ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО- КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка системи моніторингу та сповіщення про підозрілі транзакції на базі EVM-блокчейнів»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
(код, найменування спеціальності)
освітньо-професійної програми «Інженерія програмного забезпечення»
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Павло ЗІБАРОВ
(підпис)

Виконав: здобувач вищої освіти групи ПДМ-61
Павло ЗІБАРОВ

Керівник: Олена НЕГОДЕНКО
к.т.н, доцент

Рецензент: Вікторія ЖЕБКА
д.т.н., професор

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти Магістр
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедрою
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
«___» _____20__р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Зібарову Павлу Сергійовичу

1. Тема кваліфікаційної роботи: Розробка системи моніторингу та сповіщення про підозрілі транзакції на базі EVM-блокчейнів

керівник кваліфікаційної роботи Олена НЕГОДЕНКО к.т.н., доцент,

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «19» жовтня 2023р. № 145.

2. Строк подання кваліфікаційної роботи «29» грудня 2023р.
3. Вихідні дані до кваліфікаційної роботи:
1. Науково-технічна література.
2. Статистичні дані про шахрайство в блокчейні.
3. Транзакції та параметри блокчейн мереж.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз проектного середовища та проблеми.
2. Аналіз попередніх досліджень та формування моделі системи.
3. Програмна реалізація системи моніторингу підозрілих транзакцій.

5. Перелік ілюстративного матеріалу: *презентація*

1. Порівняння розробленого рішення з існуючими.
2. Математична модель системи.
3. Показники транзакції, що аналізуються.
4. Запропоновані сценарії застосування.
5. Алгоритм роботи системи.
6. Структура схема моніторингової системи.
7. Діаграма класів розробленої системи.
8. Практична реалізація.
9. Результати моделювання тестової ситуації.

6. Дата видачі завдання «19» жовтня 2023р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз існуючих наукових досліджень за темою роботи	19.10-05.11.23	
2	Вивчення матеріалів щодо принципів та засобів для роботи з блокчейном	06.11-12.11.23	
3	Дослідження технології смарт контрактів	13.11-19.11.23	
4	Аналіз потенційних загроз для користувача	20.11-26.11.23	
5	Побудова математичної моделі системи	27.11-03.12.23	
6	Розробка програмної системи	04.12-10.12.23	
7	Оформлення роботи: вступ, висновки, реферат	11.12-20.12.23	
8	Розробка демонстраційних матеріалів	21.12-29.12.23	

Здобувач вищої освіти _____ Павло ЗІБАРОВ
(підпис)

Керівник
кваліфікаційної роботи _____ Олена НЕГОДЕНКО
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 76 стор., 25 рис., 2 табл., 20 джерел.

Мета роботи – покращення рівня користувацької безпеки за рахунок моніторингу та сповіщення про підозрілі транзакції на децентралізовані протоколи.

Об`єкт дослідження: моніторингові системи транзакцій на базі EVM-блокчейнів.

Предмет дослідження: система моніторингу та сповіщення про підозрілі транзакції на базі EVM-блокчейнів.

Короткий зміст роботи: Запропоновано алгоритм та реалізацію моніторингової системи, що дозволяє повідомляти користувача про підозрілу активність зі смарт контрактом проєкту, в який користувач інвестує децентралізовані активи.

КЛЮЧОВІ СЛОВА: БЛОКЧЕЙН, КРИПТОВАЛЮТИ, ДЕЦЕНТРАЛІЗОВАНІ АКТИВИ, СМАРТ КОНТРАКТ, ТРАНЗАКЦІЯ, БЛОК, ETHEREUM, КІБЕРБЕЗПЕКА, МОНІТОРИНГ.

ABSTRACT

Text part of the master's qualification work: 76 pages, 25 pictures, 2 tables, 20 sources.

The purpose of the work – to improve the level of user security by monitoring and reporting suspicious transactions on decentralized protocols.

Object of research – transaction monitoring systems based on EVM blockchains.

Subject of research – a system for monitoring and reporting suspicious transactions based on EVM blockchains.

Summary of the work: The work presents an algorithm and implementation of a monitoring system that allows users to notify the user of suspicious activity with the smart contract of a project in which the user invests decentralized assets. Users will be able to generate requests for monitoring transactions from specific wallets and for specific protocols, while applying filters to the parameters of calling smart contract methods. Due to the precise monitoring algorithm and high-quality blockchain of the provider, the time interval between the detection of a suspicious transaction and notification is minimal and has no delays.

The mathematical model of the system developed taking into account the transaction confirmation time showed that the process of detecting a transaction in the pool of future transactions is minimal, and the process of processing a suspicious transaction by the system takes less time than the process of including the transaction in the block, which allows the user to send a response transaction, which can minimize losses from a potential threat.

During the development of the scientific article, the authors analyzed existing security solutions and approaches to preserve user assets involved in blockchain protocols. In addition to existing solutions, statistics and specific approaches to protocol hacking, as well as fraudulent projects that have been noticed stealing invested user assets, were investigated. Specific fraud cases were analyzed, including specific types of

transactions, namely transactions to withdraw liquidity from decentralized exchanges, as well as manipulations to transfer ownership of the protocol to another account or to stop the protocol.

The developed system can be used on all blockchains built on the basis of EVM - Ethereum Virtual Machine, namely Ethereum, Binance Smart Chain, Polygon, etc., which will increase the overall level of security of user assets, as well as inform users about possible threats when interacting with new unknown protocols. An important point is to update and add new types of malicious transactions to the monitoring system to ensure the prevention of more fraud cases.

Based on the designed system, automated systems can be developed that automatically send a user transaction to counteract the detected suspicious transaction.

KEYWORDS: BLOCKCHAIN, CRYPTOCURRENCIES, DECENTRALIZED ASSETS, SMART CONTRACT, TRANSACTION, BLOCK, ETHEREUM, CYBERSECURITY, MONITORING.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 8

ВСТУП 9

РОЗДІЛ 1 АНАЛІЗ ПРОЕКТНОГО СЕРЕДОВИЩА ТА ПРОБЛЕМИ 12

1.1 Дослідження складових мережі Ethereum 12

1.2 Огляд поняття смарт контрактів та їх варіантів застосування 15

1.3 Аналіз потенційних ризиків при роботі зі смарт контрактами 19

РОЗДІЛ 2 АНАЛІЗ ПОПЕРЕДНІХ ДОСЛІДЖЕНЬ ТА ФОРМУВАННЯ МОДЕЛІ СИСТЕМИ 28

2.1 Аналіз досліджень на тему безпеки в блокчейні 28

2.2 Висновки на основі розглянутих досліджень 38

2.3 Порівняльна характеристика проаналізованих безпекових рішень 38

2.4 Побудова математичної моделі системи моніторингу 41

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОНІТОРИНГОВОЇ СИСТЕМИ ПІДОЗРІЛИХ ТРАНЗАКЦІЙ 43

3.1 Алгоритм роботи розробленої системи 43

3.2 Функціональні та нефункціональні вимоги до системи 45

3.3 Використані програмні засоби в процесі розробки системи 47

3.4 Опис модулів програмної системи 50

3.5 Основні розроблені класи системи 57

3.6 Користувацький інтерфейс системи 60

3.7 Ефективність розробленої системи 63

ВИСНОВКИ 65

ПЕРЕЛІК ПОСИЛАНЬ 67

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ 69

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ERC – Ethereum request for comment

EVM – Ethereum Virtual Machine

DAPP – decentralized application

DEFI – decentralized finances

ABI – application binary interface

SQL – Structured Query Language

ООП – об’єктно-орієнтоване програмування

СУБД – система управління базами даних

БД – база даних

ІС – інформаційна система

ІТ – інформаційні технології

ПЗ – програмне забезпечення

ВСТУП

Актуальність роботи

В епоху стрімкого розвитку криптовалют та блокчейн технологій, виникає необхідність ефективного та надійного контролю за фінансовими операціями, які проводяться на основі EVM-блокчейнів. Питання безпеки звичайних користувачів, що використовують цифрові активи є нагальним, оскільки зі збільшенням популяризації зростає кількість протоколів, в тому числі ішахрайських. Користувач, що інвестує свої активи може не до кінця дослідити проєкт, таким чином наражаючи себе на потенційні загрози. Представлене рішення допоможе попередити користувача про підозрілу транзакційну активніст зі сторони власника протоколу: від адміністраторських операцій, до продажу токенів проєкту, що може призвести до обвалу ціни.

Ethereum Virtual Machine (EVM) визначає стандарт виконання смарт-контрактів на блокчейні Ethereum, надаючи широкий спектр можливостей для створення різноманітних фінансових інструментів та децентралізованих додатків. Проте, разом із зростанням популярності цих технологій, виникає необхідність забезпечення їхньої безпеки та захисту від потенційних зловживань.

Мета і задачі дослідження

Мета роботи: покращення рівня користувацької безпеки за рахунок моніторингу та сповіщення про підозрілі транзакції на децентралізовані протоколи.

Досягнення мети включало розв’язання таких **задач**:

- Дослідження основних понять та принципів роботи блокчейн
- Аналіз існуючих досліжень в сфері безпекових рішень в блокчейні
- Визначення безпекових ризиків зі сторони користувачів блокчейну
- Побудова математичної моделі системи
- Визначення вимог до системи
- Проектування архітектури системи
- Опис архітектури системи за допомогою діагр

- Програмна реалізація спроектованої системи

Об’єкт дослідження: моніторингові системи транзакцій на базі EVM-блокчейнів.

Предмет дослідження: система моніторингу та сповіщення про підозрілі транзакції на базі EVM-блокчейнів.

Методи дослідження

Основними методами, які використовувались під час написання кваліфікаційної роботи були:

1. Аналіз користувацьких безпекових рішень на базі EVM-блокченів
2. Аналіз існуючих загроз та можливих варіантів попередження та нейтралізації наслідків
3. Гіпотетичний метод – сформовано припущення про можливість відслідковування транзакцій в процесі їх додавання в блокчей
4. Емпіричний метод – на основі власного досвіду запропоновано базові сценарії застосування розробленої системи

Наукова новизна отриманих результатів

В ході виконання кваліфікаційної роботи студентом було запропоновано рішення для попередження про можливі загрози для користувацьких децентралізованих фінансів за рахуно застовсування системи для моніторингу та сповіщення про підозрілі транзакції.

Практичне значення одержаних результатів

Реалізація програмної системи дозволить користувачу самостійно організовувати заходи для захисту вкладених активів за допомогою відслідковування потенційної шахрайської діяльності протоколу чи зламу.

Особистий внесок студента

Запропоновано математичну модель моніторингової системи для підозрілих транзакцій, розроблено алгоритм фільтрації транзакцій. Розроблено програмний продукт на основі результатів дослідження

Апробація дослідження

Робота пройшла апробацію на V Міжнародній науково-практичній конференції молодих вчених та студентів «ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ І ПЕРЕДОВІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ. За результатами апробації опубліковано тези доповіді та наукову статтю.

Наукова стаття: Зібаров П.С., Негоденко О.В. Розробка системи моніторингу та сповіщення про підозрілі транзакції на базі EVM-блокчейнів. // Науковий журнал «Молодий вчений». – № 12 (124) грудень 2023, Київ.

Тези доповіді: Зібаров П.С., Негоденко О.В. Розробка системи моніторингу та сповіщення про підозрілі транзакції на базі EVM-блокчейнів. // V Міжнародна науково-практична конференція молодих вчених та студентів «ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ І ПЕРЕДОВІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ». – Київ: КПІ, 2023.

Структура та обсяг роботи

Робота викладена на 91 сторінках друкованого тексту, який складається із вступу, трьох розділів, висновків, списку використаних джерел (31 найменування). Робота містить 1 формулу, 2 таблиці, 24 рисунки та 2 додатки(16 сторінок).

1 АНАЛІЗ ПРОЕКТНОГО СЕРЕДОВИЩА ТА ПРОБЛЕМИ

Ethereum - це децентралізована платформа блокчейну, створена для розробки і розгортання смарт-контрактів та децентралізованих додатків (DApps). В основі Ethereum лежить технологія блокчейну, яка використовується для створення групи публічних записів (блоків), що зберігаються на різних комп'ютерах у мережі – вузлах.

1.1 Дослідження складових мережі Ethereum

Один із ключових елементів Ethereum - це смарт-контракти, які є самовиконувальними програмами, зберігаються на блокчейні і виконують дії відповідно до визначених умов. Ethereum також використовує свою власну криптовалюту, називану ефіром (ETH), яка використовується для оплати транзакцій та винагороди майнерам за обробку блоків.

Транзакція в Ethereum – це основний елемент взаємодії між учасниками мережі, спрямований на передачу значень (криптовалюти або інших цифрових активів) або виконання функцій в смарт-контрактах. Вона представляє собою вказівку блокчейну на виконання конкретної дії та є важливим складовим елементом, який додається до ланцюга блоків Ethereum.

Основні властивості транзакції:

—Відправник: адреса гаманця,який ініціює транзакцію.

—Отримувач: адреса гаманця або смарт-контракту, який отримує певні дані – цифрові активи, якщо це переказ нативної валюти або параметри виклику смарт контракту.

—Сума: кількість нативної криптовалюти (eth, bnb), яка надсилається відправником отримувачу.

—Дані: додаткові дані, які можуть супроводжувати транзакцію. Для звичайних переказів нативної це може бути порожній рядок, але в

контексті смарт-контрактів тут може міститися виклик функції та її параметри.

—Газ і газова ціна: газ визначає обчислювальні ресурси, необхідні для виконання транзакції. Газова ціна вказує, скільки платити за одиницю газу. Вартість транзакції розраховується як газова ціна помножена на кількість газу.

—Підпис: криптографічний підпис відправника, який підтверджує, що транзакцію відправив саме власник приватного ключа від адреси відправника.

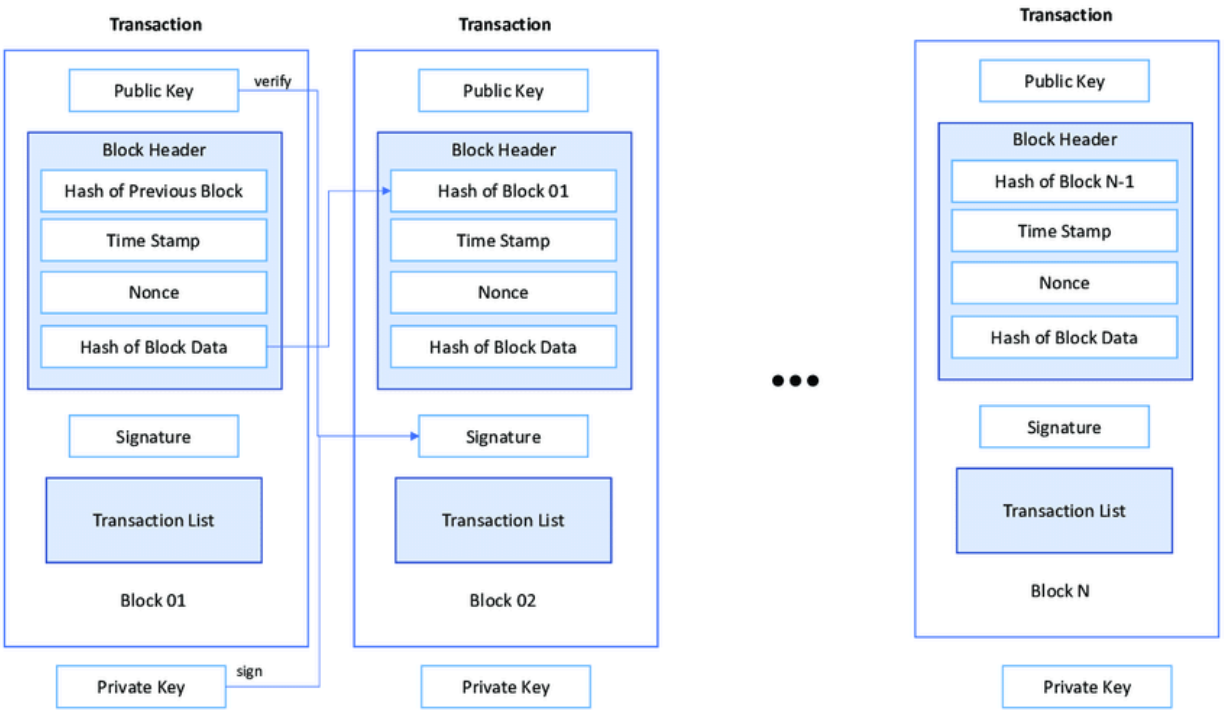


Рис. 1.1 Структура транзакцій в блокчейні Ethereum

Життєвий цикл транзакції в блокчейні Ethereum розпочинається з моменту її ініціювання користувачем. Під час цього етапу, він визначає параметри транзакції, такі як отримувач, сума переказу, параметри виклику, якщо отримувач є смарт контрактом. За виконання транзакції користувач має заплатити комісію. Значення комісії може бути змінене залежно від навантаження на мережу. Чим вища сума комісії, тим більш вірогідно, що валідатор включить транзакцію в блок.

Далі, транзакція підписується приватним ключем відправника для підтвердження його автентичності та права витрати нативної валюти. Підписана

транзакція відправляється до мережі Ethereum через вузол або групу вузлів відправника, зазвичай таким вузлом виступає додаток-гаманець.

Після відправлення транзакція потрапляє в мемпул – пул непідтверджених транзакцій, де вузли очікують на її включення в новий блок.

Валідатори вибирають транзакції з мемпулу для включення в новий блок, розглядаючи комісії, які користувачі готові заплатити за виконання їхніх транзакцій. Обраний майнер формує блок, перевіряє транзакції та включає найбільш вигідні для ньоготранзакції в цей блок.

Після того, як новий блок, що містить транзакцію, додається до ланцюга блоків, транзакція отримує підтвердження. Зазвичай, користувач вважає транзакцію безпечною після отримання кількох підтверджень, які представляють собою кількість блоків, що йдуть за поточним.

У разі включення виклику смарт контрактів у транзакцію, вони виконують свою програму, вносячи зміни в стан блокчейну. Саме за допомогою таких викликів здійснюється більшість взаємодії з децентралізованими системами.

Таким чином, весь цей цикл відображає процес взаємодії між учасниками мережі Ethereum, починаючи від власника коштів, який ініціює транзакцію, до валідаторів, які обирають та вирішують блоки, що включають цю транзакцію в ланцюг блоків.

Смарт-контракт в контексті блокчейну, зокрема в системі Ethereum, є автоматизованим, самовиконуючимся контрактом, який містить програмний код і зберігається в блокчейні. Це програма, яка автоматизовано виконує умови, записані в її коді, без необхідності посередництва або довіреності.

