Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
В.о. завідувача кафедри
кібербезпеки та захисту інформації
_____ Іван ПАРХОМЕНКО
«____» червня 2023 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)

спеціальність 125 Кібербезпека
(код і назва спеціальності)

освітній ступень бакалавр

освітня програма Кібербезпека
(назва освітньо-професійної програми)

на тему: Удосконалення програмних засобів криптогаманця у Web3 просторі на базі блокчейн технологій

Виконавець: студентка IV курсу, групи КБ-41

_____ **Нікіта СКАЧКО**
(підпис) (ім'я, прізвище)

	Ім'я, прізвище	**Підпис**
Керівник	Олександр ТОРОШАНКО	

Нормоконтроль	Андрій БІГДАН	

Київ 2023

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри кібербезпеки
та захисту інформації
_____ Сергій ТОЛЮПА
«24» жовтня 2022 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньо-професійної програми)

Студенту **КБ-41** **Скачка Нікіти Сергійовича**
(група) (прізвище ім'я по батькові)

Тема кваліфікаційної роботи Удосконалення програмних засобів криптогаманця у Web3 просторі на базі блокчейн технологій

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема кваліфікаційної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №3 від 20.10.2022 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Криптогаманці, Web3 простір

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Необхідно ознайомитися роботою криптовалютних гаманців, їх типовими розгортаннями та послугами, вразливостями з боку безпеки даних, використанням та положенням в Web3 системах, практино розробити метод захисту користувачів при роботі з криптовалютними гаманцями.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Удосконалено систему безпеки криптовалютного гаманця у Web3 просторі

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 24 жовтня 2022 року

Завдання видав _____ (підпис) Олександр ТОРОШАНКО (ім'я, прізвище)

Завдання прийняла до виконання _____ (підпис) Нікіта СКАЧКО (ім'я, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	24.10.2022 – 24.01.2023	*виконано*
2	Аналіз літератури	25.01.2023 – 28.04.2023	*виконано*
3	Обґрунтування вибору рішення	29.04.2023 – 02.05.2023	*виконано*
4	Огляд Web3 простору	03.05.2023 – 07.05.2023	*виконано*
5	Аналіз криптовалютних гаманців	08.05.2023 – 13.05.2023	*виконано*
6	Дослідження вразливостей та загроз	14.05.2023 – 17.05.2023	*виконано*
7	Розробка удосконалення до криптовалютного гаманця для забезпечення покращенної безпеки користувача	18.05.2023 – 27.05.2023	*виконано*
8	Оформлення пояснювальної записки	28.05.2023 – 01.06.2023	*виконано*
9	Підготовка до захисту кваліфікаційної роботи	02.06.2023 – 12.06.2023	*виконано*

Завдання видав _____ (підпис) Олександр ТОРОШАНКО (ім'я, прізвище)

Завдання прийняв до виконання _____ (підпис) Нікіта СКАЧКО (ім'я, прізвище)

Термін подання кваліфікаційної роботи до ЕК 12 червня 2023 року

РЕФЕРАТ

Кваліфікаційна робота складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, додатків, має 52 сторінки основного тексту, 4 таблиці та 4 рисунки. Список використаних джерел містить 20 найменувань і займає 2 сторінки.

Методи дослідження кваліфікаційної роботи:

- аналіз документів;
- аналіз літератури;
- порівняння;
- власний досвід;
- історичний метод;

Об'єктом дослідження процес захисту програмних засобів криптогаманця у Web3 просторі на базі блокчейн технологій.

Предметом дослідження в даній роботі є методи та засоби захисту власності користувача.

У роботі проаналізована існуюча література з блокчейн технологій, виконаний аналіз документів, порівняння, вивчення та узагальнення зарубіжної практики з теми безпека користувачів у Web3 просторі.

Розроблений лістинг програми, що виконує попередження щодо шкідливих транзакцій, може використовуватися користувачами для отримання інформації про безпеку своїх транзакції в криптогаманці.

Ключові слова: Захист власності, криптогаманець, безпека в Web3 просторі, програмні засоби, децентралізованість, транзакції, смарт-контракти, токени.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

DApp	–	Decentralized Application
ICO	–	Initial Coin Offering
KYC	–	Know Your Customer
2FA	–	Two-Factor Authentication
DeFi	–	Decentralized Finance
VPN	–	Virtual Private Network
NFT	–	Non-fungible token
DEX	–	Decentralized exchange
OS	–	Operating system

ЗМІСТ

ВСТУП

Актуальність теми захисту криптовалютних гаманців пов’язана зі стрімким розвитком крипто простору. Криптовалютні гаманці є важливими інструментами, які дозволяють користувачам безпечно зберігати свої цифрові активи, керувати ними та здійснювати операції з ними. Однак децентралізована та псевдонімна природа криптовалют також створює унікальні проблеми з безпекою.

Екосистема криптовалют постійно розвивається, регулярно з'являються нові технології та протоколи. Ця робота досліджує нові технології, такі як криптовалютний гаманець, блокчейн, смарт-контракт, мультипідписна автентифікація та апаратні криптовалютні гаманці безпеки, котрі можуть бути інтегровані в програмне забезпечення гарячих гаманців для підвищення безпеки та захисту власності користувачів [1, с. 49].

Багато криптовалютних гаманців є проектами з відкритим вихідним кодом, що дозволяє спільноті робити внески та проводити експертну оцінку. Кваліфакаційна робота сприятиме розвитку спільноти з відкритим вихідним кодом, розробляючи та поширюючи інноваційні програмні рішення для гаманців, тим самим приносячи користь як користувачам, так і розробникам.

Аналіз останніх досліджень та літератури. Вчені та люди які зробили вклад у розробку та вивчення криптопростору: Сатоши Накамото, Віталій Бутерін, Гевін Вуд, Девід Чаум, Ден Боне.

Мета роботи є розробка рекомендацій та покращення програмного забезпечення для збільшення рівня безпеки користувачів криптогаманців, які використовуються в криптопросторі.

Для досягнення поставленої мети необхідно вирішити такі *завдання*:

- Розглянути особливості побудови криптовалютних гаманців та Web3 простору.
- Дослідити типові вразливості в криптовалютних гаманцях.

- Розробити власний метод захисту власності користувачів криптовалютних гаманців.

Об'єктом дослідження в даній роботі є можливість захисту власності користувача в Web3 просторі.

Предметом дослідження в даній роботі є методи та засоби захисту власності користувача.

Методи дослідження кваліфікаційної роботи:

- аналіз документів;
- аналіз літератури;
- порівняння;
- власний досвід;
- історичний метод.

РОЗДІЛ 1
АНАЛІЗ ПОТОЧНОГО СТАНУ КРИПТОГАМАНЦІВ У WEB3 ПРОСТОРІ

1.1 Огляд існуючих криптогаманців на базі блокчейн технологій

Перед тим як розглянути криптогаманці дізнаємося, що таке Web3 простір та криптогаманець.

Web3 – це третє покоління інтернет-технологій, яке має на меті децентралізувати мережу та розширити можливості людей, надавши їм більший контроль над своїм цифровим життям. Вона охоплює набір технологій, протоколів і стандартів, які дозволяють створювати і використовувати децентралізовані додатки (DApps), платформи децентралізованих фінансів (DeFi) і сервіси на основі блокчейну.

Web3 побудований на основі технології блокчейн, яка є розподіленим реєстром, що реєструє транзакції на декількох комп'ютерах у прозорий і безпечний спосіб. На відміну від Web 2.0, який покладався на централізовані платформи та посередників, Web3 має на меті повернути контроль користувачам, використовуючи децентралізовані мережі та криптографічні протоколи.

Криптовалютний гаманець – це програмне забезпечення або фізичний пристрій, який дозволяє користувачам безпечно зберігати, керувати та взаємодіяти зі своїми цифровими активами, такими як криптовалюта або NFT токенів. Він надає користувачам доступ до своїх приватних ключів, які необхідні для авторизації транзакцій і доступу до коштів у відповідних мережах блокчейн, і контролювати їх, а також для управління ними.

Існують різні види криптогаманців і кожен з них має свої переваги та недоліки, розглянемо кожну класифікацію:

- Апаратні гаманці це фізичні пристрої, спеціально розроблені для безпечного зберігання криптовалют в автономному режимі. Вони генерують і зберігають приватні ключі в автономному режимі, знижуючи ризик онлайн-атак [2, с. 95].
- Програмні гаманці поділяються на:

▪ Десктопні гаманці це програмні додатки, встановлені на стаціонарних комп'ютерах або ноутбуках. Вони пропонують контроль над приватними ключами і зазвичай є більш безпечними, ніж веб- або мобільні гаманці.

▪ Мобільні гаманці це додатки, встановлені на смартфони або планшети, що забезпечують зручний доступ до криптовалют на ходу. Вони можуть мати такі функції, як сканування QR-коду для спрощення транзакцій.

▪ Веб-гаманці працюють у веб-браузерах і доступні з будь-якого пристрою, підключеного до Інтернету. Вони пропонують зручність, але можуть мати ризики для безпеки, оскільки приватні ключі зберігаються в Інтернеті [2, с. 97].

- Паперовий гаманець передбачає генерацію криптовалютної адреси і приватного ключа в режимі офлайн, зазвичай за допомогою веб-сайту або програмного забезпечення. Потім ключі роздруковуються на папері і надійно зберігаються. Паперові гаманці повністю автономні, пропонують підвищену безпеку, але обмежену зручність.

- Кастодіальні гаманці надаються сторонніми платформами, такими як криптовалютні біржі. У цих гаманцях приватні ключі зберігаються і управляються постачальником послуг. Хоча вони пропонують зручність, вони також передають контроль над активами платформі [2, с. 101].

- Гаманці з декількома підписами вимагають кілька підписів приватних ключів для авторизації транзакцій. Вони забезпечують додаткову безпеку, оскільки вимагають залучення декількох сторін для доступу та контролю над коштами.

Криптогаманців – безліч, тому є сенс розглядати найпопулярніші з них, для того, щоб охопити більшу кількість юзерів, котрі користуються тим чи іншим програмним рішенням. У кожного з них є свої особливості (таблиця 1.1). Ось декілька з них:

1. MetaMask – це популярний криптогаманець, який слугує розширенням для браузера для взаємодії з мережею блокчейн. Він підтримує Ethereum і токени на основі Ethereum, дозволяючи користувачам керувати своїми цифровими активами, підключатися до децентралізованих додатків (dApps) і безпечно зберігати приватні

ключі. MetaMask забезпечує зручний інтерфейс, підписання транзакцій та інтеграцію з різними мережами на основі Ethereum.

2. Trust Wallet – це мобільний криптогаманець, який підтримує декілька блокчейнів, включаючи Ethereum, Binance Smart Chain та інші. Він надає користувачам повний контроль над своїми приватними ключами, дозволяючи їм безпечно зберігати, відправляти, отримувати і робити ставки в різних криптовалютах. Trust Wallet також інтегрується з децентралізованими біржами (DEX) та додатками, пропонуючи безперебійний та зручний користувацький досвід [3, с. 118].

3. Ledger Live – це програмний гаманець, розроблений компанією Ledger, відомим виробником апаратних гаманців. Він дозволяє користувачам керувати своїми апаратними гаманцями (наприклад, Ledger Nano S або Ledger Nano X) і взаємодіяти з декількома блокчейн-мережами. Ledger Live підтримує різні криптовалюти, має зручний інтерфейс і дозволяє користувачам безпечно зберігати та здійснювати транзакції з цифровими активами.

4. Exodus Wallet - це десктопний і мобільний гаманець, який підтримує кілька криптовалют і блокчейнів. Він надає гладкий і інтуїтивно зрозумілий інтерфейс для управління цифровими активами, відстеження балансу портфеля і обміну криптовалют. Exodus Wallet підтримує такі функції, як резервне копіювання, безпечне зберігання ключів та інтеграція з апаратними гаманцями для підвищення безпеки.

5. Atomic Wallet - це мультивалютний гаманець, який підтримує різні блокчейни, включаючи Bitcoin, Ethereum та багато інших. Він пропонує такі функції, як децентралізовані атомарні свопи, стейкінг та вбудовані можливості обміну. Atomic Wallet надає користувачам повний контроль над своїми приватними ключами, забезпечуючи безпеку та конфіденційність їхніх цифрових активів [4, с. 1].

Таблиця 1.1

Порівняння характеристик криптовалютних гаманців

Криптогаманець	**Основні особливості**	**Підтримка криптовалют**	**Інтерфейс**	**Безпека**	**Мобільний додаток**
MetaMask	Розширення для браузера Ethereum	Ethereum та декілька	Інтуїтивний	Середня	Є
Trust Wallet	Мобільний гаманець для багатьох криптовалют	Багато	Інтуїтивний	Висока	Є
Ledger Live	Фізичний апаратний гаманець	Декілька	Професійний	Висока	Є
MyEtherWallet (MEW)	Веб-гаманець для Ethereum	Ethereum	Стандартний	Висока	Ні
Coinbase Wallet	Мобільний гаманець для криптовалют	Багато	Інтуїтивний	Висока	Є

1.2 Аналіз функціональності, безпеки та зручності використання

Проаналізуємо функціональність, безпеку та зручність використання кожного вище написаних гаманця.

Функціональність:

- MetaMask пропонує широкий спектр функцій, включаючи управління гаманцями, підписання транзакцій та інтеграцію з dApps. Він надає зручний інтерфейс для взаємодії з мережами на основі Ethereum.

- Trust Wallet підтримує кілька блокчейнів і надає такі функції, як управління гаманцем, зберігання активів, відправлення та отримання транзакцій, стейкінг, інтеграція з додатками та децентралізованими біржами (DEX).

- Ledger Live в першу чергу функціонує як супутнє програмне забезпечення для апаратних гаманців Ledger. Воно дозволяє користувачам керувати своїми апаратними гаманцями, переглядати баланси та надсилати/отримувати транзакції. Однак він може

мати обмежену функціональність у порівнянні з програмними гаманцями, розробленими спеціально для ширшої підтримки блокчейну.

- Exodus Wallet пропонує такі функції, як управління гаманцем, відстеження портфеля, вбудовані біржі та підтримку декількох криптовалют і блокчейнів. Він має на меті забезпечити безперебійний та інтуїтивно зрозумілий користувацький досвід.

