SOFTSEC 2014
کنفرانس امنیت سامانه های نرم افزاری
ایران، شیراز، ۲۵ و ۲۶ اردیبهشت ۱۳۹۳

روشی بهبود یافته برای کشف ویروس‌های فراریخت به‌کمک گراف وزن‌دار کدهای عملیاتی

علیرضا خلیلیان[۱]، سید عسگری قاسم‌پوری[۲]، بهروز ترک لادانی[۳]

[۱]دانشجوی دکتری نرم‌افزار، دانشگاه اصفهان، khalilian@eng.ui.ac.ir

[۲] هیات علمی گروه کامپیوتر و فناوری اطلاعات، دانشکده فنی و مهندسی، دانشگاه آزاد اسلامی قائمشهر، a.ghasempouri@qaemshahriau.ac.ir

[۳] دانشیار گروه مهندسی نرم‌افزار، دانشگاه اصفهان، ladani@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. یکی از روش‌های اخیر برای کشف ویروس‌های فراریخت، استفاده از گراف کدهای عملیاتی است. این روش با مقایسه میزان شباهت خانواده ویروس‌ها و فایل‌های سالم، آستانه‌ای برای دسته‌بندی ویروس از فایل سالم تعیین می‌کند. مشکل اینجاست که در بعضی از خانواده ویروس‌ها، به‌دلیل هم‌پوشانی محدوده شباهت ویروس‌ها از فایل‌های سالم، این روش دچار مشکل می‌شود. در این مقاله با افزودن وزن کدهای عملیاتی به رابطه مقایسه فایل‌ها، شباهت با دقت بیشتر اندازه‌گیری شده و خطای کشف ویروس‌ها کمتر می‌گردد. برای ارزیابی روش پیشنهادی، آزمایشی روی یک خانواده از ویروس‌های فراریخت و فایل‌های سالم صورت گرفته است. نتایج نشان می‌دهد که کاربرد وزن‌دهی به کدها، باعث مجزا شدن محدوده شباهت ویروس‌ها و فایل‌های سالم شده و دقت روش افزایش می‌یابد.

واژه‌های کلیدی

ویروس، فراریختی، مبهم‌سازی کد، گراف کدهای عملیاتی.

۱- مقدمه

امروزه بدافزارها[۱] با نرخ بالا رشد می‌کنند و هر روز چندین هزار از انواع آن‌ها پخش می‌شود [۱]. یکی از انواع بدافزارها، ویروس‌ها یا کرم‌های[۲] فراریخت[۳] هستند [۲]. این‌ها بدافزارهایی هستند که هنگام تکثیر، ساختار کد آن‌ها تغییر ریخت می‌یابد به‌طوریکه با حفظ همان عملکرد قبلی، ظاهر کد متفاوتی داشته باشند. برای انجام تغییر ریخت از روش‌های مبهم‌سازی[۴] [۲] استفاده می‌شود. مبهم‌سازی روش‌های گوناگونی دارد که برخی از آن‌ها همچون استفاده از همروندی[۵] 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 [۳] 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.

برای شناسایی بدافزارها رقابت تنگاتنگی بین مهاجمین[۶] سیستم‌های کامپیوتری و تولید کنندگان نرم‌افزارهای ضد بدافزار[۷] وجود دارد. برای طراحی و پیاده‌سازی ضد بدافزارها دو راه‌کار کلی پویا[۸] و ایستا[۹] بر اساس اجرا یا عدم اجرای کد مشکوک وجود دارد [۴]. مشکلات موجود در روش‌های پویا باعث شده که هنوز هم روش‌های ایستا محبوبیت فراوانی داشته باشند.

در سال‌های اخیر روش‌های متعددی برای شناسایی ایستای این نوع بدافزارها پیشنهاد شده‌اند [۳، ۵، ۶، ۷، ۸]، 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[۱۰] روش مبهم‌سازی به‌کار رفته در سادگی یا دشواری شناسایی

[1] Malware
[2] Worm
[3] Metamorphic
[4] Obfuscation
[5] Concurrency
[6] Attacker
[7] Anti-Malware
[8] Dynamic
[9] Static
[10] Granularity

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.

