Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(освітній рівень)

на тему: Розробка інструменту для виявлення шахрайських дій в блокчейні

Виконав: студентка VI курсу, групи СБм-61

Спеціальності:
125 «Кібербезпека та захист інформації»
(шифр і назва напряму підготовки, спеціальності)
Кобельник Анастасія Олександрівна
підпис (прізвище та ініціали)

Керівник Скарга-Бандурова І. С
підпис (прізвище та ініціали)

Нормоконтроль Стадник М. А.
підпис (прізвище та ініціали)

Завідувач кафедри Загородна Н.В.
підпис (прізвище та ініціали)

Рецензент Пастух О. А.
підпис (прізвище та ініціали)

м. Тернопіль – 2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В.
(підпис) (прізвище та ініціали)
«__» ___грудня__2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека та захист інформації
(шифр і назва спеціальності)

Студенту Кобельник Анастасії Олександрівні
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка інструменту для виявлення шахрайських дій в блокчейні

Керівник роботи Скарга-Бандурова Інна Сергіївна, д.т.н., проф.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «24» листопада 2025 року № 4/7-1024

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи історія транзакцій адреси, інформація про активи портфелю, ліквідність та обсяг торгів токенів, результати аудиту смартконтрактів.

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. Розділ 1. Основи безпеки в Web3: загрози та рішення. 1.1 Види блокчейнів і особливості адрес. 1.2 Поширені схеми шахрайства в блокчейні. 1.3 Огляд існуючих інстру -ментів та їх обмеження. Розділ 2. Проектування та реалізація інструменту. 2.1 Загальний опис проекту та архітектура системи. 2.1.1 Модуль збору загальної інформації про адресу. 2.1.2 Модуль відображення активів адреси. 2.1.3 Модуль відображення історії транзакцій. 2.1.4 Модуль візуалізації хронології активності. 2.1.5 Модуль візуалізації взаємодій адреси. 2.1.6 Модуль оцінки ризику шахрайства. 2.1.7 Модуль висновку щодо ризику взаємодії. 2.2 Тестування інструменту на реальних випадках. Розділ 3. 3.1 Безпечна взаємодія з API. 3.2 Управління та безпечне зберігання ключів доступу. 3.3 Заходи протидії типо вим веб -атакам. Розділ 4. Охорона праці та безпека в надзвичайних ситуаціях. 4.1 Охорона праці. 4.2 Захист людини від іонізуючого випромінювання. Висновки. Перелік використаних джерел.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титульна сторінка. 2. Елементи дослідження. 3. Актуальність. 4. Огляд існуючих інструментів та їх обмеження. 5. Архітектура інструменту. 6-7. Тестування на прикладі Airdrop Scam Tokens. 8. Оцінка ефективності анкети. 9. Безпечна робота з API та управління ключами. 10. Захист від типових атак. 11. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Осухівська Г.М., к.т.н., доцент, зав. кафедри КС		
Безпека в надхвичайних ситуаціях	Теслюк В.М., проректор з адміністративно-господарської роботи та будівництва		

7. Дата видачі завдання 19 вересня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	19.09.2025	*Виконано*
2.	Опрацювання джерел про виявлення шахрайства в	19.09.2025-07.10.2025	*Виконано*
	блокчейні		
3.	Виконання дослідження щодо виявлення шахрайсь-	09.10.2025-15.10.2025	*Виконано*
	ких дій на основі реальних інцидентів		
5.	Оформлення розділу «Основи безпеки в Web3:	17.10.2025-21.10.2025	*Виконано*
	загрози та рішення»		
6.	Оформлення розділу «Проектування та реалізація	24.10.2025-15.11.2025	*Виконано*
	інструменту»		
7.	Оформлення розділу «Забезпечення захисту	16.11.2025-23.11.2025	*Виконано*
	інформаційних ресурсів системи»		
8.	Виконання завдання до підрозділу «Охорона праці»	23.11.2025-27.11.2025	*Виконано*
9.	Виконання завдання до підрозділу «Захист людини	28.11.2025-3.12.2025	
	від іонізуючого випромінювання»		
10.	Оформлення кваліфікаційної роботи	08.12.2025-10.12.2025	*Виконано*
11.	Нормоконтроль	10.12.2025-15.12.2025	*Виконано*
12.	Перевірка на плагіат	10.12.2025-15.12.2025	*Виконано*
13.	Попередній захист кваліфікаційної роботи	17.12.2025-21.12.2025	*Виконано*
14.	Захист кваліфікаційної роботи	22.12.2025	

Студент _____ Кобельник А. О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Скарга-Бандурова І.С.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка інструменту для виявлення шахрайських дій в блокчейні // ОР «Магістр» // Кобельник Анастасія Олександрівна // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБм-61 // Тернопіль, 2025 // С. 112, рис. – 35, табл. – 3, кресл. – 11, додат. – 2.

Ключові слова: Web3, блокчейн, EVM, шахрайство, транзакція, криптовалюта.

Кваліфікаційна робота присвячена розробці інструменту для виявлення шахрайських дій в блокчейні.

Перший розділ закладає теоретичні основи для розуміння Web3-екосистеми. У ньому розглядаються типи блокчейнів та особливості формування адрес, з акцентом на EVM-сумісні блокчейни які є об'єктом дослідження. Далі аналізуються поширені схеми шахрайства, що становлять ключові загрози для системи. Розділ завершується оглядом наявних аналітичних інструментів, їхніх можливостей та обмежень, що дає змогу виявити прогалини на ринку й сформулювати чіткі вимоги до архітектури нової системи.

Другий розділ присвячено загальному опису архітектури та принципу функціонування системи. У ньому розглянуто структуру програми, пояснено підходи до оцінки ризиків, реалізовано модулі візуалізації поведінки адреси, які дозволяють перетворювати великі масиви даних у доступні для сприйняття графічні та аналітичні форми. Окрім того, проведено тестування готового інструменту.

Третій розділ зосереджено на впровадженні комплексного підходу до забезпечення безпеки. У ньому описано механізми підвищення стійкості API, управління конфіденційною інформацією та захисту від типових загроз, гарантуючи загальну надійність і стабільність роботи системи.

ABSTRACT

Development of a Tool for Detecting Fraudulent Activities in Blockchain // Thesis of educational level "Master" // Kobelnyk Anastasiia // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Cybersecurity, group SBm-61 // Ternopil, 2025 // p. 112, figs. 35, tbls. 3, drws. 11, apps. 2.

Keywords: Web3, blockchain, EVM, fraud, transaction, cryptocurrency.

The qualification work is devoted to the development of a tool for detecting fraudulent activities in blockchain.

The first chapter reveals the theoretical foundations for understanding the Web3 ecosystem. It examines the types of blockchains and features of address formation, with a focus on EVM-compatible networks, which are the object of study. Further, common fraud schemes that pose key threats to the system are analyzed. The chapter concludes with an overview of available analytical tools, their capabilities, and limitations, which helps to identify market gaps and formulate clear requirements for the architecture of the new system.

The second chapter is devoted to a general description of the architecture and the principle of operation of the system. It examines the structure of the program, explains approaches to risk assessment, and implements address behavior visualization modules that allow converting large data sets into understandable graphical and analytical forms. In addition, the finished tool was tested.

The third chapter focuses on implementing a comprehensive security approach. It describes mechanisms for enhancing API resilience, managing confidential information, and protecting against typical threats, ensuring overall reliability and stability of the system.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AML	—	Anti Money Laundering
CEX	—	Centralized Exchange
CTF	—	Countering the Financing of Terrorism
DEX	—	Decentralized Exchange
EVM	—	Ethereum Virtual Machine
FOMO	—	Fear of Missing Out

ЗМІСТ

ВСТУП

Актуальність теми. За даними звіту Chainalysis: Crypto Crime Mid-Year Update 2025 [1], обсяг викрадених криптоактивів тільки за перше півріччя 2025 року на 17 % перевищив показники 2022 року, який до того, за рівнем втрат вважався рекордним. Така динаміка свідчить про стрімке зростання кіберзлочинності у сфері децентралізованих фінансів, та підкреслює, наскільки користувачі технологій блокчейну стають вразливими. Адже на відміну від традиційних фінансових систем, де існують механізми скасування і блокування платежів, у блокчейні транзакції є незворотними, що суттєво ускладнює повернення викрадених коштів і робить своєчасне виявлення шахрайських операцій критично важливим.

Окрему небезпеку становить тенденція до зростання кількості атак, спрямованих безпосередньо на особисті гаманці користувачів. За даними того ж звіту Chainalysis, компрометація індивідуальних гаманців становить 23,35 % від загального обсягу викрадених коштів з початку 2025 року. Це вказує на зміщення фокусу зловмисників із масштабних зламів платформ на персональні атаки, що робить пересічних користувачів найбільш вразливою ланкою. Водночас більшість існуючих аналітичних рішень є комерційними та недоступними для широкого кола користувачів, тому виникає потреба у створенні прозорих та доступних інструментів.

Таким чином, задача створення ефективної, інтегрованої системи виявлення шахрайських дій в блокчейні здатне не лише суттєво знизити ризики і фінансові втрати користувачів, а й стимулювати розвиток відкритої блокчейн-аналітики та посилити довіру до децентралізованих технологій загалом.

Мета та задачі дослідження. Підвищення рівня безпеки користувачів при взаємодії з Web3-екосистемою шляхом створення доступного веб-інструменту, здатного агрегувати релевантні ончейн-дані, візуалізувати патерни транзакційної активності та зв'язків, а також застосовувати методи аналізу для ідентифікації та оцінки ризиків шахрайств, пов'язаних з EVM-адресами.

Для досягнення поставленої мети потрібно виконати наступні завдання:

– Дослідити види блокчейнів, поширені схеми шахрайства та оглянути існуючі інструменти для формування бази знань проекту.

– Спроектувати архітектуру системи, котра міститиме модулі збору інформації про адресу, відображення активів, історії транзакцій, візуалізації хронології активності, взаємодій адреси, оцінки ризику шахрайства і висновку.

– Ідентифікувати ключові ончейн-ознаки різних видів шахрайства та на їх основі створити правила для оцінки ризику.

– Реалізувати ключові модулі візуалізації ончейн даних адреси для комплексної аналітики, інтегрувати ці модулі в єдину систему.

– Реалізувати безпечну взаємодію з API, управління та зберігання ключів доступу, а також заходи протидії типовим веб-атакам для забезпечення захисту системи.

Об’єкт дослідження. Адреси в EVM-блокчейнах, їх транзакційна активність, інформація про їх інвестиційний портфель, взаємодії з іншими адресами та види шахрайства в Web3 екосистемі.

Предмет дослідження. Набір методів аналізу ончейн-даних для виявлення шахрайських патернів в EVM-сумісних блокчейнах.

Наукова новизна одержаних результатів кваліфікаційної роботи полягає в розробці інтегрованої архітектури інтелектуальної системи аналізу EVM-адрес, яка об’єднує в єдиному доступному інструменті повний спектр функцій провідних комерційних платформ, усуваючи фрагментацію та високу вартість існуючих рішень. Система реалізує часткову реплікацію функціоналу дорогих недоступних платформ, доповнюючи його модулями візуалізації та оцінки ризиків, забезпечуючи комплексний аналіз без підписки, з можливістю розгортання в локальному середовищі та масштабування для різних блокчейнів.

Практичне значення одержаних результатів. Розроблений інструмент може бути використаний звичайними користувачами які хочуть зберегти свій капітал. Система дозволяє оцінити рівень ризику взаємодії з конкретною адресою, що допомагає зменшити втрати від шахрайства та підвищити безпеку операцій.

Апробація результатів магістерської роботи. Основні результати проведених досліджень обговорювались на XIII науково-технічній конференції «Інформаційні моделі, системи та технології» (м.Тернопіль), 17-18 грудня 2025 р.

Публікації. Основні результати кваліфікаційної роботи опубліковано у тезах конференції: А. Кобельник, Розробка інструменту для виявлення шахрайських дій в блокчені // XIII науково-технічна конференція «Інформаційні моделі, системи та технології» (м.Тернопіль), 17-18 грудня 2025 р. (див. Додаток А).

РОЗДІЛ 1. ОСНОВИ БЕЗПЕКИ В WEB3: ЗАГРОЗИ ТА РІШЕННЯ

1.1 Види блокчейнів і особливості адрес

Архітектура блокчейнів відрізняється залежно від рівня доступу користувачів та ступеня децентралізації, що дозволяє виділити чотири основні типи:

– Приватні (Private). Контролюються організацією, доступ обмежений.

– Консорціумні (Consortium). Керуються групою організацій, доступ обмежений або надається за дозволом.

– Публічні (Public). Доступні для всіх, будь-хто може переглядати транзакції.

– Гібридні (Hybrid). Поєднують елементи приватних і публічних блокчейнів [2].

Оскільки мета даної роботи – розробка інструменту доступному пересічному користувачу, основна активність яких відбувається в блокчейнах Bitcoin, Ethereum та ін., подальше дослідження буде сфокусоване на публічних блокчейнах. Також з огляду багатошарової архітектури блокчейнів, аналіз буде здійснюватись виключно на Layer 1 та Layer 2, оскільки шари Layer 0, який відповідає за базову інфраструктуру блокчейну та Layer 3, який відповідає за взаємодію користувачів з додатками, не містять даних які можна відслідкувати в блокчейні.

Layer 1 (L1) – основний шар блокчейну, на якому визначається базова модель обліку, консенсус [3], структура даних і формат адрес, наприклад: Bitcoin, Ethereum, Solana та інші [4].

Layer 2 (L2) – це протоколи та надбудови, що працюють поверх Layer 1, призначені для збільшення пропускної здатності та зниження вартості транзакцій. Вони виконують обчислення і транзакції поза основним блокчейном, періодично фіксуючи стан на L1 для підтвердження транзакцій.

Таким чином, вони завжди успадковують фундаментальну модель блокчейну на якому вони побудовані. Така архітектурна особливість є важливою, оскільки вона дозволяє згрупувати L1 та їхні L2-рішення під єдиною логікою

збору даних. Це звужує вибір публічних блокчейнів, до найпоширеніх моделей можна віднести:

Модель UTXO (Unspent Transaction Output) є найстарішою формою обліку, що лежить в основі Bitcoin. Транзакція функціонує як операція, що використовує старі виходи (UTXO) як входи і створює нові виходи, які можна витратити в майбутньому. Для аналітики це означає, що відстежувати потрібно не баланс, а потік «монет» по графу транзакцій.

Формування адреси у моделі UTXO (як у Bitcoin) є процесом, який перетворює ваш публічний ключ на короткий, зручний для обміну псевдонім. Спочатку генерується пара криптографічних ключів (приватний/публічний). Потім публічний ключ проходить подвійне хешування (SHA256, а потім RIPEMD160), щоб отримати компактний ідентифікатор. До цього ідентифікатора додаються версійний префікс та контрольна сума для перевірки помилок. Нарешті, весь цей бінарний рядок кодується за допомогою алгоритму Base58Check, що перетворює його на кінцеву, легку для читання адресу, яку можна надати іншим для отримання коштів. Адреса, по суті, є лише публічним місцем призначення, яке доводить право власника витрачати невитрачені виходи транзакцій (UTXO), пов'язані з нею [5].

Основними представниками є Bitcoin, Bitcoin Cash, Litecoin, Cardano та Dogecoin, формування адрес яких подано на рисунку 1.1 [6].

Блокчейн	Bitcoin				Bitcoin Cash		Litecoin			Dogecoin	Cardano
Модель обліку	UTXO				UTXO		UTXO			UTXO	EUTXO
Тип скрипта	P2PKH	P2SH	Bech32	P2TR	Legacy	Cashaddr	P2PKH	P2SH	Bech32	Legacy	Base
Префікс	1	3	bc1	bc1p	1 or 3	q or p	L	M	Ltc1	D or A	addr

Рисунок 1.1 – Префікси адрес у блокчейних моделі UTXO та EUTXO

Account-based модель є домінуючою архітектурою для блокчейнів, що підтримують складні смарт-контракти, і її еталоном є Ethereum (ETH). На відміну від UTXO, ця модель підтримує глобальний стан, у якому кожна адреса (акаунт) має єдиний оновлюваний баланс. Це спрощує розробку, оскільки транзакції є командами, які змінюють стан акаунту (наприклад, переказ коштів або виклик функції контракту), подібно до традиційного банківського рахунку.

У цій моделі існують два головні об'єкти: EOA (Externally Owned Accounts), що контролюються приватним ключем користувача, та Contract Accounts, що контролюються кодом і є джерелом більшості DeFi-операцій [7].

Адреса Ethereum – це шістнадцяткова адреса з 42 символів, з додаванням префікса 0x попереду, отримана шляхом застосування криптографічної хеш-функції Keccak-256 до публічного ключа користувача, з яких береться останніх 20 байтів цього хешу. Приклад EVM адреси: 0x983873529f95132BD1812A3B52c98Fb271d2f679, ця «публічна» адреса знадобиться для отримання коштів від іншої сторони, а щоб отримати доступ до коштів на цій адресі, потрібен її закритий ключ [8].

Смарт-контракт – це програма, що зберігається та виконується безпосередньо в блокчейні. Він являє собою автоматизовану угоду: умови контракту заздалегідь закодовані, і виконання цих умов контролюється кодом, а не довірою до сторін. Смарт-контракти є основою децентралізованих фінансів, оскільки вони можуть керувати коштами, зберігати дані, випускати токени і виконувати складну логіку без втручання людини. Адреса смартконтракту залежить від адреси розгортача (EOA або іншого контракту) та його nonce – кількості вже здійснених транзакцій. Ці два значення кодуються у форматі RLP, після чого до них застосовується хеш-функція Keccak256. З отриманого хешу беруть останні 20 байт, що і стають адресою смартконтракту. Таким чином, для кожної пари (адреса розгортача + nonce) адреса контракту завжди буде однаковою, забезпечуючи унікальність [9].

Представники: Ethereum, BNB Smart Chain, Polygon, Arbitrum, Optimism.

Спеціалізовані та унікальні моделі – такі, які представляють нове покоління Layer 1 блокчейнів, які відмовилися від прямої EVM-сумісності на користь власних архітектур, орієнтованих на підвищену безпеку або безпрецедентну пропускну здатність.

Прикладом таких є акаунт-центрична модель Solana – це архітектура блокчейну, де вся інформація (стан, дані, код програм) зберігається в акаунтах (accounts), які є незалежними одиницями даних, подібними до записів у публічній базі даних з єдиною таблицею «Accounts». Кожен акаунт має

унікальну 32-байтну адресу (зазвичай публічний ключ Ed25519, закодований у Base58, наприклад, 34ioGHB37343kMxFD12zfnaEAuo5JGfD76PcULPfso2S), що слугує ключем для доступу до даних на ланцюжку [10].

Важливий аспект реалізації цього дослідження полягає у забезпеченні доступності та масштабованості рішення. На даному етапі, впровадження універсального інструменту, який би підтримував усі існуючі моделі блокчейнів (такі як UTXO, Account-based EVM та спеціалізовані Solana-подібні архітектури), вимагало б інтеграції з численними, часто платними та обмеженими API-сервісами для кожного унікального формату адреси та структури транзакцій. Це призвело б до значних витрат і обмежило б можливість запропонувати інструмент широкому колу користувачів безкоштовно.

Таким чином, хоча в екосистемі блокчейну існують різні архітектури, подальше дослідження буде сфокусоване виключно на EVM-сумісних блокчейнах. Цей вибір обґрунтований трьома ключовими факторами, що забезпечують практичну доцільність проєкту:

– EVM-блокчейни є домінуючою платформою для Децентралізованих Фінансів (DeFi) та ринку токенів, які є основними цілями шахрайства.

– Єдиний формат EVM-адреси (42-символьна шістнадцяткова адреса з префіксом 0x) є уніфікованим для десятків Layer 1 та Layer 2 блокчейнів, що дозволяє використовувати єдиний підхід до валідації та збору даних.

– Для EVM-сумісних блокчейнів існує значно ширший спектр безкоштовних або високодоступних публічних API, що важливо для розробки інструменту, який користувач зможе використовувати без обмежень.

З огляду на ці фактори, подальший аналіз, проєктування системи та впровадження механізмів безпеки будуть сфокусовані виключно на EVM-сумісних блокчейнах.

1.2 Поширені схеми шахрайства в блокчейні

Для ефективного виявлення шахрайських транзакцій вкрай важливо розуміти природу та механізми найпоширеніших афер. Ці схеми часто

використовують психологічні маніпуляції, технічні вразливості та анонімність, яку надає блокчейн. Нижче наведено опис основних видів шахрайства.

Rug Pull. Це вид шахрайства у сфері децентралізованих фінансів (DeFi), коли розробники криптовалютного проєкту, залучивши кошти інвесторів, раптово згортають усю діяльність та зникають з цими коштами. Вони буквально «витягують килим ліквідності» з-під ніг інвесторів, залишаючи їх з абсолютно знеціненими токенами.

Реалізація. Спочатку шахраї створюють новий токен на блокчейні з низькими комісіями та розміщують його на децентралізованій біржі (DEX), створюючи пул ліквідності зі своїм токеном та популярним активом (наприклад, SOL). Далі, через агресивний маркетинг у соцмережах з гучними обіцянками, вони створюють штучний ажіотаж (FOMO), спонукаючи інвесторів масово скуповувати токен. Кожна покупка підвищує ціну та наповнює пул ліквідності цінними активами інвесторів. У момент, коли в пулі накопичується значна сума, шахраї однією транзакцією виводять звідти весь цінний актив. В результаті ціна їхнього токена миттєво падає, зловмисники зникають, видаляючи сайт та соціальні мережі разом із вкраденими коштами.

Ще одним підвидом rug pull є обмеження ордерів на продаж (limit sell orders), в якому творці створюють смарт контакт із вбудованими обмеженнями продажу, що означає, що тільки вони можуть продавати токени. Звісно, що більшість цього не знає, і після того, як інвестори купують багато токенів, розробники просто отримують прибуток, продаючи всі ці активи та залишаючи людям нічого не вартий актив [11, 12].

Основні ознаки (Features):

- Раптовий сплеск вхідних транзакцій.
- Аномально високий обсяг виводів від creator/owner.
- Dump (падіння) у нетипові години чи великі операції одразу після піку.
- Відсутність Liquidity Lock – токени не заблоковані в пулі (високий ризик зняття ліквідності).
- Раптове видалення ліквідності – одна транзакція видаляє > 90% пулу.
- Контракт не верифікований на explorer, що ускладнює аудит.

– Мала кількість унікальних холдерів при великому обсязі торгів.

– LP провайдери без історії транзакцій – фантомні гаманці, завчасно створені для маскування.

– Повторювані дзеркальні операції – wash-trading між набором адрес.

– Токен безперервно росте та не падає.

Sweeper Bot Attacks – це автоматизовані скрипти (боти), які моніторять compromised гаманці (де зловмисник має доступ до приватних ключів, але гаманець порожній щодо нативного токена для газу). Бот чекає на мінімальний депозит нативного токена (наприклад, ETH, BNB, SOL), необхідний для оплати комісії, і миттєво виводить усі цінні на адресу зловмисника, використовуючи високий пріоритет транзакції для випередження власника. Це різновид drainer-атак, де автоматизація робить крадіжку ефективною [13].

Основні ознаки (Features):

– Дуже швидкий вивід (кілька секунд після надходження газу), гаманець довго «спав» (без транзакцій >1 дня), а потім різкий сплеск.

– Бот платить високу комісію (наприклад, >50 gwei), щоб його транзакція була першою.

– Після депозиту гаманець стає порожнім (>90% активів виводиться).

– Адреса шахрая має багато вхідних транзакцій, адже всі активи йдуть на одну адресу шахрая, після чого кошти можуть іти далі до міксера.

– Адреса шахрая нова (<1 день).

– Мала кількість (<50) або немає попередніх транзакцій.

Token Approvals and Wallet Drainers. Користувачі можуть надавати смарт-контрактам дозволи (approvals), які дозволяють цим контрактам списувати певні токени з їхніх гаманців від імені власника. Така механіка є фундаментальною для роботи DEX, маркетплейсів NFT та інших DeFi-сервісів, оскільки дозволяє автоматизувати обмінні операції без повторного ручного підтвердження кожної транзакції.

Водночас існує суттєвий ризик – необмежені (infinite) дозволи створюють привілейовану можливість доступу до активів користувача. У випадку, якщо контракт виявиться шкідливим, його приватні ключі або логіка можуть бути

скомпрометовані, або якщо користувач помилково схвалить фішинговий dApp, атакуюча сторона може негайно і в повному обсязі списати всі дозволені токени. Отже, механізм approval хоча і є функціонально необхідним, при неналежному управлінні суттєво підвищує операційний ризик втрати активів [14].

Основні ознаки (Features):

– Необмеженні approve-транзакції.

– Збільшення кількості approve-транзакцій за короткий проміжок часу.

– Швидкий вивід активів після approve токена.

– Гаманець надає дозвіл контракту, з яким раніше не взаємодіяв.

– Контракт, що отримує дозвіл, є нещодавно створеним та не верифікованим.

Отруєння адреси (Address Poisoning). Ця атака націлена на неуважність користувача при копіюванні адреси з історії транзакцій. Шахрай створює адресу гаманця, яка візуально дуже схожа на адресу, якою часто користуються, зазвичай збігаються перші та останні 5-6 символів. Потім він надсилає з цієї «отруєної» адреси транзакцію з нульовою вартістю. Ця транзакція з'являється у історії жертви. Наступного разу, коли жертва захоче відправити кошти, вона може машинально скопіювати з історії цю адресу-двійник і відправити гроші шахраю [15].

Основні ознаки (Features):

– Отримання кількох малих вхідних транзакцій від різних адрес (zero-value або low-value tx).

– Високий ступінь схожості адреси шахрая з адресою отримувача.

– Низька загальна активність адреси (кількість транзакцій <50).

– Малий час існування шахрайської адреси (<тижня).

– Сплеск пилових транзакцій (dust tx) у короткий період – повторювані tx від однієї адреси.

– Аномально малі вхідні суми з подальшим великим drain-ом після помилкового переказу.

Відмивання грошей (Money Laundering) – це процес приховування походження незаконних коштів та їх обмін, шляхом використання складного

ланцюга блокчейн-транзакцій. Головна мета – розірвати ончейн зв'язок між початковою «брудною» адресою (гаманцем шахрая) та кінцевою «чистою» адресою, з якої кошти можна буде вивести у фіат на легальній біржі. Типова структура відмивання грошей:

На першому етапі злочинці вже володіють криптовалютою, отриманою внаслідок незаконної діяльності – наприклад, після зламу, шахрайства або продажу нелегальних товарів.

Розшарування (Layering) – це найважливіший етап, де відбувається активне заплутування слідів. Зловмисники використовують кілька методів:

– Міксери (Mixers). Зловмисник відправляє «брудні» кошти на смарт-контракт міксера (наприклад, Tornado Cash). Цей контракт змішує його монети з монетами десятків інших користувачів, а потім дозволяє вивести ту саму суму на абсолютно нову, «чисту» адресу.

– Chain-hopping/Bridge-hopping. Кошти переказуються з одного блокчейну на інший за допомогою кросчейн мостів. Наприклад, вкрадені ETH з Ethereum переводяться в обгорнутий токен на Solana, потім обмінюються на BTC через DEX і виводяться через блокчейн Bitcoin. Кожен «стрибок» значно ускладнює відстеження, оскільки вимагає аналізу кількох незалежних реєстрів.

– Пілінгові ланцюжки (Peel Chains). Велика сума дробиться на безліч дрібніших частин, які проходять через сотні або тисячі проміжних, одноразових гаманців. Кожен такий гаманець отримує кошти, «відщипує» невелику частину на іншу адресу, а залишок відправляє далі по ланцюжку. Це створює надзвичайно складний і заплутаний граф транзакцій.

– Smurfing/Structuring – розбиття великої суми на тисячі дрібних переказів з метою уникнути уваги та порогових правил моніторингу.

– Використання приватних монет. «Брудні» BTC або ETH обмінюються на анонімні криптовалюти, де транзакції є конфіденційними. Після кількох переказів всередині мережі Monero, кошти обмінюються назад на публічну криптовалюту і виглядають як такі, що прийшли з невідомого джерела.

– Використання DEX/OTC/CEX – послідовні свопи на децентралізованих біржах і подальші виводи через позабіржові (OTC) або централізовані біржі (CEX) для інтеграції в легальний ринок (часто із KYC).

– NFT-обходи (wash-mint-burn) – використання NFT як засобу переміщення вартості і приховування походження коштів.

Інтеграція. «Відмиті» кошти, які вже не мають явного зв'язку зі злочином, депонуються на великі централізовані біржі (часто порціями) для конвертації у фіат або змішування з легальними активами [16].

Основні ознаки (Features):

– Прямі транзакції до/з відомих міксерів чи адрес які внесені до чорних списків (санкційні списки, хакерські угруповання, адреси програм-вимагачів).

– Взаємодія з кросчейн мостами, особливо якщо одразу після отримання коштів на новому блокчейні знову проходять через інший міст чи міксер.

– Структура «Peel Chains» – гаманець отримує велику суму, а потім створює довгий ланцюжок транзакцій до нових, раніше неактивних гаманців.

– Структурування (Smurfing) – розбиття великої суми на безліч дрібних, ідентичних або близьких за розміром транзакцій, що надсилаються на різні адреси, щоб уникнути порогових значень моніторингу.

– Проведення сотень транзакцій через ланцюжок проміжних гаманців за надзвичайно короткий проміжок часу, що вказує на використання скриптів.

Wash trading – це форма маніпуляції ринком, коли один і той самий трейдер (або група пов’язаних адрес) одночасно купує і продає один і той самий актив, створюючи ілюзію високої торгівельної активності чи попиту. Мета – штучно підняти обсяг торгів або ціну активу, щоб ввести в оману інших користувачів чи алгоритми бірж. Це часто використовується для накрутки статистики проектів або для відмивання коштів під виглядом «торгівлі» [17].

Основні ознаки (Features):

– Повторювані транзакції між одними й тими самими адресами.

– Однакова ціна й обсяг у кількох послідовних угодах.

– Відсутність реального приросту унікальних учасників чи ліквідності.

– Аномально висока частота торгів за короткий проміжок часу.

– Активні гаманці мають одного власника.

– Необгрунтовано високий обсяг торгів без ринкових новин.

Pump & Dump. Це схема маніпуляції ринком, у якій організатори спочатку штучно піднімають ціну певного токена (pump) через масові покупки або агресивний маркетинг, створюючи FOMO (страх пропустити прибуток). Коли ціна досягає бажаного рівня, організатори різко продають свої токени (dump), спричиняючи падіння ціни і залишаючи інших інвесторів із збитками.

Типовий життєвий цикл включає чотири життєві стадії: pre-launch, в якому підігрівають інтерес до проекту. Launch, в якому залучають промоутерів, інфлюенсерів які в свою чергу допомагають залучити більше потенційних жертв. Pump – ціна активу стрімко зростає, адже його активно купують. Dump – фаза обвалу ціни, в якій організатори розпродають свої активи, як тільки ціна токена досягне того рівня, який вони вважають прибутковим. Цей масовий розпродаж призводить до того, що пропозиція токена значно перевищує попит, тим самим знижуючи ціну [18].

Основні ознаки (Features):

– Аномальний, експоненційний ріст обсягу торгів та ціни токена, який відбувається без будь-яких фундаментальних новин, партнерств або оновлень проєкту. Цей ріст є неорганічним і надто стрімким.

– Низька ліквідність до початку «пампу». Токен, обраний для маніпуляції, зазвичай має дуже низьку ліквідність. Це дозволяє зловмисникам значно впливати на ціну навіть невеликими обсягами купівель.

– Різке збільшення кількості унікальних покупців (адрес), що купують токен у короткий проміжок.

– Різкий продаж монет з гаманців творців після пікового зростання, часто це одна чи кілька величезних транзакцій.

– Низька ліквідність після піку – пул ліквідності падає, висока волатильність, велика просадка.

– Короткі інтервали між покупкою і продажем, торгівля поза типово активними годинами, сплески торгів в нетиповий час.

– Скоординована купівля, де на етапі Pump-у значна частина купівель здійснюється невеликою групою гаманців, які діють майже одночасно (це можуть бути група власників, так і інсайдери).

– Гаманці, які були найбільш активними на етапі Pump-у, є першими і найбільшими продавцями на етапі продажу Dump-у.

Ponzi Scheme – це фінансова піраміда, яка існує доти, доки з'являються нові вкладники, далі система руйнується, а останні інвестори втрачають усі свої кошти [19].

