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

روشی برای کاهش آسیب پذیری نرم افزار در مقابل تکثیر غیر مجاز با روش نقش‌آب‌زنی گراف پویا

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

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

چکیده: علی رقم وجود روش‌های متنوع برای محافظت از نرم‌افزار در برابر سرقت و استفاده غیرمجاز، همچنان مقابله با سوء استفاده از محصولات نرم‌افزاری یکی از بزرگ‌ترین نگرانی‌های تولیدکنندگان نرم‌افزار است. دلیل اصلی این مساله ضعف‌ها و آسیب پذیری‌های شناخته شده در این روش‌ها است. این مقاله روشی برای وارسی صحت نقش‌آب بدون استخراج آن را ارائه می نماید و از آن برای کاهش آسیب پذیری‌های روش توکن نرم‌افزاری که یکی از پراستفاده‌ترین روش‌های محافظت از نرم‌افزار است، استفاده می‌کند. در روش پیشنهادی از رویکرد نقش‌آب‌زنی گراف پویا برای تعبیه و وارسی توکن مورد نظر استفاده شده است و صحت محیط اجرایی نرم‌افزار به صورت توزیع شده در نقاط مختلف نرم‌افزار در حال اجرا، وارسی می‌شود. روش پیشنهادی علاوه بر ارتقاء سطح حفاظتی به کلاس‌های تشکیل دهنده نرم‌افزار، امکان محافظت در برابر تکثیر غیرمجاز، استفاده مجدد غیرمجاز و ارائه یک گونه آزمایشی از نرم‌افزار را در اختیار تولید کننده قرار می‌دهد.

واژه‌های کلیدی: سرقت نرم‌افزار، محافظت از نرم‌افزار، نقش‌آب‌زنی گراف پویا، تکثیر غیر مجاز نرم افزار

۱- مقدمه

یکی از بزرگ‌ترین نگرانی‌ها در صنعت نرم‌افزار مساله سرقت نرم‌افزار است. در سال‌های اخیر رشد سریع و فراگیر استفاده از اینترنت مبدا به وجود آمدن نوع جدیدی از ارتباطات و تعاملات شده است. با استفاده از این فناوری می‌توان هر نوع سند دیجیتالی را تنها با چند بار کلیک موس به آسانی کپی و تکثیر نمود. سادگی، هزینه پایین و کیفیت بالا در این نوع تکثیر وتوزیع خود به مسئله سرقت نرم‌افزار[1] دامن زده است. بر این اساس روش‌های محافظت از نرم‌افزار طی یک دهه اخیر بیش از پیش مورد توجه واقع شده اند.

برای محافظت از نرم‌افزار روش‌های متفاوتی ارائه شده است که برخی مانند توکن نرم‌افزاری[2] [۲،۱]، پارش کد[3] [۳،۴]،

[1] شامل تکثیر غیرمجاز و استفاده مجدد غیرمجاز

[2] software token

[3] code partitioning

سالخوردگی نرم‌افزار[1] [۵] و نقش‌آب‌زنی نرم‌افزار[2] [۶]، بطور مستقیم روشی برای محافظت در برابر تکثیر غیرمجاز تمام و یا بخشی از نرم‌افزار ارائه می‌دهند و برخی از جمله *obfuscation*[۷،۸]، شناسایی دستکاری[3] [۹،۱۰،۱۱] و رمزنگاری[4] [۱۲،۱۳] بصورت غیر مستقیم با ارائه روشی برای جلوگیری از وارسی و دستکاری نرم‌افزار از حذف و یا تغییر کد محافظ نرم‌افزار پیشگیری می‌کنند.

در مجموع تا کنون روشی جامع و مقاوم برای حفاظت از نرم‌افزار در برابر سوء استفاده از کل و یا بخشی از آن ارائه نشده است. اما از میان روش‌های موجود، روش توکن نرم‌افزاری به دلیل هزینه پایین و قابلیت اجرایی بالا بیش از سایرین مورد توجه واقع شده است. در این مقاله با معرفی روشی جدید برای محافظت از نرم‌افزار، روشی برای بهبود انواع روش‌های توکن نرم‌افزاری ارائه می‌شود. این بهبود از طریق تعبیه توکن مورد نظر به عنوان یک نقش‌آب، در نرم‌افزار صورت می‌گیرد. برای این منظور از یکی از مقاوم‌ترین روش‌های نقش‌آب‌زنی پویا به نام «نقش‌آب‌زنی گراف پویا[5]» [۶] استفاده می‌شود. در این مقاله روشی نوین برای وارسی صحت مقدار نقش‌آب بدون نیاز به استخراج آن معرفی می‌شود. این وارسی برخلاف روش‌های متداول استخراج نقش‌آب، به صورت توزیع شده در ضمن اجرای برنامه انجام می‌پذیرد. از دیگر امتیازات روش ارائه شده نسبت به روش‌های رایج توکن نرم‌افزاری، امکان اعمال محدودیت در استفاده از وظیفه مندی‌های نرم‌افزار و افزایش سطح حفاظت از کل نرم‌افزار به کلاس‌های تشکیل دهنده آن است.

