پنجمین کنفرانس انجمن رمز ایران
۲۴ الی ۲۶ مهر ماه ۱۳۸۶ دانشگاه علم و صنعت ایران
انجمن رمز ایران
Iranian Society of Cryptology

ارائه روشی جدید در نقش‌آب‌زنی کلاس‌های جاوا با استفاده از بلاک‌های ساختگی

سعید جلیلی
دانشگاه‌تربیت مدرس ،گروه کامپیوتر،آزمایشگاه امنیت و تشخیص نفوذ
sjalili@modares.ac.ir

بی‌بی‌سمانه حسینی مقدم
دانشگاه‌تربیت مدرس ،گروه کامپیوتر،آزمایشگاه امنیت و تشخیص نفوذ
s_hosseini@modares.ac.ir

چکیده: نقش‌آب‌زنی نرم‌افزار نوعی پنهان‌نگاری نرم‌افزار است که اطلاعاتی راجع به خود نرم‌افزار را در آن تعبیه می‌نماید. تفاوت در نوع اطلاعات درج شده منجر به کاربری‌های مختلف می‌شود. این کاربری‌ها از درج مشخصات تولید کننده نرم‌افزار در راستای حفاظت از حق نشر نرم‌افزار تا درج اطلاعات وارسی یکپارچگی نرم‌افزار در راستای حفاظت از دستکاری خرابکارانه نرم‌افزار، می‌تواند متفاوت باشد. این مقاله به معرفی روشی جدید در نقش‌آب‌زنی کلاس‌های جاوا می‌پردازد که بر اساس ایده درج بلاک‌های ساختگی توسعه یافته‌است. نتایج حاصل از ارزیابی‌های انجام شده نشان می‌دهد که این روش از لحاظ نرخ داده قابل درج در کلاس‌های جاوا و میزان‌نهفتگی نسبت به سایر روش‌های نقش‌آب‌زنی دارای برتری‌های قابل توجهی است. به علاوه برخلاف سایر روش‌ها امکان ذخیره اندیس قطعات نقش‌آب را فراهم می‌آورد که باعث افزایش مقاومت آن در برابر حملات می‌شود.

واژه‌های کلیدی: نقش‌آب‌زنی نرم‌افزار، محافظت از نرم‌افزار، بلاک‌ساختگی، میزان‌نهفتگی، چارچوب ارزیابی

۱- مقدمه

نقش‌آب‌زنی نرم‌افزار یکی از روش‌های نوظهور برای حفاظت از نرم‌افزار است که امکان تعبیه یک سری اطلاعات محرمانه در نرم‌افزار را فراهم می‌آورد. نحوه تعبیه‌اطلاعات باید به گونه‌ای باشد که اولا هیچ‌گونه خدشه‌ای به وظیفه‌مندی‌های نرم‌افزار وارد نشود و ثانیا به صورت یکتایی قابل بازیابی باشد. به‌این اطلاعات محرمانه نقش‌آب گفته می‌شود. اگر چه جز تعداد معدودی [۱،۲،۳،۴] در اکثر مقالات و نوشتجات علمی، نقش‌آب حامل اطلاعاتی درباره هویت تولید کننده نرم‌افزار در راستای اثبات حق مالکیت آن، بوده‌است [۵،۶،۷،۸،۹،۱۰،۱۱،۱۲]؛ اما این اطلاعات علاوه بر مشخصات تولیدکننده و یا تکثیرکننده نرم‌افزار می‌تواند شامل داده‌های مربوط به آزمون یکپارچگی نرم‌افزار، قوانین کنترل دسترسی، آزمون اعتبار نرم‌افزار و یا کلید رمزگشایی قسمتی از نرم‌افزار باشد[۴].

در هر یک از کاربردهای نقش‌آب‌زنی نرم‌افزار انتظار می‌رود نقش‌آب، دارای ویژگی‌هایی متفاوت با کاربرد دیگر باشد. به

عنوان مثال در کاربردی که از نقش‌آب‌زنی برای درج مشخصات تولید کننده و یا تکثیر کننده نرم‌افزار استفاده می‌شود، انتظار می‌رود نقش‌آب در برابر انواع دستکاری‌های نرم‌افزار برگشت پذیر باشد. مقالات متعددی به این دسته از کاربردهای نقش‌آب‌زنی پرداخته‌اند. در مقابل در کاربردهایی مانند آزمون یکپارچگی و اعتبار نرم‌افزار و همچنین درج قوانین دسترسی به منابع، انتظار می‌رود نقش‌آب در برابر برخی دستکاری‌های نرم‌افزار شکننده و غیرقابل بازیابی باشد تا از این طریق بتوان عدم اعتبار نرم‌افزار را تشخیص داد. اما به‌طورکلی در تمامی‌کاربردها انتظار می‌رود روش نقش‌آب‌زنی از میزان نرخ داده کافی برای درج نقش‌آب مورد نظر و همچنین حداکثر میزان‌نهفتگی[1] برخوردار باشد.

انواع روش‌های نقش‌آب‌زنی نرم‌افزار بر اساس آن که روال استخراج برای تشخیص نقش‌آب نیازمند اجرای نرم‌افزار باشد یا خیر به‌ترتیب به دو دسته پویا و ایستا تقسیم می‌شوند. روش‌های نقش‌آب‌زنی ایستا نقش‌آب را در متن برنامه اجرایی تعبیه می‌کنند. در مقابل روش‌های نقش‌آب‌زنی پویا نقش‌آب را در حالت اجرایی برنامه به‌ازای یک رشته ورودی مشخص تعبیه می‌نمایند. روش‌های ایستا به دلیل دارا بودن نرخ داده بالا و قابلیت محافظت از اجزاء نرم‌افزار نسبت به روش‌های پویا دارای برتری هستند.

