دوفصل‌نامه علمی منادی امنیت فضای تولید و تبادل اطلاعات (افتا)
سال ۱۴۰۰، جلد ۱۹، شماره ۲ (۸۰–۸۷)
https://monadi.isc.org.ir

ارائه‌شده در هفدهمین کنفرانس بین المللی انجمن رمز ایران در دانشگاه علم و صنعت ایران، ۱۹ و ۲۰ شهریور ۱۳۹۹

ارائه ساز و کاری برای نظارت مالک بر قرارداد هوشمند در اتریوم★

مرتضی امیرمحسنی و صادق دری نوگورانی*

دانشکده مهندسی برق و کامپیوتر، دانشگاه تربیت‌مدرس، تهران، ایران

اطلاعات مقاله

کلمات کلیدی:
زنجیره بلوک
قرارداد هوشمند
آسیب‌پذیری
دسته
نظارت زمان اجرا

dor: 10.0000/000000000

نوع مقاله: پژوهشی

چکیده

قراردادهای هوشمند برنامه‌هایی هستند که بر روی زنجیره بلوک ثبت شده‌اند و کاربران می‌توانند از طریق تراکنش‌ها آن‌ها را اجرا کنند. نتایج اجرای قراردادهای هوشمند توسط اعضا بررسی و بر روی زنجیره بلوک ثبت می‌شود. ممکن است مبلغ قابل توجهی رمزارز در اختیار یک قرارداد هوشمند باشد و با اجرا شدن آن، این دارایی مدیریت و جابجا شود؛ بنابراین ضروری است که قرارداد در مقابل تهدیدها و حملات امن باشد. در حوزه امنیت قراردادهای هوشمند پژوهش‌هایی انجام شده، و ابزارهایی برای بررسی کد قرارداد قبل از قرارگرفتن روی زنجیره بلوک ارائه شده است. این ابزارها با بررسی کد قرارداد آسیب‌پذیری‌های آن را برای اصلاح به برنامه‌نویس اعلام می‌کنند. با وجود این ممکن است قرارداد دارای نوعی آسیب‌پذیری باشد که توسط ابزارهای موجود قابل شناسایی نیست. در برخی از چنین مواردی می‌توان در زمان حمله نشانه‌هایی را تشخیص داد که دال بر سوءاستفاده هستند. در این مقاله ساز و کاری معرفی شده است تا امکان نظارت زمان اجرا بر روی قراردادهای هوشمند بستر اتریوم را فراهم می‌کند. با این ساز و کار مالک هنگام قرارگیری کد بر روی زنجیره بلوک، تعدادی محافظ امنیتی برای قرارداد مشخص می‌کند. با تغییراتی که در فرایند درستی‌سنجی تراکنش‌ها در استخراج‌گر اتریوم داده می‌شود، به محض برقرار شدن شرایط محافظ‌ها، استخراج‌گر جلوی اجرای تراکنش مربوط به آن را خواهد گرفت. ساز و کار پیشنهادی مکمل روش‌های تحلیل کد (ایستا و پویا) است و توانایی تشخیص و جلوگیری از حملاتی را دارد که برای ابزارهای تحلیل کد ناممکن است.

© ۱۴۰۰ انجمن رمز ایران

۱ مقدمه

اتریوم[۱]، زنجیره بلوک پیشرو در جهان در زمینه ایجاد قراردادهای هوشمند[۲] یا برنامه‌های غیرمتمرکز (DApp) می‌باشد. در اتریوم استخراج‌گران برای به دست آوردن اتر[۳] (رمز ارز بومی اتریوم) فعالیت می‌کنند. اتر همچنین به منظور پرداخت هزینه‌های مورد نیاز بابت کارمزد تراکنش‌ها[۴] و سایر خدمات شبکه مورد استفاده قرار می‌گیرد.

قرارداد هوشمند یک کد برنامه نویسی بر روی زنجیره بلوک است. این کد می‌تواند هر الگوریتمی را پیاده‌سازی کند؛ و معمولاً مبادله پول، دارایی، سهام و ... را تسهیل می‌کند. قرارداد هوشمند می‌تواند شرایط موجود را بررسی کند و به طور خودکار بدون نیاز به دخالت شخص ثالث عمل مناسب را انجام دهد. از آن جا که قراردادهای هوشمند روی

★از کمیته علمی هفدهمین کنفرانس بین‌المللی انجمن رمز ایران برای داوری این مقاله تشکر می‌شود.
*نویسنده مسئول
آدرس‌های رایانامه: m.amirmohseni@modares.ac.ir (مرتضی امیرمحسنی)، dorri@modares.ac.ir (صادق دری نوگورانی)
© ۱۴۰۰ تمامی حقوق متعلق به انجمن رمز ایران است.

[1] Ethereum [2] smart contracts [3] Ether [4] transactions fee

زنجیره بلوک اجرا می‌شوند، دقیقاً همان‌گونه که برنامه‌نویسی شده‌اند، بدون دستکاری، سانسور، خرابی، و بدون نیاز به دخالت شخص ثالث توسط همه استخراج‌گران [1] اتریوم اجرا می‌شوند و نتایج آن‌ها پس از توافق جمعی بر روی زنجیره بلوک ثبت می‌شود.