Основні ознаки (Features):

– Граф виглядає як піраміда – багато нових гаманців надсилають дрібні суми наверх до організаторів, які розподіляють частину вниз, щоб підтримувати довіру, а решту ховають через кілька переказів, що легко виявити за допомогою аналізу транзакцій.

– Пряма залежність виплат від вкладів. Аналіз показує, що виплати старим інвесторам (withdraw) відбуваються майже одразу після отримання великих депозитів (deposit) від нових.

– Відсутність зовнішнього джерела доходу. Смарт-контракт не взаємодіє з легітимними джерелами прибутку (DeFi-протоколами, біржами, лендинговими платформами). Його єдиним джерелом поповнення є кошти від зовнішніх гаманців (EOA).

– Після припинення притоку нових учасників граф «застигає», нові вузли не додаються, а кошти, що надійшли останніми виводяться на кілька контрольних адрес.

Dusting attacks є методом ончейн розвідки, спрямованим на деанонімізацію та порушення приватності власників криптовалютних адрес. Зловмисник розсилає мізерні суми криптовалюти («пил» або dust) на велику кількість гаманців, для подальшого відстеження руху їх активів. Ця загроза найбільш актуальна для блокчейнів побудованих за UTXO-моделлю (Unspent Transaction Output), такою як Bitcoin. Власник гаманця, створюючи нову транзакцію, об'єднує декілька входів (UTXO), включно із «запиленим». За допомогою графового аналізу встановлюється зв'язок між усіма адресами, з яких походили

ці входи, що дозволяє побудувати профіль активності користувача та оцінити його сукупний баланс. Таким чином, «запилення» не є спробою прямого викрадення коштів, однак виступає інструментом для збору даних, для майбутніх цілеспрямованих атак [20].

– Економічно незначущий розмір транзакції. Сума, що надсилається, є надзвичайно малою (часто менше, ніж вартість стандартної комісії за транзакцію) і не має практичного економічного сенсу.

– Одна адреса-відправник створює велику кількість вихідних транзакцій з однаковою або схожою мікро-сумою на безліч непов'язаних між собою адрес-одержувачів, формуючи на графі патерн «один-до-багатьох».

– Використання нативної криптовалюти. Зазвичай для «запилення» використовується нативна валюта блокчейну (BTC, ETH), а не токени, оскільки її рух легше інтегрувати в майбутні транзакції жертви для сплати комісій.

Airdrop scam tokens – це форма атаки, що базується на методах соціальної інженерії, кінцевою метою якої є викрадення цінних активів з гаманця жертви. В основі схеми лежить роздача аірдропу безкоштовних токенів від імені нового або вже існуючого проєкту. Шахраї використовують фішингові веб-сайти або децентралізовані додатки (dApps), імітуючи офіційні ресурси. Метою яких є змусити користувачів переходити на них, щоб забрати «примарну» нагороду. Однак, замість того, щоб ініціювати транзакцію отримання токенів, dApp формує запит на підписання шкідливої транзакції, яка надає дозвіл на списання всіх активів жертви [21].

Основні ознаки (Features):

– Контракт токена створено нещодавно, і його вихідний код не верифіко-ваний на блокчейн-експлорері.

– Запити на сплату газу (fees) чи інших малих сум для отримання airdrop.

– Групові транзакції (batch-transfers), що виводять активи в одній операції після підключення до dApp.

– Велика кількість вхідних транзакцій до шахрайського контракту чи гаманця від множини адрес жертв, що формує зіркоподібну структуру.

– Низький коефіцієнт кластеризації, що вказує на ізольовані адреси жертв без взаємозв’язків між собою.

Проведений аналіз охоплює ключові ончейн ознаки поширених шахрайських схем. Однак, слід усвідомлювати, що цей перелік не є вичерпним, оскільки кількість загроз у блокчейнах постійно еволюціонує. Тому відстежити всі вектори атак неможливо. Таким чином, розроблюваний інструмент буде мати відкритий вихідний код. Цей підхід дозволить будь-якому користувачу робити свій внесок, доповнюючи систему новими патернами для ідентифікації виявлених видів шахрайства.

Отож, планується створити не статичну, а живу, керовану спільнотою систему.

1.3 Огляд існуючих інструментів та їх обмеження

Зі зростанням популярності блокчейн-технологій та децентралізованих фінансів (DeFi) виникла гостра потреба у створенні спеціалізованих аналітичних інструментів. Ці інструменти покликані допомогти користувачам, розробникам та фінансовим установам орієнтуватися у складній ончейн-екосистемі, виявляти підозрілу активність, відстежувати рух цифрових активів та оцінювати ризиковість операцій. Сучасний ринок пропонує широкий спектр рішень – від базових моніторингових сервісів до складних платформ для криміналістичних розслідувань. Однак аналіз показує, що жоден із наявних сервісів не є універсальним, кожен має власні обмеження щодо функціональності, точності, прозорості алгоритмів чи доступності для різних категорій користувачів.

Наявні інструменти можна умовно поділити на кілька категорій, кожна з яких має свої сильні сторони та функціональні обмеження:

Мультичейн-трекери портфеля – це інструменти для відстеження активів користувача на кількох блокчейнах у реальному часі, що забезпечують комплексний моніторинг інвестицій. Вони дозволяють відображати баланси, токени, NFT, стейкінг, участь у DeFi-протоколах і транзакційну історію.

Такі платформи, як Blockscan, DeBank, Zapper, Zerion, CoinStats, CoinTracker і підтримують широкий спектр блокчейнів – від популярних EVM-сумісних (Ethereum, Polygon, Arbitrum) до non-EVM (Solana, Cardano, Near). Інструменти інтегруються з гаманцями біржами, надаючи функціонал для розрахунку прибутків/збитків (P&L), оцінки загальної втрати, аналізу вартості портфеля та генерації звітів для податкових цілей. Інтерфейси пропонують інтуїтивні дашборди з візуалізаціями, гістограми для історичних даних і алерти про значні зміни балансів (див. рисунок 1.2).

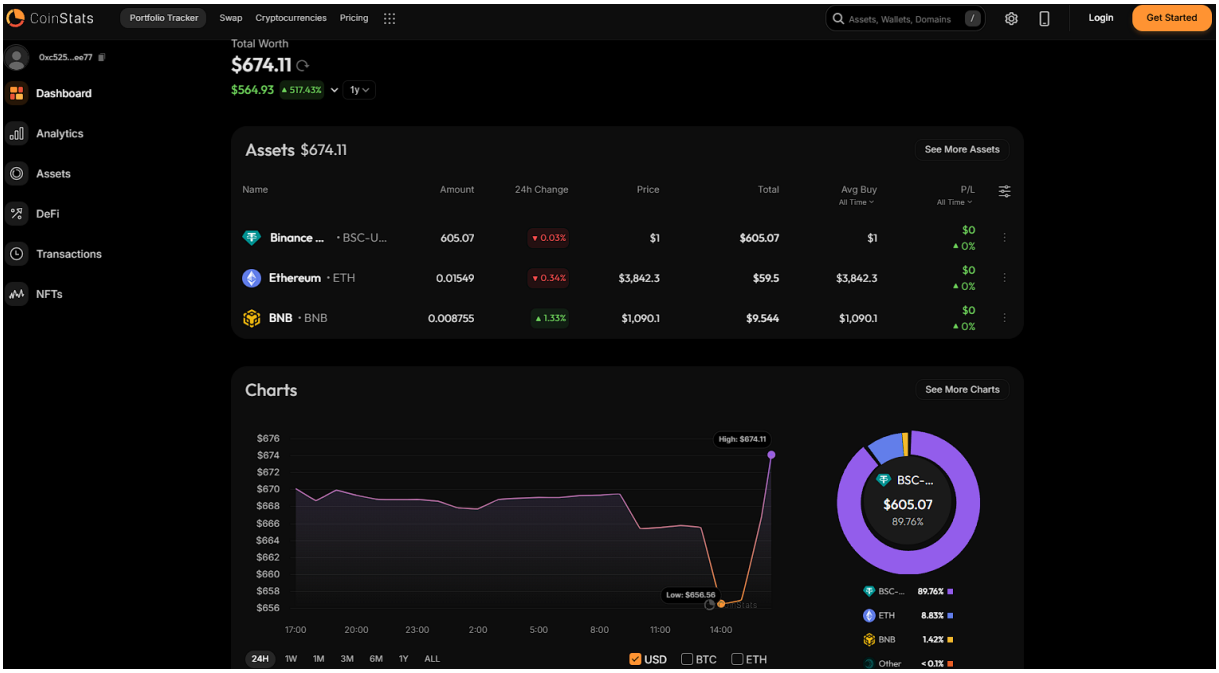

Рисунок 1.2 – Пошук адреси через трекер CoinStats

Деякі платформи, підтримують розширені DeFi-метрики (APY, TVL), а API-інтеграція дозволяє кастомізувати відображення даних для розробників.

Платформи ончейн-аналітики призначені для поглибленого аналізу блокчейн-даних, надаючи користувачам інструменти для дослідження транзакцій, балансів, метрик DeFi, NFT-ринків, міжланцюгових мостів та поведінки великих гравців (whale tracking). Dune Analytics, Nansen, Glassnode підтримують широкий спектр блокчейнів, охоплюючи десятки ланцюгів. Вони дозволяють створювати кастомні SQL-запити для аналізу даних, генерувати дашборди з візуалізаціями, відстежувати активність смарт-контрактів і моніторити «smart money» (великі транзакції від венчурних фондів чи китів). Nansen і Hubble AI,

використовують AI для прогнозування ринкових трендів і виявлення патернів (наприклад, покупка токенів перед лістингом). Dune Analytics підтримує спільні дашборди, створені ком'юніті, що ідеально для досліджень. Більшість платформ інтегруються з API для автоматизації та дозволяють експортувати дані для зовнішньої обробки.

На рисунку 1.3 подано аналіз адреси за допомогою платформи Nansen.

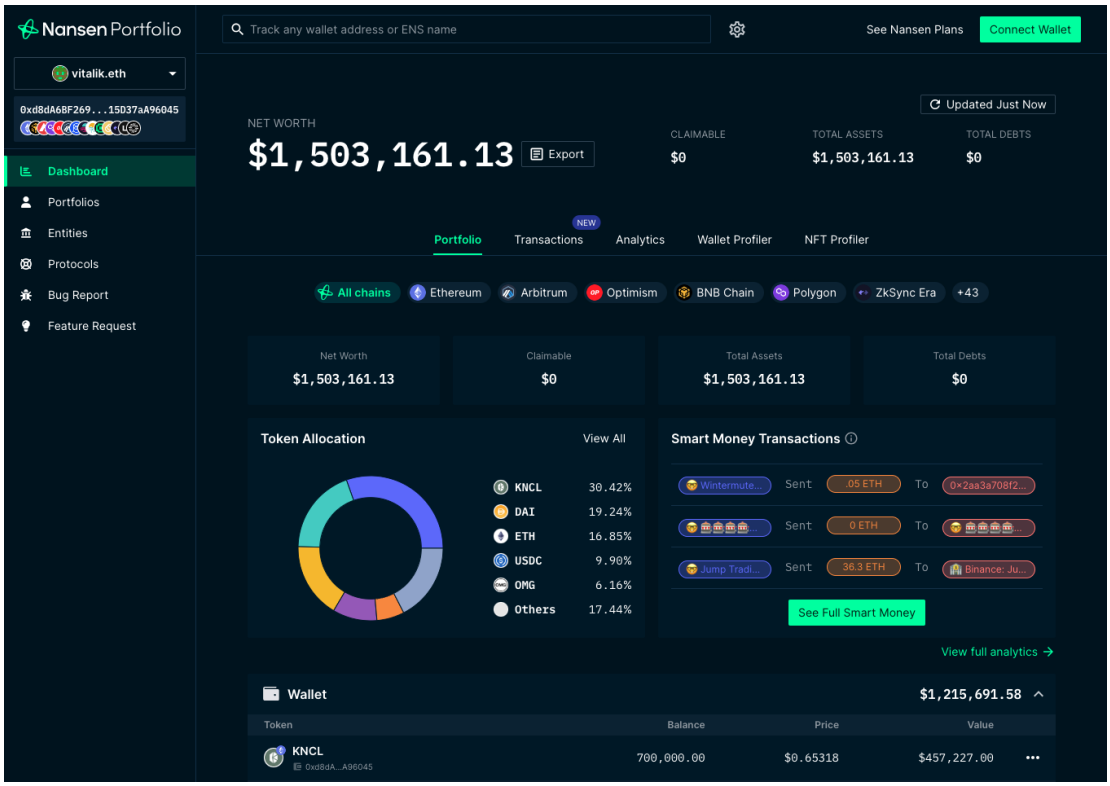

Рисунок 1.3 – Аналіз адреси за допомогою Nansen

Основними обмеженнями таких інструментів є:

– Висока вартість доступу до повних функцій. Преміум-підписки можуть коштувати сотні доларів на місяць, що обмежує використання для індивідуальних користувачів.

– Потреба у технічних знаннях. Створення кастомних дашбордів вимагає навичок програмування, що ускладнює використання для новачків без досвіду.

– Високе ресурсне навантаження. Обробка великих наборів даних вимагає значних обчислювальних ресурсів, що може призводити до повільної роботи без хмарної інфраструктури.

Інструменти для моніторингу та розслідування транзакцій KYT (Know Your Transaction) та AML (Anti-Money Laundering) призначені для моніторингу транзакцій в реальному часі, оцінки ризиків, трасування шляхів коштів та забезпечення відповідності регуляторним стандартам. Такі платформи, як Chainalysis, Elliptic, TRM Labs, Scorechain, CipherTrace та AMLBot підтримують велику кількість блокчейнів та пропонують ризик-скоринг транзакцій, кластеризацію адрес, виявлення патернів, графову візуалізацію блокчейнів з деталями, надсилати оповіщення в реальному часі про рух коштів та автоматизовані звіти.

На рисунку 1.4 показано приклад роботи з Chainalysis Reactor, в якому відстежується потік 12,49 BTC до Black Host, виявляючи зв’язок із Lazarus Group через аналіз транзакцій [22].

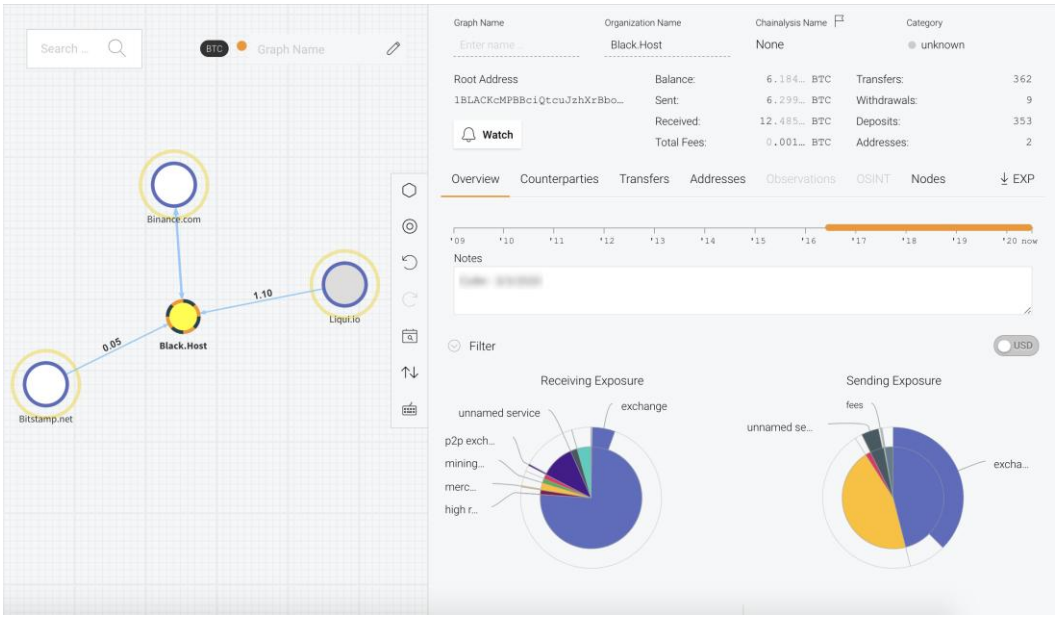

Рисунок 1.4 – Відстеження btc через Chainalysis Reactor

Однак, попри їхні значні можливості, ці інструменти мають низку суттєвих обмежень, які важливо враховувати:

– Недоступність та ціна. Це закриті корпоративні рішення з дуже високою вартістю, недоступні для звичайних користувачів.

– Вузький фокус. Основна увага приділяється ризикам AML/CFT та зв'язкам з відомими злочинними організаціями, санкційними списками,

даркнетом. Вони можуть не покривати весь спектр шахрайства, цікавий звичайним користувачам DeFi.

– Низька прозорість. Алгоритми таких систем є комерційною таємницею. Користувачі бачать лише кінцевий висновок, але не розуміють, які саме фактори та з якою вагою вплинули на результат. Це знижує довіру, ускладнює аудит для регуляторів та оскарження помилкових, а також не дозволяє зібрати чітку доказову базу для розслідувань.

– Користувачі повністю залежать від постачальника щодо оновлення алгоритмів, додавання підтримки нових блокчейнів чи оновлення баз даних (санкційні списки). Будь-які зміни потребують звернення до вендора, що збільшує час та витрати.

– Регуляторні та локальні вимоги. Фіксована логіка інструментів може не відповідати специфічним регуляторним вимогам різних без значних доопрацювань з боку постачальника.

– Обмеження адаптивність до нових загроз. Оскільки алгоритми закриті, адаптація до нових, швидкозмінних шахрайських схем відбувається повільно, оскільки залежить від циклу оновлень постачальника. Відсутність доступу до коду не дозволяє користувачам або спільноті швидко реагувати на нові загрози шляхом модифікації моделей чи правил.

Розширення безпеки для браузерів та симулятори транзакцій Web3 від фішингу, шкідливих смарт-контрактів, дренажів гаманців та помилкових підписів транзакцій, дозволяють аналізувати та моделювати операції перед їх виконанням. Такі інструменти, як Pocket Universe, AegisWeb3, Wallet Guard, Revoke.cash, Web3 Antivirus (W3A) (див. рисунок 1.5) працюють як плагіни для популярних браузерів інтегруючись з гаманцями (MetaMask, Phantom). Вони пропонують симуляцію транзакцій для перегляду потенційних наслідків (наприклад, втрата активів, доступ до токенів), сканування URL на фішинг (перевірка доменів, блокування шкідливих сайтів, revoke, алерти про ризики, аналіз смарт-контрактів та моніторинг підписів в реальному часі.

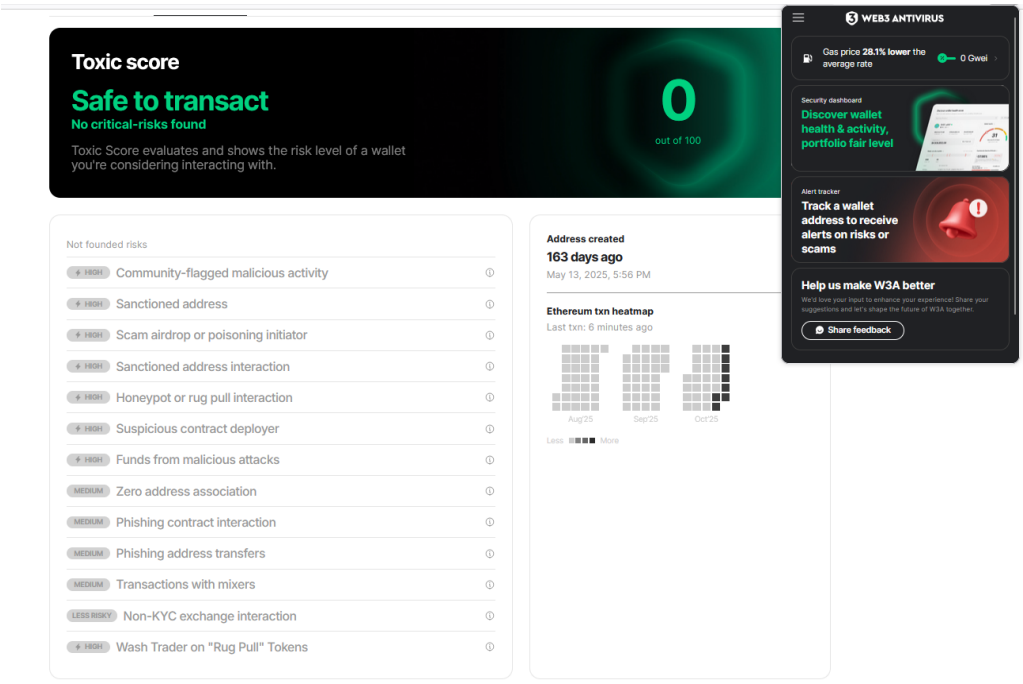

Рисунок 1.5 – Сканування адреси за допомогою Web3 Antivirus

Проте всі вони мають обмеження:

– Обмежена сумісність із браузерами. Більшість розширень доступні лише для Chrom-браузерів, але не підтримують Firefox чи Safari, що обмежує їх використання для користувачів інших платформ.

– Помилкові алерти (false positives). Симулятори транзакцій можуть позначати легітимні транзакції як підозрілі.

– Залежність від оновлень баз шкідливих сайтів і контрактів. Ефективність блокування фішингу чи дренажів залежить від актуальності баз даних, які можуть не встигати за новими загрозами.

– Вразливість розширень до атак. Фейкові Web3-розширення, які імітують легітимні інструменти, створюють ризик компрометації через фішинг, дозволяючи зловмисникам отримувати доступ до гаманців користувачів.

– Обмежена мобільна інтеграція. Більшість розширень орієнтовані на десктопні браузери, а підтримка мобільних гаманців чи браузерів часто обмежена або менш функціональна.

– Ресурсомісткість симуляцій. Аналіз транзакцій в реальному часі може сповільнювати браузер на слабких пристроях через складні обчислення для перевірки смарт-контрактів.

Сканери блокчейнів – це інструменти, які надають користувачам інтерфейс для перегляду та аналізу даних у блокчейнах, забезпечуючи прозорий доступ до транзакцій, адрес гаманців, смарт-контрактів, токенів, NFT та інших ончейн-даних. Вони діють як пошукові системи для блокчейнів, дозволяючи користувачам відстежувати деталі транзакцій (TXID, суми, часові мітки, gas fees), перевіряти баланси адрес, аналізувати активність смарт-контрактів, а також переглядати метадані блоків (хеші, майнери, винагороди).

Популярні експлорери, такі як Etherscan, BscScan, Solscan, PolygonScan, BitInfoCharts, Cardano Explorer, TronScan, Suiscan, NEAR Explorer, Aptos Explorer та ін., пропонують широкий функціонал: від базового пошуку транзакцій до аналітики, інтеграції з API для розробників та верифікації смарт-контрактів (з відкритим кодом). Деякі експлорери, як Etherscan, дозволяють користувачам додавати коментарі та теги до адрес.

На рисунку 1.6 подано скріншот результату пошуку адреси в Etherscan.

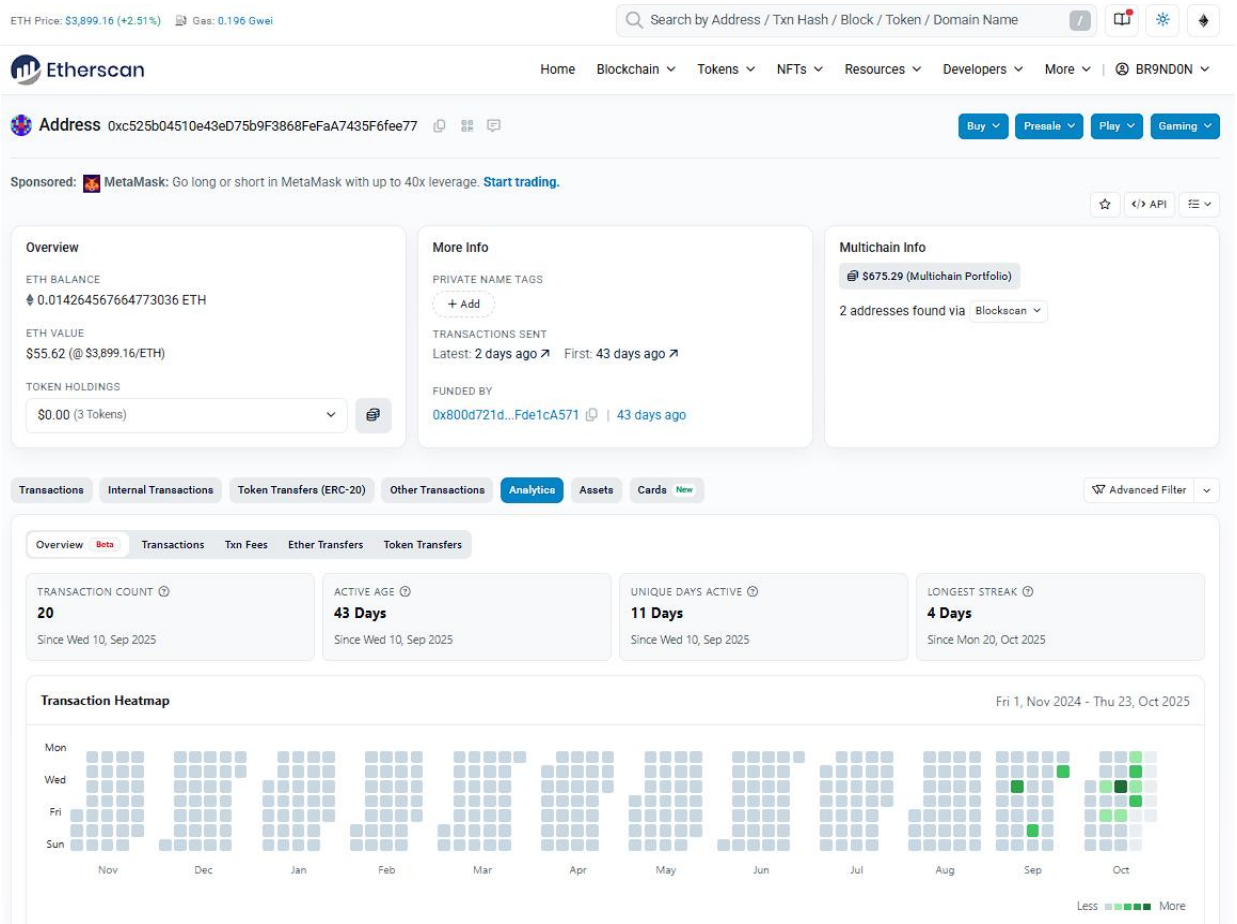

Рисунок 1.6 – Відстеження адреси через Etherscan

Інструменти для аналізу смарт-контрактів та вразливостей. Ця категорія інструментів спеціалізується на автоматичному та ручному виявленні помилок, вразливостей та потенційних векторів атак безпосередньо у коді смарт-контрактів, а також на моніторингу їхньої ончейн-поведінки після розгортання. Вони є критично важливими для розробників та аудиторів безпеки, допомагаючи запобігти експлойтам, що призводять до значних фінансових втрат.

До основних представників належать інструменти статичного аналізу, такі як Slither, MythX, CertiK, ConsenSys Diligence. Ці інструменти аналізують код, написаний популярними мовами (переважно Solidity, Vyper, Rust), на наявність вразливостей, проблеми з контролем доступу, логічні помилки тощо. Вони генерують звіти з описом ризиків, прикладами експлойтів та рекомендаціями щодо виправлення, часто інтегруючись у CI/CD процеси розробки. Підтримка охоплює переважно EVM-сумісні блокчейни, хоча частково поширюється і на non-EVM ланцюги.

Незважаючи на значну користь, ці інструменти мають суттєві обмеження. По-перше, існує обмежена підтримка нових мов смарт-контрактів (Move, Cairo, Plutus) та non-EVM блокчейн, оскільки основний фокус розробки зосереджений на Solidity та EVM-екосистемі. По-друге, використання цих інструментів та інтерпретація їхніх результатів вимагає високого рівня навичок технічної експертизи у сфері безпеки смарт-контрактів, що ускладнює їх застосування новачками, особливо враховуючи наявність хибних спрацьовувань (false positives) у звітах. По-третє, вартість просунутих функцій, повних аудитів від відомих компаній може бути дуже високою, роблячи їх недоступними для користувачів.

Користувацькі «чорні списки» та репортинг-сервіси – це інструменти, які дозволяють спільноті документувати та поширювати інформацію про шахрайські дії в блокчейні на які натрапили. Такі платформи, як Chainabuse, Cryptoscamdb, GitHub-репозиторії (ethereum-lists, scam-database), Reddit (r/CryptoScams) є цінними для швидкого інформування, але їхня ефективність обмежена суб’єктивністю даних і відсутністю глибокої аналітики.

Вони спеціалізуються на зборі, агрегації та наданні доступу до повідомлень про шахрайські адреси, вебсайти, інциденти та схеми. Вони функціонують як краудсорсингові бази даних, де жертви чи свідки діляться доказами (TXID, скріншоти), створюючи попередження для спільноти. На відміну від інструментів описаних раніше, вони не проводять ончейн-аналіз чи ризик-скоринг, а покладаються на активність користувачів і модерацію, що робить їх доступними, але менш точними.

На рисунку 1.7 подано скріншот з платформи Chainabuse на якому подано список звітів про шахрайство на блокчейні Ethereum.

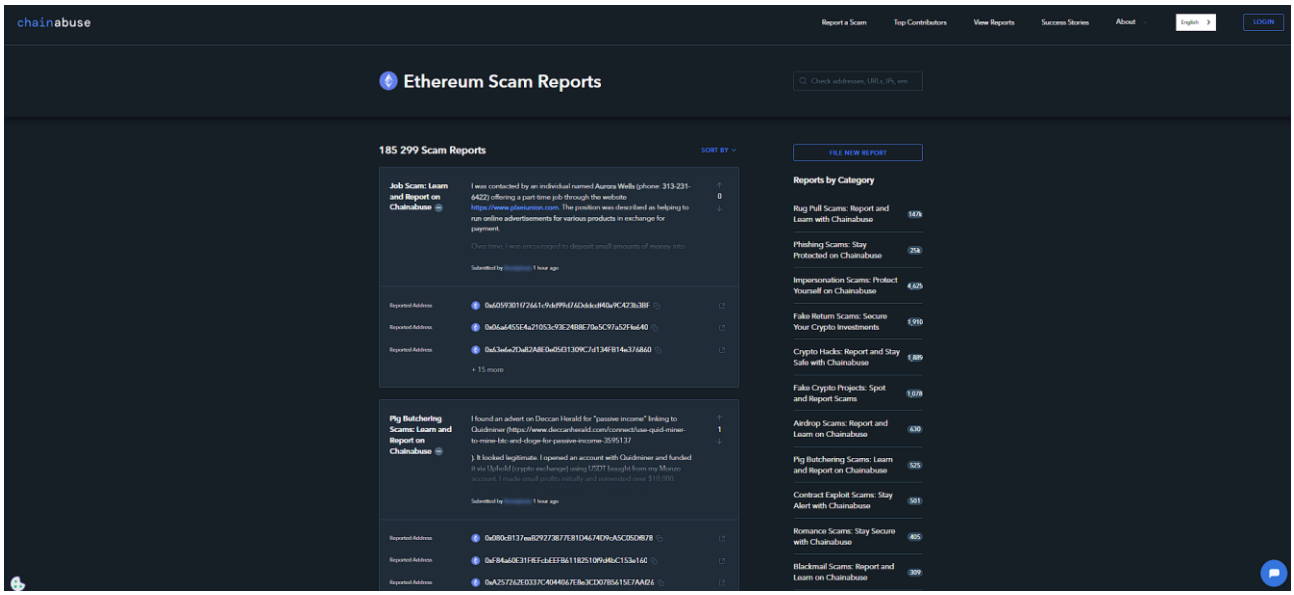

Рисунок 1.7 – Список звітів про шахрайство на блокчейні Ethereum зібраних на платформі Chainabuse

Окремим напрямком розвитку є застосування блокчейну в критичних системах – від розумних енергетичних мереж до платформ для реагування на надзвичайні події, що підвищує вимоги до прозорості, достовірності та безпеки. Дослідження [23, 24] показують, що традиційні методи кіберзахисту та збору цифрових доказів часто неефективні у децентралізованих середовищах, що в свою чергу, ускладнює виявлення та аналіз інцидентів. Це підтверджує, що навіть поза криптовалютами потрібні спеціальні інструменти аналітики та форензики для роботи з такими системами.