ادامه مقاله به این شرح سازماندهی شده است: در بخش۲ انواع روش‌های محافظت از نرم‌افزار به صورت مختصر معرفی و نقد شده است. بخش۳ به مرور اجمالی مفاهیم و کاربردهای نقش‌آب‌زنی نرم‌افزار و به طور خاص روش نقش‌آب‌زنی گراف پویا اختصاص داده شده است. در بخش۴ روش پیشنهادی این مقاله معرفی می‌شود و در بخش۵ این روش مورد ارزیابی قرار می‌گیرد. بخش۶ به مقایسه روش پیشنهادی با سایر روش‌های محافظت از نرم‌افزار می‌پردازد. در انتها بخش۷ به جمع بندی مطالب و معرفی پژوهش‌های بعدی اختصاص یافته است.

۲- نقدی بر روش‌های محافظت از نرم‌افزار

به طور کلی روش‌های محافظت از نرم‌افزار بر اساس آن که نیاز به سخت افزار خاصی دارند یا نه، به دو دسته مبتنی بر سخت افزار و مبتنی بر نرم‌افزار تقسیم می‌شوند[۱]. انواع روش‌های مبتنی بر سخت افزار علاوه بر تحمیل هزینه اضافی به ازای هر نسخه از نرم‌افزار، دارای روش‌های شکست شناخته شده ای هستند. تمرکز این مقاله برانواع روش‌های مبتنی بر نرم‌افزار است. این بخش به نقدی اجمالی بر این روش‌ها اختصاص یافته است.

انواع روش‌های توکن نرم‌افزاری با پیوند نرم‌افزار به یک مقدار و یا ویژگی منحصر به فرد به نام توکن، مانع تکثیر غیر مجاز نرم‌افزار می‌شوند. توکن می‌تواند یک کلید فعال سازی، فایل مجوز[6]، شماره سریال یک قطعه سخت افزاری، شماره سریال نسخه نرم‌افزار و دیگر موارد مشابه باشد که به صورت برخط[7] ویا برون خط[8] تخصیص می‌یابد. یک مثال خوب برای روش‌های توکن نرم‌افزاری کلید فعال سازی در ویندوز *XP* است[۱۴].

روش‌های توکن نرم‌افزاری دارای آسیب پذیری‌های شناخته شده ای هستند[۱۴]. یک سناریوی رایج برای شکست روش‌های توکن نرم‌افزاری حذف کد وارسی کننده صحت توکن مورد نظر است. برای تشخیص کد مربوطه، برنامه به ازای چند توکن ناصحیح اجرا شده و روال اجرایی برنامه با استفاده از انواع ابزارهای *debugger* پایش می‌شود. به این ترتیب نقطه شکست برنامه که کاندیدای وجود دستورات وارسی است، شناسایی می‌شود. پس از یک مرحله مهندسی معکوس دستورات، حول نقطه شکست، مجموعه دستورات وارسی کننده، شناسایی و حذف می‌شوند. سناریوی دیگر، تولید

[1] software aging
[2] software watermarking
[3] tamper detection
[4] encryption
[5] dynamic graph watermarking

[6] license file
[7] on-line
[8] off-line

تمامی‌توکن‌های مشابه و آزمون سازگاری آنها از طریق سعی و خطا است.

البته برای مقابله با سناریوی اول انواع روش‌های شناسایی دستکاری [۹،۱۰،۱۱]، *obfuscation* [۷،۸] و رمزنگاری [۱۲،۱۳] به عنوان راهکاری برای مقابله با حمله‌های مهندسی معکوس و دستکاری کد برنامه وجود دارند؛ اما ترکیب این روش‌ها با روش توکن نرم‌افزاری علی رقم بهبود سطح حفاظت با افت کارایی و مشکلاتی از قبیل پنهان کردن کلید رمز گشایی، شناسایی و حذف کد وارسی کننده صحت کد برنامه، مواجه است. همچنین برای مقابله با سناریوی دوم، تنها پیشنهاد، محدود کردن تعداد دفعات اجرای برنامه به ازای توکن‌های مختلف است که این روش هم با شناسایی و دستکاری حافظه شمارنده قابل شکست است.