- Atomic Wallet підтримує широкий спектр криптовалют і надає такі функції, як управління гаманцем, децентралізовані атомарні свопи, стейкінг і вбудовані біржі. Він пропонує користувачам контроль над своїми приватними ключами і має на меті спростити управління активами.

Безпека:

- MetaMask зберігає приватні ключі локально на пристрої користувача, гарантуючи, що користувачі мають контроль над своїми коштами. Вона надає можливості для встановлення плати за транзакційний газ і підтримує інтеграцію з апаратним гаманцем для додаткової безпеки.

- Trust Wallet реалізує надійні заходи безпеки, такі як зашифроване зберігання приватних ключів, біометрична автентифікація та підтримка апаратних гаманців. Він також дозволяє користувачам встановлювати паролі для транзакцій і керувати дозволами для dApps.

- Ledger Live фокусується на безпеці апаратних гаманців, використовуючи захищеність апаратних пристроїв для безпечного зберігання приватних ключів. Він використовує PIN-коди і пропонує додаткові функції, такі як захист парольної фрази.

- Exodus Wallet шифрує та зберігає приватні ключі локально на пристрої користувача. Він надає можливості резервного копіювання, дозволяє користувачам встановлювати PIN-коди для транзакцій та інтегрується з апаратними гаманцями для додаткової безпеки.

- Atomic Wallet реалізує децентралізовану архітектуру, де приватні ключі зберігаються тільки на пристрої користувача. Він використовує шифрування та мнемонічні фрази для резервного копіювання гаманця і підтримує інтеграцію з апаратними гаманцями.

Зручність використання:

- MetaMask пропонує зручний інтерфейс з інтеграцією розширення для браузера. Він надає чітку інформацію про транзакції та безперешкодну взаємодію з додатками, що робить його придатним як для початківців, так і для досвідчених користувачів.

- Trust Wallet має інтуїтивно зрозумілий інтерфейс мобільного додатку з простим процесом налаштування. Він пропонує зручний користувацький інтерфейс для управління гаманцями, надсилання/отримання активів та взаємодії з мобільними додатками.

- Ledger Live пропонує простий користувацький інтерфейс для управління апаратними гаманцями. Він надає чіткі деталі транзакцій і баланси, що полегшує моніторинг і контроль активів. Однак для користувачів-початківців він може бути складнішим для освоєння.

- Exodus Wallet фокусується на візуально привабливому та зручному інтерфейсі. Він пропонує спрощений досвід управління активами з такими функціями, як відстеження портфеля і вбудовані біржі, що робить його доступним для користувачів з різним рівнем знань.

- Atomic Wallet має досить важкий інтерфейс для нового користувача без досвіду. Він пропонує просте управління гаманцем, атомарні свопи в один клік та інші функції, які задовольнять потреби досвідчених користувачів, але не початківців.

Для новачків найкращим вибором є MetaMask який можна встановити як розширення до браузеру і TrustWallet який є на IOS та Android системах. Вони надають потрібний новачку функціонал і мають легкий та зрозумілий дизайн, але досить незахищені від зловмисним контрактів та транзакцій.

1.3 Виявлення недоліків та обмежень поточних програмних засобів

Хоча криптогаманці стали важливим інструментом для управління цифровими активами, вони все ще мають певні недоліки та обмеження (таблиця 1.2), які необхідно враховувати. Ось найпоширеніші недоліки сучасних криптогаманців:

- Ризики безпеки: Незважаючи на акцент на безпеку, криптогаманці все ще вразливі до таких ризиків, як злом, фішингові атаки, шкідливе програмне забезпечення та соціальна інженерія. Користувачі повинні бути обережними і дотримуватися найкращих практик безпеки, включаючи використання надійних паролів, двофакторну автентифікацію та постійне оновлення своїх пристроїв і програмного забезпечення.

- Відсутність зручного для користувача інтерфейсу: Багатьом криптогаманцям бракує інтуїтивно зрозумілого процесу реєстрації, що може відлякувати новачків від входження в криптопростір. Спрощення налаштування гаманця та надання чітких інструкцій може допомогти покращити адаптацію користувачів.

- Обмежена підтримка токенів: Деякі криптогаманці можуть мати обмеження в підтримці широкого спектру токенів. Це може обмежити користувачів в управлінні певними криптовалютами або участі в певних мережах блокчейн.

- Комісії та швидкість транзакцій: Залежно від мережі блокчейн і реалізації гаманця, користувачі можуть зіткнутися з високими комісіями за транзакції і повільною швидкістю транзакцій в періоди перевантаження мережі. Це може вплинути на зручність користування, особливо для тих, кому потрібні швидкі та економічно ефективні транзакції.

- Резервне копіювання та відновлення: Процеси резервного копіювання та відновлення гаманців можуть бути складними для користувачів. Дуже важливо мати належні механізми резервного копіювання, щоб запобігти втраті приватних ключів і коштів. Однак відповідальність за безпечне керування резервними копіями лежить на користувачах, і будь-яке неправильне поводження з резервними копіями може призвести до незворотних втрат [5, с. 4].

- Централізовані компоненти: Деякі криптогаманці можуть покладатися на централізовані компоненти, такі як централізовані сервери або сторонні сервіси, для забезпечення певних функцій або можливостей. Це створює певну залежність і потенційні єдині точки відмови.

- Відповідність нормативним вимогам: Оскільки криптовалютне законодавство продовжує розвиватися, гаманці можуть зіткнутися з проблемами при адаптації до

вимог законодавства. Деяким гаманцям може знадобитися накласти обмеження або впровадити процедури KYC, щоб відповідати нормативній базі, що потенційно обмежує конфіденційність і анонімність користувачів [6; 7; 8].

Таблиця 1.2

Недоліки криптогаманців

Криптогаманець	**Недоліки та обмеження**
MetaMask	Відсутність власної підтримки інших блокчейн-мереж, що обмежує сумісність з криптовалютами, які не базуються на Ethereum. Обмежена офлайн-функціональність, що вимагає підключення до Інтернету для більшості операцій. Відносно висока плата за газ у періоди перевантаження мережі
Trust Wallet	Для певних функцій необхідна перевірка KYC, що може обмежувати конфіденційност. Обмежений контроль над комісією за транзакції, що може призвести до збільшення витрат під час перевантаження мережі.
Exodus Wallet	Обмежена сумісність з апаратними гаманцями інших виробників, що обмежує користувачів з певними апаратними пристроями. Потенційні проблеми з часом відгуку служби підтримки в пікові періоди.
Atomic Wallet	Відносно вища комісія за транзакції порівняно з деякими іншими гаманцями на ринку. Відсутність підтримки певних сучасних DeFi-протоколів або функцій.
Ledger Live	Обмежена підтримка певних криптовалют або токенів, що змушує користувачів покладатися на альтернативні гаманці для управління певними активами. Можливі затримки в додаванні підтримки нових криптовалют.

1.4 Види та аналіз криптоатак у криптопросторі

Криптоатаки стають все більш витонченими і становлять значну загрозу для криптопростору. Фішингові атаки залишаються поширеними, коли зловмисники використовують оманливу тактику, щоб обманом змусити користувачів розкрити

конфіденційну інформацію. Ці атаки використовують людські вразливості та покладаються на методи соціальної інженерії.

Зломи бірж призвели до значних збитків, підкресливши вразливість централізованих криптовалютних бірж. Слабкі заходи безпеки та неадекватні практики зберігання були використані, що призвело до крадіжки коштів користувачів. Важливість проведення ретельного аудиту безпеки та впровадження надійних протоколів безпеки неможливо переоцінити.

Криптогаманці є ще однією мішенню для зловмисників. Несанкціонований доступ до гаманців через вразливості програмного забезпечення, скомпрометовані пристрої або прямі атаки на інфраструктуру гаманців може призвести до крадіжки коштів. Впровадження надійного захисту парольних фраз, безпечної генерації випадкових чисел та використання надійних апаратних гаманців може підвищити безпеку гаманців [9, с. 105].

Смарт-контракти, що рекламуються як децентралізовані та незмінні, також піддаються атакам. Використання вразливостей коду або логічних помилок в смарт-контрактах може дозволити зловмисникам маніпулювати або викрадати кошти, заблоковані в контракті. Аудит коду смарт-контрактів та проведення комплексних оцінок безпеки мають важливе значення для запобігання таким зловживанням.

Підміна SIM-карт стала помітною загрозою, що дозволяє обійти двофакторну автентифікацію (2FA) і отримати несанкціонований доступ до криптовалютних рахунків. Зловмисники переконують операторів мобільного зв'язку перенести номер телефону жертви на нову SIM-карту, що надає їм контроль над рахунками жертви. Впровадження альтернативних форм 2FA та підвищення обізнаності про заміну SIM-карт може зменшити цей ризик.

Фінансові піраміди та шахрайство з виведенням коштів продовжують обманювати інвесторів, які нічого не підозрюють. Шахрайські проекти обіцяють нереальні прибутки від інвестицій, зрештою руйнуються і призводять до фінансових втрат. Щоб не стати жертвою таких схем, необхідно проводити ретельну юридичну перевірку, вивчати проектні команди та проявляти обережність при інвестуванні.

Атаки на блокчейн та криптогаманці використовують вразливості в програмному або апаратному забезпеченні, що використовується в криптоекосистемі. Зловмисники впроваджують шкідливе програмне забезпечення в широко використовувані додатки гаманців або втручаються в апаратні гаманці під час виробничого процесу, ставлячи під загрозу безпеку коштів користувачів. Ретельна перевірка постачальників програмного забезпечення та використання надійних апаратних гаманців можуть зменшити ці ризики [10].

Перехоплення DNS передбачає маніпуляції з системою доменних імен (DNS) з метою перенаправлення користувачів на шахрайські веб-сайти або перехоплення їхніх комунікацій. Перенаправляючи користувачів на фальшиві веб-сайти, зловмисники можуть обманом змусити користувачів розкрити конфіденційну інформацію або отримати несанкціонований доступ до їхніх криптографічних рахунків. Перевірка автентичності веб-сайтів і використання захищених каналів зв'язку є важливими превентивними заходами.

Для боротьби з крипто-атаками крипто-спільнота та зацікавлені сторони галузі повинні зробити безпеку пріоритетом. Впровадження надійних заходів безпеки, інформування користувачів про потенційні ризики та просування найкращих практик є важливими кроками. Регулярне оновлення програмного забезпечення, дотримання надійних практик управління паролями та пильність щодо потенційних загроз є невід'ємною частиною підтримки безпечного криптосередовища [11].

Ось приклади найвідоміших зломів за допомогою успішних криптоатак:

- Злом Mt. Gox яка колись була найбільшою біржею біткоїнів у світі. У 2014 році вона зазнала масштабного злому, в результаті якого було втрачено близько 850 000 біткоїнів, що на той час коштувало приблизно 450 мільйонів доларів. Злом був пов'язаний з поганими практиками безпеки і відсутністю належного аудиту. Вважається, що хакери використовували гнучкість транзакцій, відому на той час проблему в протоколі Bitcoin, щоб маніпулювати ідентифікаторами транзакцій і багаторазово виводити кошти.

- Злом Bitfinex, це популярна криптовалютна біржа, яка була зламана в серпні 2016 року, що призвело до крадіжки 120 000 біткоїнів на суму близько $72 млн на той

час. Зловмисники скористалися вразливістю в системі гаманців з мультипідписами біржі. Bitfinex співпрацювала з BitGo, стороннім постачальником послуг безпеки, щоб впровадити рішення для гаманців з мультипідписом. Однак хакерам вдалося обійти заходи безпеки і отримати доступ до приватних ключів, що дозволило їм викрасти кошти.

- Злом Coincheck, японської криптовалютної біржи, яка була зламана в січні 2018 року, в результаті чого було викрадено 523 мільйони токенів NEM на суму близько $534 мільйонів на той час. Злом був пов'язаний з використанням біржею гарячого гаманця для зберігання великої кількості токенів NEM. Гарячі гаманці підключені до інтернету і є більш вразливими до зломів порівняно з холодними гаманцями, які є офлайновими рішеннями для зберігання коштів. Хакерам вдалося проникнути в "гарячий гаманець" і перевести кошти на власні гаманці.

- Злом Crypto.com, централізована біржа зазнала збитків у розмірі 35 мільйонів доларів США в результаті криптовалютного злому 17 січня 2022 року. Точна причина злому не була названа, але він служить нагадуванням про важливість надійних заходів безпеки в криптопросторі.

- Harmony Bridge Hack, це крос-ланцюговий міст, який був атакований, що призвело до збитків на суму 100 мільйонів доларів. Причиною злому назвали групу Lazarus, яка має зв'язки з Північною Кореєю. Зловмисники використали вразливість у крос-ланцюговому мості, що дозволило їм викрасти кошти [12; 13].

1.5 Порівняння Web3 та Web2 простору

Web3 являє собою фундаментальну зміну в тому, як працює Інтернет, порівняно з його попередником, Web2 (таблиця 1.3). У Web3 основна увага приділяється децентралізації, коли влада розподіляється між учасниками, а не зосереджується в руках кількох централізованих організацій. Така децентралізація стала можливою завдяки використанню технології блокчейн, яка забезпечує прозорий і незмінний реєстр для запису транзакцій і взаємодій [14].

Однією з ключових відмінностей між Web3 і Web2 є концепція власності та контролю користувача. У Web3 користувачі мають більший контроль над своїми даними та цифровими активами. Вони володіють криптографічними ключами, які надають їм доступ до своїх активів і дозволяють безпосередньо взаємодіяти з децентралізованими додатками (dApps). Така власність і контроль розширюють можливості користувачів, оскільки вони більше не залежать від централізованих платформ для управління та монетизації своїх даних.

Довіра та прозорість також є центральними для Web3. Технологія блокчейн забезпечує прозорість, реєструючи всі транзакції та взаємодії в публічному реєстрі, який може перевірити будь-хто. Така прозорість зменшує потребу довіряти централізованим посередникам, оскільки цілісність системи підтримується за допомогою механізмів консенсусу. Web2, з іншого боку, покладається на довіру до централізованих платформ, де користувачі часто мають обмежену видимість того, як використовуються їхні дані та як ними керують.

