HACONTI
Una plataforma para aprender sobre seguridad en Smart Contracts

Juan Schällibaum, Paula Venosa, Nicolás Macia

LINTI (Laboratorio de Investigación en Nuevas Tecnologías Informáticas),
Facultad de Informática - UNLP
{pvenosa
nmacia}@info.unlp.edu.ar
juanschallibaum@gmail.com

Resumen La creciente adopción de sistemas basados en la blockchain ha impulsado la necesidad de abordar las posibles vulnerabilidades en Smart Contracts. Eventos pasados de pérdidas millonarias subrayan la importancia de comprender y mitigar riesgos en estas implementaciones. Este artículo presenta HACONTI, una plataforma web con desafíos de ciberseguridad basados en Smart Contracts, donde cada desafío requiera explotar una vulnerabilidad para resolverlo. La finalidad de nuestra aplicación es ayudar a capacitar a las personas tanto para desarrollar Smart Contracts de manera segura, como así también para llevar a cabo pentests sobre los mismos, contribuyendo así a la ciberseguridad. Este desarrollo fue realizado en el marco de la tesina de Licenciatura en Informática de Juan Schallibaum, dirigida por los docentes Nicolás Macia y Paula Venosa

Keywords: ciberseguridad, Smart Contracts, solidity, desafíos, enseñanza, CTF

1. Introducción

La seguridad en el desarrollo de software es un componente crucial en el ámbito de la ciberseguridad. En un mundo cada vez más digitalizado, la cantidad de amenazas y vulnerabilidades crece exponencialmente. Implementar prácticas de seguridad desde las etapas iniciales del desarrollo es vital para prevenir posibles ataques y proteger la integridad de los sistemas. Integrar la seguridad en todo el ciclo de vida del desarrollo, se ha convertido en una necesidad imperiosa para asegurar que las aplicaciones sean resistentes a ataques y cumplan con los estándares de seguridad requeridos.

Capacitar a los desarrolladores y otros profesionales involucrados en el desarrollo de software sobre las mejores prácticas de seguridad es esencial. La educación continua en ciberseguridad permite a los equipos mantenerse actualizados frente a nuevas amenazas y técnicas de ataque. Esto no solo mejora la calidad

del código producido, sino que también fortalece la postura de seguridad de la organización en su conjunto.

En este contexto, la seguridad de los contratos inteligentes (smart contracts) se presenta como un área de especial interés. La creciente adopción de sistemas basados en la blockchain en los últimos años ha impulsado la necesidad de abordar de manera proactiva las posibles vulnerabilidades asociadas con el desarrollo de smart contracts en lenguajes como Solidity. En este contexto, la seguridad en el ámbito de los smart contracts se ha convertido en un desafío crucial. Los eventos pasados en los que se produjeron pérdidas millonarias [1][2][3] debido a contratos vulnerables subrayan la importancia de comprender y mitigar los riesgos asociados con estas implementaciones.

La tecnología blockchain, con sus características de criptografía y descentralización, proporciona indudables beneficios en cuanto a la seguridad de las transacciones digitales. Sin embargo, los contratos inteligentes desplegados en la blockchain son programas informáticos y, como tales, están sujetos a vulnerabilidades tanto en su diseño como en su implementación.

Los contratos inteligentes, al ser ejecutados en la blockchain, alojan grandes cantidades de valor en forma de criptomonedas, como Ether. Esto los convierte en objetivos atractivos para los atacantes, ya que el éxito de un ataque puede resultar en la pérdida de una cantidad significativa de activos digitales. Además, la naturaleza seudónima de las transacciones en la blockchain permite que los atacantes operen de forma anónima, dificultando la identificación y persecución de los responsables de los ataques. Cualquier vulnerabilidad en estos contratos puede resultar en la pérdida de una cantidad significativa de activos digitales.

Por otro lado, la característica de inmutabilidad de los contratos inteligentes, que impide su modificación una vez desplegados en la blockchain, aumenta la criticidad de las vulnerabilidades. A pesar de esta realidad, se han desarrollado técnicas para permitir la actualización de contratos inteligentes, pero estas técnicas, a su vez, pueden introducir nuevas vulnerabilidades.

