سیستم‌های کامپیوتری موجود در جهان بدون نرم‌افزار امنیتی یا با سطح امنیتی پایین [۶].

فقط در سال ۲۰۱۵ در ۷۵,۳۶۸۴ سیستم کامپیوتری، بدافزار اسب تراوا[6] کشف شده است که از این تعداد ۱۷۹,۲۰۹ مورد رمزنگاری شده بوده‌اند که نسبت به آمار سال ۲۰۱۳ که این تعداد ۱۵,۳۶۳ بوده است تقریبا آمار ۱۲ برابر شده است [۷]. بر طبق همین گزارش که در شکل ۴ نشان داده شده است، در سال ۲۰۱۵ نسبت به سال ۲۰۱۴ تعداد حملات صورت گرفته به وسیله بدافزارهای مالی در حال افزایش بوده است.



شکل ۴- آمار آزمایشگاه کاسپرسکای از افزایش حملات مالی آنلاین در سال ۲۰۱۵ نسبت به سال ۲۰۱۴ [۷].

3 Polymorphic

4 Metamorphic

5 Kaspersky

6 Trojan

طبق گزارش‌هایی که در کنفرانس RSA در سال ۲۰۱۵ منتشر شده است [۸] در بازه زمانی یک ساله ۲۰۱۴ تا ۲۰۱۵ ، حدود ۶۴ درصد ضدویروس‌های تجاری، نتوانسته‌اند یک درصد از بدافزارهای کم‌تر شناخته شده را شناسایی کنند. این در حالی است که این آزمایش‌ها در سال ۲۰۱۳ تا ۲۰۱۴ تنها ده درصد ضد ویروس‌ها موفق به انجام این کار نشده بودند.

طبق پیش‌بینی‌هایی که آزمایشگاه کاسپرسکی برای سال ۲۰۱۸ انجام داده است [۹] سرقت‌های آنلاین از مراکز حساس مانند مراکز مالی و بیمه‌ای، بانک‌ها و بورس، با رشد بالایی مواجه خواهند بود. طبق همین گزارش نفوذ به سیستم‌ها برای دسترسی به داده‌های خصوصی افراد و اخاذی از قربانیان، با مقیاس بالایی رشد خواهند داشت. با توجه به این گزارش‌ها می‌توان پی‌برد که اصولاً شناسایی بدافزارها کاری پیچیده، حساس و بسیار سخت است.

۳- بیان مسئله

در وضعیت کنونی بدافزارهای مختلف رشد بسیار زیادی کرده‌اند و باعث اختلالات اساسی در سامانه‌های حساس[۷] و حیاتی مانند اقتصادی، مالی، نظامی، پزشکی، سیاسی و غیره شده‌اند [۱۰]. مزایای اقتصادی بسیاری که در این حوزه وجود دارد، صنعت بدافزارهای مخرب را به بازاری مستعد برای تولیدکنندگان و نویسندگان بدافزارها تبدیل کرده است، از این رو روش‌های جدید و هوشمند شناسایی هم در بسیاری از مواقع در حرکت همگام با تولید بدافزارهای جدید و شناسایی آن‌ها، عاجز بوده‌اند [۲]. ساختار بدافزارهای جدید و هوشمند، در هر تکثیر با استفاده از فنون مختلف مبهم‌سازی کد[۸]، تغییر می‌یابند اما عملکرد اصلی آن‌ها حفظ می‌شود و این مسئله، شناسایی آن‌ها را بسیار دشوار و پیچیده می‌کند [۱۱]. مولد این بدافزارها همیشه در تکثیرهای بعدی کد، جهش ایجاد می‌کند به‌طوریکه برای هر ویروس در هر نفوذ، یک امضا[۹] متفاوت ایجاد می‌شود که از شناسایی توسط ضد بدافزارها فرار کند [۱۱].

روش‌های متعددی بر مبنای تحلیل ایستای[۱۰] بدافزار کار می‌کنند؛ استفاده از مدل مخفی مارکوف[۱۱] [۱۲] اگر چه در مواردی بسیار سودمند بوده است، اما نقص آن زمان‌بر بودن فیلتر کردن داده‌ها است و اینکه زمانی حجم زیادی از کد فایل سالم به بدافزار اضافه شود روش با شکست روبرو خواهد شد. روش دیگر استفاده از هیستوگرام[۱۲] فراوانی کدهای عملیاتی[۱۳] [۱۳] است که از طریق درج کد زائد و جانشینی دستورات مشابه دچار شکست می‌شود. روش شباهت گراف کدهای عملیاتی [۱۴] روش موثری است اما اگر از روش جانشینی دستورات مشابه استفاده شود دچار چالش خواهد شد. یکی از روش‌های اخیر با استفاده از فاصله کای دو[۱۴] [۱۵] طراحی شده است و سربار محاسباتی بسیار زیادی دارد. همچنین روش تحلیل مقادیر ویژه برای کشف فراریختی [۱۶] در مقابل خانواده‌هایی از بدافزار فراریختی که با حفظ آمار دستورات، شکل کد را عوض می‌کنند، روش دچار شکست می‌شود.

تاکنون چندین روش تشخیص مبتنی بر تحلیل پویای[۱۵] بدافزارها نیز ارائه شده است. در [۱۷] یک سیستم تشخیص بدافزار مبتنی بر امضاء با تأکید بر تشخیص بدافزارها را توسعه دادند. ایده اصلی این روش استخراج امضاء از بدافزار اصلی است، با فرض اینکه تمام بدافزارهای مشابه یک امضای واحد مشترک دارند اما دارای سربار محاسباتی است. در [۱۸] نیز یک روش با استفاده از روش آماری بیز مبتدی[۱۶]، جهت شناسایی کدهای مخرب ناشناخته پیشنهاد شده است که با توجه به سربار محاسباتی بالا، نتایج مطلوبی نداشته است. یک سیستم

7 Critical Systems

8 Obfuscation

9 Signature

10 Static Analysis

11 Hidden Markof Model

12 Historgram

13 Operation Code

14 Chi-Squared

15 Dynamic Analysis

16 Naïve Bayes

هوشمند تشخیص بدافزار در [۱۹] ارائه شده است که این سیستم با استفاده از انجام داده کاوی[۱۷] بر روی فراخوانی‌های سیستمی[۱۸] می‌تواند بدافزار را از برنامه بی‌خطر تشخیص دهد که آزمایش‌های آن برای اثبات دقت روش، ناکافی بوده است. بعضی مواقع روش‌های شناسایی بدافزارها به سبب جایگزین‌کردن دستورات مشابه کد اصلی، رمزنگاری[۱۹] ساختار کد بدافزار یا درج کد زائد دچار شکست می‌شوند و در برخی موارد، سربار محاسباتی بالا، دقت شناسایی و کارایی ضعیف روش‌ها، آن‌ها را دچار چالش می‌کند. به دلیل وجود چالش‌های مطرح شده، کشف کامل و قطعی همه انواع و گونه‌های مختلف بدافزارها با اعمال روش‌های مختلف و پیچیده جدید مبهم‌سازی، هنوز نیاز به پژوهش دارد و مسئله تحقیقاتی ارزشمندی است. با توجه به موارد گفته شده، روش‌های مختلفی مبتنی بر تحلیل ایستا و پویا در حوزه شناسایی بدافزارها موجودند اما هر کدام دارای نقاط ضعف و قوتی هستند که نیاز است در طرح روش‌های جدید نقاط قوت بهبود و نقاط ضعف پوشش داده شوند.