یکی از آخرین روش‌های مطرح شده در شناسایی ایستای ویروس‌های فراریخت، استفاده از گراف کدهای عملیاتی[1] است [۹]. این روش با مقایسه گراف ویروس‌های فراریخت یک خانواده و تعدادی فایل سالم[2]، دو آستانه[3] به‌دست می‌آورد: یکی برای فایل‌های ویروس فراریخت هم‌خانواده و یکی برای فایل‌های سالم. مشکل اینجاست که ممکن است این دو آستانه با هم هم‌پوشانی پیدا کنند و روش در تعیین ویروس یا سالم بودن فایل مشکوک ورودی دچار شکست شود.

در این مقاله، مشکل روش مذکور با وزن‌دهی کدها حین مقایسه برطرف می‌گردد. بنابراین رابطه مقایسه شباهت توسعه داده می‌شود. نتایج آزمایش‌های انجام شده، تأثیر این بهبود را آشکار ساخته‌اند.

ساختار ادامه مقاله به‌شرح زیر است: در بخش دوم، مروری بر ادبیات موضوع و تعاریف لازم خواهیم داشت. بخش سوم، مهمترین کارهای پیشین را مرور می‌کند. بخش چهارم به شرح روش پیشنهادی می‌پردازد. بخش پنجم آزمایش‌ها و نتایج حاصله را گزارش می‌نماید. بخش ششم هم به نتیجه‌گیری اختصاص دارد.

۲- مروری بر ادبیات موضوع

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[4] می‌نامند [۵]. 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 [۶]. ویروس‌ها و کرم‌ها اغلب از روش‌های مبهم‌سازی استفاده می‌کنند تا روش‌های شناسایی بدافزار که مبتنی بر امضا[5] کار می‌کنند، نتوانند آن‌ها را شناسایی نمایند [۶]. امضا در ساده‌ترین حالت رشته‌ای از بایت‌ها (و احتمالاً بایت‌ها و کاراکترهای جانشین[6]) است که از یک ویروس شناخته شده استخراج شده است [۱۰]. 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[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 فراریختی از جمله روش‌هایی هستند که تولیدکنندگان بدافزار برای مبهم‌سازی ساختار بدافزار مورد استفاده قرار می‌دهند.

در ویروس فراریخت، کل ساختار هنگام تکثیر تغییر می‌یابد به‌طوریکه ویروس جدیدی به‌دست آید با عملکرد همسان با ویروس قبلی و الگوی متفاوت. بخش کلیدی هر ویروس فراریخت موتور جهش[11] آن است [۵]. وظیفه این بخش تغییر شکل بدنه ویروس است و این‌کار با اعمال تبدیلات حافظ معنی روی کد صورت می‌گیرد.

برای اینکه ویروس‌ها را فراریخت کنیم، یا نسخه جهش یافته‌ای از کد ویروس تولید شود، راه‌کارهای متعدد مبهم‌سازی [۱۱] پیشنهاد شده‌اند. انواع روش‌های مبهم‌سازی عبارتند از: تعویض ثبات‌ها[12]، جانشینی[13] دستورات معادل، عوض کردن[14] ترتیب دستورات، جایگشت[15] روال‌ها، جابجایی قطعات برنامه، درج دستورات زائد[16] و جهش گرامر رسمی[17].

پیشتر در مطالعات نشان داده شده است که تعیین اینکه یک برنامه مفروض ویروس است یا خیر، یک مسأله غیرقابل تصمیم‌گیری است [۱۲]. در یک دسته‌بندی کلی می‌توان روش‌های کشف را به دو دسته ایستا و پویا تقسیم کرد. روش‌های ایستا بسیار محبوب و رایجند زیرا بدون اجرای بدافزار و از تحلیل کدش قادرند آن‌را کشف نمایند. روش‌های ایستا سربار محاسباتی کمتری دارند، کم خطرترند زیرا بدافزار را اجرا نمی‌کنند و همه مسیرهای اجرایی را بررسی می‌نمایند.

روش‌هایی که برای کشف ویروس‌های فراریخت پیشنهاد شده‌اند را می‌توان به دو دسته تقسیم کرد [۱۳]: دسته اول روش‌هایی هستند که در محصولات تجاری پیاده‌سازی شده‌اند و تعدادشان محدود است. دسته دوم روش‌های بسیار متعددی هستند که آزمایشگاهی بوده و هنوز کاربردی نشده‌اند. 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[18] یا غیر عملی بودن روش از نظر فضا و زمان محاسباتی. پس کشف فراریختی هنوز یک خلأ تحقیقاتی است.