Таким чином, аналіз ринку інструментів для ончейн-аналітики показує наявність різноманітних рішень, кожне з яких має свій функціонал та цільову аудиторію. Існують платформи для управління активами, які добре візуалізують портфель, але не аналізують ризики. Є професійні аналітичні системи, що пропонують глибокий аналіз, однак їх складність та вартість роблять їх недоступними для широкого кола користувачів. Корпоративні рішення ефективні, але їх алгоритми закриті, а фокус вузький.

Ці обмеження підкреслюють потребу в розробці універсального інструменту, який би поєднував простоту управління активами, базовий ончейн-аналіз, доступність для пересічних користувачів і безкоштовний доступ, забезпечуючи комплексний захист і оцінку ризиків у Web3-екосистемі.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ІНСТРУМЕНТУ

2.1 Загальний опис проекту та архітектура системи

Головною метою даного проєкту – створити інструмент, який дасть змогу користувачам комплексно аналізувати та оцінювати ризики, пов'язані з будь-якою адресою в EVM-сумісних блокчейнах. Інструмент має збирати ончейн-дані, візуалізувати ключові патерни активності та зв'язків, а також надавати зрозумілу оцінку ймовірності причетності адреси до різних типів шахрайської діяльності. Очікуваним результатом вважається веб-сайт у вигляді інтерактивного дашборду.

Після введення користувачем EVM-адреси, інструмент повинен:

– Збирати та відображати загальну інформацію про адресу.

– Візуалізувати склад портфеля активів.

– Надавати детальну, агреговану історію транзакцій з різних блокчейнів.

– Візуалізувати топологію зв'язків адреси з іншими контрагентами у вигляді інтерактивного графа.

– Відображати динаміку активності адреси на часовій шкалі.

– Розраховувати оцінки ризику для поширених типів шахрайства.

– Генерувати висновок з поясненням основних ризиків та фінальним рейтингом безпеки адреси.

Інструмент призначений для широкого кола користувачів екосистеми EVM, включаючи інвесторів, трейдерів та користувачів DeFi, які хочуть перевірити надійність адрес перед здійсненням транзакції. Фокус робиться не тільки на розслідуванні вже скоєних злочинів, а й на проактивній оцінці ризиків.

Вимоги до проекту:

– Функціональні. Реалізація всіх описаних вище функцій збору, аналізу та візуалізації даних.

– Продуктивність. Забезпечення прийнятного часу відгуку системи.

– Точність. Прагнення до максимально можливої точності в оцінках (з урахуванням складності задачі та доступності даних).

– Зрозумілість. Інтерфейс та результати аналізу мають бути інтуїтивно зрозумілими для користувачів з різним рівнем технічної підготовки.

– Модульність. Архітектура має бути модульною для полегшення подальшого розвитку, додавання нових функцій, типів шахрайства чи підтримки інших блокчейнів.

Архітектура системи. Проект побудовано на основі клієнт-серверної архітектури. Серверна частина відповідає за всю логіку збору даних (через API зовнішніх провайдерів ончейн-інформації), їх обробку, агрегацію та надання результатів через REST API. Клієнтська частина (фронтенд) є веб-інтерфейсом (дашбордом), який отримує дані від бекенду та відповідає за їх візуалізацію та інтерактивну взаємодію з користувачем.

На рисунку 2.1 подано вигляд архітектури проекту.

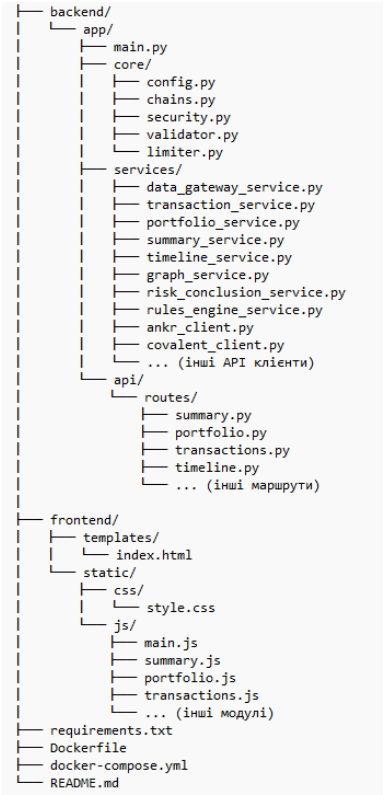

Рисунок 2.1 – Архітектура проекту

Запропонована архітектура забезпечує чітке розмежування логіки між бекендом та фронтендом, що робить систему гнучкою, масштабованою та простою в обслуговуванні.

2.1.1 Модуль збору загальної інформації про адресу

Першочерговим завданням даного модуля є агрегація та представлення ключових загальних атрибутів аналізованої EVM-адреси. Ця інформація формує базовий профіль об'єкта дослідження і відображається у вихідному інформаційному блоці інтерфейсу користувача. Метою є надання зведеного огляду основних характеристик та масштабів ончейн-активності, що передує більш глибокому аналізу.

Даний модуль повинен збирати та презентувати наступні ключові показники: поточний баланс (USD), тип адреси (EOA/смарт-контракт), кількість ідентифікованих активів, кількість активних блокчейнів, оцінка загальної кількості транзакцій (включаючи кількість вихідних/вхідних транзакцій), дата першої та останньої транзакції, а також вік адреси. Цей процес реалізується на рівні SummaryService, який використовує гібридну стратегію збору даних, що поєднує можливості кількох провайдерів. Спочатку сервіс визначає всі активні блокчейни, використовуючи запити до історії транзакцій (Etherscan API V2 / Ankr Advanced API) для ідентифікації ланцюгів, які мають активність, навіть якщо їхній баланс дорівнює нулю.

Основне джерело для отримання балансів – це GoldRush API, який робить запит по всіх знайдених блокчейнах, кешує та повертає items. У разі, якщо він віддає помилку або порожні дані, система автоматично переходить на Ankr MultiChain API як резервний механізм (fallback). Дані, отримані від Ankr, зливаються з даними GoldRush, що забезпечує максимальну повноту інформації. Після збору, ці дані використовуються для розрахунку Total Balance та Identified Assets по всіх активних блокчейнах.

Окремо збираються транзакційні характеристики. Дати першої та останньої транзакції в історії гаманця отримуються за допомогою цільових запитів до Etherscan V2 (з відповідним сортуванням asc або desc) з fallback на Ankr. Загальна кількість транзакцій (total_transactions_real) вираховується з поля total_records (Etherscan V2) або totalCount (Ankr), що дає суму вхідних та вихідних операцій, а не лише Nonce.

У результаті, summary_service віддає структуровану відповідь, яка чітко розділяє глобальні та детальні дані. Блок total містить агреговані суми (баланс, активи, загальна кількість транзакцій) з усіх виявлених активних блокчейнів. Блок per_chain містить детальний зріз для кожного блокчейну (баланс, активи, лічильники транзакцій), що використовується Frontend для динамічного фільтрування.

На рисунку 2.2 представлено вигляд блоку загальної інформації, що відображає ключові показники аналізованої адреси.

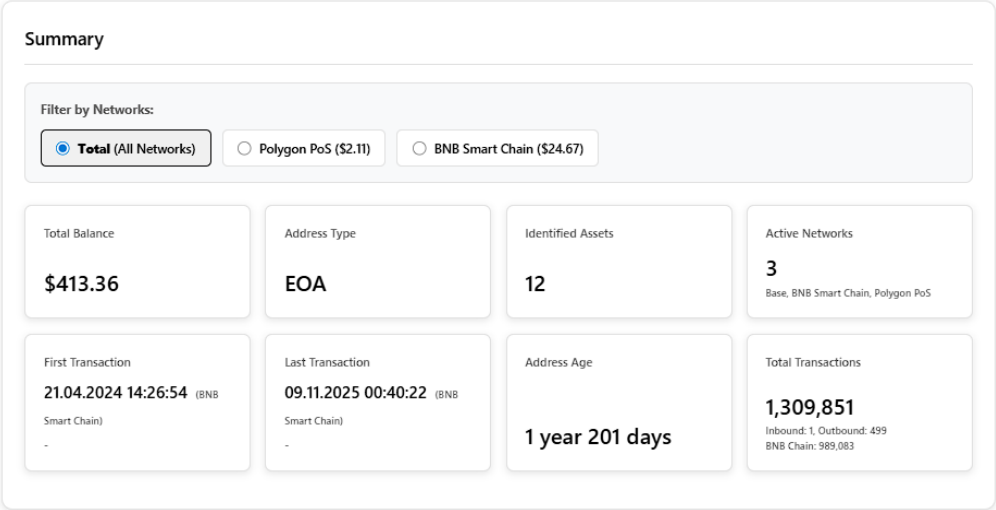

Рисунок 2.2 – Вигляд блоку «Summary»

Слід розуміти, що значна частина показників, представлених у цьому модулі, є апроксимаціями, зумовленими особливостями роботи зовнішніх API та самої природи ончейн-даних. Зокрема, дані портфеля (Баланс USD, Кількість активів) є оціночними, адже різні API-провайдери можуть показувати розбіжності через:

– Різне покриття токенів. Не всі сервіси ідентифікують або відстежують ціни для абсолютно всіх існуючих токенів, особливо нових, маловідомих чи з низькою ліквідністю.

– Різні джерела цін. Використання різних бірж (CEX/DEX) чи агрегаторів для визначення поточної вартості може призводити до невеликих відмінностей.

– Різні методики індексації та оновлення. Час оновлення даних про баланси та ціни може відрізнятися.

– Фільтрація спаму/пилу. Сервіси можуть по-різному обробляти або ігнорувати токени з мізерною вартістю («пил») чи відомі спам-токени.

Користувачеві слід сприймати представлені в цьому блоці дані як інформативний знімок поточного стану та нещодавньої активності адреси, розуміючи властиві обмеження будь-якого інструменту агрегації ончейн-даних.

2.1.2 Модуль відображення активів адреси

Цей модуль призначений для візуального представлення складу та розподілу криптоактивів, що належать аналізованій EVM-адресі. Його мета – надати користувачеві наочне розуміння того, в яких блокчейнах та конкретних токенах зосереджена основна вартість портфеля, доповнюючи загальний показник балансу з першого блоку.

Як видно на рисунку 2.3, модуль складається з трьох основних компонентів для всебічного огляду портфеля:

Ієрархічна діаграма «Розподіл активів» (Asset Distribution). Відображає структуру портфеля, де внутрішні кільця представляють блокчейни (напр., ETH, BASE, BSC), а зовнішні – окремі токени в межах цих блокчейнів. Розмір кожного сектора пропорційний його вартості в USD відносно загального балансу. На діаграмі відображаються назви токенів і блокчейнів для швидкої орієнтації, та детальна інформація (вартість, відсоткова частка) з'являється у спливаючій підказці при наведенні курсору. Ця візуалізація дозволяє швидко оцінити концентрацію капіталу в певних блокчейнах та домінуючі активи. Для реалізації використовується бібліотека Plotly.js, що забезпечує інтерактивність та можливість завантаження діаграми у форматі PNG.

Детальний «Список токенів». Представляє повний перелік усіх ідентифікованих активів, згрупованих за блокчейнами. Для кожного активу вказується кількість, вартість в USD та відсоткова частка від загального портфеля. Передбачена інтерактивна опція «Приховати токени <$0.01» (чекбокс у заголовку списку) для динамічної фільтрації «пилу» та спам-токенів.

Цей список надає точну та вичерпну інформацію про кожен актив, доповнюючи загальну картину діаграми.

Стовпчаста діаграма «Топ активів по вартості». Відображає активи у портфелі у вигляді стовпчастої діаграми, відсортованої за спаданням вартості в USD. Ця візуалізація полегшує порівняння вартості основних активів, що може бути складно зробити на секторній діаграмі, особливо при наявності одного домінуючого активу. Діаграма включає інтуїтивні елементи керування масштабуванням (zoom in/out) через панель інструментів Plotly. Для побудови використовується Plotly.js.

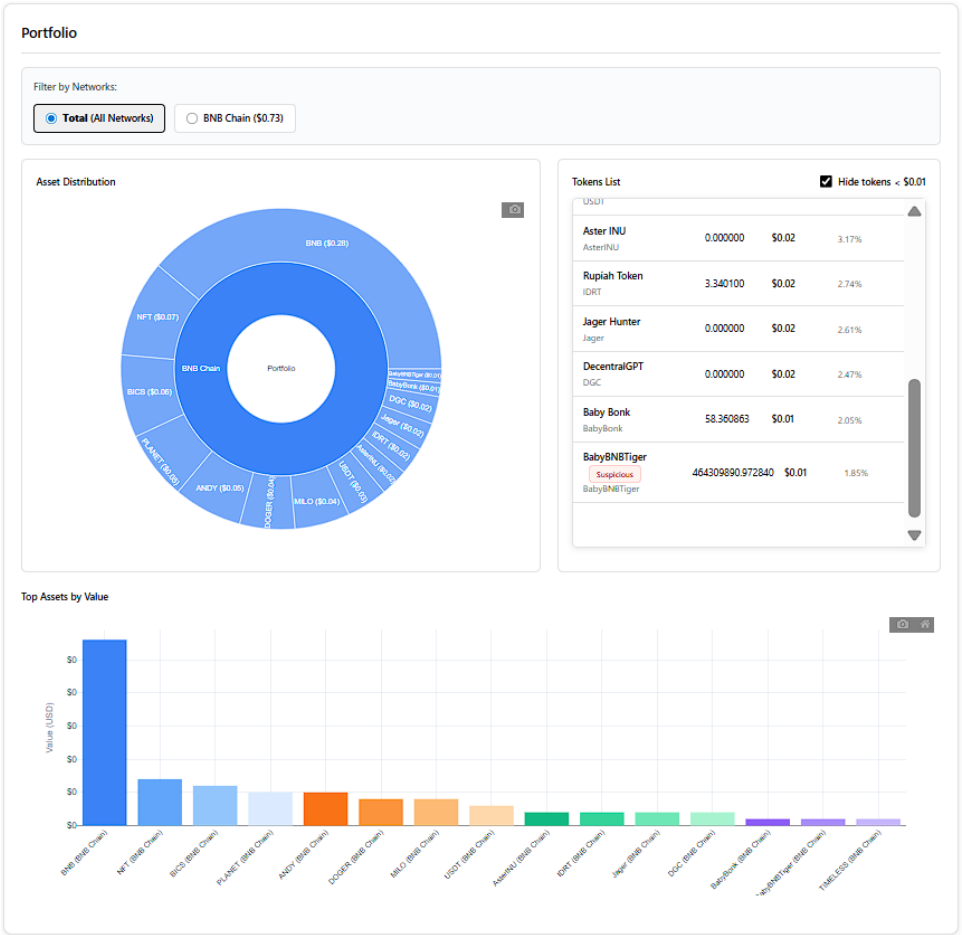

Рисунок 2.3 – Вигляд блоку «Portfolio»

Дані для цього модуля отримуються першочергово з GoldRush API, оскільки цей сервіс забезпечує найбільш повне покриття EVM-блокчейнів та точну оцінку токенів. У разі недоступності Covalent система використовує Ankr Multichain API у комбінації з прямими RPC-запитами як резервний варіант.

Візуалізація здійснюється на фронтенді (JavaScript) за допомогою бібліотеки Plotly.js з налаштуваннями для оптимальної читабельності та інтерактивності. Фільтрація токенів <$0.01 реалізована динамічно на фронтенді з перерендерингом усіх трьох компонентів при зміні стану чекбокса.

Щоб аналізувати інформацію користувач може використовувати Sunburst діаграму для швидкої оцінки основних блокчейнів та активів. Стовпчаста діаграма допомагає порівняти вартість ключових позицій. Детальний список слугує для вичерпного вивчення всіх активів, включаючи дрібні, та для отримання точних числових даних. Комбінація цих трьох компонентів надає повне та багатогранне уявлення про склад портфеля аналізованої адреси.

2.1.3 Модуль відображення історії транзакцій

Цей модуль надає користувачеві єдиний, агрегований перелік останніх транзакцій, пов'язаних з аналізованою EVM-адресою, зібраних з усіх активних блокчейнів в одному місці. Мета – дозволити дослідити повну хронологію активності, типи взаємодій, контрагентів та обсяги переказів, що є важливим для розуміння поведінки адреси та виявлення підозрілих патернів, без необхідності вручну перемикатися між різними блокчейн-експлорерами для кожного блокчейну, що значно спрощує аналіз мультичейн-активності.

Інтерфейс модуля «Transaction History» зображений на рисунку 2.4 містить наступні елементи:

Загальна статистика. Відображається загальна кількість завантажених транзакцій та кількість активних блокчейнів, з яких ці дані було зібрано.

Фільтри блокчейнів (Network Filters). Інтерактивні кнопки (чекбокси), що дозволяють користувачеві фільтрувати відображувані транзакції за конкретними блокчейнами.

Розширені фільтри та пошук (Advanced Filters & Search) – панель з додатковими інструментами для глибшого аналізу завантаженої вибірки:

- Пошук за хешем транзакції (Transaction Hash).
- Фільтр за адресою контрагента (From/To Address).

– Фільтр за методом/типом взаємодії (Method).

– Фільтр за статусом транзакції (Status) (Успішна/Невдала).

– Фільтр за мінімальною сумою в USD.

– Кнопка «Reset» для застосування та скидання фільтрів.

Таблиця транзакцій. Основний елемент, що відображає відфільтровані транзакції у вигляді таблиці зі стовпцями:

– Status – індикатор результату виконання транзакції в блокчейні.

– Blockchain – назва блокчейну в якій було здійснено транзакцію.

– Transaction Hash – унікальний ідентифікатор транзакції, що веде до повних деталей у блок-експлорері, клікабельний.

– Method – вказує на тип операції, здійсненої транзакцією.

– Block – номер блоку, в який було включено транзакцію.

– Age – час, що минув з моменту підтвердження транзакцій.

– From – адреса, яка ініціювала транзакцію, клікабельне посилання.

– To – адреса призначення чи контракт з яким взаємодіяли, клікабельне.

– Amount – кількість та тип активу, що передавався.

– Txn Fee – вартість, сплачена блокчейну за обробку транзакції.

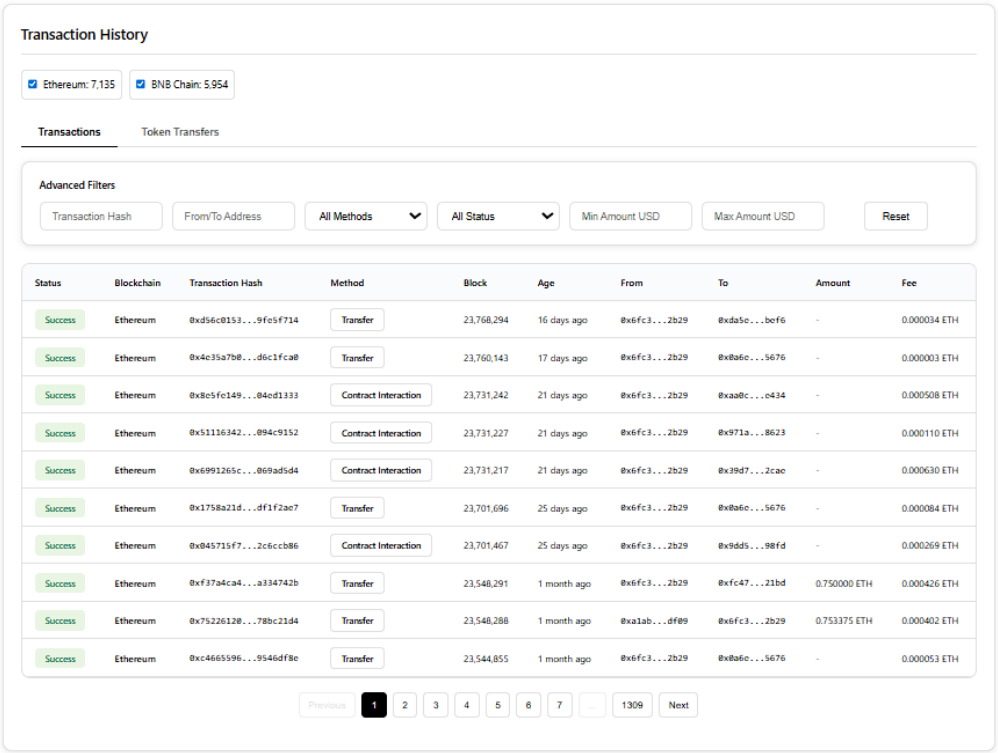

Рисунок 2.4 – Вигляд блоку «Transaction History»

Для отримання уніфікованої та хронологічно впорядкованої історії транзакцій з багатьох EVM-блокчейнів використовується Ankr Advanced API. Вибір Ankr API обґрунтований його ключовою перевагою – multichain агрегацією, яка значно спрощує отримання єдиного списку транзакцій з різних блокчейнів. Це дозволяє уникнути необхідності надсилання окремих запитів для кожної блокчейнів, що було б повільним та ресурсозатратним.

Незважаючи на тривалість завантаження, зумовлену послідовною природою пагінації (кожен запит чекає на відповідь попереднього), цей підхід визнано оптимальним для агрегації мультичейн-історії в рамках обмежень безкоштовних API-планів. Подальша обробка та відображення даних, реалізуються на стороні клієнта (фронтенді) за допомогою JavaScript. Це забезпечує швидку інтерактивність інтерфейсу після початкового завантаження raw-даних.

Користувач може розпочати з фільтрації за блокченом, щоб дослідити активність у конкретному блокчейні. Розширені фільтри дозволяють знаходити специфічні транзакції: наприклад, всі вхідні (Method: Receive), всі взаємодії з певною адресою, транзакції вище певної суми або невдалі операції (Status: Error). Клікабельні посилання на хеші та адреси дозволяють перейти до блок-експлорера для отримання вичерпної інформації.

2.1.4 Модуль візуалізації хронології активності

Даний модуль призначений для візуалізації активності EVM-адреси. Його мета – перетворити великий список історії транзакцій на зрозумілу графічну форму, що дозволяє користувачеві швидко виявляти аномалії, піки активності та бездіяльності. Як видно на рисунку 2.5, модуль складається з двох основних компонентів:

Інформаційні блоки. Надають статистичні дані для огляду активності адреси: Total Period (загальний період активності), Total Volume (сумарний обсяг транзакцій в USD), Avg Volume (середній обсяг) та Peak Day (дата найбільшої активності).

Лінійний графік активності (Timeline Chart). Відображає динаміку активності за часом. Користувач може перемикати відображення між Transaction Count (кількість транзакцій) та Volume (USD) (об'єм коштів) для кожного блокчейну. Використання подвійної осі по вісі Y (ліва – Volume (USD), права вісь – Transaction Count) дозволяє одночасно аналізувати кореляцію між частотою та фінансовою вагою транзакцій.

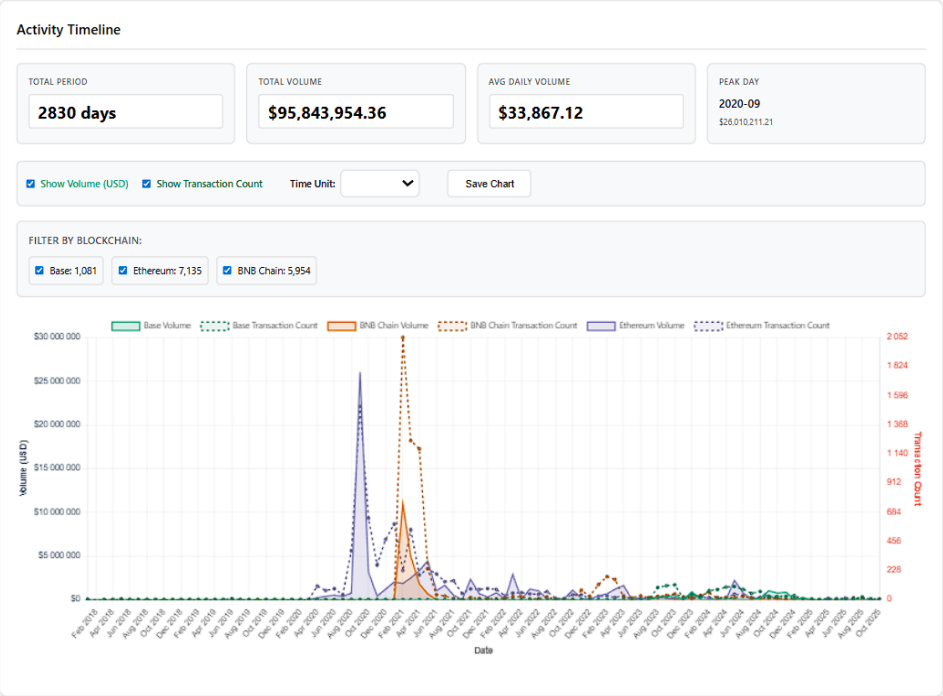

Рисунок 2.5 – Вигляд блоку «Activity Timeline»

Модуль дозволяє користувачеві обирати одиницю часу (Second, Minute, Hour, Day) та фільтрувати відображення конкретних блокчейнів через чекбокси, що дозволяє ізолювати активність.

Для візуалізації активності використовується історія транзакцій, яка попередньо була завантажена в блок Transaction History, після чого timeline_service виконує їх обробку: сортує за timestamp, групує та підсумовує активності в обрані часові проміжки. Візуалізація здійснюється на фронтенді за допомогою бібліотеки Plotly.js, що забезпечує інтерактивність та гнучкість масштабування.

Даний блок є дозволяє швидко виявити аномалії, що можуть вказувати на шахрайську активність. Для орієнтації, слід фокусуватися на різких сплесках на

графіку та співвідношенні двох вертикальних осей (Volume та Count). Аналітична цінність модуля зосереджена на дисбалансі: якщо видно великий стрибок обсягу (Volume) при майже нульовій кількості транзакцій (Count), це свідчить про переміщення великого капіталу, що може бути пов'язано з виведенням коштів. І навпаки, висока кількість транзакцій при низькому обсязі вказує на бот-активність або Wash Trading (створення штучної активності). Додатково, можна використати фільтри по блокчейнах, щоб ізолювати активність і визначити, який блокчейн є джерелом ризику.

2.1.5 Модуль візуалізації взаємодій адреси

Даний модуль призначений для візуального представлення блокчейн взаємодій аналізованої EVM-адреси з іншими адресами (контрагентами) у вигляді інтерактивного графа. Мета – допомогти користувачеві візуально ідентифікувати патерни зв'язків, основних контрагентів, потенційні кластери активності та напрямки потоків коштів, що може вказувати на специфічний характер діяльності адреси, включаючи можливі шахрайські схеми.

Реалізація графу для даного блоку є двокомпонентною:

Backend (graph_service). Використовує бібліотеку NetworkX для аналізу транзакцій, обчислення метрик вузлів (наприклад, ступінь центральності) та формування структури «вузли-ребра» (Nodes-Edges).

Frontend (graph.js). Використовує JavaScript-бібліотеку Cytoscape.js для рендерингу графу, забезпечуючи плавну інтерактивність.

Граф побудований таким чином, щоб візуально підкреслювати значущість зв'язків:

– Вузли (Nodes) представляють унікальні EVM-адреси. Розмір вузла масштабується відповідно до кількості його транзакцій (ступеня), що візуально виділяє хаби (центральні адреси) блокчейн.

– Ребра (Edges) представляють транзакційні зв'язки. Товщина ребра масштабується логарифмічно відповідно до сумарної вартості (USD)

транзакцій. Це підкреслює фінансово значущі потоки капіталу і дозволяє ігнорувати «шум» (дрібні dusting-транзакції).

– Мультичейн-накладання. Модуль підтримує відображення транзакцій з усіх обраних блокченів одночасно, дозволяючи візуально виявляти перекази між різними Layer 1/Layer 2 ланцюгами.

– Розкладки (Layouts). Користувач може обирати різні алгоритми: COSE для виявлення щільних кластерів та спільнот, Hierarchical для простеження послідовних потоків капіталу, Circle або Grid для базового огляду.

– Контроль даних. Впроваджено функціонал фільтрації за Min Value USD, що дозволяє користувачеві ігнорувати транзакції нижче встановленого порогу та зосередитися на великих, підозрілих сумах.

– Розширення вузла (Hub Detection). При натисканні на вузол система автоматично викликає get_transaction_count (на Backend) для перевірки, чи є адреса великим хабом. Якщо кількість транзакцій перевищує поріг, користувачеві пропонується ручне завантаження, щоб уникнути перевантаження графу.

На рисунку 2.6 подано вигляд блоку Transaction Network Graph.

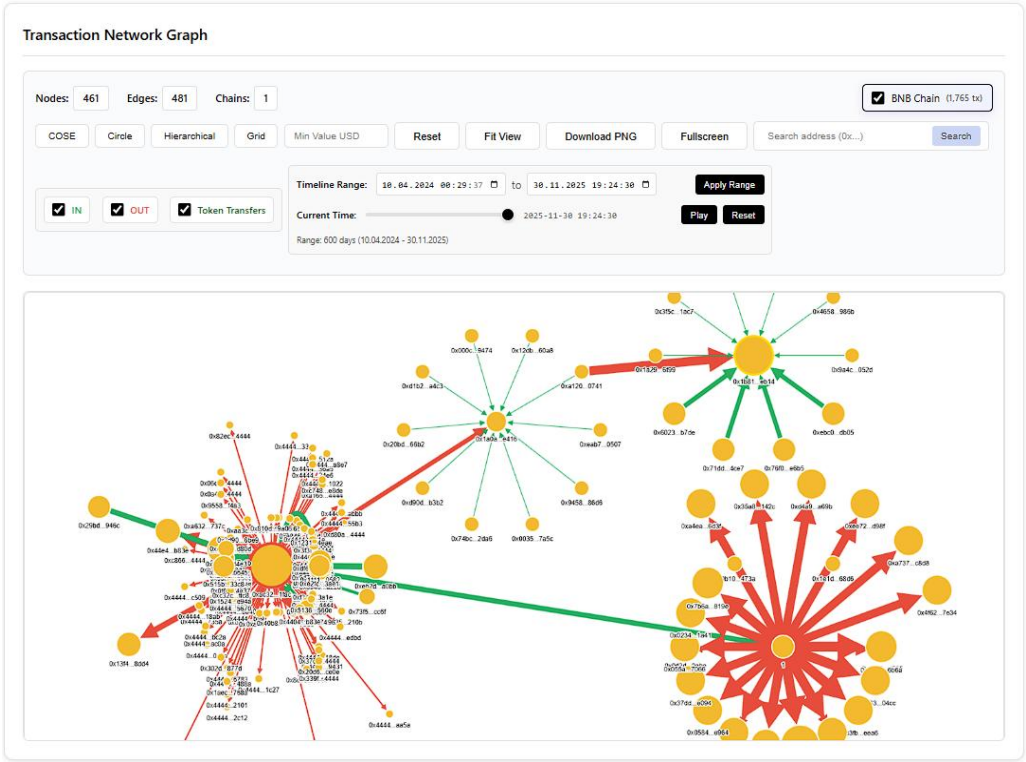

Рисунок 2.6 – Вигляд блоку «Transaction Network Graph»

Використання графу починається з вибору релевантної вибірки. Користувач може обмежити аналіз до певної кількості транзакцій або ланцюгів для забезпечення оптимальної продуктивності. Після завантаження даних, користувач може змінювати розкладки, щоб виявити хаби (знайти вузли, що взаємодіяли з цільовою адресою), простежити потік (використовувати ієрархічні розкладки, щоб візуально простежити ланцюжок транзакцій (хто поповнив гаманець, і куди кошти пішли далі), що є ключовим для виявлення відмивання грошей), ізолювати ризик (використовувати фільтри, щоб прибрати «шум» і зосередитися лише на транзакціях вище мінімальної суми).

2.1.6 Модуль оцінки ризику шахрайства

Цей модуль є центральним елементом дашборду, призначеним для проактивної оцінки потенційних ризиків, пов'язаних з досліджуваною EVM-адресою. Його основне завдання – перетворити складні ончейн-дані в зрозумілі індикатори, що дозволяють користувачеві швидко ухвалювати обґрунтовані рішення.

Оцінка ризику здійснюється виключно на основі ончейн-даних, зібраних через безкоштовні та відкриті API-провайдери та сервіси, що гарантує прозорість і відтворюваність результатів. Система працює як гнучкий набір правил (Rules Engine), які дозволяють не лише ідентифікувати ризик, а й надати користувачу точне обґрунтування того, чому адреса або контракт отримали певну оцінку, сформоване на основі аналізу реальних шахрайських схем, описи яких було знайдено у відкритих джерелах.

