مقایسه کارایی روش جدید نقش-آپکد در مقایسه با آپکد، به منظور تشخیص بدافزارهای کامپیوتری

زهرا قزل بیگلو[1]*، مجید وفایی جهان[2]

* مسئول مکاتبات:

[1] گروه کامپیوتر، دانشگاه بین المللی امام رضا (ع) ، مشهد، ایران
zahra.ghezelbigloo@yahoo.com

[2] استادیار، گروه کامپیوتر، دانشگاه آزاد اسلامی مشهد، مشهد، ایران
vafaeijahan@mshdiau.ac.ir

چکیده

یکی از روش‌های متداول در زمینه مقابله با بدافزارها استفاده از دنباله آپکدهای موجود در کد اسمبلی بدافزارها است. در این مقاله از روشی جدید مبتنی بر دسته‌بندی ساختاری آپکدها برای تشخیص بدافزارها استفاده شده است و کارایی آن نسبت به روش آپکدها مورد ارزیابی قرار گرفته است. برای این منظور از دو روش مختلف برای استخراج ویژگی مبتنی بر محتوای اسمبلی فایل‌ها استفاده شده است. سپس هر دو روش در شرایط برابر، با استفاده از دسته‌بندهای مختلف مورد آزمایش قرار گرفته‌اند. نتایج آزمایشات نشان می‌دهد که با استفاده از دسته‌بندی ساختاری آپکدها کارایی و صحت دسته‌بندهای مختلف در تشخیص بدافزارها کاهش نمی‌یابد، علاوه بر این تعداد ویژگی‌ها، حجم محاسبات، حافظه و زمان مصرفی به طور قابل توجهی کاهش می‌یابد.

کلمات کلیدی:

آپکد، نقش-آپکد، دسته بندی، صحت، زمان ساخت مدل.

۱) مقدمه

همان‌گونه که نرم افزارها در حال تکامل هستند، برخی از توسعه دهندگان اقدامات امنیتی را پیاده سازی می‌کنند تا اطمینان حاصل نمایند که محصولات آن‌ها از امنیت بالاتری برخوردار است. با این وجود بدافزارها به طور فزاینده ای سیستم‌ها را تهدید می‌کنند. همان‌طور که فناوری‌ها برای مقابله با بدافزارها پیشرفت می‌کند، بدافزارها نیز برای نفوذ به کدها تکامل می یابند.

اسکن کردن و روش‌های مبتنی بر امضاء از جمله اولین روش‌هایی بودند که برای مقابله با انواع بدافزارها تولید شدند، این روش به صورت تجاری در انواع ضد بدافزارها و ضد ویروس‌ها مورد استفاده قرار گرفت. اما یکی از مشکلات برنامه‌های کاربردی همچون ضد ویروس، کشف و از بین بردن ویروس‌ها پس از اجرا و یا ظاهر شدن در کامپیوترها است. علاوه بر آن به دلیل اینکه آسیب پذیری روش‌های مبتنی بر امضاء بسیار بالا بود، این روش برای ویروس‌های نسل بعد کارایی چندانی نداشت.

پس از آن روش‌های مبتنی بر کد برای مقابله با بدافزارها مطرح شد. در این روش‌ها سعی بر این بود که با تحلیل فایل‌ها در سطح کد، بتوان انواع کدهای مخرب، ویروس‌ها و ... را قبل از هر نوع اقدامی تشخیص داد. تلاش‌های بسیار زیادی در رابطه با تشخیص کدهای مخرب با استفاده از تحلیل و آنالیز آماری بر روی فایل‌ها در سطح کدهای باینری صورت گرفته است (Tabish, Shafiq, & Farooq, 2009; Jochheim, October 17, 2012; Kaushal, Swadas, & Prajapati, 2012). در ادامه تحلیل بدافزارها در سطح کدهای باینری، ایده استفاده از آپکدهای[1] موجود در کد اسمبلی مطرح شد.