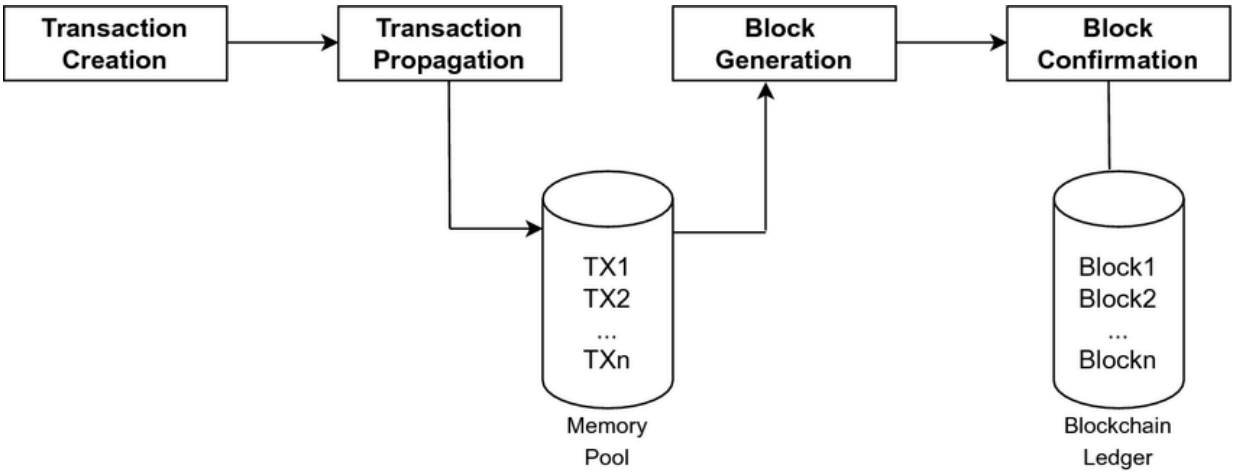


Рис. 1.2 Життєвий цикл транзакцій в блокчейні Ethereum

1.2 Огляд поняття смарт контрактів та їх варіантів застосування

Смарт контракт – це програмне рішення, що містить в собі певні дані та функції для обробки цих даних, залежно від виникнення певних умов. Виконання коду відбувається автоматизовано згідно з прописаними в коді умовами та без посередників. Смарт контракт опублікований в блокчейні має унікальну адресу ідентифікатор, за якою можна звертатись до нього.

Смарт контракт містить програмний код, написаний мовою, яка підтримується платформою блокчейну Solidity в контексті EVM-блокчейнів. Цей код визначає правила та умови виконання контракту. Користувачі блокчейну можуть виконувати цей код за допомогою транзакцій, що містять в собі дані виклику певного метода смарт контракту. Код смарт контракту є загальнодоступним та незмінним після публікації контракту в мережу.

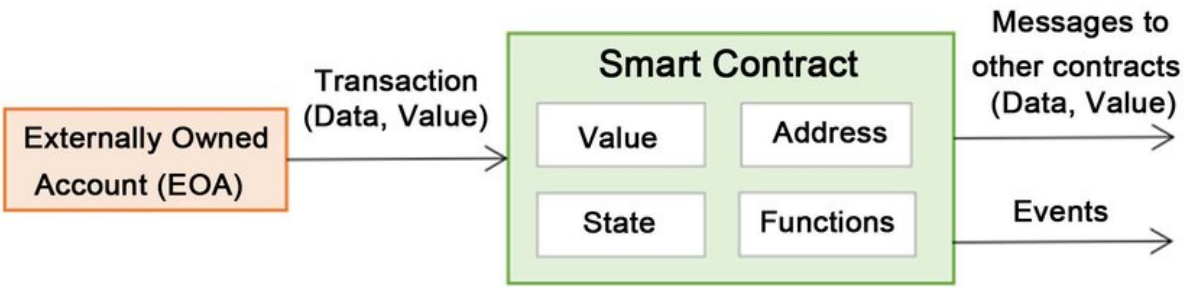


Рис. 1.3 Взаємодія аккаунту зі смарт контрактом

Також смарт контракт може зберігати в собі внутрішній стан, який може змінюватися в результаті викликів функцій смарт контракту. Стан зберігається в блокчейні та може бути переглянутий публічно. Функції на зміну стану потребують обчислювальних потужностей, відповідно на них накладається комісія, функції на читання є безкоштовними.

Нижче представлений перелік стандартів та протоколів, в основі яких закладені смарт контракти.

Токен ERC-20 є основним стандартом токенів на блокчейні Ethereum. Цей стандарт визначає, які функції та методи повинні бути реалізовані в смарт контрактах токенів на базі Ethereum. Токени ERC-20 є цифровими активами, які можуть представляти різні цінності або права, і вони використовуються для реалізації токенізованих активів, ICO та інших криптовалютних проєктів. Зазвичай токен є основою будь-якого проєкту.

На основі смарт контрактів побудовано безліч децентралізованих додатків, що набирають популярності серед власників криптовалют. Нижче представлені декілька таких проєктів та описані їхні принципи роботи.

Uniswap V2 - це децентралізована обмінна платформа, побудована на блокчейні Ethereum, яка дозволяє користувачам обмінювати різні криптовалютні токени без необхідності використання традиційних централізованих бірж. Проект створений з використанням смарт контрактів, що дозволяє автоматизовано проводити обміни, забезпечуючи ліквідність іншим користувачам. Учасники мережі можуть надавати ліквідність, ставлячи свої токени в пару токен-ETH. Такі пари називаються пулами ліквідності. Uniswap використовує принцип автоматизованого маркет-мейкінгу. Це означає, що ціни на обмін визначаються автоматично шляхом алгоритму, а не через взаємодію покупців та продавців. Користувачі можуть використовувати Uniswap для купівлі та продажу токенів.

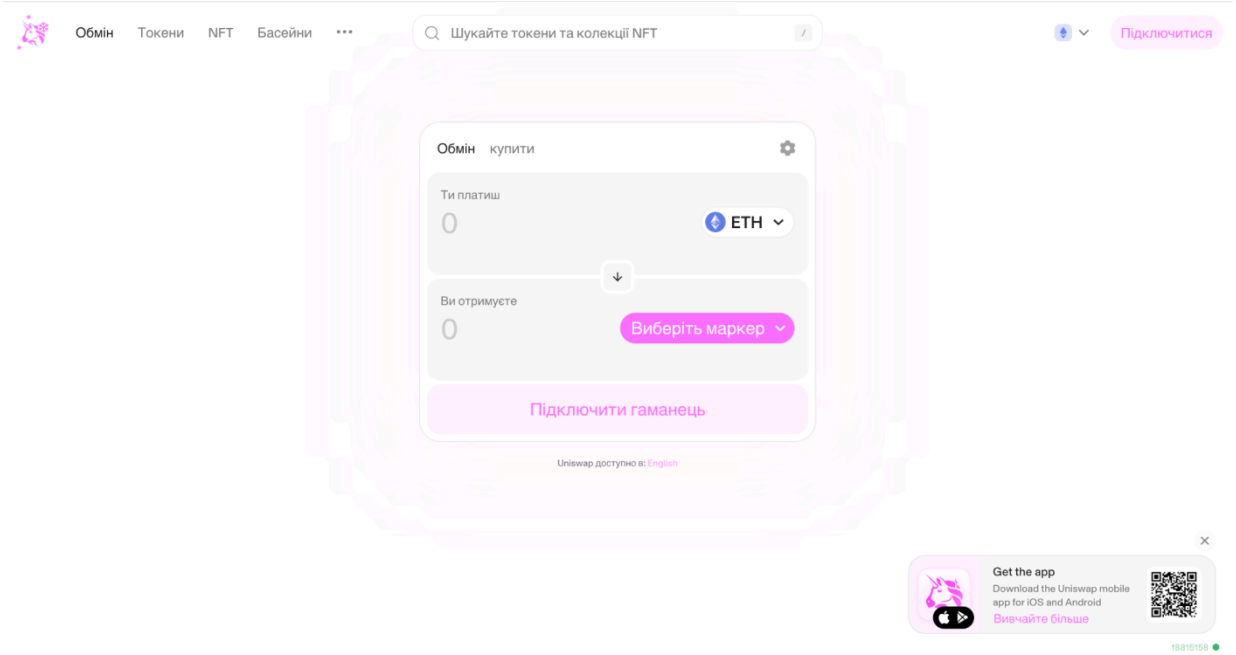


Рис. 1.4 Децентралізований протокол обміну Uniswap

AAVE - це децентралізована фінансова платформа на блокчейні Ethereum, яка надає можливість користувачам використовувати та позичати криптовалютні активи, отримуючи при цьому винагороду за надання ліквідності. Проект пропонує учасникам екосистеми ряд інноваційних фінансових послуг. Користувачі можуть внести свої криптовалютні активи до ліквіднісних пулів AAVE. Депозити генерують власні токени (aTokens), які представляють право власності на депозит. Інші користувачі можуть позичати криптовалютні активи, використовуючи свої депозити як забезпечення. Сума доступної позики залежить від вартості забезпечення та правил платформи. Пули ліквідності визначають процентні ставки для позик та депозитів на основі попиту та пропозиції. Проценти виплачуються депозитарям та вимагаються від позичальників.

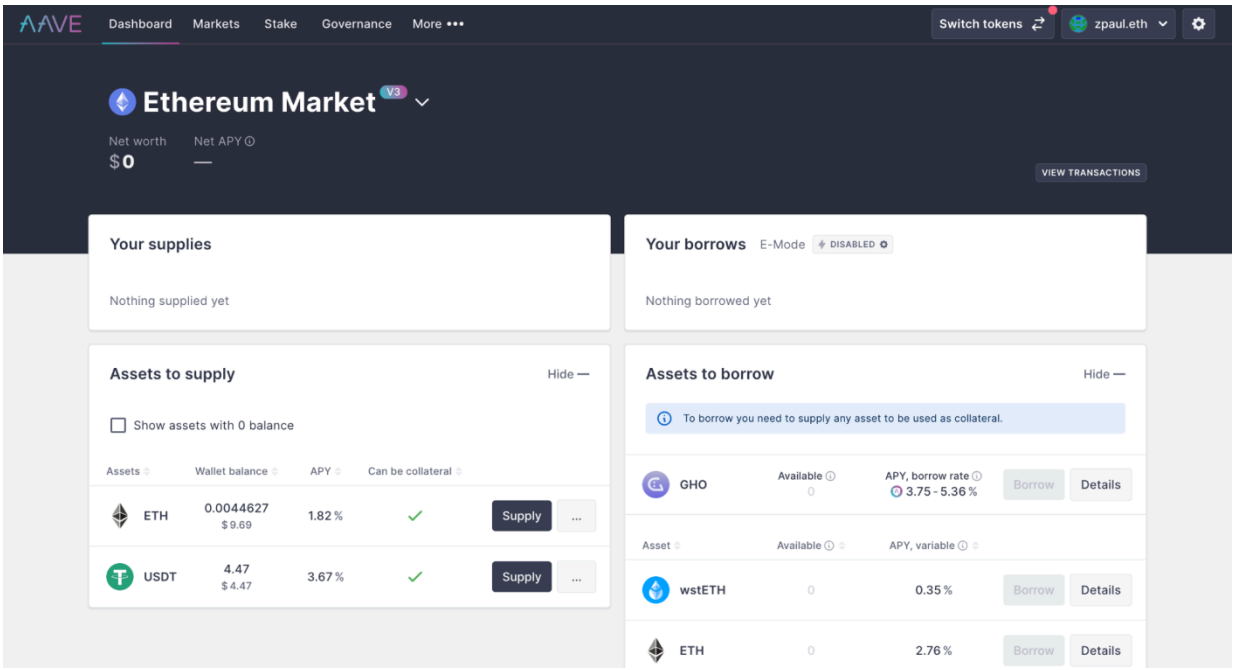


Рис. 1.5 Децентралізована платформа для позик AAVE

Гаманець Metamask - це криптовалютний гаманець і розширення для веб-переглядачів, яке дозволяє користувачам взаємодіяти з децентралізованими додатками на блокчейні Ethereum. Він є одним з найпоширеніших інструментів для зберігання, управління та обміну криптовалютою та токенами. Metamask надає безпечний інтерфейс для створення та управління Ethereum-адресами, а також для здійснення транзакцій у реальному часі. Основною функцією Metamask є надання користувачам можливості зручного доступу до децентралізованих фінансових послуг (DeFi), гри на блокчейні, колекціонування та інших додатків, які використовують технологію Ethereum. Гаманець інтегрується безпосередньо з браузером, що дозволяє легко взаємодіяти з різноманітними додатками, використовуючи один і той самий інтерфейс. Крім того, Metamask надає інструменти для управління безпекою, такі як зберігання приватного ключа локально на пристрої користувача і використання розширеного функціоналу для захисту від шахрайства та атак.

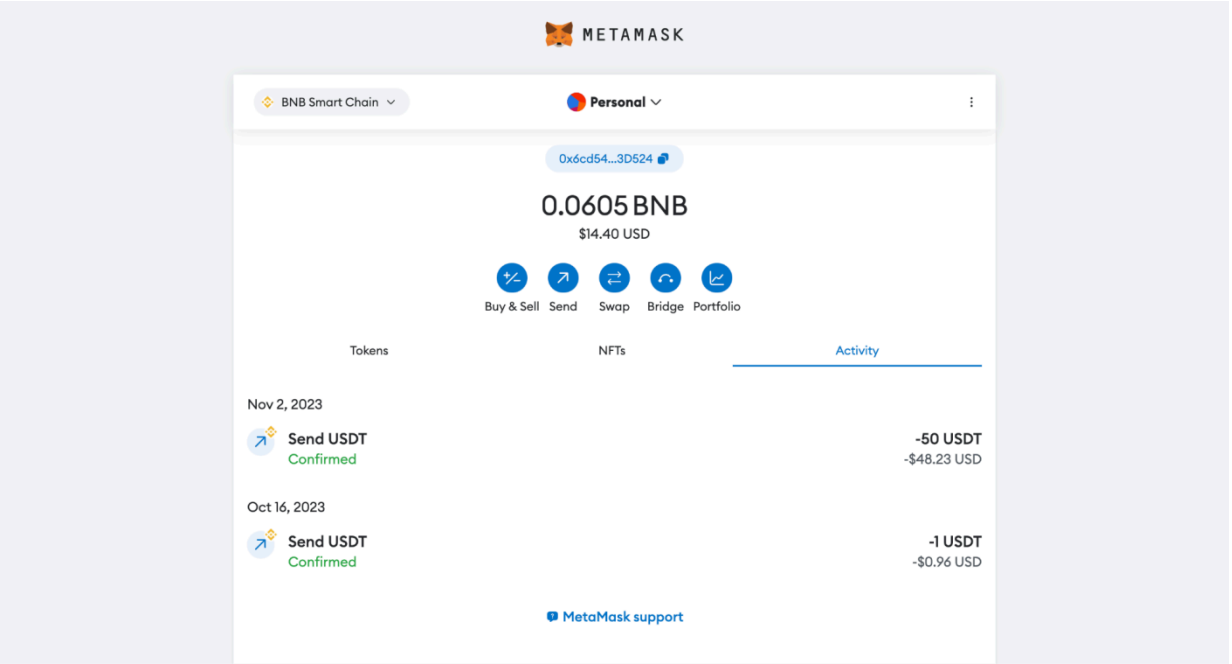


Рис. 1.6 Гаманець Metamask

1.3 Аналіз потенційних ризиків при роботі зі смарт контрактами

Популяризація блокчейнів на базі EVM, а з ними децентралізованих фінансів стимулює появу шахрайських рішень.

На наступному зображенні представлена статистика, яка відображає співвідношення кількості атак відносно рівня застосунку та компонентів системи, завдяки яким було проведено злам.

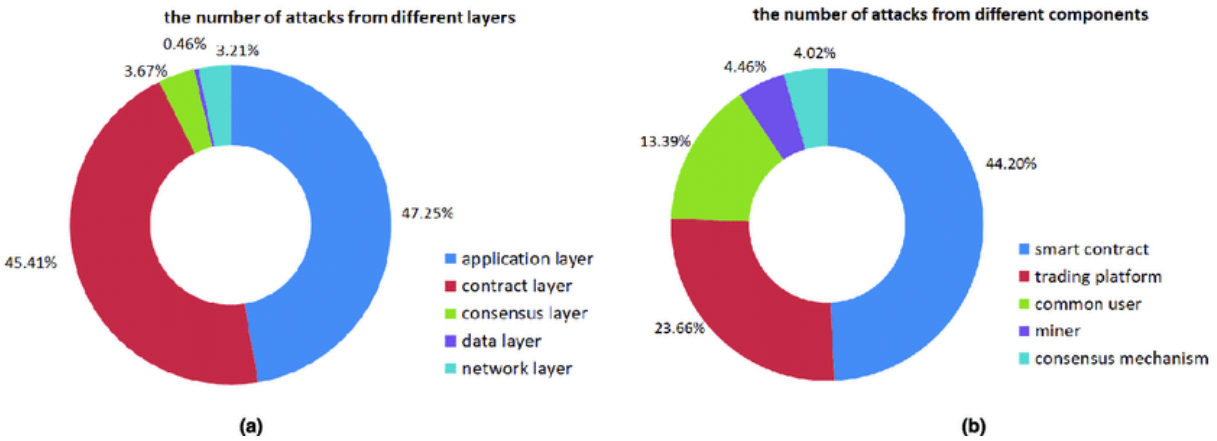


Рис. 1.7 Кількість зламів на основі рівня та частини системи

З представленого графіку видно, що значна частина зламів стосується рівня додатку, а саме частини смарт контрактів.

Статистика шахрайської активності та її результатів

Протягом останніх років децентралізовані протоколи зазнали значних втрат через ряд великих хакерських атак. В першому кварталі 2022 року через хакерські атаки було втрачено понад 682 мільйони доларів. Протягом року загальні втрати від атак на криптофонди та бізнеси перевищили 3,3 мільярда доларів.

За останні роки найбільшими за сумою втрат стали такі децентралізовані протоколи:

— Poly Network: Втрата 611 мільйонів доларів (лютий 2022). Хакер використав незахищені смарт-контракти Poly Network, вразивши три блокчейни: BSC, Polygon і Ethereum.

— Ronin: Втрата понад 615 мільйонів доларів (березень 2022). Атака на Ronin Bridge, який використовується у грі Axie Infinity, дозволила хакерам вкрасти 173,000+ ETH і понад 25 мільйонів USDC.

— Grim Finance: Втрата 30 мільйонів доларів (грудень 2021). Хакер використав атаку повторного входу (reentrancy attack), що призвело до крадіжки токенів Fantom на суму 30 мільйонів доларів.

— Meerkat Finance: Втрата 31 мільйон доларів (березень 2021). Атака сталася одразу після запуску проекту, коли хакери використали внутрішні дозволи для крадіжки 73,000 BNB і 14 мільйонів BUSD.

— Vee Finance: Втрата 35 мільйонів доларів (вересень 2021). Хакери використовували слабкі місця у перевірках змін цін під час торгівлі з плечем на базі Avalanche.

