УДК 004.056

ПОКРАЩЕННЯ БЕЗПЕКИ СМАРТ-КОНТРАКТІВ У МЕРЕЖІ ETHEREUM

Д. О. Ващенок[1], Л. Ю. Гальчинський[1]

[1] Навчально-науковий Фізико-технічний інститут

Анотація

Блокчейн-технології, вперше впроваджені Ethereum, забезпечують децентралізацію та автоматизацію угод через смарт-контракти. Незважаючи на їхню корисність, зростання кількості шахраїв у сфері криптовалют стало серйозною загрозою. Децентралізована природа криптовалют робить важким відстеження та повернення коштів, а недоліки у протоколах залишають простір для атак. Блокчейн гарантує неперезаписувальність даних, що робить його відмінним від традиційних банківських систем. iBatch, нова платформа, що пропонує значне покращення ефективності транзакцій на основі Ethereum.

Ключові слова: блокчейн, Ethereum, смарт-контракт, консенсус

Вступ

Смарт-контракт [1] – це програмний код, який зберігається на блокчейні та автоматизує виконання угод між сторонами без посередництва [2]. Його основна ідея полягає в тому, щоб забезпечити автоматичну та надійну реалізацію угод, уникаючи можливості маніпуляцій чи змін угод після їх укладання. Основні вразливості смарт-контрактів включають в себе можливість програмних помилок (багів), які можуть призвести до некоректної роботи контракту або втрати коштів, а також атаки типу "вилучення коштів"(якщо у коді контракту виявлено слабкі місця, що дозволяють зловмисникам використовувати ці уразливості для видалення коштів з контракту).

1. Аналіз проблеми

Поглиблений аналіз проблеми вразливостей смарт-контрактів виявляється критично важливим у зв'язку з розмаїттям сценаріїв атак та потенційних наслідків для користувачів інфраструктури блокчейн. Використання недоліків у коді може призвести до втрати коштів, несправедливого розподілу прав чи навіть втрати доступу до активів [3]. Наприклад, вразливість у смарт-контракті може викликати ситуацію, де зловмисник може викрасти кошти з контракту або змінити його умови в свою користь. Крім того, можливість використання уразливостей у контракті може викликати збої в децентралізованих додатках, порушуючи принципи безпеки та довіру, на яких ґрунтується технологія блокчейн. Для запобігання таким загрозам, розробники смарт-контрактів повинні використовувати кращі практики програмування та безпеки. Не менш важливо проводити регулярні аудити безпеки та оновлення смарт-контрактів для виявлення та виправлення потенційних уразливостей.

Відомим прикладом є атака на смарт-контракт «The DAO» в 2016 році, в результаті якої було викрадено близько $50 мільйонів. Цей інцидент підкреслив необхідність удосконалення безпеки смарт-контрактів та важливість аудиту коду перед його використанням. Додатково, інші подібні атаки, такі як та, що сталася з платформою DeFi Harvest Finance у 2020 році, коли було викрадено близько $24 мільйонів, підкреслюють необхідність постійного моніторингу та вдосконалення безпеки смарт-контрактів. Далі маємо задачу дослідити які вразливості можуть бути у смарт-контракти, їх природу та вплив на користувачів.

2. Наявні вразливості у смарт-контрактах

Наявні вразливості у смарт-контрактах є серйозним об'єктом переймання уваги в контексті блокчейн-екосистеми. Ці вразливості можуть походити з різноманітних джерел, включаючи помилки в програмному коді, недоліки у дизайні, неочікувані інтерпретації даних та вразливості, що виникають в результаті взаємодії з іншими компонентами системи. Навіть найменша вразливість може призвести до серйозних наслідків, таких як втрати коштів, втрата конфіденційності даних або втрата довіри до всього блокчейн-протоколу. Тому важливо постійно вдосконалювати методи аналізу та виявлення вразливостей у смарт-контрактах, а також активно застосовувати кращі практики безпеки програмного забезпечення для мінімізації ризиків. Було проаналізовано існуючі вразливості у смарт-контрактах, їх наведено нижче.

1) **Vulnerable price feed** Вразливість виникає, коли смарт-контракт використовує недостовірний або недостатньо захищений джерело ціни для виконання фінансових операцій, що може

призвести до маніпуляцій цінами та втрати коштів.