این مقاله به ارائه روشی در نقش‌آب‌زنی ایستا نرم‌افزار می‌پردازد. ایده اصلی این روش استفاده از بلاک‌های ساختگی، تعبیه نقش‌آب در آن‌ها و درج بلاک‌های مذکور در کد برنامه است. بر طبق نتایج ارزیابی‌های انجام شده این روش از حیث نرخ داده و میزان‌نهفتگی نسبت به سایر روش‌های نقش‌آب‌زنی ایستا دارای برتری‌هایی است. به علاوه امکان ذخیره اندیس نسبی نقش‌آب را بدون کاهش میزان نرخ داده فراهم می‌آورد. این اندیس‌های نسبی در جدولی خارج از نرم‌افزار نگهداری می‌شود. این در حالی است که اولا تخصیص مکانی برای درج اندیس نقش‌آب در سایر روش‌های نقش‌آب‌زنی اعم از ایستا و پویا پیش بینی نشده‌است و ثانیا یکی از دلایل آسیب‌پذیری روش‌های نقش‌آب‌زنی در برابر حملات تخریب نقش‌آب، ناشی از عدم نگهداری اندیس نقش‌آب است.

در مجموع ارزیابی روش پیشنهادی و مقایسه آن با سایر روش‌های نقش‌آب‌زنی نشان می‌دهد که این روش در کاربردهایی که نیازمند استفاده از یک روش نقش‌آب‌زنی با قابلیت بازیابی محلی نقش‌آب حتی پس از اعمال حمله تخریب نقش‌آب، امکان محافظت از اجزای نرم‌افزار و در عین حال با حداکثر نرخ‌داده و میزان‌نهفتگی هستند، بی‌رقیب است. محافظت از هر یک از کلاس‌های نرم‌افزار از طریق تعبیه مشخصه‌ای از تولید کننده نرم‌افزار در هر یک از کلاس‌های آن می‌تواند نمونه‌ای از این کاربردها باشد.

ساختار مقاله به‌این شرح است: بخش۲ به اجمال مروری بر روش‌های موجود در نقش‌آب‌زنی ایستا دارد. بخش۳ به بیان روش بلاک ساختگی می‌پردازد. بخش٤ به توصیف چارچوب ارزیابی و ویژگی‌های مورد ارزیابی تخصیص یافته است. در بخش۵ نتایج حاصل از ارزیابی و مقایسه روش‌های ایستا و روش بلاک‌ساختگی ارائه شده است. در انتها نیز بخش٦ به جمع‌بندی و نتیجه‌گیری اختصاص یافته‌است.

۲- پژوهش‌های پیشین

از آنجایی که روش پیشنهادی از نوع کدنقش‌آب‌زنی‌ایستا است؛ این مقاله در ارزیابی روش پیشنهادی آن را با سایر روش‌های کد نقش‌آب‌زنی ایستا مقایسه می‌نماید. این بخش به معرفی اجمالی هر یک از روش‌های کد نقش‌آب‌زنی ایستا می‌پردازد.

Davidson [۵] اولین روش کد نقش‌آب‌زنی ایستا را ابداع کرد. در این روش نقش‌آب درترتیب قرارگیری بلاک‌های ساده یک برنامه درج می‌شود. برای استخراج نقش‌آب‌ترتیب قرارگیری در برنامه اصلی باترتیب قرارگیری در برنامه نقش‌آب‌زنی شده مقایسه می‌شود.

Qu [۱۱] روشی بر مبنای نحوه تخصیص ثبات‌ها ارائه کرد که برای تعبیه نقش‌آب از دو ابزار گراف‌تداخل و مساله رنگ‌آمیزی گراف بهره می‌جوید. در مجموع این روش قابلیت تعبیه نقش‌آب با نرخ‌داده بسیار محدودی را دارا می‌باشد ولی در مقابل از آنجایی که تنها نحوه تخصیص ثبات‌ها را دستکاری می‌کند از میزان‌نهفتگی بسیار خوبی برخوردار است.

[1] stealthy

روش دیگری که مانند بسیاری از الگوریتم‌های نقش‌آب‌زنی رسانه ای بر اساس روش طیف گسترده[۲] توسط *Stern* [۸] ارائه شده و برداری بر اساس فراوانی عملگرها در کد برنامه تعریف می‌کند و از طریق دستکاری فراوانی عملگرها این بردار را به برداری که بر اساس مقدار نقش‌آب محاسبه می‌شود، نزدیک می‌نماید. روش *Stern* برای استخراج نقش‌آب علاوه بر کد نقش‌آب‌زنی شده به کد نرم‌افزار پیش از نقش‌آب‌زنی و همچنین مقدار نقش‌آب نیاز دارد. عمده تفاوت این روش با سایر روش‌های نقش‌آب‌زنی آن است که خروجی روال استخراج به جای مقدار نقش‌آب احتمال وجود نقش‌آب را تخمین می‌زند.

Monden [۹] روشی مختص نقش‌آب‌زنی کلاس‌های جاوا ارائه می‌کند. این روش برای تعبیه نقش‌آب یک متد ساختگی در کلاس درج می‌کند و از طریق جایگزینی عملگرها و همچنین جایگزینی عملوندهای صریح با مقادیرجدید، نقش‌آب‌را در متدساختگی درج می‌نماید.