[1] Opcode

یکی از مهم‌ترین اقدامات انجام شده در راستای بررسی آپکدها و تشخیص بدافزارهای کامپیوتری در سال ۲۰۰۷ توسط "بیلار" و همکارانش انجام شد. آن‌ها در مقاله خود به بررسی انواع آپکدها شامل نمونه‌های پر کاربرد و نمونه‌های نادر پرداخته و نشان دادند فراوانی آپکدهای نادر در فایل‌های بدافزار در مقایسه با فایل‌های سالم، تفاوت بیشتری نسبت به آپکدهای پرکاربرد دارد (Bilar, 2007). پس از آن "مسکوویچ" در سال ۲۰۰۸ از مدل n-گرم آپکدها برای تشخیص بدافزارهای کامپیوتری استفاده کردند. آن‌ها در مقاله خود نشان دادند که در صورتی که درصد فایل‌های بدافزار نزدیک به ۱۵٪ کل نمونه‌ها باشد می‌توان به دقت بالای ۹۹٪ دست یافت (Moskovitch, et al., 2008).

"سنتوس" و همکارانش در سال ۲۰۱۰ یک سیستم جهت تشخیص انواع بدافزارها مبتنی بر آپکد ارائه دادند (Santos, Brezo, Nieves, Penya, & Sanz, 2010). وی در سال بعد تحقیقات خود را در رابطه با تشخیص بدافزارها مبتنی بر روش‌های یادگیری یک-کلاس و نیمه نظارتی بر پایه آپکدها ادامه داد و به نتایج مطلوبی دست یافت (Bringas, 2011; Igor Santos, 2011). آن‌ها در سال ۲۰۱۳ اقدامات خود در زمینه تشخیص بدافزارها را کامل کردند. آن‌ها بر اساس فراوانی دنباله آپکدهای ظاهر شده در متن، روشی برای بررسی و سنجش ارتباط فراوانی هر دنباله از آپکدها ارائه نمودند. در این مقاله از متدهای داده‌کاوی جهت تشخیص بدافزارها استفاده شد (Santos, Brezo, Ugarte-Pedrero, & Bringas, 2013).

در این مقاله آپکدهای استخراج شده از کد اسمبلی هر نمونه به صورت ساختاری و مطابق با نقش آن‌ها در ۱۰ دسته طبقه‌بندی می‌شوند. سپس به جای استفاده از دنباله آپکدها و آنالیز بر روی آن‌ها از دنباله نقش-آپکدها استفاده می‌شود. به منظور ارزیابی کارایی روش ارائه شده از دو رویکرد مختلف برای استخراج ویژگی مبتنی بر محتوای اسمبلی فایل‌ها استفاده شده است. روش اول مبتنی بر فراوانی آپکدهای ظاهرشده در متن کد است و روش دوم مبتنی بر فراوانی نقش-آپکدهای متوالی ظاهرشده در متن کد است. پس از آن هر دو روش در شرایط برابر و با استفاده از دسته‌بندهای مختلف مورد آزمایش قرار گرفته‌اند. تقسیم بندی مقاله به شرح زیر می‌باشد:

در بخش بعد روش نقش-آپکد شرح داده می‌شود. نحوه استخراج ویژگی و دو رویکرد مختلف در رابطه با استخراج ویژگی‌ها در بخش ۳ ارائه شده است. در بخش ۴ تحلیل و مقایسه روش‌های عنوان شده در مقاله آورده شده است و در آخر در بخش ۵، نتیجه گیری مقاله عنوان شده است.

۲) تعریف نقش-آپکد

به منظور استخراج نقش-آپکد ابتدا هر فایل با استفاده از دی اسمبلر به رشته‌ای از دستورات ماشین تحت عنوان کد اسمبلی تبدیل می‌شود. این دستورات شامل دو بخش عملوند و آپکد می‌باشند. مثال ذکر شده در ادامه یک نمونه کد اسمبلی مربوط به یک فایل اجرایی را نشان می‌دهد.

```
pushesi
call  SUB_L0100D36F
pop  ecx
test  eax,eax
jnz  L0100D281
push0000001Ch
call  SUB_L0100D330
pop  ecx
```

به دلیل اینکه آپکدها به تنهایی نیز می‌توانند گویای اعمال یک فایل باشند از آنالیز بخش عملوندها صرف‌نظر شده و تنها دنباله آپکدهای هر فایل استخراج می‌شود. در این صورت رشته آپکد استخراج شده برای مثال بالا به صورت {push, call, pop, test, jnz, push, call, pop} خواهد بود. در اینجا از نوعی دسته‌بندی ساختاری بر اساس وظیفه و نقش تعریف شده هر آپکد استفاده شده است.

