Detección de vulnerabilidades en especificaciones de contratos inteligentes de la plataforma Ethereum

Mauro C. Argañaraz(1), Mario M. Berón(1), Pedro Rangel Henriques(2) &Daniel Riesco(1)
(1)Departamento de Informática - Facultad de Ciencias Física Matemáticas y Naturales (FCFMyN) – Universidad Nacional de San Luis
(2)Universidade do Minho - Braga, Portugal
marganaraz@gmail.com(1), {mberon, driesco}@unsl.edu.ar(1), pedrorangelhenriques@gmail.com(2)

RESUMEN

Ethereum es el principal ecosistema basado en blockchain que proporciona un entorno para codificar y ejecutar contratos inteligentes, propiedad que en estos últimos años recibió gran atención por las nuevas aplicaciones comerciales y, también, por la comunidad científica. El proceso de escritura de contratos seguros y de buen desempeño en la plataforma Ethereum es un gran desafío para los desarrolladores. Implica la aplicación de paradigmas de programación no convencionales debido a las características inherentes de la ejecución de programas de computación distribuida. Además, los errores en los contratos desplegados pueden tener graves consecuencias debido al acoplamiento inmediato del código del contrato y las transacciones financieras. El manejo directo de los activos significa que las fallas tienen más probabilidades de ser relevantes para la seguridad y tienen mayores consecuencias económicas que los errores en las aplicaciones típicas. Una serie de reportes de incidentes muestran que los problemas de seguridad se han utilizado con propósitos fraudulentos.

En este artículo, se describe una línea de investigación que se enfoca en fortalecer los aspectos de seguridad, basado en una base sólida de diseño, patrones de código establecidos y probados que faciliten el proceso de escritura de código funcional y libre de errores, para proporcionar una herramienta que permita analizar y detectar falencias de manera automática.

Palabras Claves: blockchain, Ethereum, smart contract, Solidity, Vyper, static analysis tool, verification, security patterns.

CONTEXTO

La presente línea de investigación se enmarca en el Proyecto (PO/16/93) de *"Fortalecimiento de la Seguridad de los Sistemas de Software mediante el uso de Métodos, Técnicas y Herramientas de Ingeniería Reversa"*. Realizado en conjunto con la Universidade do Minho Braga, Portugal. Recientemente aprobado por el Ministerio de Ciencia Tecnología e Innovación Productiva (Mincyt). Y por el Proyecto (P031516.) de Investigación: *"Ingeniería de Software: Conceptos, Prácticas y Herramientas para el Desarrollo de Software con Calidad"*. De la Facultad de Ciencias Físico Matemáticas y Naturales, Universidad Nacional de San Luis. Dicho proyecto es la continuación de diferentes proyectos de investigación a través de los cuales se ha logrado un importante vínculo con distintas universidades a nivel nacional e internacional. Además, se encuentra reconocido por el Programa de Incentivos.

1. INTRODUCCIÓN

El año 2017 se caracterizó por la popularidad mundial que alcanzó Bitcoin [1] llegando a su valor de mercado más alto en su breve recorrido desde su creación, hito que coincidió con el reconocimiento de Wall Street al incluirlo en los mercados de futuros. Ante la merma en el precio durante 2018, la cadena de bloques (blockchain), tecnología que subyace a Bitcoin y al resto de las criptomonedas, llegó a la cresta de la ola IT. Sin embargo, de ahora en más, el foco del mercado y de la comunidad

científica inevitablemente se concentrará en los contratos inteligentes, ya que permiten que partes no confiables manifiesten términos de contrato en el código del programa y, por lo tanto, eliminan la necesidad de un tercero confiable.

Blockchain es una tecnología que se basa en una combinación de criptografía, redes y mecanismos de incentivos para respaldar la verificación, ejecución y registro de transacciones entre diferentes partes. En términos simples, las plataformas de blockchain se pueden ver como bases de datos descentralizadas que ofrecen propiedades muy atractivas. Esto incluye la inmutabilidad de las transacciones almacenadas y la creación de confianza entre los participantes sin un tercero. Esto hace que esta arquitectura sea adecuada como un libro mayor abierto y distribuido que pueda almacenar transacciones entre partes de manera verificable y permanente.

Una aplicación destacada es el intercambio de activos digitales, conocidos como criptomonedas. Las criptomonedas más conocidas y transadas son Bitcoin, Ethereum y Litecoin. Ofrecen, más allá de la transferencia de activos digitales, la ejecución de contratos inteligentes.