Ці атаки вказують на серйозні проблеми в безпеці смарт контрактів, що включають вразливості в коді, нестачу аудиту безпеки та недосконаліть дизайну систем. Одним із шляхів попередження таких атак є використання кращих практик програмування, які включають ефективні шаблони проектування,

дотримання принципів безпеки та наявність достатнього досвіду для виявлення потенційних помилок та вразливостей.

Нижче представлені дані з сервісу DefiLama, який збирає статистику зламів децентралізованих втрат, наводить категорії зламів та проводить їх детальний аналіз.

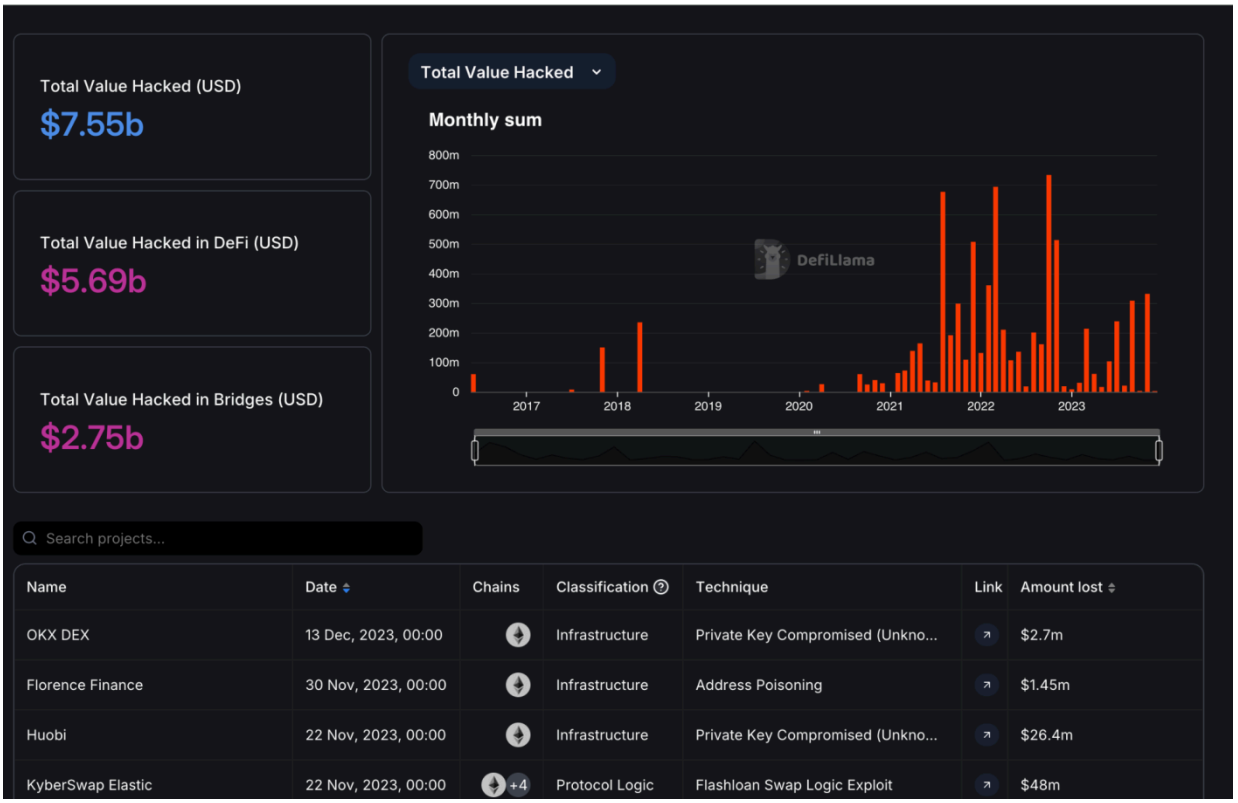


Рис. 1.8 Статистика зламів децентралізованих протоколів від DefiLama

Шахрайські протоколи в свою чергу можуть використовувати безпекові бібліотеки для обмеження доступу користувачів до залучених активів або ж змінювати логіку роботи протокола, вносячи правки в код смарт контракта. Завдати шкоди користувачу протокола можна також маніпуляціями з ціною токена проєкту, а саме витягування ліквідності та великі за об'ємом обміни. Нижче представлений більш розгорнутий опис потенційних сценаріїв застосування.

Сценарій витягування ліквідності

Користувач мережі може за допомогою сервісу Etherscan бачити баланс токенів будь-якого гаманця, головне мати адресу. Користувач знаходить адресу

власника протоколу, бачить у нього на балансі велику суму токенів проєкту. Теоретично, власник протоколу вивести ліквідність з пула децентралізованої біржі і знецінити актив.

```
function removeLiquidity(
        address tokenA,
        address tokenB,
        uint256 liquidity,
        uint256 amountAMin,
        uint256 amountBMin,
        address to,
        uint256 deadline
    ) public virtual override ensure(deadline) returns (uint256 amountA, uint256 amountB) {
        address pair = PancakeLibrary.pairFor(factory, tokenA, tokenB);
        IPancakePair(pair).transferFrom(msg.sender, pair, liquidity); // send liquidity to pair
        (uint256 amount0, uint256 amount1) = IPancakePair(pair).burn(to);
        (address token0, ) = PancakeLibrary.sortTokens(tokenA, tokenB);
        (amountA, amountB) = tokenA == token0 ? (amount0, amount1) : (amount1, amount0);
        require(amountA >= amountAMin, "PancakeRouter: INSUFFICIENT_A_AMOUNT");
        require(amountB >= amountBMin, "PancakeRouter: INSUFFICIENT_B_AMOUNT");
}



function removeLiquidityETH(
        address token,
        uint256 liquidity,
        uint256 amountTokenMin,
        uint256 amountETHMin,
        address to,
        uint256 deadline
          ) public virtual override ensure(deadline) returns (uint256 amountToken, uint256
amountETH) {
        (amountToken, amountETH) = removeLiquidity(
            token,
            WETH,
            liquidity,
            amountTokenMin,
            amountETHMin,
            address(this),
            deadline
        );
        TransferHelper.safeTransfer(token, to, amountToken);
        IWETH(WETH).withdraw(amountETH);
        TransferHelper.safeTransferETH(to, amountETH);
}
```

Сценарій обміну великої кількості токенів

Користувач мережі може за допомогою сервісу Etherscan бачити баланс токенів будь-якого гаманця, головне мати адресу. Користувач знаходить адресу певного аккаунта, бачить у нього на балансі велику суму токенів проєкту. Такий аккунт може зробити обмін активів через децентралізовану біржу на стабільні токени таким чином знецінивши актив.

```
function swapExactTokensForTokens(
        uint256 amountIn,
        uint256 amountOutMin,
        address[] calldata path,
        address to,
        uint256 deadline
    ) external virtual override ensure(deadline) returns (uint256[] memory amounts) {
        amounts = PancakeLibrary.getAmountsOut(factory, amountIn, path);
                    require(amounts[amounts.length - 1] >= amountOutMin, "PancakeRouter:
INSUFFICIENT_OUTPUT_AMOUNT");
        TransferHelper.safeTransferFrom(
            path[0],
            msg.sender,
            PancakeLibrary.pairFor(factory, path[0], path[1]),
            amounts[0]
        );
        _swap(amounts, path, to);
}

function swapTokensForExactETH(
        uint256 amountOut,
        uint256 amountInMax,
        address[] calldata path,
        address to,
        uint256 deadline
    ) external virtual override ensure(deadline) returns (uint256[] memory amounts) {
        require(path[path.length - 1] == WETH, "PancakeRouter: INVALID_PATH");
        amounts = PancakeLibrary.getAmountsIn(factory, amountOut, path);
                                    require(amounts[0] <= amountInMax, "PancakeRouter:
EXCESSIVE_INPUT_AMOUNT");
        TransferHelper.safeTransferFrom(
            path[0],
            msg.sender,
            PancakeLibrary.pairFor(factory, path[0], path[1]),
            amounts[0]
        );
        _swap(amounts, path, address(this));
        IWETH(WETH).withdraw(amounts[amounts.length - 1]);
        TransferHelper.safeTransferETH(to, amounts[amounts.length - 1]);
}
```

Сценарій арбітражу

Окрім знецінення токену, користувач може заробити на зміні курсу токенів в певному пулі ліквідності. При появі транзакції на обмін, користувач може здійснити аналогічний обмін швидше, таким чином придбавши актив до зміни його вартості.

Сценарій зупинки роботи протоколу

Користувач аналізуючи код смарт контракту протоколу помітив, що доданий модуль Pausable, що дозволяє в будь-який момент заблокувати роботу основних функцій протоколу для всіх користувачів за допомогою виклику метода pause(), відповідно заблокувавши доступ до зняття активів, що були переказані на смарт контракт.

```
function _pause() internal virtual whenNotPaused {
    _paused = true;
    emit Paused(_msgSender());
}

function _unpause() internal virtual whenPaused {
    _paused = false;
    emit Unpaused(_msgSender());
}
```

Сценарій зміни власника протоколу

Модуль Ownable дозволяє наділяти аккаунт того, хто розгортав смарт контракт правами власника. За допомогою модифікатора onlyOwner можна помічати методи, що відповідають за системні налаштування. Зазвичай передача повноважень власника протоколу – нечасте явище, тому може бути причною для занепокоєння.

```
function transferOwnership(address newOwner) public virtual onlyOwner {
    if (newOwner == address(0)) {
        revert OwnableInvalidOwner(address(0));
    }
    _transferOwnership(newOwner);
}
```

Сценарій зміни імплементації протоколу

Блокчейни на базі EVM дозволяють розгортання та використання смарт контрактів. Код смарт контрактів після публікації є статичним, його не можна змінити. Часто бувають ситуації, коли помилка в коді призводить до перевипуску протоколу та публікації нових смарт контрактів. Але було розроблене рішення, що має назву proxy pattern, що розділяє стан контракту та його логіку на два різні смарт контракти. Логіку контракту можна замінити, розгорнувши новий смарт контракт та викликавши метод setImplementation(address _contract). Зміна логіки протокола може бути використана не лише для виправлення помилок, а і для зміни логіки задля завдання збитків користувачам.

```
function upgradeAndCall(
      ITransparentUpgradeableProxy proxy,
      address implementation,
      bytes memory data
   ) public payable virtual onlyOwner {
      proxy.upgradeToAndCall{value: msg.value}(implementation, data);
}
```

Сценарій підозрілого використання системних методів

Код смарт контракту протоколу містить методи з модифікатором onlyOwner, що дозволяють власнику протокола виконувати певні операції, що недоступні звичайним користувачам. Зазвичай такі методи застосовуються для налаштування протоколу, проте вони також можуть бути використані для незаконного заволодіння користувацькими активами.

```
function claimByOwner(address user) external onlyOwner {
      uint256 amount = stakes[user];
      stakes[user] = 0;
      token.transfer(msg.sender, amount);
      emit ClaimByOwner(msg.sender, user, amount);
}
```

Сценарій маніпуляції даними оракула

Оракули – це смарт контракти, які задають певне значення ззовні інфраструктури блокчейну(виклики API, дані з датчиків тощо) для використання іншими смарт контрактами. Це може бути як певний ціновий показник, мультиплікатор тощо. Зміна таких показників відбувається за допомогою виклику метода. Зміна значення може значно повпливати на роботу системи, яка посилається на смарт контракт оракула.

```
function fulfill(
    bytes32 _requestId,
    uint256 _volume
  ) public recordChainlinkFulfillment(_requestId) {
    emit RequestVolume(_requestId, _volume);
    volume = _volume;
  }
```

Загалом було розлянуто лише основні сценарії, за допомогою яких шахраї можуть отримати вигоду з завичайних користувачів децентралізованих протоколів. Для запобігання подібним кейсам користувач повинен досліджувати проєкт, його принципи роботи, механізми, кодову базу тощо. Це допоможе побачити та зрозуміти можливі підводні камені та запобігти втратам.

2АНАЛІЗ ПОПЕРЕДНІХ ДОСЛІДЖЕНЬ ТА ФОРМУВАННЯ МОДЕЛІ СИСТЕМИ

Під час визначення проблеми та постановки завдання було проаналізовано низку досліжень та публікацій пов’язаних з захистом децентралізованих систем. Більшість досліджень спрямовані безпосередньо на захист протоколів, але не надають користувачу інструментів індивідуального захисту.

2.1 Аналіз досліджень на тему безпеки в блокчейні

Дослідження “Management and Monitoring of Blockchain Systems”

В дослідженні[1] представлене бачення моніторингової системВисновки на основі розглянутих дослідженьи, що включена в архітектуру блокчейн нод та збирає логи транзакцій, індексує їх, проводить обробку, визначаючи спільні ознаки, патерни, що можуть в подальшому допомогти визначити аномалії, що можуть виникнути при виконанні транзакції.

Автори пропонують структуру моніторингу блокчейну, що складається з:

— Агенти моніторингу, розгорнуті на кожному вузлі

— Механізму збору логів

— Еластичний кластер вузлів для обробки та індексації даних журналів

— Платформа візуалізації для надання статистики

Підсумовуючи, в тексті стверджується, що хоча блокчейн забезпечує внутрішню безпеку, моніторинг може забезпечити додаткові переваги та видимість продуктивності та стану блокчейн-систем. Запропонована система має на меті забезпечити ефективний моніторинг та управління мережею блокчейн.

Відстежуючи ключові показники, такі як продуктивність вузлів, пропускна здатність транзакцій і затримки, адміністратори можуть виявити вузькі місця і оптимізувати мережу. Сповіщення, засновані на цих показниках, можуть своєчасно сповіщати операційні команди про потенційні проблеми.

Моніторинг подій, пов'язаних з безпекою, таких як несанкціоновані транзакції, спроби подвійних витрат і збої консенсусу, може виявити зловмисну активність і атаки. Адміністратори можуть вжити відповідних заходів для зменшення ризиків та збитків. Журнали від агентів моніторингу також можна аналізувати ретроспективно для розслідування минулих інцидентів.

Платформа візуалізації дозволяє операційним командам і командам безпеки отримати цілісне уявлення про всю мережу блокчейн. Вони можуть швидко ідентифікувати вузли з ненормальною поведінкою, проблемні транзакції та інші аномалії. Такий огляд дозволяє діагностувати і вирішувати проблеми більш ефективно.

Система моніторингу та управління, яка збирає і аналізує дані з усієї мережі блокчейн, має потенціал для значного підвищення безпеки, продуктивності і надійності корпоративних розгортань блокчейн. Запропоноване в тексті рішення має на меті забезпечити організаціям видимість і контроль, необхідні для ефективної роботи блокчейн-систем в масштабах.

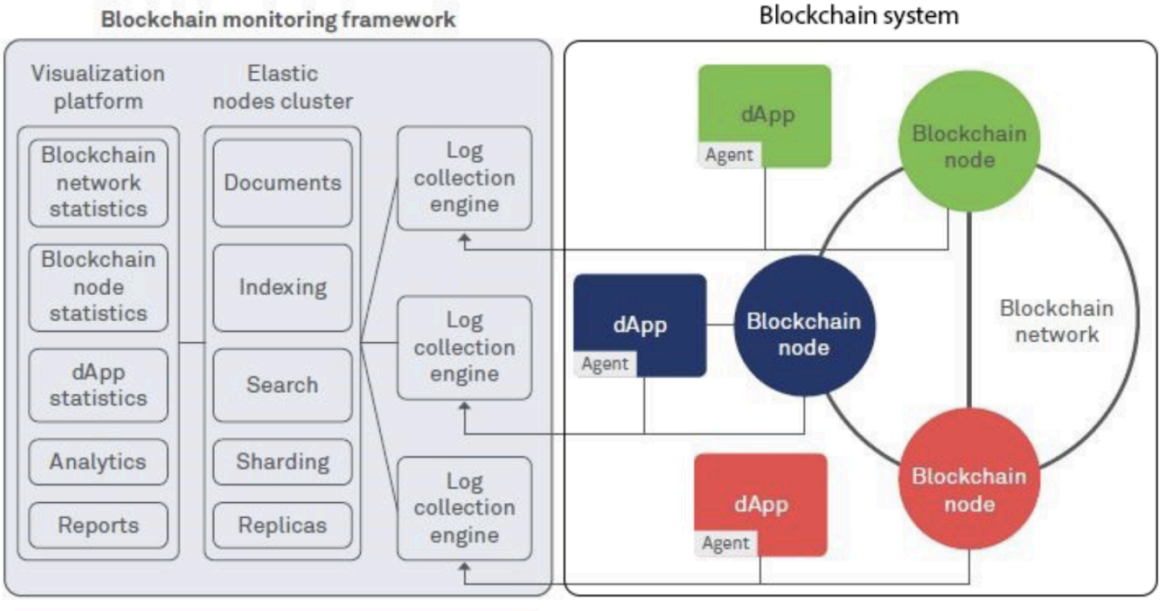


Рис. 2.1 Схема запропонованої системи моніторингу на основі блокчейн нод

Дослідження не розкриває процесу аналізу даних транзакцій, проте застосування такого рішення на рівні нод потребуватиме значної кількості технічних або людських ресурсів для дослідження визначених патернів та їх

потенційного впливу на протоколи. Кількість проєктів на основі смарт контрактів збільшується щодня, кожен протокол має свої унікальні рішення, що ускладнює роботу запропонованої системи. Відсутність логування в коді протоколу може унеможливити роботу системи для конкретного рішення.

Дослідження “BLOCKEYE: Hunting For DeFi Attacks on Blockchain”

Дослідження[2] пропонує аналіз коду протоколів та визначення залежностей на так званих оракулах – смарт контрактах, що надають протоколу поточне значення ціни актива, облікової ставки або іншого системного коефіцієнта. Маніпуляція такими значеннями може відбуватись як вручну власниками протоколу, так і змінюватись залежно від інших параметрів системи. Система налаштовує моніторинг транзакцій, які можуть скористатись цією залежністю, збирає дані про вартість транзакцій та прибуток отриманий внаслідок операції. Якщо співвідношення є високим, транзакція помічається підозрілою та аналізується детальніше. Система також передбачає аналіз послідовності транзакцій, що можуть бути включені в різні блоки.