2. Seguridad en Smart Contracts

Existen numerosas vulnerabilidades que afectan a los contratos inteligentes. A continuación, se presentan algunas de las más importantes que se abordan en los desafíos de la plataforma:

- **Reentrancy:** La vulnerabilidad de reentrancy permite a un atacante ejecutar repetidamente una función dentro de un contrato inteligente antes de que la ejecución previa haya finalizado. Esto puede causar que las diferentes invocaciones de la función interactúen de maneras no deseadas, permitiendo potencialmente que el atacante tome el control del flujo del programa y manipule su comportamiento [4].
- **Desbordamiento de enteros:** Un desbordamiento/subdesbordamiento ocurre cuando una operación aritmética excede el tamaño máximo o mínimo de un tipo de dato. Por ejemplo, en uint8, el rango es de 0 a 255. Cuando

una operación excede este rango, se produce un desbordamiento de enteros, creando un valor fuera del rango representable. Esto puede llevar a comportamientos inesperados y vulnerabilidades de seguridad [5].

- **Denegación de servicio:** Los ataques de Denegación de Servicio (DoS) tienen como objetivo inutilizar un contrato inteligente, agotando su gas, provocando reversiones o causando otros errores que impidan su correcto funcionamiento. El DoS puede manifestarse de diversas formas, como bloqueando el acceso a funciones críticas o impidiendo que otros usuarios interactúen con el contrato [6].
- **Fallas en controles de acceso:** Las fallas en los controles de acceso permiten que usuarios no autorizados ejecuten funciones restringidas o accedan a datos sensibles de un contrato inteligente. Estas vulnerabilidades pueden surgir debido a una configuración incorrecta de permisos o a la ausencia de mecanismos de autenticación adecuados. Como resultado, los atacantes pueden modificar el estado del contrato, robar fondos o realizar acciones no autorizadas, comprometiendo la integridad y seguridad del sistema [7].
- **Inclusión de información privada dentro de la Blockchain:** Hay que tener en cuenta que el almacenamiento de los contratos inteligentes es público. Incluso las variables de tipo privadas pueden ser leídas desde un explorador de bloques. Por lo tanto, incluir información privada sin cifrar dentro del almacenamiento de un contrato representa una vulnerabilidad de alto riesgo, ya que compromete la confidencialidad de la información [8].
- **Uso de fuentes débiles de aleatoriedad:** El uso de fuentes débiles de aleatoriedad en smart contracts representa un riesgo considerable. En Ethereum, generar números aleatorios confiables es extremadamente difícil. El uso de block.timestamp y otros datos controlados por mineros puede ser manipulado, permitiendo que se prevean resultados en juegos de azar o se comprometa la equidad en contratos importantes [9].
- **Ataques de phishing descentralizados**: Si un contrato utiliza la propiedad *tx.origin* en lugar de *msg.sender* para tomar decisiones importantes en su lógica, puede crear una vulnerabilidad. Esto podría permitir que un atacante, a través de un contrato inteligente intermedio, interactúe con el contrato vulnerable en nombre de una víctima. Para que la explotación se pueda dar, el atacante debería convencer mediante phishing a la víctima para que llame a una función de su contrato intermedio [10].
- **Envío forzado de ether:** Cuando en la lógica de un contrato se toman desiciones en base a su saldo, asumiendo que el mismo tiene un valor específico, que no podría cambiar, se crea una vulnerabilidad de seguridad. Por más que el contrato no incluya funciones *payable*, siempre existe la posibilidad de forzar el envío de ether a un contrato mediante la función *selfdestruct()*, interfiriendo de un modo inesperado en el flujo del contrato [11].
- **Llamadas inseguras a contratos externos:** La función *delegatecall()* es una variante de llamada a mensajes que permite ejecutar código de una dirección diferente en el contexto del contrato que realiza la llamada, sin cambiar *msg.sender* ni *msg.value*. Aunque esto facilita la carga dinámica de código durante la ejecución, el contrato llamado puede modificar el alma-

cenamiento y controlar el saldo del contrato llamador. Por lo tanto, usar delegatecall() con contratos no confiables es altamente riesgoso debido al potencial de manipulación de datos y saldo del contrato llamador [12].

3. HACONTI (propuesta)

HACONTI es una plataforma web que brinda desafíos de ciberseguridad basados en Smart Contracts desarrollados en Solidity. Cada nivel o desafío de la plataforma consta de un contrato inteligente con alguna vulnerabilidad que debe ser explotada para su resolución.