Venkatesan [۱۲] نقش‌آب را به صورت یک عدد صحیح در نظر می‌گیرد و یک گراف شمارش پذیر متناظر با آن تولید می‌کند. رئوس این گراف را یک سری بلاک ساده ساختگی تشکیل می‌دهد. سپس این گراف را به نحوی با گراف کنترل جریان برنامه گره می‌زند.

Arboit[۱۰] بر روی نقش‌آب‌زنی برنامه‌های جاوا با استفاده از عبارات همیشه صحیح/غلط متمرکزشده‌است. بیت‌های نقش‌آب دراطلاعات موجود درعبارت درج می‌شود. علاوه بر آن هر عبارت شامل اندیس بیت‌هایی که نگهداری می‌کند نیز است که این مساله از نرخ داده آن می‌کاهد.

۳- روش پیشنهادی (بلاک‌ساختگی)

بلاک‌ساختگی را بلاک ساده‌ای تعریف می‌کنیم که یک دستور پرش شرطی به ابتدای آن اضافه شده باشد به طوری که اولا عبارت شرط آن را یک عبارت همیشه درست تشکیل دهد و ثانیا مقصد پرش آن اولین دستور بعد از بلاک مورد نظر باشد.

روش بلاک‌ساختگی از ایده تولید و درج بلاک‌های ساختگی به عنوان مکانی برای تعبیه نقش‌آب، بهره می‌برد. برای تولید هر بلاک‌ساختگی به یک شرط همیشه‌درست و یک بدنه نیازمندیم. روش بلاک‌ساختگی شرط همیشه‌درست را با استفاده از اطلاعات جدول عبارات همیشه صحیح/غلط تولید می‌کند و برای تولید بدنه بلاک، از الگوهای موجود در سایر بلاک‌های ساده کد نرم‌افزار استفاده می‌کند؛ و از این طریق تامین حداکثر شباهت میان بلاک‌ساختگی و سایر بلاک‌های متدهای کلاس را تضمین می‌نماید. برای تعبیه نقش‌آب در بلاک‌ساختگی نیز از جایگذاری عملگرهایی با نرخ فراوانی مشابه، ساختار نحوی یکسان و دارای سازگاری نوع استفاده می‌شود.

این بخش در ادامه به شرح جزئیات اجزای‌روش‌پیشنهادی شامل چگونگی تشکیل جدول عبارات‌همیشه‌صحیح/غلط، دسته‌بندی عملگرهای مشابه، فرآیند تعبیه و استخراج نقش‌آب می‌پردازد.

۳-۱- تشکیل جدول عبارات همیشه صحیح/غلط

عبارت همیشه صحیح/غلط به عبارتی گفته می‌شود که مستقل از مقدار متغییرهای به کار رفته در آن همواره ارزش یکسانی داشته باشد. انواع عبارات همیشه صحیح/غلط که به صورت خودکار قابل تولید و تعبیه در نرم‌افزار هستند شامل سه گروه (۱)مبتنی بر تئوری اعداد، (۲)مبتنی بر نام‌های مستعار اشیاء و (۳)مبتنی بر کدهای همروند هستند. عبارات همیشه صحیح/غلط تولید شده بر اساس تئوری اعداد، با استفاده ازترکیب حملات تحلیل ایستا و پویا قابل شناسایی است[۱۳]. اما شناسایی عبارات مبتنی بر نام‌های مستعار و کدهای همروند با صرف هزینه‌های نمایی[۳] همراه‌است[۱۴].

روش بلاک‌ساختگی برای تشکیل جدول عبارات همیشه صحیح/غلط، علاوه بر استفاده از عبارات مبتنی بر نام‌های مستعار و کدهای همروند، از خصیصه‌های نامتغییر کلاس[۴] نیز استفاده می‌کند. استفاده از خصیصه‌های نامتغییر کلاس دارای دو مزیت است که باعث می‌شود شناسایی آنها با صرف هزینه‌های مضاعف همراه باشد. این مزایا عبارتند از: (۱)استفاده از خصیصه‌های نامتغییر کلاس امکان به کارگیری خصیصه‌های نمونه‌ها[۵] را، علاوه بر به کارگیری خصیصه‌های کلاس[۶] و

[2] spread spectrum

[3] به‌ترتیب از مرتبه 2^n و $n!$

[4] class invariants

[5] instance variable

متغیرهای محلی متدها، در ساختار عبارت همیشه صحیح/غلط،به وجود می‌آورد. تحلیل مقدار خصیصه‌های نمونه تنها از طریق تحلیل معنایی صورت پذیر است. (۲)عبارتی مشابه با ساختار و ویژگی‌های سایر عبارت‌های رایج در کد برنامه، مانند کنترل‌های مربوط به دامنه خصیصه‌های کلاس، تولید می‌کند که تمایز عبارت همیشه صحیح/غلط را از سایر عبارات مشکل می‌سازد.

استفاده از خصیصه‌های نامتغییر علاوه بر افزایش هزینه حمله شناسایی‌عبارات‌همیشه‌صحیح/غلط‌دارای‌مزیت‌سادگی بکارگیری نیز است. در استفاده از عبارات مبتنی بر نام‌های مستعار و کدهای همروند لازم است پیش از مکان کنترل شرط، تکه‌کدی به کدمتد موردنظر اضافه‌شود که مسئولیت ساخت نمونه، اشاره‌گر و یا مقداردهی متغیر را برعهده دارد. اما لازم نیست این کار در استفاده از خصیصه‌های نامتغییر کلاس انجام شود.