امروزه یکی از مسائل مهم در حوزه فناوری زنجیره بلوک و قراردادهای هوشمند، امنیت قراردادهای هوشمند است. قراردادهای هوشمند اتریوم اغلب برای کاربردهای مالی استفاده می‌شوند و پس از اینکه ثبت شوند، امکان تغییر کد آن‌ها برای رفع عیب و سایر اصلاحات ممکن نیست. لذا باید کد قراردادها امن و فاقد آسیب‌پذیری باشد. تاکنون پژوهش‌های متعددی در زمینه امنیت قراردادهای هوشمند انجام شده و ابزارهایی به این منظور تولید شده است. این ابزارها کد منبع و/یا زبان ماشین EVM [2] قرارداد را دریافت می‌کنند و پس از تحلیل، آسیب‌پذیری‌های موجود در آن را برای اصلاح توسط برنامه‌نویس اعلام می‌کنند. در این مقاله ساز و کاری معرفی خواهد شد که مکملی برای ابزارهای موجود می‌باشد. این ساز و کار نظارت روی قرارداد هوشمند را در هنگام اجرا ممکن می‌سازد و جلوی سوءاستفاده از آسیب‌پذیری‌هایی که قبل از اجرای واقعی قابل شناسایی نیستند را می‌گیرد.

نوآوری‌های این مقاله عبارتند از: تشخیص سوءاستفاده از آسیب‌پذیری‌ها با معرفی و استفاده از محافظ‌های امنیتی، جلوگیری از سوءاستفاده در زمان اجرای قرارداد، امکان تشخیص آسیب‌پذیری‌های ناشناخته و امکان استفاده از محافظ‌های امنیتی برای مالک قرارداد حتی بدون نیاز به دانش برنامه‌نویسی. در ادامه مقاله ابتدا به بررسی پیش‌زمینه‌ها در بخش ۲ می‌پردازیم؛ سپس در بخش ۳ مروری بر کارهای انجام‌شده خواهیم داشت. در بخش ۴ روش پیشنهادی ارائه، و در بخش ۵ نحوه پیاده‌سازی بیان می‌شود. در بخش ۶ ساز و کار پیشنهادی با کارهای پیشین مقایسه می‌شود. نهایتاً مقاله با جمع‌بندی در بخش ۷ به پایان می‌رسد.

۲ پیش‌زمینه‌ها

زنجیره بلوک یک دفتر کل توزیع‌شده مورد اعتماد است که داده را بین شبکه‌ای نظیر به نظیر به اشتراک می‌گذارد. زنجیره بلوک اولین بار توسط شخصی ناشناس با نام مستعار ناکاموتو ساتوشی، خالق بیت‌کوین، به منظور انتقال مستقیم ارزهای دیجیتال بدون نیاز به عامل سوم مورد اعتماد معرفی شد. همان‌طور که از نام زنجیره بلوک مشخص است، زنجیره‌ای از بلوک‌ها به ترتیب زمان ایجاد آن‌ها می‌باشد. هر بلوک با یک مقدار درهم‌سازی‌شده‌ای [3] شناسایی می‌شود و به بلوک قبلی خود با اشاره به مقدار درهم‌سازی‌شده آن بلوک ارجاع داده می‌شود. تنها استثنا، اولین بلوک می‌باشد که قبل از آن بلوکی وجود ندارد و به بلوک دیگری اشاره نمی‌کند.

کاربردهای زنجیره بلوک تنها محدود به تراکنش‌های مالی نمی‌شود. این امکان وجود دارد که برنامه‌هایی را به نام قراردادهای هوشمند که مربوط به تراکنش‌های پیچیده‌ای هستند و به صورت خودکار اجرا می‌شوند، ایجاد کرد. قرارداد هوشمند برنامه‌ای است که روی زنجیره بلوک ذخیره می‌شود و به صورت خودکار و بدون نیاز به واسطه‌های مورد اعتماد، شرایط مورد نظرش را اجرا می‌کند. قراردادهای هوشمند باعث ایجاد جهشی قابل توجه در دنیای زنجیره بلوک شدند و عصر جدیدی را برای زنجیره بلوک ایجاد کرده‌اند.

۱.۲ اتریوم

اتریوم اولین بستر عمومی زنجیره بلوک می‌باشد که از قراردادهای هوشمند پیشرفته و اختصاصی با استفاده از ماشین مجازی تورینگ کاملی [4] به نام ماشین مجازی اتریوم پشتیبانی می‌کند. ماشین مجازی اتریوم محیط زمان اجرا برای قراردادهای هوشمند می‌باشد و هر گره [5] شبکه اتریوم یک پیاده‌سازی از ماشین مجازی اتریوم و دستوراتی مشابه را اجرا می‌کند. برای برنامه‌نویسی قراردادهای هوشمند اتریوم چند زبان برنامه‌نویسی سطح بالا ارائه شده است و محبوب‌ترین آن‌ها، Solidity نام دارد. کد منبع قرارداد به هریک از زبان‌های مذکور به بایت‌کد ماشین مجازی اتریوم کامپایل می‌شود و روی زنجیره بلوک ثبت می‌شود تا اجرا شود.

اتریوم در حال حاضر می‌تواند برای طراحی انواع برنامه‌های غیرمتمرکز در زمینه‌های سیستم‌های سلامت، زنجیره تأمین، اینترنت اشیاء، هویت دیجیتالی، نگهداری مدارک، رأی‌گیری و ... مورد استفاده قرار بگیرد.

۲.۲ قرارداد هوشمند

قرارداد هوشمند مجموعه‌ای از توابع است که هر یک با دنباله‌ای از دستورات بایت‌کد تعریف شده‌اند. یکی از ویژگی‌های قابل توجه قراردادهای هوشمند این است که آن‌ها می‌توانند به/از کاربران و قراردادهای دیگر اتر منتقل کنند. زمانی که یک قرارداد با موفقیت ایجاد شود، توسط یک آدرس شناسایی می‌شود. هر قرارداد می‌تواند مقداری رمزارز اتر در اختیار داشته باشد، دارای فضای ذخیره‌سازی اختصاصی، و کد اجرایی از قبل تعریف‌شده‌اش می‌باشد.