Інтероперабельність - ще один ключовий аспект Web3, спрямований на створення безперешкодної взаємодії між різними мережами і протоколами блокчейну, що дозволяє користувачам і додаткам взаємодіяти з різними платформами. Ця інтероперабельність дозволяє передавати активи та інформацію між різними системами, сприяючи створенню більш пов'язаної і гнучкої екосистеми. У Web2 інтероперабельність обмежена, а дані, як правило, ізольовані в межах окремих платформ.

Програмованість є визначальною рисою Web3. Смарт-контракти, засновані на технології блокчейн, дозволяють виконувати заздалегідь визначені правила та угоди в децентралізований і автоматизований спосіб. Така програмованість уможливлює розробку складних dApps та автоматизацію різних процесів, пропонуючи нові можливості для інновацій та ефективності. На противагу цьому, додаткам Web2 бракує такого ж рівня програмованості та автоматизації.

Спільне управління - це значний відхід від централізованих моделей управління Web2. У Web3 рішення приймаються колективно через механізми консенсусу, а учасники мають право голосу у визначенні напряму та функціонування системи.

Децентралізовані автономні організації (DAO) є прикладом такого підходу, керованого громадою, де зацікавлені сторони можуть голосувати за пропозиції та впливати на прийняття рішень. Платформи Web2, з іншого боку, зазвичай керуються центральним органом або власником платформи.

Конфіденційність є ще одним важливим аспектом у Web3. З підвищенням обізнаності про проблеми конфіденційності даних, Web3 прагне надати користувачам більше контролю над їхньою особистою інформацією. Методи шифрування та контрольовані користувачем механізми обміну даними дозволяють користувачам визначати, якою мірою їхні дані будуть доступні іншим особам. З іншого боку, платформи Web2 часто збирають і використовують дані користувачів для цільової реклами та монетизації, що викликає занепокоєння щодо конфіденційності та безпеки даних.

На закінчення, перехід від Web2 до Web3 являє собою зміну парадигми функціонування інтернету. Web3 запроваджує децентралізацію, власність і контроль користувачів, довіру і прозорість, інтероперабельність, програмованість, громадське управління і конфіденційність як основні принципи. Ці досягнення пропонують користувачам більший контроль над своїми даними та активами, сприяють прозорості та довірі через децентралізовані механізми, а також сприяють створенню більш інклюзивної та спільної інтернет-екосистеми.

Таблиця 1.3

Порівняння інтернет просторів

Порівняння	Web2	Web3
Централізація, децентралізація	Веб2 характеризується централізованими платформами, де кілька великих компаній контролюють більшість користувацьких даних і сервісів. Така централізація може призвести до проблем з конфіденційністю, правом власності на дані та цензурою.	Web3 має на меті створити децентралізований Інтернет, де користувачі мають більше контролю над своїми даними, а послуги надаються через децентралізовані платформи. Це може призвести до підвищення рівня приватності, права власності на дані та стійкості до цензури.

Продовження табл. 1.3

Порівняння	Web2	Web3
Право власності на дані	У Web2 користувачі часто не володіють своїми даними, оскільки вони зберігаються і контролюються централізованими платформами. Це може призвести до порушення конфіденційності та відсутності контролю над особистою інформацією.	У Web3 користувачі мають більше контролю над своїми даними, оскільки вони зберігаються на децентралізованих платформах. Це може призвести до підвищення рівня конфіденційності та володіння даними.
Монетизація	Монетизація в Web2 в першу чергу залежить від реклами та збору даних, що може призвести до нав'язливої реклами та проблем з конфіденційністю.	Web3 впроваджує нові моделі монетизації, такі як економіка на основі токенів і децентралізовані фінанси (DeFi), які можуть надати користувачам більше контролю над своїми активами і потенційно зменшити залежність від реклами.
Довіра та безпека	Довіра до Web2 залежить від централізованої влади, яка може бути вразливою до хакерських атак, витоків даних та цензури.	Web3 використовує технологію блокчейн і криптографічні методи для створення надійних, безпечних систем, які не залежать від централізованих органів влади. Це може призвести до підвищення безпеки та стійкості до цензури.
Технології та інфраструктура	Web2 побудований на традиційній архітектурі клієнт-сервер, з централізованими серверами, що надають послуги користувачам.	Web3 побудований на децентралізованих технологіях, таких як блокчейн і однорангові мережі, які можуть забезпечити більш стійку і розподілену інфраструктуру.

Висновки за розділом 1

Поточний стан криптогаманців у Web3-просторі характеризується швидким зростанням та інноваціями. За останні роки кількість криптогаманців, що використовуються, зросла в геометричній прогресії, і зараз існує велика кількість різноманітних гаманців на вибір.

Це зростання зумовлене низкою факторів, зокрема, все більшим прийняттям криптовалют підприємствами та установами, зростанням популярності децентралізованих фінансів (DeFi) та підвищенням рівня обізнаності про потенційні переваги технології блокчейн.

Зростання популярності та поширення криптовалют приваблює зловмисників, які намагаються використати вразливості гаманців, бірж та смарт-контрактів. Ці атаки часто спрямовані на користувачів, які нічого не підозрюють, користуючись їхньою необізнаністю або недостатньою безпекою.

Щоб зменшити ці ризики, користувачам і платформам вкрай важливо визначити пріоритетність заходів безпеки. Це включає впровадження надійних механізмів автентифікації, навчання користувачів найкращим практикам, регулярне оновлення програмного забезпечення та прошивки, проведення ретельного аудиту коду, а також використання шифрування та криптографічних протоколів для захисту конфіденційної інформації.

Крім того, співпраця між зацікавленими сторонами галузі, включаючи розробників, дослідників безпеки та регуляторні органи, має важливе значення для того, щоб випереджати нові загрози. Обмін знаннями, передовим досвідом та інформацією про загрози може сприяти створенню більш безпечного середовища для користувачів і розробці надійних рішень для захисту.

Поки криптопростір продовжує розвиватися, користувачам і платформам вкрай важливо зберігати пильність, адаптуватися до нових загроз і постійно вдосконалювати заходи безпеки.

Незважаючи на це зростання, все ще існує низка проблем, які необхідно вирішити, перш ніж криптовалютні гаманці стануть загальноприйнятими. Ці

проблеми включають безпеку криптогаманців, відсутність регулювання та складність використання криптогаманців.

Коротко про ключові функції, які користувачі шукають в криптогаманцях:

Безпека: криптогаманці – це, по суті, цифрові сховища для зберігання криптовалюти та NFT токенів, тому дуже важливо, щоб вони були безпечними, адже користувачі хочуть бути впевненими, що їхні кошти знаходяться в безпеці, і що вони зможуть отримати до них доступ, коли їм це буде потрібно.

Простота використання: криптогаманці можуть бути складними у використанні, особливо для тих, хто не знайомий зі світом криптовалют, тому користувачі хочуть мати гаманець, який простий у використанні та розумінні, і який не вимагає багато технічних знань для роботи.

Функції: криптогаманці мають різноманітні функції, такі як можливість надсилати та отримувати криптовалюту, зберігати декілька криптовалют та брати участь у DeFi-додатках. Користувачі хочуть мати гаманець, який пропонує необхідні їм функції і який постійно оновлюється новими можливостями.

РОЗДІЛ 2
ОСНОВНІ ЗАГРОЗИ І МЕТОДИ ЇХ УСУНЕННЯ В КРИПТОГАМАНЦЯХ

2.1 Вивчення потреб користувачів Web3 простору

Вивчення потреб користувачів Web3 має вирішальне значення для розуміння їхніх уподобань, проблем та очікувань у децентралізованій веб-екосистемі. Визначивши та задовольнивши ці потреби, розробники, платформи та постачальники послуг можуть створювати більш орієнтовані на користувача рішення, які задовольнятимуть мінливі потреби користувачів Web3. Нижче описані ключові потреби користувачів Web3:

Безпека і конфіденційність. Користувачі Web3 надають великого значення безпеці та конфіденційності. Вони прагнуть надійних заходів безпеки, включаючи безпечне управління ключами, шифрування та захист від злому або несанкціонованого доступу. Функції, орієнтовані на конфіденційність, такі як анонімність, контроль даних і децентралізоване управління ідентифікацією, також є важливими для користувачів Web3.

Зручні інтерфейси. Користувачі Web3 прагнуть мати інтуїтивно зрозумілі та зручні інтерфейси, які полегшують взаємодію з децентралізованими додатками (dApps) та управління своїми цифровими активами. Спрощені процеси адаптації, чітка навігація та безперешкодний користувацький досвід є важливими факторами для ширшого впровадження технологій Web3.

Інтероперабельність та міжланцюгова підтримка. З поширенням різноманітних мереж блокчейн, користувачі Web3 часто потребують інтероперабельності та підтримки міжмережевих ланцюжків. Вони хочуть мати можливість безперешкодно передавати і використовувати активи між різними блокчейнами і отримувати доступ до більш широкого спектру децентралізованих послуг без обмежень і складнощів.

Масштабованість і швидкість транзакцій. З розвитком Web3 користувачі очікують масштабованих рішень, здатних обробляти більший обсяг транзакцій і

забезпечувати більш швидкий час підтвердження. Попитом користуються рішення, які вирішують проблеми перевантаженості мережі і високих тарифів на газ, зберігаючи при цьому ефективність і завершеність транзакцій.

Можливості децентралізованих фінансів (DeFi). Користувачі Web3, особливо ті, хто займається DeFi, шукають гаманці та платформи, які пропонують комплексну підтримку протоколів децентралізованого фінансування. Це включає в себе такі функції, як вирощування врожаю, забезпечення ліквідності, кредитування, запозичення і свопи токенів.

Спільнота та управління. Користувачі Web3 цінують активні та залучені спільноти навколо проектів і платформ. Вони цінують прозорі моделі управління, залучення спільноти до процесів прийняття рішень та можливість брати участь у формуванні напрямку проектів Web3.

Освіта та підтримка. Користувачі Web3 часто потребують освітніх ресурсів, навчальних посібників та чуйної підтримки клієнтів, щоб розібратися в складнощах децентралізованих технологій. Доступ до документації, поширених запитань та онлайн-спільнот може допомогти користувачам долати труднощі та приймати обґрунтовані рішення.

Для подальшого розгляду цієї теми нам потрібні дані, що саме користувачі Web3 проектів вважають найважливішою потребою. В процесі пошуку ми не знайшли досліджень на цю тему, тому прийняли рішення провести самостійне дослідження.

Мета дослідження: Вивчення потреб користувачів Web3 простору.

Завдання дослідження:

- Вивчення потреб користувачів Web3 простору, розуміння їхніх уподобань, проблем та очікувань у децентралізованій веб-екосистемі.

- **Досліджувані:** у дослідженні брали участь 30 користувачів Web3 простору, 23 чоловіка та 7 жінок. Вік досліджуваних від 18 до 35 років.

Методи дослідження. Для одержання повноцінних і достовірних результатів використалися методи : тестування, статистичний аналіз отриманих даних.

Хід дослідження: Дослідження проводилось у вигляді тесту на основі платформи «Google форма». Досліджуваним задали питання «Що для вас є ключовою потребою при користуванні Web3 простором» та на вибір давалось 7 варіантів відповіді:

А) Безпека і конфіденційність.

Б) Зручні інтерфейси.

В) Інтероперабельність та міжланцюгова підтримка.

Д) Масштабованість і швидкість транзакцій.

Е) Можливості децентралізованих фінансів (DeFi).

Ж) Спільнота та управління.

З) Освіта та підтримка.

За результатами опитування ДОДАТОК А:

Варіант А) Безпека і конфіденційність обрали 18 досліджуваних;

Варіант Б) Зручні інтерфейси - 4 досліджуваних;

Варіант В) Інтероперабельність; міжланцюгова підтримка - 2 досліджуваних;

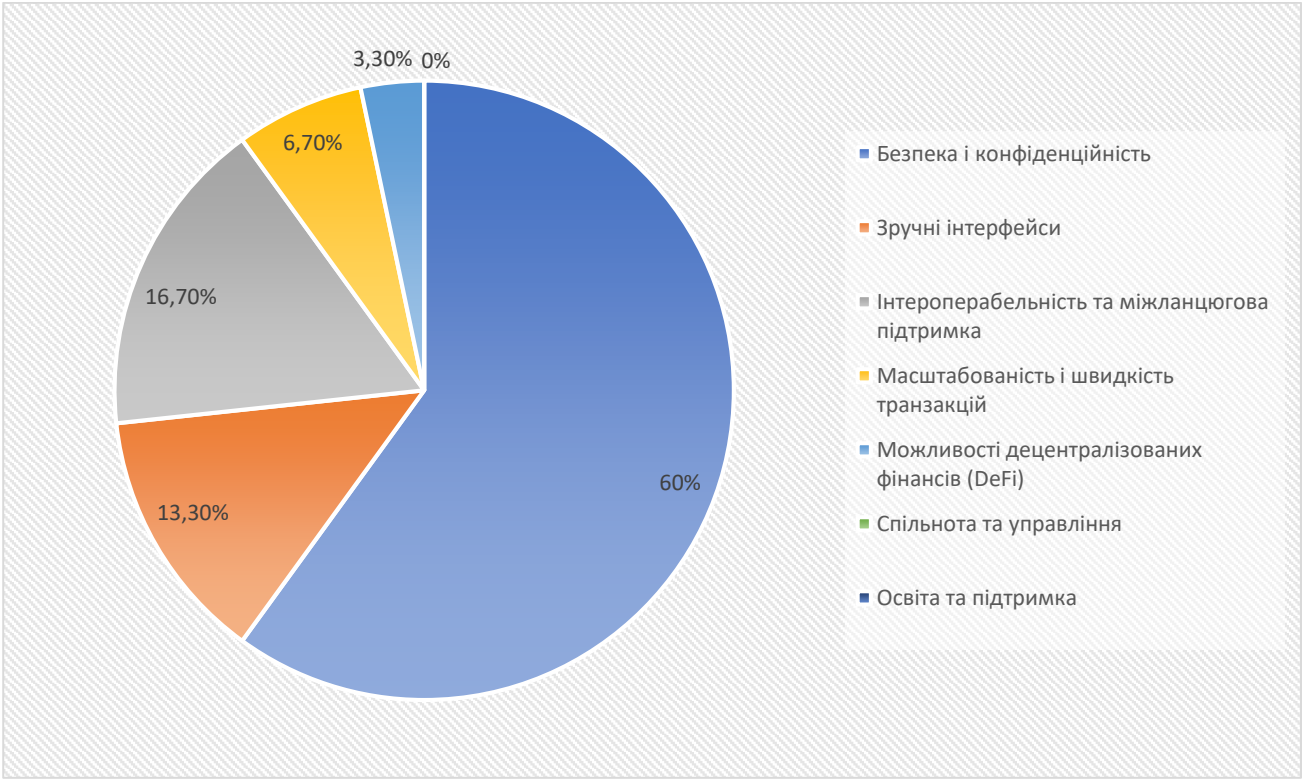
Варіант Д) Масштабованість і швидкість транзакцій - 5 досліджуваних;

Варіант Е) Можливості децентралізованих фінансів (DeFi) - 1 досліджуваний;

Варіанти Ж) Спільнота та управління та З) Освіта та підтримка - обрали 0 досліджуваних.