در این روش ابتدا نقش‌های مختلف موجود برای آپکدها تعریف می‌شود، سپس آپکد ها در دسته‌هایی متناسب با نقش (وظیفه) هر آپکد دسته‌بندی می‌شوند. برای این منظور پس از مشخص شدن نوع آپکدها و تعداد نقش‌های مورد نظر مطابق با منابع معتبر، آپکدها در گروه‌های مربوط به خود دسته‌بندی می‌شود (MazeGen, 2009; x86 instruction listings, 2014). برای مثال مطابق با تعریف ارائه شده در مرجع راهنمای آپکد X86 [1]، نقش و وظیفه آپکدهای "And" و "Or" عملیات منطقی [2] است (X86 Opcode, 2013). در این مقاله به هر آپکد یک عدد به عنوان نقش آن اختصاص داده می‌شود. جدول ۱ نقش‌های تعریف شده در این مقاله را با ذکر نمونه آپکدهای موجود عنوان می‌کند.

[1] X86 Opcode and Instruction Reference Home
[2] Logical

جدول ۱) نقش-آپکدها تعریف شده

شماره	نقش‌های تعریف شده	نمونه آپکد
۱	آپکدهای مربوط به عملیات منطقی	Add, or
۲	آپکدهای مرتبط با حافظه	Mov, stos
۳	آپکدهای مرتبط با پشته	Push, pop
۴	آپکدهای مربوط به عملیات کنترلی و وظعیتی	Test, call
۵	آپکدهای پیشوندی	Fs, repe
۶	آپکدهای مربوط به عملیات سیستمی و ورودی/خروجی	In, out
۷	آپکدهای چند ساختاری و توسعه یافته	Nop, fpu
۸	آپکدهای مربوط به انجام عملیات منطقی که با حرف "f" شروع می‌شوند.	Fmul, fsub
۹	آپکدهای خاص	Punpckhbw
۱۰	سایر آپکدها	…

۳) استخراج ویژگی‌ها

استخراج ویژگی‌ها به عنوان یکی از مهم‌ترین بخش‌ها در تشخیص، دسته‌بندی، پیش‌بینی و سایر روش‌های داده کاوی می‌باشد. عموماً با افزایش تعداد ویژگی‌ها ابعاد فضای ویژگی‌ها افزایش می‌یابد و در پی آن پیچیدگی مسئله افزایش خواهد یافت به همین منظور یافتن کمترین تعداد ویژگی‌ها و اثر گذارترین آن‌ها همواره یکی از چالش‌های مهم در مسائل داده کاوی بوده است. در این مقاله سعی بر این است که با کمترین تعداد ویژگی و کاهش بار محاسباتی مسئله بتوان موثرترین ویژگی‌ها را در تشخیص بدافزارها از فایل‌های اجرایی یافت. در هر دو روش از هیچ گونه روش خاص برای استخراج ویژگی‌ها استفاده نخواهد شد.

- **نقش-آپکد**

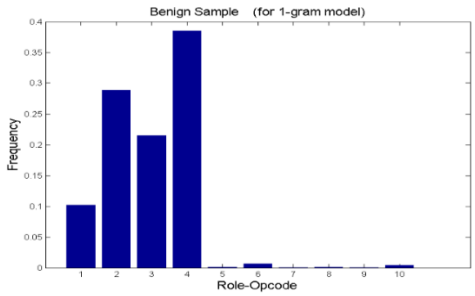
در این مقاله پس از استخراج دنباله نقش-آپکدها مطابق با روش ارائه شده در بخش سوم، فراوانی دنباله نقش-آپکدهای متوالی به طول n، مشاهده شده در متن کد اسمبلی هر فایل به عنوان ویژگی مربوط به آن فایل، مورد استفاده قرار گرفته است. برای مثال برای دنباله نقش-آپکدها به طول ۲ مربوط به مثال ذکر شده در بخش سوم برابر است با