۴- بدافزار

بدافزار برنامه‌ای است که برای برآورده کردن قصد مضر مهاجم یا نویسنده آن، طراحی شده است [۲۰]. از دیدگاه تخصصی، بدافزار در حالت کلی برنامه‌ای است کامپیوتری، که هدف آن آسیب‌رسانی به سیستم‌های کامپیوتری می‌باشد [۱]. بدافزارها تعمداً برای انجام عملیات بدون مجوز، معمولا مخرب یا نامطلوب طراحی شده‌اند [۲۱]. بدافزارها علاوه بر اینکه با عملیات خرابکارانه به منابع و اطلاعات سیستم‌ها دسترسی پیدا کرده و به سرویس‌دهی و فعالیت‌های سیستم‌ها آسیب وارد می‌کنند، با سرقت غیرمجاز اطلاعات شخصی افراد هزینه‌های روحی و روانی زیادی در کنار ضررهای مالی، در زمینه افشای حریم خصوصی وارد می‌کنند [۲]. بدافزارها معمولا اقدامات و کارهای عادی و معمول سیستم را متوقف ساخته، فایل‌ها را آلوده کرده و آن‌ها را خراب یا حذف می‌کنند، اطلاعات کاربران را سرقت می‌نمایند و به منابع سیستم کامپیوتری مانند دیسک سخت، آسیب جدی وارد می‌کنند [۱]. بدافزارها را می‌توان در دسته‌های کلی طبقه‌بندی کرد مانند ویروس[۲۰]، کرم[۲۱]، اسب تراوا، جاسوس‌افزار[۲۲] و برنامه‌های دیگر که به هدف تخریب و آسیب‌رسانی تولید می‌شوند [۱۰]. تولید بدافزارها با اهداف تخریب و آسیب‌رسانی، به شدت با نرخ بالایی در حال رشد است و تمامی آمارها نشان دهنده این امر هستند که تاکیدی بیشتر بر مقابله با آن‌ها باید صورت گیرد و روش‌های دقیق‌تر و کاراتر برای شناسایی و جلوگیری از نفوذ آن‌ها، پیشنهاد شود [۲۲]. اولین نمونه برنامه‌های مخرب ویروس‌ها بوده‌اند. انگیزه اولیه برای سازنده‌های چنین بدافزارهایی معمولا نشان دادن آسیب‌پذیری‌های امنیتی[۲۳] و یا نمایش توانایی فنی بوده است [۲۳]. امروزه اقتصادهای زیرزمینی بر اساس طراحی بدافزارها شکل گرفته‌اند. در شرایط کنونی دیگر انگیزه‌ها از آن موارد اولیه فراتر رفته است و چشم‌اندازهای مالی انگیزه‌های اصلی این حوزه هستند [۲۴]. به همین دلیل، مسئله شناسایی بدافزارها در دو دهه اخیر مورد توجه شرکت‌های ضدبدافزار و محققان بوده است.

۵- ویروس، کرم و اسب تراوا

تقسیم‌بندی بدافزارها به انواع مختلفی از قبیل ویروس، کرم، اسب تراوا، جاسوس‌افزار و موارد دیگر، فقط از این جهت است که هر کدام از این انواع بدافزار، رفتارهای مخرب مشابه را دارند [۲۵].

17 Data Mining
18 System Call
19 Encryption
20 Virus
21 Worm
22 Spyware
23 Security Vulnerability

۱-۵- ویروس

در ویروس‌ها بدافزاری هستند که هنگام اجرا شدن، سعی در تکثیر خود در کدهای اجرایی دیگر دارند [۲۳]. اگر ویروس در این کار موفق شود، کد اجرایی به ویروس، آلوده شده است [۲۳]. ویروس‌ها به طور کلی، با ادغام خود در فایل اجرایی دیگر، اجرا می‌شوند. فایل‌های اجرایی که ویروس‌ها خود را به آن‌ها می‌چسپاند، فایل‌های آلوده می‌گویند [۲۶]. در حقیقت ویروس باید به فایل دیگری بچسپد تا بتواند اجرا شود و نیاز به مداخله انسانی دارد تا منتقل شود یعنی وقتی برنامه آلوده شده اجرا می‌گردد، کد ویروس هم به تبع آن اجرا شده و سایر فایل‌ها را نیز آلوده می‌کند و حاصل آن ایجاد خرابی‌ها و مشکلاتی اساسی برای سیستم است لذا می‌توان گفت ویروس‌ها برنامه‌هایی خودتکثیر هستند [۲۷]. همان‌طور که گفته شد ویروس‌ها خاصیت خودتکثیری دارند و نرخ رشد جمعیت آن‌ها مثبت است. ویروس‌ها برای انتقال، نیاز به مداخله انسانی دارند. آن‌ها می‌توانند از طریق رسانه‌های انتقال، خود را به سیستم‌های دیگر منتقل کنند، اما برعکس کرم‌ها، که در ادامه مورد بررسی قرار خواهند گرفت، توانایی انتقال از طریق شبکه را ندارند [۲۲]. شکل 5 ساختار کد یک ویروس را نشان می‌دهد.

۲-۵- کرم

کرم‌ها نیز مانند ویروس‌ها خاصیت خودتکثیری دارند. کرم‌ها برای تکثیر و بقا به کد اجرایی دیگری نیاز ندارند و معمولا بدون دخالت انسانی، از طریق شبکه‌هایی مانند اینترنت از سیستمی به سیستم دیگر منتقل می‌شوند [۲۳]. کرم‌ها به مداخله انسانی برای انتقال نیاز ندارند و همچنین نیاز به هیچ فایل اجرایی برای نفوذ و مداخله در سیستم میزبان ندارند، بنا بر این مورد، کرم‌ها بدافزارهای مستقلی هستند که معمولا تحت شبکه منتقل می‌شوند [۱]. کرم‌ها نه تنها می‌توانند به سیستم میزبان آسیب برسانند، بلکه قادرند به دیگر سیستم‌های موجود در سطح شبکه نیز آسیب‌هایی را وارد کنند. همچنین کرم‌ها با تکثیر خود می‌توانند شبکه را اشباع کرده و از کار بیاندازند [۲۹]. کرم‌ها رایج‌ترین بدافزار در محیط شبکه‌هایی مانند اینترنت می‌باشند که تعریف آن به این صورت است که یک برنامه‌ای است که می‌تواند بصورت مستقل اجرا شود و بهصورت کامل فعالیت نسخه‌های خود را به سیستم‌های دیگر انتشار دهد که این بازتولید از خصوصیات رفتاری یک کرم است [۲۳]. در سال ۲۰۰۱ کرم The Code Red در اولین روز انتشار حدود ۳۵۹ هزار میزبان اینترنتی را آلوده کرده است [۳۰]. امروزه از کرم‌ها برای استفاده در حملات DDOS[۲۴] استفاده می‌شود که در آن چند کامپیوتر آلوده شده به وسیله کرم، به صورت هماهنگ سعی در از دسترس خارج کردن و یا استفاده از منابع یا پهنای باند شبکه به‌خاطر اهداف خاصی دارند [۳۱].

```
a)
BE0400000        mov    esi, 000000004     ; “  ?”
8BDD             mov    ebx, ebp
B90C000000       mov    ecx, 00000000C   ; “  ?”
81C088000000     add    eax, 000000088   ; “  ê”
8B38             mov    edi, [eax]
89BC8B18110000   mov    [ebx] [ecx] * 4 [00001118]  , edi
2BC6             sub    eax, esi
49               dec    ecx
```

شکل ۵- ساختار کد یک ویروس [۲۸].

24 Distributed Denial Of Service

۳-۵- اسب تراوا

نام این بدافزار از داستان فتح شهر تروا به دست یونانیان گرفته شده است. یونانیان با وارد کردن اسب چوبی بزرگی که به ظاهر هدیه‌ای برای آن‌ها بود، وارد این شهر شدند. شب هنگام سربازان یونانی از اسب‌چوبی خارج شدند و شهر تروا را تصاحب کردند [۱۰]. این بدافزار، برنامه‌ای است که با حفظ ظاهر بی‌خطر خود، کارهای مخفی مخربی انجام می‌دهد [۲۳]. اسب تراوا، خود تکثیر نیست ولی دارای خاصیت انگلی است یعنی برای انتقال، به فایل‌های اجرایی دیگر وابسته می‌باشد [۲۳]. اسب تراوا یک برنامه نفوذی است که به سیستم‌عامل، دسترسی سطح بالا پیدا می‌کند و درحالیکه به نظر می‌رسد در حال انجام یک کار عادی و روتین سیستم است، یک برنامه یا داده ناخواسته را روی سیستم نصب می‌کند که اغلب دارای یک درِپشتی[۲۵] برای دسترسی غیرمجاز به کامپیوتر مقصد است [۱۰]. این درِپشتی‌ها گرایش به دیده نشدن توسط کاربران دارند اما ممکن است باعث کند شدن کامپیوتر شوند. اسب تراوا تلاش برای تزریق شدن به فایل‌ها مانند ویروس‌های کامپیوتری را ندارند اما ممکن است اطلاعات را به سرقت ببرند و یا به کامپیوتر میزبان صدمه بزنند. این نوع بدافزارها ممکن است به وسیله دانلود ناخواسته و یا نصب بازیهای آنلاین یا برنامه‌های تحت شبکه به هدف دسترسی داشته باشند [۱۰]. یک مثال از عملیات مخرب اسب تراوا برنامه ورود به سیستم گذرواژه دزد است. این برنامه قبل از برنامه ورود به سیستم اصلی اجرا می‌شود و درست مانند آن، نام کاربری و گذرواژه کاربر را می‌گیرد و اطلاعات را برای سازنده خود ذخیره کرده و به کاربر پیغام ورود اطلاعات نادرست می‌دهد. کاربر می‌پندارد که اطلاعات خود را نادرست وارد کرده است و درصدد ورود مجدد اطلاعات خود بر می‌آید. در این لحظه گذرواژه دزد، برنامه ورود به سیستم اصلی را اجرا کرده تا اطلاعات کاربر را گرفته و به او اجازه ورود دهد.