Таблиця 2.1

Потреби користувачів Web3 простору



А) Безпека і конфіденційність — 60%, Б) Зручні інтерфейси — 13,3%, В) Інтероперабельність та міжланцюгова підтримка — 16,7%, Д) Масштабованість і швидкість транзакцій — 6,7%, Е) Можливості децентралізованих фінансів (DeFi) — 3,3%, Ж) Спільнота та управління — 0%, З) Освіта та підтримка — 0%

Проаналізуваши відповіді (таблиця 2.1) ми можемо дійти висновку, що найважливішою потребою для користувачів Web3 простору є Безпека та Конфіденційність, її обрали більшість досліджуваних. Також важливими виявились Зручний інтерфейс та інтероперабельність та міжланцюгова підтримка. Можливість децентралізованих фінансів, спільнота та управління, та освіта й підтримка — майже не обирались. Але ми не вважаємо їх менш значущими потребами, проте вони не є першорядними.

Спираючись на результати дослідження ми будемо рухатись в напрямку покращення досвіду Web3 користувачів задовольняючи їх потребу в безпеці та конфіденційності.

2.2 Визначення функціональних та безпекових вимог до криптогаманця

Визначення функціональних вимог та вимог безпеки до криптогаманця має вирішальне значення для забезпечення його ефективності, зручності використання та захисту активів користувачів, адже виконання цих вимог забезпечить легкий вхід для нових користувачів [15, с. 2].

Функції криптогаманця мають відповідати таким мінімальним критеріям:

- Створення гаманців

▪Користувачі повинні мати можливість легко створювати новий гаманець за допомогою безпечного та інтуїтивно зрозумілого процесу, включаючи генерацію унікального приватного ключа та мнемонічної фрази.

- Управління гаманцем

▪Гаманець повинен дозволяти користувачам переглядати та керувати своїми криптовалютними активами, включаючи баланс, історію транзакцій та управління токенами.

- Підтримка криптовалют

▪Гаманець повинен підтримувати широкий спектр криптовалют і токенів, дозволяючи користувачам керувати різними цифровими активами в єдиному інтерфейсі.

- Функціональність транзакцій

▪Гаманець має підтримувати безпечні та безперебійні транзакції, включаючи відправлення, отримання та запит криптовалюти, а також можливість встановлювати тарифи та швидкість транзакцій.

- Інтеграція з блокчейнами

▪Гаманець повинен підтримувати кілька мереж блокчейн, що дозволить користувачам взаємодіяти з різними криптовалютами і децентралізованими додатками (DApps) в різних ланцюжках, але це не обов’язково.

- Взаємодія зі смарт-контрактами

▪Якщо гаманець підтримує функціонал смарт-контрактів, користувачі повинні мати можливість легко і безпечно взаємодіяти зі смарт-контрактами та здійснювати транзакції за їх допомогою.

- Зручний інтерфейс

▪Гаманець повинен мати інтуїтивно зрозумілий і зручний інтерфейс, що забезпечує чітку навігацію, прості для розуміння функції і корисні підказки або інструкції.

Вимоги з безпеки мають включати в себе надійні рішення, котрі можуть забезпечити найкращий захист на ринку власності користувача. Вони включають у себе:

- Захист приватних ключів

▪Гаманець повинен впроваджувати надійні заходи для безпечного зберігання та захисту приватних ключів користувачів, такі як шифрування, апаратне зберігання ключів або безпечні анклави.

- Двофакторна автентифікація (2FA)

▪Надання додаткової функції 2FA додає додатковий рівень безпеки, вимагаючи додаткового етапу перевірки для доступу до гаманця або схвалення транзакцій.

- Біометрична автентифікація

▪Включення біометричної автентифікації, наприклад, відбитків пальців або розпізнавання обличчя, може підвищити безпеку гаманця та покращити користувацький досвід.

- Безпечний зв'язок

▪Гаманець повинен використовувати безпечні протоколи зв'язку, такі як HTTPS, для захисту даних користувача і транзакцій під час спілкування із зовнішніми сервісами.

- Аудит коду та тестування безпеки

▪Регулярний аудит коду, оцінка безпеки та тестування на проникнення повинні проводитися для виявлення та усунення вразливостей і забезпечення стійкості гаманця до потенційних атак.

- Регулярні оновлення та виправлення

▪Гаманець повинен мати механізм для своєчасного оновлення програмного забезпечення та патчів безпеки для усунення будь-яких виявлених вразливостей або експлойтів.

- Резервне копіювання та відновлення

▪Впровадження безпечного та зручного для користувача процесу резервного копіювання та відновлення, включаючи мнемонічні фрази для резервного копіювання та чіткі інструкції, допомагає користувачам захистити свої кошти у разі втрати гаманця або виходу пристрою з ладу.

- Навчання та сповіщення з питань безпеки

▪Гаманець повинен надавати освітні ресурси про найкращі практики безпеки та сповіщати користувачів про потенційні ризики безпеки, такі як спроби фішингу або підозрілі транзакції, а також надсилати попередження або повідомлення [3, с. 114].

2.3 Врахування зручності використання та інтерфейсу

При розробці успішного криптогаманця важливо враховувати зручність використання та інтерфейс. Мета полягає в тому, щоб забезпечити користувачам інтуїтивно зрозумілий і безперешкодний досвід роботи. Це означає, що інтерфейс гаманця має бути зручним для навігації, з чіткою структурою та ієрархією інформації. Він повинен адаптуватися до різних пристроїв і розмірів екранів, забезпечуючи адаптивний дизайн, який оптимізує взаємодію з користувачем. При виборі типографіки пріоритетом має бути читабельність, розбірливі шрифти та відповідні розміри [16].

З огляду на те, що криптогаманці містять цінні цифрові активи, безпека має першорядне значення. Гаманці повинні пропонувати такі функції, як 2-факторна автентифікація, біометрія (відбитки пальців/ідентифікатор особи), захист паролем і

автоматичний вихід з системи після періодів неактивності. Користувачі також повинні мати можливість створювати резервні копії та відновлювати свої гаманці за потреби. Розробка простих і безпечних методів керування ключами, включаючи можливість зберігання ключів в зашифрованому вигляді, резервне копіювання та відновлення ключів.

Гаманець повинен бути сумісним з усіма основними операційними системами, такими як Windows, MacOS, Android та iOS. Він також повинен підтримувати широкий спектр криптовалют, а не лише одну чи дві. Чим більше монет і токенів підтримується, тим краще для більшості користувачів.

Послідовність у мові дизайну має вирішальне значення для формування у користувача звички до нього. Інтерфейс повинен використовувати єдиний візуальний стиль, включаючи узгоджені колірні схеми, іконки та візуальні елементи. Така узгодженість покращує розуміння користувача та навігацію всередині гаманця.

Важливим є забезпечення чіткого та своєчасного зворотного зв'язку з користувачами. Оновлення статусу транзакції, повідомлення про помилки та підказки повинні передаватися ефективно, допомагаючи користувачам розуміти результати своїх дій і запобігаючи плутанині або розчаруванню.

Відкритий вихідний код. Для багатьох користувачів криптовалют найбільш привабливими є гаманці з відкритим вихідним кодом і прозорим кодом. Гаманці з відкритим кодом дозволяють спільноті проводити аудит коду і перевіряти його на наявність вразливостей. Це допомагає зміцнити довіру до безпеки гаманця.

Криптогаманці потрібно регулярно оновлювати, щоб виправляти будь-які проблеми з безпекою, додавати нові функції та підтримувати найновіші криптовалюти. Розробники гаманців повинні випускати оновлення на постійній основі, щоб надати користувачам найновіші та найбезпечніші можливості.

Опції кастомізації дозволяють користувачам пристосовувати інтерфейс гаманця до своїх уподобань. Це може включати вибір колірних тем, бажаних налаштувань відображення або розташування елементів відповідно до індивідуальних потреб. Персоналізація сприяє формуванню почуття причетності та покращує загальний користувацький досвід.

Початкові навчальні матеріали, надання посібників, відео-туторіалів для допомоги користувачам ознайомитися з функціями та можливостями криптогаманця.

Загалом зручність використання та інтерфейс криптогаманця мають зацікавити широке коло користувачів, які можуть варіюватися за рівнем технічної освіти та досвіду. Забезпечення легкості використання та зрозумілості інтерфейсу допоможе залучити більше людей до використання криптовалют та блокчейн-технологій.

Включення в інтерфейс гаманця легкодоступних довідкових ресурсів, таких як поширені запитання, посібники користувача та контактна інформація служби підтримки, допомагає користувачам вирішувати проблеми або знаходити відповіді на свої запитання без необхідності залишати середовище гаманця. Швидкий доступ до допомоги сприяє підвищенню задоволеності користувачів та їхньої впевненості у використанні гаманця.

Розробка інтерфейсу гаманця з урахуванням вимог доступності забезпечує інклюзивність для користувачів з обмеженими можливостями. Дотримання рекомендацій з веб-доступності забезпечує сумісність з допоміжними технологіями та альтернативними методами введення, гарантуючи безперебійну роботу для всіх користувачів.

При розробці інтерфейсу криптогаманця слід також враховувати візуальну привабливість. Це може включати використання зручних кольорових схем, приємних для очей шрифтів, привабливого дизайну та анімацій, що покращують загальний візуальний досвід користувача.

Юзабіліті-тестування є життєво важливим для збору відгуків та визначення сфер для покращення. Залучаючи репрезентативних користувачів, можна виявити проблеми юзабіліті, що дозволяє ітеративно вдосконалювати інтерфейс. Такий ітеративний підхід з часом підвищує зручність використання гаманця та задоволеність користувачів.

Врахування цих факторів і постійне вдосконалення юзабіліті та інтерфейсу криптогаманця сприятиме позитивному користувацькому досвіду, що призведе до підвищення рівня прийняття та задоволеності серед користувачів.

Висновки за розділом 2

Отже, безпека криптогаманців має першорядне значення в криптопросторі. Для фахівця з кібербезпеки дуже важливо розуміти і протидіяти основним загрозам, з якими стикаються криптогаманці, щоб забезпечити захист цифрових активів користувачів.

Однією з основних загроз є ризик несанкціонованого доступу до гаманців. Це може відбуватися різними способами, включаючи фішингові атаки, шкідливе програмне забезпечення та соціальну інженерію. Щоб зменшити цей ризик, важливо навчати користувачів найкращим практикам, таким як використання надійних, унікальних паролів, двофакторна автентифікація та обережне ставлення до підозрілих посилань і повідомлень.

Ще однією значною загрозою є компрометація приватних ключів або ключових фраз. Захист цих важливих облікових даних є життєво необхідним для запобігання несанкціонованому доступу та втраті коштів. Заохочення користувачів до використання апаратних гаманців, які зберігають приватні ключі в автономному режимі, або впровадження схем мультипідпису може підвищити безпеку їхніх гаманців.

Для ефективної боротьби з цими загрозами слід застосовувати кілька ключових методів. Освіта та обізнаність користувачів відіграють життєво важливу роль, допомагаючи їм розпізнавати та уникати пасток у сфері безпеки. Поширення найкращих практик, таких як надійне управління паролями, обізнаність щодо фішингу та перевірка автентичності веб-сайтів, може значно знизити рівень успішності атак.

Багатофакторна автентифікація забезпечує додатковий рівень захисту, вимагаючи від користувачів надання додаткових факторів перевірки, таких як біометричні дані або одноразові паролі, для доступу до криптоакаунтів. Це знижує ризик несанкціонованого доступу, навіть якщо облікові дані для входу скомпрометовані.

Вибирайте надійні та безпечні гаманці та біржі. Для захисту коштів користувачів і особистої інформації слід використовувати надійне шифрування, безпечне зберігання ключів і регулярні перевірки безпеки. Апаратні гаманці, які зберігають приватні ключі в автономному режимі, забезпечують підвищений захист від онлайн-загроз.

Регулярне оновлення програмного забезпечення та управління виправленнями мають вирішальне значення для усунення відомих вразливостей і захисту від нових загроз. Використання захищених протоколів зв'язку, таких як HTTPS, гарантує, що передача даних зашифрована і захищена від підслуховування.

Для виявлення та усунення вразливостей у криптосистемах слід регулярно проводити аудит безпеки та тестування на проникнення. Планування реагування на інциденти та відновлення також має важливе значення для ефективного реагування на порушення безпеки, включаючи локалізацію, розслідування та своєчасне відновлення.

Вирішення проблеми шахрайських або шкідливих додатків, які видають себе за легальні гаманці, має вирішальне значення. Користувачі повинні знати, як перевіряти автентичність додатків гаманців і завантажувати їх з офіційних джерел. Впровадження надійних процесів перевірки магазинів додатків і використання систем оцінки репутації також може допомогти знизити ризик появи фальшивих гаманців.

Крім того, значну загрозу становлять атаки соціальної інженерії, спрямовані безпосередньо на користувачів, такі як підміна SIM-карт або видача себе за іншу особу. Впровадження надійних процесів перевірки особистих даних, заохочення користувачів до використання безпечних каналів зв'язку та підвищення обізнаності про ці типи атак може допомогти зменшити ризик.