{(push,call),(call,pop),(Pop,test),(Test,jnz),(Jnz,push),(Push,call),(call,pop)}

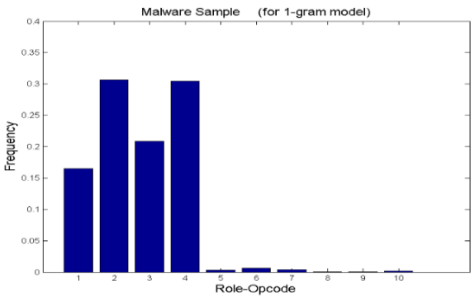
که پس از تبدیل به نقش-آپکد به صورت دنباله زیر خواهد شد.

{(3-4),(4-3),(3-4),(4-4),(4-3),(3-4),(4-3)}

بنابراین فراوانی مدل n-گرم (زیر دنباله‌های متوالی با طول n) از نقش-آپکدها به عنوان ویژگی، مورد تجزیه و تحلیل قرار خواهد گرفت. برای وضوح بیشتر فراوانی زیر دنباله‌های متوالی از نقش-آپکدها با طول یک و دو توضیح داده شده است. با این توصیف برای مدل ۱-گرم، تنها ۱۰ ویژگی استخراج خواهد شد. تصاویر ۱و ۲ میانگین مقدار فراوانی برای زیر دنباله‌های با طول یک را نشان می‌دهند. با توجه به شکل مشاهده می‌شود، فراوانی آپکدهای پیشوندی و آپکدهایی که وظیفه آن‌ها انجام عملیات منطقی است، در بدافزارها بیشتر از فایل‌های سالم می‌باشد. علاوه بر این می‌توان گفت در بدافزارها تعداد آپکدهایی که وظایف آن‌ها انجام عملیات منطقی است و آپکدهایی که وظایف آن‌ها مربوط به پشته است تقریباً با یکدیگر برابر است، این در حالی است که در فایل‌های سالم آپکدهایی که مرتبط با پشته هستند به نسبت بیشتر از آپکدهایی هستند که وظیفه آن‌ها انجام عملیات منطقی است.



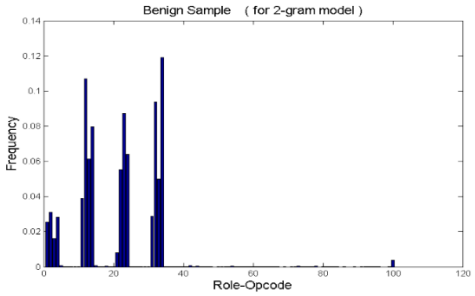
شکل ۱) میانگین فراوانی مدل ۱-گرم از نقش-آپکدها (فایل سالم)



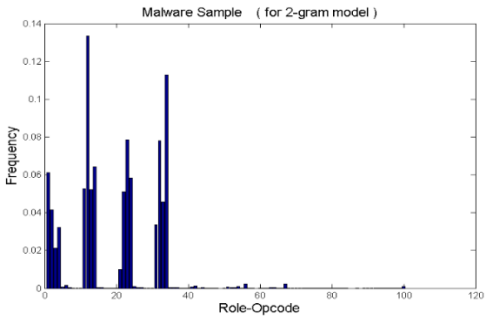
شکل ۲) میانگین فراوانی مدل ۱-گرم از نقش-آپکدها (فایل بدافزار)

برای مدل ۲-گرم (زیر دنباله‌های به طول ۲)، تعداد ویژگی‌های استخراج شده در کل ۱۰۰ نمونه می‌باشد که در بعضی موارد هیچ یک از آن‌ها در کل نمونه‌ها مشاهده نشده‌اند. در این مقاله از مجموع ۱۰۰ ویژگی ۹۷ مورد در نمونه‌های مورد آزمایش مشاهده شده است. تصاویر ۳ و ۴

میانگین فراوانی زیر دنباله از نقش-آپکدها به طول ۲ را نشـان می‌دهند.