در ادامه تمرکز بیشتر بر روی بدافزارهای جدید و هوشمند است و روش‌های شناسایی آن‌ها و ابزار مورد استفاده برای شناسایی، ارائه خواهد شد. ویروس‌ها و کرم‌ها برای مقابله با شناسایی شدن اغلب از فنون مبهم‌سازی کد استفاده می‌کنند تا روش‌های شناسایی که در ادامه توضیح داده خواهند شد، نتوانند آن‌ها را شناسایی کنند [۱]. مبهم‌سازی درک برنامه را دشوار می‌سازد. در حوزه بدافزارها، روش‌هایی مانند رمزنگاری، نیمه چندریختی[۲۶]، چندریختی و فراریختی از جمله روش‌هایی هستند که تولیدکنندگان بدافزار برای مبهم‌سازی ساختار بدافزار، از آن‌ها استفاده می‌کنند [۱۰]. برای تکثیر و تولید نسخه‌های گوناگون بدافزار با مبهم‌سازی، دو روش اصلی وجود دارد : یکی گذاشتن موتور تکثیر داخل خود بدافزار و دیگری داشتن نرم‌افزاری مستقل که روی یک سیستم اجرا شود و یک‌باره چندین نسخه تکثیری از بدافزار بسازد [۱۰]. در ادامه به‌دقت فنون مبهم‌سازی مورد بررسی قرار خواهند گرفت.

۶- فنون مبهم سازی کد

بدافزارهای نوین از روش‌های مختلفی برای فرار از شناسایی شدن بهره می‌برند. در بخش قبل توضیح داده شد که در تکثیرهای مختلفی از این نوع بدافزارها، ساختار کد متفاوت خواهد بود و عملکرد یکسان باقی می‌ماند که این ساختار کد متفاوت، با استفاده از فنون مختلف مبهم‌سازی کد شکل می‌گیرد. برای اینکه بدافزار مبهم‌سازی شوند، یا نسخه جهش یافته‌ای از کد ویروس تولید گردد، راه‌کارهای متعدد مبهم‌سازی پیشنهاد شده‌اند [۳۴]. مبهم‌سازی هم در بخش داده‌ای و هم در بخش جریان کنترلی صورت می‌گیرد. مبهم‌سازی جریان کنترلی با جابجایی دستورات و استفاده از دستورات پرش غیرشرطی صورت می‌گیرد و مبهم‌سازی جریان داده‌ای هم با روش‌هایی همچون تعویض دستورات، درج کد اضافه، جایگزینی دستورات معادل، تعویض ثبات‌ها و جایگشت زیر روال‌ها، تحقق می‌یابد [۳۵]. شکل‌های ۶، ۷، ۸ و ۹ نمونه‌های مختلفی از تغییر در ترتیب اجرای کد[۲۷]، معاوضه ثبات‌ها[۲۸]، تغییر در جایگذاری زیر روال‌ها[۲۹] و تبدیل دستورات[۳۰]، که از فنون مختلف مبهم‌سازی کد می‌باشند را نشان داده‌اند. در ادامه هر کدام از روش‌ها به‌صورت دقیق بررسی خواهند شد.

25 Back Door

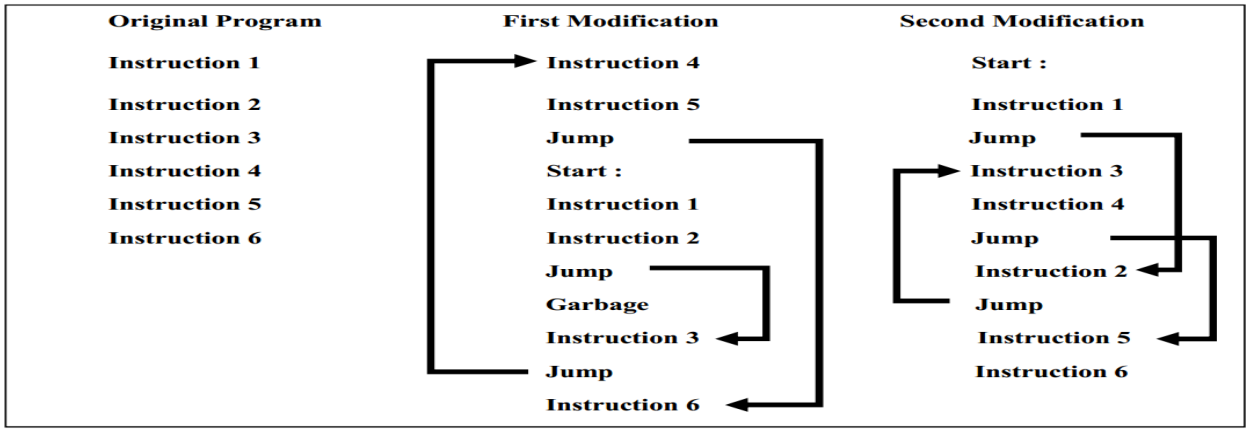
26 Oligomorphism

27 Instruction Reordering

28 Register Swapping

29 Subroutine Permutation

30 Instruction Transformation



شکل ۶- تغییر در ترتیب اجرای دستورات، یکی از فنون مبهم‌سازی کد [۲۸].

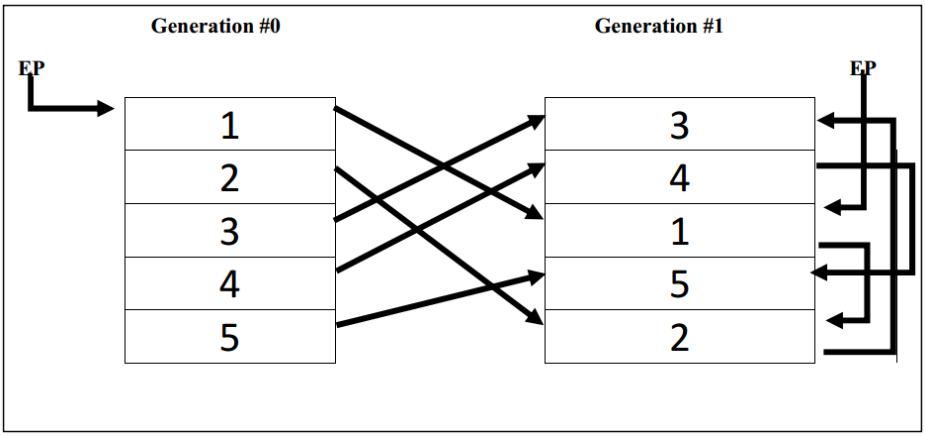
XOR Reg, -1	--->	NOT Reg
SUB Mem, Imm	--->	ADD Mem, -Imm
XOR Reg, 0	--->	MOV Reg, 0
ADD Reg, 0	--->	NOP
AND Mem, 0	--->	MOV Mem, 0
XOR Reg, Reg	--->	MOV Reg, 0
SUB Reg, Reg	--->	MOV Reg, 0
AND Reg, Reg	--->	CMP Reg, 0
TEST Reg, Reg	--->	CMP Reg, 0
LEA Reg, [Imm]	--->	MOV Reg, Imm
MOV Mem, Mem	--->	NOP

شکل ۷- انتقال و تبدیل یک به یک دستورات، یکی از روش‌های مبهم‌سازی کد [۲۸].

```
a)
BE04000000      mov     esi, 000000004     ; “  ?”
8BDD            mov     ebx, ebp
B90C000000      mov     ecx, 00000000C     ; “  ?”
81C088000000    add     eax, 000000088     ; “  ê”
8B38            mov     edi, [eax]
89BC8B18110000  mov     [ebx] [ecx] * 4 [00001118]  , edi
2BC6            sub     eax, esi
49              dec     ecx

b)
BB04000000      mov     ebx, 000000004     ; “  ?”
8BCD            mov     ecx, ebp
BF0C000000      mov     edi, 00000000C     ; “  ?”
81C088000000    add     eax, 000000088     ; “  ê”
8B30            mov     esi, [eax]
89B4B920110000  mov     [ecx] [edi] * 4 [00001120]  , esi
2BC3            sub     eax, ebx
4F              dec     edi
```

شکل ۸- معاوضه ثبات‌ها، یکی از روش‌های مبهم‌سازی کد [۲۸].