Los contratos inteligentes son esencialmente dos conceptos combinados. Una es la noción de software. Código frío y austero que hace lo que está escrito y se ejecuta para que el mundo lo vea. La otra es la idea de un acuerdo entre las partes. Son programas informáticos que facilitan, verifican y hacen cumplir la negociación y ejecución de contratos legales. Se ejecutan a través de transacciones de blockchain, interactúan con las criptomonedas y tienen interfaces para manejar la información de los participantes del contrato. Cuando se ejecuta en la cadena de bloques, un contrato inteligente se convierte en una entidad autónoma que ejecuta automáticamente acciones específicas cuando se cumplen ciertas condiciones.

Ethereum es la principal plataforma de computación distribuida pública basada en blockchain que proporciona un entorno de ejecución de contratos inteligentes dentro del contexto de una máquina virtual descentralizada, conocida como Ethereum Virtual Machine (EVM) [2, 3].

La máquina virtual EVM maneja el cómputo y el estado de los contratos y se basa en un lenguaje basado en una estructura de pila con un conjunto predefinido de instrucciones (códigos de operación) [3]. En esencia, un contrato es simplemente una serie de declaraciones de código de operación, que son ejecutadas secuencialmente por la máquina virtual EVM. La máquina EVM puede considerarse como una computadora global descentralizada en la que se ejecutan todos los contratos inteligentes. Aunque se comporta como una computadora gigante, es más bien una red de máquinas discretas más pequeñas en comunicación constante.

Los contratos inteligentes en Ethereum generalmente se escriben en lenguajes de alto nivel y luego se compilan en bytecode EVM. Los lenguajes de alto nivel son LLL (lenguaje tipo Lisp de bajo nivel), Serpent (un lenguaje de estilo Python), Vyper (un lenguaje de tipo Python), Bamboo (lenguaje basado en estados) y Solidity (un lenguaje estilo Javascript) [4]. LLL y Serpent se desarrollaron en las primeras etapas de la plataforma, mientras que Vyper y Bamboo se encuentran actualmente en desarrollo por la fundación Ethereum. El lenguaje más prominente y ampliamente adoptado es Solidity.

Solidity es un lenguaje de programación de alto nivel orientado a contratos. Su sintaxis es similar a JavaScript, está tipado de manera estática y admite herencia y polimorfismo, así como bibliotecas y tipos complejos definidos por el usuario.

El proceso de escritura de contratos seguros y de buen desempeño en Ethereum

es una tarea difícil para los desarrolladores. Implica la aplicación de paradigmas de programación no convencionales, debido a las características inherentes de la ejecución de programas basados en blockchain. Además, los errores en los contratos desplegados pueden tener graves consecuencias, debido al acoplamiento inmediato del código del contrato y las transacciones financieras. Por lo tanto, es beneficioso tener una base sólida de diseño, patrones de código establecidos y probados que faciliten el proceso de escritura de código funcional y libre de errores, y herramientas que permitan analizar y detectar falencias de manera automática.

Un análisis de los contratos inteligentes existentes realizado por Bartoletti y Pompianu [5] muestra que las plataformas Bitcoin y Ethereum se centran principalmente en los contratos financieros. En otras palabras, la mayor parte del código del programa define cómo se mueven los activos (dinero). Por lo tanto, es crucial que la ejecución del contrato se realice correctamente. El manejo directo de los activos significa que las fallas tienen más probabilidades de ser relevantes para la seguridad y tienen mayores consecuencias financieras que los errores en las aplicaciones típicas.

Los incidentes, como el desbordamiento de valor en Bitcoin, o el ataque al proyecto DAO en Ethereum, causaron que una bifurcación dura de la blockchain anulara las transacciones maliciosas. Estos incidentes muestran que los problemas de seguridad se han utilizado con propósitos fraudulentos. Muchas de estas vulnerabilidades se pueden clasificar en tres grupos: lenguaje de programación, máquina virtual (EVM) y las peculiaridades propias de la blockchain, y pueden abordarse siguiendo las mejores prácticas para escribir contratos inteligentes seguros, que se encuentran dispersos en toda la comunidad Ethereum.

2. LÍNEAS DE INVESTIGACIÓN Y DESARROLLO

En la búsqueda de asegurar los contratos inteligentes en la plataforma Ethereum se han adoptado diferentes enfoques, cubriendo aspectos como la semántica formal, patrones de seguridad y herramientas de verificación. De acuerdo con el tipo de enfoque abordado, se distingue entre enfoques de verificación y de diseño.