[1] Opcode Graph
[2] Benign
[3] Threshold
[4] Infected
[5] Signature-based
[6] Wildcard
[7] Scanner
[8] Encryption
[9] Oligomorphism
[10] Polymorphism
[11] Mutation
[12] Register Swap
[13] Substitution
[14] Transposition
[15] Permutation
[16] Garbage
[17] Formal Grammar
[18] False Positive

۳- کارهای پیشین

تاکنون روش‌های متعددی برای کشف ایستای ویروس‌های فرایخت معرفی شده‌اند. برخی از مهمترین روش‌ها عبارتند از: (۱) طبقه‌بندی تبدیلات یکتاسازی[1] [۸] که روش N-gram را معرفی کرده است، (۲) اعمال ترتیب روی دستورات برنامه در جهت کمک به پویشگر ضد ویروس‌ها [۱۴]، (۳) کشف ویروس‌های فرایخت کامپیوتری مبتنی بر تفکیک‌پذیری[2] با استفاده از راه‌کار کنترل افزونگی [۱۵]، (۴) کشف بدافزار خودجهشی با استفاده از تطابق گراف جریان کنترل [۱۶]، (۵) کشف ویروس‌های کامپیوتری فرایخت با استفاده از مشخصه‌های جبری [۱۷]، (۶) شکار موتورهای مولد فرایختی [۷] که با مدل مخفی مارکوف[3] به کشف ویروس‌های فرایخت می‌پردازد، (۷) استفاده از امضای موتور برای کشف بدافزارهای فرایخت [۱۸]، (۸) دسته‌بندی گونه‌های ویروس فرایخت با استفاده از هیستوگرام فراوانی کدهای عملیاتی [۱۹] که از روش‌های داده کاوی بهره برده است، (۹) ویروس‌های ویژه[4] برای تشخیص ویروس‌های فرایخت [۲۰] که از روش پردازش چهره و جبر خطی به‌منظور کشف فرایختی استفاده کرده است، (۱۰) کشف بدافزار با روش مبتنی بر گراف و استفاده از تحلیل پویا [۲۱]، (۱۱) شباهت گراف کدهای عملیاتی و کشف فرایختی [۹]، (۱۲) کشف ویروس‌های فرایخت با استفاده از فاصله کای دو[5] [۳]، (۱۳) فاصله جانشینی ساده و کشف ویروس‌های فرایخت [۱۰] که یکی از روش‌های رمزنگاری را جهت کشف فرایختی توسعه می‌دهد و (۱۴) بدافزار فرایخت و بی‌نظمی ساختاری[6].

از بین روش‌هایی که ذکر شد، مورد ۱۱ که در سال ۲۰۱۲ مطرح شده، مبنای تحقیق حاضر بوده است و در بخش بعدی پیرامون عملکرد و مشکل آن بحث خواهد شد.

۴- روش پیشنهادی

یکی از روش‌های مؤثر در کشف ایستای ویروس‌های فرایخت، استفاده از شباهت گراف کدهای عملیاتی است که در بخش قبلی ذکر شد. به‌طور دقیق عملکرد این روش چنین است:

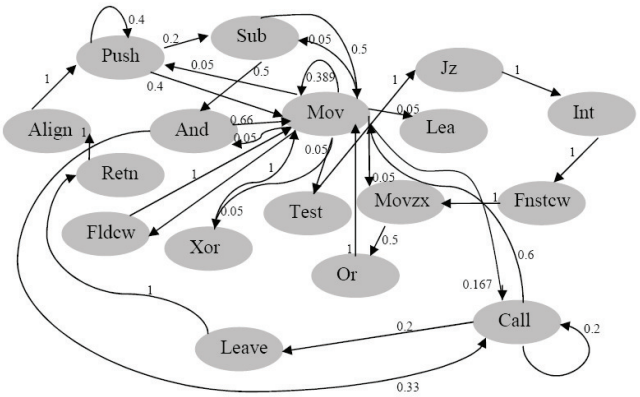
ابتدا کد اجرایی ویروس در شکل دودویی به دستورهای اسمبلی برگردانده می‌شود. نمونه‌ای از این کد در شکل ۱ دیده می‌شود.