روش‌های دیگری نیز برای مقابله با سرقت کل و یا بخشی از نرم‌افزار وجود دارند که اساس کار آنها افزایش احتمال شناسایی نسخه غیرمجاز است. از جمله این روش‌ها نقش‌آب‌زنی نرم‌افزار و سالخوردگی نرم‌افزار است. نقش‌آب‌زنی[۶] با تعبیه اطلاعاتی مبین هویت تولید و یا تکثیر کننده نرم‌افزار، مالک واقعی آن را مشخص می‌کند. اگرچه روش نقش‌آب‌زنی با افزایش ریسک سرقت، باعث کاهش انگیزه سوء استفاده کننده می‌شود اما راهکاری برای مقابله با تکثیر غیرمجاز، ارائه نمی‌دهد.

روش سالخوردگی نرم‌افزار[۵] با استفاده از سیاست ارائه مکرر گونه‌های جدید در بازه‌های زمانی کوتاه مانند ۲ تا ۳ ماه، باعث کاهش سودمندی نسخه‌های غیرمجاز ایجاد شده در بازه‌های زمانی فوق، می‌شود. البته عملیاتی کردن این روش در بسیاری از کاربردها ناممکن است.

دسته دیگری از روش‌های محافظت از نرم‌افزار پارش کد[۳،۴] است که با مجزا کردن کدهای حیاتی نرم‌افزار از سایر بخش‌های آن و عدم ارائه مستقیم کدهای حیاتی به کاربر، مانع تکثیر غیرمجاز نرم‌افزار می‌شود. از جمله روش‌های پیشنهادی ارائه کدهای حیاتی به صورت یک حافظه *ROM* و یا در قالب *RPC*[1] است. اگر چه انواع روش‌های پارش کد نسبت به سایر روش‌های محافظت از نرم‌افزار از آسیب پذیری کمتری برخوردار هستند اما دارای ضعف‌هایی مانند افت قابل ملاحظه کارایی، افزایش هزینه و محدودیت کاربری هستند.

در مجموع، تا کنون روشی جامع و مقاوم برای حفاظت از نرم‌افزار در برابر سوء استفاده از کل و یا بخشی از آن ارائه نشده است.

۳- معرفی نقش‌آب‌زنی گراف پویا

نقش‌آب‌زنی روشی برای درج یک سری اطلاعات محرمانه در نرم‌افزار است، به گونه ای که اولا هیچ گونه خدشه ای به وظیفه مندی‌های آن وارد نشود و ثانیا به صورت یکتایی قابل بازیابی باشد. به این اطلاعات محرمانه نقش‌آب گفته می‌شود. اگر چه جز تعدادی معدود [۱۵] در اکثر مقالات و نوشتجات علمی، نقش‌آب حامل اطلاعاتی درباره هویت تولید کننده نرم‌افزار بوده است و نقش‌آب‌زنی به عنوان روشی برای اثبات حق مالکیت تولید کننده نرم‌افزار معرفی شده است؛ اما نقش‌آب علاوه بر مشخصات تولید و یا تکثیر کننده نرم‌افزار می‌تواند شامل داده‌های مربوط به آزمون یکپارچگی نرم‌افزار، قوانین کنترل دسترسی، آزمون اعتبار نرم‌افزار و یا کلید رمز گشایی قسمتی از نرم‌افزار باشد. در این مقاله از نقش‌آب‌زنی برای تعبیه و وارسی توکن به عنوان اطلاعات وارسی مجوز استفاده از نرم‌افزار، در زمان اجرا استفاده می‌شود.

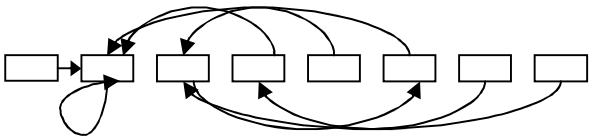
نقش‌آب‌زنی گراف پویا توسط *Collberge* وهمکارش به عنوان اولین روش نقش‌آب‌زنی پویا ارائه شده است[۶] و به دلیل ویژگی‌ها و قابلیت‌های آن نسبت به سایر روش‌های نقش‌آب‌زنی پویا همچنان از عمومیت و محبوبیت بالاتری برخوردار است.

در نقش‌آب‌زنی گراف پویا یک عدد صحیح مانند n به عنوان نقش‌آب در برنامه تعبیه می‌شود. برای تعبیه نقش‌آب، گرافی در تناظر یک به یک با n در نظر گرفته می‌شود. تکه کدی که وظیفه ساخت این گراف در حافظه *heap* را دارد، تولید می‌شود و این تکه کد به k قسمت تقسیم شده، به گونه ای در برنامه درج می‌شود که پس از وارد کردن رشته ورودی I گراف معادل نقش‌آب در حافظه ساخته شود. ویژگی مهم این روش

[1] Remote Procedure Call

آن است که به جای درج مستقیم مقدار نقش‌آب، کد تولید کننده نقش‌آب در قالب یک گراف را در برنامه درج می‌کند.