شکل ۹- تغییر جایگذاری زیر روال‌ها، یکی از روش‌های مبهم‌سازی کد [۲۸].

با استفاده از فنون مبهم‌سازی، کد بسیار سخت‌تر و پیچیده‌تر قابل شناسایی خواهد بود [۳۶]. هدف اصلی، تغییر شکل ویروس در عین نگه داشتن عملکرد آن است [۲۹]. برای کد بدافزار ، فنون مبهم‌سازی باعث ایجاد تفاوت بسیاری مابین کد اصلی و کد تولید شده می‌شود، به‌طوریکه بسیار سخت‌تر قابل فهم باشد اما همان رفتار و عملکرد را خواهد داشت. مبهم‌سازی کد در بدافزار با توجه به کاری که در نهایت باید انجام شود دارای تفاوت‌هایی می‌باشد. همچنین اضافه کردن کد مرده[۳۱] و یا بلاک بزرگی از کد، از فایل‌های سالم در داخل فایل بدافزار از روش‌های موثر در جلوگیری از کشف بدافزارها است [۱۱]. روش‌های 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 آتاماتای غیر

31 Dead Code

قطعی[۳۲] می‌باشد زیرا گذر از هر نماد به نماد دیگر امکان‌پذیر است [۳۳]. در جدول ۱ دو نمونه ویروس با فنون مختلف مبهم‌سازی دیده می‌شود. همچنین در جدول ۲ نمونه‌های مختلف فنون مبهم‌سازی کد مورد بررسی قرار گرفته‌اند.

جدول ۱- بررسی دو نمونه ویروس که از فنون مختلف مبهم‌سازی استفاده کرده‌اند [۳۶].

نام ویروس	{Win32,Linux}/Simile	Win95/Zmist
ساخته شده توسط	The Mental Driller	Zombie
تاریخ انتشار	2002	2000
تکنیک استفاده شده	EPO مبهم‌سازی نقطه ورود اما با این تفاوت که هر بار به‌صورت تصادفی از جایی از کد برنامه شروع شود	EPO یا مبهم‌سازی نقطه ورود + جابه‌جایی کد + دستورات پرش غیرشرطی
حمله به	فایل‌های اجرایی قابل‌حمل بر روی شبکه	فایل‌های اجرایی قابل‌حمل .EXE
موتور مورد استفاده	RDTSC (Read Time Stamp Counter)	RPME (Real Permutation Engine)

جدول ۲- فنون مختلف مبهم‌سازی کد [۳۶].

شماره	بدافزار	عملکرد	نام دیگر	تکنیک
۱	Evol	الگوهای دودویی کد با تزریق کدهای مرده مثل NOP یا خط خالی عوض می‌شود	تزریق کد اضافه یا تزریق کد بی‌ارزش[۳۳]	تزریق کد مرده
۲	RegSwap	در هر تکثیر بدافزار اسامی ثبات‌ها تغییر می‌یابند	تعویض ثبات‌های مورد استفاده	نام‌گذاری دوباره ثبات‌ها
۳	W32/MetaPhor	دستورالعمل‌ها با دستورات معادل جایگزین می‌شوند	جایگزینی کدهای معادل	جایگزینی دستورات
۴	Win32/Ghost	ترتیب دستورات در هر تکثیر تغییراتی اساسی می‌کند و با پرش‌های غیرشرطی ترتیب واقعی رعایت خواهد شد	انتقال مکان دستورات	انتقال دستورات
۵	Win95/Zperm	ترتیب توابع با کد اولیه تفاوت خواهد داشت	جایگزینی توابع	انتقال توابع
۶	Win32/Simile	روال‌های فراخوانی با کدهای واقعی جایگزین می‌شوند	توابع بر خط	روال‌های برخط
۷	Win32/Simile	کدهای واقعی با استفاده از روال‌ها جایگزین خواهند شد	توابع خارجی	روال‌های خارجی
۸	Zmist	برگرداندن کد به یک حالت کوچک‌تر و ایجاد دوباره فایل	برگردان کد	مهاجرت کد
۹	Win95/Zperm	در هر تکثیر کدهایی با کد میزبان جایگزین خواهند شد	جایگزینی کد میزبان	جهش کد میزبان

32 Non-Deterministic Automata

33 Garbage

در جدول ۳ سطح دشواری کشف برخی از فنون مبهم‌سازی بررسی شده‌اند. همچنین در جدول ۴ فنون مبهم‌سازی که در چند بدافزار تولید شده توسط هکرها پیاده‌سازی شده‌اند را به‌طور خلاصه نشان داده است. الگوهایی که در جدول ارائه شده، تقریبا در ویروس‌های فراریخت کنونی نیز استفاده می‌شوند و در بدترین حالات ترکیبی از آن‌ها مورد استفاده قرار می‌گیرد. باید توجه داشت که در اکثر بدافزارهای فراریخت از ترکیبی از روش‌های مختلف و پیچیده مبهم‌سازی کد استفاده می‌گردد و این خود باعث فرار از شناسایی شدن این نوع بدافزارها خواهد شد.

جدول ۳- مقایسه سطح دشواری فنون مبهم‌سازی کد [۲۸].

فنون مبهم‌سازی	ساده	متوسط	سخت
جابجایی دستورات	+		
درج کد زائد		+	
تعویض ثبات‌ها	+		
جانشینی دستورات			+
تعویض دستورات			+

جدول ۴- مقایسه روش‌های مبهم‌سازی کد در بدافزارها [۳۷].

فنون مبهم‌سازی	Evol 2000	Zmist 2001	Zperm 2000	Regswap 2000	MetaPHOR 2001
جانشینی دستورات				+	
جایگشت	+	+			+
درج کد زائد	+	+			+
جانشینی متغیر	+	+		+	+
تغییر جریان کنترلی		+	+		+

۷- نقاط ضعف بدافزارهای نوین

در خلال توضیحات در ارتباط با بدافزارها، به‌صورت غیرمستقیم به نقاط ضعف و قوت آن‌ها اشاره شد. نکته حائز اهمیت این است که در بدافزارها، موتورها تکثیر همه چیز را در ساختار کد بدافزار تغییر نمی‌دهند. عملیات بر روی تعداد بسیاری از ثبات‌ها عوض نمی‌شود که در نتیجه محتوای برخی از ثبات‌ها اصلا تغییر نمی‌کنند؛ و بسیاری از کدهای عملیاتی در بدنه کد نیز بلاتغییر می‌مانند. این موارد می تواند معیارهایی را برای شناسایی آن‌ها فراهم آورد. البته این بدافزارها، هنگام انتشار، کد خودشان را به گونه دیگری با همان عملکرد تبدیل می‌کنند تا پویشگرهای[۳۴] مبتنی بر امضا، نتوانند آن‌ها را شناسایی نمایند و برای این کار از فنون مبهم‌سازی گفته شده در بخش قبل بهره می‌گیرند. چون سازندگان و نویسندگان بدافزارها، از نقطه ضعف پویشگرهای ضد بدافزار باخبر هستند، کشف این نوع ویروس دشوار شده است. پاشنه آشیل یک بدافزار ، نیازش به تحلیل خودش است یعنی پویشگر باید بتواند ویروس را با روشی مشابه روش به‌کار رفته توسط خود ویروس برای تحلیل خودش، تحلیل کند [۳۲]. در این صورت ضد بدافزار نیاز به یک تغییر دهنده معکوس دارد که با اعمال قوانین تبدیل در جهت عکس، کد اصلی ویروس را بازیابی کند. مشکل اینجاست که باید حداقل یک نمونه از ویروس در اختیار تحلیلگران باشد تا بتوانند قوانین تبدیل، مفروضات و الگوریتم‌ها را استخراج کنند.