У випадках, коли автоматичний збір даних через API обмежений чи ускладнений через масштабність атак, модуль пропонує альтернативний інтерактивний режим – анкету для оцінки ризику. Це дозволяє користувачу безпосередньо брати участь у процесі аналізу: відповідаючи на кілька ключових питань, він оцінює взаємодію з адресою та виявляє потенційні загрози. Окрім цього, інтерактивна анкета легко адаптується до нових схем атак без необхідності переписувати код, що робить її ефективним інструментом для навчання та швидкого реагування на появу нових видів шахрайства в блокчейні.

Такий формат дозволяє поєднати аналіз з навчальним ефектом [25], де користувач набуває практичних навичок розпізнавання шахрайства, що допоможе йому в подальшому самостійно виявляти підозрілі схеми, значно знижуючи ймовірність стати жертвою атак чи інших форм соціальної інженерії.

Після заповнення анкети система миттєво формує Risk Score, визначає роль учасника (жертва, шахрай, скам-токен або чиста адреса) і надає детальні рекомендації щодо безпечних дій. Всі питання та їхні ваги базуються на тих же евристиках Rules Engine, що використовуються для автоматичного аналізу.

Наприклад, спробуємо проаналізувати загрозу «Token Approvals» (див. рисунок 2.7), що відповідає за аналіз дозволів, надані користувачем смарт-контрактам, при компрометації яких, можливе списання усіх активів.

Для збору даних підблок використовує каскадний підхід через три API: основний – Covalent, fallback №1 – Moralis, fallback №2 – Revoke.cash. Таке поєднання гарантує максимальне покриття блокчейнів і дозволяє отримати повні й актуальні дані про всі активні approvals.

Процес аналізу складається з кількох кроків:

– Отримання approvals. Кожен блокчейн опитуються паралельно, після чого дані нормалізуються у стандартний формат, що включає інформацію про токен, spender, ліміт дозволу та тип токена (ERC-20, ERC-721, ERC-1155).

– Визначення unlimited approvals. Якщо allowance дорівнює MAX_UINT256 або еквівалентному значенню, він класифікується як «Unlimited». Такі дозволи несуть критичний ризик, оскільки будь-який сторонній контракт може зняти весь баланс токена.

– Обчислення risk_score. Оцінка ризику формується на основі типу дозволів токенів. Якщо хоча б один апрув є безлімітним (Unlimited), ризик встановлюється на 100%, оскільки це дає спендеру повний контроль над токенами. Для лімітних дозволів оцінка ризику встановлюється як 80%, оскільки власник залишається обмеженим певною сумою. Такі оцінки ризику формуються виходячи з обережності: навіть перевірені та, на перший погляд, надійні смарт-контракти у бути скомпрометовані, тому approvals на будь-які контракти завжди розглядаються як ризик для власника адреси.

– Формування звіту. Підблок відображає показники, які пояснюють, чому певна адреса отримала відповідний рейтинг.

– Отримання рекомендацій. Користувач отримує чіткі поради, залежно від ризику – від регулярного моніторингу дозволів до негайного відкликання через revoke.cash.

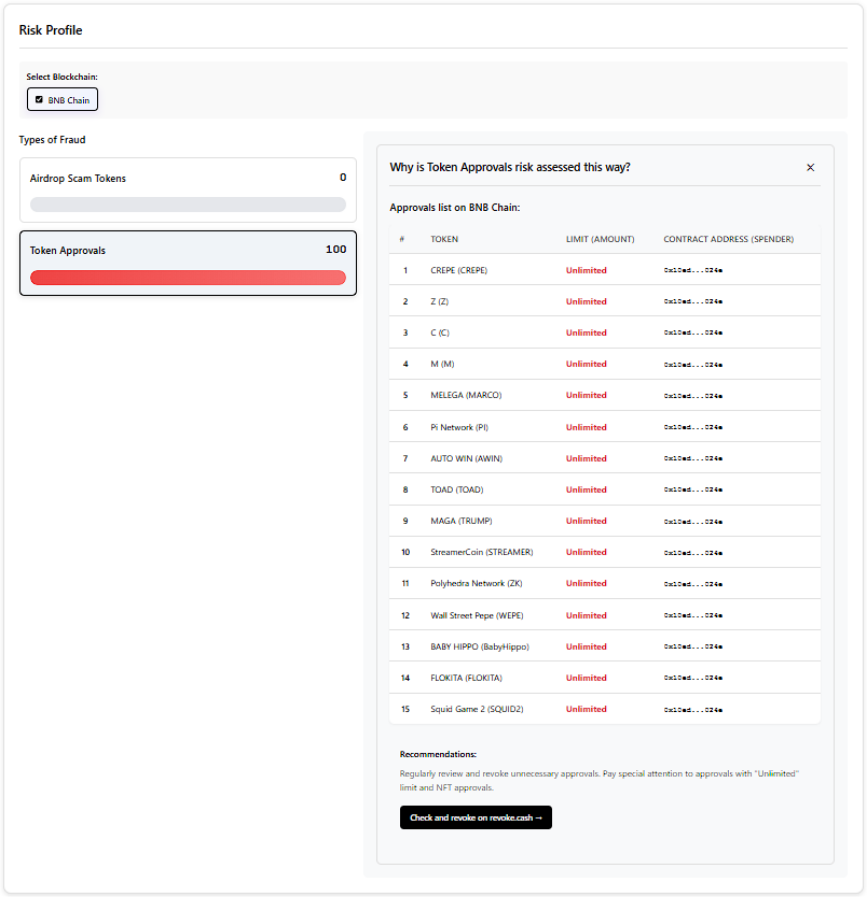

Рисунок 2.7 – Аналіз загрози «Token Approvals»

Підсумовуючи, завдяки інтеграції різноманітних ончейн-ознак та їх зваженому аналізу, модуль «Risk Profile» надає універсальне рішення для ідентифікації відомих та потенційних шахрайських патернів. Це дозволяє користувачам своєчасно реагувати на небезпеки, мінімізуючи ймовірність фінансових втрат. Таким чином, модуль сприяє створенню більш безпечного та інформованого середовища для взаємодії з децентралізованими протоколами та активами.

2.1.7 Модуль висновку щодо ризику взаємодії

Модуль «Conclusion» є завершальним елементом системи аналізу та відповідає за формування остаточного вердикту щодо безпеки взаємодії з досліджуваною EVM-адресою. На відміну від блоку Risk Profile, який надає детальний розбір по кожному типу шахрайства, даний модуль агрегує всі отримані дані й подає користувачу єдине узагальнене рішення у вигляді висновку, який містить текстовий висновок та фінальну оцінку ризику від 0 до 100%, де 0% – низький ризик, а 100% – критичний (див. рисунок 2.8).

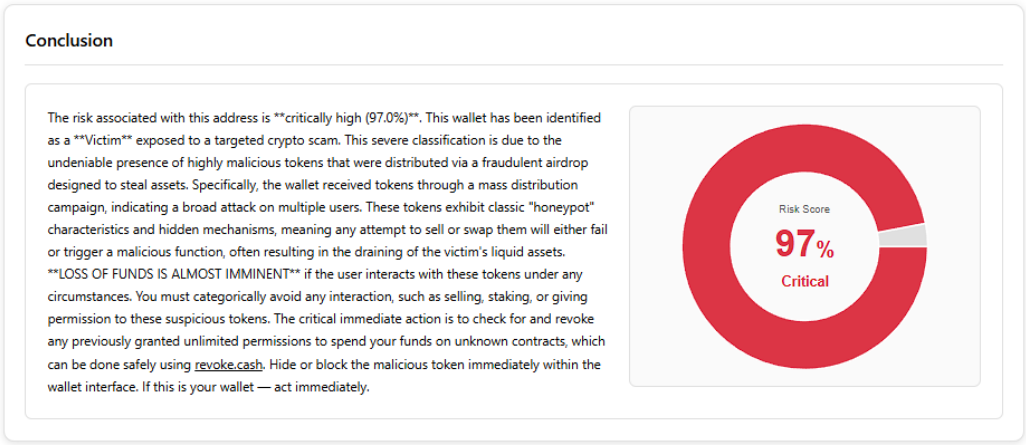

Рисунок 2.8 – Вигляд блоку «Conclusion»

Формування фінальної оцінки ризику відбувається за принципом домінуючої загрози. Загальний бал не є середнім арифметичним – він визначається найбільшим значенням серед усіх модулів, тобто модуль із найвищим показником формує кінцевий рівень небезпеки. Якщо серед підмодулів є критичний індикатор (наприклад, 100% Token Approvals через безлімітний дозвіл), фінальний бал одразу дорівнює цій величині. Якщо ж критичного значення немає, оцінка встановлюється на рівні найвищого виявленого ризику – наприклад, при показниках Airdrop Scam Tokens 0%, Address Poisoning 0%, Token Approvals 80% кінцева оцінка становитиме 80%. Такий підхід виправданий тим, що навіть одна критична вразливість здатна повністю нівелювати безпечність адреси та створює реальну можливість миттєвої компрометації активів.

Після того як система обчислює загальний Risk Score, запускається механізм генерації висновку. У його основі працює мовна модель, інтегрована через Google AI Studio API. Використання цього сервісу дає змогу формувати описи, які не лише перераховують ключові чинники небезпеки, а й логічно пояснюють, як ці чинники взаємодіють між собою. У запиті передаються метрики з Risk Profile і підсумкова оцінка ризику, після чого мовна модель повертає сформований опис.

Такий підхід дає змогу отримувати висновки, які не обмежуються сухим переліком технічних параметрів. Мовна модель формує узгоджений текст, що враховує тональність відповідно до рівня загрози: нейтральну при низькому ризику, насторожувальну при середньому та попереджувальну при високому. Це робить результат придатним для сприйняття користувачем, який не має технічних знань. Текстовий висновок містить систематизоване пояснення: чому адреса отримала певну оцінку, які аспекти вплинули на неї найбільше, та яким має бути рекомендований рівень обережності.

Таким чином, модуль висновку завершує роботу всієї системи аналізу, підсумовуючи результати роботи всіх блоків, що дозволяє користувачу швидко оцінити стан адреси та прийняти обґрунтоване рішення щодо подальшої взаємодії.

2.2 Тестування інструменту на реальних випадках

Для оцінки ефективності розробленого інструменту було проведено тестування на реальних адресах. Такий підхід дозволить не лише оцінити роботу кожного підблоку інструменту, але й провести повноцінне ончейн-розслідування, перевіряючи історію транзакцій, взаємодії з адресами та можливі прояви шахрайської активності. Тестування проводилось через реалізований інструмент та паралельно верифікувалось на інших відкритих джерелах (сканерів блокчейнів, dex, revoke.cash та ін.), щоб оцінити точність, повноту та надійність результатів.

Перевірка буде проходити на прикладі шахрайства Airdrop Scam Tokens, що обґрунтовано його поширеністю. Практично кожен користувач, який взаємодіє зі смарт-контрактами, хоча б раз отримував безкоштовні токени, які могли мати значну вартість. Однак такі токени часто виявляються шахрайськими, і будь-яка спроба їх продати або обміняти може призвести до фінансових втрат. Таким чином, тестування цього виду шахрайства дозволяє оцінити ефективність інструменту саме на найбільш розповсюджених сценаріях взаємодії з криптоактивами.

Також для максимально об’єктивної оцінки ефективності було проведено тестування, яке моделює реальний шлях жертви від моменту отримання скам-токена до повної втрати коштів.

Етап 1 – створення «Airdrop» токена. Шахрайство починається ще до взаємодії з жертвою – на етапі створення шкідливого токена. Шахрай розгортає смартконтракт, у якому закладає потрібні пастки: блокування продажу для звичайних користувачів (honeypot), встановлення комісій на купівлю/продаж (buy/sell tax), приховані функції виклику сторонніх контрактів (External Calls) та ін. Зазвичай такі токени створюють з схожою чи навіть ідентичною назвою, логотипом до реальних токенів, які вже торгуються на різних платформах. Такий підхід дозволяє скам-токенам створювати оманливе враження безпечності під час поверхневої перевірки, за відсутності глибшої перевірки користувач має більше шансів потрапити у пастку цього виду шахрайства.

Проаналізуємо токени які отримав користувач з етапу 2. До прикладу, візьмемо токен Tradoor (0x5707c78C86eB3E1C121Fd79e6 b38a5C2Ca835E72), який за назвою ідентичний до реально токена Tradoor адреса якого – 0x9123400446a56176Eb1B6BE9ee5CF703e409F492. Якщо навіть поверхнево перевірити у сканері блокчейну на якому створений цей токен (BNB Smart Chain) можна побачити суттєву розбіжність – в реального токена верифікований код, дата його створення більше становить більше 3х місяців, є посилання на офіційний сайт та ін. (див. рисунок 2.9).

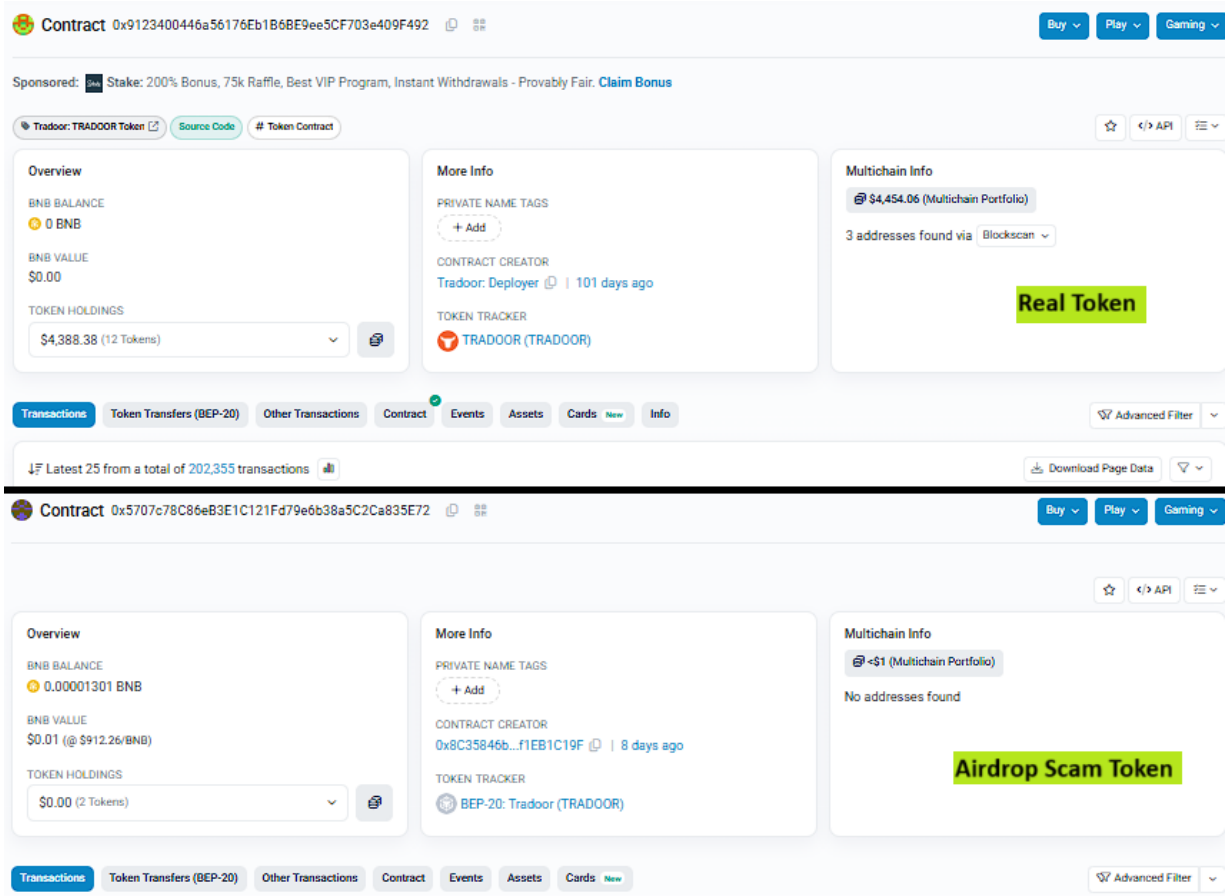

Рисунок 2.9 – Відмінність між справжнім та скам-токеном Tradoor

Якщо порівняти адреси токенів із даними на торгових аналітичних платформах наприклад, Dex Screener, можна чітко простежити різницю між поведінкою справжнього активу та скам-токена. У реальних токенів графік цін зазвичай формується природно: спостерігаються органічні коливання, чергуються періоди зростання та спадів, присутня об’єктивна ліквідність, а історія торгів охоплює значні часові проміжки. Натомість графік шахрайського токена має неприродний та «аномальний» характер. Часто він виглядає майже прямолінійним або «запилено» зростаючим – шахраї штучно накачують ціну через контрольовані маніпуляції обсягами торгів, щоб токен та став більш помітним для користувачів (зазвичай скам токени, через свою низьку вартість автоматично приховуються через функцію більшості гаманців та бірж «приховати активи вартістю < 1$»), проте якщо штучно «накачати ціну» то актив вийде з цієї категорії та буде створювати ілюзію потенційного прибутку та мотивує користувача спробувати продати небажаний токен. Однак реалізувати цей продаж практично неможливо: такі токени часто виявляються

honeypot-контрактами, мають приховані комісії (інколи 90–100%), забороняють продаж або провокують списання коштів під час будь-якої взаємодії.

На рисунку 2.10 наведено порівняння графіків реального токена (верхній) та скам-токена (нижній), де чітко видно контрасти: реальний актив демонструє збалансовану ліквідність, реальний обсяг операцій та сталий торговий інтервал, тоді як скам-токен має різко завищену ціну, відсутність реальної ліквідності, мінімальну або контрольовану кількість транзакцій та надзвичайно коротку історію існування.

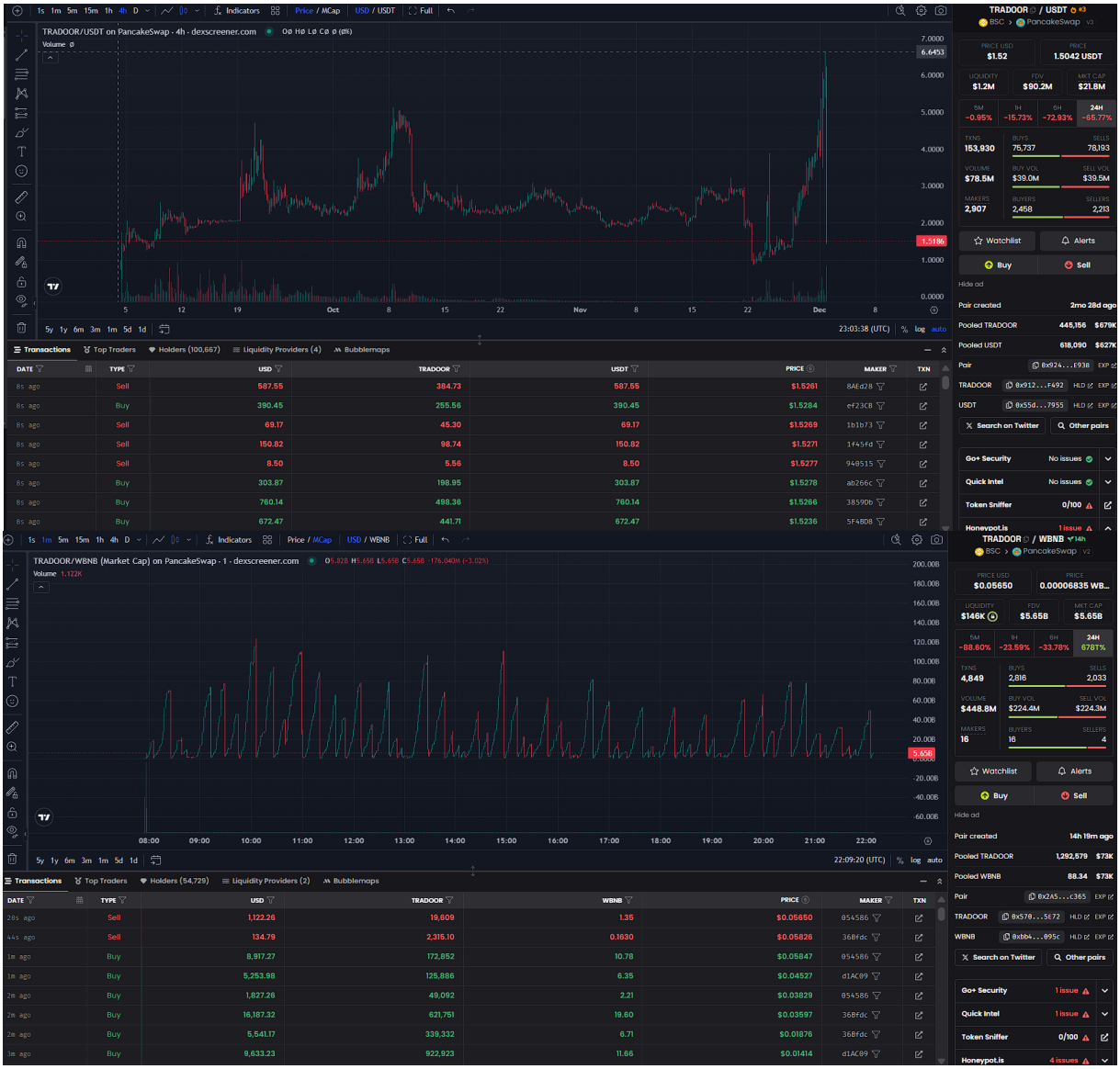

Рисунок 2.10 – Порівняння графіків реального та скам-токена Tradoor

Окрім цього, є ще один спосіб розвитку цього виду шахрайства – зловмисники створюють фейковий токен, у назву якого вбудовують фішингове

посилання, назву сайту, соціальну мережу або прямі заклики на кшталт «Claim reward», «Connect wallet», «Get 500% APY» (див. рисунок 2.11).

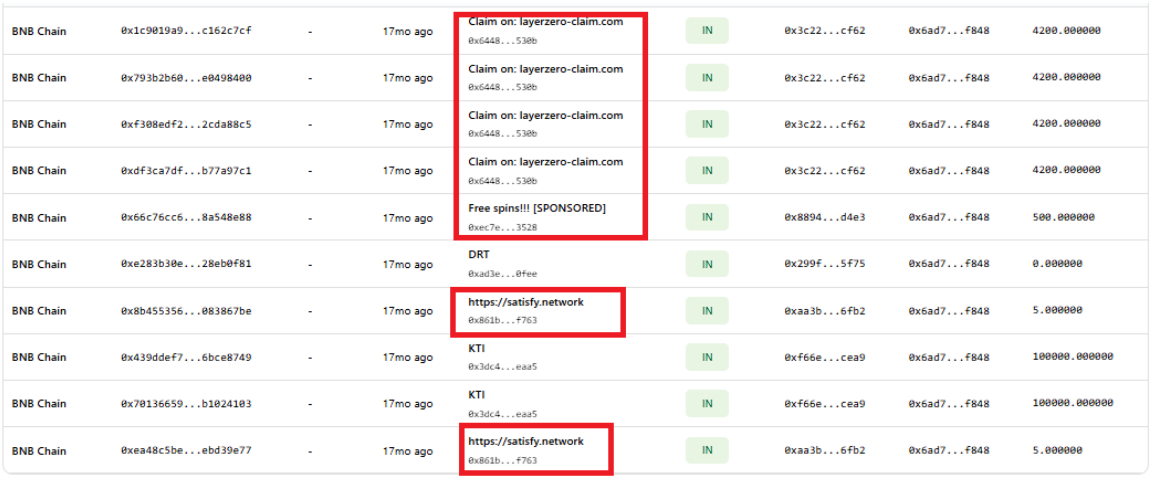

Рисунок 2.11 – Вигляд токенів з фішинговим посиланням

Зацікавившись, жертва переходить на зазначений сайт, який зазвичай є копією відомих сервісів чи сайтом для отримання нагороди. Там її спонукають підписати небезпечні транзакції, надати дозволи, через які шахраї можуть отримати контроль над гаманцем і викрасти активи.

Етап 2 – взаємодія користувача з легальними контрактами. Перед початком шахрайської активності користувач зазвичай здійснює взаємодію з легітимними смартконтрактами – бере участь у продажах токенів, стейкінгу, використанні DeFi-протоколів чи DApps. Всі такі взаємодії фіксуються в блокчейні, що створює прозорі поведінкові сигнатури. Шахраї активно використовують цю відкритість блокчейну: вони аналізують дані з блокчейну, виявляють адреси, що нещодавно виконували транзакції з реальними контрактами, та формують базу «активних» та «платоспроможних» гаманців. Ці адреси стають основними цілями для подальшої масової розсилки фальшивих токенів.

В результаті користувач стає жертвою атаки не через власну помилку, а через властиву блокчейну прозорість і відкритість до даних. Це робить такі атаки масштабними, автоматизованими та часто непомітними для користувача до моменту, коли він вирішить взаємодіяти з отриманим токеном.

Наприклад, власник 0x227FA8B44ca5A824E04038ecDd8699759cB8351d хоче перевірити свій гаманець на згадані вище види шахрайства. Переходить в розроблений інструмент, вводить свою адресу, обирає блокчейн на якому буде відбуватись перевірка, в даному випадку, серед активних BNB Smart Chain (BSC), який і оберемо. Після завантаження усіх даних в блоці Protfolio користувач може побачити токени які він не купував/клеймив (див. рисунок 2.12).

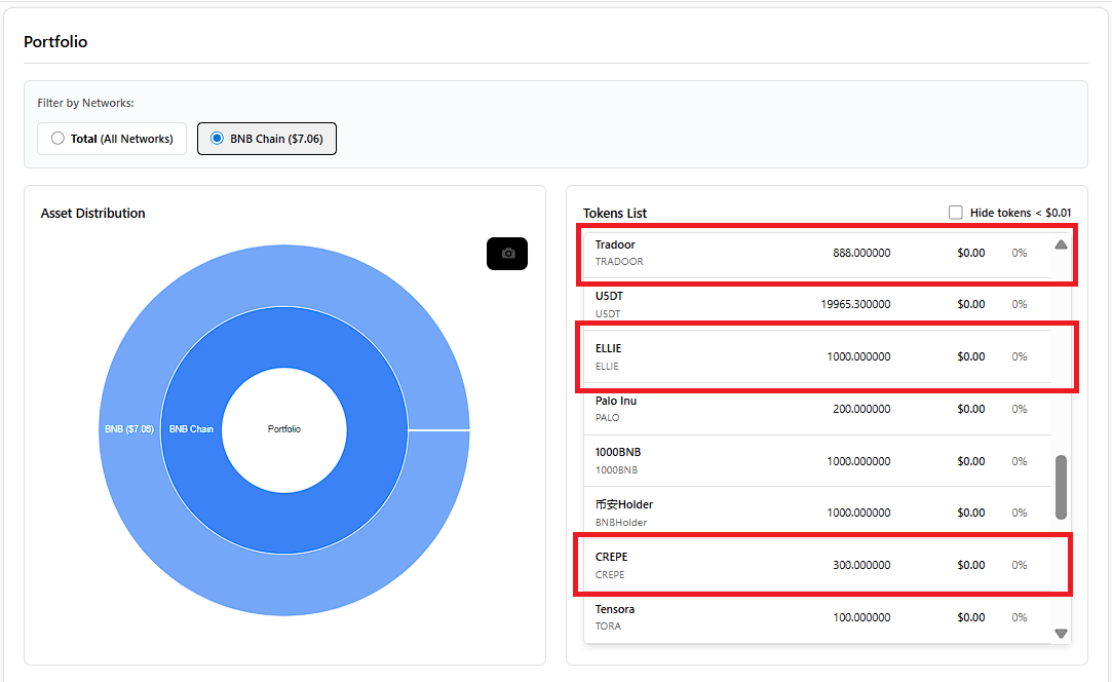

Рисунок 2.12 – Перегляд активів на 0x227FA8B44ca5A824E04038ecDd8699759cB8351d

Але як ця адреса їх отримала? Для цього спускаємось нижче до блоку Transaction History. У другій вкладці – Token Transfers, користувач може побачити історію взаємодії з токенами. З рисунку 2.13 видно, що після транзакції яку адреса здійснила – свапнула (Exact Input V2Swap) токени «Koala» (0xF758Ed7351420ef648A4A8b2080C5787f51Cbb30) через кросчейн платформу для обміну активами – Transit Finance (0x07964f135f276412b3182a3B2407b8dd45000000), отримала на свій гаманець кілька токенів: CREPE (0xcF4e482B40F1F7d73D12fE4963bF2a23aF9e149C), ELLIE (0x5F058E67F1bC6292Ecc25C923C1bE1d2a54486A5) та Tradoor (0x5707c78C86eB3E1C121Fd79e6b38a5C2Ca835E72) за один і той самий час.

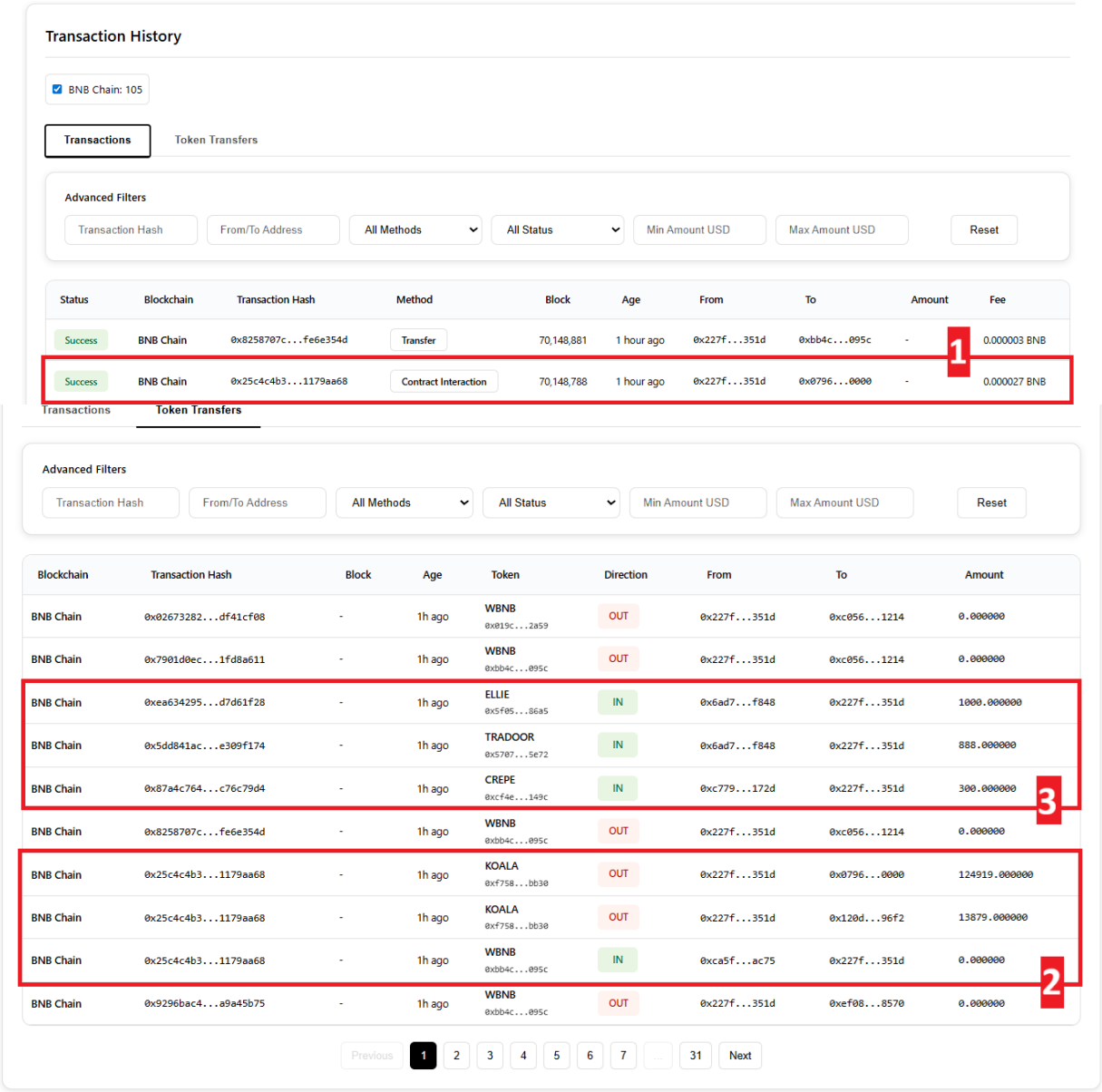

Рисунок 2.13 – Перевірка процесу отримання токенів CREPE, ELLIE та Tradoor

Перевірити достовірність цього можна на сканері відповідного блокчейну, в даному випадку це BscScan – сканер блокчейну BNB Smart Chain, перейти на який можна через посилання у футері інструменту, чи безпосередньо до перегляду хешей транзакцій чи адрес для цього просто натиснувши на них. Наприклад, на рисунку 2.14 подано скріншот з BscScan в якому показано як в результаті взаємодії з платформою Transit Finance, користувач зміг отримати токен Wrapped BNB (WBNB).