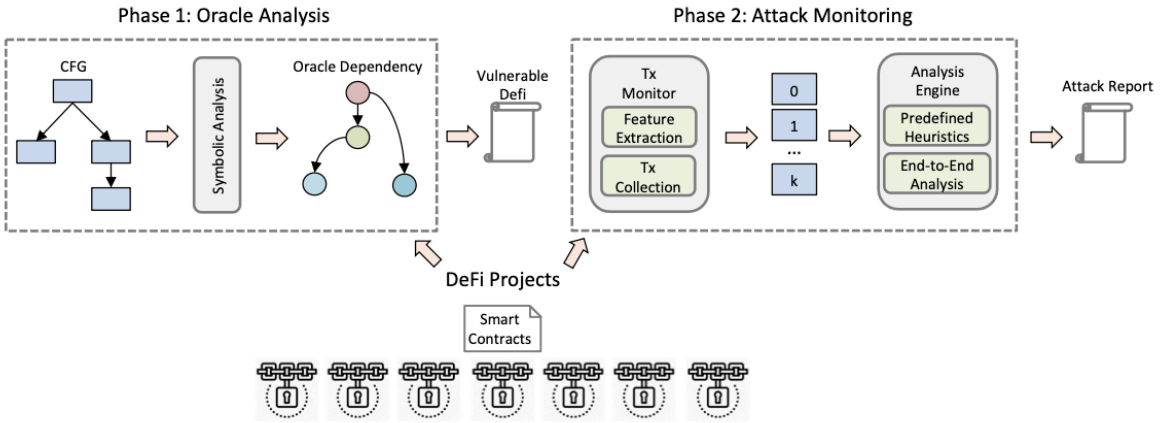


Рис. 2.2 Схематичне зображення процесу роботи BLOCKEYE

Продукт виявляє потенційно вразливі DeFi-проєкти за допомогою автоматичного процесу аналізу безпеки, який виконує символічні міркування над потоком даних про важливі стани сервісу, такі як ціна активу, і перевіряє, чи можуть вони бути піддані зовнішнім маніпуляціям.

BLOCKEYE працює в два етапи. Спочатку він виконує символьний аналіз смарт-контрактів, щоб виявити проєкти, які залежать від цінових оракулів інших DeFi-проектів. Ці залежні від оракулів проєкти піддаються ризику маніпуляцій.

По-друге, система відстежує транзакції вразливих додатків в режимі реального часу для виявлення потенційних атак. Він збирає транзакції, пов'язані з цільовим проєктом. Потім він перевіряє, чи не порушують транзакції попередньо визначені правила, наприклад, чи не генерують вони аномально великий прибуток за короткий час. Це свідчить про атаку.

Автори застосували розроблену систему до кількох популярних проєктів і виявили потенційні атаки, про які раніше не повідомлялося. В експериментах BLOCKEYE успішно ідентифікував всі оракул-залежні проєкти без помилкових спрацьовувань, перевершивши існуючі інструменти. Система також відслідковує реальні арбітражні транзакції.

Система пропонує алгоритм визначення специфічного типу вразливості та аналізує транзакції, які можуть повпливати на роботу системи за рахунок даної вразливості. Окрім цього, система не може попередити транзакцію, яка вплине на протокол, а лише помічає транзакцію для подальшого дослідження та пошуку рішення. Рішення спрямоване на протоколи конкретного типу і не може бути повноцінно застосоване звичайним користувачем мережі.

Дослідження “Підвищення рівня безпеки смарт контрактів в мережі Ethereum від шахрайства за рахунок використання реверсивних токенів”

Робота[3] пропонує підвищення рівня користувацької безпеки за рахунок впровадження нового стандарту токенів, що дозволяє повертати кошти, які були привласнені внаслідок шахрайських дій, заморожувати їх в разі підозрілої активності власника активів тощо. Рішення впроваджується як надбудова над токенами стандартів ERC-20, ERC-721.

У тексті обговорюється проблема втрати коштів користувачами мережі Ethereum через шахрайство та пропонується використовувати реверсивні токени

для підвищення безпеки та зменшення втрат. Поточні протоколи Ethereum мають вразливості, які дозволяють здійснювати незаконні транзакції. Незмінність даних у блокчейні є перевагою, але вона також означає, що транзакції не можуть бути скасовані або видалені.

Автори пропонують наступні рішення для нейтралізації можливих наслідків шкідливих транзакцій.

— Зворотні запити на транзакції, коли жертви подають запит на заморожування підозрілих активів разом із доказами та заставою.
— Арбітражний процес, коли судді розглядають докази і приймають рішення про заморожування або відхилення запиту.
— Алгоритм заморожування токенів ERC-20, який відстежує вкрадені кошти через графіки транзакцій і заморожує кошти на підозрілих адресах. Не всі втрачені кошти можуть бути гарантовано повернуті.
— Токени ERC-721 також зберігатимуть інформацію про конкретні токени, історію володіння та заморожений статус.

Запропонований підхід дозволяє зворотну сумісність, але вимагає виконання контрактів. В якості альтернативи, стандартні контракти ERC20/ERC721 можуть бути розширені за допомогою проксі-контракту, хоча це вимагає додаткового вивчення.

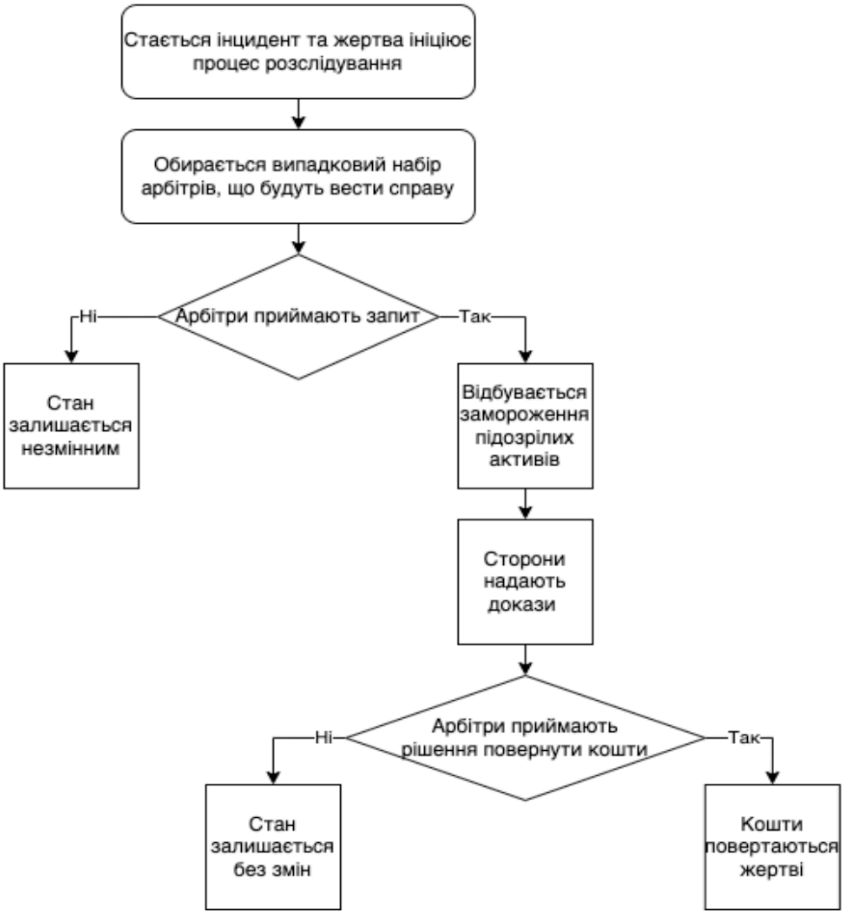


Рис. 2.3 Алгоритм роботи реверсивних токенів

Реверсивні токени токени є багатообіцяючим підходом до відновлення втрачених коштів і підвищення безпеки. Однак необхідні подальші дослідження для оптимізації алгоритму заморожування, оцінки його впливу на мережу і визначення складності заміни існуючих стандартів на реверсивні версії.

Проаналізоване рішення створює можливість повернути викрадені кошти, що може нейтралізувати наслідки шахрайства та вберегти користувача від потенційних збитків. Проте застосування такого стандарту буде можливо лише для нових токенів що випускаються, а більшість уже випущених токенів, в тому числі стабільні, все так само матимуть звичайний функціонал та не можуть бути перевипущені.

Дослідження “S-gram: Towards Semantic-Aware Security Auditing for Ethereum Smart Contracts”

Автори статті вказують на проблеми з аудитом безпеки[4] смарт контрактів та висувають три основні виклики: моделювання механізмів Ethereum, кодування доступу до сховища, і виявлення чутливості до потоку. Автори представляють власний метод – S-gram, який є технікою аудиту безпеки для контрактів мовою Solidity в Ethereum. S-gram використовує моделювання мовою на основі N-gram та легкого статичного аналізу контракту. Вони стверджують, що S-gram дозволяє проводити швидкий та масштабований аудит безпеки, оскільки модель навчається лише один раз, а сам аудит обмежується розрахунком ймовірностей.

S-gram – це семантично-орієнтована методика аудиту безпеки, запропонована для смарт-контрактів Ethereum. Вона спрямована на усунення обмежень існуючих методів аудиту безпеки. Ключова ідея S-gram полягає в поєднанні моделювання мовою N-gram і легкого статичного аналізу контрактів. Він може вивчати статистичні закономірності токенів смарт-контрактів, а також захоплювати високорівневу семантику.

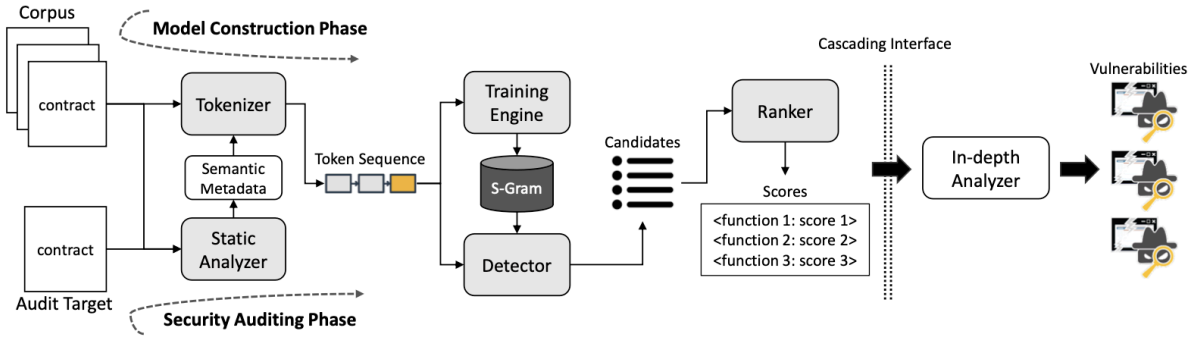


Рис. 2.4 Принцип роботи фреймворку S-gram

S-gram працює в два етапи: побудова моделі та аудит безпеки. На етапі побудови моделі вона виконує статичний аналіз для створення семантичних метаданих для корпусу смарт-контрактів. Це використовується для навчання мовної моделі S-gram. На етапі аудиту безпеки S-gram аналізує цільовий контракт і використовує мовну модель для виявлення нерегулярних послідовностей токенів,

які вказують на потенційні вразливості. Він також може ранжувати функції на основі їх "оцінки безпеки" для оптимізації інструментів поглибленого аналізу.

S-gram кодує доступ до сховища та визначає операції, чутливі до потоку, щоб отримати більше семантики. Під час токенізації вона генерує токени на основі типів вузлів AST. Оцінка показує, що S-gram досягає понад 90% точності у виявленні потенційних вразливостей. Вона також виявила раніше невідомі проблеми та покращила ефективність фаззера Solidity.

Автоматизація аудиту смарт контрактів може пришвидшити та зробити дешевшим даний процес. Проте машинне навчання потребуватиме валідних даних для навчання моделі для визначення можлививх безпекових вразливостей.

Дослідження “Ethereum smart contract security research: survey and future research opportunities”

Автори статті розглядають 7 викликів[5] та майбутні напрямки досліджень у галузі безпеки розумних контрактів на блокчейні. Декілька ключових викликів та можливих рішень включають:

— Виявлення аномальних контрактів: Важкість розрізнення між нормальними смарт контрактами та контрактами, які використовуються для злочинної діяльності. Запропоновано можливе рішення через використання машинного навчання для виявлення відмінностей.

— Виявлення високих витрат: Проблеми з виявленням витрат, зокрема визначення витратних шаблонів у контрактах. Запропоновано використання ефективних шаблонів для зменшення витрат.

— Розвиток безпечної мови програмування: Потреба у розробці мов програмування, яка забезпечує безпеку смарт контрактів без впливу на їхню продуктивність.

— Виявлення вразливостей смарт контрактів: Розробка інструментів для ідентифікації та використання вразливостей у смарт контрактах.

— Безпечне оновлення контрактів: Розробка методів безпечного оновлення смарт контрактів без впливу на безпеку блокчейну.

— Семантика EVM та мережі блокчейну: Складність визначення семантики EVM та симуляції системи. Запропоновано покращення екосистеми блокчейну для зменшення цієї проблеми.

— Захист приватних даних в смарт контрактах: Розробка ефективних підходів, таких як гомоморфне шифрування, для захисту приватних даних в смарт контрактах.

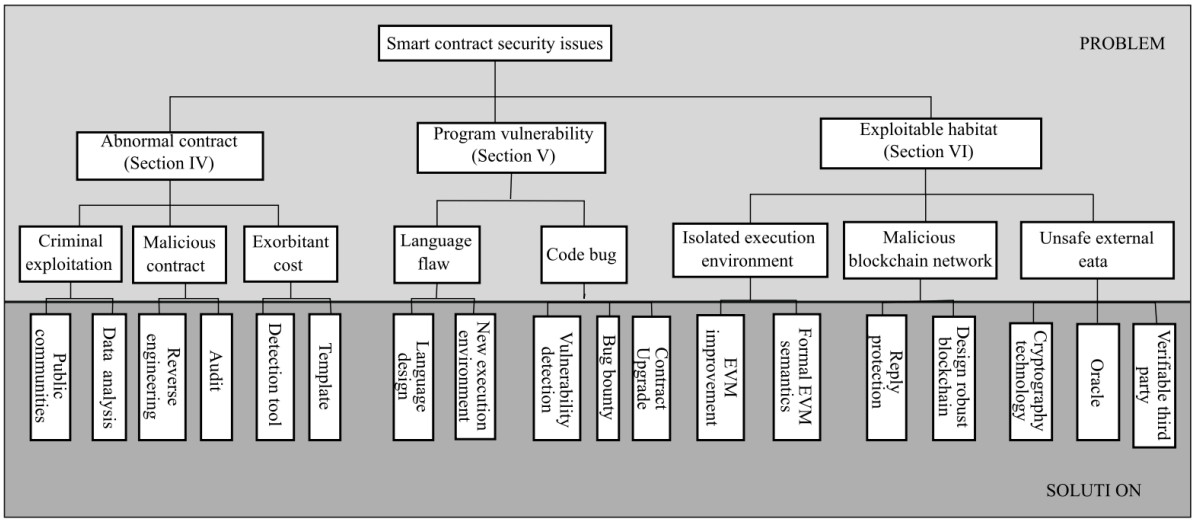


Рис. 2.5 Потенційні загрози та способи їх вирішення

У цій статті проаналізовано існуючі дослідження безпеки смарт-контрактів Ethereum за період з 2015 по 2019 рік. Автори класифікують проблеми безпеки смарт-контрактів за трьома напрямками: аномальні контракти, програмні вразливості та середовище, яке можна експлуатувати.

Щодо аномальних контрактів, автори обговорюють злочинну експлуатацію смарт-контрактів, зловмисні контракти та дорогі контракти, які витрачають газ. Вони розглядають різні методи виявлення та пом'якшення цих проблем.

Що стосується програмних вразливостей, автори розглядають мовні недоліки в мовах програмування смарт-контрактів і помилки в коді, допущені розробниками. Вони обговорюють інструменти для виявлення вразливостей статично і динамічно. Також розглядаються програми оновлення контрактів та винагороди за виправлення помилок.

Щодо середовища, яке можна експлуатувати, автори обговорюють проблеми, пов'язані з ізольованим середовищем виконання EVM, мережею Blockchain та небезпечними зовнішніми даними. Вони аналізують методи генерації випадкових чисел і надання даних від оракулів.

Автори статті вдало визначають можливі загрози, проте не надають конкретних рішень для їх попередження чи нейтралізації.

Дослідження “ZEUS: Analyzing Safety of Smart Contracts”

ZEUS – це фреймворк для аналізу безпеки смарт-контрактів Ethereum. У статті описано ZEUS та проведено його оцінку на більш ніж 22 000 реальних смарт контрактів Solidity. Система спочатку переводить контракти Solidity в біт-код LLVM. Потім він вставляє умови перевірки, засновані на заданих користувачем політиках коректності та справедливості. Система використовує абстрактну інтерпретацію і перевірку символічної моделі для верифікації контрактів.

Оцінка показала, що понад 94% контрактів (на суму понад 500 мільйонів доларів) містили помилки, пов'язані з повторним входом, неперевіреними транзакціями, цілочисельними переповненнями та іншими проблемами. Аналізатор мав нуль помилкових спрацьовувань і низький рівень помилкових спрацьовувань. Він зміг перевірити 97% контрактів протягом 1 хвилини, в той час як існуючий інструмент закінчився або не зміг надати результати для 46% контрактів.

Система може бути поширена на інші блокчейни та інструменти верифікації. У статті демонструється перенесення контракту на Hyperledger Fabric та його перевірка за допомогою верифікатора SMACK.

Таким чином, проєкт представляє ефективний фреймворк для автоматичного аналізу смарт контрактів на предмет безпеки та справедливості. Його надійність, масштабованість і низький рівень помилкових спрацьовувань роблять його корисним інструментом для перевірки реальних смарт-контрактів.

2.2 Висновки на основі розглянутих досліджень

Розглянуті дослідження спрямовані на підвищення рівня безпеки протоколів на основі смарт контрактів. Нижче наведено перелік основних рішень, що були розглянуті:

— Система статичного аналізу
— Аудит смарт контрактів
— Моніторингова система в реальному часі на основі машинного навчання
— Моніторингова система на рівні блокчейн вузлів
— Стандарт реверсивних токенів

Розглянуті рішення безумовно мають як свої переваги, так і недоліки: серед яких людський фактор при аудиті, навчання моделей для визначення атак та статичного аналізу. Розроблені системи спрямовані на безпосередній захист децентралізованих протоколів.

Підбиваючи підсумки аналізу досліджень, можна сказати що представлені рішення лише опосередковано впливають на безпеку користувача і можуть бути застосовані лише за ініціативи власників протоколів. Крім цього, рішення надають захист від специфічних вразливостей, відповідно не можуть бути застосовані для більш широкого спектру вразливостей.

2.3 Порівняльна характеристика проаналізованих безпекових рішень

Аналіз та забезпечення безпеки смарт-контрактів є невід'ємною частиною блокчейн розробки, особливо в умовах зростаючого інтересу до децентралізованих фінансових та інших платформ. В процесі аналізу наукових досліджень виділено чотири основні варіанти безпекових рішень: стаичний аналіз використання безпекових бібліотек OpenZeppelin, проведення аудиту смарт-контрактів та використання моніторингових систем для пошуку підозрілих транзакцій. Переваги та недоліки рішень представлені у таблиці.