در مقابل مزایای یاد شده استفاده از خصیصه‌های نامتغییر کلاس با دو مساله مواجه است: (۱)تولید عبارت همیشه صحیح/غلط به صورت کاملا خودکار ممکن نیست و لازم است از دانش طراح و برنامه نویس استفاده شود و (۲)ممکن است خصیصه‌های نامتغییر کلاس در طول عمر نرم‌افزار ثابت نباشد و با تغییر ویژگی‌های طراحی و یا برنامه نویسی نرم‌افزار، تغییر کند. البته هر دو این مشکلات از طریق به روز نگاه داشتن جدول عبارات همیشه صحیح/غلط قابل مقابله‌است.

۳-۲- دسته‌بندی عملگرهای مشابه

در روش پیشنهادی به منظور تامین حداکثر میزان‌نهفتگی تنها عملگرهایی با یکدیگر جایگزین می‌شوند که دارای نرخ وقوع مشابه هستند و برای گذر از راستی آزما بایت‌کد[۷] بدون بروز خطا در زمان بارگذاری کلاس، تنها عملگرهایی با یکدیگر جایگزین می‌شوند که دارای نحویکسان و سازگاری نوع هستند. برای این منظور در گام اول کلیه عملگرها براساس نحو یکسان و سازگاری نوع، دسته‌بندی می‌شوند و سپس در دومین گام هر دسته عملگر به‌گونه‌ای به دسته‌های کوچکتر تقسیم می‌شوند که اعضای هر دسته دارای نرخ وقوع مشابه باشند. نرخ وقوع مشابه به‌این معناست که در هر دسته بیشترین نرخ وقوع حداکثر α برابر کمترین نرخ وقوع باشد. شکل(۱) نحوه دسته‌بندی عملگرهای حسابی و صحیح *Java Bytecode* را پس از انجام گام اول و دوم دسته‌بندی، به‌ازای $\alpha=5$ نشان می‌دهد.

opcode	usage rate	assigned bit
imul	0.10%	00
idiv	0.05%	01
irem	0.02%	10
ior	0.06%	11

opcode	usage rate	assigned bit
iadd	0.64%	00
isub	0.35%	01
iand	0.21%	1

شکل ۱: نمونه ای از دسته‌بندی دستورات

هرچه α کوچک‌تر باشد نرخ وقوع اعضای هر دسته به یکدیگر نزدیک‌تر شده و در نتیجه میزان‌نهفتگی بیشتر می‌شود ولی در مقابل، کوچک بودن α باعث می‌شود تعداد دسته‌ها بیشتر و تعداد اعضای هر دسته کمتر شود و در نتیجه نرخ‌داده کاهش یابد. به‌طورکلی در دامنه کدینگ اطلاعات میانگین تعداد بیت قابل درج با هر انتخاب از میان n گزینه از رابطه(۱) محاسبه می‌شود[۱۵].

$$b(n)=\lceil \log_2(n)-1 \rceil + \frac{n-2^{\lceil \log_2(n)-1 \rceil}}{2^{\lceil \log_2(n)-1 \rceil}} \qquad \text{رابطه(۱)}$$

بر اساس این رابطه $b(n+m)>b(n)+b(m)$ است. به عنوان مثال برای حالتی که $n=m$ است به سادگی می‌توان نشان داد $b(2n)>2b(n)$ است. بنابراین تقسیم هر دسته به دسته‌های کوچک‌تر باعث کاهش تعداد بیت قابل درج و در نتیجه کاهش نرخ داده می‌شود.

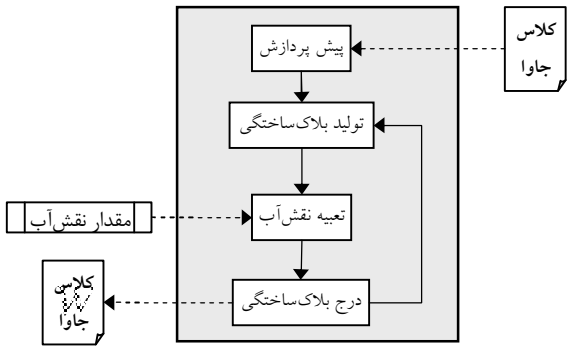
با استدلالی مشابه هر چه α بزرگ‌تر انتخاب شود میزان‌نهفتگی کاهش می‌یابد و در مقابل نرخ‌داده افزایش می‌یابد. پس از انجام دسته‌بندی، به هریک از اعضای هر دسته یک رشته دودویی منحصر به فرد تخصیص داده می‌شود.

۳-۳- فرآیند تعبیه نقش‌آب در کلاس

فرآیند تعبیه با یک سری عملیات به عنوان پیش پردازش آغاز می‌شود. پس از انجام پیش پردازش، روال تعبیه شامل سه مرحله است که تا زمانی که کل نقش‌آب در کلاس تعبیه شود یا اندازه هر یک از متدهای کلاس از β برابر اندازه اولیه بیشتر شود، در یک حلقه تکرار می‌شوند. واضح است که اندازه ضریب β با نرخ داده قابل درج در کلاس نسبت مستقیم و با میزان‌نهفتگی نسبت عکس دارد. شکل(۲) مراحل تعبیه نقش‌آب

[6] class(static) variable
[7] bytecode verifier

را نشان می‌دهد. در این بخش هر یک از این مراحل به‌اختصار توضیح داده می‌شود.



شکل ۲: فرایند تعبیه نقش‌آب

پیش پردازش

ابتدا کلیه بلاک‌های‌ساده موجود در متدهای کلاس استخراج شده و در یک لیست به نام b قرار داده می‌شوند. سپس لیست موردنظر بر اساس تعدادبیت قابل‌درج در هر بلاک به‌ترتیب نزولی مرتب می‌شوند. در گام پایانی متدهای موجود در کلاس در یک لیست قرار داده می‌شوند. این لیست را m می‌نامیم. سپس حداکثر تعداد بایت قابل درج در هر متد بر اساس اندازه فعلی متد و ضریب β محاسبه می‌شود و لیست m بر اساس حداکثر اندازه مجاز به‌ترتیب نزولی مرتب می‌شود.