34 Scanner

۸- کشف بدافزار

در عملکرد نرم‌افزارهای ضدبدافزار سه مرحله اساسی وجود دارد که عبارتند از :

۱- کشف[۳۵]

۲- شناسایی[۳۶]

۳- پاک‌سازی[۳۷]

تمرکز در اینجا بر کشف و شناسایی بدافزار است. همچنین در یک دسته‌بندی کلی می‌توان روش‌های کشف را به دو دسته ایستا و پویا تقسیم کرد. روش‌های تحلیل ایستا بسیار محبوب و رایجند زیرا بدون اجرای بدافزار و از تحلیل کد آن، قادرند بدافزار را کشف نمایند. روش‌های ایستا سربار محاسباتی کمتری دارند، کم خطرترند زیرا بدافزار را اجرا نمی‌کنند و همه مسیرهای اجرایی را بررسی می‌کنند. در مقابل، روش‌های پویا با اجرای بدافزار می‌توانند آن‌ها را کشف کنند. این روش‌ها عمدتا برای کشف ویروس‌های چندریختی و فراریختی مورد استفاده قرار می‌گیرند. همانندسازی یا نماسازی نمونه‌ای از روش‌های پویا است. از محدودیت‌های روش‌های پویا می توان سربار اجرایی بالا، بررسی یک یا چند مسیر اجرایی محدود را نام برد [۲۳]. روش‌های کشف بدافزار را می‌توان تقسیم‌بندی جزئی‌تری هم کرد. در این حالت روش‌های مبتنی بر امضا، روش‌های مبتنی بر ناهنجاری[۳۸]، وارسی کننده‌های صحت[۳۹] و روش‌های کشف مبتنی بر یادگیری ماشین[۴۰] مطرح می‌شود. کشف بدافزارها با روش‌های مبتنی بر امضا تقریبا رایج‌ترین راه‌کار کشف ویروس‌ها می‌باشد اما در کنار آن‌ها با توجه به پیچیده شدن و اعمال مبهم‌سازی که توسط تولیدکنندگان بدافزارها صورت می‌گیرد از روش‌های دیگری نیز برای شناسایی استفاده می‌گردد [۳۸].

۹- روش‌های تحلیل ایستا

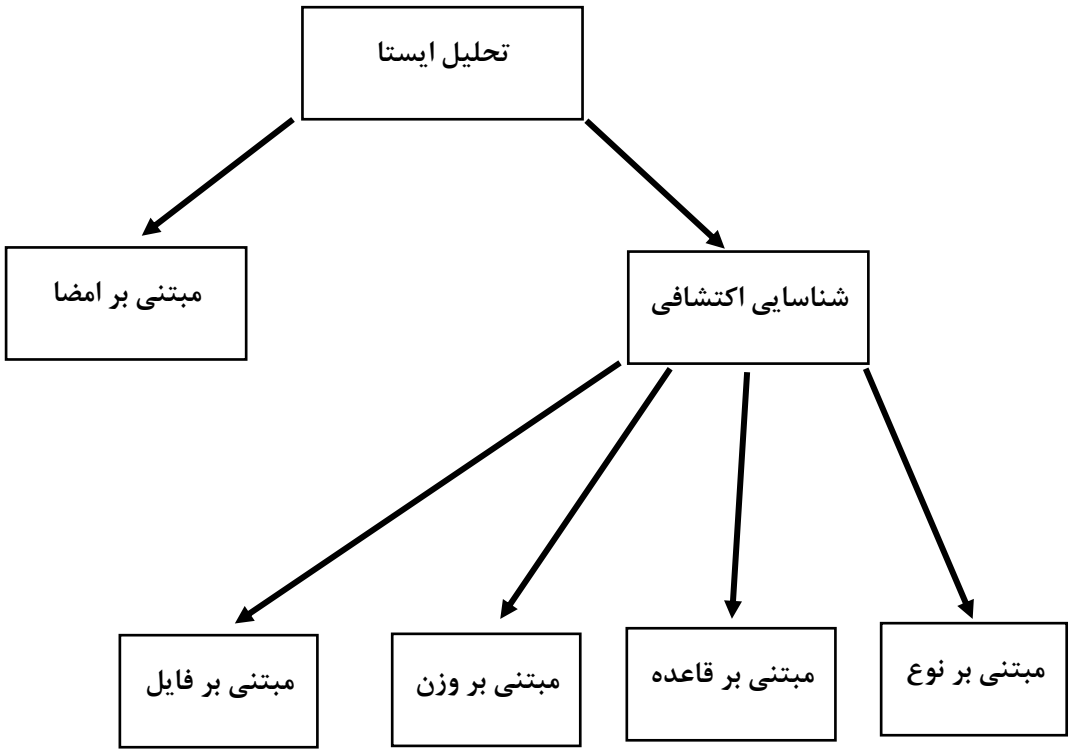
در روش‌های تحلیل ایستا، ارزیابی و شناسایی بدون اجرای فایل مورد بررسی انجام می‌گیرد. با استفاده از ابزار مهندسی معکوس[۴۱] و فنون آن، تحلیل ایستا به ساختار کد برنامه‌ها دسترسی پیدا می‌کند [۳۹]. برای انجام تحلیل ایستا، برگردان‌کننده کد، دیباگر[۴۲] و تحلیل-کننده برنامه مورد استفاده قرار خواهند گرفت [۴۰]. روش‌های تحلیل ایستا می‌توانند بر برنامه‌های مختلف با ساختار مختلف اعمال شوند. در تحلیل ایستا با دسترسی به ساختار کد برنامه، با استفاده از ابزاری می‌توان مبتنی بر خصوصیات و معیارهای مختلف عملیات شناسایی را انجام داد. ابزارهای تحلیل ایستا همچنین می‌توانند برای نمایش ساختار دودویی[۴۳] یک برنامه استفاده شوند. هنگامی یک برنامه در قالب یک فایل اجرایی دودویی، کامپایل[۴۴] می‌شود، دسته‌ای از اطلاعات مانند حجم ساختار داده‌ها و یا متغیرها، از بین می‌رود که این عملیات تحلیل روی کد را بسیار پیچیده می‌کند [۴۱]. ابزار تحلیل ایستا همچنین برای استخراج اطلاعات مفیدی از برنامه استفاده می‌شوند. به‌طور مثال، گراف‌های فراخوانی یک تحلیل کلی از توابعی که در ساختار کد فراخوانی می‌شوند، خواهد داد. اگر تحلیل ایستا قادر به محاسبه مقادیر احتمالی پارامترها باشد، این دانش می‌تواند برای مکانیزم محافظتی پیشرفته مورد استفاده قرار گیرد. مشکل روش‌های مبتنی بر تحلیل ایستا به‌صورت کلی این است که همیشه دسترسی به ساختار کد بدافزار ممکن نیست [۲۳]. همچنین امکان دارد بدافزارها ساختار کد خود را رمزنگاری کنند که به‌طور کلی روش‌های تحلیل ایستا در مقابل آن، ناتوان خواهند بود [۴۴]. برای مقابله با این ضعف‌ها

35 Detection
36 Identification
37 Disinfection
38 Anomaly-Based
39 Integrity Checker
40 Machine Learning
41 Reverse Engineering
42 Debugger
43 Binary
44 Compile

روش‌های تحلیل پویا نیز مدنظر قرار گرفته‌اند. در شکل ۱۰ فنون شناسایی بدافزار مبتنی بر تحلیل ایستا نشان داده شده است که به دو دسته تقسیم می‌شوند. در ادامه هر کدام بررسی خواهند شد.

۱-۹- روش‌های مبتنی بر امضا

این روش‌ها با عناوین دیگری مانند تطبیق الگو[45] و یا فنون اثر انگشت[46] نیز نامیده می‌شوند. یک امضا ترتیبی از بایت‌های مستخرج از برنامه یا بدافزار می‌باشد که به‌صورت منحصربه‌فرد است. برای شناسایی یک بدافزار از روی کد، شناساگر بدافزار برای یک امضای شناخته شده از قبل، در کد جستجو می‌کند. ضدبدافزارهای تجاری، به‌طور معمول مبتنی بر امضاهای ذخیره شده در بانک‌های اطلاعاتیشان، در کد بدافزارها به دنبال ترتیبی از بایت‌ها که تطبیقی با امضاها داشته باشند، پویش انجام می‌دهند که فایل‌های مخرب را شناسایی کنند [۴۱].



شکل ۱۰- دسته‌بندی کلی روش‌های شناسایی بدافزار مبتنی بر تحلیل ایستا [۴۱].

۲-۹- روش‌های شناسایی اکتشافی[47]

این روش‌ها فنون شناسایی فعالانه[48] نیز نامیده می‌شوند [۴۲]. در این روش‌ها به جای جستجو برای یک امضای خاص در ساختار کد، شناساگر بدافزار برای دستوراتی جستجو می‌کند که در حال حاضر در فایل مورد نظر وجود ندارند. این فنون باعث شناسایی انواع جدید بدافزارها که هنوز کشف نشده‌اند، خواهد شد [۴۱]. روش‌های شناسایی اکتشافی دارای انواع مختلفی است که در ادامه مورد بحث قرار می‌گیرند.