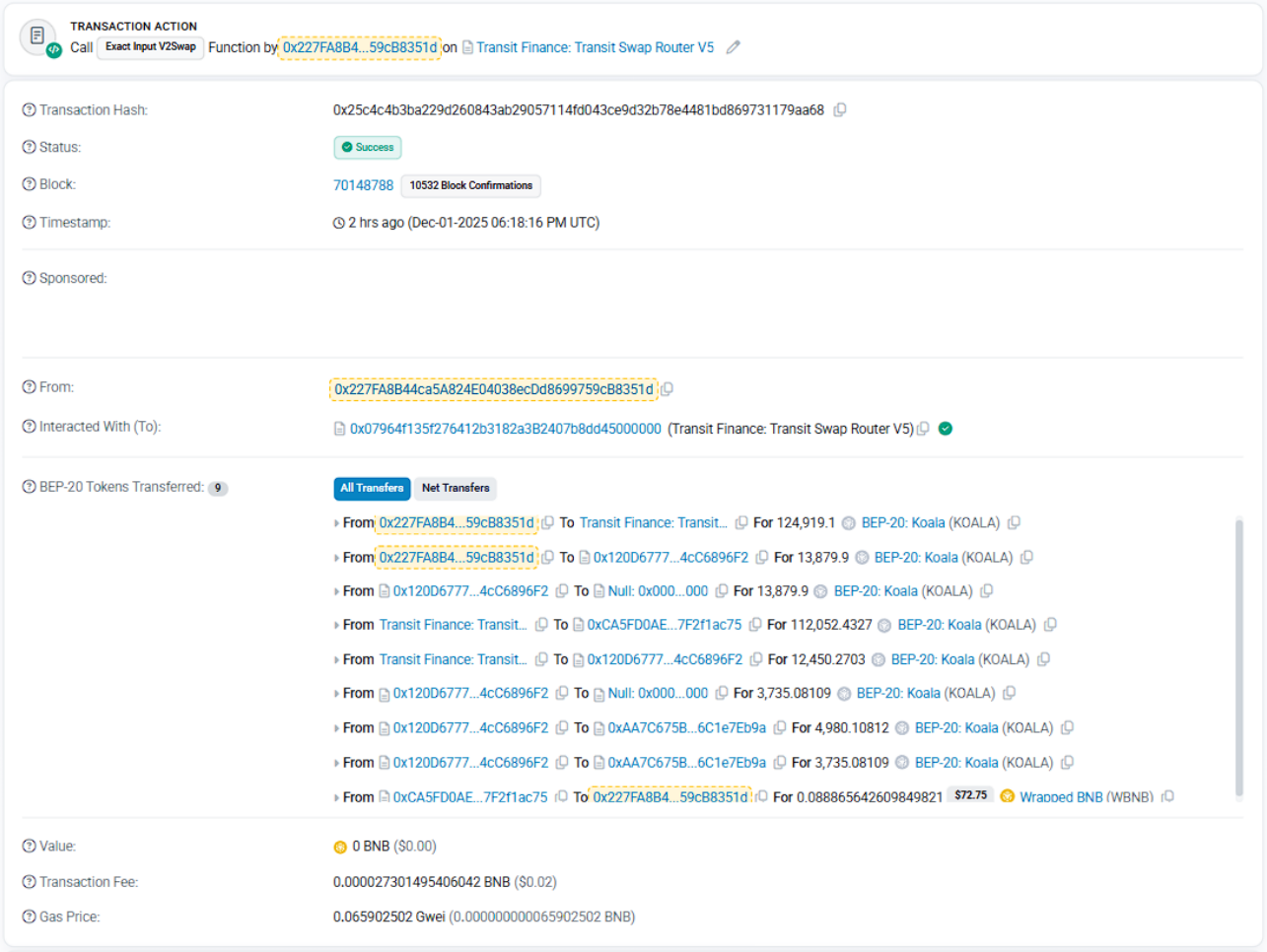

Рисунок 2.14 – Перегляд транзакції, де користувач взаємодіяв з Transit Finance

Етап 3 – масове розсилання скам-токенів. На цьому етапі шахраї здійснюють автоматизовану розсилку підроблених токенів на велику кількість гаманців. Використовуючи скрипти або спеціальні боти, вони відправляють тисячі транзакцій із мінімальною вартістю газу, щоб їхні токени з'явилися у переліку активів користувача.

Як видно з рисунку 2.15, скам-токен Tradoor (0x5707c78C…2Ca835E72) був розісланий ще на значну кількість гаманців.

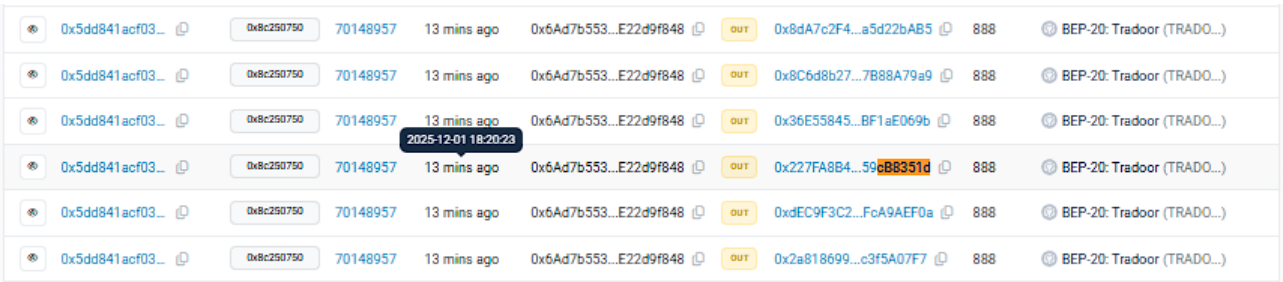

Рисунок 2.15 – Масове розсилання скам-токенів

З даного скріншоту видно, що адреса ймовірного шахрая – 0x6Ad7b553D3622C7D1cCfe702E5a8BC7E22d9f848, на це вказує масове надсилання однакової кількості токенів (по 888) на кожен гаманець за короткий період часу. Така однотипність і висока частота транзакцій може свідчити про використання скриптів або ботів для масового поширення скам-токенів. Ще одним індикатором є повторюваність хешів транзакцій: для значної кількості відправлених операцій використовується той самий хеш, що вказує на формування транзакцій у межах одного групового виклику (batch operation). Це підтверджує, що розсилання відбувається не вручну, а за допомогою автоматизованого інструмента.

При переході за цим хешем (див. рисунок 2.16) видно, що всі операції ініціювалися через контракт 0x8Ac78419A75f5fb42d057F3494687834cc59146A, який, ймовірно, є ботом призначеним для масової дистрибуції токенів. Використання такого контракту підкреслює системність розсилки, що є типовою ознакою для будь-яких шахрайських схем.

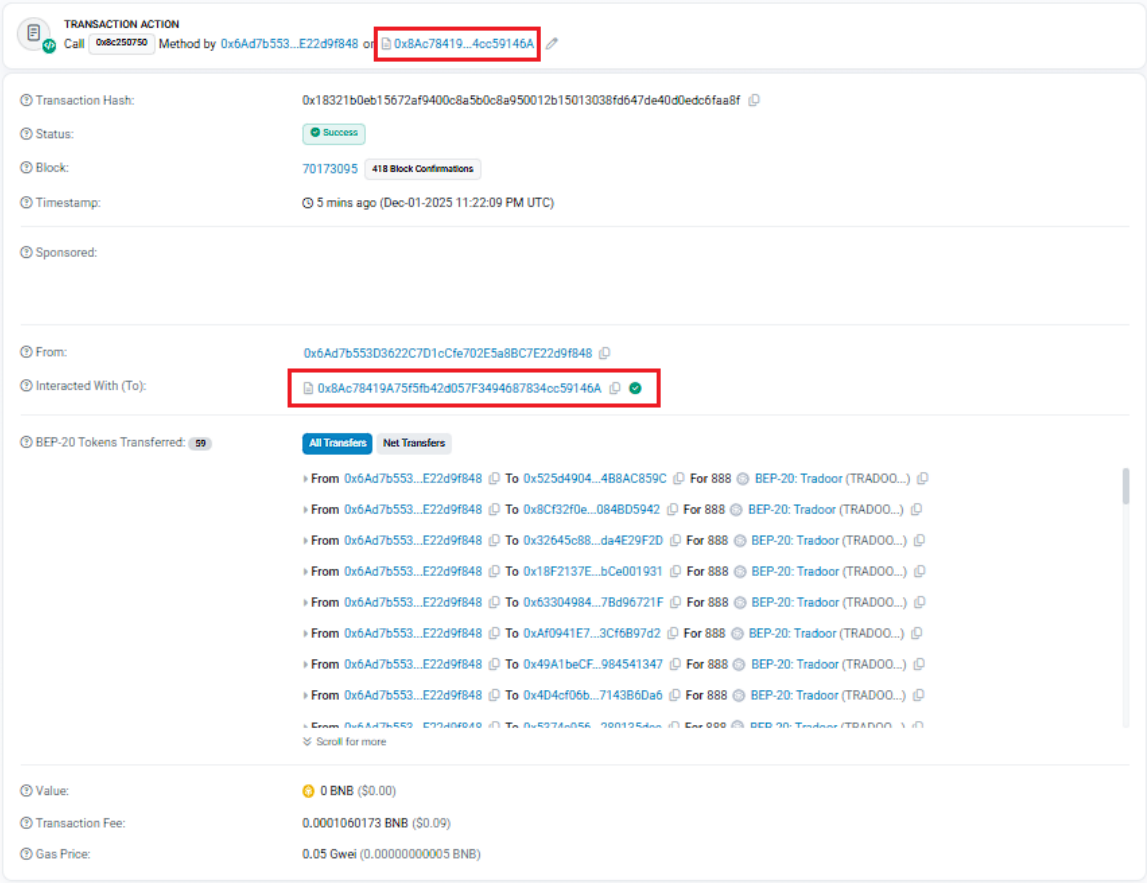

Рисунок 2.16 – Масове розсилання токенів за допомогою контракту

Етап 4, залучення жертви до взаємодії з токеном. Після того, як токени надходять на гаманець жертви, подальша взаємодія з ними залежить від кінцевої мети зловмисника. У деяких випадках токени залишається «сплячими», тобто вони просто надходять на гаманець та автоматично приховуються через свою майже нульову вартість. Користувач може навіть не усвідомлювати, що його обдурили, доки не перегляне історію транзакцій чи не перевірить баланс через різні сервіси (див. рисунок 2.17).

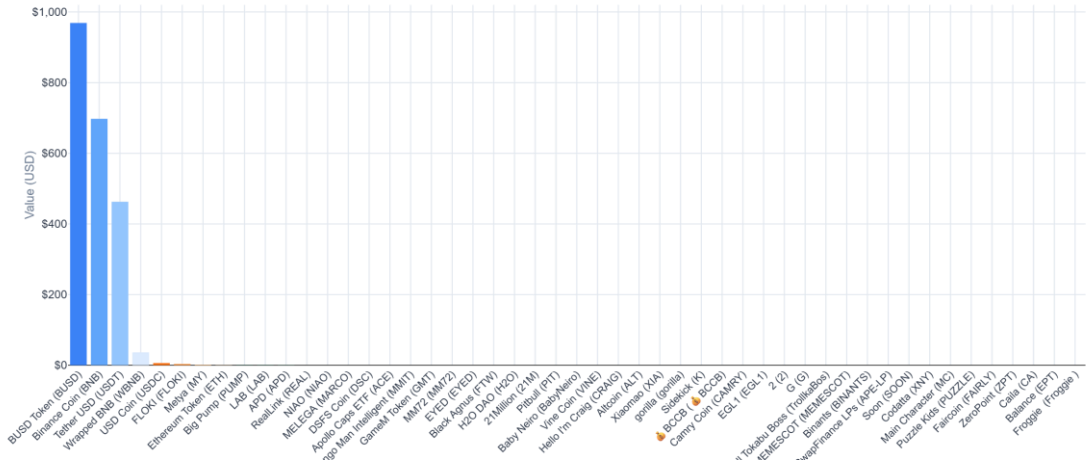

Рисунок 2.17 – Графік вартості активів

В інших випадках, шахраї будуть намагатись привернути увагу користувача до токена: це може проявлятися у вигляді «накачування» токену великими обсягами, повідомлень про нібито бонуси або через посилання на фішингові сайти, які пропонують підключити гаманець і здійснити певні дії. Така стратегія спрямована на те, щоб жертва взаємодіяла з контрактом, що відкриває шахраю можливість контролювати активи.

Таким чином, взаємодія жертви з токеном залежить від того, що ставить шахрай за мету: надіслати токен для його статистики, змусити користувача продати його чи підключитися до підозрілого контракту.

Етап 5 – етап взаємодії адреси з шахрайським токеном. Коли користувач переходить до реальної дії з отриманим токеном – наприклад, намагається його продати, обміняти, клеймити обіцяні «нагороди» чи просто підтверджує запропоновану транзакцію – саме тут шахрайська схема переходить у свою

найбільш небезпечну фазу. В цей момент жертва фактично взаємодіє зі смартконтрактом, у якому прихована шкідлива логіка.

Зазвичай користувач вважає, що виконує стандартну операцію, тому не помічає ризику. Проте підтвердження такої взаємодії може означати, що він:

– Надає контракту дозвіл керувати його активами.

– Активує функції, які блокують продаж або переведення токена.

– Вмикає приховані механізми, через які шахраї зможуть надалі списувати кошти з гаманця.

Після першого підтвердження контракт може одразу або з певною затримкою виконати шкідливі дії: обмежити користувача в операціях із токеном чи вивести активи з його гаманця. Найважливіше тут те, що ініціатором взаємодії є сама жертва, тож система сприймає це як «дозволену» дію.

Таким чином, цей етап є моментом, коли потенційний ризик перетворюється на реальну загрозу, а подальша шкода залежить від того, які дозволи або дії користувач несвідомо підтвердив.

Етап 6, експлуатація активів жертви. Після того, як жертва надала дозвіл (approval) чи виконала взаємодію з контрактом (наприклад, спробувала продати токен чи клеймити «нагороду»), шахрайська логіка активується. Це може статися одразу або з затримкою.

Отримавши доступ до активів жертви шахрай може:

– Автоматичне стягнення коштів (drain) з гаманця через approval.

– Списання ETH, USDT, BNB чи інших активів.

– Передача активів на адреси шахрая або пов'язані гаманці.

Шкода тут максимальна: жертва втрачає не тільки токен, а й інші активи. Якщо атака масова, шахрай може дренувати десятки гаманців одночасно, накопичуючи прибуток.

Етап 7. Замітання слідів та монетизація. Після завершення атаки шахраї починають приховувати свою діяльність, щоб уникнути подальшого відстеження. Спершу вони виводять отримані активи через криптоміксери, кросчейн-мости або централізовані біржі, що дозволяє розмити слід транзакцій та ускладнити їхнє відстеження.

Далі зловмисники можуть змінити або повністю знищити контракт токена, використовуючи SelfDestruct чи зміну власника, а у випадку проксі – оновити імплементацію. Це допомагає приховати шкідливі функції та мінімізувати можливість технічного аналізу.

Після цього цикл може повторюватися: зловмисники створюють новий токен, запускають чергову масову розсилку та продовжують використовувати ту ж тактику для отримання нових жертв.

В даному випадку, масова розсилка скам токену Tradoor (0x5707…5E72), ймовірно, здійснюється з метою штучного збільшення кількості холдерів та створення ілюзії «справжнього токена». Такий підхід зменшує підозри у користувачів під час поверхневого або навіть глибшого аналізу, оскільки велика кількість адрес тримає цей токен, і це може виглядати як природній інтерес до проєкту. Однак, переглянувши рисунок 2.10 ще раз, можна побачити, контракт цього токена має ознаки honeypot, що свідчить про повний контроль активу з боку шахрая. Саме зловмисник штучно формує обсяги торгівлі та створює видимість «живого» проєкту, щоб неуважний користувач навіть при глибшому аналізі міг потрапити у пастку.

Також з рисунку 2.10 видно, що реальний токен Tradoor у цей період демонстрував стрімке зростання вартості, що створювало серед користувачів відчутний ефект FOMO. Шахрай, ймовірно, і використав цей момент: розраховував, що частина користувачів буде поспіхом шукати спосіб придбати цей актив і під час пошуку просто не зверне уваги на контракт. У результаті жертва могла обрати підроблений токен замість справжнього, здійснити покупку та втратити кошти, які безпосередньо переходили до зловмисника.

Таким чином, масова розсилка скам-токена слугує лише інструментом створення штучного інформаційного шуму, який підсилює ілюзію популярності активу. Реальними ж жертвами стають користувачі, які на тлі зростання вартості справжнього токена Tradoor та під дією FOMO намагаються його придбати. Шахрай розраховує, що під час пошуку користувач неуважно перевірить деталі та випадково купить саме підроблений токен, а не справжній актив, внаслідок чого кошти одразу підуть на адресу зловмисника.

Як видно з кількості інформації, яку необхідно проаналізувати при виявленні шахрайських схем, автоматичне отримання всіх необхідних показників через безкоштовні API є обмеженим. Тому для ефективної оцінки ризику було розроблено інтерактивний підхід, який залучає користувача до процесу аналізу. Такий метод має подвійний ефект: юзери не лише отримують оцінку, але й можливість самостійно проаналізувати вектори атаки, розібратися в логіці шахрайства та прокачати обізнаність для майбутніх взаємодій із контрактами.

Саме для цього в модулі Risk Profile розроблена анкета, яка складається з кількох коротких запитань, що охоплюють взаємодію з airdrop-токеном. Після її заповнення користувач отримує результат в який входять: оцінка ризику (Risk Score), тип учасника та рекомендації щодо подальших дій.

Наприклад спробуємо пройти дане опитування для перевірки ризику скам-токена Tradoor. На рисунках 2.18-2.19 подано відповіді на запитання.

Питання 1. Для перевірки токена на ознаки шахрайства, пропонується перейти на сайти Apespace чи Token Sniffer. В даному випадку проведемо аналіз через сервіс Apespace, з рисунку 2.18 видно, що токен виглядає дуже підозріло: є ознаки Honeypot, є мітка що цей токен «Airdrop», контракт не верифікований.

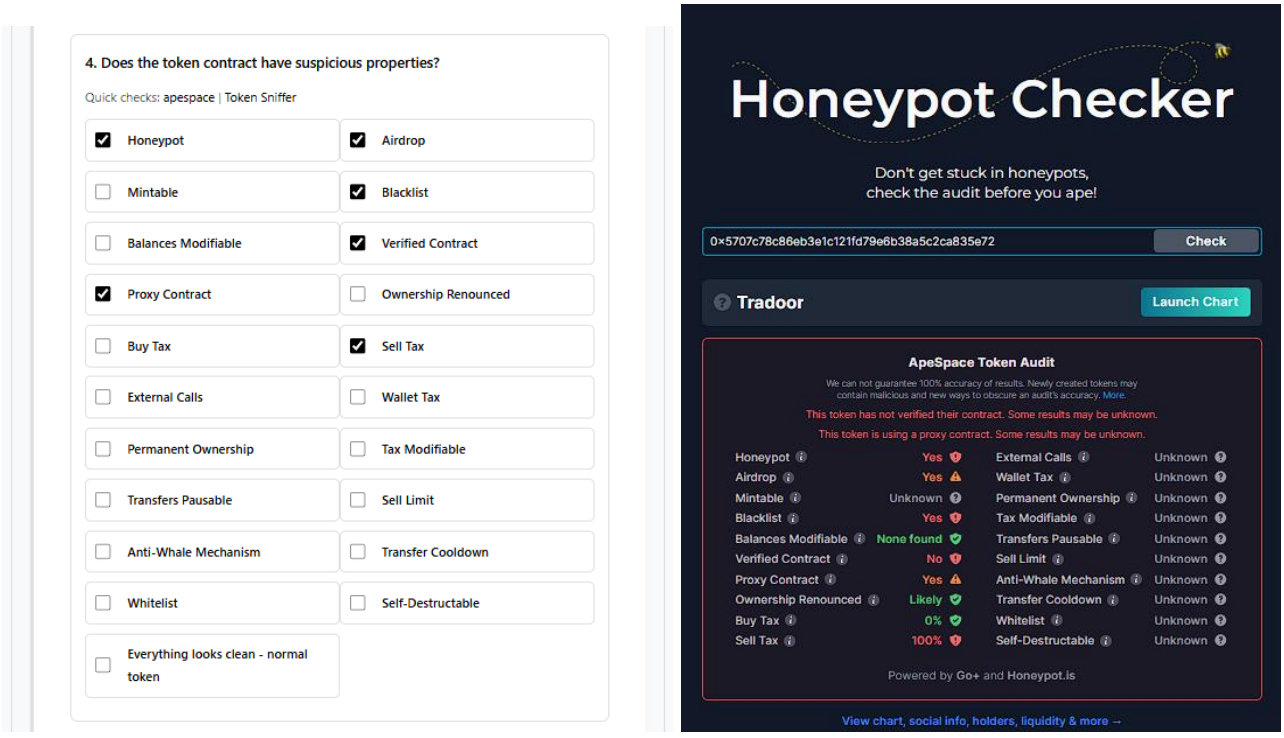

Рисунок 2.18 – Заповнення питання №1 за допомогою сервісу Apespace

На рисунку 2.19 подано відповіді на наступні питання:

– Питання 2. Ні, токен не містить закликів до дії.

– Питання 3. Токен створено два дні тому.

– Питання 4-6. Перейшовши за допоміжними посиланнями під питанням, було знайдено, що dev (developer) має <0.01% токенів та кити (whales, top 10 wallets) тримають 93,39%, причому в однієї 70,63% під управлінням, більшість холдерів це звичайні адреси.

– Питання 7. Перейшовши за посиланнями на GMGN чи DEX Screener, видно що токен має підозрілий патерн торгівлі: на графіку обігу токена спостерігаються повторювані різкі піки та обвали (див. рисунок 2.10).

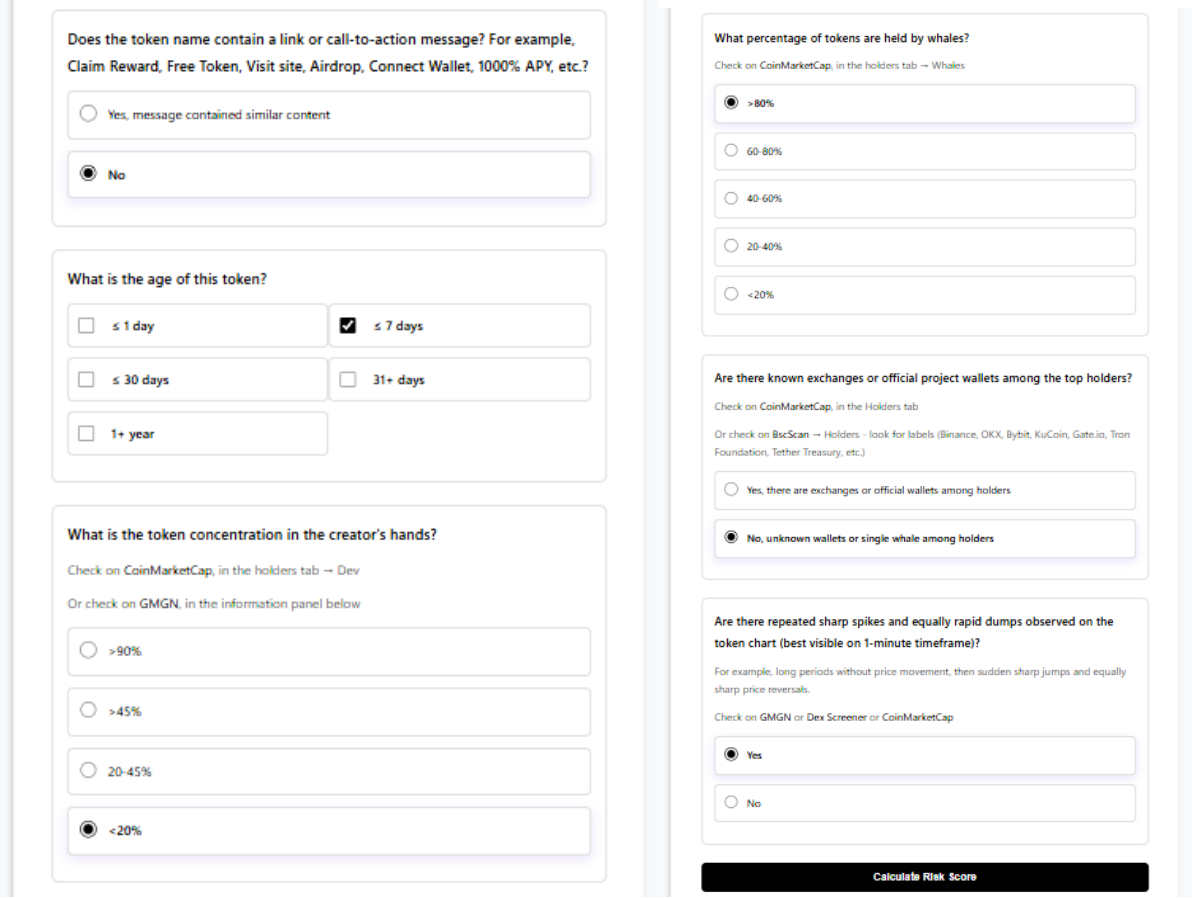

Рисунок 2.19 – Відповіді на питання №2-7

На рисунку 2.20 представлено результат оцінки ризику «Airdrop Scam Tokens», який підтверджує, що аналізований токен має ознаки шахрайства.

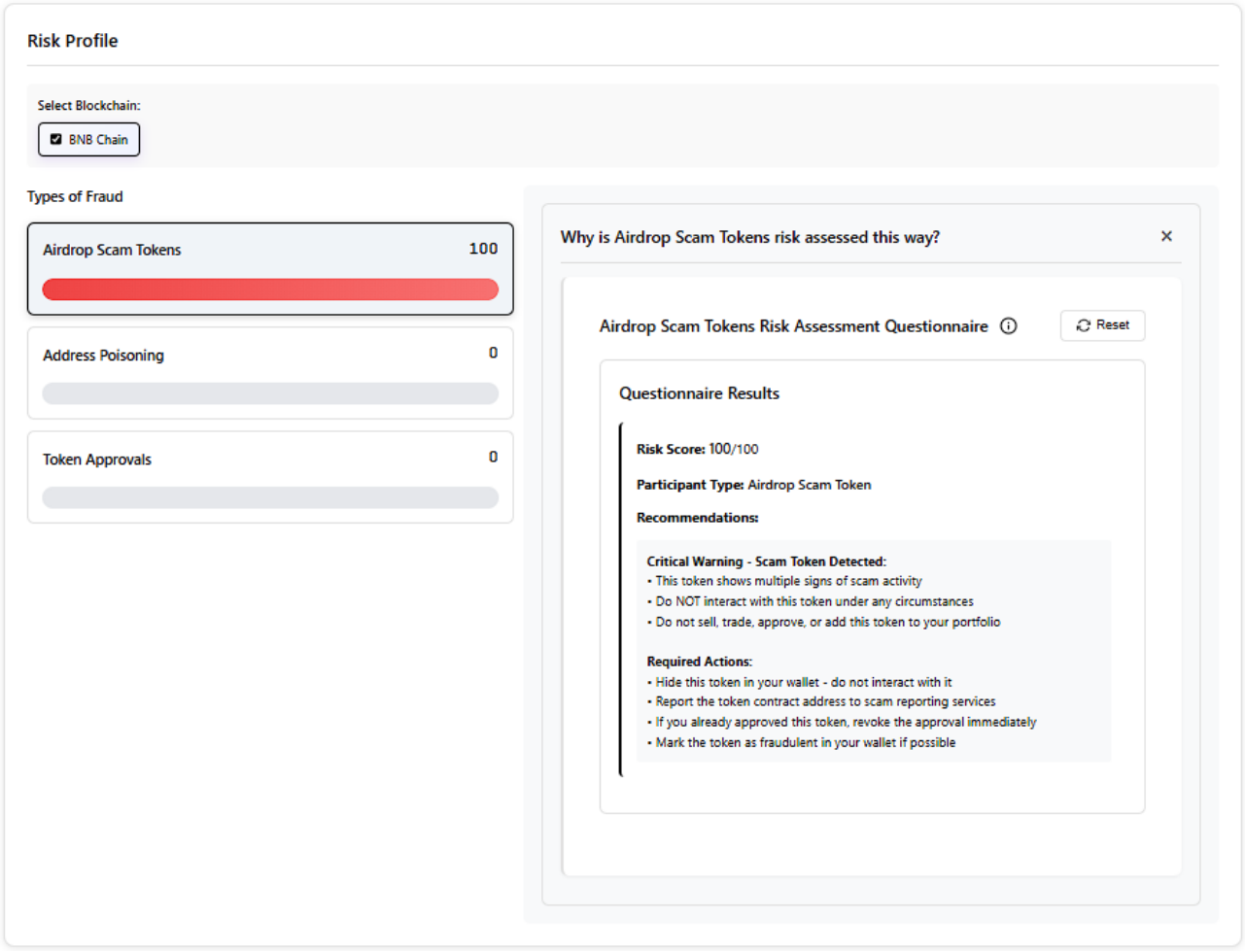

Рисунок 2.20 – Результат проходження анкети

Детальний опис розслідування, наведений вище, буде винесений на окрему сторінку сайту. Користувач зможе прочитати його пройшовши через посилання «Detailed Guide: Detailed breakdown of the «Airdrop Scam Tokens» scheme and tool instructions», яке відкривається після натискання на іконку інформації біля назви шахрайства. Там користувач може глибше ознайомитися з логікою Airdrop Scam Tokens, навчитися проводити самостійну перевірку чи просто прочитати інструкцію як пройти цю інтерактивну анкету.

Для оцінки ефективності анкети проведено тестування на 250 реальних адресах, промаркований список яких знаходиться в додатку А. В таблиці 2.1 подано зіставлення фактичного статусу адрес із результатами, отриманими за допомогою анкети. Це дає змогу визначити частку правильних відповідей, а також кількість хибнопозитивних і хибнонегативних спрацювань, що слугує основою для оцінки загальної якості й надійності роботи анкети.

Таблиця 2.1 – Результати тестування анкети

Категорія	Кількість адрес	Правильно визначено	Хибно-позитивні	Хибно-негативні	Точність, %
Жертва	50	50	0	0	100
Шахрай	50	50	0	0	100
Скам-токен	50	50	0	0	100
Чистий токен	50	50	0	0	100
Чиста адреса	50	50	0	0	100

Отримані результати підтверджують високу ефективність запропонова-них в анкеті правил. Водночас слід враховувати, що при неточному заповненні цієї анкети користувачем можливі певні відхилення. Це може бути зумовлено відсутністю необхідних данних, тимчасовою недостоступністю зовнішніх сервісів чи розвитком атаки. Проте навіть у таких ситуаціях інтерактивний підхід забезпечує значно вищу точність і глибше розуміння ризиків, ніж просте покладання на автоматичні інструменти, які часто дають хибнопозитивні/ хибнонегативні спрацювання через обмеженість відкритих API.

РОЗДІЛ 3. ЗАБЕЗПЕЧЕННЯ ЗАХИСТУ ІНФОРМАЦІЙНИХ РЕСУРСІВ СИСТЕМИ

3.1 Безпечна взаємодія з API

Безпечна взаємодія з зовнішніми API є фундаментальною складовою архітектури системи аналізу блокчейн-адрес, оскільки саме через ці інтерфейси відбувається отримання on-chain даних. У системі використовуються безліч API сервісів, які забезпечують уніфікований доступ до транзакційної історії, балансів, метаданих токенів та смарт-контрактів на різних блокчейнах.