کاربران تراکنش‌ها را به سه منظور به شبکه اتریوم ارسال می‌کنند: (۱) ایجاد قرارداد جدید، (۲) فراخوانی توابع قرارداد و (۳) انتقال اتر از/به قراردادها یا کاربران دیگر. تمامی تراکنش‌ها روی داده‌ساختار عمومی و فقط قابل افزودن زنجیره بلوک ذخیره می‌شوند. توالی تراکنش‌ها روی زنجیره بلوک تعیین‌کننده سابقه و وضعیت هر قرارداد و موجودی هر قرارداد و کاربر می‌باشد.

در اتریوم، کارمزدی بابت اجرای تراکنش‌های مالی و فراخوانی قراردادهای هوشمند به استخراج‌گران پرداخته می‌شود که با واحد گس [6] سنجیده می‌شود. گس را می‌توان به عنوان سوخت موردنیاز در شبکه اتریوم در نظر گرفت. این هزینه بابت اعتبارسنجی تراکنش، استخراج

[1] miners [2] Ethereum Virtual Machine [3] hashed value

[4] Turing complete [5] node [6] gas

بلوک و به طور کلی محاسبات سنگینی است که استخراجگران انجام می‌دهند. همچنین پرداخت گس به عنوان یک عامل بازدارنده در برابر اتلاف منابع شبکه با تراکنش‌های کم‌ارزش و ناخواسته، عمل می‌کند.

قراردادها عموماً دارای اندوخته اتر و واجد ارزش اقتصادی می‌باشند؛ لذا ضروری است که درست اجرا شدن آن‌ها تضمین شود. برای این منظور اتریوم به یک مرکز مورد اعتماد متکی نیست؛ در عوض، هر تراکنش توسط شبکه بزرگی از اعضای غیرقابل اعتماد برای یکدیگر به نام استخراجگران پردازش می‌شود. تضادهای موجود در اجرای قراردادها توسط یک پروتکل اجماع بر مبنای معماهای اثبات کار[1] حل می‌شود. در حالت ایده‌آل اجرای قراردادها زمانی درست است که حمله‌کننده نتواند اکثریت توان محاسباتی شبکه را تحت کنترل خود داشته باشد.

۳ کارهای انجام‌شده

در این بخش به معرفی تعدادی از پژوهش‌های اخیر در حوزه امنیت قراردادهای هوشمند می‌پردازیم. Atzei و همکاران در [۱] آسیب‌پذیری‌های مربوط به قراردادهای هوشمند اتریوم را در سه دسته، شامل مسائل مربوط به (۱) زبان Solidity، (۲) بایت‌کد ماشین مجازی اتریوم، و (۳) زنجیره بلوک تقسیم کرده، و هر یک را با جزئیات معرفی کرده‌اند.

Loi Luu و همکاران در [۲] ابزار Oyente را توسعه داده‌اند که ابزاری برای بررسی قراردادهای هوشمند اتریوم بر پایه اجرای نمادین[2] می‌باشد و می‌تواند چهار آسیب‌پذیری رایج قراردادها را تشخیص دهد. با بررسی حدود ۱۹۰۰۰ قرارداد توسط این ابزار مشخص شده است که بیش از ۴۵ درصد آن‌ها دارای حداقل یکی از آسیب‌پذیری‌های بیان‌شده بوده‌اند.

در [۳] Tsankov و همکاران، یک بستر بررسی امنیت قراردادهای هوشمند به نام Securify را توسعه داده‌اند که قرارداد را با استفاده از الگوهای امنیتی (مجموعه الگوهای مطابقت و نقض) بررسی می‌کند. تمام رفتارهای قرارداد با یکی از موارد مطابقت، نقض و هشدار برچسب زده، و به کاربر اعلام می‌شوند.

Chen و همکارانش در [۴] نقطه ضعف مصرف غیرضروری گس را در قراردادهای هوشمند بررسی کرده‌اند. آن‌ها هفت الگوی بهینه‌شده بر مبنای زبان Solidity برای قراردادهای هوشمند معرفی می‌کنند. آن‌ها همچنین ابزاری به نام Gasper را معرفی کرده‌اند که به منظور تشخیص خودکار سه مورد از هفت الگوی بیان‌شده که بیشتر به وقوع می‌پیوندند، استفاده می‌شود.

Joshua Ellul و همکارش در [۵] ابزار ContractLarva را برای درستی‌سنجی زمان اجرای[3] قراردادهای هوشمند نوشته‌شده به زبان Solidity توسعه داده‌اند. این ابزار، قرارداد و همچنین خصوصیات تعیین‌شده و مربوط به قرارداد را به عنوان ورودی دریافت می‌کند و در نهایت یک قرارداد جدید به زبان Solidity تولید می‌کند که رفتارش همانند قرارداد اولیه است اما رفتار زمان اجرا را نیز در برابر خصوصیات بررسی می‌کند.