45 Pattern Matching

46 Fingerprinting Technique

47 Hueristic Detection Technique

48 Proactive Technique

۳-۹- روش‌های شناسایی اکتشافی مبتنی بر تحلیل فایل[۴۹]

این دسته، روش‌های شناسایی مبتنی بر تحلیل فایل نیز نام برده می‌شوند. در این روش، فایل از نظرهای متفاوتی مانند محتوای فایل، اهداف فایل، محل قرارگیری فایل و فعالیت‌های فایل، مورد تحلیل قرار می‌گیرد [۴۱]. اگر فایل مورد نظر، به طور مثال شامل دستوراتی باشد که اطلاعاتی را حذف کند یا باعث تخریب و آسیب‌رسانی در سیستم شود، در دسته فایل‌های مخرب قرار می‌گیرد. به صورت واضح‌تر یعنی اینکه در ساختار فایل، فعالیت‌هایی در محدوده آسیب‌رسانی و نفوذ انجام گیرد، در دسته فایل‌های مخرب طبقه‌بندی خواهد شد.

۴-۹- روش‌های شناسایی اکتشافی مبتنی بر تحلیل وزن[۵۰]

در این روش هر برنامه بر اساس خصوصیات و مشخصات مختلفی وزن‌دهی می‌شود. اگر نسبت به ویژگی‌های مختلف ، مقادیر وزن‌ها نسبت به حد آستانه تعیین شده در آن روش، بسیار بالاتر باشد می‌تواند در دسته فایل‌های مخرب قرار گیرد.

۵-۹- روش‌های شناسایی اکتشافی مبتنی بر قاعده[۵۱]

در این روش مجموعه‌ای از قواعد و قوانین از برنامه‌ها استخراج می‌شود. این قواعد با معیارهای شناخته شده و تعیین شده از قبل تطبیق داده خواهند شد. اگر تطبیق صورت نگیرد برنامه مورد نظر در دسته بدافزارها طبقه‌بندی خواهد شد.

۶-۹- روش‌های شناسایی اکتشافی مبتنی بر نوع[۵۲]

این روش مشابه با روش‌های مبتنی بر امضا، عملیات شناسایی را انجام می‌دهد. در مواردی ساختار کد و حتی عملکرد بدافزارهای یک خانواده، متفاوت است اما خصوصیاتی مشترک در آن وجود دارد که فقط انواع بدافزارهای آن خانواده خاص، دارای آن خصوصیات می‌باشند. با استفاده از این روش می‌توان حتی بدافزارهای جدید را نیز کشف کرد و به طور گسترده در ضد بدافزارها استفاده می‌شود. در جدول ۵ چند ابزار شناسایی بدافزار مبتنی بر تحلیل ایستا دیده می‌شود.

جدول ۵- چند ابزار شناسایی بدافزار مبتنی بر تحلیل ایستا [۴۳].

شماره	روش شناسایی	نام ابزار
۱	روش مبتنی بر امضا	PEid
۲	روش مبتنی بر استخراج محتوی	Hex Rays Compiler
۳	روش مبتنی بر امضا	Resource Hacker
۴	روش مبتنی بر ناهنجاری	Dependency Walker

۱۰- روش‌های تحلیل پویا

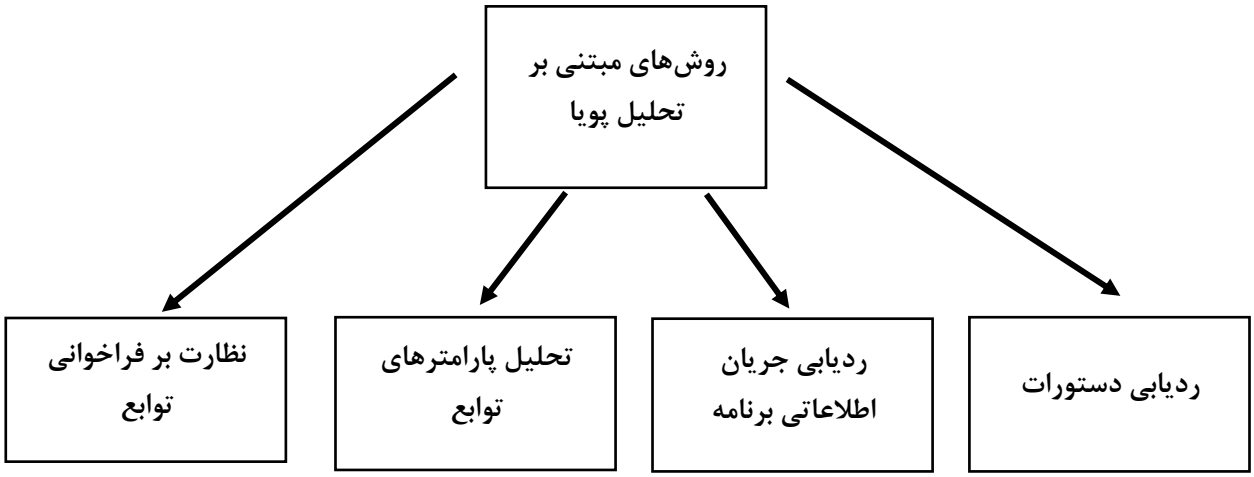
تحلیل فعالیت‌ها و اقدامات انجام شده یک برنامه، در طول زمانی که اجرا می‌شود را تحلیل پویا گویند [۲۳]. تحلیل پویا با استفاده از روش‌ها و فنون مختلفی انجام می‌گیرد که در شکل ۱۱ یک دسته‌بندی کلی از آن‌ها نشان داده شده‌اند و در ادامه مورد بحث قرار خواهند گرفت.

49 File Based Hueristic Analysis

50 Weight Based Hueristic Analysis

51 Rule Based Hueristic Analysis

52 Generic Siqnature Analysis



شکل ۱۱- روش‌های شناسایی مبتنی بر تحلیل پویا [۲۳].

۱-۱۰- نظارت بر فراخوانی توابع[۵۳]

به طور معمول یک تابع شامل کدی است که یک فعالیت خاص را انجام می‌دهد، مانند محاسبه فاکتوریل یک مقدار و یا ساخت یک فایل که در قالب یک تابع انجام می‌شود. استفاده از تابع باعث سهولت در استفاده مجدد می‌شود و نگهداری برنامه آسان‌تر خواهد شد. ساختار پیاده‌سازی کد یک تابع و فراخوانی آن، می‌تواند شامل اطلاعات مفیدی برای تحلیل باشد و یک نمای کلی از رفتار برنامه را نشان دهد. یکی از راه‌های ممکن برای نظارت بر توابعی که توسط برنامه فراخوانی می‌شوند، رهگیری یا ردیابی فراخوانی‌ها است [۲۳]. این ردیابی می‌تواند شامل موارد مختلفی از قبیل ثبت فراخوانی‌ها در یک فایل و یا تحلیل پارامترهای ورودی یک تابع باشد. از انواع نظارت فراخوانی توابع می‌توان به کار روی رابط کاربردی برنامه‌نویسی API[۵۴] و نظارت بر فراخوانی‌های سیستمی نام برد.

۲-۱۰- تحلیل پارامترهای توابع[۵۵]

هنگامی‌که پارامترهای توابع در تحلیل ایستا بررسی می‌شدند، سعی بر مدنظر قرار دادن مقادیر ممکن و نوع آن‌ها، با استفاده از روش‌های تحلیل ایستا بوده است اما در تحلیل پارامتر توابع به‌صورت پویا، تمرکز بر مقادیر واقعی جابه‌جا شده در هنگام فراخوانی تابع در زمان روند اجرایی برنامه می‌باشد [۲۳]. ردیابی پارامترها و مقادیر بازگشتی توابع در روند اجرایی برنامه، تفاوت رفتار و خصوصیات توابع مختلف را در فراخوانی‌های متفاوت نشان می‌دهد.