Слід чітко диференціювати публічні компоненти проекту від приватних даних, пов'язаних із його конкретним розгортанням. Оскільки інструмент має відкритий вихідний код, його аналітична логіка та алгоритми є публічними, так само як і on-chain дані, які він обробляє. Таким чином, завдання захисту взаємодії з API полягає не в приховуванні цих загальнодоступних аспектів, а в забезпеченні безпеки конфіденційних даних конкретного екземпляра системи.

По-перше, це стосується API-ключів, що належать саме цьому розгортанню, з метою запобігання несанкціонованим фінансовим витратам та вичерпанню квот. По-друге, це захист операційних патернів: хоча як інструмент аналізує адреси, відомо з вихідного коду, які саме адреси, в якій послідовності та в який момент часу аналізує конкретна людина, є чутливою операційною інформацією. По-третє, це стосується приватної конфігурації екземпляра, яка може включати кастомно-навчені моделі або набори правил, що не входять до публічного репозиторію.

Для вирішення цих завдань, в архітектурі системи реалізовано комплексний підхід, що базується на ізоляції ключів доступу через архітектурну абстракцію. В системі, що складається з багатьох мікросервісів, надання кожному з них прямого доступу до API ключів призводить до низки критичних ризиків:

– Порушення принципу мінімальних привілеїв (PoLP). Сервіс скорингу, якому для роботи потрібен лише результат аналізу, не повинен мати доступу до ключів API, що збирають вихідні дані.

– Збільшення поверхні атаки. Компрометація будь-якого з цих мікросервісів автоматично призводить до компрометації ключів зовнішніх API.

– Ускладнена ротація секретів. При необхідності заміни API-ключа довелося б оновлювати конфігурацію, зупиняти та перезапускати кожен мікросервіс, що володіє цим ключем, що призводить до простоїв.

Для вирішення цих проблем впроваджено спеціалізований «Мікросервіс-шлюз збору даних» (DataGatewayService). Цей компонент діє як єдина, вузькоспеціалізована точка входу до всіх зовнішніх постачальників on-chain даних, реалізуючи принцип єдиної відповідальності (Single Responsibility Principle). Замість того, щоб сервіси бізнес-логіки мали прямі залежності від наприклад, AnkrClient, вони взаємодіють виключно з уніфікованим інтерфейсом DataGatewayService [26].

Це є прямою реалізацією принципу мінімальних привілеїв (PoLP), сервіси бізнес-логіки мають привілеї лише на запит даних у шлюзу, але не мають привілеїв на читання секретів, необхідних для отримання цих даних (див. лістинг 3.1).

Лістинг 3.1 – Реалізація TransactionService з використанням шлюзу

```
from app.services.data_gateway_service import DataGatewayService
class TransactionService:
   def __init__(self):
      self.data_gateway = DataGatewayService()
   async def get_tx_data(self, address: str):
      raw_transactions = await self.data_gateway.get_all_transactions(...)
```

Вся логіка отримання, кешування та ротації API-ключів інкапсульована всередині DataGatewayService за допомогою вбудованого компонента

SecretsManager, який є єдиним компонентом у системі, що містить для доступу до секретів. У поточній реалізації він використовує режим «fallback», зчитуючи ключі зі змінних середовища.

Також в архітектуру закладено основі для безпечного розгортання у виробничих середовищах. Шляхом встановлення змінних, SecretsManager автоматично перемикається на використання AWS Secrets Manager або HashiCorp Vault. Для розгортання в AWS доступ до секретів жорстко обмежується лише для IAM-ролі, асоційованої з процесом DataGatewayService. Це реалізується через політику доступу, що унеможливлює читання секретів будь-яким іншим компонентом системи (див. лістинг 3.2) [27].

Лістинг 3.2 – IAM-політика для доступу до AWS Secrets Manager

```
{
   "Effect": "Allow",
   "Action": "secretsmanager:GetSecretValue",
   "Resource": [
   "arn:aws:secretsmanager:region:account:secret:ankr-api-key-
*" ]
}
```

Завдяки цьому, ключі завантажуються «на льоту» безпосередньо в оперативну пам'ять процесу шлюзу і ніколи не зберігаються на диску у відкритому вигляді.

Забезпечення конфіденційності під час передачі. Для цього всі вихідні з'єднання, що ініціюються до ендпоінтів зовнішніх API-сервісів, примусово використовують протокол HTTPS (TLS 1.2+). Програмно це реалізовано через HTTP-клієнт httpx, де параметр верифікації сертифіката явно ввімкнено (verify=True). Це означає, що перед відправкою будь-яких даних, клієнт проводить повне TLS-рукостискання (handshake), під час якого він перевіряє автентичність SSL-сертифіката сервера. Цей механізм є ключовим захистом від атак «Людина посередині» (Man-in-the-Middle). Якщо зловмисник спробує перехопити з'єднання, надавши фальшивий сертифікат, httpx розпізнає підроб-

ку, негайно розірве з'єднання і видасть помилку верифікації. Таким чином, система гарантує, що критично важливі дані – як самі API-ключі в заголовках автентифікації, так і параметри запитів, що розкривають патерни аналізу – ніколи не покинуть шлюз і не будуть відправлені неперевіреному вузлу.

Зловживання або неконтрольоване використання квот API є прямим операційним та фінансовим ризиком. Оскільки для системи аналізу шахрайства важливо отримувати дані в реальному часі, агресивне кешування даних (наприклад, історії транзакцій) є неприйнятним. Це призвело б до аналізу застарілих даних, оскільки користувач не зміг би миттєво побачити нові, потенційно шахрайські транзакції.

Тому, для захисту від вичерпання квот, основним механізмом є не кешування, а примусове обмеження частоти запитів. Цей функціонал реалізовано безпосередньо у DataGatewayService за допомогою вбудованого компонента RateLimiter. Компонент використовує ефективний алгоритм «ковзного вікна» (sliding window), який відстежує кількість запитів за останній проміжок часу, забезпечуючи значно точніший контроль, ніж просте «фіксоване вікно» [28]. Ліміти є гнучкими та налаштовуються через змінні середовища, що дозволяє адаптувати їх для різних умов (розробка, тестування, продакшн).

Оскільки даний інструмент є проектом з відкритим вихідним кодом, його архітектура має бути доступною для будь-якого користувача «з коробки». Тому система за замовчуванням налаштована на роботу з Freemium-планом API-провайдерів, таких як Ankr Advanced API, який має суворе обмеження у 30 запитів на хвилину [29].

Водночас, архітектура є гнучкою, адже користувачі, яким потрібна вища пропускна здатність, можуть придбати платну підписку API з вищими лімітами і їм не доведеться модифікувати вихідний код, а достатньо лише змінити значення змінних середовища GATEWAY_RATE_LIMIT і GATEWAY_RATE_WINDOW відповідно до умов їхнього нового плану.

В лістингу 3.3 показано конфігурацію за замовчуванням, орієнтовану на Freemium-план.

Лістинг 3.3 – Конфігурація Rate Limiter для Freemium-плану

```
#Вміст .env файлу
GATEWAY_RATE_LIMIT=30
GATEWAY_RATE_WINDOW=60
self.rate_limiter = RateLimiter(
      max_requests=self.settings.GATEWAY_RATE_LIMIT,
          window_seconds=self.settings.GATEWAY_RATE_WINDOW)
```

Цей підхід забезпечує надійний захист від атак грубого перебору (brute-force) або DoS. Він також є важливим у гіпотетичному сценарії компрометації одного з внутрішніх сервісів. Якщо зловмисник спробує виконати масову кількість запитів через шлюз, RateLimiter негайно заблокує цю аномальну активність, запобігаючи вичерпанню платних квот API та пов'язаним фінансовим збиткам.

Впровадження механізму відмовостійкості (Circuit Breaker). Зовнішні API за своєю природою є ненадійними, вони можуть бути тимчасово недоступні (HTTP 5xx), перевантажені (HTTP 429) або відповідати із затримкою (TimeoutException). Для запобігання каскадним збоям, коли відмова одного зовнішнього сервісу призводить до зависання та відмови всієї системи, впроваджено «Автоматичний вимикач» (Circuit Breaker) [30]. Оскільки проект використовує asyncio для асинхронних операцій, стандартні синхронні бібліотеки (як circuitbreaker) не підходять. Тому було розроблено власну, асинхронну реалізацію – клас AsyncCircuitBreaker. Механізм функціонує як машина станів, що відстежує «здоров'я» зовнішнього API:

– CircuitState.CLOSED (Закрито) – нормальний стан. Запити вільно проходять до API. Лічильник помилок скидається при кожному успішному виклику.

– CircuitState.OPEN (Розімкнено) – стан збою. Досягнуто порогу помилок (failure_threshold). Усі запити негайно блокуються (реалізація Fail-Fast), повертаючи CircuitBreakerError без спроби з'єднання. Запускається таймер відновлення (reset_timeout).

– CircuitState.HALF_OPEN (Напіврозімкнено) – стан перевірки. Після закінчення reset_timeout, «вимикач» дозволяє пройти рівно одному тестовому запиту. Якщо він успішний, стан повертається до CLOSED. Якщо ні, до OPEN.

Для гнучкості, параметри «вимикача» винесено у Settings (Pydantic) і зчитуються зі змінних середовища:

– CIRCUIT_FAILURE_THRESHOLD (за замовчуванням = 5). Кількість поспіль помилок, необхідних для розмикання.

– CIRCUIT_RESET_TIMEOUT (за замовчуванням = 60). Час у секундах, який «вимикач» чекає у стані OPEN перед переходом до HALF_OPEN.

Також важливим є коректне визначення збою. «Вимикач» реагує не на всі помилки. Наприклад, HTTP 404 (Not Found) не є збоєм API. У реалізації _is_failure збоями вважаються лише:

– Серверні помилки HTTP 5xx.

– Перевищення лімітів HTTP 429 (Too Many Requests).

– Таймаути httpx.TimeoutException

Для того, щоб AsyncCircuitBreaker міг «бачити» ці помилки, API-клієнти були модифіковані для генерації виключень при невдалих відповідях, використовуючи response.raise_for_status().

Інтеграція реалізована через дворівневу абстракцію для забезпечення чистоти коду та надійної обробки помилок (див. лістинг 3.4).

Рівень _make_api_call (захист). Створено приватний метод, який безпосередньо «огортає» виклик API логікою circuit_breaker.call(). Це єдина точка, де відбувається взаємодія з «вимикачем».

Рівень _safe_api_call (обробка). Створено другий приватний метод-обгортку, який викликає _make_api_call всередині блоку try...except. Його завдання – перехопити CircuitBreakerError (який виникає при «fail-fast») та інші помилки, залогувати їх і безпечно повернути default_return (напр., None), запобігаючи «падінню» сервісу, що викликав.

Лістинг 3.4 – Дворівнева абстракція для відмовостійкості

```
    def __init__(self):
    self.circuit_breaker = AsyncCircuitBreaker(
    failure_threshold=self.settings.CIRCUIT_FAILURE_THRESHOLD
reset_timeout=self.settings.CIRCUIT_RESET_TIMEOUT)
    async def _make_api_call(self, api_func: Callable, *args,
**kwargs) -> Any:
    return await self.circuit_breaker.call(api_func, *args,
**kwargs)
    async def _safe_api_call(self, api_func: Callable, *args,
default_return=None, **kwargs) -> Any:
    try:
    return await self._make_api_call(api_func, *args, **kwargs)
    except CircuitBreakerError as e:
    print(f"[DataGateway] Circuit breaker OPEN: {str(e)}")
    return default_return
    except (httpx.HTTPStatusError, httpx.TimeoutException) as e:
    print(f"[DataGateway] API error: {type(e).__name__}:
{str(e)[:100]}")
    return default_return
    except Exception as e:
    print(f"[DataGateway] Unexpected error: {type(e).__name__}:
{str(e)[:100]}")
    return default_return
```

Всі методи DataGatewayService були модифіковані для використання _safe_api_call, що гарантує захист зовнішніх викликів (див. лістинг 3.5).

Лістинг 3.5 – Рефакторинг методу get_token_balances

```
    return await self._safe_api_call(
    self._covalent_client.get_token_balances, address,
    chain_name,
    default_return=None)
```

Комбінація цих механізмів в єдиному DataGatewayService перетворює його на надійний та безпечний «фасад». Решта сервісів повністю абстраговані від складності управління ключами, захисту квот та обробки збоїв API, що значно спрощує загальну архітектуру системи.

3.2 Управління та безпечне зберігання ключів доступу

Як відомо, API-ключі є критичними активами для будь-якого програмного забезпечення, що взаємодіє з зовнішніми сервісами. Тому окрім забезпечення безпеки даних під час передачі, що деталізовано у попередньому розділі, невід'ємним компонентом комплексної стратегії захисту є гарантування конфіденційності та цілісності секретів у стані спокою.

Найпоширеніша та водночас найнебезпечніша помилка, відома як жорстке кодування (hardcoding), полягає у збереженні секретів безпосередньо у вихідному коді. Це створює катастрофічну вразливість. Як тільки код потрапляє до системи контролю версій (Git), навіть у приватний репозиторій, ключ стає назавжди «закарбованим» у його історії [31]. Будь-яка людина, що має доступ до коду, автоматично отримує доступ до ключів. Крім того, такий підхід робить процес ротації ключів (їх періодичної заміни) надзвичайно складним, оскільки вимагає зміни коду, повторного тестування та повного перезавантаження (re-deploy) всього додатка.

Для повного усунення цих ризиків, як в попередньому пункті було зазначено, в архітектурі системи реалізовано принцип абстракції через спеціалізований компонент – клас SecretsManager. Його головна мета – бути єдиним, централізованим «сейфом» для всіх секретів. Решта додатка, зокрема DataGatewayService, повністю ізольована від знання про те, де і як зберігаються ключі. DataGatewayService просто «просить» у SecretsManager потрібний йому ключ, а SecretsManager вже сам вирішує, звідки його безпечно дістати.

Ця абстракція є гнучкою, оскільки вона дозволяє SecretsManager підтримувати різні стратегії зберігання, що є важливим для open-source проекту, який має бути простим для одних користувачів і надбезпечним для інших.

Для переважної більшості користувачів, які будуть розгортати цей інструмент на сучасних хмарних PaaS-платформах (Platform as a Service) на кшталт Render, або запускати його локально для тестування, SecretsManager використовує свій резервний механізм. Цей механізм зчитує ключі зі змінних середовища (Environment Variables) [32].

Коли користувач розгортає додаток на Render, він не завантажує жодних .env файлів. Натомість, він відкриває безпечний веб-інтерфейс платформи, переходить у вкладку «Environment Variables» і вводить туди свої ключі COVALENT_API_KEY та ANKR_API_KEY (див. рисунок 3.1).

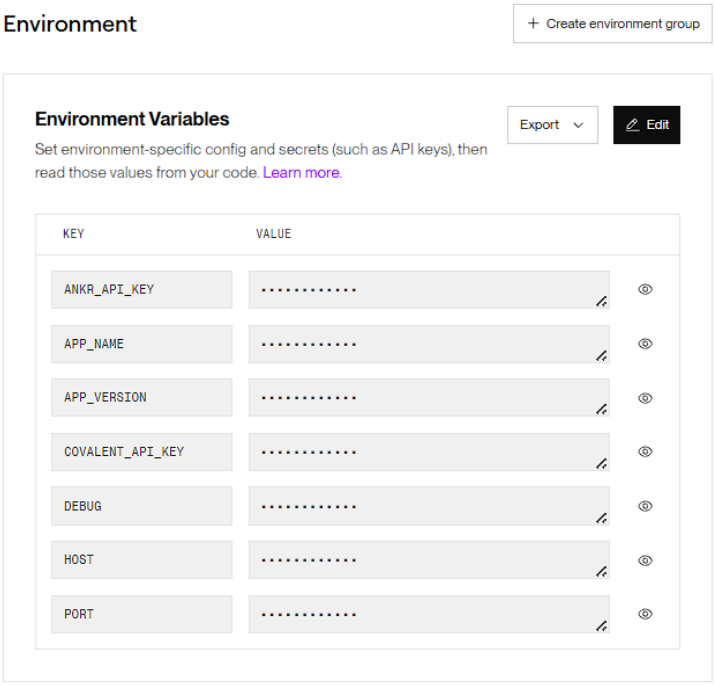

Рисунок 3.1 – Завантаження змінних середовища проєкту

У цей момент Render бере ці ключі, шифрує їх і зберігає у своєму власному захищеному сховищі. Коли Render запускає додаток, він безпечно завантажує ці секрети безпосередньо у середовище виконання нашого процесу. Саме тут і спрацьовує SecretsManager. Його fallback-логіка (реалізована у методі _get_secret_from_env) через self.settings отримує ключі, які Pydantic Settings автоматично зчитує зі змінних середовища процесу (або з .env файлу для локальної розробки) (див. лістинг 3.6). У випадку з Render, де .env файли не використовуються, ключі зчитуваються безпосередньо з оперативної пам'яті

процесу, куди їх завантажує платформа. Ключі ніколи не існують у відкритому вигляді на жорсткому диску сервера і ніколи не потрапляють у Git-репозиторій.

Лістинг 3.6 – Fallback механізм отримання секретів зі змінних середовища

```
    def _get_secret_from_env(self, secret_name: str) ->
Optional[str]:
    env_mapping = {
    "covalent_api_key": self.settings.COVALENT_API_KEY,
    "ankr_api_key": self.settings.ANKR_API_KEY,
    }
    return env_mapping.get(secret_name.lower())

    async def get_secret(self, secret_name: str, ...) ->
Optional[str]:
    if not secret_value:
    secret_value = self._get_secret_from_env(secret_name)
    return secret_value
```

Таким чином, для користувачів досягається баланс: максимальна простота, просто вставити ключ у веб-форму, та високий рівень безпеки, де ключі зашифровані ніколи не лежать у коді.

Для великих компаній (Enterprise), які мають власні суворі політики безпеки, SecretsManager має архітектурну підготовку для інтеграції з AWS Secrets Manager. Метод _get_secret_from_aws містить детальні коментарі та готовий до реалізації код (закоментований), який можна активувати після налаштування AWS infrastructure (IAM Role, IAM Policy). Натомість, додаток запускається на сервері AWS (наприклад, ECS) зі спеціальною «цифровою ідентичністю» IAM-роллю (наприклад, Data-Gateway-Role).

Адміністратор AWS створює IAM-політику (яка згадувалась у попередньому пункті, лістинг 3.2), що діє як «дозвіл». Ця політика чітко каже: «Дозволити суб'єкту з роллю Data-Gateway-Role виконувати дію secretsmanager:GetSecretValue (читати секрет), але тільки для ресурсів covalent-api-key та ankr-api-key».

Коли SecretsManager (метод _get_secret_from_aws) спробує отримати ключ, він використає бібліотеку boto3. boto3 автоматично «побачить», що він запущений з IAM-роллю, і використає її для автентифікації в AWS. AWS перевірить політику і, якщо все гаразд, поверне ключ. Це максимальний рівень безпеки, оскільки ключі ніколи не покидають зашифроване сховище AWS і видаються лише тимчасово, довіреному процесу [33].

Нарешті, DataGatewayService використовує ці секрети максимально ефективно. Звернення до змінних середовища або, тим більше, до AWS Secrets Manager при кожному запиті є неефективним, тому реалізовано два механізми оптимізації.

Ліниве завантаження (Lazy Loading). Код не завантажує ключі під час запуску, він чекає, і лише коли надходить перший API-запит від користувача, DataGatewayService викликає _ensure_keys_loaded. Це прискорює запуск додатку та економить ресурси (див. лістинг 3.7) [34].

Лістинг 3.7 – Ліниве завантаження у DataGatewayService

```
    async def _ensure_keys_loaded(self):
       if self._keys_loaded:
             return
    ankr_key                          =                          await
self.secrets_manager.get_secret("ankr_api_key")
    self._ankr_client  =  AnkrClient(api_key=ankr_key)  if  ankr_key
else AnkrClient()
    self._covalent_client  =  CovalentClient(api_key=covalent_key)
if covalent_key else CovalentClient()
    self._keys_loaded = True
```

Кешування секретів. Після того, як SecretsManager (у методі get_secret) вперше отримав ключ (з Render чи AWS), він зберігає його у self._secrets_cache на _cache_ttl = 3600 (1 годину). Це означає, що додаток робить лише один запит на отримання секрету на годину, а не на кожен клік, що є надзвичайно ефективним (див. лістинг 3.8) [35].

Лістинг 3.8 – Налаштування механізму кешування секретів

```
def __init__(self):
self.settings = get_settings()
self._secrets_cache = {}
self._secrets_cache_time = {}
self._cache_ttl = 3600  # 1 година
```

Таким чином, реалізований підхід до управління ключами є багаторівневим та гнучким. Він повністю усуває ризики жорсткого кодування. Завдяки абстракції SecretsManager, проєкт є одночасно надзвичайно простим у розгортанні для звичайних користувачів через PaaS-платформи (як Render), де безпека забезпечується самою платформою, і готовим до впровадження у складних корпоративних середовищах (як AWS), які вимагають гранулярного контролю доступу через IAM. Оптимізація через «ліниве завантаження» та кешування гарантує, що ця безпека не впливає на продуктивність системи.

3.3. Заходи протидії типовим веб-атакам

Як було проаналізовано у попередніх пунктах, архітектура системи, хоч і функціональна, є вразливою до низки типових веб-атак, визначених у стандартах OWASP Top 10. Публічний Backend API є основною поверхнею атаки, тоді як Frontend є вектором для атак на кінцевого користувача. Для нейтралізації цих загроз була впроваджена багаторівнева стратегія захисту, яка охоплює захист на рівні сервера (обмеження запитів, обробка помилок), на рівні передачі даних (заголовки безпеки) та на рівні клієнта (санітизація виводу). Нижче детально описано кожен із впроваджених механізмів.

Захист від атак «Відмова в обслуговуванні» (DoS). Внутрішній RateLimiter шлюзу DataGatewayService захищає лише зовнішні API-квоти, але не захищає власний сервер. Зловмисник може генерувати тисячі запитів на публічні ендпоінти (напр., /api/summary/), що призведе до 100% завантаження CPU та пам'яті сервера FastAPI (наприклад, на платформі Render) і повної відмови системи для легітимних користувачів.

Для захисту сервера на рівні FastAPI було інтегровано middleware-обмежувач slowapi. Цей механізм спрацьовує на самому початку обробки запиту, до того, як він потрапить до ресурсоємної бізнес-логіки. На відміну від внутрішнього лімітера шлюзу, slowapi налаштований на ідентифікацію клієнта за його IP-адресою (key_func=get_remote_address). Для всіх ключових API-ендпоінтів було встановлено консервативний ліміт (30 запитів на хвилину), що унеможливлює DoS-атаки грубого перебору (див. лістинг 3.9-3.10) [36].

Лістинг 3.9 – Ініціалізація slowapi у main.py

```
from fastapi import FastAPI, Request
from slowapi import Limiter, _rate_limit_exceeded_handler
from slowapi.util import get_remote_address
from slowapi.errors import RateLimitExceeded
limiter = Limiter(key_func=get_remote_address)
app = FastAPI()
app.state.limiter = limiter
app.add_exception_handler(RateLimitExceeded,_rate_limit_exceed
ed_handler)
```

Лістинг 3.10 – Застосування декоратора до ендпоінту timeline.py

```
@router.get("/{address}")
@limiter.limit("30/minute")
async def get_timeline(…)
```

Як показано на рисунку 3.2, при спробі перевищити встановлений ліміт (понад 30 запитів на хвилину) за допомогою автоматизованого скрипта у терміналі PowerShell, сервер коректно припиняє обробляти запити. Після 32-го успішного запиту (ліміт у «ковзному вікні»), всі наступні спроби миттєво отримують помилку HTTP 429 Too Many Requests.

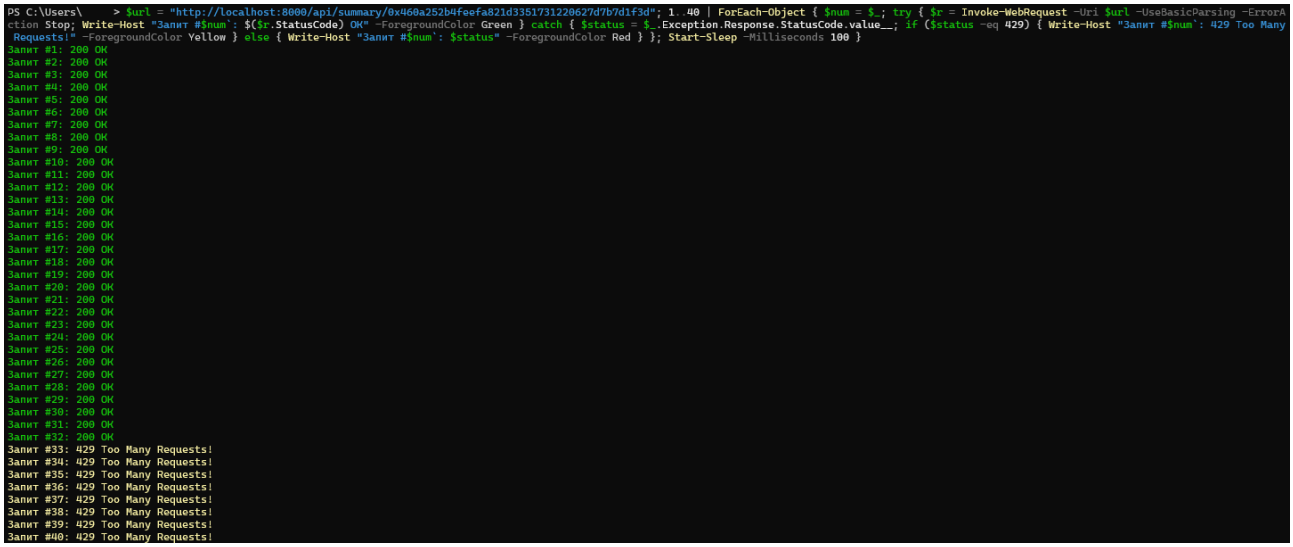

Рисунок 3.2 – Результат стрес-тестування slowapi у терміналі PowerShell

Захист від ін’єкційних атак (Injection). Загроза ін'єкційних атак виникає, коли неконтрольовані дані від користувача використовуються для побудови «сирих» команд, що виконуються на сервері [37]. У даному проєкті ця загроза була нейтралізована на двох взаємодоповнюючих рівнях: архітектурному та на рівні фреймворку, що забезпечує високу стійкість системи.

На архітектурному рівні загроза класичної SQL-ін'єкції усувається завдяки тому, що система не взаємодіє безпосередньо з реляційною базою даних (SQL) для обробки запитів користувача. Вхідні дані (адреса) передаються до DataGatewayService, який, у свою чергу, формує JSON-RPC запити до зовнішніх API. Ці зовнішні API є ізольованими «чорними скриньками», які не вразливі до класичних SQL-ін'єкцій з боку нашого кінцевого користувача. Це архітектурне рішення ізолює систему від цього вектора атаки.

Цей архітектурний захист посилюється суворою вхідною валідацією на рівні фреймворку. Весь вхідний трафік у FastAPI примусово проходить через механізм валідації типів даних Pydantic. Цей принцип застосовується до кожного параметра, який приймає Backend API:

Поле для вводу адреси. Система очікує, що параметр {address} буде рядком, який відповідає чіткому формату EVM-адреси (регулярному виразу ^0x[a-fA-F0-9]{40}$). Якщо зловмисник спробує передати будь-який інший,

потенційно шкідливий, рядок (наприклад, фрагмент SQL), він буде відхилений Pydantic до того, як потрапить до будь-якої бізнес-логіки (див. рисунок 3.3).

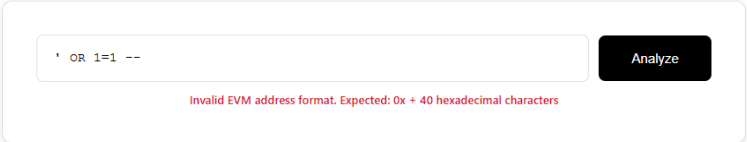

Рисунок 3.3 – Нейтралізація ін'єкційної атаки за допомогою валідації адреси

Принцип суворої вхідної валідації поширюється на всі додаткові поля фільтрації у системі, зокрема у блоках Transaction History та Transaction Network Graph. Ці поля, незважаючи на їх допоміжну функцію, можуть створювати нові вектори для атак (наприклад, ін'єкції JSON-параметрів або відмови в обслуговуванні).

Параметри адрес. Поля фільтрації, призначені для введення адреси, валідуються за тим самим регулярним виразом EVM-адреси (^0x[a-fA-F0-9]{40}$). Це запобігає спробі впровадження шкідливого SQL-коду, оскільки він не відповідає цьому формату.

Параметри фільтрації (Числа). Поля, призначені для лімітування вибірки або фільтрації за сумою, мають бути строго визначені як цілі числа (int) або числа з плаваючою комою (float) (див. рисунок 3.4).

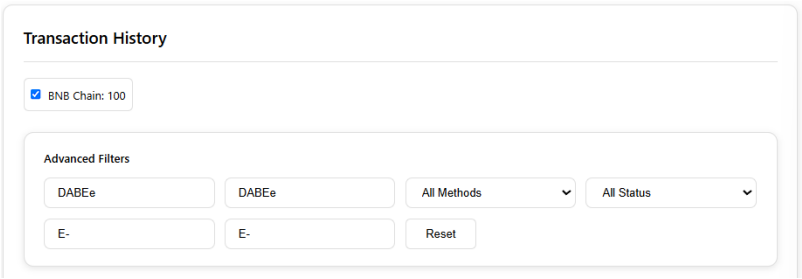

Рисунок 3.4 – Спроба впровадження шкідливого SQL-фрагмента

Захист від атак на браузер (XSS, Clickjacking, MIME-Sniffing). Ця група атак спрямована на кінцевого користувача шляхом маніпуляції довірою до веб-додатку. Її мета – змусити браузер жертви виконати шкідливий JavaScript-код у контексті довіреного домену, що може призвести до крадіжки сесійних токенів

або несанкціонованих дій. Для нейтралізації загроз було реалізовано двокомпонентний захист: на рівні Backend (інструкції для браузера) та на рівні Frontend (обробка даних).

Найбільш критичною є загроза Stored XSS. Вона виникає, коли шкідливий HTML/JavaScript-код (наприклад, <img src=x onerror=alert(1)>) закладається зловмисником у публічні дані блокчейну (наприклад, у назву токена), індексується зовнішніми API і потім некоректно відображається у браузері користувача. Це є найбільш небезпечним вектором, оскільки атака автоматично вражає кожного, хто перегляне скомпрометований токен. Нейтралізація цієї загрози реалізується через багаторівневу стратегію, що зосереджена на запобіганні виконанню шкідливого коду браузером.

Впровадження захисних HTTP-заголовків (Backend Бар'єр). Перший рівень захисту від атак, спрямованих на кінцевого користувача, реалізовано на рівні Backend API шляхом впровадження Security Headers Middleware. Цей механізм додає до кожної HTTP-відповіді набір інструкцій, які керують поведінкою браузера, запобігаючи використанню «старих» вразливостей, які браузери можуть ігнорувати за замовчуванням.

Зокрема, цей middleware нейтралізує кілька ключових загроз:

– X-XSS-Protection: 1; mode=block. Вмикає вбудований XSS-фільтр браузера, який блокує рендеринг всієї сторінки, якщо виявляє ознаки міжсайтового скриптингу.

– X-Frame-Options: DENY. Забезпечує захист від Clickjacking, забороняючи завантаження вмісту сайту у фреймах (<iframe>) на сторонніх доменах.

– X-Content-Type-Options: nosniff. Захищає від MIME-Sniffing, забороняючи браузеру «вгадувати» тип контенту та примушуючи його довіряти оголошеному серверу Content-Type [38].

– Strict-Transport-Security (HSTS). Заголовок додається лише в Production-режимі і примусово вимагає використання HTTPS, посилюючи захист каналу зв'язку (див. лістинг 3.11).

Лістинг 3.11 – Реалізація Middleware

```
@app.middleware("http")
async def add_security_headers(request: Request, call_next):
response = await call_next(request)
response.headers["X-Content-Type-Options"] = "nosniff"
response.headers["X-Frame-Options"] = "DENY"
response.headers["X-XSS-Protection"] = "1; mode=block"
response.headers["Referrer-Policy"] = "strict-origin-when-cross-origin"
if not settings.DEBUG:
response.headers["Strict-Transport-Security"] = "max-age=31536000; includeSubDomains"
return response
```

На рисунку 3.5 подано скріншот заголовків відповіді API, що підтверджує коректність встановлених інструкцій безпеки.