В цілому, криптоіндустрія повинна постійно адаптувати заходи безпеки до мінливого ландшафту загроз. Впроваджуючи ці методи усунення, індустрія може підвищити безпеку, захистити активи користувачів і створити більш безпечне середовище для процвітання криптовалют.

На закінчення, як фахівець з кібербезпеки в криптопросторі, важливо залишатися пильним і проактивним у виявленні та пом'якшенні загроз для

криптогаманців. Навчання користувачів, впровадження надійних заходів безпеки, регулярне оновлення програмного забезпечення та проведення ретельних оцінок безпеки є важливими кроками у забезпеченні безпеки та цілісності криптогаманців. Усуваючи ці загрози та впроваджуючи відповідні заходи безпеки, ми можемо підвищити довіру користувачів до криптоекосистеми.

РОЗДІЛ 3
РОЗРОБКА УДОСКОНАЛЕНОГО ПРОГРАМНОГО ЗАСОБУ КРИПТОГАМАНЦЯ

3.1 Огляд наявних криптогаманців, що можуть бути покращенні

Проаналізувавши інформацію з джерел та власного досвіду виділю пару найпопулярніших криптовалютних гаманців, кожен з яких має свій набір функцій та обмежень:

1. MetaMask
2. Trust Wallet
3. Ledger Live
4. MyEtherWallet (MEW)
5. Coinbase Wallet

Хоча ці гаманці широко використовуються і загалом є надійними, завжди є сфери, де їх можна покращити.

Гаманці завжди можуть виграти від покращення заходів безпеки, таких як підтримка мультипідпису, біометрична автентифікація та більш надійні методи шифрування.

Багато гаманців обмежені певними криптовалютами або блокчейнами. Покращення інтероперабельності між різними блокчейнами і токенами полегшить користувачам управління різноманітними криптоактивами в одному гаманці.

Спрощення користувальницького інтерфейсу і його інтуїтивна зрозумілість можуть допомогти залучити нових користувачів і полегшити існуючим користувачам навігацію в гаманці. Це включає кращу організацію функцій, чіткіші інструкції та зручніший дизайн.

Деякі гаманці є централізованими, тобто вони покладаються на одну компанію або сервер для управління даними користувачів. Децентралізація таких гаманців підвищить їхню стійкість до атак і зменшить ризик виникнення єдиної точки відмови.

Гаманці можуть підвищити рівень конфіденційності, впроваджуючи такі функції, як стелс-адреси, змішування монет і докази з нульовим рівнем знання. Ці функції допомагають захистити ідентифікаційні дані користувачів і деталі транзакцій від легкого відстеження.

Зі збільшенням кількості користувачів криптовалют гаманці повинні бути здатні обробляти більші обсяги транзакцій. Впровадження позаланцюгових рішень, таких як Lightning Network або сайдчейни, може допомогти гаманцям ефективніше масштабуватися.

Гаманці можна вдосконалити шляхом інтеграції з іншими сервісами, такими як платформи децентралізованих фінансів (DeFi), децентралізовані біржі (DEX) та інші додатки на основі блокчейну. Це дозволить користувачам отримати доступ до ширшого спектру послуг безпосередньо зі свого гаманця.

Саме MetaMask наразі є найпопулярнішим децентралізованим криптовалютним гаманцем який використовують як новачки так і досвідчені користувачі, адже він має дуже зручний інтерфейс та усі функції криптогаманця. Головним його недоліком є безпека, бо через стрімкий зріст криптосвіту з’являється все більше зловмисників які використовують те, що MetaMask не інформує своїх користувачів про шкідливі транзакції і майже кожен користувач стикався з цим гірким досвідом, коли він просто підтвердив транзакцію та втратив усі наявні кошти та токени з гаманця. Було прийнято рішення удосконалити безпеку застосунку MetaMask, щоб уникнути варіанту прийняття шкідливої транзакції.

Trust Wallet найпопулярніший децентралізований криптовалютний гаманець на пристроях на базі Android та iOS систем. Нещодавно був випущенний як доповнення до браузеру, але не набув такої популярності як і MetaMask. Має аналогічні проблеми в сфері безпеки як в MetaMask, а саме не інформує користувачів про шкідливі транзакції на мобільних пристроях.

Ledger Live найнадійниший криптовалютний гаманець, який не має недоліків в сфері безпеки. Це апаратний криптовалютний гаманець який спочатку потрібно купити, щоб використовувати. Не має шансів бути зламаним. Але нещодавно розробники анонсували майбутню функцію про можливість відновлення мнемонічної

фрази криптогаманця, яку в майбутньому зможуть запросити правоохоронці у випадку з підозрами, що ставить під питання безпеку та надійність цього криптогаманця і робить його централізованим.

MyEtherWallet (MEW) досить непопулярний та не дуже надійних криптовалютний гаманець, адже він знаходиться на веб-сторінці в браузері, а у разі відмови доступу до сайту ви не зможете використовувати саме цей гаманець, єдиним виходом з цієї ситуації буде відновлення гаманця за вашою мнемонічною фразою в іншому криптогаманці. Але до нього можна доєднатися з будь-якого криптовалютного гаманця та отримати увесь функціонал цього застосунку.

Coinbase Wallet дуже популярний централізований криптогаманець серед американських користувачів. Досить надійний та має багатофакторний захист. Але він керується компанією, що перечить основним правилам криптосвіту та у разі банкрутства компанії вона буде використовувати кошти користувачів. Має дуже високі комісії та обмежену підтримку криптовалют та мереж.

3.2 Розробка архітектури программи удосконалення криптогаманця

Безпека власності користувача в криптовалютному гаманці під назвою MetaMask була удосконалена за такою архітектурою:

1. Розробка функцій безпеки
2. Розробка інтерфейс користувача
3. Документація про використання

Майже усі блокчейни мають у собі можливість розгортання так званого смарт-контракту, який широко використовується в повсякденному житті користувачів Web3 світу.

Смарт-контракт – це цифровий контракт, що самостійно виконується, побудований на технології блокчейн. Це комп'ютерна програма або код, який автоматично виконує заздалегідь визначені дії або умови, як тільки виконуються задані критерії. Смарт-контракти призначені для полегшення, перевірки та

забезпечення виконання угод або транзакцій без необхідності залучення посередників, таких як банки або юридичні установи [17, с. 1754; 18, с. 35].

Смарт-контракти працюють на децентралізованих платформах, як правило, блокчейн-мережах, таких як Ethereum, і зберігаються та виконуються в розподіленій мережі комп'ютерів, забезпечуючи прозорість, безпеку та незмінність. Вони пишуться мовами програмування, спеціально пристосованими для розробки смарт-контрактів, такими як Solidity для Ethereum.

Смарт-контракти можуть представляти різні типи угод, від простих, як-от передача цифрових активів (криптовалют) між сторонами, до більш складних, що включають гаманці з декількома підписами, протоколи децентралізованих фінансів (DeFi), управління ланцюжками поставок або навіть системи голосування. Умови контракту, включаючи правила, зобов'язання і наслідки, закодовані безпосередньо в коді смарт-контракту [19, с. 2908].

Після розгортання в блокчейні смарт-контракти стають загальнодоступними, а їх виконання запускається транзакціями або певними подіями. Децентралізована природа блокчейну гарантує, що виконання смарт-контракту є прозорим, незмінним і захищеним від підробки.

Смарт-контракти мають потенціал для автоматизації та оптимізації багатьох традиційних процесів, зменшення потреби в посередниках, посилення безпеки та підвищення ефективності в різних галузях. Однак важливо зазначити, що хоча смарт-контракти є потужним інструментом, їх розробка та впровадження вимагають ретельного вивчення потенційних вразливостей та ретельного тестування для забезпечення їх надійності та безпеки.

Наразі функції смарт-контрактів посідають перше місце в рейтингу інструментів шахраїв, які дають змогу обдурити звичайних людей і відібрати їхні криптовалюти з NFT [20, с. 89].

Удосконалення безпеки криптовалютного гаманця стосується функції смарт-контракту під назвою SetApprovalforAll.

setApprovalForAll – це функція, яка зустрічається в смарт-контрактах на блокчейн-платформах, що підтримують стандарти токенів ERC-721 або ERC-1155,

таких як Ethereum. Вона використовується для надання або відкликання дозволу оператору на управління всіма токенами, що належать певній адресі. Простими словами – вона може надати зловмиснику асболютно повний доступ до всіх токенів на вашому криптовалютному гаманці. Єдиний випадок коли ця функція може використовуватися безпечно – це для продажі ваших токенів на маркетплейсі.

Функція SetApprovalforAll дуже популярна у сфері криптовалютного шахрайства. Тому важливо усвідомлювати, коли її безпечно схвалювати, а коли краще проігнорувати.

Тому задля забезпечення безпеки користувачів криптовалютного гаманця була розроблена функція яка отримує адресу гаманця, що запитав дозвіл на використання токенів та зрівнює її зі списком перевірених адрес, якщо вона збігається, тоді транзакція безпечна, адже наданий дозвіл буде використаний лише для продажу обраних токенів через маркетплейс, якщо ж збігів немає, то транзакція небезпечна адже дозвіл на використання токенів був запрошений зловмисником.

Для того щоб розробити функцію спочатку потрібно завантажити та запустити проект MetaMask який є Open Sourse, це дозволяє нам дивитися та змінювати повний код проекту, а також розроблювати корисні модифікації. Проект MetaMask є на веб-сервісі GitHub.

Для розробки була використана операційна система Linux Ubuntu та інструмент для редагування коду Visual Code.

Для успішної компіляції проекту та його запуску потрібні такі компоненти: Node.js 16, Yarn v3, Corepack, npm. Тестування розширення проводилось у браузері Firefox та Chrome.

Для виконання роботи знадобилися такі файли: confirm-approve-content-component.js; transaction.ts; messages.json.

Код функції яка знаходиться в файлі confirm-approve-content-component.js:

```
renderWarning() {
  const { t } = this.context;
  const {
```

```
  toAddress,
  isSetApproveForAll,
} = this.props;

let isSafeAddress = false;
Object.values(SafeWallets).forEach((walletAddress) => {
   if (walletAddress.toLowerCase() === toAddress.toLowerCase()) {
     isSafeAddress = true;
   }
});
```

Спочатку функція записує в поля властивості компоненту, як отримуються з контексту. Далі йде цикл який перевіряє поточний гаманець із заздалегідь заданими беспечними гаманцями. В кінці за допомогою умови ми отримуємо результат який дасть відповідь на те, чи безпечна транзакція. Повний код в додатку Б.

3.3 Реалізація покращенної безпеки користувача

Після запуску проекту MetaMask та написання коду функції захисту, потрібно розробити інтерфейс та після цього скомпілювати і використати модифікований проект.

Для демонстрації виконаної роботи були використані рисунки транзакцій та порівняні звичайна і модифікована версії між собою. Демонстративне робоче середовище на системі Linux та з веб-браузером Firefox, але розширення працює на будь-якій системі та сумісне з браузерами Chrome і Firefox.

Для того щоб продемонструвати результат було ініційовано виконання функції setApprovalForAll в двух випадках: з довіреною адресою та будь-якою іншою.

Перший варіант це безпечна транзакція. Порівняємо обидві транзакції які можна побачити на рисунку 3.1 та рисунку 3.2.

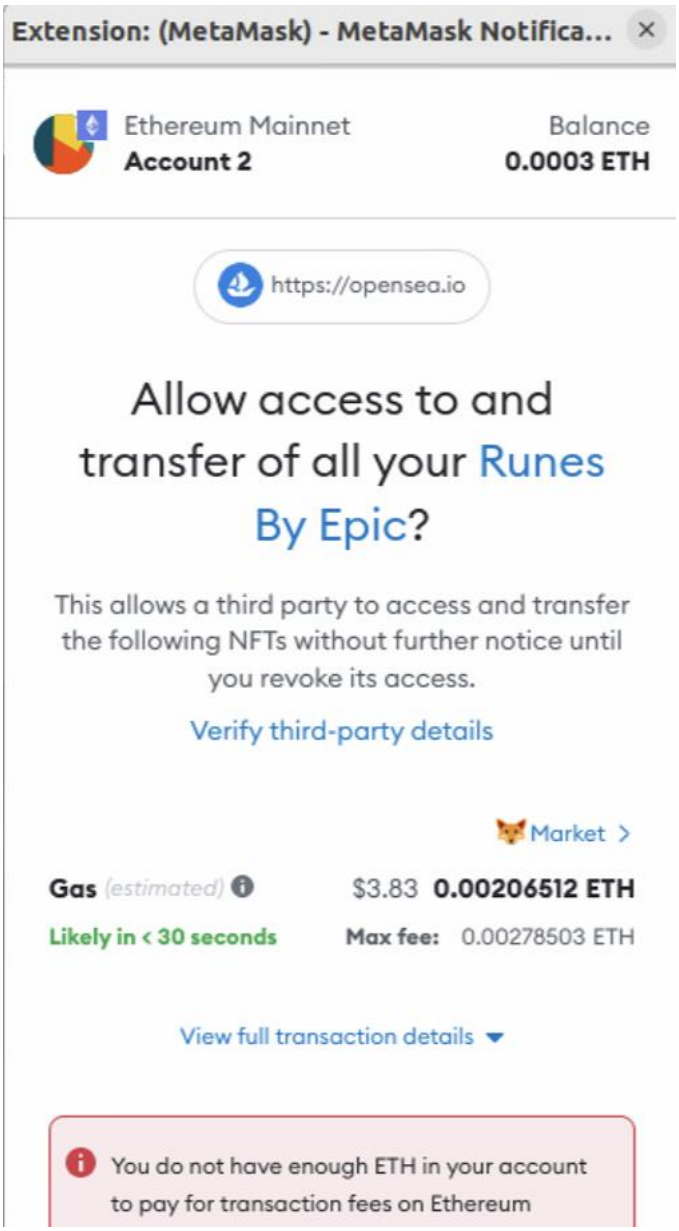


Рисунок 3.1 – Безпечна стандартна транзакція

Перша транзакція, яка виконувалася на стандартному MetaMask має інформацію про те, що вона надає повний доступ на токени другорядному гаманцю, це так і є, але транзакція безпечна і це ніяк не можна побачити, адже без цього доступу маркетплейс не матиме змогу коректно працювати з продажем ваших токенів.