۳-۱۰- ردیابی جریان اطلاعاتی برنامه[۵۶]

یک روش متعامد برای نظارت بر فراخوانی توابع در طول اجرای یک برنامه، تحلیل چگونگی جریان داده‌های برنامه[۵۷] است. هدف از ردیابی جریان اطلاعات، روشن ساختن این مسئله است که داده‌ها در سیستم، در طول زمان اجرا به چه صورت انتشار داده می‌شوند [۲۳]. به‌طور کلی، داده‌هایی که باید مورد نظارت قرار گیرند به‌طور خاص با برچسب‌های مربوطه مشخص خواهند شد. هر زمانی که داده توسط برنامه پردازش شد، آن برچسب‌ها هم همراه داده منتشر می‌شوند. به‌عنوان مثال در زمان دستورات انتساب و استفاده از یک متغیر جریان اطلاعات، آن برچسب زده خواهد شد. علاوه بر این داده‌هایی که برچسب آلوده زده شده‌اند در انتشارهای بعدی نیز مورد نظارت هستند.

53 Function Call Monitoring

54 Application programming interface

55 Function Parameter Analysis

56 Information Flow Tracking

57 Program Data Flow

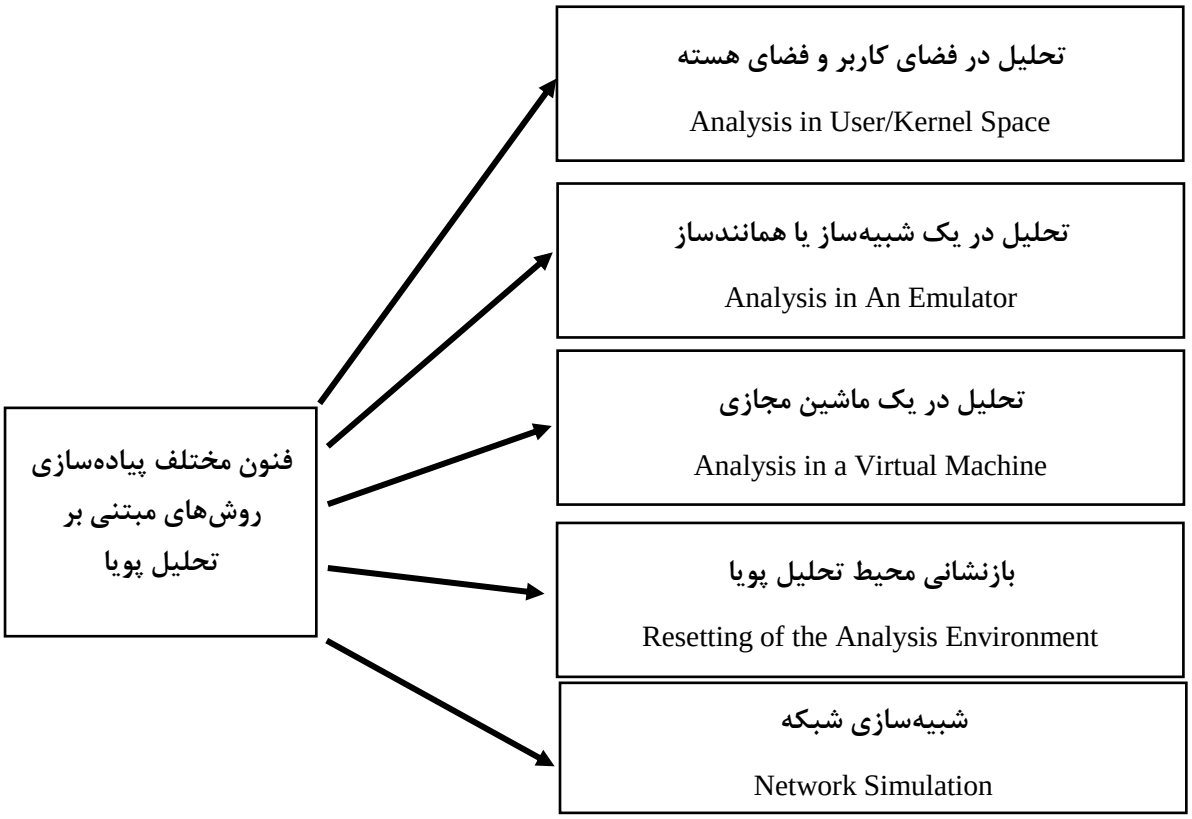
به‌طور مثال استفاده از یک اشاره‌گر آلوده به عنوان ایندکس[58] یک آرایه[59] و یا به عنوان شروط یک دستور، که با نظارت بر انتشار آن‌ها، می‌توان ردیابی اطلاعات را در جریان برنامه ادامه داد. ردیابی جریان اطلاعاتی برنامه دارای انواع مختلفی از قبیل وابستگی مستقیم بین داده‌ها، وابستگی آدرس‌ها و وابستگی جریان کنترلی[60] است.

۴-۱۰- ردیابی دستورات[61]

یک منبع ارزشمند از اطلاعات برای تحلیل رفتار یک برنامه ردیابی دستورات آن است. نمونه‌ها و ترتیب‌هایی از دستورات در طول اجرای برنامه با استفاده از معیارهایی، مورد تحلیل قرار می‌گیرند. حقیقت این است که تفسیر و استفاده از این مورد مقداری پیچیده است اما شامل اطلاعات مهمی در سطح بالایی از انتزاع مانند تحلیل گزارش‌های سیستم و یا فراخوانی توابع، نخواهد بود [۲۳].

۵-۱۰- فنون مختلف پیاده‌سازی روش‌های تحلیل پویا

پیاده‌سازی سیستم‌های تحلیل بدافزار، نیازمند برقراری شرایط ویژه‌ای است. وجود سطوح اولویت مختلف CPU برای تحلیل و ارزیابی اجرای بدافزارها ضروری است. معمولا بدافزارها زمانی تشخیص بدهند که در یک سیستم واقعی در حال اجرا نیستند فعالیت‌های اصلی خود را که شامل تخریب و آسیب‌رسانی است انجام نمی‌دهند که این باعث می‌شود که از شناسایی فرار کنند [۲۳]. جدول ۶ چند ابزار شناسایی بدافزار مبتنی بر تحلیل پویا را بررسی کرده است. همچنین شکل ۱۲ فنون مختلف پیاده‌سازی روش‌های تحلیل پویا را نشان می‌دهد. همچنین در جدول ۷ نیز چند ابزار شناسایی آنلاین بدافزارها که مبتنی بر تحلیل ایستا و پویا هستند، نشان داده شده است.



شکل ۱۲- فنون مختلف پیاده‌سازی روش‌های مبتنی بر تحلیل پویا [۲۳].

58 Index

59 Array

60 Control Flow

61 Instruction Tree

جدول ۶- چند ابزار شناسایی بدافزار مبتنی بر تحلیل پویا [۴۳].

شماره	روش شناسایی	نام ابزار
۱	روش‌های مبتنی بر ناهنجاری	IDA Pro
۲	شناسایی اکتشافی	OllyDbg
۳	تطبیق رشته	Regshot
۴	روش مبتنی بر احتمال	Process Monitor
۵	روش فضای حالت احتمالی	Process Explorer
۶	روش‌های مبتنی بر ناهنجاری، احتمالی و فاصله	Immunity Debugger

جدول ۷- چند ابزار شناسایی آنلاین بدافزارها مبتنی بر تحلیل ایستا و پویا [۴۳].

شماره	روش شناسایی	نام ابزار
۱	روش های مبتنی بر اکتشاف	Virus Total
۲	روش های مبتنی بر رفتار	Anubis
۳	روش های مبتنی بر رفتار	Threat Expert
۴	روش های مبتنی بر اکتشاف	Eureka
۵	روش های مبتنی بر امضا	Comodo
۶	روش های مبتنی بر امضا	Threat Analyzer
۷	روش های مبتنی بر رفتار	Metascan

۱۱- جمع‌بندی

بدافزارهای نوین چالش عمده‌ای در کشف دارند. ساختار این بدافزارها در هر تکثیر با استفاده از فنون مختلف مبهم‌سازی کد، تغییر می‌یابد اما عملکرد اصلی آن‌ها حفظ می‌شود و این مسئله، شناسایی آن‌ها را بسیار دشوار و پیچیده می‌کند. مبهم‌سازی یکی از روش‌هایی است که نویسندگان ویروس‌ها و بدافزارهای کامپیوتری به کمک آن، مانع کشف ویروس شده یا کشف ویروس را دشوار می‌سازند. مولد این بدافزارها همیشه در تکثیرهای بعدی کد، جهش ایجاد می‌کند به‌طوریکه برای هر ویروس در هر نفوذ، یک امضای متفاوت ایجاد می‌شود که از شناسایی شدن توسط ضد بدافزارها فرار کند. در یک دسته‌بندی کلی می‌توان روش‌های کشف بدافزار را به دو دسته ایستا و پویا تقسیم کرد. روش‌های ایستا بسیار محبوب و رایجند زیرا بدون اجرای بدافزار و از تحلیل ساختار کد آن، قادرند آن‌ها را کشف نمایند. در مقابل، روش‌های پویا با اجرای بدافزار می‌توانند آن‌ها را کشف کنند که این کار معمولا نیاز به همانندسازی دارد. بعضی مواقع روش‌های شناسایی بدافزارها به سبب جایگزین‌کردن دستورات مشابه کد اصلی، رمزنگاری ساختار کد بدافزار یا درج کد زائد دچار شکست می‌شوند و در برخی موارد، سربار محاسباتی بالا، دقت شناسایی و کارایی ضعیف روش‌ها، آن‌ها را دچار چالش می‌کند. ابزارهای شناسایی مبتنی بر تحلیل ایستا، پویا و ابزارهای آنلاین مورد بررسی قرار گرفتند.