2) **Reentrancy** Атака, при якій зловмисник може повторно викликати функції смарт-контракту перед закінченням попередніх операцій, що може призвести до втрати коштів.
3) **Incorrect deploy and post-deploy settings** Вразливість виникає, коли конфігурації або налаштування в смарт-контрактах, які встановлюються під час або після розгортання, не перевіряються на достовірність або помилково не приділяють уваги.
4) **Inflation attack** Вразливість, при якій атакуючий може використовувати недоліки в грошовій політиці токена або контракту, що призводить до непередбачених випусків або спалювання токенів, маніпулюючи їх вартістю та обсягом.
5) **Related to stolen private keys** Вразливість виникає при крадіжці приватних ключів, що дозволяє зловмисникам мати повний контроль над смарт-контрактом та пов'язаними з ним активами.
6) **Attack with fake contract** Вразливість, коли зловмисники створюють фальшивий контракт, що намагається імітувати реальний контракт зі злоумисною логікою для шахрайства.
7) **Incorrect calculations and accuracy loss** Вразливість, коли смарт-контракт неправильно виконує математичні операції або втрачає точність у результаті обчислень, що може призвести до втрати коштів або некоректного розподілу токенів.
8) **Race condition** Вразливість, коли порядок виконання транзакцій має вплив на результат виконання контракту, що може бути використано для атак.
9) **Incorrect work with erc20 token** Вразливість, коли смарт-контракт неправильно взаємодіє з токенами ERC20, що може призвести до втрати коштів або некоректного розподілу токенів.

Механізм роботи вразливості reentrancy Рис. 1:

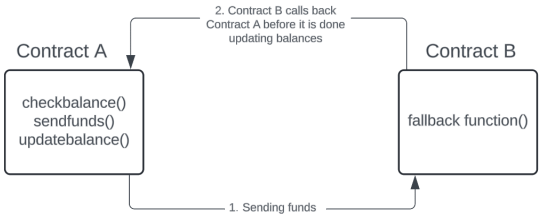

Рис. 1. Приклад вразливості Reentrancy

Ряд дослідників пропонують ідею скасування підозрілих транзакції для зменшення втрат користувачів, для чого треба провести глибокий аналіз механізму арбітражу повернення транзакцій [4]. Інші бачать підвищення кібербезпеки мережі за рахунок безпечного пакетування викликів смарт-контрактів проти ненадійного сервера ретрансляції поза мережею [5]. Існують також інші підходи. Тому надалі доцільно розглянути різні підходи для виявлення і нейтралізації вразливостей.

3. Інструменти для виявлення вразливостей у смарт-контрактах

У даному розділі буде обговорено поточні методи виявлення вразливостей у смарт-контрактах, включаючи традиційні методи та методи, що базуються на глибокому навчанні. Традиційні методи включають у себе символьне виконання, формальну верифікацію та тестування фаззінгом. Один з інструментів, який використовує статичне символьне виконання для виявлення потенційних вразливостей у смарт-контрактах, називається Oyente. Ще один інструмент, під назвою Mythril, поєднує статичне символьне виконання, аналіз забруднення та перевірку потоку керування для підвищення точності виявлення вразливостей. Інструмент Slither може виявляти вразливості, перетворюючи код контракту в проміжне представлення. ContractFuzzer є першим фреймворком виявлення вразливостей, який застосовує техніки тестування фаззінгом для смарт-контрактів. Множину інструментів для виявлення вразливостей у смарт-контрактах можна поділити на традиційні методи (такі як символьне виконання, формальна верифікація та тестування фаззінгом) та методи глибокого навчання (такі як ReChecker та AME). Традиційні методи акцентуються на статичних символьних методах виконання, в той час як методи глибокого навчання включають різні моделі, такі як BLSTM-ATT, графові нейронні мережі та механізми самоуваги. Ці методи відіграють важливу роль у підвищенні безпеки смарт-контрактів. На Рис. 2 показана різниця між традиційними методами та методами машинного навчання.