Josselin Feist و همکاران در [۶] یک چارچوب تحلیل ایستای[4] قراردادهای هوشمند به نام Slither معرفی کرده‌اند. این ابزار کد Solidity را دریافت کرده و آن را به صورت نمایش کد میانی به نام SlithIR تبدیل می‌کند که استفاده از تکنیک‌های مرسوم تحلیل برنامه‌ها را ممکن می‌سازد. این ابزار دارای چهار کاربرد اساسی شامل ۱) تشخیص خودکار آسیب‌پذیری‌ها، ۲) بهبود درک کاربر از قراردادها، ۳) کمک به بررسی کد و ۴) تشخیص خودکار بهینه‌سازی کد می‌باشد.

راهکارهایی که تاکنون برای امنیت قراردادهای هوشمند ارائه شده‌اند نیازمند بررسی و بعضاً دستکاری کد منبع قرارداد هستند؛ در حالی که اولاً همه حملات ناشی از آسیب‌پذیری‌هایی نیستند که بتوان با تحلیل کد منبع شناسایی کرد؛ ثانیاً به دلیل اتصال شُل[5] و تعیین آدرس پویای برخی فراخوانی‌های میان‌قراردادی، برخی آسیب‌پذیری‌ها تا زمانی که مورد سوءاستفاده قرار نگیرند قابل تشخیص نمی‌باشند. مشکلی که اغلب ابزارهای معرفی شده دارند، توانایی تشخیص تعداد محدودی از آسیب‌پذیری‌هاست که قبلاً شناخته شده‌اند. اما با نظارت زمان اجرای قراردادها که به صورت محدود در [۵] پیاده‌سازی شده است، و موضوع مقاله حاضر نیز هست، هنگام هر رخداد ناخواسته عمدی (حمله) یا غیرعمد (باگ)، امکان جلوگیری وجود دارد.

از منظر کاربرپسندی، در تمامی ابزارهای موجود کد قرارداد باید تحلیل شود و نتایج آن برای کاربرانی قابل استفاده است که دانش برنامه‌نویسی داشته باشند؛ در حالی که ممکن است کاربری علاقمند به کاربردهای مختلف قراردادهای هوشمند باشد، اما دانش برنامه‌نویسی نداشته، و خودش کد قرارداد را ننوشته باشد. مثال بسیار معروف چنین حالتی، کیف پول Parity است که به علت وجود آسیب‌پذیری کاربران بسیاری را تحت تأثیر قرار داد [۷]. با توجه به این موضوع در این مقاله ساز و کاری ارائه می‌شود که به صورتی کاربر پسند، تعدادی محافظ امنیتی[6] در اختیار مالک قرارداد قرار دهد تا هنگام برقراری شروط مربوط به محافظ‌ها، تراکنش مورد نظر لغو[7] شود.

۴ روش پیشنهادی

کلیت مراحل روش پیشنهادی در شکل ۱ به طور ساده نمایش داده شده است. ابتدا کد قرارداد توسط برنامه‌نویس و با استفاده از ابزارهای موجود مورد تحلیل قرار می‌گیرد و نقاط ضعف شناسایی‌شده برطرف می‌شود. سپس مالک قرارداد از ساز و کاری که جزئیات بیشتر آن در ادامه تشریح می‌شود، استفاده می‌کند. این ساز و کار تعدادی محافظ امنیتی به شکلی کاربر پسند در اختیار مالک قرارداد قرار می‌دهد تا مواردی را از بین آن‌ها انتخاب، و/یا تنظیم کند. هنگام قرارگیری قرارداد روی شبکه، محافظ‌های امنیتی نیز همراه با قرارداد ارسال می‌شوند. با تغییراتی که

[1] proof of work [2] symbolic execution [3] runtime verification

[4] static analysis [5] loose coupling [6] security guard [7] revert

شکل ۱. نمایی کلی از مراحل ساز و کار

در فرایند درستی‌سنجی تراکنش‌ها ایجاد شده است، هنگامی که خللی در محافظها شناسایی شود، تراکنش مربوط لغو می‌شود. محافظهای امنیتی این اطمینان خاطر را به مالک قرارداد می‌دهند که در صورت وجود آسیب‌پذیری در قرارداد، امکان سوءاستفاده از آن از بین برود.

در این روش، خودِ آسیب‌پذیری‌ها مورد توجه قرار نمی‌گیرند؛ مسئله اصلی مورد توجه، نتیجه‌ای است که در صورت سوءاستفاده از آسیب‌پذیری قرارداد به وقوع می‌پیوندد. لذا این روش قادر است جلوی برخی حملات که ناشی از آسیب‌پذیری‌های ناشناخته هستند را بگیرد. همچنین برخی آسیب‌پذیری‌ها که تشخیص آن‌ها در روش‌های دیگر ناممکن یا دشوار است، در این روش قابل تشخیص هستند. در این روش نیاز به تحلیل کد قرارداد نمی‌باشد، زیرا از لایه‌ای بیرونی به اتفاقات مرتبط با قرارداد نگاه می‌شود.

با توجه به موارد بیان‌شده، این ساز و کار را می‌توان به عنوان یک نوآوری در حوزه امنیت قراردادهای هوشمند در نظر گرفت؛ زیرا: (۱) با شیوه‌ای جدید و با استفاده از محافظهای امنیتی، سوءاستفاده از آسیب‌پذیری‌ها تشخیص داده می‌شود؛ (۲) بلافاصله جلوی سوءاستفاده گرفته می‌شود؛ (۳) روش، محدود به آسیب‌پذیری‌های شناخته‌شده نبوده و می‌تواند حملاتی که برای اولین بار از یک آسیب‌پذیری استفاده می‌کنند (روز صفر) را نیز تشخیص دهد. (۴) استفاده از محافظهای امنیتی برای مالک قرارداد حتی بدون نیاز به دانش برنامه‌نویسی نیز ممکن است.