گراف‌هایی که دارای تناظر یک به یک با اعداد صحیح هستند گراف‌های شمارش پذیر نامیده می‌شوند. برای ایجاد یک گراف شمارش پذیر معادل عدد صحیح n روش‌های مختلفی وجود دارد[۱۶]. «کدنگاری شمارش مبنای- k^1» نمونه ای از روش‌های ایجاد تناظر میان عدد صحیح و گراف شمارش پذیر است. شکل (۱) گراف کدنگاری شمارش مبنای- ۸ معادل عدد ۴۶۲۰۱۴۸ را نشان می‌دهد. این گراف یک لیست پیوندی چرخشی از چپ به راست است که برای وضوح بیشتر، در شکل، پیوندهای چرخشی این لیست نشان داده نشده است. هر گره معادل یک رقم از عدد در مبنای ۸ است و هر یال خروجی از گره بیانگر مقدار آن رقم می‌باشد. عدم وجود یال بیانگر رقم صفر، وجود طوقه[2] بیانگر رقم یک، لینک به گره بعدی بیانگر رقم ۲ و به همین‌ترتیب الی آخر، است.



$$n = 4620148 = (21477564)_8$$

شکل (۱):گراف مبنای ۸ معادل عدد ۴۶۲۰۱۴۸

برای استخراج نقش‌آب، برنامه به ازای ورودی I اجرا شده و گراف مربوطه از $heap$ بازیابی می‌شود. نقش‌آب با توجه به ساختار این گراف محاسبه می‌شود.

۴- روش پیشنهادی

در روش پیشنهادی برای حفاظت هر یک از کلاس‌های نرم‌افزار در برابر سرقت و یا استفاده غیرمجاز، اطلاعاتی مبین محیط اجرایی مجاز نرم‌افزار در قالب یک توکن در هر یک از کلاس‌های نسخه مورد نظر تعبیه می‌شود. به این‌ترتیب تنها زمانی کاربر U اجازه استفاده از کلاس C از نرم‌افزار S را دارد که وارسی صحت توکن حاکی از سازگاری توکن تعبیه شده در C با محیط اجرایی آن در سیستم متعلق به کاربر U باشد. بنابراین تعبیه مکانیزم محافظت از نرم‌افزار شامل دو بخش تعبیه توکن و وارسی زمان اجرای آن است.

در روش پیشنهادی تعبیه و وارسی زمان اجرای توکن با استفاده از روش نقش‌آب‌زنی گراف پویا صورت می‌گیرد. به طور کلی عملیاتی کردن روش‌های نقش‌آب‌زنی پویا برای تعبیه و وارسی زمان اجرای اطلاعات با دو مشکل اساسی روبرو است. اول آنکه نقش‌آب در روش‌های پویا تنها پس از ورود یک رشته ورودی خاص تولید می‌گردد و به ازای سایر ورودی‌ها قابل دریافت و استخراج نیست. راه حل پیشنهادی این مقاله گنجاندن کدهای مربوط به تولید نقش‌آب در مسیر اصلی اجرای سازنده کلاس‌های تشکیل دهنده برنامه است. به این‌ترتیب اجرای این کدها در زمان بارگذاری کلاس به ازای هر نوع ورودی برنامه تضمین می‌شود.

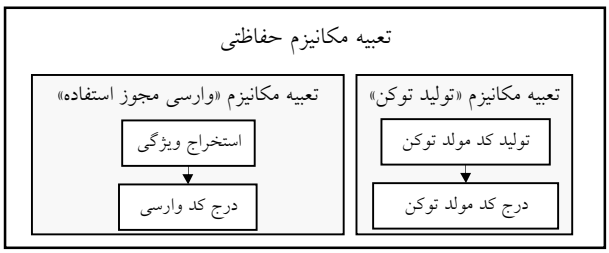
چالش دیگر در استفاده از نقش‌آب‌زنی برای تعبیه داده‌های کنترلی زمان اجرا، مساله عرضه روال استخراج است. از آنجا که اغلب روال استخراج، دوگان روال تعبیه است یک مهاجم با در اختیار داشتن روال استخراج می‌تواند به سادگی نقش‌آب را مکان یابی کند. در کاربردهایی مانند اثبات حق مالکیت، روال استخراج در اختیار استفاده کننده قرار نمی‌گیرد بنابراین مهاجم برای مکان یابی نقش‌آب باید تنها به یک سری ویژگی‌های آماری برنامه و بخت خود اکتفا کند. اما در کاربردهایی که نیازمند استخراج نقش آب در زمان اجرا است، به نظر می‌رسد از ارائه روال استخراج به استفاده کننده که یک مهاجم بالقوه است؛ گریزی نیست. دلیل اصلی تمرکز مقالات در بکارگیری روش نقش‌آب‌زنی در کاربرد اثبات حق مالکیت و عدم توسعه نقش آب زنی در دیگر کاربرد های بالقوه آن، همین مساله است.