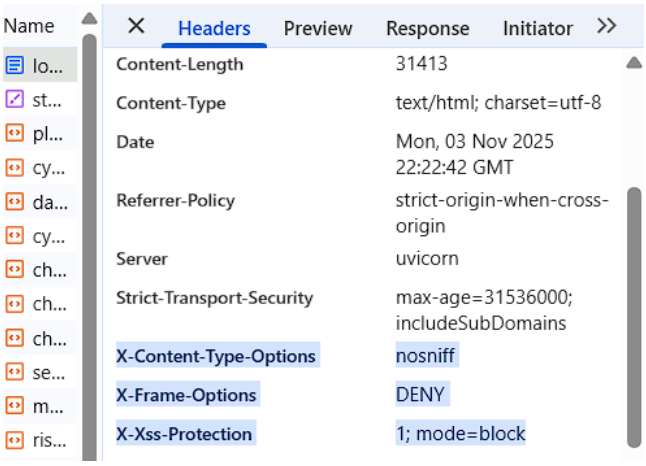

Рисунок 3.5 – Підтвердження захисту: скріншот заголовків відповіді API

Суворе екранування виводу (Frontend Бар'єр). Це єдиний надійний захист від Stored XSS, оскільки він усуває корінну причину – виконання несанкціонованого HTML-коду. Замість того, щоб покла-датися на вразливі функції (element.innerHTML), які інтерпретують рядок як виконуваний HTML, було впроваджено централізований механізм HTML-екранування (Escaping).

Для цього створено модуль security.js, який містить функцію escapeHTML() для безпечного перетворення даних. Ця функція цілеспрямовано

замінює керуючі символи HTML на їх безпечні текстові еквіваленти (див. таблиця 3.1) [39].

Таблиця 3.1 – Механізм екранування символів для захисту від XSS-атак

Символ	Безпечний еквівалент	Призначення
<	<	Початок HTML-тегу
>	>	Кінець HTML-тегу
&	&	Екранування символу амперсанда
‘	"	Екранування одинарних лапок
“	'	Екранування подвійних лапок

Кожен модуль Frontend, що працює з динамічними даними, використовує цю функцію. Наприклад, у portfolio.js функція екранування гарантує, що назви токенів (token.name та token.symbol) завжди відображаються як безпечний текст. Ефективність цього механізму підтверджується демонстраційним тестом: замість того, щоб впроваджувати цей код безпосередньо у блокчейн, було використано контрольовану ін'єкцію на рівні Backend API. Сервіс, відповідальний за формування даних для блоку «Portfolio» (portfolio_service.py), був тимчасово модифікований, щоб повертати шкідливий рядок <img src=x onerror=alert('XSS_ATTACK_TEST')> у полі token.name замість реальних даних.

З рисунка 3.6 видно, що браузер не виконує код, функція екранування (escapeHTML()) перетворює керуючі символи < та > на безпечні мнемоніки (< та >), що змушує браузер відображати весь рядок як нешкідливий текст.

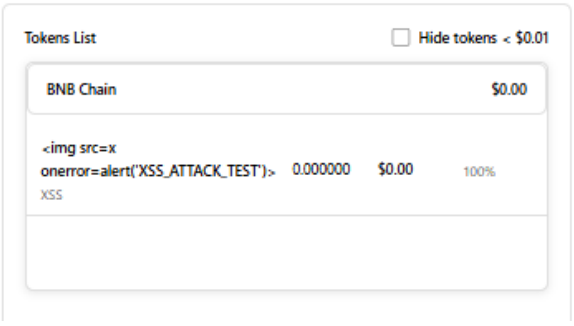

Рисунок 3.6 – Демонстрація ефективності HTML-екранування

Захист від міждоменних атак та розкриття інформації. Ця категорія загроз спрямована на використання помилок у конфігурації сервера та маніпуляції механізмами, що контролюють доступ до API. Захист тут реалізується шляхом впровадження суворих політик на рівні Middleware (програмного забезпечення, що перехоплює HTTP-запити до їх надходження до основної логіки). Загрози Cross-Site Request Forgery (CSRF) та загальні CORS Misconfiguration нейтралізуються шляхом застосування принципу найменших привілеїв до дозволених HTTP-методів та заголовків [40].

Неправильно налаштований CORS дозволяє шкідливому сайту робити будь-які запити до API, використовуючи автентифікаційні дані користувача, що може призвести до CSRF-атаки. Замість того, щоб дозволяти всі методи та заголовки (allow_methods=["*"], allow_headers=["*"]), списки було звужено лише до абсолютно необхідних для роботи Frontend: allow_methods обмежено тільки на GET (отримання даних) та POST (надсилання даних), а allow_headers обмежено лише на Content-Type та Authorization (див. лістинг 3.12) [41].

Лістинг 3.12 – Конфігурація CORS у FastAPI

```
from fastapi.middleware.cors import CORSMiddleware
app.add_middleware(
CORSMiddleware,
allow_origins=["http://localhost:8000"],
allow_credentials=True,
allow_methods=["GET", "POST"],
allow_headers=["Content-Type", "Authorization"],
)
```

Верифікація цієї політики проводилася шляхом імітації міждоменного попереднього запиту, який браузер автоматично надсилає для перевірки дозволів. Для цього було використано інструмент cURL у терміналі PowerShell, щоб надіслати OPTIONS запити з невідомого домену (Origin: http://evil-site.com).

Тест на заборонений метод. Було надіслано запит на дозвіл використання несанкціонованого методу (Access-Control-Request-Method: PUT).

Тест на заборонений заголовок. Було надіслано запит на дозвіл використання несанкціонованого заголовка (Access-Control-Request-Headers: X-My-Evil-Header).

Як показано на рисунку 3.7, в обох випадках сервер коректно ідентифікував порушення політики. Він не включив PUT та X-My-Evil-Header до списку дозволених (access-control-allow-methods: GET, POST) і негайно відхилив обидва запити зі статусом HTTP 400 Bad Request та відповідними повідомленнями (Disallowed CORS origin, method та Disallowed CORS origin, headers). Це підтверджує, що політика найменших привілеїв успішно впроваджена.

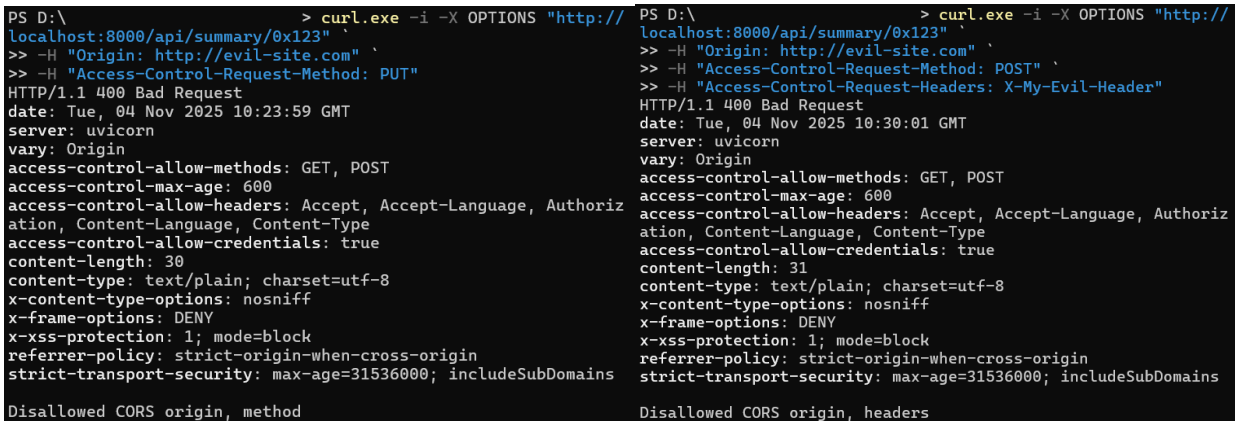

Рисунок 3.7 – Верифікація політики CORS: відхилення несанкціонованих методів та заголовків

Обмеження розміру запиту (Request Size Limiting). Цей механізм є важливим доповненням до захисту від DoS-атак. Загроза виникає, коли зловмисник надсилає надзвичайно великий JSON-об'єкт, щоб вичерпати всю оперативну пам'ять сервера. Щоб запобігти цьому було впроваджено middleware limit_request_size, який перехоплює заголовок Content-Length запиту ще до початку завантаження тіла. Якщо розмір запиту перевищує встановлений ліміт (у даному випадку 1 мегабайт), запит негайно відхиляється з помилкою HTTP 413 Payload Too Large (див. лістинг 3.13) [42].

Лістинг 3.13 – Реалізація Request Size Limiting Middleware

```
@app.middleware("http")
async def limit_request_size(request: Request, call_next):
# Перехоплюємо заголовок Content-Length
if request.headers.get("content-length"):
size = int(request.headers["content-length"])
     if size > 1024 * 1024:  # 1MB limit
from fastapi import HTTPException
raise HTTPException(status_code=413, detail="Request too large")
return await call_next(request)
```

Перевірка проводиться шляхом надсилання POST-запиту з тілом, розмір якого перевищує 1 МБ (див. рисунок 3.8). Сервер, не виконавши жодної бізнес-логіки, коректно повертає статус HTTP 413 Payload Too Large, що підтверджує захист від DoS-атак, що використовують надмірне споживання пам'яті.

Рисунок 3.8 – Демонстрація захисту від DoS: Відхилення запиту з помилкою HTTP 413 Payload Too Large при перевищенні встановленого ліміту (1 МБ).

Таким чином, реалізовані заходи створюють комплексну, багаторівневу стратегію захисту, яка ефективно нейтралізує всі ідентифіковані типові веб-атаки. Такий підхід гарантує, що навіть у разі збою одного механізму, інші бар'єри безпеки залишаються на місці, забезпечуючи високий рівень стійкості та надійності для всього додатка.

РОЗДІЛ 4. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

Оскільки весь життєвий цикл проекту «Розробки інструменту для виявлення шахрайства в блокчейні» передбачає безперервну взаємодію з комп'ютерними системами, підтримка здоров'я, як розробника, так і кінцевих користувачів стає ключовим пріоритетом. Тривалий час роботи перед екраном, повторювані завдання та малорухливий характер, як розробки, так і використ-тання програмного забезпечення можуть призвести до фізичного перенапруження, втоми очей та інших проблем зі здоров'ям, якщо їх належним чином не контролювати. Тому дотримання стандартів охорони праці та протипожежної безпеки є необхідним, аби гарантувати безпечні умови та високу продуктивність.

Забезпечення безпечних умов праці в Україні ґрунтується на національних законодавчих та нормативних актах, що встановлюють єдиний підхід до організації охорони праці. Базові положення закріплені в Конституції України, та деталізовані в Кодексі законів про працю (КЗпП) [43] та Законі України «Про охорону праці» [44]. Ці документи встановлюють юридичну відповідальність роботодавця за створення безпечних і нешкідливих умов праці, проведення інструктажів, забезпечення засобами захисту, організацію медичних оглядів та впровадження профілактичних заходів. Працівники, у свою чергу, мають гарантоване державою право на безпечне робоче середовище незалежно від того, чи виконується робота на підприємстві, у віддаленому офісі чи дистанційно.

Особлива увага приділяється роботі з екранними пристроями, оскільки вся робота передбачає тривале перебування за комп’ютером що створює навантаження на зір, опорно-руховий апарат та нервову систему. Основні вимоги визначені НПАОП 0.00-7.15-18 [45] та ДСанПіН 3.3.2-007-98 [46]. Ці нормативи регламентують:

– Роботодавець повинен забезпечити навчання і перевірку знань працівників з питань охорони праці та безпечного використання екранних пристроїв до початку роботи з ними, а також у випадках модифікації та організації роботи обладнання.

– Для розробників програм із застосуванням ЕОМ, слід призначати регламентовану перерву для відпочинку тривалістю 15 хвилин через кожну годину роботи за ВДТ.

– У всіх випадках, коли виробничі обставини не дозволяють застосувати регламентовані перерви, тривалість безперервної роботи з ВДТ не повинна перевищувати 4 години.

– Яскравість світильників загального освітлення в зоні кутів випромінювання від 50 до 90 град. з вертикаллю в повздовжній та поперечній площинах має становити не більше ніж 200 кд/м2, захисний кут світильників – не менше ніж 40 град.

– Символи на екранних пристроях мають бути чіткими, відповідного розміру. Між символами і рядками символів має бути належна відстань.

– Зображення на екрані має бути стабільним, без миготінь або інших видів нестабільності.

– Яскравість та/або контрастність символів має легко регулюватися працівником під час роботи з екранними пристроями, а також швидко адаптуватися до навколишніх умов.

– Вибираючи екрани, слід надавати перевагу таким, які легко та вільно повертаються і нахиляються відповідно до потреби працівника.

– Екран не має відблискувати або відбивати світло, щоб не викликати дискомфорту у працівника під час роботи з екранними пристроями.

– Вибираючи клавіатуру, слід надавати перевагу такій, яка відкидається і є автономною (відокремленою від екрана), щоб працівник міг вибрати зручну робочу позу й уникнути втоми рук (кисті і верхньої частини руки).

– Поверхня клавіатури має бути матовою, щоб уникнути віддзеркалювання. Розташування клавіш і самі клавіші мають полегшувати

роботу із клавіатурою. Позначення клавіш повинно бути достатньо контрастним і розбірливим.

– Розташування пристрою введення – виведення інформації має забезпечувати добру видимість екрана ВДТ, зручність ручного керування в зоні досяжності моторного поля і за висотою – 900...1300 мм, за шириною 400...500 мм.

– Для забезпечення допустимих рівнів шуму на робочих місцях слід застосовувати засоби звукопоглинання, вибір яких має обгрунтовуватись спеціальними інженерно-акустичними розрахунками.

– Робоче крісло має бути стійким і дозволяти працівнику з екранними пристроями легко рухатися та займати зручне положення.

Організація робочого місця має відповідати сучасним ергономічним стандартам, зокрема у ДСТУ 8604:2015 [47], що передбачає правильне розташування монітора, забезпечення оптимальної робочої пози та застосування крісел із регульованими елементами. Санітарні норми встановлюють необхідну відстань від очей до екрана, правила розміщення джерел світла для запобігання відблискам, а також вимоги до параметрів мікроклімату – температури, вологості й вентиляції. Дотримання цих норм повинно мінімізувати ризики перевтоми, порушення постави та зорового напруження.

Не менш важливою складовою безпечних умов праці є дотримання правил протипожежної безпеки. Приміщення, у яких розміщується комп’ютерна техніка, повинні відповідати вимогам ДБН В.2.5-56:2014 [48], що регламентують систему протипожежного захисту, включаючи встановлення автоматичних пожежних сигналізацій, вибір типів датчиків та порядок їх технічного обслуговування. Робочі зони повинні бути забезпечені первинними засобами пожежогасіння – переважно вогнегасниками, придатними для ліквідації загорянь електрообладнання. Їх технічний стан контролюється відповідно до ДСТУ 4297:2004 [49], що передбачає періодичні огляди, перевірку працездатності та правила експлуатації.

Вимоги електробезпеки регулюються ДНАОП 0.00-1.21-98 [50], метою є усунення ризику ураження електричним струмом та запобігання аваріям у

мережах, що живлять комп’ютерне обладнання. Норми передбачають використання трипровідних електромереж з окремим захисним провідником, правильну організацію електричного заземлення, а також заборону підключення комп’ютерної техніки через адаптери або подовжувачі, що не забезпечують повноцінний захист. Особлива увага приділяється вимогам до аварійного відключення електроживлення у приміщеннях з великою кількістю робочих місць, що мінімізує ризики у разі короткого замикання чи перегріву обладнання.

Запропоновані рішення забезпечують повну відповідність вимогам охорони праці: робоче місце облаштоване за ергономічними стандартами, екран і клавіатура налаштовані для мінімізації навантаження на очі та руки, передбачені регламентовані перерви, контроль освітлення, шуму та мікроклімату. Дотримані також вимоги електробезпеки та протипожежного захисту, включно із заземленням, аварійним відключенням і наявністю первинних засобів гасіння. Це гарантує безпечну і комфортну роботу як при розробці, так і при використанні інструменту для виявлення шахрайства в блокчейні.

4.2 Захист людини від іонізуючого випромінювання

Іонізуюче випромінювання – це форма енергії, здатна утворювати заряджені атоми й молекули у середовищі, з яким взаємодіє. Воно має як природні джерела (космічні промені, природні радіонукліди Землі), так і штучні: ядерні реактори, рентгенівські установки, прискорювачі частинок і радіоактивні ізотопи. До іонізуючих випромінювань належать корпускулярні (альфа-, бета-випромінювання та нейтрони) й електромагнітні – гама- та рентгенівські промені.

Їхня ключова особливість полягає у високій енергії, здатній порушувати структуру біологічних клітин і навіть зумовлювати їх загибель. На ці випромінювання не реагують органи чуття людини, що робить їх особливо небезпечними. Водночас іонізуюче випромінювання має широке практичне застосування – від медицини та промисловості до сільського господарства й атомної енергетики.

Основними документами, якими регламентується радіаційна безпека в Україні є ДНАОП 00.3-3.24-97, «Норми радіаційної безпеки України» НРБУ-97/Д-2000 та в «Основні санітарні норми України» ОСПУ-2000. За допустимими основними дозовими границями, встановлюються такі категорії осіб, які опромінюються:

– Категорія А – персонал, що має безпосередній зв'язок з джерелами іонізуючого випромінювання. Загальна доза опромінення на рік –5 бер (50 мЗв).

– Категорія Б – персонал, що безпосередньо не працює із радіоактивними речовинами, але за умови розміщення на робочих або місцях проживання може потрапити під дію опромінення. Гранична доза опромінення – 0,5 бер/рік.

– Категорія В – решта населення держави. Доза не нормується, але не повинна перевищувати природній фон – від 40 до 200 мбер/рік.

Для осіб категорій А і Б НРБУ-97 встановлюють ліміти ефективної й еквівалентної доз за календарний рік. Обмеження опромінення категорії В здійснюється введенням лімітів річної ефективної та еквівалентної доз для критичних груп осіб категорії Б. Остання означає, що значення річної дози опромінення осіб, що входять до критичної групи, не повинно перевищувати ліміту дози, встановленого для категорії В (див. таблиця. 4.1).

Таблиця 4.1 – Ліміти доз сумарного опромінення

Ліміти доз, мЗв·рік-1	Категорія опромінюваних осіб		
	А	Б	В
$ЛД_E$ (ліміт ефективної дози)	20	2	1
Ліміти еквівалентної дози			
$ЛД_{lens}$ (для кришталика ока)	150	15	15
$ЛД_{Skin}$ (для шкіри)	500	50	50
$ЛД_{exstrim}$ (для кистей та стіп)	500	50	-

Чисельні значення наведених в таблиці 4.1 основних дозових лімітів НРБУ-97 встановлюють на рівнях, що виключають можливість виникнення детерміністичних ефектів опромінення і одночасно гарантують настільки

низьку ймовірність виникнення стохастичних ефектів опромінення, що вона є прийнятною як для окремих осіб, так і для суспільства в цілому.

Крім лімітів ефективної й еквівалентної річних доз, НРБУ-97 встановлено допустимі рівні надходження радіонуклідів в організм людини за календарний рік, потужності еквівалентної дози, концентрації радіонуклідів у повітрі, питній воді та раціоні, щільності потоку частинок, забруднення шкіри, спецодягу, робочих поверхонь тощо. Значення окремого допустимого рівня розраховується за умови, що створена ним річна доза не повинна перевищувати ліміту відповідної дози. При багатократному радіаційному опроміненні допустимі рівні визначаються за умови, щоб річна сумарна доза від усіх джерел випромінювання не перевищувала відповідного ліміту дози [51].

Для захисту від зовнішнього опромінювання, яке має місце при роботі із закритими джерелами випромінювання (такими, які виключають можливість потрапляння радіоактивних речовин в навколишнє середовище), основні зусилля необхідно направити на попередження переопромінення персоналу шляхом:

- Збільшення відстані між джерелом випромінювання і людиною.
- Скорочення тривалості роботи в зоні випромінювання.
- Екранування джерела випромінювання.

Під внутрішнім опроміненням розуміють вплив на організм людини випромінювань радіоактивних речовин, що потрапляють всередину організму. На дверях приміщень, у яких проводиться робота з відкритими джерелами радіоактивного випромінювання, повинен знаходитися знак радіаційної небезпеки.

Радіоактивні речовини повинні знаходитися в спеціальних приміщеннях. По кожному з них необхідно вести суворий облік надходжень і витрат, щоб виключити можливість їх безконтрольного використання. Порядок транспортування радіоактивних речовин регламентується спеціальними правилами. Радіоактивні речовини перевозять у спеціальних контейнерах і спеціально обладнаним транспортом. До організацій і установ, у яких постійно виконуються роботи з радіоактивними речовинами, підвищені вимоги з охорони праці. Керівництво цих організацій зобов'язане розробити детальні інструкції, в яких викладено порядок проведення робіт, облік збереження та

використання джерел випромінювання, збір та знешкодження відходів, порядок проведення дозиметричного контролю [52].

Особливе значення при роботі з відкритими джерелами радіоактивного випромінювання має особиста гігієна та засоби індивідуального захисту працюючого. В залежності від виду виконуваних робіт і небезпечності цих робіт застосовують спецодяг (комбінезони чи костюми), спецбілизну, шкарпетки, спецвзуття, рукавиці, респіратори. У випадках можливого забруднення повітря радіоактивним пилом, газами чи парою – додатково використовувати респіратори (ШБ-1 Пелюстка). Також використовуються ще спеціальні пневматичні костюми з пластичних матеріалів (ЛГ-4), а також гумові чоботи. При використанні ЗІЗ потрібно обов'язково звертати увагу на послідовність їх вдягання і знімання. Після роботи з радіоактивними речовинами необхідно добре вимити руки і обличчя, а також перевірити їх чистоту дозиметричними приладами.

Безпеку роботи з радіоактивними речовинами (РАР) і джерелами випромінювання можна забезпечити тоді, коли є організований систематичний контроль за рівнем зовнішнього і внутрішнього опромінення персоналу, а також за рівнем радіації навколишнього середовища. Всі, хто працює з РАР повинні бути забезпечені індивідуальними дозиметрами для контролю дози гама-випромінювань, яку одержує кожен працівник окремо. Для виявлення і кількісного вимірювання радіоактивного випромінювання використовують методи:

– Іонізаційний – вимірювання рівня іонізації випромінювання.

– Сцинтиляційний – вимірювання інтенсивності світлових спалахів, які виникають у речовинах, що люмінесціюють при проходженні крізь них іонізуючих випромінювань.

– Фотографічний – ґрунтується на здатності фото емульсійного шару під дією РАР темніти після проявлення).

– Хімічний – здатність деяких розчинів змінювати свій колір під дією іонізуючих випромінювань [53].

Результати усіх видів радіаційного контролю повинні зберігатися протягом 30-ти років. При індивідуальному контролі ведуть облік річної дози опромінення, а також сумарної дози за весь період професійної діяльності.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи освітнього рівня «Магістр» було здійснено комплексне дослідження ончейн-інформації в EVM-сумісних блокчейнах з метою розроблення системи виявлення шахрайської активності. Проведений аналіз дозволив обґрунтувати методологію оцінювання ризиковості адрес і їхніх поведінкових характеристик, а дослідження поширених схем шахрайства у Web3 дозволило виокремити специфічні ознаки, що можуть слугувати індикаторами злочинної активності.

На основі цього сформовано концепцію та архітектуру системи, функціонування якої охоплює модулі збору даних, побудови транзакційних профілів, аналізу активів, візуалізації активності, а також формування кінцевого висновку щодо рівня ризику. Реалізована система скорингу забезпечує можливість інтерпретації поведінки адреси та надає кількісну оцінку ризиковості.

Спроектована архітектура дала змогу інтегрувати всі функціональні компоненти в єдине рішення, здатне обробляти значні обсяги даних, а проведене тестування підтвердило коректність реалізованих механізмів. Значну увагу приділено аспектам інформаційної безпеки: захисту ключів доступу, забезпеченню цілісності даних, протидії типовим веб-загрозам та мінімізації ризиків несанкціонованого доступу до системи.

Підсумовуючи, можна зробити висновок, що поставлену мету досягнуто: сформовано науково й практично обґрунтовану концепцію системи аналізу ончейн-даних, створено архітектуру програмного забезпечення та реалізовано ключові модулі, необхідні для виявлення ризикових адрес у блокчейні. Розроблений підхід може слугувати основою для побудови аналітичних платформ та розслідування ончейн-інцидентів.

Подальші перспективи в цьому напрямку можуть зосереджуватись на кількох ключових напрямках. Наприклад, розширення аналітики в різних блокчейнах: об’єднання даних з різних блокчейнів, дослідження взаємодій через кросчейн мости та їх візуалізація на графах; впровадження модуля для

аналізу смарт-контрактів, що дозволить не лише оцінювати поведінку адреси, а й виявляти потенційно небезпечні функції та логіку всередині контрактів. У сфері безпеки програмного забезпечення варто продовжувати вдосконалювати захист взаємодії з API, оптимізувати процеси зберігання ключів та впроваджувати системи моніторингу аномалій [54].

Таким чином, проведена робота створює міцну основу для подальших досліджень і практичних розробок у сфері аналізу ончейн-даних та виявлення шахрайської активності. Розроблена система має потенціал перетворитися на комплексний аналітичний інструмент для Web3-екосистеми, здатний підвищувати рівень безпеки, довіри та прозорості.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 2025 Crypto Crime Mid-year Update: Stolen Funds Surge as DPRK Sets New Records. Chainalysis. URL: https://www.chainalysis.com/blog/2025-crypto-crime-mid-year-update/ (дата звернення: 06.10.2025).

2. What Are The Different Types of Blockchain Technology? Blockchains. URL: https://101blockchains.com/types-of-blockchain/ (дата звернення: 17.10.2025).

3. Karpinski, M., Kovalchuk, L., Kochan, R., Oliynykov, R., Rodinko, M., & Wiecław, L. (2021). Blockchain technologies : probability of double-spend attack on a proof-of-stake consensus. Sensors, 21(19), 1-14. doi: 10.3390/s21196408. (дата звернення: 17.10.2025).

4. What are Blockchain Layers? SoluLab. URL: https://www.solulab.com/what-are-blockchain-layers/ (дата звернення: 01.10.2025).

5. Bitcoin address. BitcoinWiki. URL: https://bitcoinwiki.org/wiki/bitcoin-address (дата звернення: 17.10.2025).

6. What is a Bitcoin unspent transaction output (UTXO)?. Payward, Inc. URL: https://www.kraken.com/uk/learn/what-is-bitcoin-unspent-transaction-output-utxo (дата звернення: 17.10.2025).

7. UTXO vs. Account-Based Chains. Glassnode. URL: https://docs.glassnode.com/guides-and-tutorials/on-chain-concepts/utxo-vs.-account-based-chains (дата звернення: 17.10.2025).

8. Addresses: Substrate & EVM | Unique docs. Unique docs. URL: https://docs.unique.network/build/tech-concepts/addresses/ (дата звернення: 17.10.2025).

9. Understanding EVM Deterministic Addresses. Medium. URL: https://blog.blockmagnates.com/understanding-evm-deterministic-addresses-f5c886a3ad23 (дата звернення: 17.10.2025).

10. Solana Account Model. Solana Foundation. URL: https://solana.com/uk/docs/core/accounts (дата звернення: 17.10.2025).

11. Rug Pull Scams. DataVisor. URL: https://www.datavisor.com/wiki/rug-pull-scams (дата звернення: 19.10.2025).

12. Unveiling Rug Pull Schemes in Crypto Token via Code-and-Transaction Fusion Analysis. arXiv. URL: https://arxiv.org/html/2506.18398v1 (дата звернення: 19.10.2025).

13. A crypto influencer is under a sweeper bot attack. How can Beosin help recover his funds?. Beosin. URL: https://beosin.com/resources/a-crypto-influencer-is-under-a-sweeper-bot-attack-how-can-b (дата звернення: 19.10.2025).

14. Token Approvals and Wallet Drainers: How to Keep Your Assets Safe. Trust Wallet. URL: https://trustwallet.com/uk/blog/security/token-approvals-and-wallet-drainers-how-to-keep-your-assets-safe (дата звернення: 19.10.2025).

15. Anatomy of an Address Poisoning Scam. Chainalysis. URL: https://www.chainalysis.com/blog/address-poisoning-scam/ (дата звернення: 19.10.2025).

16. Money Laundering in Crypto: How Criminals Hide Their Tracks. Merkle Science. URL: https://www.merklescience.com/blog/money-laundering-in-crypto-how-criminals-hide-their-tracks (дата звернення: 19.10.2025).

17. Crypto Wash Trading: What It Is and How to Avoid It. dYdX International Ltd. URL: https://www.dydx.xyz/crypto-learning/wash-trading-crypto (дата звернення: 19.10.2025).

18. What is a pump and dump in crypto?. Coinbase. URL: https://www.coinbase.com/ru/learn/crypto-glossary/what-is-a-pump-and-dump-in-crypto (дата звернення: 19.10.2025).

19. Crypto Ponzi schemes – inside the insidious investment traps. TechForing Ltd. URL: https://techforing.com/resources/articles/crypto-ponzi-scheme (дата звернення: 19.10.2025).

20. Dusting attacks & airdrop scam tokens. Trezor company. URL: https://trezor.io/support/troubleshooting/coins-tokens/dusting-attacks-airdrop-scam-tokens?srsltid=AfmBOoqnlsq3qSxncc-Z09B9dnxAOQ33doq-wHXWsM0_e37FrxT81VD3 (дата звернення: 19.10.2025).

21. Airdrop Scams in Crypto and How to Avoid Them. OneKey. URL: https://onekey.so/blog/ru/ecosystem/airdrop-scams-in-crypto-and-how-to-avoid-them/ (дата звернення: 19.10.2025).

22. Anonymity Services' Usage of Cryptocurrency and Role in Cybercrime. Chainalysis. URL: https://www.chainalysis.com/blog/anonymity-services-cryptocurrency/ (дата звернення: 19.10.2025).

23. Kotsiuba, I., Velykzhanin, A., Biloborodov, O., Skarga-Bandurova, I., Biloborodova, T., Yanovich, Y., & Zhygulin, V. (2018, December). Blockchain evolution: from bitcoin to forensic in smart grids. In 2018 IEEE international conference on big data (big data) (pp. 3100-3106). IEEE. (дата звернення: 19.10.2025).

24. Shevchuk, R., Lishchynskyy, I., Ciura, M., Lyzun, M., Kozak, R., & Kasianchuk, M. (2025). Application of Blockchain Technology in Emergency Management Systems: A Bibliometric Analysis. Applied Sciences, 15(10), 5405. (дата звернення: 19.10.2025).

25. Tymoshchuk D., Yatskiv V., Tymoshchuk V., Yatskiv N. Interactive cybersecurity training system based on simulation environments. Measuring and computing devices in technological processes, 2024 (4). P. 215-220. https://doi.org/10.31891/2219-9365-2024-80-26 (дата звернення: 13.11.2025).

26. Thakur, A. Microservices Patterns : API Gateway Pattern. Medium. URL: https://medium.com/@abhi.strike/microservices-patterns-api-gateway-pattern-5c4567370a54 (дата звернення: 16.11.2025).

27. Informatica Documentation. IAM role configuration for AWS Secrets Manager. URL: https://docs.informatica.com/cloud-common-services/administrator/current-version/organization-administration/general-and-security-settings/secrets-manager-configuration/aws-secrets-manager-configuration/iam-role-configuration-for-aws-secrets-manager.html (дата звернення: 16.11.2025).

28. El-Bably, M. Rate Limiting: The Sliding Window Algorithm. Medium. URL: https://medium.com/@m-elbably/rate-limiting-the-sliding-window-algorithm-daa1d91e6196 (дата звернення: 16.11.2025).

29. RPC Service Plans. Ankr. URL: https://www.ankr.com/docs/rpc-service/service-plans/ (дата звернення: 16.11.2025).

30. Fowler, M. CircuitBreaker. URL: https://martinfowler.com/bliki/Circuit-Breaker.html (дата звернення: 16.11.2025).

31. GitGuardian Blog. Git Secrets: Why Secrets inside Git are such a Problem URL: https://blog.gitguardian.com/secrets-credentials-api-git/ (дата звернення: 19.11.2025).