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 ۱: نمونه‌ای از دستورها و کدهای عملیاتی یک ویروس [۹]

سپس این دستورها از ابتدا تا انتها به‌ترتیب خوانده شده و دستورها به‌شکل دوتایی جدا می‌شوند؛ یعنی هر دستور و دستور بعدی‌اش یک ترکیب دوتایی در نظر گرفته می‌شود. در خواندن دستورها فقط کد عملیاتی هر دستور خوانده می‌شود و از عملوندها صرف‌نظر می‌گردد. برای نمونه در مورد دو دستور متوالی MOV ax, bx و JNZ loop فقط کد عملیاتی آن‌ها یعنی MOV و JNZ در نظر گرفته می‌شوند.

از روی این دستورها و ترکیب‌های دوتایی، یک گراف ساخته می‌شود. هر گره این گراف یکی از دستورهاست. اگر کد عملیاتی JNZ بعد از MOV ظاهر شده باشد، یک یال از گره‌ی MOV به‌سمت JNZ کشیده می‌شود و وزن آن مقدار یک گذاشته می‌شود. اگر جای دیگری از کد ویروس، این ترکیب دوباره تکرار شود، وزن این یال یک واحد افزایش می‌یابد. در نهایت به یک گراف جهت‌دار و وزن‌دار می‌رسیم. شکل ۲ نمونه‌ای از این گراف را نمایش می‌دهد.



شکل ۲: گراف حاصل از تحلیل کدهای عملیاتی متوالی [۹]

در مرحله بعد، گراف به یک ماتریس مجاورت تبدیل می‌شود. این ماتریس مربعی است و تعداد سطرها یا ستون‌هایش برابر تعداد کدهای عملیاتی منحصر به‌فرد کد اسمبلی ویروس است. هر ترکیب دوتایی کدها در سطر و ستون‌های این ماتریس دیده می‌شود به‌طوریکه دستور قبلی در سطر و دستور بعدی در ستون قرار دارد و درایه تقاطع آن‌ها، وزن یال متناظر را نشان می‌دهد. جدول ۱ بخشی از چنین ماتریسی را به تصویر کشیده است.

[1] Uniqueness
[2] Resolution-based
[3] Hidden Markov Model (HMM)
[4] Eigenvirus
[5] Chi Squared
[6] Structural Entropy

جدول ۱: ماتریس مجاورت گراف کدهای عملیاتی [۹]

	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

سپس درایه‌های ماتریس نرمال‌سازی می‌شوند. برای این منظور، هر درایه به مجموع عناصر سطر خودش تقسیم می‌شود. این تقسیم ماتریس را به‌صورت احتمالی سطری[1] در می‌آورد. هدف این‌کار این است که اثر طول فایل در محاسبات خنثی شود و فایل‌های گوناگون از نظر عددی با هم قابل مقایسه باشند.

برای استفاده از روش شباهت گراف‌ها باید دو محدوده محاسبه شود. ابتدا گراف فایل‌های ویروس یک خانواده فراریخت ساخته شده و دو به دو با هم مقایسه می‌شوند. برای مقایسه از رابطه ۱ استفاده می‌گردد و امتیازی از مقایسه هر دو گراف به‌دست می‌آید. حداقل و حداکثر مقدار امتیازهای حاصل از مقایسه‌ها را نگهداری می‌کنیم. سپس یک مجموعه فایل سالم هم در نظر گرفته و گراف هر یک از فایل‌های سالم با گراف هر یک از ویروس‌های فراریخت هم مقایسه شده و امتیازها ثبت می‌شوند. حداقل و حداکثر این امتیازها نیز ثبت می‌شود. اکنون دو محدوده از امتیازها در اختیار است که یکی به ویروس‌ها و دیگری به فایل‌های سالم تعلق دارد.

$$score(A,B) = \frac{1}{N^2}\left(\sum_{i,j=0}^{N-1}\left|a_{ij} - b_{ij}\right|\right)^2 \qquad (۱)$$