Таблиця 2.1

Порівняння характеристик безпекових рішень

Характеристика	**Безпекові бібліотеки OpenZeppelin**	**Статичний аналіз**	**Аудит смарт-контрактів**	**Система моніторингу підозрілих транзакцій**
Визначення	OpenZeppelin надає набір готових безпекових бібліотек для розробки смарт-контрактів.	Аналізує код на наявність неточностей та базових передумов для проведення атаки	Аудитори вивчають код смарт-контракту для виявлення потенційних уразливостей та помилок.	Система аналізує транзакції на предмет підозрілих дій або викликів.
Рівень автоматизації	Високий рівень автоматизація завдяки готовим бібліотекам.	Високий рівень автоматизації, проте потребує добре прописаних умов або натренованих моделей	Потребує активної участі аудиторів та вручну проведеного аналізу коду.	Автоматично виявляє аномальні транзакції на основі певних правил.
Час виконання	Відразу доступні готові рішення, що спрощує розробку.	Залежить від розміру та складності коду контракту	Залежить від розміру та складності коду контракту.	Постійно моніторить транзакції в реальному часі.
Спрощення розробки	Забезпечує готові рішення для популярних проблем безпеки.	Ускладнює розробку за рахунок додаткових рекомендацій щодо виправлення помилок	Ручна робота з кодом, можливість впровадження виправлень помилок під час аудиту.	Відсутнє пряме впливання на розробку, рекомендації можуть бути внесені після аудиту.
Складність впровадження	Легко впроваджується за допомогою імпорту бібліотек.	Часто вбудовується в середовище розробки	Залежить від розміру та складності контракту, може бути часомісткою і ресурсозатратною.	Відносно легке впровадження, але не надає безпосереднього захисту від уразливостей.

Продовження таблиці 2.1

Порівняння характеристик безпекових рішень

Характеристика	**Безпекові бібліотеки OpenZeppelin**	**Статичний аналіз**	**Аудит смарт-контрактів**	**Система моніторингу підозрілих транзакцій**
Практичність використання	Ідеально підходить для швидкого розгортання проекту з високим рівнем безпеки.	Застосовується під час розробки децентралізованої частини проєкту	Ефективний, але може бути витратний у великих проектах.	Забезпечує практичності завдяки постійному моніторингу.
Вартість	Безкоштовні бібліотеки OpenZeppelin, але можливі витрати на їх адаптацію та налагодження.	Залежить від класу системи та проєкту	Вартість аудиту залежить від обсягу робіт і розміру контракту.	Вартість послуг якісного блокчейн провайдера
Спрямування	Захист протоколу. Не може бути використаний користувачем.	Випралення вразливостей та недоліків на етапі розробки системи	Захист протоколу і користувача. Не може бути використаний користувачем.	Захист протоколу і користувача. Може бути використана користувачем

Статичний аналіз: включає в себе оцінку коду контракту без його виконання, щоб виявити потенційні вразливості, помилки та неефективні практики кодування. Статичний аналіз допомагає ідентифікувати проблеми, такі як переповнення цілих чисел, проблеми з доступом до приватних даних, реентрантні атаки та інші вразливості, характерні для смартконтрактів. Статичний аналіз дозволяє розробникам виявляти та виправляти помилки до того, як контракти будуть розгорнуті на блокчейні.

Бібліотеки OpenZeppelin: пропонують ефективний та швидкий спосіб забезпечити базовий рівень безпеки, зокрема для популярних сценаріїв. Однак вони можуть вимагати додаткової настройки для врахування конкретних вимог проекту.

Аудит смарт-контрактів: аудит є важливим етапом для виявлення та усунення потенційних уразливостей, але вимагає витрат часу та ресурсів. При цьому він надає високий рівень індивідуалізації та адаптації до конкретних потреб проекту.

Моніторинг підозрілих транзакцій: системи моніторингу дозволяють виявляти підозрілі дії в реальному часі, що робить їх ефективним інструментом для виявлення шахраїв та кіберзлочинців. Запропоноване в рамках наукового рішення дослідження дозволить користувачу самостійно впливати на свій рівень безпеки шляхом встановлення моніторингу на визначені типи тригер транзакцій.

2.4 Побудова математичної моделі системи моніторингу

Математична модель системи моніторингу підозрілих транзакцій базується на критичному принципі ефективності обробки транзакцій. Згідно з цією моделлю, часовий проміжок, необхідний для обробки та аналізу транзакції системою, повинен бути значно меншим, ніж час, який триває процес підтвердження транзакції в блокчейні. Цей принцип є фундаментальним для забезпечення здатності користувачів попереджати або нейтралізувати потенційно шкідливі транзакції.

$$\begin{cases} t_c = t_i - t_p \\ t_{pr} = t_f - t_d \\ t_{pr} < t_c \\ t_{pr} \to \min \end{cases} \quad (2.1)$$

де,

t_c – час підтвердження транзакції

t_p – час додавання транзакції в пул

t_i – час додавання транзакції в блок

t_{pr} – час обробки транзакції

t_d – час фіксування транзакції системою

t_f – час завершення обробки транзакції

Нижче представлене схематичне зображення часового проміжку процесу підтвердження транзакції та її обробки системою.

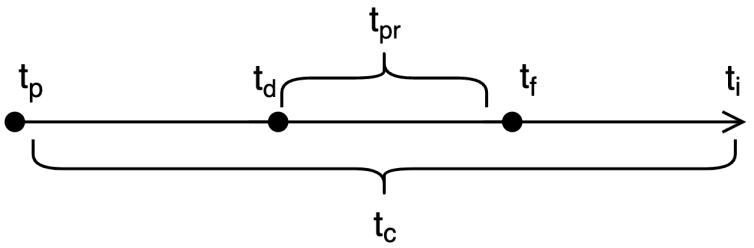


Рис. 2.6 Зображення часового проміжку процесу підтвердження та обробки транзакції

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОНІТОРИНГОВОЇ СИСТЕМИ ПІДОЗРІЛИХ ТРАНЗАКЦІЙ

3.1 Алгоритм роботи розробленої системи

Користувач може підготуватись до можливих сценаріїв атаки створивши запит на стеження, вказавши необхідні параметри, а саме адресу власника протоколу, адресу протокола, сигнатуру метода, виклик якого буде відстежуватись та, в разі потреби, параметри виклику для фільтрації специфічних значень.

Після створення запиту система підписується на оновлення транзакцій, що потрапляють в так званий мемпул – список, з якого валідатори мережі вибирають транзакції для подальшого включення в блок. Ці транзакції ще не оброблені, тому вони можуть бути випереджені за рахунок внесення більшої плати, що є пріоритетом при виборі транзакцій валідатором.

При появі транзакції, яка співпадає за відправником чи отримувачем в мережі – система починає обробку транзакції, а саме перевірку транзакції за вказаними користувачем критеріями. В разі відсутності збігу за певним параметром – транзакція відкидається. Якщо ж всі параметри транзакції задовольняють запит користувача – користувачу відправляється сповіщення через визначений канал комунікації.

Графічне зображення алгоритму роботи системи представлено на рисунку 3.1.

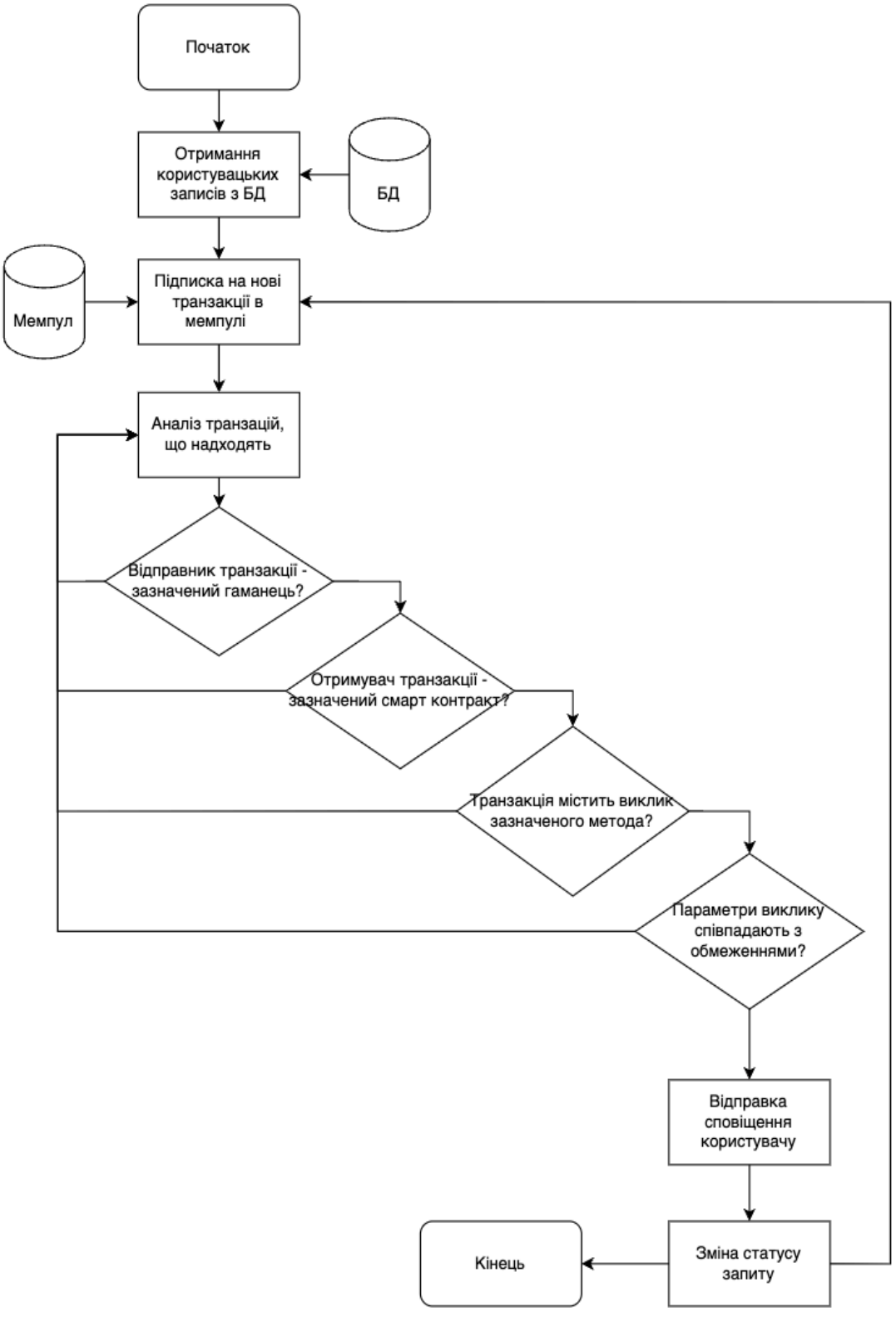


Рис. 3.1 Алгоритм роботи моніторингової системи підозрілих транзакцій

Структурна схема запропонованого рішення представлена на рисунку 3.2.

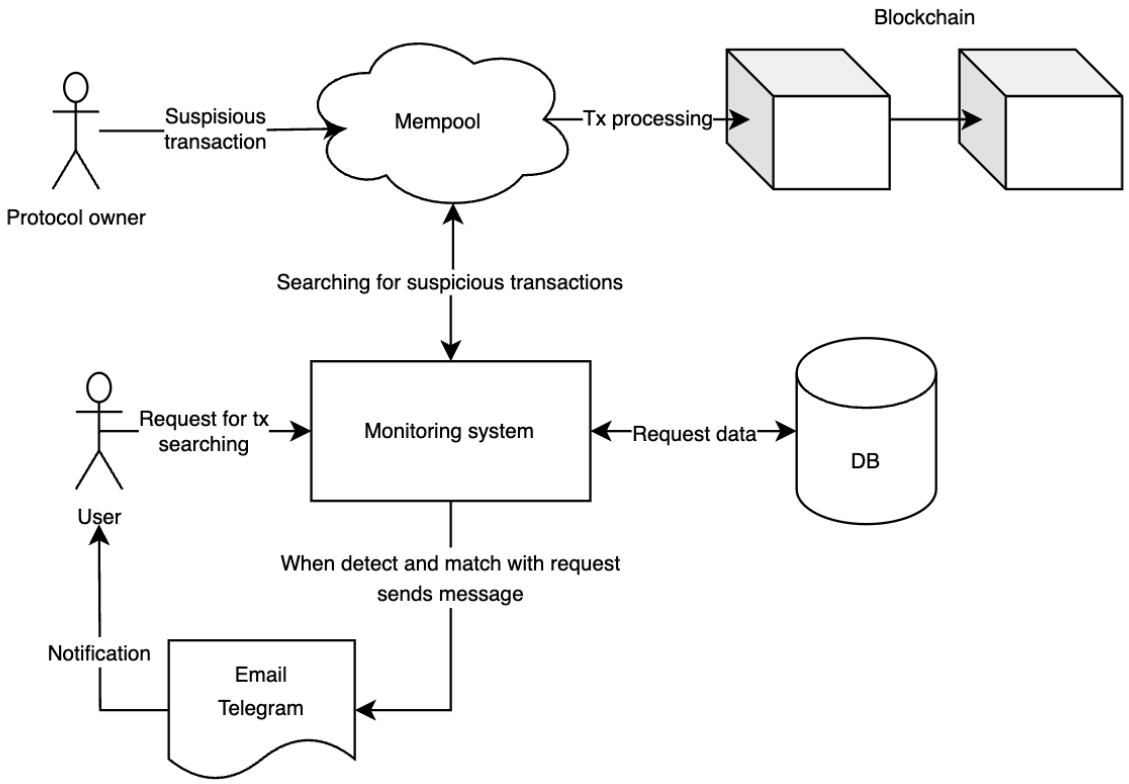


Рис. 3.2 Структурна схема моніторингової системи підозрілих транзакцій

3.2 Функціональні та нефункціональні вимоги до системи

Моніторингова система підозрілих транзакцій, що розробляється, має відповідати ряду функціональних вимог для ефективної та точної роботи. Ось детальний опис цих вимог:

— **Створення запитів на відслідковування транзакцій:** Система повинна дозволяти користувачам легко створювати та конфігурувати запити для моніторингу транзакцій. Це включає в себе можливість вказувати специфічні параметри, такі як ідентифікатори відправника/одержувача, суми транзакцій, теги або ключові слова, пов'язані з транзакцією.

— **Додавання нових типів транзакцій:** Система має бути гнучкою та здатною адаптуватися до нових типів транзакцій, які можуть з'явитися в результаті еволюції блокчейн-технологій. Це включає можливість інтеграції нових шаблонів та правил для ідентифікації та обробки цих транзакцій.

— **Фільтрація транзакцій за заданими параметрами виклику:** Важливою вимогою є можливість фільтрувати транзакції на основі заданих користувачами параметрів. Це має включати детальний аналіз та перевірку кожної транзакції, щоб визначити, чи вона відповідає встановленим критеріям.
— **Робота на різних блокчейнах, що підтримують EVM (Ethereum Virtual Machine):** Система має бути сумісною з різними блокчейнами, що базуються на EVM. Це означає можливість інтеграції та взаємодії з різними блокчейн-мережами, що дозволяє розширити сферу моніторингу та аналізу.
— **Відправка сповіщень при знаходженні транзакції, що задовольняє вказані умови:** Ключовою функцією системи є здатність сповіщати користувачів про виявлені транзакції, що відповідають їхнім запитам. Система має надавати швидкі та надійні сповіщення через різні канали, такі як електронна пошта, SMS, мобільні додатки, або інші інтегровані комунікаційні платформи.

Для моніторингової системи підозрілих транзакцій важливо визначити не лише функціональні, а й нефункціональні вимоги, які забезпечують її надійність, ефективність та безпеку. Ось деталізація цих вимог:

— **Час обробки транзакції системою:** Час, необхідний для обробки транзакції системою, має бути мінімальним і, ідеально, меншим за час, необхідний для включення транзакції в блок на блокчейні. Це означає, що система має бути оптимізована для швидкого виявлення та обробки транзакцій, використовуючи ефективні алгоритми та високопродуктивну інфраструктуру.
— **Можливість інтеграції з іншими системами:** Система має бути сумісною та гнучкою для інтеграції з різними фінансовими та аналітичними системами. Це включає наявність API для зовнішньої інтеграції, підтримку різних форматів даних та протоколів комунікації.

— **Модульна архітектура:** Система повинна мати модульну архітектуру, яка дозволяє легко розширювати функціональність та виправляти помилки. Модульність сприяє зручності розробки та тестування, а також дозволяє швидко адаптувати систему до змінних вимог ринку та технологій.
— **Високий рівень аутентифікації та авторизації:** Система має забезпечувати надійні механізми аутентифікації та авторизації, щоб обмежити несанкціонований доступ до даних та функцій системи. Це може включати багаторівневу аутентифікацію, шифрування даних, а також ретельне керування правами доступу користувачів.
— **Масштабованість системи:** Система має бути спроектована таким чином, щоб вона могла ефективно працювати при збільшенні обсягу обробки транзакцій та кількості користувачів. Це включає використання розподілених систем, оптимізацію баз даних, та можливість горизонтального масштабування.

Нефункціональні вимоги мають забезпечити стабільну, безпечну, та високопродуктивну роботу системи. Важливо також забезпечити високу стійкість до помилок та здатність до швидкого відновлення після збоїв або перебоїв у роботі.

3.3 Використані програмні засоби в процесі розробки системи

Основними засобами, що були використані під час розробки програмної системи є фреймворки Nest.js, grammy.js. Для взаємодії з блокчейном, а саме отримання, обробки транзакцій була використана бібліотека ethers.js. Створені запити зберігаються в базі даних PostgreSQL.

Nest.js – модульний фреймворк для розробки серверних застосунків на базі Node.js та TypeScript. Заснований на принципах Angular, Nest.js пропонує розробникам чітку структуру та високий рівень абстракції, що сприяє швидкій та організованій розробці. Однією з ключових особливостей є використання декораторів та dependency injection, що спрощує створення та управління

модулями, контролерами та провайдерами. Nest.js також включає в себе вбудовані засоби для автоматичної генерації API документації, що полегшує взаємодію з API та підвищує його читабельність.

Фреймворк слідує архітектурному шаблону MVC (Model-View-Controller), що дозволяє ефективно розділяти логіку додатку та покращує його тестованість. Nest.js підтримує використання WebSocket для реального часу та обробки подій, що робить його ідеальним вибором для створення сучасних веб-застосунків. Його інтеграція з іншими бібліотеками та фреймворками, а також розширенням можливостей за допомогою middleware, робить його потужним інструментом для розробників, які шукають гнучкість та швидкість у розробці серверних застосунків.

PostgreSQL – це об'єктно-реляційна система управління базами даних, яка відома своєю надійністю, розширюваністю та великою кількістю функцій. PostgreSQL використовує SQL-мову для взаємодії з даними та підтримує ряд додаткових можливостей, які включають схеми, відкладені індекси, тригери та велику кількість типів даних.