شکل ۳) میانگین فراوانی مدل ۲-گرم از نقش-آپکدها (فایل سالم)

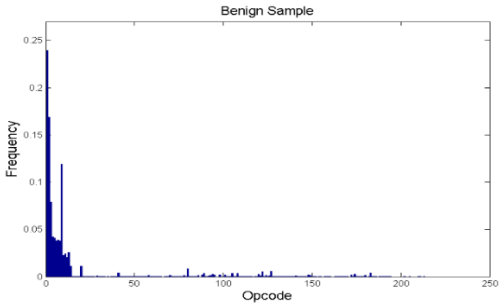


شکل ۴) میانگین فراوانی مدل ۲-گرم از نقش-آپکدها (فایل بدافزار)

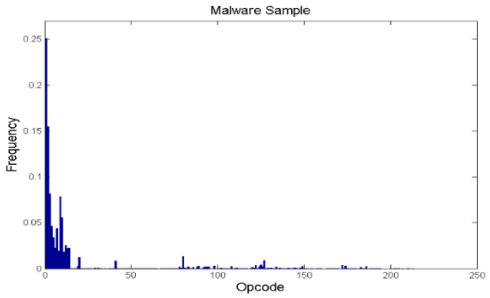
با توجه به شکل ۳ و ۴ مشاهده مـی‌شـود در بسـیاری از موارد میانگین فراوانی نقـش-آپکـدها بـه طـول ۲ بسـیار کـم هسـتند، بـا ایـن وجـود تفاوت‌هـایی بـین فایل‌هـای سـالم و بدافزارها وجود دارد. برای مثال فراوانی زوج نمونه‌های متوالی از آپکدهای مربوط به عملیات منطقـی و عملیـات مـرتبط بـا حافظه در بدافزارها خیلی بیشتر از فایل‌هـای سـالم اسـت در حالی که فراوانی زوج نمونه‌های متـوالی از آپکـدهایی کـه بـه ترتیب عملیات مربوط به حافظه-کنترلی و نیز کنترلی-حافظه را بر عهده دارند در فایل‌های سالم بیشتر از فایل‌های اجرایـی است.

- **آپکد**

در روش دوم تنها رشته آپکدهای موجود اسـتخراج شـده و از فراوانی آن‌ها به منظور مقایسه با روش اول استفاده مـی‌کنیم. تعـداد آپکـدهای اسـتخراج شـده در ایـن مقالـه ۲۱۵ مـورد می‌باشد. مطابق با حالـت قبـل بـا اسـتفاده از مـدل n-گـرم، فراوانی زیر دنباله ها با طول مشخص استخراج می‌شود. بـرای مثال در مـدل ۱-گـرم، ۲۱۵ حالـت وجـود دارد تصـاویر زیـر میانگین فروانی زیر دنباله از آپکـدها بـه طـول یـک را نشـان می‌دهد. آپکدها با شماره های بین ۱ تا ۱۰ شامل آپکدهای پر کاربرد می‌باشند و پس از آن مطابق با حروف الفبای انگلیسـی شماره گذاری شده اند. با توجه به شـکل‌هـای ۵ و ۶ مشـاهده می‌شود که میانگین فراوانی برای آپکدهایی با شماره های بین ۱۰ تا ۲۰ در فایل‌های سالم و بدافزار به نسبت سـایر آپکـدها، تفاوت بیشتری دارد، در حالی که میانگین فراوانـی آپکـدهای پرکاربرد تفاوت کمتری دارند.



شکل ۵) میانگین فراوانی مدل ۱-گرم از آپکدها (فایل سالم)



شکل ۶) میانگین فراوانی مدل ۱-گرم از آپکدها (فایل بدافزار)