La plataforma cuenta con un total de treinta niveles, agrupados en tres categorías distintas, cada uno con una dificultad asociada. Dos de las características más destacables de HACONTI son la abundante información contextual que se proporciona sobre las vulnerabilidades de los niveles y el novedoso sistema de puntos que presenta.

Figura 1. Menú principal de HACONTI

3.1. Categorías

- **Categoría 1 (Introductoria)**: niveles diseñados para familiarizar al jugador con el entorno de Ethereum y los Smart Contracts. Aquí, los jugadores obtendrán habilidades para desplegar e interactuar con contratos, y adquirirán conocimientos fundamentales de Solidity ya que la plataforma no requiere experiencia previa, permitiendo así la resolución de desafíos posteriores.
- **Categoría 2 (Vulnerabilidades simples)**: niveles que buscan enseñar sobre alguna vulnerabilidad concreta que debe ser explotada. La particularidad

es que la vulnerabilidad de cada nivel está asociada a un SWC[1] y/o SCSVS[2] específico.
- **Categoría 3 (Vulnerabilidades complejas y/o encadenadas)**: niveles de dificultad mayor que puedan encadenar distintas vulnerabilidades simples, poniendo a prueba las habilidades adquiridas por el jugador al resolver niveles previos.

3.2. Información asociada a los desafíos

La plataforma se enfoca en proporcionar una amplia cantidad de información relacionada con cada nivel. Esto tiene como objetivo ayudar al jugador a resolver el nivel, comprender a fondo la vulnerabilidad que se aborda, aprender cómo mitigarla y comprender su impacto en situaciones del mundo real. La información que se ofrece inicialmente al jugador, antes de resolver el nivel, incluye:

- **Objetivos**: muestran los requisitos para superar el nivel.
- **Guía**: instrucciones que se encuentran únicamente en los niveles introductorios de la categoría 1. Su objetivo principal es proporcionar al jugador las pautas necesarias para resolver el nivel actual, al mismo tiempo que establecen una base fundamental para abordar con éxito desafíos más avanzados en categorías posteriores.
- **Descripción**: se encuentra en los niveles de las categorías 2 y 3, en lugar de la guía proporcionada en la categoría 1. Sirve para brindar detalles básicos sobre el desafío, orientando al jugador.
- **Hints**: una serie de ayudas y pistas para superar el nivel en cuestión. El acceso a estas pistas requiere una cierta cantidad de puntos.
- **Hints adicionales**: ofrecen una capa adicional de ayuda si las pistas anteriores no son suficientes. Requieren más puntos para desbloquearse que las pistas normales.
- **SWC**: identificador del SWC asociado a la vulnerabilidad del nivel, junto con el enlace al sitio oficial del registro de SWC. Al consultar el enlace el jugador puede entender detalladamente la vulnerabilidad abordada en el nivel, así como también ejemplos de contratos que sufren la misma vulnerabilidad, código corregido para mitigar la vulnerabilidad de ejemplo, enlaces externos con más información, etc [13]. Para desbloquearlos es necesario contar con más cantidad de puntos que para desbloquear los hints adicionales.
- **SCSVS**: los elementos del checklist de SCSVS que no se cumplen (y que generan la vulnerabilidad). Brinda información útil de manera similar a la del SWC [14]. Para desbloquearlos es necesario contar con la misma cantidad de puntos que para desbloquear el SWC asociado.

Después de resolver el nivel, se desbloquean las pistas, pistas adicionales, SWC y SCSVS sin necesidad de contar con puntos de desbloqueo. Además, se brinda al jugador la siguiente información:

[1] SWC: Smart Contract Weakness Classification https://swcregistry.io/
[2] SCSVS: Smart Contract Security Verification Standard https://github.com/ComposableSecurity/SCSVS

- **Información adicional**: detalles técnicos adicionales sobre la resolución del nivel. Ideal para entender a fondo detalles que pueden haber pasado por alto sobre la vulnerabilidad o dudas pendientes que puedan haber quedado incluso luego de resolver el nivel. Por otra parte en algunos niveles se mencionan de modo teórico las contramedidas para mitigar la vulnerabilidad.
- **Código corregido**: proporciona el código fuente corregido del contrato asociado al nivel, el cual ya no es vulnerable. Esto permite mostrar cómo se mitiga la vulnerabilidad en la práctica. Si existen varias contramedidas posibles, se presentan en secciones separadas, especificando cada una de ellas.
- **Casos reales**: ejemplos del mundo real en los que la explotación de una vulnerabilidad similar a la del nivel resultó en el robo de millones de dólares de un contrato vulnerable. Esto ayuda a comprender el impacto que lo aprendido en el juego puede tener en situaciones reales.