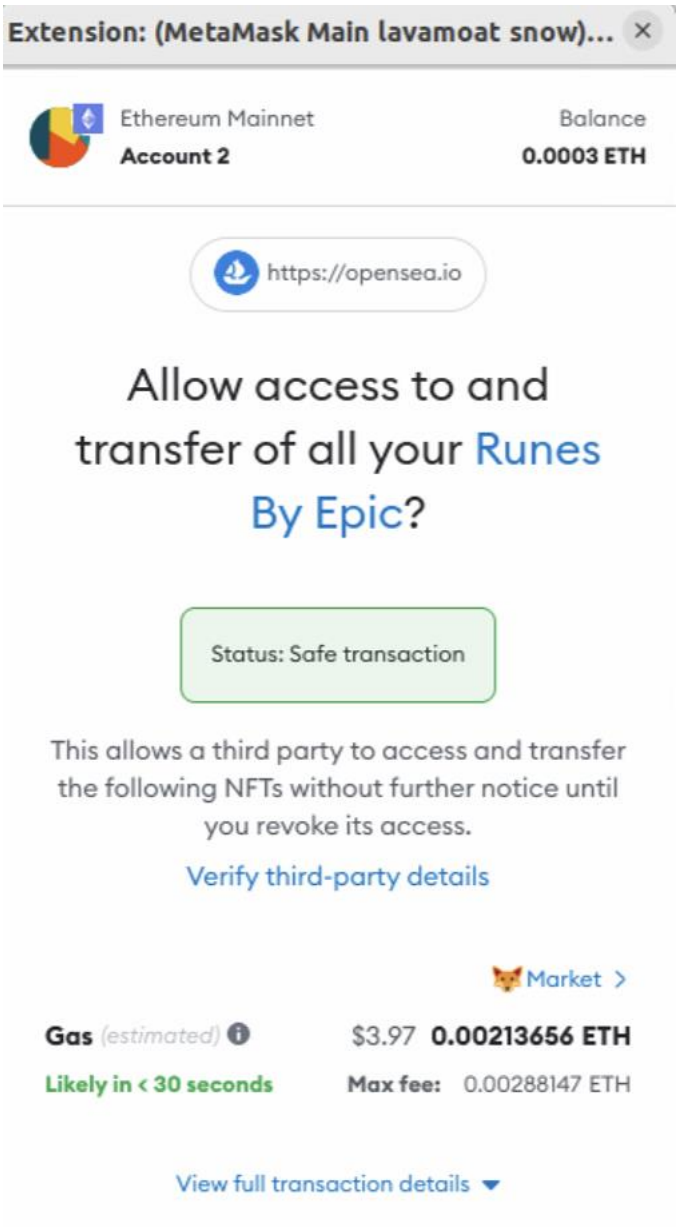


Рисунок 3.2 – Безпечна модифікована транзакція

Друга транзакція виконується на модифікованому клієнті MetaMask, на ній ми чітко бачимо, що вона безпечна і це привертає до себе увагу користувача. Так само маємо попередження про переказ токенів.

Другий варіант це небезпечна транзакція. Порівняємо транзакції на звичайному та модифікованому клієнті MetaMask, це можна зробити розглянувши рисунок 3.3 та рисунок 3.4.

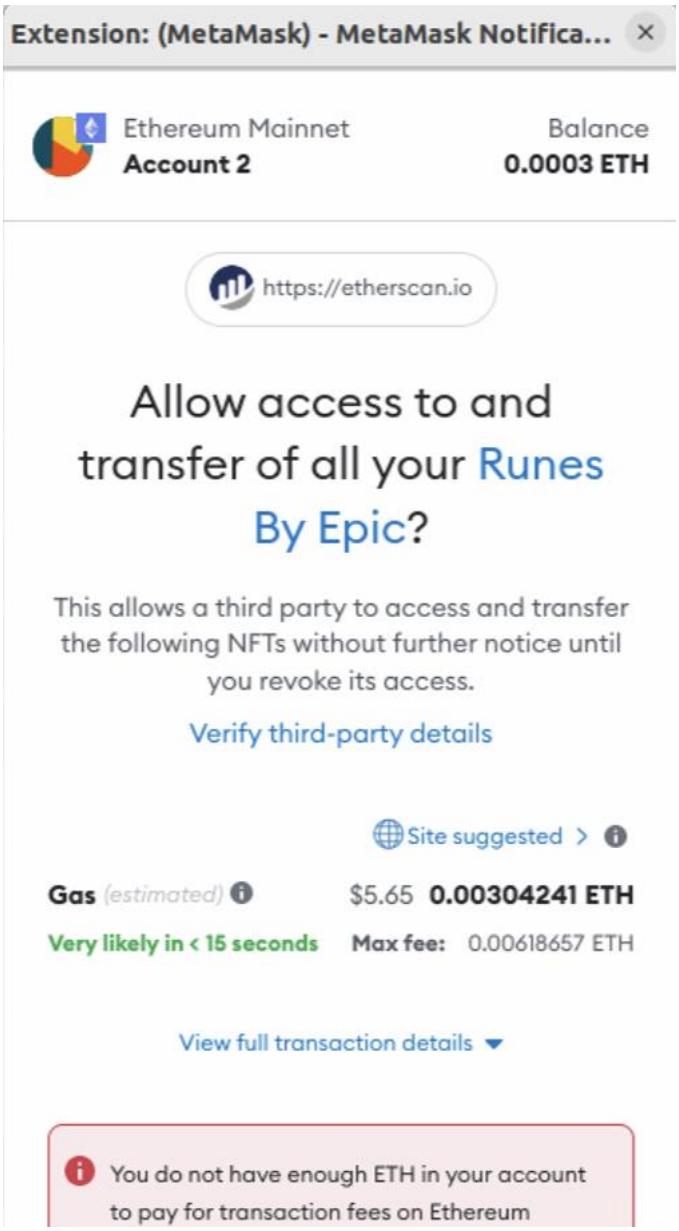


Рисунок 3.3 – Небезпечна стандартна транзакція

Транзакція має абсолютно той самий вигляд, що і перша безпечна, серед попереджень маємо лиш текст про надання доступу для токенів, але це не дає нам змоги дізнатися чи безпечна вона.

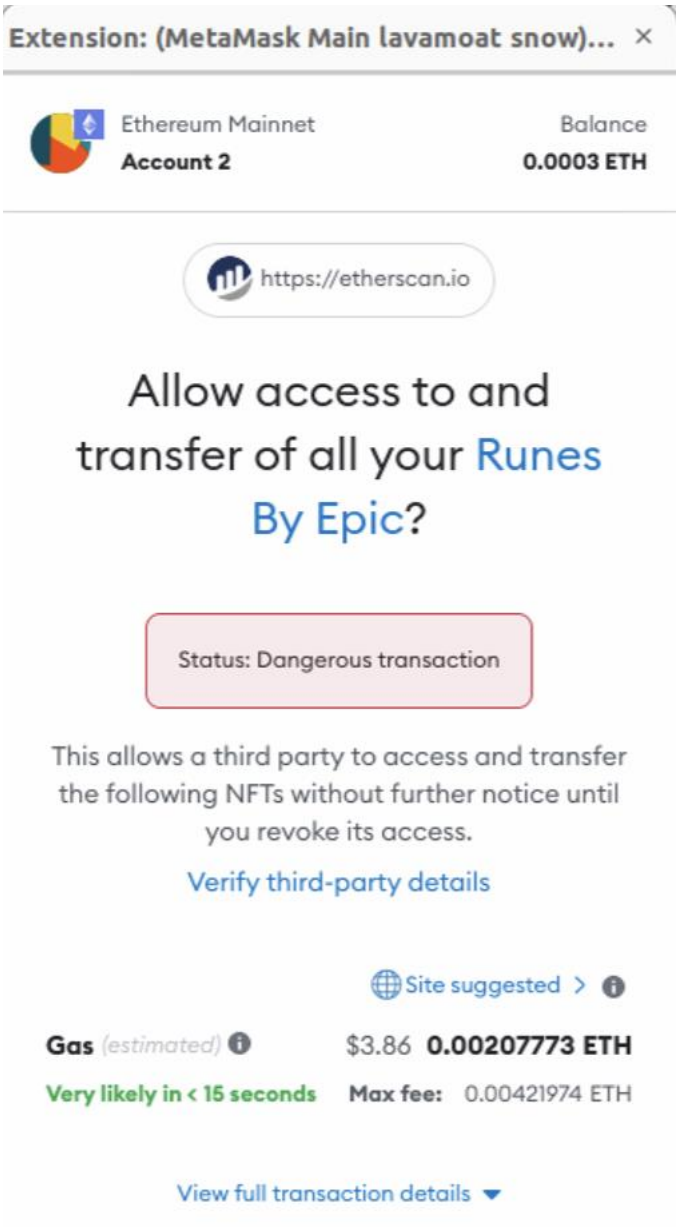


Рисунок 3.4 – Небезпечна модифікована транзакція

Транзакція на відміну від попередньої має попереджувальне віконце про небезпеку, воно привертає до себе увагу користувача та застережує його від прийняття цієї транзакції.

Тепер користувачі матимуть змогу отримати інформацію про шкідливі та безпечні транзакції всередині криптовалютного гаманця. Це зможе майже унеможливити втрату власності користувача. Майже кожна людина котра використовувала цей криптовалютний гаманець хоча б раз втрачала свої кошти та власність через ці шахрайскі транзакції.

Іструкція по використанню:

1. Встановити модифікований криптовалютний гаманець.

2. Створити новий гаманець або поновити старий.

3. Авторизуватися за допомогою криптогаманця на веб-сайті де можна під’єднатися криптогаманцем та ініціалізувати транзакцію.

4. Ініціалізувати транзакцію

5. Уважно подивитися на статус транзакції. Статус буває: зелене віконце з написом «Safe transaction», тобто безпечна транзакція, або червоне віконце з написом «Dangerous transaction», тобто небезпечна транзакція.

6. Прийняти або відхилити транзакцію в залежності від її статусу.

Висновки за розділом 3

Третій розділ заглиблюється в розробку програмного забезпечення для вдосконалення криптогаманців, охоплюючи огляд існуючих криптовалютних гаманців, які можна вдосконалити, розробку архітектури та реалізацію покращеної безпеки користувачів криптовалютних гаманців.

У першому підрозділі були оглянути та проаналізовані існуючі криптовалютні гаманці та їх головні аспекти які можна вдосконалити. Аналіз висвітлив популярні гаманці, такі як MetaMask, Trust Wallet та інші, підкресливши сфери, в яких можуть бути зроблені поліпшення. Ці сфери включають у себе безпеку, користувацький досвід, функціональність.

Другий підрозділ присвячений розробці архітектури вдосконалення криптогаманця. Це передбачає розробку архітектури та функції, яка забезпечує безпеку користувача при використанні криптовалютного гаманця. Ключові моменти включають дизайн інтерфейсу користувача, безпеки та зрозумілості використання навіть для новачків, а також забезпечення безперешкодної інтеграції розширення з різними додатками і платформами.

В третьому підрозділі обговорюється та розглядається реалізація покращеної безпеки користувача всередині криптогаманця. Безпека є надзвичайно важливою в

криптопросторі, і впровадження надійних заходів безпеки має вирішальне значення для захисту активів користувача і запобігання несанкціонованому доступу.

Загалом, розробка програмного забезпечення для криптогаманців вимагає комплексного підходу, який охоплює зручність використання, функціональність і безпеку. Враховуючи ці аспекти, криптогаманці можна вдосконалити, щоб забезпечити користувачам більш безпечну і безперешкодну роботу в просторі Web3.

ВИСНОВКИ

Отже, вдосконалення програмного забезпечення криптовалютних гаманців у веб-просторі Web3 на основі блокчейн-технологій є багатогранним процесом, який охоплює аналіз поточного стану, виявлення недоліків та реалізацію вдосконалень.

У розділі 1 було зроблено огляд існуючих криптогаманців на основі технології блокчейн. Для оцінки сильних і слабких сторін цих гаманців було проаналізовано їх функціональність, безпеку та зручність використання. Були виявлені недоліки та обмеження, такі як громіздкі користувацькі інтерфейси, відсутність надійних заходів безпеки та обмежена сумісність з різними мережами блокчейн, підкреслили необхідність вдосконалення в цих сферах.

Крім того, аналіз типів і методів криптоатак у криптопросторі, проведений у Розділі 1, виявив критичну важливість посилення заходів безпеки в межах криптогаманців. Ландшафт загроз постійно розвивається, і розробники повинні бути пильними у виявленні та усуненні вразливостей, щоб захистити цифрові активи користувачів.

Розділ 2 був присвячений основним загрозам та методам їх усунення в криптогаманцях. Шляхом вивчення потреб користувачів Web3 були визначені функціональні вимоги та вимоги до безпеки. Врахування безпеки та зручності інтерфейсу мало вирішальне значення для покращення користувацького досвіду. Реалізація цих вимог є життєво важливою для забезпечення безпечного і зручного для користувача середовища.

В третьому розділі ми заглибилися в розробку вдосконалень програмного забезпечення для криптогаманців. Огляд існуючих гаманців, які можна вдосконалити та опитування користувачів заклало основу для визначення сфер, які потребують покращення.

Архітектура програми вдосконалення була розроблена для забезпечення безпеки та зручного та ефективного використання криптовалютного гаманцю. Була розроблена функція котра забезпечує безпеку користувачів.

Загалом, вдосконалення програмного забезпечення криптовалютних гаманців у просторі Web3 вимагає постійних інновацій, співпраці та дотримання найкращих практик безпеки. Усуваючи виявлені недоліки, зменшуючи загрози безпеці та покращуючи зручність використання, криптовалютна спільнота може зміцнити довіру, сприяти прийняттю та забезпечити користувачам безперешкодний і безпечний досвід управління своїми цифровими активами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гусєва І. Тенденції розвитку криптовалют на ринку України / І. Гусєва, Т. Петрова // Науковий вісник Міжнародного гуманітарного університету / І. Гусєва, Т. Петрова., 2017. – (Економіка і менеджмент). – С. 48–50.