در این رابطه N تعداد رئوس گراف یا همان سطرها یا ستون‌های ماتریس است. همچنین a_{ij} و b_{ij} درایه‌های متناظر در دو ماتریس A و B هستند. حداقل و حداکثر امتیاز حاصل از رابطه ۱ به‌ترتیب صفر و چهار است [۹]. وقتی فایل مشکوکی داده می‌شود، یکی از فایل‌های ویروس فراریخت به‌طور تصادفی انتخاب شده و گراف دو فایل با هم مقایسه می‌شود. بر حسب اینکه امتیاز شباهت در هر یک از دو محدوده قرار گیرد، فایل مشکوک به‌عنوان سالم یا ویروس دسته‌بندی می‌شود.

مشکل اینجاست که اگر دو محدوده یاد شده با هم هم‌پوشانی داشته باشند، روش در تشخیص فایل سالم از ویروس دچار اندکی شکست می‌گردد. برای حل این مشکل در این مقاله از روش وزن‌دهی استفاده می‌شود. ایده اصلی این است که برخی از زوج دستورهای متوالی در یک خانواده ویروس‌های فراریخت، فراوانی بیشتری از سایر زوج دستورها دارند. پس کافی است فراوانی هر یک از زوج دستورهای متوالی را در هر یک از ویروس‌های فراریخت یک خانواده به‌دست آورده و میانگین آن‌ها را محاسبه کنیم. این میانگین به‌عنوان وزن در رابطه مقایسه دو گراف وارد می‌شود و رابطه جدید ۲ به‌دست می‌آید:

$$score(A,B) = \frac{1}{N^2}\left(\sum_{i=0}^{N-1}\frac{\left(\sum_{j=0}^{N-1} w_{ij}\left|a_{ij}-b_{ij}\right|\right)}{\sum_{j=0}^{N-1} w_{ij}}\right)^2 \qquad (۲)$$

در این رابطه، w_{ij} وزن یک جفت کد عملیاتی i و j است که i پیش از j ظاهر شده است. این وزن متوسط فراوانی جفت کدهاست. بقیه اجزاء این رابطه همانند رابطه ۱ هستند. این رابطه کمک می‌کند که تفاوت فایل‌های ویروس و فایل‌های سالم بیشتر مشخص شود؛ یعنی هم‌پوشانی محدوده‌های از پیش محاسبه شده از بین رود یا کم شود. تأثیر نهایی آن، افزایش دقت در کشف ویروس‌های فراریخت خواهد بود.

۵- ارزیابی

به‌منظور ارزیابی روش پیشنهادی، روش شباهت گراف کدهای عملیاتی با زبان جاوا پیاده‌سازی شد. برای مقایسه فایل ویروس‌ها و فایل‌های سالم، یک‌بار از رابطه ۱ و بار دیگر از رابطه ۲ استفاده شده است. همچنین، برای استفاده از رابطه ۲ نیاز به پیش‌پردازش هم هست. لذا کدی نوشته شد که فایل ویروس‌های فراریخت خانواده تحت آزمایش را خوانده و طبق روال تشریح شده در بخش قبلی، وزن هر دو کد عملیاتی متوالی را محاسبه کرده و ذخیره نماید.

برای داده‌های آزمایش، از ۲۰۰ فایل ویروس فراریخت خانواده IDAN و ۲۰ فایل سالم اجرایی که از [۲۲] قابل دریافت هستند، استفاده شد. فهرست فایل‌های سالم در جدول ۲ ذکر شده‌اند. فایل ویروس‌ها و فایل‌های سالم به‌صورت دستورات نشان داده شده در شکل ۱ هستند و برنامه آزمایش کدهای عملیاتی را جدا کرده است.