Esta abundante información enriquece la experiencia del jugador y promueve una comprensión profunda de las vulnerabilidades y su importancia en el contexto de la seguridad de contratos inteligentes.

3.3. Sistema de puntos

Los puntos se obtienen al resolver los niveles y, cuanto mayor es la dificultad del nivel, más puntos suma. Las dificultades de los niveles son las siguientes:

- **Muy fácil:** su resolución suma 50 puntos.
- **Fácil:** su resolución suma 100 puntos.
- **Normal:** su resolución suma 150 puntos.
- **Difícil:** su resolución suma 200 puntos.
- **Muy difícil:** su resolución suma 300 puntos.

Estos puntos sirven para posicionarse en un ranking de cara al público, similar a muchas plataformas de CTF online. Sin embargo, lo novedoso que incorpora HACONTI es que, además, los puntos son necesarios para desbloquear hints y hints adicionales, así como información de SWC y SCSVS de otros niveles. Esta información resulta de gran ayuda para los jugadores al resolver los niveles.

Adicionalmente, los puntos son necesarios para personalizar diversas secciones del perfil del jugador. Por ejemplo, se requieren 200 puntos para establecer un nick, 500 puntos para elegir la bandera del país, y acumulando más puntos se puede añadir una descripción en el perfil, así como canales de contacto como dirección de email, sitio web, y cuentas de GitHub, LinkedIn y Twitter.

3.4. Componentes de la plataforma

- **Front-end:** Es la vista de la aplicación con la que el jugador interactúa desde el navegador. Actúa como intermediario entre el jugador, la API y los contratos inteligentes de la plataforma.

- **API:** Almacena la información desbloqueable y consulta al contrato principal de la plataforma para determinar si el jugador cumple con los requisitos necesarios para acceder a la información (por ejemplo, si tiene los puntos suficientes o ha completado los niveles requeridos).
- **Contratos inteligentes:** Son el núcleo de la plataforma y se dividen en las siguientes categorías:
  - **Contrato principal de la plataforma:** Almacena en la Blockchain las direcciones de los contratos "Factory"[15] de cada nivel, las direcciones de los jugadores y los puntos acumulados por cada uno. Este contrato se comunica con los contratos "Factory"de los niveles para crear y validar las instancias de los niveles.
  - **Contratos "Factory"de los niveles:** Su función principal es desplegar dinámicamente distintos contratos de un mismo nivel para diferentes jugadores. Cada contrato de nivel es una instancia asociada a un jugador específico, lo que permite un aislamiento entre los contratos de diferentes jugadores. Los contratos "Factory"proveen métodos para crear y validar las instancias de los niveles.
  - **Contratos de las instancias de niveles:** Al hacer clic en el botón Çrear Instancia"dentro de un nivel determinado, el jugador despliega un nuevo contrato que representa una instancia del nivel que está resolviendo. Para resolver los niveles, el jugador debe interactuar directamente con estos contratos.

3.5. Optimización de la plataforma

HACONTI se enfoca en ofrecer una experiencia fluida a los jugadores mediante diversas estrategias. En el backend, el contrato principal emplea estructuras auxiliares para acceder eficientemente a los datos y optimizar el rendimiento en cada función. En el frontend, se realiza una única solicitud inicial a la API después de la autenticación del jugador o al completar un nivel, almacenando la información localmente para garantizar cargas instantáneas mientras navega entre componentes y niveles. Sin embargo, al navegar directamente a través de URL externas, la plataforma puede requerir nuevas solicitudes a la API para mantener los datos actualizados.

3.6. Seguridad de la plataforma

Si bien la finalidad de la plataforma es brindar a los usuarios desafíos de ciberseguridad en smart contracts, lo cierto es que también se tiene muy en cuenta la seguridad de la plataforma en sí, y la de sus respectivos usuarios.

Para mantener la seguridad en la plataforma se emplea un esquema sólido de autenticación y gestión de sesiones sobre la API. Por otra parte, se emplean correctamente los componentes del front-end y de la API para evitar ataques de XSS y SQL Injection.