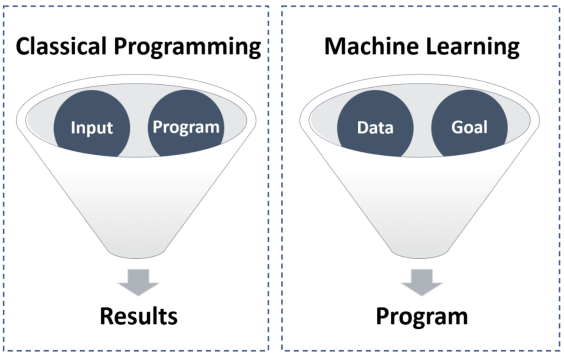

Рис. 2. Різниця між класичними методами, що грунтуються на програмуванні і машинним навчанням

4. Запропоноване рішення

Варіантом для розгляду є iBatch [5] – проміжне програмне забезпечення, що працює поверх існуючої мережі Ethereum, пропонуючи безпечну пакетну

передачу викликів смарт-контрактів на ненадійний ретрансляційний сервер поза ланцюжком. Це рішення з низькими накладними витратами, яке перевіряє пакетні виклики смарт-контрактів на сервері без додаткових станів користувацьких нонсенсів. iBatch значно економить витрати на газ, знижуючи їх на $14,6\%$ - $59,1\%$ на один виклик, що дає змогу більшій частині користувачів використовувати мережу. Також цей інструмент дає змогу автоматично переписувати смарт-контракти для безпроблемної інтеграції зі старими додатками, які були запущені декілька років назад. Проблема поєднання старих смарт-контрактів з наявними налаштуваннями мережами іноді стає дуже актуальною, і починають з'являтися нові можливості використати ці вразливості. Наразі визначено, що iBatch являє собою досить потужне та економічне рішення для масштабування Ethereum. Завдяки своїм можливостям перевірки пакетних викликів, адаптивної пакетної обробки та автоматичного переписування смарт-контрактів, iBatch пропонує безпечний та ефективний спосіб покращити продуктивність та економність Ethereum.

Висновки

Було проаналізовану проблему вразливостей у смарт-контрактах. Стало зрозуміло що це питання є критичним для користувачів мережі та доволі поширеним. Багато розробників смарт-контрактів не приділяють належної уваги для їх створення, або смарт-контракт може бути просто застарілим, і бути вразливим. Для більш глибокого аналізу було розглянуто більшість основних вразливостей, і шляхи їх виявлення. З проведеного дослідження було виявлено, що статичний аналіз коду не є універсальним способом для оцінки смарт-контракту, і треба шукати більш кращу версію вирішення цього питання. Було проведено аналіз динамічних додатків для виявлення вразливостей, які показали себе краще ніж попередні, але не є ідеальним варіантом для створення безпечного середовища. Саме тому була поставлена задача, яка покращить безпеку мережі, і зробить її більш масштабованою. Виявлено що цим інструментом може слугувати iBatch, який слугує мостом між безпекою смарт-контракту та масштабованістю мережі.

Перелік використаних джерел

1. *Buterin V.* Ethereum White Paper. A Next Generation Smart Contract. Decentralized Application Platform. — URL: https://ethereum.org/en/whitepaper.
2. EIP721: Non-Fungible Token Standard. Ethereum Improvement Proposals no. 721 / W. Entriken, D. Shirley, J. Evans, N. Sachs. — URL: https://eips.ethereum.org/EIPS/eip-721.
3. *Moser M.*, *Eyal I.*, *Sirer E. G.* Bitcoin covenants. In Financial Cryptography // Lecture Notes in Computer Science. — 2016. — URL: https://workbench.cisecurity.org.
4. *Топчій М.*, *Гальчинський Л.* Підвищення рівня безпеки смарт-контрактів в мережі ethereum від шахрайства за рахунок використання реверсивних токенів // Collection of Scientific Papers «ΛΌГОΣ», (November 11, 2022; Paris, France). — 2022.
5. Towards saving blockchain fees via secure and cost-effective batching of smart-contract invocations / Y. Wang, K. Li, Y. Tang, J. Chen, Q. Zhang, X. Luo, T. Chen. — URL: https://www.researchgate.net/publication/367231490_Towards_Saving_Blockchain_Fees_via_Secure_and_Cost-Effective_Batching_of_Smart-Contract_Invocations.