جدول ۲: اسامی برنامه‌های سالم مورد استفاده در آزمایش‌ها

as	sleep	strip	readelf	nm
gcc	dmesg	date	touch	rm
mknod	mount	kill	file	systemctl
objdump	size	nasm	ld	grep

از آن‌جائیکه ۲۰۰ فایل ویروس و ۲۰ فایل سالم وجود دارد، برای مقایسه ویروس‌ها، ترکیب دو از ۲۰۰ یا ۱۹۹۰۰ مقایسه بین ویروس‌ها صورت گرفت. همچنین شباهت هر یک از فایل‌های ویروس با هر یک از فایل‌های سالم هم در ۴۰۰۰ مقایسه اندازه‌گیری شد. از هر اندازه‌گیری، امتیازی به‌دست آمد و همه امتیازهای هر مقایسه ثبت شدند. سپس، برای هر روش دو محدوده یکی برای فایل‌های سالم و ویروس‌ها و دیگری برای فایل‌های ویروس فراریخت محاسبه شدند. در هر محدوده، حداقل، حداکثر و میانگین امتیازها به‌دست آمد. جدول‌های ۳ تا ۶ حداقل و حداکثر امتیازها را برای فایل‌های ویروس و سالم با رابطه ۱ و رابطه ۲ نشان می‌دهند.

[1] Row Stochastic

جدول ۳: حداقل و حداکثر امتیاز شباهت بین ویروس‌ها با روش قبلی

حداقل	میانگین	حداکثر
۰/۷۵۵۴۱۶۴	۰/۲۵۶۲۰۸۶	۱/۳۱۴۵۹۱۷

جدول ۴: حداقل و حداکثر امتیاز شباهت بین ویروس‌ها و فایل‌های سالم با روش قبلی

حداقل	میانگین	حداکثر
۱/۱۵۷۸۱۶۸	۱/۳۵۲۸۷۱۶	۱/۶۳۵۵۸۸۴

جدول۵: حداقل و حداکثر امتیاز شباهت بین ویروس‌ها با روش پیشنهادی

حداقل	میانگین	حداکثر
۰/۰۰۲۲۸۱۰	۰/۰۷۰۵۷۲۳	۰/۱۹۰۰۴۳۷

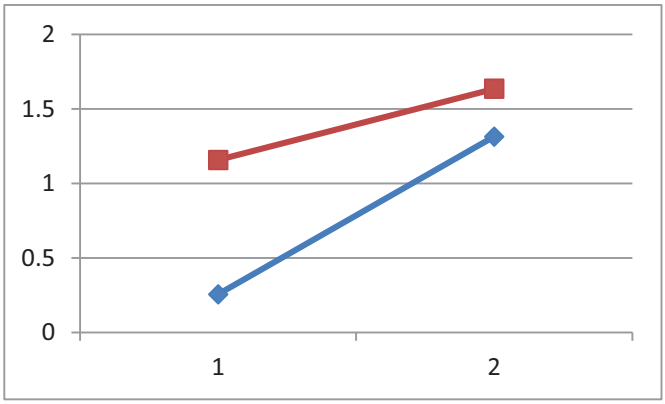
جدول ۶: حداقل و حداکثر امتیاز شباهت بین ویروس‌ها و فایل‌های سالم با روش پیشنهادی

حداقل	میانگین	حداکثر
۰/۳۸۷۵۵۵۸	۰/۵۳۶۰۵۱۱	۰/۶۵۱۷۷۹۵

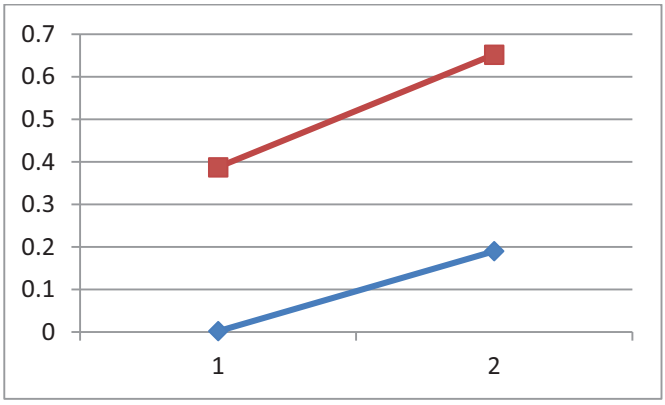
با دقت در نتایج، معلوم می‌شود که بازه آستانه حداقل و حداکثر مقدار امتیاز شباهت در جدول ۳ با بازه حداقل و حداکثر در جدول ۴ هم‌پوشانی دارند. زیرا حداکثر امتیاز شباهت در جدول ۳، مقدار حدود ۱/۳۱ و حداقل امتیاز در جدول ۴ حدود ۱/۱۵ است. پس اگر امتیاز شباهت یک فایل مشکوک ورودی برای مثال ۱/۲ شود، روش قادر به دسته‌بندی آن نیست.