Adicionalmente, se aplica rate limiting sobre todos los endpoints de la API, y se implementa reCAPTCHA en el formulario de registro para evitar registros

masivos por parte de bots. También se devuelven las cabeceras importantes de seguridad en las respuestas a cualquier llamado de la API, y se manejan adecuadamente los errores y excepciones para evitar que una excepción no manejada comprometa la disponibilidad de la API, y para evitar que se muestren mensajes de error con información sensible a los usuarios.

Finalmente, también se implementan protecciones contra ataques de tipo CSRF, y se verificó cuidadosamente la lógica de los contratos inteligentes del back-end para evitar cualquier clase de vulnerabilidad en la misma.

3.7. Privacidad de los usuarios

HACONTI toma en serio la privacidad de sus usuarios. Es conocido que en la red de Ethereum todas las transacciones son públicas [16]. Por este motivo, se evita almacenar cualquier clase de información personal de los usuarios en la Blockchain.

Por otra parte, la base de datos de la API tampoco almacena información sensible de los jugadores. Además, los registros de acceso del servidor, que incluyen las direcciones IP y los detalles del navegador de los usuarios, se mantienen confidenciales y no se comparten ni venden a terceros.

4. Experiencias

La plataforma fue divulgada por Moodle (UNLP), Twitter, LinkedIn, grupos de Telegram, grupos de Discord y Reddit. Se recibió una gran cantidad de feedbacks positivos, sobre todo de personas iniciadas, o transicionando desde otra área de ciberseguridad.

Se realizó una breve encuesta a través de Google Forms, que reveló que más del 80 % de los participantes consideraron excelente la plataforma en general, así como los niveles y la interfaz gráfica. El porcentaje restante la calificó como buena. Además, más del 70 % de los encuestados indicaron haber aprendido mucho sobre Solidity y vulnerabilidades en Smart Contracts jugando en HACONTI.

5. Conclusiones

Tras haber desarrollado HACONTI, se han logrado la gran mayoría de objetivos propuestos para el proyecto, ya que la plataforma desarrollada cuenta con las siguientes características:

- Es intuitiva y fácil de usar, incluso por personas sin conocimientos previos sobre Smart Contracts y el lenguaje Solidity.
- Incluye 30 niveles diferentes en los que se abordan un gran abanico de vulnerabilidades. Muchos de los niveles aportan un enfoque único y distintivo en relación a los desafíos de otras plataformas.

- Los primeros cuatro niveles de la plataforma son introductorios y guían al jugador novato de manera teórico-práctica a comprender en qué consisten los Smart Contracts, cómo interactuar con ellos, y cómo se desarrollan en Solidity. Por otra parte guían al jugador a preparar su entorno para resolver niveles posteriores.
- Los últimos tres niveles incluyen una serie de vulnerabilidades que deben ser explotadas en cadena para resolverlos. Dichos niveles, altamente desafiantes, sirven a modo de poner a prueba lo aprendido en niveles previos.
- Por cada nivel se presenta una cantidad de información abundante, que no solo abarca los objetivos de superación y la descripción del desafío, sino que también incluye hints, hints avanzadas, información relacionada con SWC y SCSVS, información adicional, el código fuente corregido del nivel, casos del mundo real, etc. Esta información brinda un mayor contexto y una comprensión más amplia sobre la vulnerabilidad que se explota en cada nivel. Cada sección informativa suele ser extensa y detallada para cubrir la mayor cantidad de puntos posibles.
- Incluye un sistema de puntos novedoso en el que el jugador puede desbloquear hints, hints adicionales e información vinculada a SWC y SCSVS, a medida que supera los niveles y va acumulando puntos. El diseño de la plataforma está hecho de tal manera que solo los jugadores con los puntos necesarios puedan acceder a la información desbloqueable y que la misma no esté disponible para cualquiera.
- Los componentes de la plataforma, es decir, el front-end y los distintos elementos del back-end fueron desarrollados meticulosamente teniendo en cuenta la optimización, seguridad y legibilidad del código para su fácil mantenimiento.

Las características mencionadas previamente permiten que las personas sin conocimientos previos en seguridad en Smart Contracts ni el lenguaje Solidity, puedan aprender del campo de manera exhaustiva, en un entorno controlado, seguro, simple, divertido y muy informativo. De esta manera, estarán capacitadas para desarrollar sus propios contratos de forma segura y llevar a cabo pentests sobre los mismos.