تولید بلاک‌ساختگی

برای تولید یک بلاک‌ساختگی به یک بدنه و یک شرط همیشه صحیح نیاز است. برای تولید بدنه یکی از بلاک‌های ساده موجود در لیست b انتخاب و مورد استفاده قرار می‌گیرد. این انتخاب بر اساس اولویت دهی به بلاک‌هایی که تا کنون برای تولید بلاک‌ساختگی مورد استفاده قرار نگرفته اند و با توجه به تعداد بیت تعبیه نشده نقش‌آب صورت می‌گیرد. بلاک انتخاب شده نباید دارای ظرفیت تعبیه تعداد بیت بیشتری نسبت به تعداد بیت باقی مانده نقش‌آب باشد. برای کاهش ظرفیت بلاک می‌توان تعدادی از عملگرهای آن را حذف کرد.

به منظور ساخت یک شرط همیشه درست از جدول عبارات همیشه صحیح/غلط یک رکورد که قبلا استفاده نشده است انتخاب می‌شود و یک سری دستور متناظر با آن به گونه‌ای تولید می‌شوند که برقراری شرط مربوطه را بررسی کنند. عبارت در صورتی که از نوع همیشه غلط باشد با یک دستور پرش شرطی از نوع «اگر نه» همراه می‌شود و اگر از نوع همیشه درست باشد با یک دستور پرش شرطی از نوع «اگر» همراه می‌شود. آدرس مقصد دستور پرش شرطی به آدرس اولین دستور بعد از بلاک‌ساختگی تنظیم می‌شود.

تعبیه نقش‌آب

برای تعبیه نقش‌آب، بایت‌کدهای موجود در بدنه بلاک‌ساختگی به‌ترتیب مورد بررسی قرار می‌گیرند. به‌ازای هر عملگر p در صورتی که تعداد هم گروه‌های p بیش از صفر باشد، بیت‌های تعبیه نشده نقش‌آب با الگوی بیتی تخصیص داده شده به p و هم گروه‌های آن مقایسه می‌شود و با هر کدام که هم‌خوانی داشت؛ با p جایگزین می‌گردد. یادآوری می‌شود از آنجایی که هم گروه‌های p دارای ساختار نحوی مشابه و سازگاری نوع هستند جایگزینی p با هر یک از هم گروه‌هایش هیچ‌گونه دستکاری در عملوندها و دستورات قبل‌و بعد را باعث نمی‌شود. پس‌از تعبیه نقش‌آب «اندیس اولین بیت تعبیه شده» در بلاک‌ساختگی در جدول‌عبارات‌همیشه‌صحیح/غلط مقابل عبارت مورد استفاده در ساخت بلاک‌ساختگی مورد نظر ثبت می‌شود.

درج بلاک‌ساختگی

پس از تولید بلاک‌ساختگی d و تعبیه نقش‌آب در آن نوبت به انتخاب یک متد برای درج بلاک‌ساختگی d در آن می‌رسد. برای این منظور بزرگ‌ترین متدی که درج d در آن باعث تخطی از حداکثر اندازه مجاز آن نشود انتخاب می‌شود.

پس از انتخاب متد، یکی از رئوس گراف کنترل جریان متد به نام v که در مسیر اصلی روند اجرایی متد قرار دارد و درون هیچ حلقه‌ای نمی‌باشد، انتخاب می‌شود. بلاک‌ساختگی بلافاصله پس از آخرین دستور v درج می‌شود. سپس آدرس مقصد دستور پرش شرطی ابتدای d به آدرس اولین دستور بعد از بلاک v تنظیم می‌شود. سایر آدرس‌های دستورات پرش متد نیز درصورت نیاز دوباره تنظیم می‌شوند.

۳-۴- روال استخراج

به منظور استخراج نقش‌آب تعبیه شده در یک کلاس، ابتدا جدول عبارات همیشه صحیح/غلط بر اساس «اندیس اولین بیت

تعبیه شده» به‌ترتیب صعودی مرتب می‌شوند. سپس از ابتدای جدول به ازای هر رکورد یک بلاک‌ساختگی در کلاس مورد نظر جستجو می‌شود. این جستجو بر اساس الگوی شرط ابتدای بلاک انجام می‌پذیرد. در صورت موفقیت آمیز بودن جستجو، نقش‌آب بر اساس جدول دسته‌بندی عملگرها از بلاک‌ساختگی شناسایی شده استخراج شده و در اندیس مربوطه درج می‌شود. در صورت موفقیت آمیز نبودن جستجو، مکان قسمتی از نقش‌آب که انتظار می‌رفت در بلاک پیدا نشده تعبیه شده باشد به صورت *random* با صفر و یک پر می‌شود. این مکان بر اساس اندیس‌های نگهداری شده در جدول شناسایی می‌شود.

به‌این‌ترتیب اولا روش پیشنهادی مستقل ازترتیب قرار گیری بلاک‌ها است و ثانیا حتی پس از اعمال حمله تخریب نقش‌آب از نوع تغییر دهنده‌ترتیب قرار گیری بلاک‌ها و یا دستکاری بخشی از کد کلاس مربوطه، باز هم نقش‌آب به صورت محلی قابل استخراج است.