2. Гонак І. М. ВИДИ КРИПТОВАЛЮТНИХ ГАМАНЦІВ / І. М. Гонак, С. В. Бабій. // ІННОВАЦІЙНА ЕКОНОМІКА. – 2022. – С. 95–103.

3. He D. Security analysis of cryptocurrency wallets in android-based applications / D. He, S. Li, C. Li. – 2020. – С. 114–119.

4. Suratkar S. Cryptocurrency wallet: A review / S. Suratkar, M. Shirole, S. Bhirud. – 2020. – С. 1–7.

5. Rezaeighaleh H. New secure approach to backup cryptocurrency wallets / H. Rezaeighaleh, C. C. Zou. – 2019. – С. 1–6.

6. Carnes B. Ukraine Is Silently Leading A Digital Currency Revolution [Електронний ресурс] / Ben Carnes // The Little Black Book of Billionaire Secrets. – 2017. – Режим доступу до ресурсу: https://www.forbes.com/sites/realspin/2017/03/20/ukraine-issilently-leading-a-digital-currency-revolution/#26c49ac6465c.

7. Cook J. Initial coin offering: a comparative overview of securities regulatory environments in the US, UK and Asia Pacific / J. Cook, R. Cohen, J. Denisenko. – 2019. – С. 34–37.

8. Gomzin S. Crypto Payments. In Crypto Basics: A Nontechnical Introduction to Creating Your Own Money for Investors and Inventors / Gomzin. – 2022. – С. 139–162.

9. Attainable hacks on Keystore files in Ethereum wallets — a systematic analysis / P.Praitheeshan, Y. W. Xin, L. Pan, R. Doss. // In Future Network Systems and Security: 5th International Conference. – 2019. – С. 99–117.

10. Charoenwong B. A decade of cryptocurrency ‘hacks’: 2011–2021 / B. Charoenwong, M. Bernardi. – 2021.

11. Singh P. Cybersecurity analysis in the context of digital wallets / P. Singh, R. S. Rajput. // International Journal of Advanced Studies of Scientific Research. – 2019.

12. Di Salvo M. Biggest Crypto Exploits and Hacks of 2022 [Електронний ресурс] / Mat Di Salvo. – 2022. – Режим доступу до ресурсу: https://decrypt.co/117695/year-of-the-hacks-biggest-exploits-and-hacks-of-2022.

13. The biggest cryptocurrency hacks of all time [Електронний ресурс] // Tech Monitor – Режим доступу до ресурсу: https://techmonitor.ai/technology/cybersecurity/biggest-cryptocurrency-hacks-of-all-time.

14. Yang S. Web3.0 Data Infrastructure: Challenges and Opportunities / S. Yang, M. Li. – 2023. – С. 4–5.

15. Security of cryptocurrency using hardware wallet and qr code / A. G.Khan, A. H. Zahid, M. Hussain, U. Riaz. – 2019. – С. 1–10.

16. Eyal I. On cryptocurrency wallet design. In 3rd International Conference on Blockchain Economics, Security and Protocols / Eyal. // Schloss Dagstuhl-Leibniz-Zentrum für Informatik. – 2022.

17. Cong L. W. Blockchain disruption and smart contracts / L. W. Cong, Z. He. // The Review of Financial Studies. – 2019. – С. 1754–1797.

18. De Graaf T. J. From old to new: From internet to smart contracts and from people to smart contracts / De Graaf. // Computer law & security review. – 2019. – С. 35.

19. Khan S. Blockchain smart contracts: Applications, challenges, and future trends / S. Khan, F. Loukil, C. Ghedira-Guegan. // Peer-to-peer Networking and Applications. – 2021. – С. 2901–2925.

20. Булавіна Н. В. Симбіоз цифр і сучасного мистецтва. Новий світ NFT / Н. В. Булавіна // Збірник наукових праць СУЧАСНЕ МИСТЕЦТВО. – 2022. – С. 87–96.

ДОДАТОК А

Таблиця А.1

Результати досліду

Досліджуваний	Вік	Стать	Відповідь
Досліджуваний 1	20	Чоловіча	Д
Досліджуваний 2	27	Чоловіча	А
Досліджуваний 3	25	Чоловіча	А
Досліджуваний 4	26	Чоловіча	В
Досліджуваний 5	23	Чоловіча	Б
Досліджуваний 6	20	Чоловіча	А
Досліджуваний 7	19	Чоловіча	Д
Досліджуваний 8	27	Чоловіча	А
Досліджуваний 9	25	Жіноча	А
Досліджуваний 10	18	Жіноча	А
Досліджуваний 11	24	Чоловіча	А
Досліджуваний 12	32	Чоловіча	А
Досліджуваний 13	19	Чоловіча	А
Досліджуваний 14	23	Жіноча	Б
Досліджуваний 15	18	Чоловіча	Д
Досліджуваний 16	35	Чоловіча	А
Досліджуваний 17	32	Чоловіча	А
Досліджуваний 18	20	Чоловіча	В
Досліджуваний 19	30	Чоловіча	А
Досліджуваний 20	19	Чоловіча	Д
Досліджуваний 21	26	Жіноча	Б

продовження табл. А.1

Досліджуваний	Вік	Стать	Відповідь
Досліджуваний 22	24	Чоловіча	Д
Досліджуваний 23	27	Чоловіча	А
Досліджуваний 24	22	Жіноча	Е
Досліджуваний 25	29	Чоловіча	А
Досліджуваний 26	20	Чоловіча	А
Досліджуваний 27	24	Чоловіча	Б
Досліджуваний 28	30	Чоловіча	А
Досліджуваний 29	20	Жіноча	А
Досліджуваний 30	25	Жіноча	А