PostgreSQL має потужну систему підтримки складних запитів, включаючи вкладені запити, віконні функції, зовнішні об'єднання та загальні таблиці виразів. Також вона підтримує розширені функції безпеки, такі як шифрування даних та контроль доступу на основі ролей. PostgreSQL також відомий своєю високою налаштовуваністю та можливістю розширення через додаткові модулі, що дозволяє розширити функціональність бази даних за допомогою сторонніх розширень.

СУБД підтримується активною та відкритою спільнотою розробників, які постійно працюють над покращенням системи. Спільнота надає різноманітні ресурси для допомоги користувачам, включаючи документацію, навчальні матеріали та активні форуми для обговорення проблем та найкращих практик. Через свою популярність, PostgreSQL широко інтегрований з іншими технологіями та платформами, роблячи його вибором для розробників у всьому світі для великих та складних проектів баз даних.

ethers.js – бібліотека для взаємодії з Ethereum-блокчейном. Бібліотека пропонує функціонал для створення, підпису та відправлення транзакцій, а також операцій з гаманцями. ethers.js побудована з використанням TypeScript, що забезпечує високий рівень типізації. Бібліотека підтримує роботу з різними мережами Ethereum, включаючи тестові мережі.

Бібліотека пропонує набір класів та функцій, що дозволяють взаємодіяти з Ethereum блокчейном, включаючи Wallet для керування приватними ключами та виконання транзакцій, Provider для підключення до Ethereum вузлів та здійснення читання даних з блокчейну, та Contract для взаємодії зі смарт-контрактами. Ethers.js також підтримує роботу з різними стандартами токенів Ethereum, включаючи ERC-20 та ERC-721, і надає вбудовані утиліти для роботи з рядками, числами та іншими форматами даних, характерними для Ethereum.

Через свою високу продуктивність та легкість інтеграції, ethers.js широко використовується у розробці децентралізованих додатків (DApps), особливо тих, що залучають взаємодію зі смарт-контрактами Ethereum. Ця бібліотека активно підтримується спільнотою розробників, що забезпечує постійне оновлення та покращення її функціоналу. Крім того, широке використання ethers.js у спільноті Ethereum сприяє створенню численних навчальних матеріалів та ресурсів, які допомагають новим розробникам швидко освоїти особливості роботи з Ethereum блокчейном.

grammy.js – бібліотека JavaScript, призначена для створення ботів для месенджера Telegram. Бібліотека забезпечує легкий і гнучкий інтерфейс для взаємодії з Telegram Bot API, що дозволяє розробникам швидко розробляти та розгортати ботів. Grammy.js підтримує всі основні функції Telegram Bot API, включаючи відправку повідомлень, фото, відео, та інших типів медіа, а також управління чатами і групами. Бібліотека славиться своєю високою продуктивністю та мінімалізмом, що робить її ідеальним вибором для створення швидких і надійних ботів.

Бібліотека використовує сучасні функції JavaScript, такі як асинхронні операції (async/await), для спрощення роботи з асинхронними запитами до

Telegram API. Grammy.js пропонує модульну структуру, що дозволяє легко інтегрувати додаткові плагіни та розширення, які розширюють функціональність бота. Наприклад, розробники можуть використовувати плагіни для обробки команд, управління сесіями, або додавання спеціалізованої логіки обробки повідомлень. Така гнучкість робить grammy.js відмінним вибором для розробки ботів різної складності.

3.4 Опис модулів програмної системи

Серверна частина проєкту, складаючи критичну основу для загальної архітектури, організована у вигляді різноманітних модулів, кожен з яких має чітко визначену роль та функціональність. Ця модульна структура забезпечує гнучкість та масштабованість системи, дозволяючи легко додавати, модифікувати чи вилучати компоненти залежно від потреб проєкту.

Взаємодія між основними модулями системи та сторонніми сутностями представлена на рисунку нижче.

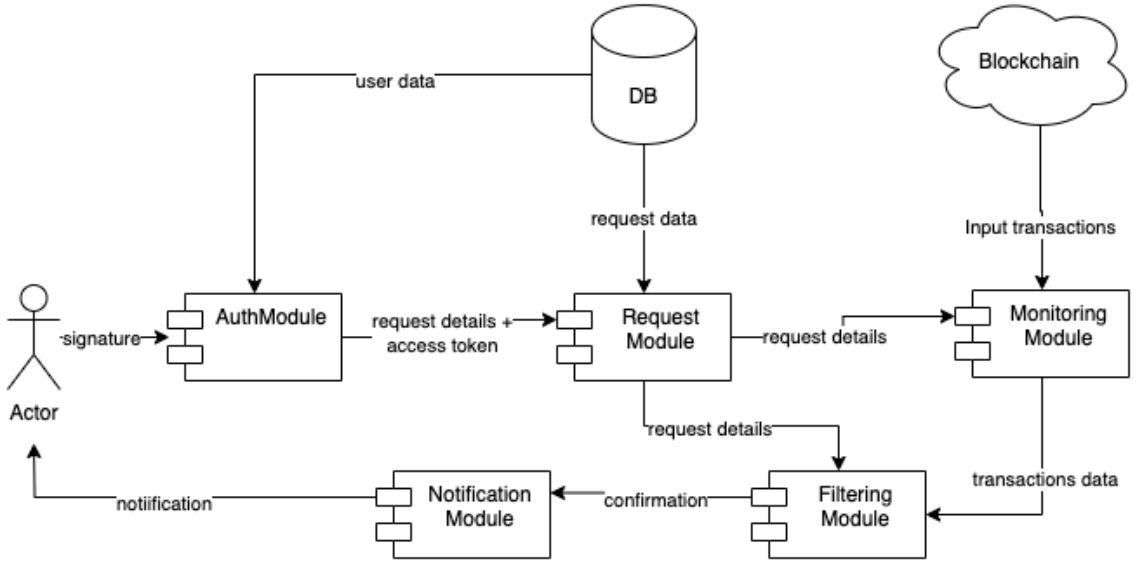


Рис. 3.3 Діаграма взаємодії між модулями системи

AuthModule – відповідає за автентифікацію користувачів у системі, працює шляхом верифікації підписаного користувачем повідомлення за допомогою його цифрового гаманця. Процес автентифікації починається з отримання системою підписаного повідомлення. Воно включає в себе зашифровану інформацію, яка

дозволяє ідентифікувати користувача. На наступному етапі система екстрагує з цього повідомлення адресу цифрового гаманця, яким було здійснено підпис.

Після цього відбувається ключовий момент процесу – перевірка наявності адреси гаманця у внутрішній базі даних клієнтів системи. Ця база даних містить інформацію про всіх зареєстрованих користувачів та їхні цифрові гаманці. Якщо адреса гаманця виявляється у базі, система вважає користувача автентифікованим.

У випадку успішної автентифікації, система генерує та видає користувачу спеціальний токен – accessToken. Цей токен є своєрідним "ключем" до інших сервісів системи. Користувачі повинні включати цей токен у заголовки своїх запитів при взаємодії з іншими контролерами системи. Таким чином, accessToken служить додатковим рівнем захисту, оскільки підтверджує, що запит відправлено автентифікованим користувачем.

Зазначений нижче код є частиною класу AuthService, який виконує ці функції. Код включає в себе алгоритми для обробки підписаного повідомлення, екстракції адреси гаманця, взаємодії з базою даних для перевірки автентичності, а також генерації та видачі accessToken.

```
@Injectable()
export class AuthService {
  constructor(
  public async getAccessToken(
    createAccessTokenDto: CreateAccessTokenRequestDto,
  ): Promise<CreateAccessTokenResponseDto> {
    const clientEntity = await this.clientRepository.findOne({
      where: {
        clientAddress: createAccessTokenDto.clientAddress.toLowerCase(),
      },
    });
    if (!clientEntity) {
      throw new BadRequestException('Client not found');
    }
    if (addMinutes(new Date(clientEntity.updatedAt), 10) > new Date()) {
      throw new BadRequestException('Message was expired');
    }
    let signedAddress;
    try {
                                    signedAddress      =      verifyMessage(clientEntity.tempCode,
createAccessTokenDto.signedMessage);
    } catch (err) {
      console.log(err);
```

```
    throw new BadRequestException('Incorrect signed message');
  }

  if (signedAddress.toLowerCase() !== clientEntity.clientAddress) {
    throw new ForbiddenException('Invalid signed message');
  }

  this.clientRepository.update(clientEntity.id, { tempCode: null });

  return {
    accessToken: sign({ id: clientEntity.id }, process.env.JWT_SECRET, {
      expiresIn: '2h',
    }),
  };
}
```

RequestModule – ключова складова системи моніторингу транзакцій, виконує важливу роль у взаємодії з користувачем. Цей модуль забезпечує інтерфейс для створення, редагування та видалення запитів користувачем, пов'язаних із моніторингом транзакцій. Даний модуль було розроблено з метою забезпечити гнучкість та ефективність управління запитами з боку користувачів.

Створення запиту включає в себе збір детальної інформації про параметри моніторингу, які задає користувач. Це може включати специфікацію певних типів транзакцій, визначення меж часових проміжків, фільтрацію за учасниками транзакції, сумами та іншими критеріями. RequestModule обробляє ці дані та створює структурований запит, який потім передається до системи моніторингу.

Редагування існуючих запитів користувачем відбувається через спеціально розроблений інтерфейс, який дозволяє модифікувати параметри моніторингу. Користувач має можливість оновити критерії транзакцій, додати нові умови або змінити часові рамки запиту. Після внесення змін, RequestModule переформатує запит та оновлює його в системі.

Видалення запитів також є важливою функцією модуля. Користувач може анулювати неактуальні запити на моніторинг, що дозволяє звільнити ресурси системи та уникнути зайвого накопичення даних. RequestModule надає інструменти для виявлення та безпечного видалення таких запитів, забезпечуючи, при цьому, збереження цілісності даних у системі.

Технічно, RequestModule включає в себе набір API-ендпойнтів для обробки запитів користувача, а також взаємодії з базою даних для зберігання та управління цими запитами. Цей модуль також інтегрований з модулями безпеки та автентифікації системи для забезпечення належного рівня доступу та контролю за діями користувачів.

```
@ApiOperation({
  description: 'Create request',
})
@ApiBody({
  type: CreateRequestReqDto,
})
@ApiHeaders([{ name: 'Authorization', description: 'Access token', required: true }])
@ApiCreatedResponse({
  type: RequestEntity,
})
@UseGuards(AuthGuard)
@Post('')
public async createRequest(
        @Body(new ValidationRequestBodyPipe<CreateRequestReqDto>()) request: CreateRequestReqDto,
  @HeaderAuthorization() jwtPayload: JwtPayloadInterface,
): Promise<CreateRequestResponseDto> {
  return this.requestService.createRequest(request, jwtPayload);
}

@ApiOperation({
  description: 'Cancel Request',
})
@ApiHeaders([{ name: 'Authorization', description: 'Access token', required: true }])
@UseGuards(AuthGuard)
@Delete(':id')
public async cancelRequest(
  @Param('id', new ParseIntPipe()) id: number,
  @HeaderAuthorization() jwtPayload: JwtPayloadInterface,
): Promise<{ message: string }> {
  return this.requestService.cancelRequest(id, jwtPayload);
}

@ApiOperation({
  description: 'Update request',
})
@ApiHeaders([{ name: 'Authorization', description: 'Access token', required: true }])
@UseGuards(AuthGuard)
@ApiResponse({
  status: 200,
})
@Put(':id')
public async updateRequest(
```

```
  @Body() request: UpdateRequestReqDto,
  @Param('id', new ParseIntPipe()) id: number,
  @HeaderAuthorization() jwtPayload: JwtPayloadInterface,
): Promise<void> {
  return this.requestService.updateRequest(request, id, jwtPayload);
}

@ApiOperation({
  description: 'Get methods',
})
@Get('/methods')
public async getTriggerMethods(): Promise<Array<GetTriggerMethodsResponseDto>> {
  return this.requestService.getTriggerMethods();
}
```

MonitoringModule – модуль стежить за транзакціями у мемпулі (пулі транзакцій, що очікують на підтвердження в блокчейні), керує процесом підписки та перепідписки на транзакції, відповідно до змін у деталях запитів користувачів.

Технічно, MonitoringModule реалізований так, щоб підтримувати постійне з'єднання з мережею блокчейна для безперервного моніторингу транзакцій у мемпулі. Модуль автоматично реєструється на отримання сповіщень про нові транзакції, що відповідають критеріям моніторингу, визначеним у запитах користувачів. Коли деталі запиту змінюються, MonitoringModule виконує перепідписку, оновлюючи параметри моніторингу відповідно до нових вимог.

Після отримання повідомлення про нову транзакцію, модуль ініціює процес її обробки. Це включає передачу даних про транзакцію до модуля фільтрації, який виконує додаткову перевірку та визначає, чи відповідає транзакція умовам, встановленим у запиті. Після успішної фільтрації та отримання підтвердження, MonitoringModule активізує модуль сповіщень, який відповідає за інформування користувачів про нові транзакції.

MonitoringModule включає механізми для роботи з веб-сокетами, що забезпечують швидке отримання даних про транзакції. Також модуль містить алгоритми для динамічного управління підписками, що дозволяє адаптуватися до змінних запитів користувачів без необхідності перезавантаження або переривання роботи системи. Робочий стан з'єднання постійно відслідковується, а автоматичні

механізми відновлення використовуються для забезпечення безперервності моніторингу.

```
this.ws.on('message', async (message) => {
    message = JSON.parse(message);
    const messageType = this.getTypeOfMessage(message);

    switch (messageType) {
      case BloxrouteMessageTypeEnum.Transaction: {
        try {
          const transactionDto = fromWsMessageToTransactionMessage(message);
                    const triggerMethod = getWsTxType(transactionDto.transaction.data,
this.triggerMethods);
          if (!triggerMethod) {
            break;
          }
          const decodedData = decodeData(triggerMethod, transactionDto.transaction.data);
          const requestEntities = this.filteringService.filterRequestsByIncomingTransaction(
            this.requests,
            transactionDto,
            decodedData,
            triggerMethod,
          );

          for (let req of requestEntities) {
            const sentRequest = this.notificationService.sendNotification(
              req,
              this.chainId,
            );
            this.txWs.send(sentRequest);

            const reqHistory = fromTxMessageToRequestHistoryEntityMap(
              transactionDto,
              req.id,
              decodedData,
            );
            this.requestHistoryRepository.createRequestHistory(reqHistory);
          }
```

FilteringModule – перевіряє відповідность транзакцій запитам користувачів. Цей модуль реалізований для того, щоб забезпечити високу точність та ефективність у фільтрації транзакцій, використовуючи розроблені алгоритми та бази даних.

Процес фільтрації в FilteringModule починається з отримання списку активних запитів користувачів з бази даних. Ця база даних містить детальну

інформацію про параметри моніторингу, визначені користувачами, включаючи критерії, як-от ідентифікатори відправника та отримувача, характеристики транзакцій, та інші специфічні умови.

Кожна отримана транзакція проходить через серію перевірок у FilteringModule. Модуль аналізує дані транзакції, парсить її атрибути, такі як ідентифікатори учасників, суми, час відправлення, та інші параметри. Це включає детальний аналіз структури транзакції, що може відрізнятися в залежності від її типу, наприклад, проста передача коштів, контрактні виклики, або інші специфічні операції.

На основі цього аналізу, FilteringModule порівнює параметри кожної транзакції з критеріями у запитах користувачів. Цей процес включає алгоритми зіставлення, щоб точно визначити, чи відповідає транзакція запиту. У випадку збігу, транзакція вважається відповідною запиту, та модуль ініціює виклик до Notification Module.

NotificationModule активізується після того, як FilteringModule підтверджує збіг між вхідною транзакцією та користувацьким запитом. Він відповідає за генерацію та відправлення сповіщень користувачам, інформуючи їх про виявлені транзакції, що відповідають їхнім критеріям моніторингу. NotificationModule розроблений для підтримки різноманітних каналів комунікації, таких як електронна пошта, месенджери. Це забезпечує гнучкість у виборі каналу комунікації для кожного користувача, заснованого на їх персональних перевагах або вимогах.

Коли збіг між транзакцією та запитом виявляється, NotificationModule формує сповіщення, яке містить всю важливу інформацію про транзакцію. Це включає деталі такі, як ідентифікатор транзакції, суму, час відправлення, учасників транзакції, та інші релевантні дані. Сповіщення також можуть бути кастомізовані згідно з вимогами користувача, включаючи різні формати відображення або додаткові посилання для подальшого аналізу транзакції.

Для відправлення сповіщень, модуль використовує інтеграцію з зовнішніми сервісами комунікацій або вбудовані рішення. Наприклад, для електронних листів

використані SMTP-сервери, а для месенджерів - API Telegram. Така інтеграція дозволяє NotificationModule швидко та ефективно взаємодіяти з користувачами.

3.5 Основні розроблені класи системи

Діаграма класів. Діаграма класів є ключовим компонентом UML (Unified Modeling Language) і відіграє важливу роль у проектуванні об'єктно-орієнтованих систем. Вона надає візуальний огляд класів, які складають систему, та їх взаємодії. Кожен клас на діаграмі представлений у формі прямокутника, розділеного на три секції: назва класу, атрибути (або поля) та методи (або функції). Назва класу зазвичай розміщується у верхній секції, атрибути - у середній, а методи - у нижній. Ці елементи допомагають розробникам зрозуміти, як класи пов'язані між собою та як вони взаємодіють у межах системи.

Відносини між класами на діаграмі представлені за допомогою ліній та стрілок, що вказують на взаємодії та залежності між класами. Основні типи відносин включають наслідування, асоціацію, агрегацію та композицію. Наслідування використовується для вказівки на те, що один клас (підклас) наслідує властивості та поведінку іншого класу (базового класу). Асоціація відображає зв'язки між об'єктами різних класів. Агрегація та композиція є особливими типами асоціацій, які описують відносини цілого до його частин.

Діаграма класів не тільки допомагає у візуалізації структури системи, але й служить основою для створення більш детальних моделей поведінки, таких як діаграми послідовностей або діаграми станів. Це робить її незамінною для розуміння та документування складних об'єктно-орієнтованих систем. Окрім того, діаграма класів часто використовується як основа для кодогенерації в інструментах автоматизованого проектування.

Розглянемо основні класи розробленої системи, їх методи, властивості, а також зв'язки між ними.

Основні класи системи та їх характеристики:

— User:

Властивості: id, walletAddress, requests.

Опис: сутність користувача системи.

— **Request:**

Властивості: id, user, trigger, status.

Опис: сутність користувача системи, містить посилання на сутність користувача системи та умови тригер транзакції.

— **Trigger:**

Властивості: id, sender, receiver, transactionType, params, transaction.

Опис: сутність тригер транзакції, зберігає в собі параметри, визначені користувачем, за якими відбувається фільтрація вхідних транзакцій, а також посилання на вхідну транзакцію, що задовольняє умови.

— **Transaction**:

Властивості: id, sender, receiver, data, trigger.

Опис: сутність вхідної транзакції, зберігає в собі стандартні атрибути транзакції, необхідна для збереження історії транзакцій, що задовольнили визначені користувацькі параметри.