مهم‌ترین مسائل در طراحی این ساز و کار شامل سه مورد می‌باشد: (۱) تعیین این که محافظهای امنیتی شامل چه مواردی باشند؛ (۲) بررسی هزینه‌های جدیدی که با استفاده از محافظهای امنیتی ایجاد می‌شود؛ و (۳) آسیب‌پذیری منع خدمت محافظهای امنیتی، که این مسائل در ادامه مورد بررسی قرار می‌گیرند.

۱.۴ محافظهای امنیتی

محافظهای امنیتی به صورت شرایطی هستند که در صورت برقراری، اجرای قرارداد مشکوک به حمله و سوءاستفاده است. محافظهای بالقوه در قالب چند گزینه قابل انتخاب در اختیار مالک قرار می‌گیرد تا یک یا چند مورد از آن‌ها را انتخاب و پارامترهای آن‌ها را تنظیم کند. در شرایط حمله یا سوءاستفاده از آسیب‌پذیری، مقادیر پارامترها به نحوی تنظیم می‌شود که شرط مربوط به یک یا چند محافظ امنیتی نقض، و تراکنش لغو می‌شود.

با توجه به این‌که این ساز و کار برای قراردادهای متعددی با کاربردهای مختلف استفاده خواهد شد، محافظهای امنیتی به دو دسته تقسیم شده‌اند: (۱) محافظ عمومی (غیروابسته به قرارداد) و (۲) محافظ وابسته (به قرارداد).

محافظهای عمومی مواردی هستند که به قرارداد وابسته نیستند و برای تمامی قراردادها قابل اعمال می‌باشند. محافظهای وابسته مواردی هستند که وابسته به قرارداد و کاربرد خاص آن هستند و با توجه به قرارداد تعریف می‌شوند. در ادامه چند محافظ امنیتی و حمله مربوط به آن‌ها معرفی می‌شوند.

۱.۱.۴ محافظ امنیتی مصرف گس

این محافظ عمومی هنگامی که مصرف گس از حد آستانه مشخص‌شده عبور کند، تراکنش را لغو می‌کند. چنین تشخیصی با روش‌های دیگر و

از درون کد یا مستقل از شرایط واقعی اجرای قرارداد ناممکن است. به عنوان مثالی از کاربرد این محافظ می‌توان به رخداد حمله به قرارداد FairWin اشاره کرد که طی آن موجودی قرارداد تخلیه شد. نشانه‌ای که پژوهشگران را مشکوک کرد، مصرف بالا و غیرعادی گس در بازه یک ماهه بود [۸]. با استفاده از این محافظ عمومی، وقوع چنین مواردی غیرممکن می‌شود زیرا با مصرف غیرعادی گس، تراکنش لغو می‌شود.

۲.۱.۴ محافظ امنیتی بر روی موجودی قرارداد

به عنوان مثال دیگری از محافظ‌های عمومی می‌توان شرط روی موجودی اتر قرارداد را بررسی کرد. فرض کنید قراردادی برای کاربرد کیف پول استفاده می‌شود و در شرایط عادی مبلغی که طی بازه‌ای ۲۴ ساعته از آن برداشته می‌شود حداکثر برابر مقداری مشخص (مثلاً ۱۰۰ اتر) باشد؛ در چنین موردی در صورت استفاده از محافظ امنیتی می‌توان این محدودیت را اعمال کرد تا در صورتی که به هر دلیلی بیشتر از این مقدار برداشت شود، تراکنش لغو گردد. در صورتی که از روش پیشنهادی استفاده نشود، در جای جای کد که احتمال تغییر موجودی می‌رود باید شرایط را چک کرد که مستعد خطا و نادیده گرفته‌شدن است.

۳.۱.۴ محافظ امنیتی بر روی موجودی قرارداد

این محافظ امنیتی، نوعی محافظ وابسته است که می‌تواند تغییرات در متغیرهای قرارداد را کنترل کند. یکی از متغیرهای بسیار کلیدی در بسیاری قراردادها، متغیری است که آدرس مالک یا شرکای قرارداد در آن نگهداری می‌شود. مهاجم ابتدا با سوءاستفاده از یک آسیب‌پذیری اقدام به تغییر مالک (یا شرکا)، و پس از آن با اختیارات مالک (یا شریک) کلاه‌برداری می‌کند. وجود این محافظ امنیتی می‌تواند جلوی سوءاستفاده ذکر شده و موارد مشابه را بگیرد؛ در حالی که چنین کاری با سایر روش‌ها به سادگی ممکن نیست و ممکن است باعث افت خوانایی کد قرارداد نیز بشود.

۴.۱.۴ محافظ امنیتی توابع تو در تو

این محافظ به عنوان نوع دیگری از محافظ‌های وابسته، به منظور کنترل قابلیت فراخوانی تو در توی هر یک از توابع قرارداد در نظر گرفته شده است. بعضی قراردادها به نحوی نوشته نشده‌اند که توابع آن‌ها قابل فراخوانی تو در تو باشند؛ در این صورت با مشخص کردن چنین توابعی، محافظ مناسب می‌تواند با کنترل پشته فراخوانی جلوی سوءاستفاده از این نقطه ضعف را بگیرد.