اما با دقت در نتایج جدول‌های ۵ و ۶ مشاهده می‌شود که بازه حداقل و حداکثر امتیاز در جدول ۵ هیچ اشتراکی با محدوده امتیاز در جدول ۶ ندارد. پس مقایسه نتایج این دو جفت جدول، تأثیر استفاده از وزن‌دهی را در رابطه محاسبه شباهت آشکار می‌سازد. به‌عبارت بهتر، بعد از اینکه فایل مشکوکی داده شد، یکی از فایل‌های ویروس فراریخت به تصادف انتخاب شده و با در نظر گرفتن رابطه ۲ و وزن‌های از پیش محاسبه شده، امتیاز شباهت این فایل مشکوک محاسبه می‌گردد. چون دو محدوده جدول ۵ و ۶ از هم جداست، فایل ورودی منحصراً در یکی از دو محدوده جدول ۵ یا جدول ۶ قرار می‌گیرد. در نتیجه ابهامی برای دسته‌بندی فایل مشکوک ورودی به‌عنوان سالم یا ویروس وجود نخواهد داشت.

برای درک بهتر تفاوت دو محدوده، شکل‌های ۳ و ۴ دو محدوده را به‌تصویر کشیده‌اند. در این دو شکل، محور عمودی مقدار آستانه‌ها را نشان می‌دهد. همچنین، خط بالایی مربوط به محدوده مقایسه فایل‌های ویروس و سالم است و خط پایینی محدوده مربوط به مقایسه فایل‌های ویروس با یکدیگر را مشخص می‌کند. شکل ۳ مقایسه محدوده‌ها با رابطه ۱ و شکل ۴ نیز محدوده‌های حاصل از رابطه ۲ را نشان می‌دهد. ملاحظه می‌شود که دو خط شکل ۳ در محور عمودی با هم هم‌پوشانی دارند ولی دو خط شکل ۴ کاملاً از هم جدا هستند.



شکل ۳: مقایسه محدوده شباهت فایل‌های ویروس با هم (خط پایینی) و محدوده شباهت فایل‌های ویروس و سالم با هم (خط بالایی) با استفاده از رابطه ۱ (رابطه موجود)



شکل ۴: مقایسه محدوده شباهت فایل‌های ویروس با هم (خط پایینی) و محدوده شباهت فایل‌های ویروس و سالم با هم (خط بالایی) با استفاده از رابطه ۲ (رابطه پیشنهادی)

۶- نتیجه‌گیری

در این مقاله یکی از روش‌های اخیر در کشف ایستای ویروس‌های فراریخت تشریح شد. مشکل این روش در هم‌پوشانی احتمالی محدوده تشخیص ویروس از فایل سالم است. با وزن‌دهی فراوانی زوج دستورها، رابطه مقایسه شباهت گراف کدها توسعه داده شد. برای ارزیابی روش، آزمایش‌هایی روی یکی از خانواده ویروس‌های فراریخت رایج در مطالعات صورت گرفت. آزمایش‌ها روش قبلی و روش پیشنهادی را مقایسه کرده است. نتایج آزمایش‌ها کارایی روش پیشنهادی را در افزایش دقت کشف بر مبنای شباهت کدهای عملیاتی نشان می‌دهد.

مراجع

[1] McAfee Labs. Mcafee threats report: First quarter 2013. Technical report, McAfee, 2013.
[2] J. Aycock, *Computer Viruses and Malware,* Advances in Information Security, Vol. 22, Springer-Verlag, 2006.
[3] A. H. Toderici and M. Stamp, “Chi-squared distance and metamorphic virus detection,” *J. Comput. Virol.* Vol. 9, No. 1, pp. 1-14, 2013.

Computers, Tenerife, Spain, December 16-18, pp. 265-270, 2005.

[16] 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.

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

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] Mark Stamp Website in San Jose State University, [Online], http://cs.sjsu.edu/~stamp/viruses/

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

[5] 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.

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

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

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

[9] 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.

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

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

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

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

[14] 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.

[15] 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