۴) مقایسه روش‌های نقش-آپکد و آپکد

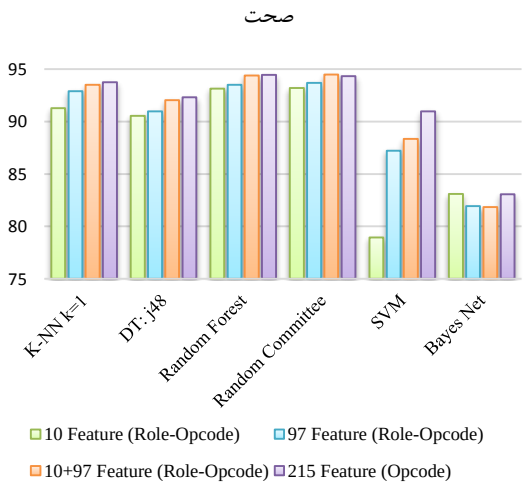
در این مقاله از پایگاه داده‌ای شامل ۳۴۴۵ نمونه فایل بـدافزار و ۱۴۹۷ نمونه فایل سالم استفاده شده اسـت کـه در مجمـوع برابر با ۴۹۴۲ نمونـه می‌باشـد. فایل‌هـای سـالم توسـط آنتـی ویروس مربوط به شرکت ماکروسافت مورد آزمایش قرار گرفته است (kaspersky lab, 2013). مجموعه بدافزارها از چنـدین سـایت معتبـر همچـون virusshare، Malwarelu و Malshare است که در ۶ ماه اول سال ۲۰۱۳ دریافـت شـده اسـت (Roberts, 2013; Wood, 2013; Rascagneres, 2013).

در روش اول از هر نمونه دنباله آپکدهای ظاهر شده اسـتخراج شده است که در مجموع ۲۱۵ مورد آپکد مشاهده شده اسـت. پس از استخراج دنبالـه آپکـدها فراوانـی احتمـالی هـر نمونـه محاسبه شده است. ابعاد فضای ویژگـی در روش اول برابـر بـا

۲۱۵ است. در روش دوم پس از استخراج دنباله آپکدها از هر نمونه، دنباله نقش-آپکدها مطابق با روش ارائه شده در بخش ۲ و ۳ به‌دست می‌آید. سپس فراوانی احتمالی زیر دنباله از نقش_آپکدهای متوالی به طول یک و دو محاسبه می‌گردد. آزمایشات به طور جداگانه بر روی سه حالت ۱-گرم با ۱۰ ویژگی، ۲-گرم با ۹۷ ویژگی و ترکیب مدل۱-گرم و ۲-گرم با ۱۰۷ ویژگی انجام گرفته است. به منظور مقایسه دو روش ارائه شده در این مقاله از دو معیار صحت[1] و زمان مورد نیاز برای ساخت مدل استفاده شده است. کلیه نتایج به دست آمده با استفاده از نرم افزار داده کاوی وکا[2] صورت گرفته است. سیستم مورد استفاده دارای ۸ گیگابایت رم، پردازش‌گر ۷ هسته‌ای شرکت اینتل[3] و ویندوز ۸ کامل ۶۴ بیتی است.

• صحت دسته‌بندی

شکل ۷ میزان صحت و دقت دسته‌بندهای مختلف را به ازای تعداد ویژگی‌های متفاوت و دو روش ارائه شده نشان می‌دهد. با توجه به تصویر کاملا مشخص است که در بسیاری از موارد نتایج حاصل از روش آپکدها و نتایج مربوط به روش نقش آپکد (برای حالت ۱۰۷ ویژگی) بسیار به هم نزدیک است. اما روش نقش آپکد در رابطه با دسته بند ماشین بردار پشتیبان نتایج مطلوب تری دارد.

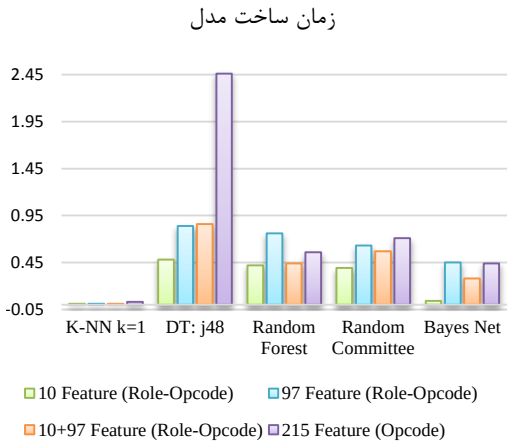


شکل ۷) نمودار مربوط به صحت دسته بندهای مختلف

بهترین نتیجه مشاهده شده در این نمودار مربوط به روش نقش-آپکد با استفاده از دسته بند تجمع تصادفی[4] و به ازای ۱۰۷ ویژگی می‌باشد که برابر است با ۹۴,۵٪ است. و بهترین نتیجه مشاهده شده برای روش آپکدها مربوط به دسته بند جنگل تصادفی[5] و برابر با ۹۴,۵٪ به ازای ۲۱۵ ویژگی، می‌باشد.