— **AuthService:**

Властивості: userRepository.

Методи: verifyUser(), generateAccessToken(), generateRefreshToken().

Опис: авторизація та автентифікація користувача для подальшої взаємодії з частинами системи.

— **RequestService:**

Властивості: requestRepository, authService.

Методи: createRequest(), updateRequest(), cancelRequest(), getUserRequests(), getRequestById().

Опис: взаємодія з сутністю запиту на моніторинг, створення, оновлення, скасування активних запитів.

— **MonitoringService:**

Властивості: provider, filteringService.

Методи: startMonitoring(), onTransaction(), onRequest().

Опис: створення зв’язку з блокчейн провайдером, підписка на отримання нових транзакцій.

— **FilteringService:**

Властивості: requestService, monitoringService, notificationService.

Методи: filterTransaction().

Опис: фільтрація вхідних транзакцій відповідно до запитів користувача.

— **NotificationService:**

Властивості: userRepository, requestRepository.

Методи: sendNotification().

Опис: відправка сповіщень користувачам при виявленні підозрілих транзакцій.

Зв'язки між класами системи

Transaction-Request: Клас "Transaction" взаємодіє з "Request" для фільтрації та виявлення транзакцій, які відповідають критеріям запиту.

User-Request: Клас "User" асоціюється з "MonitoringRequest", оскільки користувачі створюють та керують цими запитами.

NotificationService-User: "NotificationService" пов'язаний з "User", адже сповіщення надсилаються конкретним користувачам.

MonitoringService-Transaction: "MonitoringService" використовується для отримання даних про транзакції, які потім представляються через клас "Transaction".

Ця діаграма класів відображає структуру та взаємозв'язки між основними компонентами системи, що забезпечує її функціональність та гнучкість. Кожен клас виконує специфічні завдання і взаємодіє з іншими класами, формуючи координовану систему для моніторингу та аналізу транзакцій.

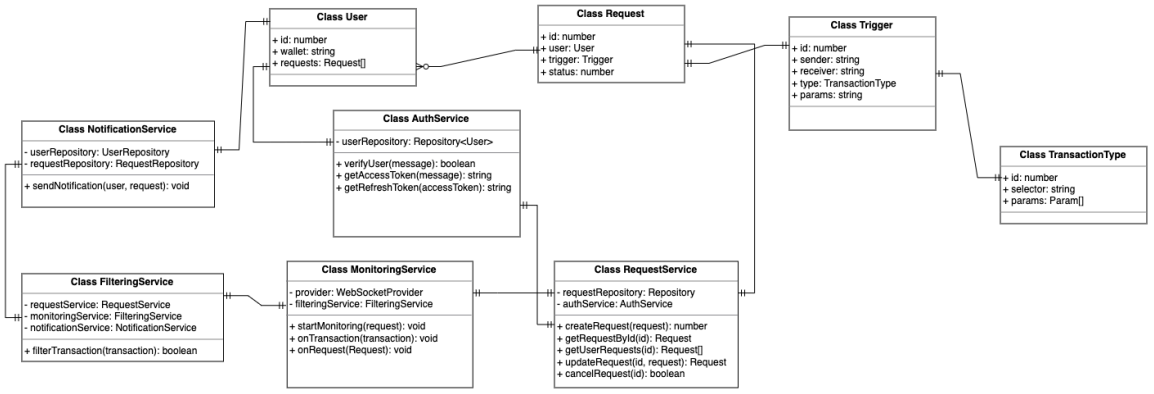


Рис. 3.4 Діаграма класів моніторингової системи

3.6 Користувацький інтерфейс системи

Розглянемо процес створення запиту користувачем. Взаємодія користувача з системою відбувається через інтерфейс Telegram бота.

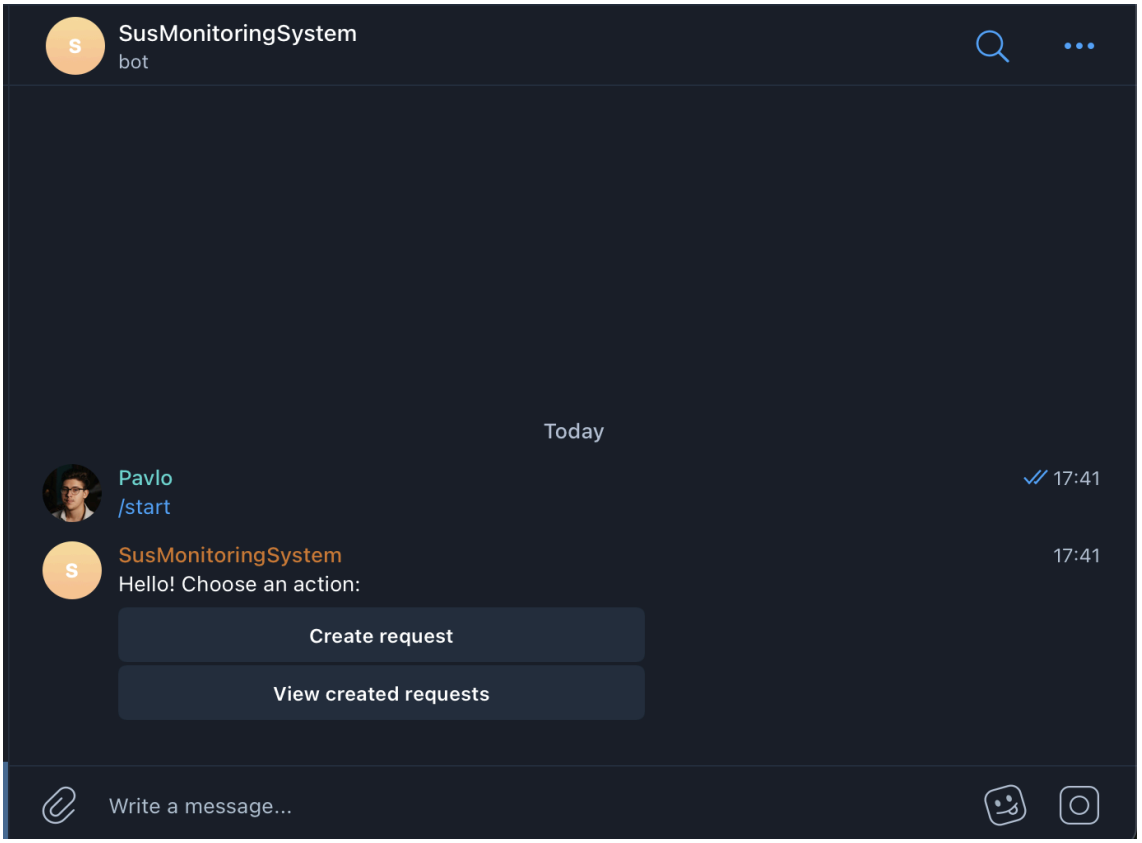


Рис. 3.5 Стартове меню системи

При натистканні на кнопку “Create request” відкривається меню створення запиту. Процес створення запиту починається з вибору типу тригер транзакції. Меню вибору зображено на рисунку 3.5.

Далі система залежно від типу транзакції попросить користувача ввести параметри транзакції, що буде відслідковуватись. Першим параметром запиту незалежно від типу транзакції є ідентифікатор мережі, в якій транзакція буде відслідковуватись.

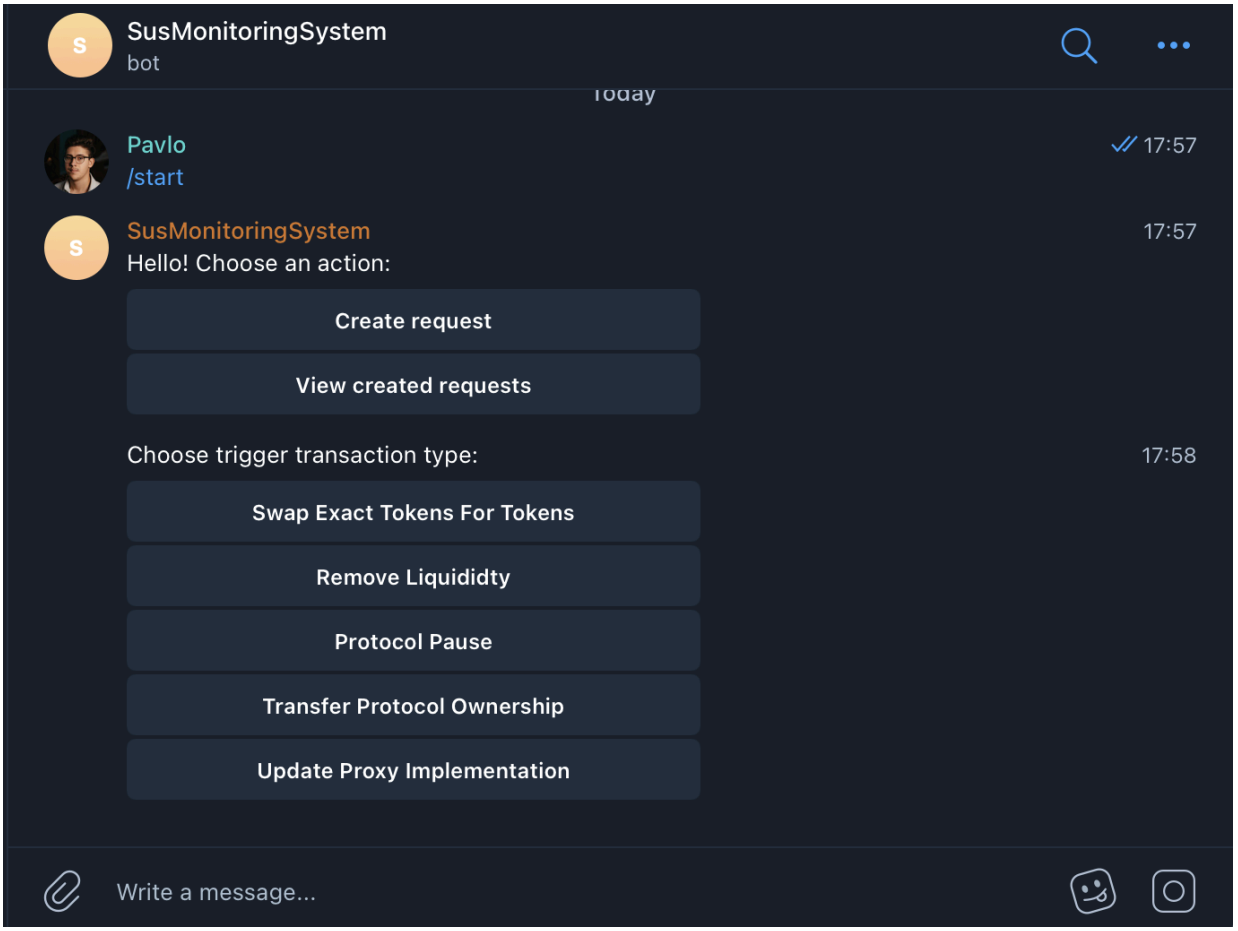


Рис. 3.6 Меню вибору типу запиту

Розглянемо приклад параметрів на основі транзакції типу SwapExactTokensForTokens. Процес представлено на рисунку 3.6.

Спершу користувач вводить адресу відправника транзакції на обмін токенами, після цього вводить пару токенів в якій буде проводитись обмін. В даному випадку вказано пару токенів Cake - USDT. Також користувач може

вказати мінімальну вхідну суму та адресу отримувача для фільтрації транзакції. Після введення значень всіх показників система попросить користувача підтвердити вказані параметри. Після натискання на кнопку “Confirm” система почне відслідковувати транзакції, що надходять в мемпул.

Рис. 3.7 Введення параметрів запиту

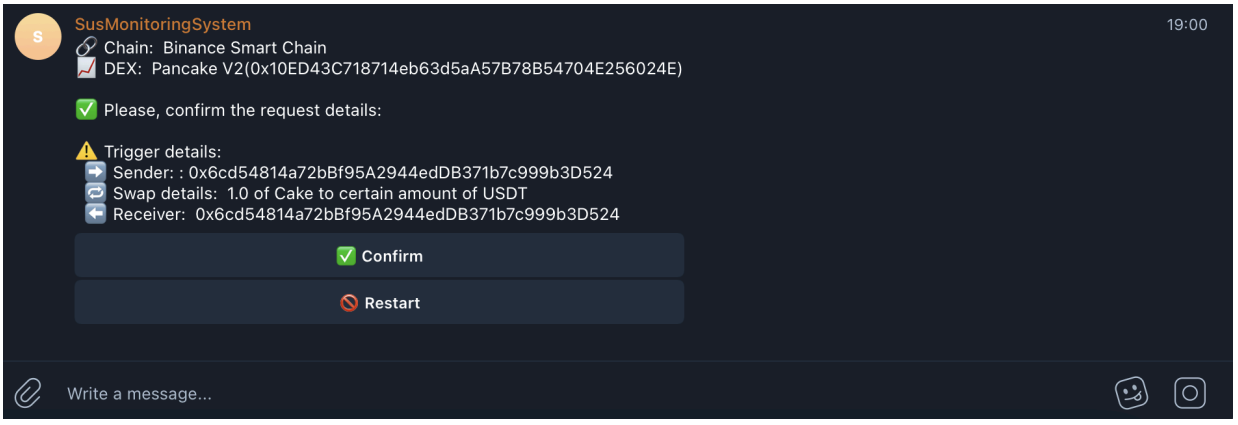


Рис. 3.8 Підтвердження створення запиту

При появі відповідної транзакції в мемпулі система відправить повідомлення користувачу з деталями транзакції та посиланням на сервіс сканеру транзакції.

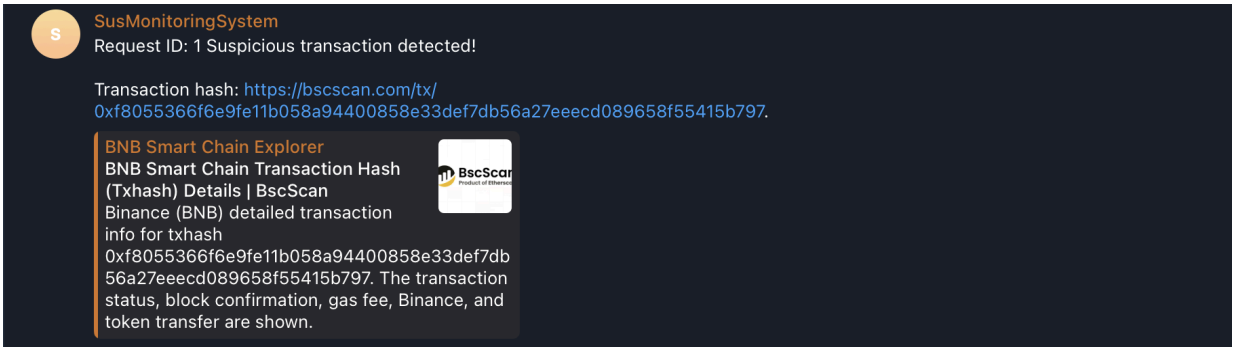


Рис. 3.9 Сповіщення користувача про знаходження підозрілої транзакції по заданим критеріям

За посиланням з повідомлення можна відслідкувати деталі транзакції, яка була відловлена. Як бачимо, транзакція задовольнила вказані в запиті критерії, після чого було відправлене сповіщення.

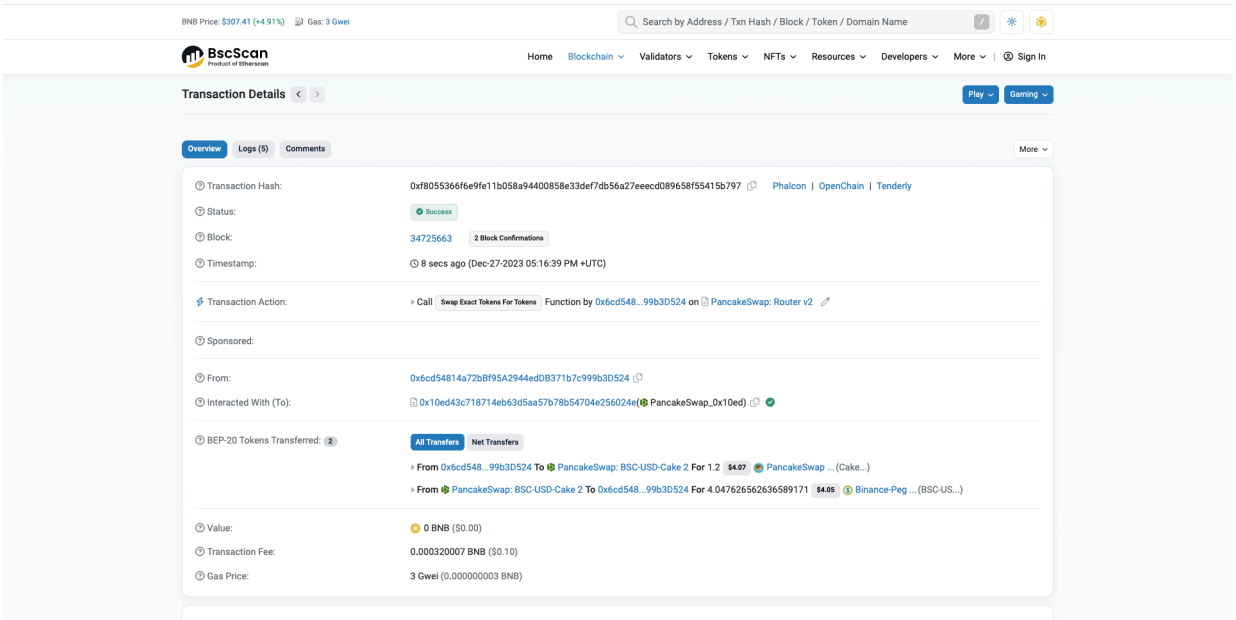


Рис. 3.10 Транзакція, що була відловлена системою

3.7 Ефективність розробленої системи

В таблиці представлені часові показники процесу моніторингу транзакції на зупинку протоколу. Розрахунки проводились для обміну токенів – swapExactTokensForTokens на платформі PancakeSwapV2 в мережі Binance Smart

Chain. Зафіксовано час відправки транзакції (t_p) та час фіксування транзакції системою(t_d). Різниця склала 0.109 сек. Часова різниця між часом включення транзакції до блоку(t_c) та часом обробки транзакції системою та надсилання сповіщення(t_{pr}) склала 2.223 сек. Показники, отримані в результаті дослідження задовольняють математичну модель.

Таблиця 3.1

Розрахунки показників системи за математичною моделлю

Показник	**Значення**
tp – час додавання транзакції в пул	15:16:37:884
ti – час додавання транзакції в блок	15:16:39:143
tc – час підтвердження транзакції	2.791 сек
td – час фіксування транзакції системою	15:16:37:993
tf – час завершення обробки транзакції	15:16:38:561
tpr – час обробки транзакції	0.568 сек
tc - tpr – різниця між часом підтвердження і часом обробки	2.223 сек
tp - td – різниця між часом додавання транзакції в пул та часом фіксування транзакції системою	0.109 сек

ВИСНОВКИ

В результаті дослідження було запропоновано програмну систему, яка аналізує вхідні транзакції в блокчейн та відокремлює ті, що потрапляють під користувацькі критерії.