6. Trabajo a futuro

A futuro se contempla la posibilidad de integrar la plataforma en cursos de seguridad de la Facultad de Informática.

Por otra parte, se pretende añadir niveles basados en nuevas vulnerabilidades, o bien añadir niveles introductorios, como por ejemplo, uno que explique en detalle cómo funciona el storage de un contrato.

Adicionalmente, se planea agregar nuevas funcionalidades, como por ejemplo, incluir métodos auxiliares en JavaScript para realizar tareas frecuentes desde la consola del navegador, y también mejorar la accesibilidad de la plataforma.

Referencias

[1] B. Meyer. "Ethereum Smart Contract Vulnerabilities". En: *CyberNews* (15 de nov. de 2023). URL: https://cybernews.com/security/ethereum-smart-contract-vulnerabilities/.

[2] D. Z. Morris. "How the DAO Hack Changed Ethereum and Crypto". En: *Consensus Magazine - Coindesk* (9 de mayo de 2023). URL: https://www.coindesk.com/consensus-magazine/2023/05/09/coindesk-turns-10-how-the-dao-hack-changed-ethereum-and-crypto/.

[3] S. Palladino. "The Parity Wallet Hack Explained". En: *OpenZeppelin Blog* (19 de jul. de 2017). URL: https://blog.openzeppelin.com/on-the-parity-wallet-multisig-hack-405a8c12e8f7.

[4] *Smart Contract Security Field Guide: Reentrancy*. 2023. URL: https://scsfg.io/hackers/reentrancy/.

[5] Shashank. "Integer Overflow and Underflow in Smart Contracts". En: *SolidityScan* (31 de ago. de 2022). URL: https://blog.solidityscan.com/integer-overflow-and-underflow-in-smart-contracts-9598032b5a99.

[6] "Common Vulnerabilities in Solidity: Denial of Service (DOS)". En: *SlowMist* (30 de mar. de 2023). URL: https://www.slowmist.com/articles/solidity-security/Common-Vulnerabilities-in-Solidity-Denial-of-Service-DOS.html.

[7] "Access Control Vulnerabilities in Solidity Smart Contracts". En: *Immunebytes Blog* (12 de sep. de 2023). URL: https://www.immunebytes.com/blog/access-control-vulnerabilities-in-solidity-smart-contracts.

[8] Natachi Nnamaka. "Attack Vectors in Solidity #4: Unencrypted Private Data On-Chain". En: *Medium* (27 de ene. de 2023). URL: https://medium.com/@natachigram/attack-vectors-in-solidity-4-unencrypted-private-data-on-chain-cf4f3ff1cf71.

[9] Martin Liguori. "Preventing the Source of Randomness Vulnerability". En: *Infuy Blog* (8 de mar. de 2023). URL: https://www.infuy.com/blog/preventing-the-source-of-randomness-vulnerability.

[10] Yogesh Kulkarni. "tx.origin Phishing Attack — Smart Contract Security". En: *Finxter Academy* (1 de ene. de 2023). URL: https://academy.finxter.com/tx-origin-phishing-attack-smart-contract-security/.

[11] *Smart Contract Security Field Guide: Unexpected Ether*. 2023. URL: https://scsfg.io/hackers/unexpected-ether/.

[12] *Understanding Delegatecall Vulnerabilities in Solidity: A Comprehensive Guide*. 2023. URL: https://auditfirst.io/blog/understanding-delegatecall-vulnerabilities-in-solidity.

[13] G. (@thec00n) Wagner. "Smart Contract Weakness Classification (SWC)". En: *Ethereum Improvement Proposal (EIP) 1470* (). URL: https://eips.ethereum.org/EIPS/eip-1470.

[14] *Smart Contract Weakness Classification (SWC) Registry*. URL: https://swcregistry.io/.

[15] Wissal Haji. "Learn Solidity: The Factory Pattern". En: *Better Programming* (13 de dic. de 2020). URL: https://betterprogramming.pub/learn-solidity-the-factory-pattern-75d11c3e7d29.

[16] Jon Law. "Traceability of Ethereum Transactions". En: *Coinmonks* (15 de mar. de 2022). URL: https://medium.com/coinmonks/traceability-of-ethereum-transactions-93a880384a20.