32. Render. Default Environment Variables. URL: https://render.com/docs/environment-variables (дата звернення: 19.11.2025).

33. Amazon AWS Documentation. Identity-based policies - AWS Secrets Manager. URL: https://docs.aws.amazon.com/secretsmanager/latest/userguide/auth-and-access_iam-policies.html (дата звернення: 19.11.2025).

34. Lazy loading. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/Performance/Guides/Lazy_loading (дата звернення: 19.11.2025).

35. Latif, S. Implementing a Custom Cache in Python. Medium. URL:https://medium.com/becoming-a-better-software-developer/implementing-a-custom-cache-in-python-68c39ece8a8 (дата звернення: 19.11.2025).

36. Slowapi Documentation. URL: https://slowapi.readthedocs.io/en/latest/ (дата звернення: 22.11.2025).

37. OWASP. SQL Injection. URL: https://owasp.org/www-community/attacks/SQL_Injection (дата звернення: 22.11.2025).

38. Addressing security vulnerabilities by HTTP Security Headers. Fortinet Document Library. URL: https://help.fortinet.com/fweb/581/Content/FortiWeb/

fortiweb-admin/http_header_ security.htm (дата звернення: 22.11.2025).

39. OWASP. Cross Site Scripting Prevention – OWASP Cheat Sheet. URL: Serieshttps://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html (дата звернення: 22.11.2025).

40. OWASP. Cross Site Request Forgery (CSRF). URL: https://owasp.org/www-community/attacks/csrf (дата звернення: 22.11.2025).

41. FastAPI. CORS (Cross-Origin Resource Sharing). URL: https://fastapi.tiangolo.com/tutorial/cors/ (дата звернення: 22.11.2025).

42. MDN Web Docs. 413 Payload Too Large. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/413 (дата звернення: 22.11.2025).

43. Кодекс законів про працю України. Верховна Рада України. URL: https://zakon.rada.gov.ua/laws/show/322-08#Text (дата звернення: 24.11.2025).

44. Про охорону праці. Верховна Рада України. URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text (дата звернення: 24.11.2025).

45. Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями. Верховна Рада України. URL: https://zakon.rada.gov.ua/laws/show/z0508-18#Text (дата звернення: 24.11.2025).

46. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин. Верховна Рада України. URL: https://zakon.rada.gov.ua/rada/show/v0007282-98#Text (дата звернення: 26.11.2025).

47. ДСТУ 8604:2015 Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги. БУДСТАНДАРТ Online URL: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=71028 (дата звернення: 26.11.2025).

48. ДБН В.2.5-56:2014 «Системи протипожежного захисту». Експертно-технічний центр. URL: https://etz.com.ua/systemy-protypozhezhnogo-zahystu/ (дата звернення: 26.11.2025).

49. ДСТУ 4297:2004 – Технічне обслуговування вогнегасників. ksv.biz.ua. URL: https://www. ksv.biz.ua/GOST/DSTY_ALL/DSTY3/dsty_4297-2004.pdf (дата звернення: 26.11.2025).

50. Про затвердження Правил безпечної експлуатації електроустановок споживачів (ДНАОП 0.00-1.21-98). Верховна Рада України. URL: https://zakon.rada.gov.ua/laws/show/z0093-98#Text (дата звернення: 26.11.2025).

51. Про введення в дію Державних гігієнічних нормативів «Норми радіаційної безпеки України (НРБУ-97)». Верховна Рада України. URL: https://zakon.rada.gov.ua/rada/show/v0062282-97#Text (дата звернення: 01.12.2025).

52. Дія іонізуючого випромінення на організм людини. Львівський державний університет безпеки життєдіяльності ДСНС України. URL: https://virt.ldubgd.edu.ua/mod/page/view.php?id=1577(дата звернення: 01.12.2025).

53. Види іонізуючих випромінювань, їх фізична природа та особливості розповсюдження. ATutor. URL: https://dl.tntu.edu.ua/content.php?cid=289166 (дата звернення: 01.12.2025).

54. Lypa B., Horyn I., Zagorodna N., Tymoshchuk D., Lechachenko T. Comparison of feature extraction tools for network traffic data. CEUR Workshop Proceedings. 2024. vol. 3896. P. 1-11 (дата звернення: 01.12.2025).

55. Харченко О., Яцишин В. Розробка та керування вимогами до програмного забезпечення на основі моделі якості. Вісник ТДТУ. Тернопіль, 2009. Т. 14. №1. С. 201-207.

56. Shevchuk, R., Lishchynskyy, I., Ciura, M., Lyzun, M., Kozak, R., & Kasianchuk, M. (2025). Application of Blockchain Technology in Emergency Management Systems: A Bibliometric Analysis. Applied Sciences, 15(10), 5405.

57. Stadnyk, M., & Palamar, A. (2022). Project management features in the cybersecurity area. Вісник Тернопільського національного технічного університету, 106(2), 54-62.

58. Derkach, M., Matiuk, D., Skarga-Bandurova, I., & Zagorodna, N. (2025). CrypticWave: A zero-persistence ephemeral messaging system with client-side encryption.

59. Revniuk, O., Zagorodna, N., Kozak, R., & Yavorskyy, B. (2025). Development of an information system for the quantitative assessment of web application security based on the OWASP ASVS standard. Вісник Тернопільського національного технічного університету, 118(2), 56-65.

60. Derkach, M., Matiuk, D., Skarga-Bandurova, I., Biloborodova, T., & Zagorodna, N. (2024, October). A Robust Brain-Computer Interface for Reliable Cognitive State Classification and Device Control. In 2024 14th International Conference on Dependable Systems, Services and Technologies (DESSERT) (pp. 1-9). IEEE.

ДОДАТКИ

Додаток А Список адрес для тестування анкети

0x6Ad7b553D3622C7D1cCfe702E5a8BC7E22d9f848, шахрай

0xc779e354e0833B954753ea4E4784393a5c39172d, шахрай

0x7FCd25389dAD3F5C8c657586a6B09d673606b670, шахрай

0xA981c9d556cC05Eabe840811a7F1c9635D5C6616, шахрай

0x7d6c53A4bB1468AcF8016c917eD2eAB506d50c18, шахрай

0xFA165b79AAb830c73F6458ebb1c1A93d64bB0fBD, шахрай

0xF9119A4E227A5d9D6B53074f1C1fae87A85Bbd1E, шахрай

0x457983910c848C79BFc36643C4FD7C8783CdA583, шахрай

0x2041818D15eD7BaA10880ACE24eCE7a4505C8b47, шахрай

0xa80Eb788D41dC122E4D63f10C961204320F0D76C, шахрай

0x6692C76B00a56Eb9AB614296F9b364873c4B08B2, шахрай

0xe39c6E2165921E9798Cf7c22f4ffdB4c529C2965, шахрай

0x45e3A3eA33a2529DE82B7e64CBcfc84850679b1d, шахрай

0x0509ae24AEd2aCBCb7f115fe8344Acef70895237, шахрай

0xDc5Cb57711aC6eA18Bc9e07404a3Fa2a9B4913e9, шахрай

0x3d1F2aC606C0Db8796b2D6Fd23BabAc10f1a9719, шахрай

0xcF13050B1C6162879fCAadaE6295FDC30cb12105, шахрай

0x6198deb6d25F4CFd116150a800ECC18B73Eda98f, шахрай

0x55211f32b23AB491BD6271c719fa859fe6CA3024, шахрай

0x079A88eB7f92a429BdDD03508b36B908a305663F, шахрай

0x0f96374988a82446381BaeF0a6e93B5ce22A8b3E, шахрай

0x6f8c0Be6acA01A613ad2cC6077aC6dc41652a871, шахрай

0x3BC4F349252a65d07BeFE8FE656FceEF05fF3Bff, шахрай

0xB86615B09C49930DFAEFcb7B61E6285eD755ffdD, шахрай

0xEf3587feCF52D061ad662B0cAc4Ce63A660a0279, шахрай

0x817a3EbEeE02B80c2B461CC1050228050467Eb65, шахрай

0x7Da4b2c7D4436Ba35c6846855032aC070AA8B91B, шахрай

0x928F8e6cAEC51ADFd568a0F068F484Fc193C02aC, шахрай

0x9898ACE08C6f5Ec99e66B09d4379b90B9B4B5eC7, шахрай

0x63b8c3b60b558dD6EAc212A33c5f591e111fA457, шахрай

Продовження Додатку А

0xabCfd924a8580B70D06803e2A20C1E1B9BE3A0BE, шахрай
0x2187CE526526e0267f943cD4843d4Eb55aAAc6B5, шахрай
0x943999a6D7Bf015D992A741FEED6fADf0Cd03d6e, шахрай
0xA9892BD02e289B3913693Ca8683EFE1c9C2F2481, шахрай
0xCce1015f10052d4D4116886A8842b38C71b2D186, шахрай
0xEf3587feCF52D061ad662B0cAc4Ce63A660a0279, шахрай
0x3aCFcf07257805c11b9D726a4dfa1E505315b111, шахрай
0x2192AC1e96478cf064b10C2258f0e4c6529925f2, шахрай
0x41bd950a5186EDB30290f0454C3b0D5C66d50C13, шахрай
0x5668c688b621Cc9F5710597CF2DCF0A77729d34c, шахрай
0x92105c9037F72eE35cbf42aaA03672968EA77884, шахрай
0x63D6E0F42d3174Ff5014408421467cb2C403Fa14, шахрай
0xc5272f3D8D72C653D7c7bcb7Ed78380f7FC0D88c, шахрай
0xcC82A4bD6fc5E0D37d888564E95C263c96363087, шахрай
0x2DDc3eEd66376202E87190239a43e8cc83BecAf4, шахрай
0xFf95B23552D29F53ee5dE454654210aA0E3C328b, шахрай
0x3AD636877D92435A238a6BEf76b79c082AA82A53, шахрай
0xdd1C71cD19e4136247295c8df913A540b9a6F3C6, шахрай
0xFFB7d41716018573c36370E6166d153C25083aBe, шахрай
0x9CdcA3BAA620EEF0e5cB65edE626A9A4834674ba, шахрай
0xf35717a56e190f9C81b2F2c07d62fB4f12CBFD3A, airdrop scam token
0xE1A913f12e2215CdeEF8Fa2257cbA7FDb9B3bf60, airdrop scam token
0x439C86d5D2fDD16884e21dbd43FA5D781cAB8A1d, airdrop scam token
0xE29A7F7c89e42F94B9b9B18770b9A0C635A0E418, airdrop scam token
0xd3683cC752010a137575d55C14618F50B556Dfa1, airdrop scam token
0x0d44441617e2D1A2D5dCc254a18c83FC57a115cc, airdrop scam token
0x22d150642734734fA2E4d9B251aF95Ea38c89f0c, airdrop scam token
0xA7C3cE3fDE5e5C5f3393d0aA8A0335Affa5B5571, airdrop scam token
0x955Ac77feDee5f3aAa8B17B8cEb5Dd64e94b63bA, airdrop scam token

Продовження Додатку А

0x98198644AB2Fa1C9570d7159e13D30E3E70D2984, airdrop scam token
0x6692C76B00a56Eb9AB614296F9b364873c4B08B2, airdrop scam token
0x74721B13971dA1F421A7b106B0532B49fC5Dc2d9, airdrop scam token
0x7CBd47336C51CC77d50C1EFA5369A83F683A45bC, airdrop scam token
0x7B6B8c4b73151EF4fc62892B44e4bd2d161CFA3E, airdrop scam token
0x93185cD7d7ACF0317d4dd644A63Bd6a26cF5D1D5, airdrop scam token
0xb8d04CC939e5A47DaC39DdC17E1BF081CD3c1E98, airdrop scam token
0xe79Db5b49840b69DCFc84411DeDbC83f20F7857a, airdrop scam token
0x8bF2f42dF79e8CfeA153a935AC42e1c27F56F03b, airdrop scam token
0x6c06E9B2134b8d82BfC9291D87798cac5619C3ef, airdrop scam token
0xD02aDd4B1e80a5f82454568309Dd48b57B9a27ae, airdrop scam token
0x5b823EEFEB66D0B4c694CBC8b73e7d523FB04BD4, airdrop scam token
0xF1458FFB89A30fA3E3BFF83517E689495f8B9902, airdrop scam token
0xe891c8AD61C5bB63FE8DCe16bd31780dB18E304d, airdrop scam token
0xcAeEdF4259E7f157451F36d82dD275A2047747F3, airdrop scam token
0x7f1b6fB28901b11d32C288Ec840771035eE580B5, airdrop scam token
0x27F47F82e0a8BD06eaB4778E7809E86c977B098C, airdrop scam token
0x3ED18F61be1D19b49f97739A127dCEAFDeEcCbE5, airdrop scam token
0x600056530c01f54Dc19A5b3F584f6539b08e3E34, airdrop scam token
0x3206a702B7a6Ad4f5B8a6CE50142a18b28bb6C8D, airdrop scam token
0x54D4d5fC253d1F3A74C7B65DaE1DD5f4AcB05a47, airdrop scam token
0x29c8c9bFcB0198479Ac07F71e9DE1B014dD0D379, airdrop scam token
0xd4F28E7bC69545C0D9E3647300cbbd0DfB6a670c, airdrop scam token
0xC7D3Fe00134069aE7769bC7D93c71EFC18391cB1, airdrop scam token
0xcBA360df45Bbb3DdD97dA98d4B26Fc860ebC2d32, airdrop scam token
0xbB057b406ADfBF0930D71f8bac0413132e30BBC3, airdrop scam token
0xfa6D07FB903fE51f1A3a6357791F1De49DEeB2e3, airdrop scam token
0x5Cfd2F9c803B363f240988fEA1424E38d09aC155, airdrop scam token
0x6114b3A7Cc3E891e2F7DED71e98430a91700D7Ea, airdrop scam token

Продовження Додатку А

0x10D280E9f1C91c84fA1A1F03c1b247c198822A0D, airdrop scam token
0x2088933e4242cc7020Fb8Fb18481c7d22F3e8a55, airdrop scam token
0x161A4682A69A0Cf35713268f1348A068D745A5D2, airdrop scam token
0xC8bAf729A1eC6E864437B890349934195138977F, airdrop scam token
0x7A481A68D1eb076f6941A64885E69851b90ea15e, airdrop scam token
0x43267fc194469d6d94CE7a1922AaEbB86F74B65d, airdrop scam token
0xFC75410f6f4B3f00bAeA1a95afde14De51Df01FB, airdrop scam token
0x4e0Ec8FBb7fac6B1819E21A252622E51AC2bC6b2, airdrop scam token
0xCde3b89A788EF622fEDDF05a6b8fC1d6Faa6D31a, airdrop scam token
0xFFF851B3777586A6F0eb0FB1EeD29180988b8F90, airdrop scam token
0x31ab9DF53b1917C9E7B25aDD37F8A2a8aDa1A6eE, airdrop scam token
0x91f9Fd4F9b657396028F271b38Aca3B30acB6350, airdrop scam token
0x25d887Ce7a35172C62FeBFD67a1856F20FaEbB00, чистий токен
0x7083609fCE4d1d8Dc0C979AAb8c869Ea2C873402, чистий токен
0x2859e4544C4bB03966803b044A93563Bd2D0DD4D, чистий токен
0xbA2aE424d960c26247Dd6c32edC70B295c744C43, чистий токен
0x1D2F0da169ceB9fC7B3144628dB156f3F6c60dBE, чистий токен
0x0555E30da8f98308EdB960aa94C0Db47230d2B9c, чистий токен
0x2170Ed0880ac9A755fd29B2688956BD959F933F8, чистий токен
0x570A5D26f7765Ecb712C0924E4De545B89fD43dF, чистий токен
0x524bC91Dc82d6b90EF29F76A3ECAaBAffFD490Bc, чистий токен
0xCE7de646e7208a4Ef112cb6ed5038FA6cC6b12e3, чистий токен
0xC0BC84e95864BdfDCd1CCFB8A3AA522E79Ca1410, чистий токен
0x000ae314e2a2172a039b26378814c252734f556a, чистий токен
0x4A64515E5E1d1073e83f30cB97BEd20400b66E10, чистий токен
0x4c9EDD5852cd905f086C759E8383e09bff1E68B3, чистий токен
0xBc65ad17c5C0a2A4D159fa5a503f4992c7B545FE, чистий токен
0x3c3a81e81dc49A522A592e7622A7E711c06bf354, чистий токен
0x582d872A1B094FC48F5DE31D3B73F2D9bE47def1, чистий токен

Продовження Додатку А

0x6c3ea90364068520062907708BEdFcAbA0e23A0e8, чистий токен

0x9D39A5DE30e57443BfF2A8307A4256c8797A3497, чистий токен

0x6B175474E89094C44Da98b954EedeAC495271d0F, чистий токен

0x4da27a545c0c5B758a6BA100e3a049001de870f5, чистий токен

0x57e114B691Db790C35207b2e685D4A43181e6061, чистий токен

0x85F17Cf997934a597031b2E18a9aB6ebD4B9f6a4, чистий токен

0x7712c34205737192402172409a8F7ccef8aA2AEc, чистий токен

0x514910771AF9Ca656af840dff83E8264EcF986CA, чистий токен

0xe343167631d89B6Ffc58B88d6b7fB0228795491D, чистий токен

0x163f8C2467924be0ae7B5347228CABF260318753, чистий токен

0x4B948d64dE1F71fCd12fB586f4c776421a35b3eE, чистий токен

0x53E0bca35eC356BD5ddDFebbD1Fc0fD03FaBad39, чистий токен

0x61299774020dA444Af134c82fa83E3810b309991, чистий токен

0x2C89bbc92BD86F8075d1DEcc58C7F4E0107f286b, чистий токен

0xFF7F8F301F7A706E3CfD3D2275f5dc0b9EE8009B, чистий токен

0x6985884C4392D348587B19cb9eAAf157F13271cd, чистий токен

0x45c32fA6DF82ead1e2EF74d17b76547EDdFaFF89, чистий токен

0x9c2C5fd7b07E95EE044DDeba0E97a665F142394f, чистий токен

0x3Cef98bb43d732E2F285eE605a8158cDE967D219, чистий токен

0x65e7a112dB1142EAE919201b1232f7aA488Ed83c, чистий токен

0xb3Ed0A426155B79B898849803E3B36552f7ED507, чистий токен

0x3e6648c5a70a150a88bce65f4ad4d506fe15d2af, чистий токен

0xE4D5c6aE46ADFAF04313081e8C0052A30b6Dd724, чистий токен

0x40BD670A58238e6E230c430BBb5cE6ec0d40df48, чистий токен

0xdA5e1988097297dCdc1f90D4dFE7909e847CBeF6, чистий токен

0x696F9436B67233384889472Cd7cD58A6fB5DF4f1, чистий токен

0x226A2FA2556C48245E57cd1cbA4C6c9e67077DD2, чистий токен

0x2B11834Ed1FeAEd4b4b3a86A6F571315E25A884D, чистий токен

0xdC035D45d973E3EC169d2276DDab16f1e407384F, чистий токен

Продовження Додатку А

0x3EE2200Efb3400fAbB9AacF31297cBdD1d435D47, чистий токен

0x43C934A845205F0b514417d757d7235B8f53f1B9, чистий токен

0x6FDcdfef7c496407cCb0cEC90f9C5Aaa1Cc8D888, чистий токен

0x031b41e504677879370e9DBcF937283A8691Fa7f, чистий токен

0x0B64B442504Cdfe25Ac223d477BB596b2F26dDE7, жертва

0x52B9928A698Bc9c6343c31C89692A5Ed5dD7Ea6d, жертва

0x787630Ab62EbF1C033186fcb04457fde70A140E7, жертва

0x7d0a456ab97B398c7E9cAEa266161288aA862e09, жертва

0x358505FFD867eb71a79b976f889be29f4ff8F884, жертва

0xFd75e3B46d81492E6869E747db64CDde78E0a7cf, жертва

0x408C841544810065Ea2F3785A56D55358246Ac8c, жертва

0xF856b8143366eA82e5ea1271b43373c4851a72eD, жертва

0x846C63cDA738e089878FA3Db2608Fe1123864bf0, жертва

0x8A8B81f105Fe965A1a4DbE128A97A4eB35fA9500, жертва

0x57AAd9a904635de3998ea541F5Bb6AcEd0ccfC44, жертва

0xddF0Bb6d4B18eCC5213e448933de8750bed5156c, жертва

0x29ab0711A9C1A9cE232F22f8d914EB2d77c95024, жертва

0x1aD88E035117e6fe281A154C6032C66357925565, жертва

0x9D022615FdB3aD59e5fcf9C7246aF4DA6A161f1B, жертва

0xD2E3aCf71832f75232B01388505965548d774E0A, жертва

0x72D5C9b95653F581163427996a53a413EeF187f3, жертва

0xA87bdAe5A8BfCe65d00ad8b5cd893031193c029A, жертва

0xE692e0704a5481aDC4A230C170f16884EeF8cf0B, жертва

0x61552D35f92cd07a82aF8646E4BDF70B3D2CAb65, жертва

0x856c2A1f746E5249e4aAa19837985A5302cc1599, жертва

0xD005615A287B6d00724D9a722E19737D6F47Cc27, жертва

0xB37A19E271D79c0Ab84d3a434f8DB5e5aaC33210, жертва

0x0D02C09833D1420348d7F65192B6bF9279A776bF, жертва

0x149958aA806F8d11cA745ffDEAE713B52531A74D, жертва

Продовження Додатку А

0xB20041E8Ad5c0B6D7530578DCA6288FeE38e3ac5, жертва

0x73Fc76cF0367A899E643B9D649eBfC01fb7d1EE5, жертва

0x78715332C1E65489873353E7A828CfD54887A963, жертва

0x68a358e445accB6F7d97B7755dD480e77F523020, жертва

0xfC53963B606f044816815a6F2D717356a139CE93, жертва

0x9dc81196e17bEa1B30f432FFE6aF3680608FF653, жертва

0x4B205AfCC63a07De26612BF3a755c7da383fE106, жертва

0xB3efc31E19A82E65aFc73E8C1CeBc961DA269f2a, жертва

0x8Fd4AAb5eB67292e6eE0d97fc5b6d9496091bbA4, жертва

0xd76DF6d807Af7182D0c237CB6Ca1C44dca6fc315, жертва

0xEEB82ec4E108B9F1fc0a4838a855f87E5C8ea22b, жертва

0x2c57B1f2ef108a524CBE15B9C58C088583F59783, жертва

0x7400485cAa2D822de6e850d5460F7F4460945257, жертва

0xdd53cFa385792d1bf0730330Ed0aE8C20D161BC4, жертва

0x7736a721ce90abDB8183F9CbE9E6F6B61C32CEf2, жертва

0x4788cECFbED4a2cBc3E006368A54f4C5174a3cd8, жертва

0xF01D2A83a89beE031667eda60428f2fe5760C4c9, жертва

0xe62d677A3A52B283E6362D1f81AD5c838855e673, жертва

0x12EEc835685F3A3BddEc084bFD320fee7f5F33fe, жертва

0x660Fa51E06E73BBeA6ea836d693dEb6061368017, жертва

0xBe87a5A9707937e9CEb21B13980c3964F1903e3c, жертва

0xBc22bDB8F1c6485fFf3631DFA4955e5fB0B0F3AD, жертва

0x68279d8E732A2E8d2A122ba2a92fD310B2eC5f4b, жертва

0x014603F096f9142C0F2908413E00575E1E357ee2, жертва

0x54bde1FB00a2FE6e2C01686d655b5818BC6C90b3, жертва

0xc7075E54f55f7c7393Ba296E11cA4Bb9ebcD97D8, чиста адреса

0xb037f6CB00F48379869507af56759Ca3f7E44F04, чиста адреса

0xa7156D011c762E41bAb80818b0bFEE6f1018e49e, чиста адреса

0x573bC32b228BB2599c598bD8Fe0988861A25071f, чиста адреса

Продовження Додатку А

0xDb2420c780aEbD89bd6a5eb4c5dd722ca6133B6C, чиста адреса
0xa7156D011c762E41bAb80818b0bFEE6f1018e49e, чиста адреса
0x7d2619824B2e4F12757899d711CCE80A73427D6E, чиста адреса
0x17eb56780d0dAefB98422723658E396cE1069711, чиста адреса
0xB1741C259190B7Ee5DC9C8878b2fD7E75f0f2904, чиста адреса
0xCCadCc763240E47cd985484D632Eb232E71cA282, чиста адреса
0xE249324a45A1b07568Fa8A5E84Eed149BF635635, чиста адреса
0xAFcA144d681dc2D042225b295e33d9c27369B3B9, чиста адреса
0x4302b02a4AE2f16f2E355B2bD3c55dF06aCa91eB, чиста адреса
0x43D1C354098672202e92abeF6BFA6659A1Ae021B, чиста адреса
0xd7BfD4F9de8016C0A28FD1AA8A3AcbA460563492, чиста адреса
0xb037f6CB00F48379869507af56759Ca3f7E44F04, чиста адреса
0xDa357cC29aa4d2B6aF4B910a12D39bCd4A253cd0, чиста адреса
0x3447F702C4bc1539793b593Bb69eA5E65a21f651, чиста адреса
0x09FB4C25f675982AE41C004E4275861C074f1ACF, чиста адреса
0xE83E7283fb293fE4839FaABb5F4D8AA8d28e3f58, чиста адреса
0x0029E1FEb7ff6bA4f7CAA3FAB2d5B0F5790e1fE0, чиста адреса
0x66448D6818F5208390b8754c20Db8ec624e2F328, чиста адреса
0x578c2B9A0FEA689809d052D36501a38DDE4102e6, чиста адреса
0x3B699bB64969Cbb38B7C59e3Ae3ab2C32Fc189D9, чиста адреса
0xF9a70404cBF6195063Acfbd3Cd8D0EC69033956d, чиста адреса
0xDD90b39c16CA26a669BCF62Fe2b19e319F9a2b96, чиста адреса
0x4A21aC09586aE28bbF8C09Ba2F7d02C96A015701, чиста адреса
0x305aB48EBCb69dDca2a23dbCc70D2Ffa7677be3D, чиста адреса
0xeb96FacF9080b2d93857f41430BDD1fF6d165337, чиста адреса
0x658DB5361048B680f74Cf1d85772C5a9C6719a8b, чиста адреса
0x54DbfEc36b7963C43BE2fa0d23d2BB3F00E0bd7d, чиста адреса
0xFb78A2F8BC1df22B6f19ac27680A052fCEEc5127, чиста адреса
0xF6D2C7249B4a08E238EEcc04103632a0cdbfa948, чиста адреса

Продовження Додатку А

0x59202EC8978Ed18ceC70cED926dF193A6C8db5b7, чиста адреса

0x19A12c1eBCb4E6c5ffb89B55d726DE77fd4cCb58, чиста адреса

0xa367e5711486B68fa3c04D63C759d88946Ac744f, чиста адреса

0x1296a3b1413415A16c0523838F58c5462676bAD0, чиста адреса

0x5d6d03d2f9D1D913FA19B577b6F8Be2B4f944A4e, чиста адреса

0xe8ee98B0AeF2a558C5Eb22Ee10D1Ceb28844f9bf, чиста адреса

0xA41Ac33e5C8572b928c2F57f205ff0c60b0a1905, чиста адреса

0xf23ad2F7929acac07A64B045C124623A756df50a, чиста адреса

0x159b6970E6f133535832fFa190424C6a9f8Ec1e2, чиста адреса

0x90183529de066ad2139f35C2DE4c9A95392d12dE, чиста адреса

0x8ACE61b9c273e6e49e4f3FF4F4145cEe7149Ba58, чиста адреса

0x51280a942444ad85C9D121655ea15F067df8B964, чиста адреса

0x2aFd0A57d0277a8dc6999A3F7de129F090Bc877A, чиста адреса

0x9450118D83f6994cf8569E2Fca0f88f16e6B6a3b, чиста адреса

0xF3E2618E5B76f95154376f2b45851F11aCee8070, чиста адреса

0x8ec67Bc701295B0eaF652651A56ab2A49A4A4712, чиста адреса

0xfc5caF8Bb93e3CF9E3853C7028A0560870e14288, чиста адреса

Додаток Б Публікація

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
імені ІВАНА ПУЛЮЯ
НАУКОВЕ ТОВАРИСТВО ім. ШЕВЧЕНКА

XIII

НАУКОВО-ТЕХНІЧНА

К О Н Ф Е Р Е Н Ц І Я

Інформаційні моделі, системи та технології

17-18 грудня 2025 року

ТЕРНОПІЛЬ – 2025

УДК 004.56
А. Кобельник; І. Скарга-Бандурова, д. т. н., проф.
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

РОЗРОБКА ІНСТРУМЕНТУ ДЛЯ ВИЯВЛЕННЯ ШАХРАЙСЬКИХ ДІЙ В БЛОКЧЕЙНІ

UDC 004.56
A. Kobelnyk; I. Skarga-Bandurova, Doctor of Technical Sciences, Prof.

DEVELOPMENT OF A BLOCKCHAIN FRAUD DETECTION TOOL

Розвиток блокчейну за останні кілька років перетворив його з нішевої технології на глобальну інфраструктуру вартістю понад 3,2 трлн доларів [1], але водночас створив ідеальне середовище для наймасштабнішого в історії перерозподілу капіталу через шахрайство. Децентралізація, прозорість та незмінність даних у блокчейн-середовищах забезпечують нові можливості для користувачів, проте одночасно створюють сприятливі умови для появи складних, автоматизованих форм шахрайства. Масштабність і відкритість Web3 призводять до того, що виявлення ризиків і відокремлення легітимної активності від шахрайської стає технічно складним завданням.

Провідні аналітичні платформи, такі як Chainalysis [2], Elliptic [3], TRM Labs [4], що спеціалізуються на криптобезпеці та AML-комплаєнсі, пропонують моніторинг транзакцій та криміналістичну експертизу. Однак, їхні послуги орієнтовані насамперед на великі фінансові установи та правоохоронні органи, вимагаючи значних фінансових ресурсів і доступу до закритих корпоративних інтерфейсів. Це залишає звичайний користувачів практично без інструментів для самостійного аналізу даних.

З огляду на це, актуальним стає створення доступного інструменту, який дозволяє кожному користувачеві самостійно оцінювати безпеку взаємодії з EVM-адресами. Він вирішує кілька ключових завдань: збір і систематизацію відкритих ончейн-даних, наочну візуалізацію транзакційних взаємодій, проведення розслідувань та формування зрозумілої оцінки ризику.

Інструмент використовує дані з відкритих джерел і безкоштовних API, що робить його доступним для широкого кола користувачів і незалежним від дорогих корпоративних рішень. Отримані дані відображаються у вигляді графів, що дозволяє швидко оцінити структуру зв'язків між адресами та помітити потенційно підозрілу активність. Оцінка ризику здійснюється автоматично або інтерактивно у вигляді анкети, що забезпечує працездатність навіть при обмеженому доступі до потрібних даних через API. Такий підхід не лише зберігає функціональність системи, а й перетворює процес аналізу на навчальний досвід: користувач освоює принципи розпізнавання шахрайської активності та може застосовувати ці знання в майбутньому. Проведене тестування на реальних випадках показало, що інструмент демонструє високу точність оцінки ризику та надійно ідентифікує підозрілу активність у різних сценаріях.

Перспективи розвитку охоплюють підтримку non-EVM блкочейнів, вдосконалення методів аналітики та взаємодій через кросчейн мости, розширення оцінки різних видів шахрайства та впровадження модуля для аналізу смарт-контрактів, що дозволить виявляти потенційно небезпечну логіку всередині контрактів. Додатково розглядається створення бази знань, що сприятиме формуванню в користувачів практичних навичок оцінювання ризиків і загальної цифрової безпеки, адже усвідомлене розуміння механізмів шахрайства та принципів безпечної поведінки суттєво знижує ймовірність фінансових втрат.

Література

1. Cryptocu Global Cryptocurrency Market Cap Charts. CoinGecko. URL: https://www.coingecko.com/en/charts.
2. The Blockchain Data Platform. Chainalysis. URL: https://www.chainalysis.com/.
3. Elliptic: Blockchain Analytics & Crypto Compliance Solutions. Elliptic. URL: https://www.elliptic.co/.
4. TRM Labs | Blockchain Intelligence Platform. TRM Labs. URL: https://www.trmlabs.com/.