Досліджено основні поняття, що фігурують в системі, а саме транзакція та смарт контракт, проаналізовано основні варіанти застосування смарт контрактів та основні варіанти атак з застосуванням смарт контрактів.

Було розроблено математичну модель, показником якої є час перевірки підозрілої транзакції згідно заданими користувачем параметрами. Запропоновано алгоритм моніторингу та сповіщення користувача про появу підозрілих транзакцій в мемпулі.

В результаті проектування було запропоновано програмну систему, яка аналізує вхідні транзакції в блокчейн та відокремлює ті, що потрапляють під користувацькі критерії. Система задовольняє умови математичної моделі згідно з тестовими вимірами. Система може бути розширена додатковими сценаріями для моніторингу. Компоненти системи було описано текстово та графічно, за допомогою діаграм класів та взаємодії.

На основі результатів проектування було розроблено програмну систему за допомогою технологій Nest.js, PostgreSQL, grammy.js та ethers.js. Розроблена модель задовольняє визначені функціональні та нефункціональні вимоги.

Система задовольняє умови математичної моделі згідно з тестовими вимірами. Система може бути розширена додатковими сценаріями для моніторингу. Також система є універсальною для блокчейнів Ethereum, BSC, Polygon, оскільки побудована з врахуванням архітектури EVM.

Застосування даної системи є доцільним в контексті попередження атак на протоколи, в яких користувач зберігає власні активи. Основною перевагою рішення є можливість користувача самостійно налаштовувати систему під свої потреби, не очікуючи на аудит чи імплементацію безпекових рішень зі сторони протоколу.

Подальше дослідження даної тематики може бути спрямоване на автоматизацію процесу реагування на підозрілу транзакцію вихідною

транзакцією, що потребує доступу до користувацького гаманця, або завчасного формування відповідної транзакції.

ПЕРЕЛІК ПОСИЛАНЬ

1. Wang B., Liu H., Liu C., Yang Z., Ren Q., Zheng H., Lei Hong. Blockeye: Hunting For DeFi Attacks on Blockchain. Oxford-Hainan Blockchain Research Institute, 2020. С. 1-4.
2. Kangaa D., Azzouazib M., Yassine M. Ghoumrarib E., Daifb A. Management and Monitoring of Blockchain Systems, *International Workshop on Blockchain Security.* Листопад 2-5, 2020. Madeira, Portugal, 2020. С. 605-612.
3. Топчій М., Гальчинський Л., Підвищення Рівня безпеки смарт-контрактів в мережі Ethereum від шахрайства за рахунок використання реверсивних токенів, *République française*. Листопад 11, 2022. Paris, France, 2022. С.71-77. DOI 10.36074/logos-11.11.2022.20.
4. H. Liu, C. Liu, W. Zhao, Y. Jiang, and J. Sun, “S-gram: towards semantic-aware security auditing for ethereum smart contracts,” in ASE. ACM, 2018, pp. 814–819.
5. Zeli WANG, Hai JIN, Weiqi DAI, Kim-Kwang Raymond CHOO, Deqing ZOU “Ethereum smart contract security research: survey and future research opportunities”, Frontiers of Computer Science, 2020
6. Ethereum Foundation. Ethereum Virtual Machine (EVM). 2015. №1. С. 1-4. URL: https://ethereum.org/en/developers/docs/evm/.
7. Wood G. Ethereum: a secure decentralized transaction ledger. 2014. № 1. С. 1-3. URL: https://ethereum.github.io/yellowpaper/paper.pdf
8. Szabo N. Smart Contracts: Building Blocks for Digital Markets. 1996. №1 С.1-2. URL: https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html
9. OpenZeppelin. Utilities: Pausable. 2023. №5. С. 1-3. URL: https://docs.openzeppelin.com/contracts/5.x/api/utils#Pausable
10. OpenZeppelin. Access Control: Ownable. 2023. №5. С. 1-3. URL: https://docs.openzeppelin.com/contracts/5.x/api/access#Ownable
11. D. Bernard, M. Azouazi, M. Yassine, and E. Ghoumrari, “ScienceDirect Blockchain management and monitoring .,” vol. 01, 2020

12. Phitchayaphong Tantikul, Sudsanguan Ngamsuriyaroj, “Exploring Vulnerabilities in Solidity Smart Contract”, 2020 URL: https://www.scitepress.org/Papers/2020/89098/89098.pdf
13. Jing Huang, Kuo Zhou, Ao Xiong and Dongmeng Li “Smart Contract Vulnerability Detection Model Based on Multi-Task Learning” URL: https://mdpi-res.com/d_attachment/sensors/sensors-22-01829/article_deploy/sensors-22-01829-v2.pdf?version=1645873574
14. Ardit Dika, “Ethereum Smart Contracts: Security Vulnerabilities and Security Tools”. URL: https://ntnuopen.ntnu.no/ntnu-xmlui/bitstream/handle/11250/2479191/18400_FULLTEXT.pdf
15. Sarwar Sayeed, Hector Marco-Gisbert, Tom Caira, “Smart Contract Attacks and Protections”. URL: https://www.researchgate.net/publication/338926064_Smart_Contract_Attacks_and_Protections
16. L. Luu, D.H. Chu, H. Olickel, P. Saxena, and A. Hobor, “Making smart contracts smarter,” in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, vol. 01, 2016.
17. P. Tsankov, A. Dan, D. D. Cohen, A. Gervais, F. Buenzli, M. Vechev, “Securify: Practical security analysis of smart contracts,” arXiv preprint arXiv:1806.01143, vol. 00, 2018.
18. C. Liu, H. Liu, Z. Cao, Z. Chen, B. Chen, and B. Roscoe, “Reguard: finding reentrancy bugs in smart contracts,” in ICSE (Companion). ACM, 2018.
19. Z. Yang, H. Liu, Y. Li, H. Zheng, L. Wang, and B. Chen, “Seraph: enabling cross-platform security analysis for evm and wasm smart contracts,” in Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings, 2020.
20. Everett Hildenbrandt, Manasvi Saxena, Xiaoran Zhu, Nishant Rodrigues, Philip Daian, Dwight Guth, and Grigore Rosu. KEVM: A Complete Semantics of the Ethereum Virtual Machine. Technical Report, 2017.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

(Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Магістерська робота

«РОЗРОБКА СИСТЕМИ МОНІТОРИНГУ ТА СПОВІЩЕННЯ ПРО ПІДОЗРІЛІ ТРАНЗАКЦІЇ НА БАЗІ EVM-БЛОКЧЕЙНІВ»

Виконав: студент групи ПДМ-61 Зібаров Павло Сергійович

Керівник: к.т.н., доц., доцент кафедри ІПЗ Негоденко Олена Василівна

Київ - 2024

МЕТА, ОБ'ЄКТА ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: покращення рівня користувацької безпеки за рахунок моніторингу та сповіщення про підозрілі транзакції на децентралізовані протоколи.

Об'єкт дослідження: моніторингові системи транзакцій на базі EVM-блокчейнів.

Предмет дослідження: система моніторингу та сповіщення про підозрілі транзакції на базі EVM-блокчейнів.

АКТУАЛЬНІСТЬ РОБОТИ

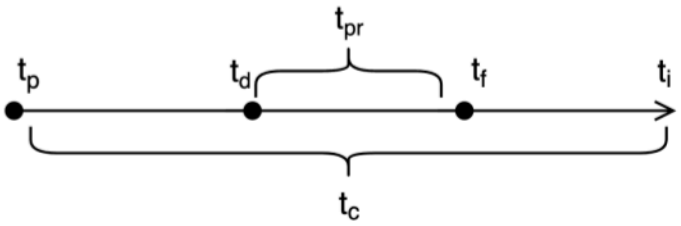
- Популяризація технології блокчейн та криптовалют
- Збільшення кількості децентралізованих систем та протоколів на EVM-мережах
- Збільшення кількості атак на децентралізовані протоколи
- Збільшення кількості шахрайських проєктів

ПОРІВНЯННЯ РОЗРОБЛЕНОГО РІШЕННЯ З ІСНУЮЧИМИ

Засіб захисту	Спрямування	Переваги	Недоліки
Аудит смарт контракту	Захист протоколу	Загальноприйнята практика	Людський фактор, зазвичай результати купуються
Безпекові бібліотеки	Захист протоколу	Додають можливості при розробці смарт контракту	Мають бути впроваджені при розробці протоколу
Моніторингові системи	Захист протоколу	Аналіз комплексних атак типу flash loan і тд на ходу	Потребують глибокого дослідження протоколу, що підлягає захисту
Розроблене рішення	Захист користувача	Набір готових сценаріїв; може бути кастомізоване новими сценаріями; надає час на реакцію	Для автоматизації реакцій потребує підписаної транзакції чи доступу до гаманця

МАТЕМАТИЧНА МОДЕЛЬ СИСТЕМИ

$$\begin{cases} t_c = t_i - t_p \\ t_{pr} = t_f - t_d \\ t_{pr} < t_c \\ t_{pr} \to \min \end{cases}$$



де,

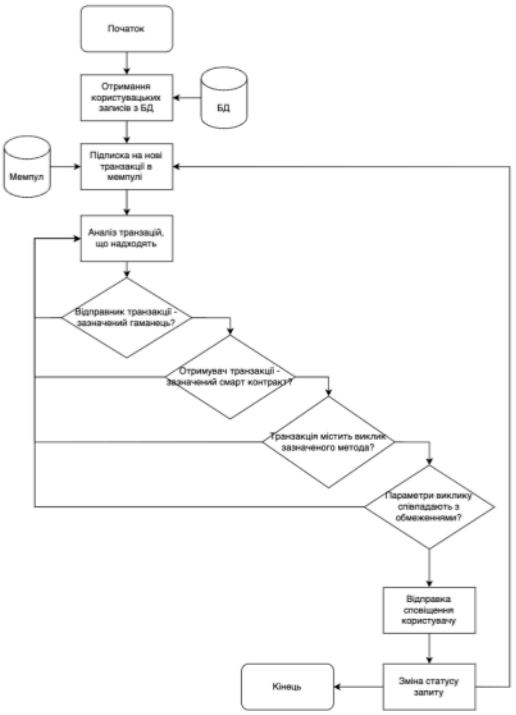
t_c – час підтвердження транзакції
t_p – час додавання транзакції в пул
t_i – час додавання транзакції в блок
t_{pr} – час обробки транзакції
t_d – час фіксування транзакції системою
t_f – час завершення обробки транзакції

ПОКАЗНИКИ ТРАНЗАКЦІЇ ЩО АНАЛІЗУЮТЬСЯ

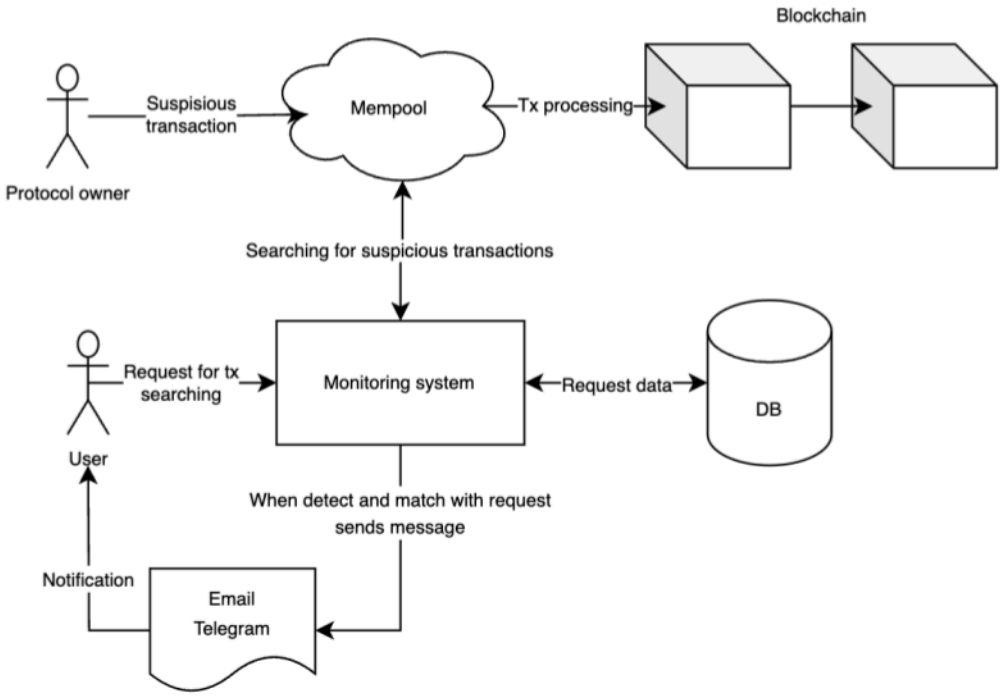
Показники	Опис	Приклад
sender	відправник транзакції, owner протоколу/власник токенів	0xD3BbF6….cff4FF3726e2
receiver	смарт контракт, до якого звертаються	0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D - адреса децентралізованої біржі Uniswap V2 Router
nonce	порядковий номер транзакції відправника, для фільтрації старих транзакцій	за замовчуванням береться поточний, порядковий номер останньої транзакції + 1
data.selector	сигнатура метода, що викликається	0xbaa2abde - сигнатура методу витягування ліквідності - removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)
data.values	значення параметрів, що передаються при виклику смарт контракта	значення параметрів tokenA, tokenB, amountIn тощо

ЗАПРОПОНОВАНІ СЦЕНАРІЇ ЗАСТОСУВАННЯ

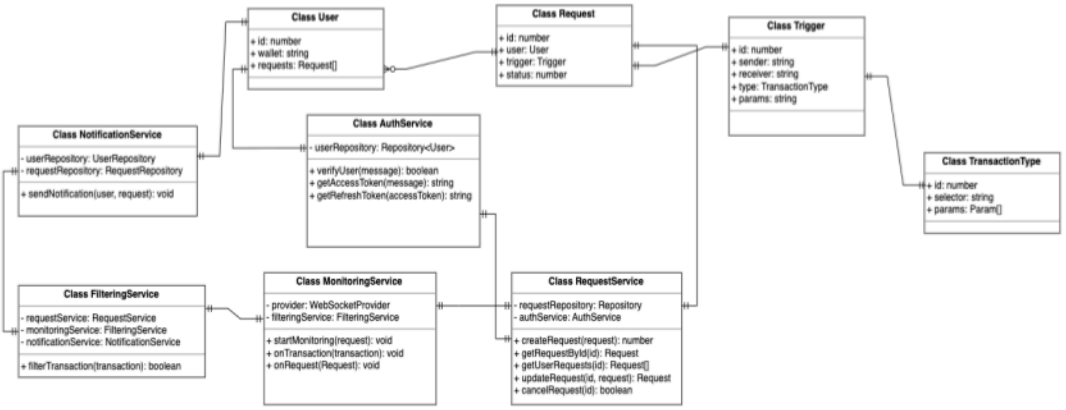
АЛГОРИТМ РОБОТИ СИСТЕМИ



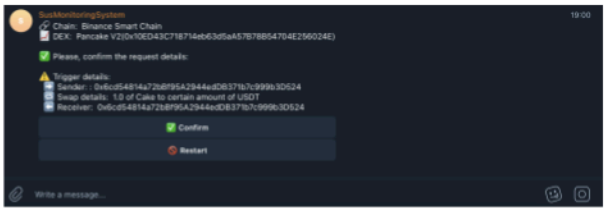
СТРУКТУРНА СХЕМА МОНІТОРИНГОВОЇ СИСТЕМИ

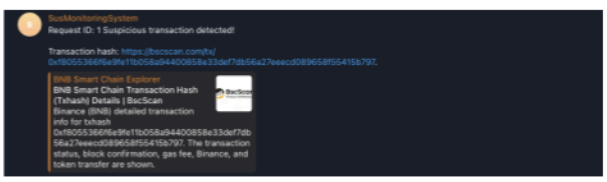


ДІАГРАМА КЛАСІВ РОЗРОБЛЕНОЇ СИСТЕМИ



ПРАКТИЧНА РЕАЛІЗАЦІЯ





- NodeJS API – приймає запити на стеження від користувача. Моніторингова система отримує від блокчейн нод дані про вхідні транзакції, обробляє та сповіщає користувача.
- Взаємодія з системою здійснюється через Telegram бота.
- Доступні блокчейни Ethereum, Binance Smart Chain, Polygon.

РЕЗУЛЬТАТИ МОДЕЛЮВАННЯ ТЕСТОВОЇ СИТУАЦІЇ

Показник	Значення
tp – час додавання транзакції в пул	15:16:37:884
ti – час додавання транзакції в блок	15:16:39:143
tc – час підтвердження транзакції	2.791 сек
td – час фіксування транзакції системою	15:16:37:993
tf – час завершення обробки транзакції	15:16:38:561
tpr – час обробки транзакції	0.568 сек
tc - tpr – різниця між часом підтвердження і часом обробки	2.223 сек
tp - td – різниця між часом додавання транзакції в пул та часом фіксування транзакції системою	0.109 сек

ВИСНОВКИ

1. Проаналізовано існуючі підходи до підвищення безпекового фактору в сфері децентралізованих систем. Базовими інструментами є аудит смарт контрактів та використання спеціальних безпекових бібліотек. Проте такі заходи спрямовані на перевірку надійності протоколу і лише опосередковано впливають на користувача.

2. Розроблено математичну модель, показником якої є час перевірки підозрілої транзакції згідно заданими користувачем параметрами. Запропоновано алгоритм моніторингу та сповіщення користувача про появу підозрілих транзакцій в мемпулі.

3. Розроблено програмну систему, яка аналізує вхідні транзакції в блокчейн та відокремлює ті, що потрапляють під користувацькі критерії. Система задовольняє умови математичної моделі згідно з тестовими вимірами. Система може бути розширена додатковими сценаріями для моніторингу. Також система є універсальною для блокчейнів Ethereum, BSC, Polygon, оскільки побудована з врахуванням архітектури EVM.

ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

Статті:

1. Зібаров П.С., Негоденко О.В. Розробка системи моніторингу та сповіщення про підозрілі транзакції на базі EVM-блокчейнів. // Науковий журнал «Молодий вчений». – № 12 (124) грудень 2023, Київ.

Тези доповідей:

1. Зібаров П.С., Негоденко О.В. Розробка системи моніторингу та сповіщення про підозрілі транзакції на базі EVM-блокчейнів. // V Міжнародна науково-практична конференція молодих вчених та студентів «ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ І ПЕРЕДОВІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ». – Київ: КПІ, 2023.

ДЯКУЮ ЗА УВАГУ!