۵.۱.۴ محافظ امنیتی ترتیب تراکنش‌ها

این محافظ وابسته در سطحی متفاوت با محافظ‌های دیگر است و یکی دیگر از نمونه کاربردهای برجسته ساز و کار پیشنهادی در مقایسه با سایر روش‌ها است. این محافظ وظیفه جلوگیری از اجرای چند تراکنش مشخص در یک بلوک را به عهده دارد. به علت عدم قطعیتی که در ترتیب تراکنش‌ها در یک بلوک وجود دارد، استخراج‌گر می‌تواند تراکنش‌ها را به نحوی در بلوک بچیند که به نفع خودش یا شخصی خاص باشد. با استفاده از این محافظ می‌توان توابعی از یک یا چند قرارداد را مشخص کرد که امکان اجرا در یک بلوک را نداشته باشند.

۲.۴ هزینه ساز و کار پیشنهادی

با فعال‌کردن محافظ‌های امنیتی، اجرای قرارداد مستلزم بررسی محافظ‌ها است که به نوبه خود هزینه محاسباتی (و لو اندک) ایجاد می‌کند. از آنجا که اصولاً هزینه‌های اجرای قراردادهای هوشمند با کارمزد (گس) پرداخته می‌شود، منطقی است که این هزینه جدید نیز لحاظ شود. در مورد این که این هزینه را چه کسی بپردازد، سه گزینه ممکن است که به شرح آن‌ها می‌پردازیم: (۱) تمامی هزینه را مالک قرارداد متحمل شود؛ (۲) تمامی هزینه‌ها را کاربر متحمل شود؛ و (۳) هزینه‌ها به گونه‌ای متناسب بین کاربر و مالک تقسیم شود.

از میان حالات فوق، حالت سوم انتخاب بهتری می‌باشد، زیرا با اعمال محافظ‌های امنیتی، هم کاربران با اطمینان خاطر از عدم سوءاستفاده از دارایی‌هایشان و هم مالک با امن‌تر بودن قراردادش سود می‌برند. این ساز و کار برای شبکه اتریوم طراحی شده است که در آن همواره کسی هزینه را می‌پردازد که آغازکننده تراکنش است. بنابراین با کمینه دستکاری در عملکرد شبکه، هزینه‌ای (گس) که بر عهده مالک است در زمان استقرار قرارداد روی شبکه، و هزینه کاربران هنگام فراخوانی قرارداد دریافت می‌شود. به این منظور هزینه دستورات مرتبط با محافظ‌های امنیتی به جدولِ صورت هزینه‌های EVM افزوده می‌شود.

ممکن است ایجاد این هزینه‌ها برای کاربران ناخوشایند باشد به همین دلیل هزینه‌های ایجادشده در کمترین مقدار ممکن قرار می‌گیرند. در این خصوص باید به چند مسئله توجه کافی شود: (۱) محافظ‌های امنیتی باید از لحاظ مصرف منابع به صورت بهینه باشند که این مسئله در بخش پیاده‌سازی مورد توجه قرار می‌گیرد. (۲) کد قرارداد بعد از برنامه‌نویسی، باید توسط ابزارهای تحلیل ایستا و پویا به طور کامل بررسی شود تا فاقد آسیب‌پذیری‌های قابل تشخیص باشد و نیاز به استفاده از محافظ امنیتی (و ایجاد هزینه) برای تشخیص چنین آسیب‌پذیری‌هایی وجود نداشته باشد. (۳) به منظور به حداقل رساندن هزینه‌ها، از محافظ‌های امنیتی تنها برای آسیب‌پذیری‌های با درجه اهمیت بالا استفاده شود؛ مانند آسیب‌پذیری‌هایی که منجر به تخلیه کامل موجودی قرارداد می‌شوند.

با توجه به این‌که در فرایند پیاده‌سازی محافظ‌های امنیتی سعی بر این است که محافظ‌ها در بهینه‌ترین حالت ممکن ایجاد شوند، بنابراین هزینه‌های ایجادشده نیز تا حد ممکن کاهش می‌یابد. به طور کلی به نظر می‌رسد مقداری هزینه‌کردن (بابت استفاده از محافظ امنیتی) نسبت به از دست دادن کل موجودی طی یک آسیب‌پذیری ناشناخته، گزینه قابل قبولی برای کاربران باشد. علاوه بر مواردی که بیان شد، لازم به ذکر است در صورتی که در آینده این ساز و کار برای سایر بسترها نیز طراحی شود، ممکن است بتوان از سایر گزینه‌های موجود برای پوشش هزینه‌ها نیز استفاده کرد.