۴- توصیف روش ارزیابی

از آنجایی که رویکرد نقش‌آب‌زنی نرم‌افزار تنها یک دهه قدمت دارد؛ تاکنون چارچوبی استاندارد و جامع برای ارزیابی روش‌های آن معرفی نشده است. مقالات معدودی به‌ارزیابی روش‌های نقش‌آب‌زنی پرداخته‌اند[۱۶،۱۷،۱۸،۱۹،۲۰،۲۱] که به دلیل یکسان نبودن مجموعه نرم‌افزار مورد ارزیابی و همچنین روش ارزیابی، امکان مقایسه مستقیم نتایج آنها وجود ندارد. مساله دیگر در ارزیابی روش‌های نقش‌آب‌زنی، کیفی بودن معیارهای آن است. تنها معیاری که تاکنون برای ارزیابی کمی‌آن اتفاق نظر حاصل شده‌است ارزیابی نرخ‌داده می‌باشد.

این مقاله برای مقایسه روش پیشنهادی بلاک‌ساختگی با سایر روش‌های نقش‌آب‌زنی ایستا چارچوبی برای ارزیابی ارائه کرده و کلیه روش‌های ایستا به علاوه روش بلاک‌ساختگی را در این چارچوب واحد مورد ارزیابی قرار می‌دهد. در این بخش هر یک از معیارهای ارزیابی روش‌های نقش‌آب‌زنی معرفی شده و روش پیشنهادی این مقاله برای ارزیابی هر یک معرفی می‌شود.

نرخ داده: نرخ داده معیاری برای توصیف کمی‌اطلاعات قابل درج در یک نرم‌افزار است[۶]. ازآنجایی‌که روش‌های مورد ارزیابی ایستا بوده و همگی نقش‌آب را در کد نرم‌افزار درج می‌نمایند؛ برای ارزیابی نرخ‌داده، میانگین بیشینه تعداد بیت قابل درج در هر ۱۰۰۰ بایت‌کد جاوا اندازه‌گیری می‌شود. برای محاسبه این میانگین از بیشینه تعداد بیت قابل درج در کلاس‌های مختلف استفاده می‌شود.

میزان‌نهفتگی: میزان‌نهفتگی معیاری برای توصیف میزان نامحسوس بودن نقش‌آب برای مهاجم است[۶]. نامحسوس بودن نقش‌آب معادل عدم توانایی مهاجم در به‌دست آوردن برآوردی برای مکان نقش‌آب است. مکان نقش‌آب ارتباط نظیر به نظیر با روش و الگوی درج نقش‌آب دارد. برای رسیدن به تخمینی از مکان نقش‌آب از دو روش تحلیل ایستا و پویای کل و یا قسمتی از نرم‌افزار، می‌توان استفاده کرد.

از آنجایی که روش‌های تحلیل پویا با صرف هزینه‌های بالایی برای مهاجم همراه هستند و همچنین روش‌های نقش‌آب‌زنی ایستا اغلب تاثیر ناچیزی بر روی ویژگی‌های پویای نرم‌افزار دارند، این مقاله از اطلاعاتی که تحلیل پویا در اختیار مهاجم قرار می‌دهد صرف نظر می‌کند.

به منظور دستیابی به معیاری برای مقایسه میزان‌نهفتگی روش‌ها، ابتدا یک سری خصیصه را که اعمال حداقل یکی از روش‌های نقش‌آب‌زنی ایستا باعث تغییر محتوای آن می‌شود؛ استخراج می‌گردد. سپس هر یک از این خصیصه‌ها در مجموعه‌ای از کلاس‌های نقش‌آب‌زنی شده با استفاده از هر یک از روش‌های ایستا، به علاوه مجموعه‌ای از کلاس‌های بدون نقش‌آب که آنها را کلاس عادی می‌نامیم، مورد اندازه‌گیری قرار می‌گیرد. پس از آن با استفاده از یک ابزار یادگیری قوانینی برای شناسایی هر روش نقش‌آب‌زنی استخراج می‌شود. در آخر برای رسیدن به برآوردی از دقت این قوانین در شناسایی نوع روش نقش‌آب‌زنی مورد استفاده، درصدی از کلاس‌های نقش‌آب‌زنی شده که در مجموعه یادگیری حضور نداشته اند مورد آزمون قرار می‌گیرد.

هر چه قوانین یادگیری شده برای شناسایی روش‌نقش‌آب‌زنی x بهتر بتوانند کلاس‌های‌نقش‌آب‌زنی‌شده با این روش را شناسایی کنند، می‌توان این‌گونه نتیجه‌گرفت‌که روشx دارای ویژگی‌هایی است که با احتمال موفقیت بالاتری مهاجم را از طریق تحلیل ایستا و وارسی این ویژگی‌ها به مکان نقش‌آب و روش مورد

استفاده برای نقش‌آب‌زنی هدایت می‌کند. نسبت کلاس‌هایی که به درستی دسته‌بندی نشده‌اند به کل کلاس‌ها به عنوان معیاری برای اندازه‌گیری میزان‌نهفتگی مورد استفاده قرار می‌گیرد.

نوع استخراج: در نقش‌آب‌زنی نرم‌افزار بسته به اینکه روال استخراج نقش‌آب نیازبه دسترسی به کد برنامه قبل از تعبیه نقش‌آب داشته باشد و یا بدون آن عمل استخراج را انجام دهد، روش *informed* و یا *blind* نامیده می‌شود. در بسیاری از کاربری‌ها امکان دسترسی به کد پیش از نقش‌آب‌زنی وجود ندارد. بنابراین روش‌های *informed* دارای کاربرد محدودتری نسبت به روش‌های *blind* هستند.