El objetivo de los enfoques de verificación es comprobar que los contratos inteligentes escritos en lenguajes de alto nivel existentes (como Solidity por mencionar alguno) cumplan con una política o especificación de seguridad, entre los que se pueden mencionar:

- Herramientas de análisis estático para la búsqueda automática de bugs [6, 7, 8, 9]
- Herramientas de análisis estático para la verificación automática de propiedades genéricas [10, 11, 12, 13]
- Frameworks para pruebas semi-automatizadas de propiedades específicas del contrato [14, 15, 16, 17]
- Monitoreo dinámico de propiedades de seguridad predefinidas [18, 19]

En contraste, los enfoques de diseño apuntan a facilitar la creación de contratos inteligentes seguros al proporcionar frameworks para su desarrollo: abarcan nuevos lenguajes que son más susceptibles de verificación, proporcionan una semántica clara y sencilla que es entendible por los desarrolladores de contratos inteligentes o que permiten una codificación directa de las políticas de seguridad deseadas. Además, se incluyen trabajos que tienen como objetivo proporcionar patrones de diseño para contratos inteligentes seguros.

- Lenguajes de alto nivel [20, 21, 22, 23]
- Lenguajes intermedios [24]
- Patrones de seguridad [25]

- Herramientas [26]
- Ontologías [27, 28]

La línea de investigación presentada en este artículo busca plantear una estrategia para la detección de vulnerabilidades, nutriéndose de ambos campos de estudio. Por un lado, utiliza un enfoque de verificación con las siguientes características:

- Lenguaje destino: lenguajes de alto nivel de la plataforma Ethereum (se selecciona Solidity como caso de estudio de mercado y Vyper como caso de estudio experimental)
- Método de análisis: estático
- Garantías: búsqueda de bugs
- Grado de automatización: verificación automatizada

Por el lado de los enfoques de diseño, se toman como base los patrones de seguridad para lenguajes existentes [25] y la idea de construir una representación intermedia para analizar los aspectos de seguridad.

3. RESULTADOS OBTENIDOS/ESPERADOS

El trabajo de investigación apunta a responder qué especificaciones tienen vulnerabilidades a través de la ejecución de análisis léxicos y sintácticos. El análisis estático garantiza una cobertura completa sin ejecutar el programa y lo suficientemente rápido en un código de tamaño razonable.

Para llevar adelante la propuesta, se creó el proyecto open source denominado OpenBalthazar. Este proyecto consiste en una herramienta web de análisis estático para los contratos inteligentes de la plataforma Ethereum implementada con Microsoft .NET Core.

Actualmente está implementado Solidity y se inició el desarrollo de las reglas de Vyper, si bien es una herramienta extensible y se pueden incorporar nuevos lenguajes como Bamboo a través de los mecanismos previstos.

En el proyecto se utiliza la librería ANTLR 4 y las gramáticas de Solidity y Vyper para generar el árbol de análisis (AST). Este árbol se puede enriquecer con información adicional utilizando algoritmos y técnicas de procesamiento de lenguajes. Las reglas de verificación construidas utilizan un repositorio de patrones que definen los criterios en términos del árbol.

Los investigadores de esta línea continuarán con estudios en este campo con el objetivo de perfeccionarse en el área y realizar un seguimiento de nuevas amenazas, vulnerabilidades y ciberataques en materia de despliegue y ejecución de contratos inteligentes. También se planea seguir con los siguientes trabajos futuros:

- Generalización para otras plataformas de blockchain que soporten contratos inteligentes, tales como NEM, NEO, Cardano o Hyperledger.
- Obtención de código fuente a partir de bytecode EVM para aplicar el análisis que se propone en esta línea de investigación.
- Análisis dinámico de código fuente, lo cual implica correr el contrato y considerar sólo un conjunto de todas las ejecuciones posibles en base a datos de entrada arbitrarios.
- Análisis de los aspectos de seguridad que surjan de la interoperabilidad con otras plataformas blockchain, también conocida como cross-chain smart contracts.

4. FORMACIÓN DE RECURSOS HUMANOS

El equipo de profesionales de la UNSL que forman parte de la línea de investigación de este trabajo llevan adelante diferentes trabajos finales integradores de *Ingeniería en Informática, Ingeniería en Computación, Licenciatura en Ciencias de la Computación*, y en un futuro próximo trabajos finales de especialización, tesis de

maestría y doctorado. En particular, las investigaciones desarrolladas en este trabajo forman parte del lineamiento inicial como trabajo final de uno de los autores para optar al grado de Doctor en Ingeniería Informática en la UNSL.