مراجع

[1]Baysa, D., Low, R. M., & Stamp, M. Structural entropy and metamorphic malware. Journal of computer virology and hacking techniques, 9(4), 179-192, 2013.

[2]Chen, L., Li, T., Abdulhayoglu, M., & Ye, Y. Intelligent malware detection based on file relation graphs. In Semantic Computing (ICSC), 2015 IEEE International Conference on (pp. 85-92). IEEE, 2015.

[3]Canfora, G., Mercaldo, F., Visaggio, C. A., & Di Notte, P. Metamorphic malware detection using code metrics. Information Security Journal: A Global Perspective, 23(3), 57-67, 2014.
[4]https://www.mcafee.com/us/resources/reports/rp-quarterly-threats-dec-2017.pdf
[5]https://www.microsoft.com/en-us/security/portal/mmpc/default.aspx
[6]http://download.microsoft.com/download/E/8/B/E8B5CEE5-9FF6-4419-B7BF-698D2604E2B2/Microsoft_Security_Intelligence_Report_Volume_20_Key_Findings_Summary_English.pdf
[7]https://securelist.com/files/2015/12/Kaspersky-Security-Bulletin-2015_FINAL_EN.pdf
[8]Kruegel, Christopher. Evasive Malware Exposed and Deconstructed presented at RSA Conference, San Francisco, April 2015, from website: http://www.rsaconference.com/writable/presentations/file_upload/crwd-t08-evasive-malwareexposed-and-deconstructed.pdf
[9]https://kasperskycontenthub.com/securelist/files/2016/11/KL_Predictions_2017.pdf
[10]Aycock, J. Computer Viruses and Malware (Advances in Information Security). Secaucus, 2006.
[11]Canfora, G., Iannaccone, A. N., &Visaggio, C. A. Static analysis for the detection of metamorphic computer viruses using repeated-instructions counting heuristics. Journal of Computer Virology and Hacking Techniques, 10(1), 11-27, 2014.
[12]Attaluri, S. Detecting metamorphic viruses using profile hidden markov models (Doctoral dissertation, San Jose State University), 2007.
[13]Rad, B. B., &Masrom, M. Metamorphic virus variants classification using opcode frequency histogram. arXiv preprint arXiv:1104.3228. 2011.
[14]Runwal, N., Low, R. M., & Stamp, M. Opcode graph similarity and metamorphic detection. Journal in Computer Virology, 8(1-2), 37-52, 2012.
[15]Toderici, A. H., & Stamp, M. Chi-squared distance and metamorphic virus detection. Journal of Computer Virology and Hacking Techniques, 9(1), 1-14, 2013.
[16]Deshpande, S., Park, Y., & Stamp, M. Eigenvalue analysis for metamorphic detection. Journal of computer virology and hacking techniques, 10(1), 53-65, 2014.
[17]Sung, A.H., Xu, J., Chavez, P., Mukkamala, S. Static Analyzer of Vicious Executables (SAVE). in Proceedings of the 20th Annual Computer Security Applications Conference, pp. 326-334, 2004.
[18]Schultz, M.G., Eskin, E., Zadok, F., Stolfo, S.J. Data Mining Methods for Detection of New Malicious Executables. in Proceeding of IEEE International Conference on Data Mining, pp. 38-49, 2001.
[19]Ye, Y., Wang, D., Li, T., Ye, D. IMDS: Intelligent Malware Detection System. in Proccedings of ACM International Conference on Knowlege Discovery and Data Mining, pp. 1043-1047, 2007
[20]Moser, A., Kruegel, C., And Kirda, E. Exploring multiple execution paths for malware analysis. In Proceedings of the IEEE Symposium on Security and Privacy, 2007.
[21]Al Daoud, E., Jebril, I. H., & Zaqaibeh, B. Computer virus strategies and detection methods. Int. J. Open Problems Compt. Math, 1(2), 12-20, 2008.
[22]Khalilian, A., Baraani, A. "An Investigation and Comparison of Metamorphic Virus Detection and Current Challenges", Biannual Journal Monadi for Cyberspace Security (AFTA), Iran Society of Cryptography, No. 2, Serial 4, pp. 33-63, 2014, (in Persian)
[23]Egele, M., Scholte, T., Kirda, E., & Kruegel, C. A survey on automated dynamic malware-analysis techniques and tools. ACM Computing Surveys (CSUR), 44(2), 6, 2012.
[24]Tanachaiwiwat, S., & Helmy, A. VACCINE: War of the worms in wired and wireless networks. In IEEE INFOCOM (pp. 05-859), 2006.

[25]Zhuge, J., Holz, T., Song, C., Guo, J., Han, X., & Zou, W. Studying malicious websites and the underground economy on the Chinese web. In Managing Information Risk and the Economics of Security (pp. 225-244). Springer US, 2009.

[26]Sridhara, S. M., & Stamp, M. Metamorphic worm that carries its own morphing engine. Journal of Computer Virology and Hacking Techniques, 9(2), 49-58, 2013.

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

[28]Saleh, M. Towards Metamorphic Virus Recognition Using Eigenviruses. arXiv preprint arXiv:1206.5871, 2012.

[29]Runwal, Neha. Graph technique for metamorphic virus detection. Master's thesis, San Jose State University, 2011.

[30]Moore, D., Shannon, C., And Claffy, K. C. Code-red: a case study on the spread and victims of an Internet worm. In Proceedings of the Internet Measurement Workshop. 273–284, 2002.

[31]Kanich, C., Kreibich, C., Levchenko, K., Enright, B., Voelker, G. M., Paxson, V., & Savage, S. Spamalytics: An empirical analysis of spam marketing conversion. In Proceedings of the 15th ACM conference on Computer and communications security (pp. 3-14). ACM, 2008.

[32]Konstantinou, E., & Wolthusen, S. Metamorphic virus: Analysis and detection. Royal Holloway University of London, 15, 2008.

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

[34]Szor, P. The art of computer virus research and defense. Pearson Education, 2005.

[35]Patel, M. Similarity tests for metamorphic virus detection (Doctoral dissertation, San Jose State University), 2011.

Mehra, V., Jain, V., &Uppal, D. DaCoMM: Detection and Classification of Metamorphic Malware. In Communication Systems and Network Technologies (CSNT), 2015 Fifth International Conference on (pp. 668-673). IEEE, 2015.

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

[37]Lin, D., & Stamp, M. Hunting for undetectable metamorphic viruses. Journal in computer virology, 7(3), 201-214, 2011.

[38]Bergeron, J., Debbabi, M., Desharnais, J., M., E., M., Lavoie Y.&Tawbi, N. Static Detection of Malicious Code in executables programs. International Journal of Req Engineering, 2001.

[39]Chen, H., Dean, D., & Wagner, D. Model Checking One Million Lines of C Code. In NDSS (Vol. 4, pp. 171-185), 2004.

[40]Gaikwad, P., Motwani, D., Shinde, V., Survey on Malware Detection Techniques. International Journal of Modern Trends in Engineering and Research ISSN 2349-9745, 2014.

[41]McGraw, G., Morrisett, G. Attacking malicious code: A report to the infosec research council. IEEE Software,17(5):33–44, 2000.

[42]Pandey, S. K., &Mehtre, B. M. Performance of malware detection tools: A comparison. In Advanced Communication Control and Computing Technologies (ICACCCT), 2014 International Conference on (pp. 1811-1817). IEEE, 2014.

[43]Golbaghi, H., Vahidi-Asl, M., Khalilian, A., "A New Approach for Metamorphic Malware Detection by Static Analysis of Registers and Opcodes", Computing Science Journal, Vol. 4, pp. 3-15, (in Persian), 2017