۵- نتایج حاصل از ارزیابی و مقایسه روش‌ها

برای ارزیابی روش‌های نقش‌آب‌زنی ایستا از پیاده‌سازی که ابزار *Sandmark* ارائه می‌دهد، استفاده شده‌است. *Sandmark* ابزاری برای انجام تحقیقات در حوزه روش‌های محافظت از نرم‌افزار است[۲۲] و در پیاده‌سازی روش پیشنهادی $\alpha=5$ و $\beta=2$ درنظر گرفته‌شده است. کلیه ارزیابی‌ها با درج شش نقش‌آب مختلف با اندازه‌های متفاوت در ۲۱ کلاس جاوا انجام شده است. این در حالی است که جامع‌ترین ارزیابی که تاکنون صورت گرفته در دامنه نقش‌آب‌زنی نرم‌افزار[۲۳] تنها از نقش‌آب‌زنی ۱۰ کلاس جاوا بهره جسته است. این ۲۱ کلاس از میان ده هزار کلاس جاوا به صورت تصادفی انتخاب شده‌اند. گردآوری بانکی از ده هزار کلاس از طریق استخراج کلاس‌های موجود در ده برنامه کاربردی واقعی با ویژگی‌های مختلف صورت پذیرفته است.

به منظور ارزیابی میزان‌نهفتگی از ابزار *C4.5*[۲۴] برای دسته‌بندی روش‌های نقش‌آب‌زنی استفاده شده است. ۷۰ درصد کلاس‌های نقش‌آب‌زنی شده با هر یک از روش‌های ایستا به علاوه کلاس‌های عادی، به عنوان داده‌های آموزشی و ۳۰ درصد باقی مانده به عنوان داده‌های آزمون مورد استفاده قرار گرفته است. جدول(۲) اطلاعات مربوط به تعداد قوانین، دقت دسته‌بندی داده‌های آموزشی و دقت دسته‌بندی داده‌های آزمون را به تفکیک روش نقش‌آب‌زنی نشان می‌دهد. جدول(۳) نتایج حاصل از ارزیابی هر یک از روش‌های نقش‌آب‌زنی ایستا و روش پیشنهادی بلاک‌ساختگی را نشان می‌دهد.

جدول۲: تعداد قوانین و دقت دسته‌بندی

نام روش نقش‌آب‌زنی	تعداد قوانین	دقت دسته‌بندی‌داده‌های آزمون
بدون نقش‌آب	۲	٪ ۱۰۰
Arboit [۱۰]	۱	٪ ۱۰۰
Davidson [۵]	۳	٪ ۵۰/۰
Monden [۹]	۱	٪ ۱۰۰
Qu [۱۱]	۱	٪ ۰/۰
Stern [۸]	۳	٪ ۲۷/۸
Venkatesan [۱۲]	۱	٪ ۱۰۰
بلاک‌ساختگی	۵	٪ ۱۴/۲

سه روش *Venkatesan*، *Stern* و *Monden* دارای نرخ داده‌ای بیشتر از روش پیشنهادی بلاک‌ساختگی هستند. اما دو روش *Venkatesan* و *Monden* دارای میزان‌نهفتگی بسیار پایینی هستند و روش *Stern* به دلیل *informed* بودن آن دارای کاربری محدودی است. به علاوه روش *Stern* قابلیت بازیابی مقدار نقش‌آب را ندارد و تنها از طریق مقایسه کد قبل و بعد از نقش‌آب‌زنی می‌تواند وجود و یا عدم وجود نقش‌آب را تایید و یا رد کند. تنها روشی که از نظر میزان‌نهفتگی نسبت به روش پیشنهادی بلاک‌ساختگی دارای مزیت است روش *Qu* است که دارای نرخ‌داده بسیار کمی‌است که در بسیاری از کاربردها قابل قبول نمی‌باشد. ازمیان ۲۱ کلاس مورد آزمون، ۱۷ کلاس در استفاده از روش *Qu* دارای نرخ داده صفر بودند.

به علاوه‌از آنجایی که روش بلاک‌ساختگی تنها روشی است که اندیس قطعات تعبیه شده نقش‌آب را نیز ذخیره می‌نماید حتی پس از اعمال حمله تخریب نقش‌آب به قسمتی از کد، امکان بازیابی نقش‌آب تعبیه شده در سایر بخش‌ها وجود دارد.

۶- نتیجه گیری

این مقاله به ارائه روشی نو در نقش‌آب‌زنی کلاس‌های جاوا پرداخته است. ایده اصلی این روش درج بلاک‌های ساختگی در متدهای کلاس به عنوان فضایی برای تعبیه نقش‌آب است. استفاده از بلاک‌های ساختگی نه‌تنها نرخ داده مناسبی را برای این روش به ارمغان آورده است بلکه امکان ذخیره اندیس قطعات نقش‌آب را فراهم می‌آورد. به علاوه نحوه درج نقش‌آب در بلاک‌های ساختگی به‌گونه‌ای تنظیم شده است که حداقل تاثیر را در ویژگی‌های آماری کلاس مربوطه داشته باشد.