5. BIBLIOGRAFÍA

[1] Nakamoto, S., (2008), Bitcoin: Un Sistema de Efectivo Electrónico Usuario-a-Usuario, http://bitcoin.org/bitcoin.pdf

[2] Ethereum White Paper: https://github.com/ethereum/wiki/wiki/White-Paper

[3] Ethereum Yellow Paper: https://ethereum.github.io/yellowpaper/paper.pdf

[4] Solidity — solidity 0.4.18 documentation. [Online]. Available: https://media.readthedocs.org/pdf/solidity/develop/solidity.pdf

[5] Bartoletti, M. & Pompianu, L. (2017) *An empirical analysis of smart contracts: platforms, applications, and design patterns*. arXiv preprint arXiv:1703.06322.

[6] Luu, L., Chu, D.H., Olickel, H., Saxena, P., Hobor, A.: Making smart contracts smarter. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, ACM (2016) 254–269

[7] Grishchenko, I., Maffei, M., Schneidewind, C.: A semantic framework for the security analysis of ethereum smart contracts. In: Proceedings of the 7th International Conference on Principles of Security and Trust (POST), Springer (2018)

[8] Zhou, E., Hua, S., Pi, B., Sun, J., Nomura, Y., Yamashita, K., Kurihara, H.: Security assurance for smart contract. In: New Technologies, Mobility and Security (NTMS), 2018 9th IFIP International Conference on, IEEE (2018) 1–5

[9] Nikolic, I., Kolluri, A., Sergey, I., Saxena, P., Hobor, A.: Finding the greedy, prodigal, and suicidal contracts at scale. arXiv preprint arXiv:1802.06038 (2018)

[10] Kalra, S., Goel, S., Dhawan, M., Sharma, S.: Zeus: Analyzing safety of smart contracts, NDSS (2018)

[11] Buenzli, F., Dan, A., Drachsler-Cohen, D., Gervais, A., Tsankov, P., Vechev, M.: Securify (2017) Available at http://securify.ch.

[12] Mythril Available at https://github.com/ConsenSys/mythril.

[13] Manticore Available at https://github.com/trailofbits/manticore.

[14] Hirai, Y.: Defining the ethereum virtual machine for interactive theorem provers. In: International Conference on Financial Cryptography and Data Security, Springer (2017) 520–535

[15] Amani, S., B´egel, M., Bortin, M., Staples, M.: Towards verifying ethereum smart contract bytecode in isabelle/hol. CPP. ACM. To appear (2018)

[16] Hildenbrandt, E., Saxena, M., Zhu, X., Rodrigues, N., Daian, P., Guth, D., Rosu, G.: Kevm: A complete semantics of the ethereum virtual machine. Technical report (2017)

[17] Bhargavan, K., Delignat-Lavaud, A., Fournet, C., Gollamudi, A., Gonthier, G., Kobeissi, N., Kulatova, N., Rastogi, A., Sibut-Pinote, T., Swamy, N., et al.: Formal verification of smart contracts: Short paper. In: Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security, ACM (2016) 91–96

[18] Grossman, S., Abraham, I., Golan-Gueta, G., Michalevsky, Y., Rinetzky, N., Sagiv, M., Zohar, Y.: Online detection of effectively callback free objects with applications to smart contracts. Proceedings of the ACM on Programming Languages 2(POPL) (2017) 48

[19] Cook, T., Latham, A., Lee, J.H.: Dappguard: Active monitoring and defense for solidity smart contracts

[20] OConnor, R.: Simplicity: A new language for blockchains. arXiv preprint arXiv:1711.03028 (2017)

[21] Pettersson, J., Edström, R.: Safer smart contracts through type-driven development

[22] Coblenz, M.: Obsidian: A safer blockchain programming language. In: Software Engineering Companion (ICSE-C), 2017 IEEE/ACM 39th International Conference on, IEEE (2017) 97–99

[23] Schrans, F., Eisenbach, S., Drossopoulou, S.: Writing safe smart contracts in flint

[24] Sergey, I., Kumar, A., Hobor, A.: Scilla: a smart contract intermediate-level language. arXiv preprint arXiv:1801.00687 (2018)

[25] W¨ohrer, M., Zdun, U.: Smart contracts: Security patterns in the ethereum ecosystem and solidity. (2018)

[26] Mavridou, A., Laszka, A.: Designing secure ethereum smart contracts: A finite state machine based approach. arXiv preprint arXiv:1711.09327 (2017)

[27] Ugarte, H.: BLONDiE https://github.com/hedugaro/Blondie

[28] Sitio oficial de la ontología EthOn: https://github.com/ConsenSys/EthOn