۳.۴ آسیب‌پذیری منع خدمت محافظ‌های امنیتی

با تغییرات ایجادشده در استخراج‌گر، امکان لغو تراکنش‌های مشکوک توسط ساز و کار وجود دارد. ممکن است سوءاستفاده از این امر به منع خدمت منجر شود. در این خصوص سه حالت را بررسی می‌کنیم: (۱) در واقع هیچ‌گونه سوءاستفاده‌ای از قرارداد رخ نداده باشد و تراکنش‌های عادی به اشتباه لغو شوند. به منظور پیشگیری از چنین اتفاقی، از یک سو تعیین محافظ‌های امنیتی توسط مالک، و از سوی دیگر پیاده‌سازی ساز و کار، باید با دقت کافی انجام شوند تا حالات واقعی سوءاستفاده از عملکرد عادی متمایز شود و امکان خطا به حداقل برسد. (۲) کاربر بدنیت شرایطی ایجاد کند که تراکنش خاصی با موفقیت اجرا نشود و بخواهد از ویژگی لغو تراکنش‌های مشکوک به نفع خودش استفاده کند. با توجه به این که امکان دخالت مستقیم در اجرای تراکنش سایر کاربران وجود ندارد، کاربر بدنیت فقط می‌تواند کاربران را ترغیب به اجرای تراکنشی کند که لغو خواهد شد. در این صورت محافظ امنیتی کار خود را به درستی انجام داده و جلوی مهاجم را گرفته است. (۳) اجرای تراکنش‌ها تحت تأثیر سربار محافظ‌ها قرار گیرد. چنان‌که پیش‌تر ذکر شد، هزینه اجرای محافظ‌ها با گس پرداخته می‌شود و مقدار گس قابل مصرف برای اجرای تراکنش‌های هر بلوک در اتریوم سقف مشخصی دارد (در حال حاضر ۱۰ میلیون). لذا افزودن محافظ‌های امنیتی موجب امکان‌پذیر شدن حمله منع خدمت بر روی استخراج‌گرها نمی‌شود. اما مقداری از گس مصرفی هر تراکنش را به خود اختصاص می‌دهد. بنابراین باید در پیاده‌سازی محافظ‌های امنیتی، پیچیدگی محاسباتی محافظ‌ها کم باشد تا عملیات با حداقل سربار انجام شود. همچنین محافظ‌های امنیتی باید به صورت حداقلی بر روی تراکنش‌هایی گذاشته شوند که گس بسیار زیادی مصرف می‌کنند تا موجب محدودیت بیشتری برای آن‌ها نشوند. راهکار دیگر شکستن چنین تراکنش‌هایی به تراکنش‌های کم‌مصرف‌تر است.

۵ نحوه پیاده‌سازی محافظ‌های امنیتی

در این مقاله برای پیاده‌سازی محافظ‌های امنیتی از یکی از کلاینت‌های اتریوم به نام Hyperledger Besu استفاده شده است. این کلاینت که پروتکل اتریوم را پیاده‌سازی می‌کند، به زبان Java نوشته شده است. بخشی از کد Besu وظیفه اعتبارسنجی تراکنش‌ها را بر عهده دارد. در این کد تعدادی شرط وجود دارد که نقض هر یک منجر به نامعتبر شناخته‌شدن و لغو تراکنش می‌شود.

به منظور پیاده‌سازی محافظ‌های امنیتی شروط جدیدی به کد اضافه، و هنگام اعتبارسنجی تراکنش‌ها بررسی می‌شوند. به عنوان مثال، یک پیاده‌سازی ساده از محافظ امنیتی که ارسال اتر را برای برخی از مقادیر محدود می‌کند به صورت قطعه کد شکل ۲ به کد اعتبارسنجی تراکنش‌ها اضافه می‌شود.

در قطعه کد شکل ۲، هر گاه مقدار اتر ارسالی در تراکنش برابر etherValue باشد، تراکنش در مرحله اعتبارسنجی، نامعتبر شناخته شده

```
Wei value = Wei.of(BigInteger.valueOf(etherValue)
        .multiply(BigInteger.TEN.pow(18)));
if (transaction.getValue().equals(value)) {
    return ValidationResult.invalid(
    TransactionInvalidReason.TX_VALUE_NOT_ALLOWED,
    String.format("Sending some amounts of Ether is not allowed.”));
    }
```

شکل ۲. قطعه کد پیاده‌سازی یک محافظ امنیتی در Besu (Java)

و لغو می‌شود. مقدار etherValue نیز توسط مالک قرارداد تعیین می‌شود. به طرز مشابه و برای آسیب‌پذیری‌های مختلف شروط دیگری به کلاینت اضافه می‌شود تا تمامی تراکنش‌های ناشی از وقوع آسیب‌پذیری لغو شوند.

۶ ارزیابی

در این بخش به ارزیابی ساز و کار معرفی‌شده و مقایسه آن با سایر روش‌ها می‌پردازیم. به این منظور ساز و کار پیشنهادی با چهار مورد از ابزارهای مهم موجود مقایسه می‌شود تا نقاط قوت و ضعف این ساز و کار بهتر مشخص شود. در جدول ۱ برخی ویژگی‌های هر ابزار و همچنین ساز و کار مشاهده می‌شود. بر اساس این جدول، یکی از نکات مهم و قابل توجه، عدم نیاز ساز و کار پیشنهادی به در دسترس بودن کد قرارداد هوشمند است که در سایر روش‌ها و ابزارها این‌گونه نیست. لذا استفاده از این ساز و کار محدود به زبان برنامه‌نویسی قرارداد هوشمند نیز نمی‌باشد. نکته مهم دیگر این است که سایر ابزارها تنها قبل از قرارگیری کد قرارداد روی شبکه قابل استفاده هستند در حالی که در ساز و کار پیشنهادی محافظ‌های امنیتی در هنگام اجرا، از قرارداد محافظت می‌کنند و حملات را لغو می‌نمایند.

همان‌طور که بیان شد، در ساز و کار پیشنهادی، قرارداد تغییری نمی‌کند؛ بلکه استخراج‌گر به نحوی تغییر یافته است که بتواند محافظ‌های امنیتی را هنگام اجرای قراردادها و تراکنش‌ها لحاظ کند. عدم تغییر قرارداد جنبه مثبت، و نیاز به اصلاح استخراج‌گر جنبه منفی این امر است. همچنین افزایش هزینه (گس) استقرار و اجرای قرارداد، نقطه‌ضعف این ساز و کار است. البته این مسئله در ابزارهایی مانند [۵] که کدی را به قرارداد اضافه می‌کنند هم وجود دارد.