راهکار پیشنهادی این مقاله برای مواجهه با مساله عرضه روال استخراج نقش‌آب، وارسی یک سری ویژگی‌های محلی در نقش‌آب، به جای استخراج یکباره و مقایسه درستی کل آن است. این ویژگی‌ها از نظر کمی و کیفی باید به گونه ای انتخاب شوند که احتمال سازگاری آنها با نقش‌آبی با مقدار

[1] Radix-k Encoding Enumeration
[2] loop

متفاوت با آنچه مورد تایید است، بسیار ناچیز باشد تا احتمال تایید نادرست کمینه گردد.

در روش پیشنهادی روش ساخت نرم‌افزار، شی گرا فرض شده است و سطح حفاظت از نرم‌افزار به کلاس‌های تشکیل دهنده آن ارتقا یافته است. به این‌ترتیب می‌توان مجوز استفاده را به ازای هریک از کلاس‌های اصلی و حیاتی نرم‌افزار صادر کرده و استفاده از وظیفه مندی‌های نرم‌افزار را در سطح کلاس‌ها محدود نمود. شکل(۲) اجزاء تعبیه مکانیزم حفاظتی در نرم‌افزار را در محیط تولید کننده نرم‌افزار و پیش از نصب نرم‌افزار نشان می‌دهد. هر یک از این اجزاء در ادامه به اختصار شرح داده می‌شوند. ورودی روال «تعبیه مکانیزم حفاظتی» هریک از کلاس‌های نرم‌افزار که باید مورد حفاظت واقع شود، است و خروجی آن نرم‌افزار آماده ارائه به کاربر می‌باشد.



شکل (۲) فعالیت‌های تعبیه مکانیزم حفاظت از نرم‌افزار

۴-۱- تعبیه مکانیزم «تولید توکن»

مکانیزم تولید توکن براساس روش نقش‌آب‌زنی گراف پویا شامل دو بخش: (۱)تولید تکه کد سازنده گراف معادل مقدار توکن و (۲)درج این کد در لا به لای کدهای متد *create* کلاس مورد نظر است. به این‌ترتیب بلافاصله پس از بارگذاری کلاس با اجرای متد *create*، مقدار توکن بر اساس ویژگی‌های محیط اجرایی کلاس، محاسبه و در قالب یک گراف در حافظه *heap* ساخته می‌شود.

تولید کد مولد توکن

کد مولد توکن در قالب یک گراف شمارش پذیر، دارای چهار بخش: (۱)کد جمع آوری کننده اطلاعات محیطی کلاس در راستای محاسبه توکن در زمان اجرا، (۲)محاسبه توکن، (۲)کد تولید کننده نقش‌آب بر اساس توکن محاسبه شده و (۳)کد سازنده گراف معادل نقش‌آب محاسبه شده، است. تصمیم گیری درباره نوع اطلاعات محیطی مورد نظر و نحوه محاسبه توکن وابسته به روش توکن نرم‌افزاری انتخابی است[۱۴]. تابع مولد نقش‌آب بر اساس مقدار توکن، در شکل ساده خود، می‌تواند توکن را به صورت یک رشته باینری در نظر گرفته و عدد صحیح معادل این رشته باینری را به عنوان خروجی تولید کند. همچنین هر نوع تابع دیگری که بتواند یک رابطه یک به یک میان توکن و یک عدد صحیح برقرار کند نیز قابل قبول است.

درج کد مولد توکن

برای درج کد مولد توکن در نرم‌افزار، آن را به چند بخش تقسیم کرده و آن را در بین کدهای مسیر اصلی متد *create* کلاس مورد نظر، اضافه می‌کنیم. به این‌ترتیب کد مولد توکن در ضمن اجرای متد *create* به طور کامل اجرا می‌شود.

۴-۲- تعبیه مکانیزم «وارسی مجوز استفاده»

در زمان اجرای نرم‌افزار به منظور وارسی سازگاری ویژگی‌های فعلی محیط اجرایی کلاس با آنچه مورد تایید است، یک سری روابط عددی میان ارقام نقش‌آب مورد تایید استخراج شده و وجود این روابط در زمان اجرا در گراف تشکیل شده توسط متد *create*، که از این پس آن را G می‌نامیم، بررسی می‌شود. این بررسی در ضمن اجرای مسیر اصلی سایر متدهای کلاس مورد نظر صورت می‌گیرد.