جدول۳: نتایج حاصل از ارزیابی

	بلاک‌ساختگی	[۱۲] Venkatesan	[۸] Stern	[۱۱] Qu	[۹] Monden	[۵] Davidson	[۱۰] Arboit
نرخ داده	۷/۷	۸/۲	بیشتر از ۱۰	۰/۱	بیشتر از ۱۰	۶/۷	۴/۳
میزان‌نهفتگی	٪ ۸۶	٪ ۰	٪ ۷۲	٪ ۱۰۰	٪ ۰	٪ ۵۰	٪ ۰
نوع استخراج	*blind*	*blind*	*informed*	*blind*	*blind*	*informed*	*blind*

نتایج حاصل از ارزیابی میزان‌نهفتگی موید این ویژگی است. در مجموع ارزیابی روش پیشنهادی و مقایسه آن با سایر روش‌های نقش‌آب‌زنی نشان می‌دهد که این روش در کاربردهایی که نیازمند استفاده از یک روش نقش‌آب‌زنی با قابلیت بازیابی محلی نقش‌آب حتی پس از اعمال حمله تخریب نقش‌آب، امکان محافظت از اجزای نرم‌افزار و در عین حال با حداکثر نرخ‌داده و میزان‌نهفتگی هستند، بی‌رقیب است.

۷- سپاسگزاری

این مقاله با حمایت مالی مرکزتحقیقات‌مخابرات‌ایران انجام‌شده است.

۸- مراجع

[1] W. Zhul, C. Thomborson and F. Wang, “A Survey of Software Watermarking”, *IEEE international conference on intelligence and security informatics*, vol. 34, pp. 454-458, 2005.

[۲] س.جلیلی و س.حسینی مقدم، "نهان‌نگاری و نقش‌آب‌زنی برنامه‌های جاوا با رویکرد متدهای‌ساختگی"، مجموعه مقالات دوازدهمین کنفرانس بین‌المللی انجمن‌کامپیوتر ایران، دانشگاه شهید بهشتی، ۱۳۸۵.

[3] O. Esparza, M. Fernandez and M. Soriano, “Protecting mobile agents by using traceability techniques”, *In International Conference on Information Technology*, pp. 264- 268, 2003.

[4] J. Nagra, C. Thomborson and C. Collberg, “A functional taxonomy for software watermarking”, *In Proc. The 25th Australasian Computer Science Conference*, pp. 177-186, 2002.

[5] R. Davidson and N. Myhrvold. “*Method and system for generating and auditing a signature for a computer program*”, US Patent 5, pp. 559-884, 1996.

[6] C. Collberg and C. Thomborson, “Software watermarking: Models and dynamic embedding”, *In Symposium on Principles of Programming Languages*, pp. 311–324, 1999.

[7] C. Collberg, E. Carter, S. Debray, A. Huntwork, J. Kececioglu, C. Linn and M. Stepp, “Dynamic path-based software watermarking”, *ACM SIGPLAN Notices, Proceedings of ACM conference on Programming language design and implementation (SIGPLAN)*, vol. 39, No. 6, 2004.

[8] J. Stern, G. Hachez, F. Koeune, and J. Quisquater, “Robust Object Watermarking: Application to Code”, *Information Hiding Workshop*, pp. 368-378, 1999.

[9] A. Monden, H. Iida, K. Matsumoto, K. Inoue and K. Torii, “A Practical Method for Watermarking Java Programs”, Proc. *The 24th Computer Software and Applications Conference* , pp. 191-197, 2000.

[10] G. Arboit, “A Method for Watermarking Java Programs via Opaque Predicates”, *Proc. The Fifth International Conference on Electronic Commerce Research*, 2002.

[11] G. Qu and M. Potkonjak, “Analysis of Watermarking Techniques for Graph Coloring Problem”, *Proc. IEEE/ACM International Conference on Computer Aided Design, ACM Press*, pp. 190-93,1998.

[12] R. Venkatesan, V. Vazirani, and S. Sinha, “A Graph Theoretic Approach to Software Watermarking”, *Proc. The 4th International Information Hiding Workshop*, pp. 51-65, 2001.

[13] M. D. Preda, M. Madou, K. D. Bosschere, R. Giacobazzi, “Opaque Predicates Detection by Abstract Interpretation”, *Algebraic Methodology and Software Technology, 11th International Conference*. Springer-Verlag. Lecture Notes in Computer Science. Vol. 4019. pp. 81-95, 2006.

[14] C. Collberg, C. Thomborson, and D. Low, “Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs”, *ACM Symposium on the Principles of Programming Languages (POPL)*, pp. 184-196, 1998.

[15] B. Anckaert, “Hiding Information in Programs”, *Fifth FTW PhD Symposium*, 2004.

[16] K. Hattanda and S. Ichikawa, “*The Evaluation of Davidson’s digital signature scheme*”, IEICE Trans, Fundamentals, E87-A, No.1, pp. 224-225, 2004.

[17] G. Myles, C. Collberg, Z. Heidepriem and A. Navabi, “*The Evaluation of Two Software Watermarking Algorithms*”, Software: Practice and Experience, vol. 35, No.10, pp. 923-938, 2005.

[18] G. Myles and C. Colberg, “Software watermarking through register allocation: Implementation, analysis and attacks”, *International Conference onSecurity and Cryptology*, 2003.

[19] G. Myles and C. Colberg, “Software watermarking via opaque predicates: Implementation, analysis and attacks”, *7th International Conference on Electronic Commerce Reasearch*, 2004.

[20] T. Sahoo and C. Colberg, “Software watermarking in the frequency domain”, Technical Report, Department of Computer Science, University of Arizona, 2004.

[21] C. Colberg, A. Huntwork, E. Carter and G. Townsend, “Graph theoretic software watermarks: Implementation, analysis and attacks”, *6th International Information Hiding Workshop*, 2004.

[22] C. Collberg, G. Myles and A .Huntwork, “Sandmark: A Tool for Software Protection Research”, *IEEE Security and Privacy*, Vol. 1, No. 04, pp. 40-49, 2003.

[23] G. Myles, “Software Theft Detection Through Program Identification”, PHD thesis, University of Arizona, 2006.

[24] Quinlan J.R. C4.5 Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, CA, 1993.