از لحاظ توانایی جلوگیری از سوءاستفاده، فقط ساز و کار پیشنهادی توانایی جلوگیری از سوءاستفاده‌هایی را دارد که در بخش قبل معرفی شدند. به طور خاص، جایگاه ساز و کار پیشنهادی در مقایسه با سایر روش‌ها و ابزارها در جدول ۲ مشخص شده است (تنها کد سوءاستفاده ذکر شده است.)

همان‌گونه که در جدول ۲ مشاهده می‌شود، هر گاه (در زمان اجرا)، سوءاستفاده‌ای رخ دهد، ساز و کار پیشنهادی می‌تواند آن را تشخیص دهد و مانع آن شود. سایر ابزارها نیز می‌توانند با تحلیل کد قرارداد، برخی موارد را تشخیص دهند ولی از میان ابزارها، هیچ یک قادر به تشخیص تمامی موارد نیست.

آسیب‌پذیری‌های دیگری نیز برای قراردادهای هوشمند اتریوم وجود

جدول ۱. مقایسه برخی ویژگی‌های سایر روش‌ها و ابزارها با ساز و کار پیشنهادی

ابزار / ویژگی	عدم نیاز به کد قرارداد	محدود نبودن به زبان برنامه‌نویسی خاص	عدم تغییر در استخراج‌گر	عدم ایجاد هزینه برای کاربر	زمان استفاده	روش تشخیص
ساز و کار پیشنهادی	✔	✔	✘	✘	قبل از ثبت و هنگام اجرای قرارداد	تعیین محافظ‌های امنیتی برای لغو تراکنش‌های مشکوک
Oyente [۲]	✘	✔	✔	✔	قبل از ثبت قرارداد	اجرای نمادین
Securify [۳]	✘	✔	✔	✔	قبل از ثبت قرارداد	درستی‌سنجی صوری
Gasper [۴]	✘	✘ (Solidity)	✔	✔	قبل از ثبت قرارداد	اجرای نمادین
Slither [۶]	✘	✘ (Solidity)	✔	✔	قبل از ثبت قرارداد	تحلیل ایستا

جدول ۲. مقایسه توانایی جلوگیری از سوءاستفاده

ابزار / کد سوءاستفاده	(۱)	(۲)	(۳)	(۴)	(۵)
ساز و کار پیشنهادی	✔	✔	✔	✔	✔
Oyente [۲]	✘	✔	✘	✔	✔
Securify [۳]	✘	✘	✘	✔	✔
Gasper [۴]	*✘	✘	✘	✘	✘
Slither [۶]	✘	✘	✘	✔	✘

*لازم به ذکر است Gasper تنها قادر است با تحلیل کد قرارداد، موارد مربوط به برنامه‌نویسی قرارداد که منجر به مصرف غیرضروری گس می‌شوند را تشخیص دهد.

دارند؛ اما در مرحله تحلیل کد قرارداد (توسط برنامه‌نویس) شناسایی می‌شوند. لذا در مقام مقایسه موضوعیت ندارند و در جدول ۲ نیامده‌اند.

۷ جمع‌بندی و نتیجه‌گیری

امروزه با توجه به رشد سریع قراردادهای هوشمند اتریوم، امنیت آن‌ها به مسئله‌ای مهم تبدیل شده است. در همین راستا، ساز و کاری به منظور نظارت مالک روی قرارداد هوشمند در این مقاله معرفی شد. با استفاده از این ساز و کار، مالک قرارداد بدون نیاز به دانش برنامه‌نویسی، تعدادی محافظ امنیتی برای قرارداد تعیین می‌کند و پس از آن، ساز و کار با توجه به محافظ‌های تعیین‌شده، در صورت تشخیص وقوع حمله به قرارداد، تراکنش را لغو می‌کند و به این ترتیب دارایی‌ها محفوظ می‌مانند.

در پژوهش‌های آینده و در ادامه‌ی این پژوهش، می‌توان دامنه آسیب‌پذیری‌های قابل تشخیص توسط ساز و کار را افزایش داد و همچنین چنین ساز و کاری را برای بسترهای دیگری به جز اتریوم که از قراردادهای هوشمند پشتیبانی می‌کنند، ارائه داد.

مراجع

[1] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. A survey of attacks on ethereum smart contracts (sok). In *International conference on principles of security and trust*, pages 164–186. Springer, 2017.

[2] Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor. Making smart contracts smarter. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 254–269, 2016.

[3] Petar Tsankov, Andrei Dan, Dana Drachsler-Cohen, Arthur Gervais, Florian Buenzli, and Martin Vechev. Securify: Practical security analysis of smart contracts. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 67–82, 2018.

[4] Ting Chen, Xiaoqi Li, Xiapu Luo, and Xiaosong Zhang. Under-optimized smart contracts devour your money. In *2017 IEEE 24th international conference on software analysis, evolution and reengineering (SANER)*, pages 442–446. IEEE, 2017.

[5] Joshua Ellul and Gordon J Pace. Runtime verification of ethereum smart contracts. In *2018 14th European Dependable Computing Conference (EDCC)*, pages 158–163. IEEE, 2018.

[6] Josselin Feist, Gustavo Grieco, and Alex Groce. Slither: a static analysis framework for smart contracts. In *2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*, pages 8–15. IEEE, 2019.

[7] A postmortem on the parity multi-sig library self-destruct. Available: https://www.parity.io/a-postmortem-on-the-parity-multi-sig-library-self-destruct/, 2017. [Accessed:January 20, 2020].

[8] The collapse of fairwin's $125m ponzi scheme. Available: https://medium.com/@PhABC/the-collapse-of-fairwins-125m-ponzi-scheme-61a66b273420, 2019. [Accessed:January 20, 2020].