بنابراین تعبیه مکانیزم وارسی مجوز استفاده، شامل دو بخش: (۱)استخراج یک سری روابط و ویژگی‌ها میان ارقام نقش‌آب مورد تایید و (۲)تولید و درج کد وارسی کننده این ویژگی‌ها است.

استخراج ویژگی

برای استخراج ویژگی‌هایی در مقدار نقش‌آب مورد تایید لازم است ابتدا توکن براساس اطلاعات سیستم مشتری محاسبه و سپس نقش‌آب براساس آن به دست آید. برای محاسبه توکن مورد تایید، لازم است اطلاعات لازم از طریق برقراری ارتباط با سیستم مشتری جمع آوری شود. پس از جمع آوری این

اطلاعات بر اساس انتخاب‌های صورت گرفته در فعالیت «تولید کد مولد توکن» برای (۱)نوع توکن نرم‌افزاری و (۲)تابع نگاشت توکن به یک عدد صحیح انتخاب شده، به‌ترتیب مقدار توکن و عدد صحیح متناظر با آن که مقدار نقش‌آب را تشکیل می‌دهد، محاسبه می‌شود. سپس یک سری روابط عددی میان ارقام نقش‌آب مورد انتظار شناسایی می‌شود.

این روابط می‌تواند میان یک یا چند رقم نقش‌آب برقرار باشد و یا خیلی ساده مبین مقدار یک رقم خاص از نقش‌آب باشد. به عنوان نمونه در مثال شکل(۱) مجموع سه رقم اول نقش‌آب در مبنای۸ برابر رقم چهارم نقش‌آب در مبنای۸ است.

درج کد وارسی

به منظور وارسی سازگاری ویژگی‌های فعلی محیط اجرایی کلاس مورد نظر با آنچه مورد تایید و انتظار است، به جای استخراج یکباره نقش‌آب و مقایسه آن با معادل عددی توکن، معادل ویژگی‌های فعلی محیط اجرایی کلاس، وجود ویژگی‌ها و روابط استخراج شده در مرحله قبل در گراف G، در زمان اجرا بررسی می‌شود.

برای این منظور به ازای هر ویژگی استخراج شده کدی حاوی کنترل برقراری این ویژگی در گراف G تولید و در مسیر اصلی سایر متدهای کلاس مورد نظر درج می‌شود. برای تخصیص این وارسی‌ها به متدهای کلاس یک انتخاب آن است که متناسب با میزان ارزش افزوده هر متد، وظیفه بررسی روابط به متدها تخصیص داده شود. بنابراین در متدهایی که هیچ گونه ارزش افزوده ای ندارند و به عنوان مثال تنها واسط کاربر را مدیریت می‌کنند، هیچ گونه وارسی انجام نمی‌گیرد.

دقت داشته باشید که برای وارسی ویژگی‌های استخراج شده لازم نیست ارقام نقش‌آب حتی به صورت محلی صریحا استخراج شوند. به عناون مثال برای وارسی ویژگی ذکر شده در بخش پیشین لزومی‌ندارد چهار رقم اول نقش‌آب استخراج شود بلکه کافی است اشاره گر p به‌ترتیب مقصد گره‌های اول، دوم و سوم را به صورت یک گره در هرگام دنبال کند و هم زمان با هر گام p اشاره گر q مقصد گره چهارم را گره به گره دنبال نماید. تنها در صورتی برقرای رابطه مورد نظر تایید می‌شود که اشاره گر مربوط به گره چهارم هم زمان با اشاره گر گره سوم به مقصد برسد.

۵- ارزیابی روش پیشنهادی

روش پیشنهادی از دو منظر سطح حفاظت و مقاومت در برابر حملات احتمالی قابل ارزیابی است.

سطح حفاظت

در صورتی که توکن حاوی اطلاعاتی متناظر با محیط اجرایی مجاز نرم‌افزار مانند شماره سریال دیسک سخت و یا ویژگی از سیستم عامل باشد، با استفاده از این روش می‌توان هر یک از کلاس‌ها را در برابر تکثیر غیرمجاز و استفاده مجدد غیرمجاز محافظت کرد.

در صورتی که روال وارسی توکن فقط و فقط در روال‌هایی که دارای ارزش افزوده برای کاربر هستند مانند روال‌هایی که نتایج محاسبات را بر روی دیسک و یا پایگاه داده منعکس می‌کنند؛ تعبیه شود و توکن حاوی ویژگی‌های همیشه غلط از محیط سیستم باشد، با استفاده از این روش می‌توان یک گونه آزمایشی از نرم‌افزار تهیه کرد.

میزان مقاومت

ستون چپ جدول(۱) فهرستی از حملات شناخته شده[۱۷] بر علیه روش‌های محافظت از نرم‌افزار را نشان می‌دهد. ستون راست این جدول مبین مقاوم و یا آسیب پذیر بودن روش پیشنهادی در برابر هر یک از این حملات است.