• مدت زمان ساخت مدل

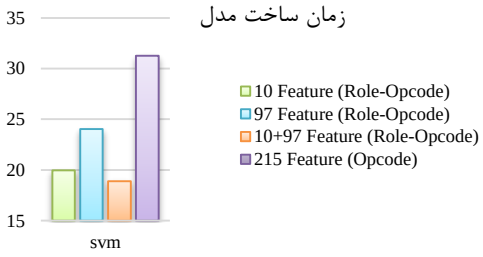
شکل ۸ نمودار مربوط به زمان مورد نیاز برای ساخت مدل در دسته‌بندهای مختلف را نشان می‌دهد. برای وضوح بیشتر اعداد و تفاوت زیاد نتایج مربوط به دسته بند ماشین بردار پشتیبان، نتایج حاصل از این دسته بند به صورت جدا گانه در تصویر ۹ ارائه شده است.



شکل ۸) نمودار مربوط به زمان مورد نیاز برای ساخت مدل در دسته بندهای مختلف

کمترین زمان مورد نیاز برای ساخت مدل با توجه به شکل ۸ مربوط به روش نقش آپکد و به ازای ۱۰ ویژگی بوده است که به دلیل پایین بودن تعداد ویژگی‌ها بسیار طبیعی می‌باشد. از نکات قابل توجه در این شکل کاهش زمان ساخت در این روش به ازای ترکیب ویژگی‌های مربوط به مدل ۱-گرم و ۲-گرم است. در این حالت با وجود اینکه تعداد ویژگی‌ها به ۱۰۷ مورد افزایش یافته است اما زمان ساخت مدل نسبت به حالت ۹۷ ویژگی کمتر است. بیشترین زمان سپری شده نیز مربوط به روش آپکدها می‌باشد. یکی از نکات قابل توجه در تصویر ۹، علاوه بر اینکه مدت زمان ساخت مدل برای حالت ترکیب ویژگی‌های مدل‌های ۱-گرم و ۲-گرم (مشابه با تصویر ۸) از زمان ساخت مدل در روش‌آپکدها با ۲۱۵ ویژگی و نقش-آپکدها مربوط به مدل ۲-گرم کمتر است، نسبت به زمان مورد نیاز برای ساخت مدل ۱-گرم نیز کمتر است.

[1] Accracy
[2] Weka
[3] Intel core i7
[4] Random Committee
[5] Random Forect



شکل ۹) نمودار زمان ساخت مدل برای دسته‌بند ماشین بردار پشتیبان

۵) نتیجه گیری

در این مقاله به منظور ارزیابی عملکرد و کارایی روش جدید مبتنی بر نقش و وظیفه آپکدها، تحت عنوان نقش-آپکد، این روش با روش مشابه خود با نام آپکد مقایسه شده است. برای مقایسه این دو روش از دو رویکرد مختلف برای استخراج ویژگی مبتنی بر محتوای اسمبلی فایل‌ها استفاده شده است. روش اول مبتنی بر فراوانی آپکدهای ظاهرشده در متن کد است و روش دوم مبتنی بر فراوانی نقش-آپکدهای متوالی ظاهرشده در متن کد است. پس از آن هر دو روش در شرایط برابر و با استفاده از دسته‌بندهای مختلف مورد آزمایش قرار گرفته‌اند. با توجه به اینکه تعداد آپکدها در منابع مختلف به طور میانگین ۲۵۰ مورد می باشد، با افزایش مقدار n در مدل n-گرم میزان تعداد ویژگی‌ها، حافظه مصرفی، پیچیدگی محاسبات در بخش انتخاب ویژگی و زمان مورد نیاز برای ساخت مدل به طور نمایی افزایش می‌یابد، اما در روش جدید نقش-آپکد به دلیل کم بودن تعداد نقش‌ها، تعداد ویژگی‌ها، حافظه مصرفی، پیچیدگی محاسبات و زمان مورد نیاز برای ساخت مدل به طور قابل توجهی کاهش می‌یابد. نتیجه آزمایشات مربوط به دقت دسته بندهای مختلف نشان می‌دهد، روش آپکدها علیرغم اینکه با جزئیات بیشتری به دسته بندی نمونه‌ها می‌پردازد، اما برتری خاصی نسبت به روش نقش-آپکد ندارد. بنابراین می‌توان امیدوار بود با استفاده از روش نقش-آپکد و محاسبه مقادیر بزرگ‌تر n و اعمال روش‌های انتخاب ویژگی مناسب به نتایج مطلوبی در تشخیص انواع بدافزارهای کامپیوتری دست یافت.