برای اعمال حمله *key generation* لازم است بتوان تمامی‌حالات ممکن برای کلید را که در اینجا همان توکن است، تولید کرده و بررسی کرد که آیا با جایگزینی آن با کلید فعلی باز هم روال اجرایی برنامه در سیستم مهاجم با شکست مواجه می‌شود یا خیر. اما در روش پیشنهادی اولا وجود توکنی که هم مطابق خصوصیات محیطی نرم‌افزار باشد و هم دارای ویژگی‌های درج شده در روال وارسی باشد، بعید به نظر می‌رسد و ثانیا بررسی روال اجرایی نرم‌افزار به ازای توکن

جدید منوط به درج مجدد مکانیزم وارسی نرم‌افزار است که با هزینه بالایی برای مهاجم همراه می‌باشد.

جدول ۱: آسیب پذیری روش پیشنهادی دربرابر حملات

مدل حمله	آسیب پذیری
key generation	به صورت خودکار غیر ممکن
reverse engineering	همراه با صرف هزینه‌های نمایی
static code analysis	همراه با صرف هزینه‌های نمایی
single check point failure	افزایش خطی هزینه مشابه برای روش توکن نرم افزاری
monitor program's instructions	افزایش خطی هزینه مشابه برای روش توکن نرم افزاری
monitor network connections	کاملا مقاوم
modify operating system kernel	آسیب پذیر

بنابر اصل مشکل بودن *alias analysis* تحلیل و بررسی کدهای مربوط به تولید و وارسی گراف پویا با صرف هزینه های نمایی[1] همراه است[۱۸]. به علاوه مقدار مورد تایید برای توکن در هیچ کجای برنامه به صورت صریح نگهداری نمی‌شود. بنابراین حمله‌های *reverse engineering* و *static code analysis* بی اثر هستند. همچنین از آنجایی که مکانیزم وارسی در کل کلاس توزیع شده است هزینه اعمال حملات *single check point failure* و *monitor program's instructions* چندین برابر شده است.

حمله *monitor network connections* در روش مورد نظر مصداقی نمی‌تواند داشته باشد.

علی رقم مقاومت در برابر حملات ذکر شده روش پیشنهادی در برابر حملاتی که به نحوی محیط اجرایی نرم‌افزار را به صورت مجازی شبیه سازی می‌کنند، مانند حمله *modify operating system kernel* آسیب پذیر است. ضعف دیگر این روش که ضعف سرشتی روش‌های توکن نرم‌افزاری است؛ حساسیت بالا نسبت به ویژگی‌های محیطی نرم‌افزار است. برای مثال نرم‌افزاری که به شماره سریال دیسک سخت پیوند خورده است در صورت خرابی دیسک سخت، کاربر را دچار مشکل می‌کند.

۶- مقایسه با سایر روش‌های محافظت از نرم‌افزار

روش پیشنهادی در مقایسه با انواع روش‌های مبتنی بر سخت افزار، وابسته به تامین یک سخت افزار اضافی به ازای هر نسخه نیست. هر چند این روش می‌تواند با هر یک از روش‌های محافظتی مبتنی بر سخت افزار ترکیب شود.

این روش مانند روش‌های توکن نرم‌افزاری مزیت پیوند نرم‌افزار به یک سری اطلاعات خارج از آن را دارد ولی برخلاف آن‌ها در برابر حملات *key generation* و *single check point failure* آسیب پذیر نیست. برخلاف روش رمزنگاری سربار اجرایی بالایی را تحمیل نمی‌کند ولی مانند آن قابلیت محافظت از تک تک اجزای نرم‌افزار را تا سطح کلاس‌ها دارا می‌باشد. همچنین به دلیل مشکل بودن *alias analysis* در برابر حمله *reverse engineering* مقاوم است و نیازی به استفاده از انواع روش‌های *obfuscation* نیست.

روش پیشنهادی برخلاف سایر روش‌های نقش‌آب‌زنی نرم‌افزار که دارای ماهیتی منفعل هستند، فعال است و دربرابر استفاده غیرمجاز از نرم‌افزار عکس العمل نشان می‌دهد. همچنین مکانیزم وارسی آن بر خلاف سایر روش‌ها نقش‌آب را به صورت صریح و مستقیم استخراج و وارسی نمی‌کند در نتیجه از میزان نهفتگی بالاتری برخوردار است.

مانند روش‌های مبتنی بر سالخوردگی نرم‌افزار کنترل مجوز استفاده از نرم‌افزار را به تعویق نمی‌اندازد و برخلاف روش‌های پارش کد کاربر را مجبور به برقراری ارتباط مستمر با سرور و یا استفاده از سخت افزار خاصی، نمی‌کند.

۷- نتیجه گیری

در این مقاله روشی برای کاهش آسیب پذیری‌های روش توکن نرم‌افزاری با استفاده از رویکرد نقش‌آب‌زنی نرم‌افزار ارائه شده و با ارائه روشی نو مساله عرضه روال استخراج نقش‌آب به کاربر، پوشش داده شده است. این روش امکان محافظت از هر یک از کلاس‌های نرم‌افزار در برابر تکثیر غیرمجاز و استفاده

[1] از مرتبه 2^n [۱۸]

مجدد غیرمجاز را فراهم می‌آورد. به علاوه امکان ارائه یک گونه آزمایشی از نرم‌افزار را در اختیار تولید کننده قرار می‌دهد.

طی ارزیابی‌های انجام شده نشان داده شد که روش پیشنهادی در برابر انواع حملات مطرح در دامنه محافظت از نرم‌افزار، به استثنا شبیه سازی محیط اجرایی نرم‌افزار مانند حمله *modify operating system kernel* مقاوم است.

از آنجایی که روش پیشنهادی جزء اولین کوشش‌ها در زمینه استفاده از رویکرد نقش‌آب‌زنی برای وارسی‌های زمان اجرا است، توسعه کاربری این رویکرد برای وارسی زمان اجرا در کاربردهای دیگر به عنوان پژوهش‌های آتی پیشنهاد می‌شود.

۸- تشکر و قدردانی

این تحقیق با حمایت مالی مرکز تحقیقات مخابرات ایران انجام شده است.

۹- مراجع

[1] P. T. Devanbu and S. Stubblebine, "Software Engineering for Security: A Roadmap", *The Future of Software Engineering*, ACM Press, pp. 227-239, 2000.

[2] B. S. Joshi, "Computer software security system", *United States Patent 4*, pp. 688-169, 1987.

[3] T. Sander and C. Tschudin, "On sofware protection via function hiding", *In 2nd International Workshop on Information Hiding*, 1998.

[4] L. J. Tolman and A. J. Etstrom, "Anti-piracy system using separate storage and alternate execution of selected public and proprietary portions of computer programs", *United States Patent 4*, pp. 646-234, 1987.

[5] M. Jakobsson and K. Reiter, "Discouraging Software Piracy Using Software Aging", *Proceedings of 1st ACM Workshop on Security and Privacy in Digital Rights Management*, LNCS, vol. 2320, pp. 1-12, 2002.

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

[7] F. Cohen, "Operating System Protection Through Program Evolution", *Computers and Security*, vol. 12, no. 6, pp. 565–584, 1993.

[8] C. Collberg and C. Thomborson, "Watermarking, Tamper-Proofing, and Obfuscation – Tools for Software Protection", *IEEE Transactions on Software Engineering*, vol. 28, no. 8, pp. 735-746, 2002.

[9] M. J. Atallah, E. D. Bryant and M. R. Stytz, "A survey of anti-tamper technologies". *CROSSTALK, The Journal of Defense Software Engineering*, pp. 12-16, 2004.

[10] B. Horne, L. Matheson, C. Sheehan and R. Tarjan, "Dynamic Self-Checking Techniques for Improved Tamper Resistance", *Proc. 1st ACM Workshop on Digital Rights Management*, Springer LNCS 2320, pp. 141–159, 2002.

[11] Y. Chen, R. Venkatesan, M. Cary, R. Pang, S. Sinha and M. Jakubowski, "Oblivious Hashing: A Stealthy Software Integrity Verification Primitive", *Proc. 5th Information Hiding Workshop*, Springer LNCS 2578, pp. 400–414, 2002.

[12] A. Herzberg, S. S. Pinter, "Public protection of software", pp. 371–393, *ACM Trans.Computer Systems*, vol.5, no.4, 1987.

[13] R. Anderson, "Information Hiding", LNCS, vol. 1174, Cambridge, U.K., 1996.

[14] G. Hachez, "A Comparative Study of Software Protection Tools Suited for E-Commerce with Contributions to Software Watermarking and Smart Cards," *Ph.D. Thesis*, Universite Catholique de Louvain, Belgium, ch. 2, pp. 38-39, 2003.

[15] J. Nagra, C. Thomborson and C. Collberg, "A functional taxonomy for software watermarking", *In Proc. of the 25th Australasian Computer Science Conference*, pp. 177-186, 2002.

[16] F. Harary and E. Palmer, "Graphical Enumeration". *Academic Press*, New York, 1973.

[17] B. Blietz, "Software Tamper Resistance Through Dynamic Monitoring", Phd thesis, Iowa State University, 2004.

[18] 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.