مراجع

Bilar, D. (2007). OpCodes as predictor for malware. *International Journal of International Journal of, 1*(2), 156-168.

Bringas, I. S. (2011). Using opcode sequences in single-class learning to detect unknown malware. *IET Information Security*.

Igor Santos, B. S. (2011). Opcode-sequence-based Semi-supervised Unknown Malware Detection. *Computational Intelligence in Security for Information Systems, 6694*, 50-57.

Jochheim, B. (October 17, 2012). On the Automatic Detection of Embedded Malicious Binary Code using Signal Processing Techniques Project Report. Project, Hamburg University of Application Sciences, Hamburg.

kaspersky lab. (2013). Retrieved from http://www.kaspersky.com/pure

Kaushal, K., Swadas, P., & Prajapati, N. (2012, July). Metamorphic Malware Detection Using Statistical Analysis. *International Journal of Soft Computing and Engineering (IJSCE), 2*(3), 49-53.

MazeGen. (2009, 1 20). *X86 Opcode and Instruction Reference*. Retrieved from http://ref.x86asm.net/

Moskovitch, R., Feher, C., Tzachar, N., Berger, E., Gitelman, M., Dolev, S., & Elovici, Y. (2008). Unknown Malcode Detection Using OPCODE Representation. *1st European Conference on Intelligence and Security Informatics*, (pp. 204-215).

Rad, B. B., Masrom, M., & Ibrahim, S. (2012). Opcodes Histogram for Classifying Metamorphic Portable Executables Malware. *International Conference on e-Learning and e-Technologies in Education (ICEEE)*.

Rascagneres, P. (2013, 3). *malware.lu*. Retrieved from http://www.malware.lu/pages/company.html

Roberts, J.-M. (2013, 4 24). *VirusShare*. Retrieved from http://virusshare.com/

Runwal, N., Low, R. M., & Stamp, M. (2012). Opcode graph similarity and metamorphic detection. *Journal in Computer Virology, 8*(1-2), 37-52.

Santos, I., Brezo, F., Nieves, J., Penya, Y. K., & Sanz, B. (2010). Idea: Opcode-sequence-based Malware Detection. *Engineering Secure Software and System*.

Santos, I., Brezo, F., Ugarte-Pedrero, X., & Bringas, P. G. (2013). Opcode sequences as representation of executables for data-mining-based unknown malware detection. *Information Sciences, 231*, 64-82.

Shabtai, A., Moskovitch, R., Feher, C., Dolev, S., & Elovici, a. Y. (2012). Detecting unknown malicious code by applying classification techniques on OpCode patterns. *Security Informatics*.

Tabish, S. M., Shafiq, M. Z., & Farooq, M. (2009). Malware detection using statistical analysis of byte-level file content. *ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics*, (pp. 23-31). New York, NY, USA.

Wood, D. (2013, 4). *VirusSign*. Retrieved from http://www.virussign.com/downloads.html

x86 instruction listings. (2014, Junuary 3). Retrieved from http://en.wikipedia.org/wiki/X86_instruction_listings

X86 Opcode. (2013). Retrieved from X86 Opcode and Instruction Reference Home: http://ref.x86asm.net/coder32.html