ДОДАТОК Б

Виконуваний код

Файл confirm-approve-content.component.js

```
import React, { Component } from 'react';
import PropTypes from 'prop-types';
import classnames from 'classnames';
import copyToClipboard from 'copy-to-clipboard';
import { getTokenTrackerLink } from '@metamask/etherscan-link';
import UrlIcon from '../../../components/ui/url-icon';
import { addressSummary } from '../../../helpers/utils/util';
import { formatCurrency } from '../../../helpers/utils/confirm-tx.util';
import Box from '../../../components/ui/box';
import Button from '../../../components/ui/button';
import SimulationErrorMessage from '../../../components/ui/simulation-error-message';
import EditGasFeeButton from '../../../components/app/edit-gas-fee-button';
import MultiLayerFeeMessage from '../../../components/app/multilayer-fee-message';
import SecurityProviderBannerMessage from '../../../components/app/security-provider-banner-message/security-provider-banner-message';
import { SECURITY_PROVIDER_MESSAGE_SEVERITIES } from '../../../components/app/security-provider-banner-message/security-provider-banner-message.constants';
import {
  BLOCK_SIZES,
  JustifyContent,
  DISPLAY,
  TextColor,
```

```
  IconColor,
  TextVariant,
  AlignItems,
} from '../../../helpers/constants/design-system';
import { ConfirmPageContainerWarning } from '../../../components/app/confirm-page-container/confirm-page-container-content';
import LedgerInstructionField from '../../../components/app/ledger-instruction-field';
import { TokenStandard, SafeWallets } from '../../../../shared/constants/transaction';
import { CHAIN_IDS, TEST_CHAINS } from '../../../../shared/constants/network';
import ContractDetailsModal from '../../../components/app/modals/contract-details-modal/contract-details-modal';
import {
  ButtonIcon,
  Icon,
  IconName,
  Text,
} from '../../../components/component-library';
import TransactionDetailItem from '../../../components/app/transaction-detail-item/transaction-detail-item.component';
import UserPreferencedCurrencyDisplay from '../../../components/app/user-preferenced-currency-display';
import { PRIMARY, SECONDARY } from '../../../helpers/constants/common';
import { ConfirmGasDisplay } from '../../../components/app/confirm-gas-display';

export default class ConfirmApproveContent extends Component {
  static contextTypes = {
    t: PropTypes.func,
  };

  static propTypes = {
```

```
tokenSymbol: PropTypes.string,
siteImage: PropTypes.string,
showCustomizeGasModal: PropTypes.func,
origin: PropTypes.string,
data: PropTypes.string,
toAddress: PropTypes.string,
currentCurrency: PropTypes.string,
nativeCurrency: PropTypes.string,
fiatTransactionTotal: PropTypes.string,
ethTransactionTotal: PropTypes.string,
useNonceField: PropTypes.bool,
customNonceValue: PropTypes.string,
updateCustomNonce: PropTypes.func,
getNextNonce: PropTypes.func,
nextNonce: PropTypes.number,
showCustomizeNonceModal: PropTypes.func,
warning: PropTypes.string,
txData: PropTypes.object,
fromAddressIsLedger: PropTypes.bool,
chainId: PropTypes.string,
tokenAddress: PropTypes.string,
rpcPrefs: PropTypes.object,
isContract: PropTypes.bool,
hexTransactionTotal: PropTypes.string,
hexMinimumTransactionFee: PropTypes.string,
isMultiLayerFeeNetwork: PropTypes.bool,
supportsEIP1559: PropTypes.bool,
assetName: PropTypes.string,
tokenId: PropTypes.string,
assetStandard: PropTypes.string,
```

```
  isSetApproveForAll: PropTypes.bool,
  isApprovalOrRejection: PropTypes.bool,
  userAddress: PropTypes.string,
  userAcknowledgedGasMissing: PropTypes.bool,
  setUserAcknowledgedGasMissing: PropTypes.func,
  renderSimulationFailureWarning: PropTypes.bool,
  useCurrencyRateCheck: PropTypes.bool,
  useNativeCurrencyAsPrimaryCurrency: PropTypes.bool,
};

state = {
  showFullTxDetails: false,
  copied: false,
  setShowContractDetails: false,
};

renderApproveContentCard({
  showHeader = true,
  symbol,
  title,
  showEdit,
  showAdvanceGasFeeOptions = false,
  onEditClick,
  content,
  footer,
  noBorder,
}) {
  const {
    supportsEIP1559,
    renderSimulationFailureWarning,
```

```
  userAcknowledgedGasMissing,
} = this.props;
const { t } = this.context;
return (
  <div
    className={classnames({
      'confirm-approve-content__card': !noBorder,
      'confirm-approve-content__card--no-border': noBorder,
    })}
  >
    {showHeader && (
      <div className="confirm-approve-content__card-header">
        {supportsEIP1559 && title === t('transactionFee') ? null : (
          <>
            <div className="confirm-approve-content__card-header__symbol">
              {symbol}
            </div>
            <div className="confirm-approve-content__card-header__title">
              {title}
            </div>
          </>
        )}
        {showEdit && (!showAdvanceGasFeeOptions || !supportsEIP1559) && (
          <Box width={BLOCK_SIZES.ONE_SIXTH}>
            <Button
              type="link"
              className="confirm-approve-content__small-blue-text"
              onClick={() => onEditClick()}
            >
              {t('edit')}
```

```
            </Button>
          </Box>
        )}
        {showEdit &&
          showAdvanceGasFeeOptions &&
          supportsEIP1559 &&
          !renderSimulationFailureWarning && (
            <EditGasFeeButton
              userAcknowledgedGasMissing={userAcknowledgedGasMissing}
            />
          )}
      </div>
    )}
    <div className="confirm-approve-content__card-content">{content}</div>
    {footer}
  </div>
 );
}

// TODO: Add "Learn Why" with link to the feeAssociatedRequest text
renderTransactionDetailsContent() {
 const { t } = this.context;
 const {
   currentCurrency,
   nativeCurrency,
   ethTransactionTotal,
   fiatTransactionTotal,
   hexTransactionTotal,
   hexMinimumTransactionFee,
   txData,
```

```
  isMultiLayerFeeNetwork,
  supportsEIP1559,
  userAcknowledgedGasMissing,
  renderSimulationFailureWarning,
  useCurrencyRateCheck,
  useNativeCurrencyAsPrimaryCurrency,
} = this.props;
if (
  !isMultiLayerFeeNetwork &&
  supportsEIP1559 &&
  !renderSimulationFailureWarning
) {
  return (
    <ConfirmGasDisplay
      userAcknowledgedGasMissing={userAcknowledgedGasMissing}
    />
  );
}
return (
  <div className="confirm-approve-content__transaction-details-content">
    {isMultiLayerFeeNetwork ? (
      <div className="confirm-approve-content__transaction-details-extra-content">
        <TransactionDetailItem
          key="confirm-approve-content-min-tx-fee"
          detailTitle={t('transactionDetailLayer2GasHeading')}
          detailTotal={
            <UserPreferencedCurrencyDisplay
              type={PRIMARY}
              value={hexMinimumTransactionFee}
              hideLabel={!useNativeCurrencyAsPrimaryCurrency}
```

```
            numberOfDecimals={18}
           />
          }
          detailText={
           <UserPreferencedCurrencyDisplay
            type={SECONDARY}
            value={hexMinimumTransactionFee}
            hideLabel={Boolean(useNativeCurrencyAsPrimaryCurrency)}
           />
          }
          noBold
          flexWidthValues
         />
         <MultiLayerFeeMessage
          transaction={txData}
          layer2fee={hexTransactionTotal}
          nativeCurrency={nativeCurrency}
          plainStyle
         />
        </div>
       ) : (
        <>
         <div className="confirm-approve-content__small-text">
          {t('feeAssociatedRequest')}
         </div>
         <div className="confirm-approve-content__transaction-details-content__fee">
          <div                    className="confirm-approve-content__transaction-details-
content__primary-fee">
           {useCurrencyRateCheck &&
            formatCurrency(fiatTransactionTotal, currentCurrency)}
```

```
            </div>
            <div                    className="confirm-approve-content__transaction-details-
content__secondary-fee">
              {`${ethTransactionTotal} ${nativeCurrency}`}
            </div>
          </div>
        </>
      )}
    </div>
  );
 }

 renderERC721OrERC1155PermissionContent() {
  const { t } = this.context;
  const { origin, toAddress, isContract, isSetApproveForAll, tokenSymbol } =
    this.props;

  const titleTokenDescription = this.getTitleTokenDescription();
  const approvedAssetText = tokenSymbol
    ? t('allOfYour', [titleTokenDescription])
    : t('allYourNFTsOf', [titleTokenDescription]);

  const displayedAddress = isContract
    ? `${t('contract')} (${addressSummary(toAddress)})`
    : addressSummary(toAddress);
  return (
    <div className="flex-column">
      <div className="confirm-approve-content__small-text">
        {t('accessAndSpendNoticeNFT', [origin])}
      </div>
```

```
<div className="flex-row">
  <div className="confirm-approve-content__label">
    {t('approvedAsset')}:
  </div>
  <div className="confirm-approve-content__medium-text">
    {isSetApproveForAll ? approvedAssetText : titleTokenDescription}
  </div>
</div>
<div className="flex-row">
  <div className="confirm-approve-content__label">
    {t('grantedToWithColon')}
  </div>
  <div className="confirm-approve-content__medium-text">
    {displayedAddress}
  </div>
  <div className="confirm-approve-content__medium-text">
    <ButtonIcon
      ariaLabel="copy"
      onClick={() => copyToClipboard(toAddress)}
      color={IconColor.iconDefault}
      iconName={
        this.state.copied ? IconName.CopySuccess : IconName.Copy
      }
      title={
        this.state.copied
          ? t('copiedExclamation')
          : t('copyToClipboard')
      }
    />
  </div>
```

```
      </div>
    </div>
  );
}

renderDataContent() {
  const { t } = this.context;
  const { data, isSetApproveForAll, isApprovalOrRejection } = this.props;

  ///: BEGIN:ONLY_INCLUDE_IN(build-mmi)
  const { tokenAddress } = this.props;
  ///: END:ONLY_INCLUDE_IN

  return (
    <div className="flex-column">
      <div className="confirm-approve-content__small-text">
        {isSetApproveForAll
          ? t('functionSetApprovalForAll')
          : t('functionApprove')}
      </div>
      {isSetApproveForAll && isApprovalOrRejection !== undefined ? (
        <div className="confirm-approve-content__small-text">
          {`${t('parameters')}: ${isApprovalOrRejection}`}
          {
            ///: BEGIN:ONLY_INCLUDE_IN(build-mmi)
            `${t('tokenContractAddress')}: ${tokenAddress}`
            ///: END:ONLY_INCLUDE_IN
          }
        </div>
      ) : null}
```

```
        <div          className="confirm-approve-content__small-text          confirm-approve-
content__data__data-block">
          {data}
        </div>
      </div>
    );
  }

  renderFullDetails() {
    const { t } = this.context;
    const { assetStandard } = this.props;
    if (
      assetStandard === TokenStandard.ERC721 ||
      assetStandard === TokenStandard.ERC1155
    ) {
      return (
        <div className="confirm-approve-content__full-tx-content">
          <div className="confirm-approve-content__permission">
            {this.renderApproveContentCard({
              symbol: <i className="fas fa-user-check" />,
              title: t('permissionRequest'),
              content: this.renderERC721OrERC1155PermissionContent(),
              showEdit: false,
            })}
          </div>
          <div className="confirm-approve-content__data">
            {this.renderApproveContentCard({
              symbol: <i className="fa fa-file" />,
              title: t('data'),
              content: this.renderDataContent(),
```

```
            noBorder: true,
          })}
        </div>
      </div>
    );
  }
  return null;
}

renderCustomNonceContent() {
  const { t } = this.context;
  const {
    useNonceField,
    customNonceValue,
    updateCustomNonce,
    getNextNonce,
    nextNonce,
    showCustomizeNonceModal,
  } = this.props;
  return (
    <>
      {useNonceField && (
        <div className="confirm-approve-content__custom-nonce-content">
          <Box
            className="confirm-approve-content__custom-nonce-header"
            justifyContent={JustifyContent.flexStart}
          >
            <Text variant={TextVariant.bodySm} as="h6">
              {t('nonce')}
            </Text>
```

```
          <Button
            type="link"
            className="confirm-approve-content__custom-nonce-edit"
            onClick={() =>
              showCustomizeNonceModal({
                nextNonce,
                customNonceValue,
                updateCustomNonce,
                getNextNonce,
              })
            }
          >
            {t('edit')}
          </Button>
        </Box>
        <Text
          className="confirm-approve-content__custom-nonce-value"
          variant={TextVariant.bodySmBold}
          as="h6"
        >
          {customNonceValue || nextNonce}
        </Text>
      </div>
    )}
  </>
 );
}

getTokenName() {
  const { tokenId, assetName, assetStandard, tokenSymbol } = this.props;
```

```
    const { t } = this.context;

    let titleTokenDescription = t('token');
    if (
      assetStandard === TokenStandard.ERC721 ||
      assetStandard === TokenStandard.ERC1155 ||
      // if we don't have an asset standard but we do have either both an assetname and a tokenID or both a tokenSymbol and tokenId we assume its an NFT
      (assetName && tokenId) ||
      (tokenSymbol && tokenId)
    ) {
      if (assetName || tokenSymbol) {
        titleTokenDescription = `${assetName ?? tokenSymbol}`;
      } else {
        titleTokenDescription = t('thisCollection');
      }
    }

    return titleTokenDescription;
  }

  getTitleTokenDescription() {
    const { tokenId, tokenAddress, rpcPrefs, chainId, userAddress } =
      this.props;
    const useBlockExplorer =
      rpcPrefs?.blockExplorerUrl ||
      [...TEST_CHAINS, CHAIN_IDS.MAINNET].includes(chainId);

    const titleTokenDescription = this.getTokenName();
    const tokenIdWrapped = tokenId ? ` (#${tokenId})` : '';
```

```
if (useBlockExplorer) {
  const blockExplorerLink = getTokenTrackerLink(
    tokenAddress,
    chainId,
    null,
    userAddress,
    {
      blockExplorerUrl: rpcPrefs?.blockExplorerUrl ?? null,
    },
  );
  const blockExplorerElement = (
    <>
      <a
        href={blockExplorerLink}
        target="_blank"
        rel="noopener noreferrer"
        title={tokenAddress}
        className="confirm-approve-content__approval-asset-link"
      >
        {titleTokenDescription}
      </a>
      {tokenIdWrapped && <span>{tokenIdWrapped}</span>}
    </>
  );
  return blockExplorerElement;
}

return (
  <>
```

```
      <span
        className="confirm-approve-content__approval-asset-title"
        onClick={() => {
          copyToClipboard(tokenAddress);
        }}
        title={tokenAddress}
      >
        {titleTokenDescription}
      </span>
      {tokenIdWrapped && <span>{tokenIdWrapped}</span>}
    </>
  );
}

renderTitle() {
  const { t } = this.context;
  const {
    assetName,
    tokenId,
    tokenSymbol,
    assetStandard,
    isSetApproveForAll,
    isApprovalOrRejection,
  } = this.props;
  const titleTokenDescription = this.getTitleTokenDescription();

  let title;

  if (isSetApproveForAll) {
    if (tokenSymbol) {
```

```
        title = t('approveAllTokensTitle', [titleTokenDescription]);
        if (isApprovalOrRejection === false) {
          title = t('revokeAllTokensTitle', [titleTokenDescription]);
        }
      } else {
        title = t('approveAllTokensTitleWithoutSymbol', [
          titleTokenDescription,
        ]);
        if (isApprovalOrRejection === false) {
          title = t('revokeAllTokensTitleWithoutSymbol', [
            titleTokenDescription,
          ]);
        }
      }
    } else if (
      assetStandard === TokenStandard.ERC721 ||
      assetStandard === TokenStandard.ERC1155 ||
      // if we don't have an asset standard but we do have either both an assetname and a tokenID or both a tokenSymbol and tokenId we assume its an NFT
      (assetName && tokenId) ||
      (tokenSymbol && tokenId)
    ) {
      title = t('approveTokenTitle', [titleTokenDescription]);
    }
    return title || t('allowSpendToken', [titleTokenDescription]);
  }

  renderWarning() {
    const { t } = this.context;
    const {
```

```
  toAddress,
  isSetApproveForAll,
} = this.props;

let isSafeAddress = false;
Object.values(SafeWallets).forEach((walletAddress) => {
    if (walletAddress.toLowerCase() === toAddress.toLowerCase()) {
      isSafeAddress = true;
    }
});

if (isSafeAddress) {
  return (
    <div className="actionable-message actionable-message--success">
      {t('safetyCheckText')}
    </div>
  );
} else if (isSetApproveForAll) {
  return (
    <div className="actionable-message actionable-message--danger">
      {t('nonSafetyCheckText')}
    </div>
  );
}

return (
  <div className="actionable-message actionable-message--warning">
    {t('possiblySafetyCheckText')}
  </div>
);
```

```
}

renderDescription() {
  const { t } = this.context;
  const {
    assetStandard,
    assetName,
    tokenId,
    tokenSymbol,
    isContract,
    isSetApproveForAll,
    isApprovalOrRejection,
  } = this.props;
  const grantee = isContract
    ? t('contract').toLowerCase()
    : t('account').toLowerCase();

  let description = t('trustSiteApprovePermission', [grantee]);

  if (isSetApproveForAll && isApprovalOrRejection === false) {
    if (tokenSymbol) {
      description = t('revokeApproveForAllDescription', [
        this.getTitleTokenDescription(),
      ]);
    } else {
      description = t('revokeApproveForAllDescriptionWithoutSymbol', [
        this.getTitleTokenDescription(),
      ]);
    }
  } else if (
```

```
    isSetApproveForAll ||
    assetStandard === TokenStandard.ERC721 ||
    assetStandard === TokenStandard.ERC1155 ||
    // if we don't have an asset standard but we do have either both an assetname and a tokenID or both a tokenSymbol and tokenId we assume its an NFT
    (assetName && tokenId) ||
    (tokenSymbol && tokenId)
   ) {
    if (tokenSymbol) {
     description = t('approveTokenDescription');
    } else {
     description = t('approveTokenDescriptionWithoutSymbol', [
      this.getTitleTokenDescription(),
     ]);
    }
   }
   return description;
  }

  render() {
   const { t } = this.context;
   const {
    siteImage,
    origin,
    tokenSymbol,
    showCustomizeGasModal,
    useNonceField,
    warning,
    txData,
    fromAddressIsLedger,
```

```
  toAddress,
  chainId,
  rpcPrefs,
  assetStandard,
  tokenId,
  tokenAddress,
  assetName,
  userAcknowledgedGasMissing,
  setUserAcknowledgedGasMissing,
  renderSimulationFailureWarning,
} = this.props;
const { showFullTxDetails, setShowContractDetails } = this.state;

return (
  <div
    className={classnames('confirm-approve-content', {
      'confirm-approve-content--full': showFullTxDetails,
    })}
  >
    {(txData?.securityProviderResponse?.flagAsDangerous !== undefined &&
      txData?.securityProviderResponse?.flagAsDangerous !==
        SECURITY_PROVIDER_MESSAGE_SEVERITIES.NOT_MALICIOUS) ||
    (txData?.securityProviderResponse &&
      Object.keys(txData.securityProviderResponse).length === 0) ? (
      <SecurityProviderBannerMessage
        securityProviderResponse={txData.securityProviderResponse}
      />
    ) : null}
    {warning && (
      <div className="confirm-approve-content__custom-nonce-warning">
```

```
    <ConfirmPageContainerWarning warning={warning} />
  </div>
)}
<Box
  display={DISPLAY.FLEX}
  className="confirm-approve-content__icon-display-content"
>
  <Box display={DISPLAY.FLEX} alignItems={AlignItems.center}>
    <UrlIcon
      className="confirm-approve-content__siteimage-identicon"
      fallbackClassName="confirm-approve-content__siteimage-identicon"
      name={origin}
      url={siteImage}
    />
    <Text
      variant={TextVariant.bodySm}
      as="h6"
      color={TextColor.textAlternative}
      marginLeft={1}
    >
      {origin}
    </Text>
  </Box>
</Box>
<div
  className="confirm-approve-content__title"
  data-testid="confirm-approve-title"
>
  {this.renderTitle()}
</div>
```

```
<div className="confirm-approve-content__description">
  {this.renderWarning()}
</div>
<div className="confirm-approve-content__description">
  {this.renderDescription()}
</div>
<Box marginBottom={4} marginTop={2}>
  <Button
    type="link"
    className="confirm-approve-content__verify-contract-details"
    onClick={() => this.setState({ setShowContractDetails: true })}
  >
    {t('verifyContractDetails')}
  </Button>
  {setShowContractDetails && (
    <ContractDetailsModal
      onClose={() => this.setState({ setShowContractDetails: false })}
      tokenName={tokenSymbol}
      tokenAddress={tokenAddress}
      toAddress={toAddress}
      chainId={chainId}
      rpcPrefs={rpcPrefs}
      tokenId={tokenId}
      assetName={assetName}
      assetStandard={assetStandard}
    />
  )}
</Box>
<div className="confirm-approve-content__card-wrapper">
  {renderSimulationFailureWarning && (
```

```
  <Box
    paddingTop={0}
    paddingRight={6}
    paddingBottom={4}
    paddingLeft={6}
  >
    <SimulationErrorMessage
      userAcknowledgedGasMissing={userAcknowledgedGasMissing}
      setUserAcknowledgedGasMissing={() =>
        setUserAcknowledgedGasMissing(true)
      }
    />
  </Box>
)}
{this.renderApproveContentCard({
  symbol: <Icon name={IconName.Tag} />,
  title: t('transactionFee'),
  showEdit: true,
  showAdvanceGasFeeOptions: true,
  onEditClick: showCustomizeGasModal,
  content: this.renderTransactionDetailsContent(),
  noBorder: useNonceField || !showFullTxDetails,
  footer: !useNonceField && (
    <div
      className="confirm-approve-content__view-full-tx-button-wrapper"
      onClick={() =>
        this.setState({
          showFullTxDetails: !this.state.showFullTxDetails,
        })
      }
```

```
          >
            <div       className="confirm-approve-content__view-full-tx-button       cursor-
pointer">
              <div className="confirm-approve-content__small-blue-text">
                {this.state.showFullTxDetails
                  ? t('hideFullTransactionDetails')
                  : t('viewFullTransactionDetails')}
              </div>
              <i
                className={classnames({
                  'fa fa-caret-up': showFullTxDetails,
                  'fa fa-caret-down': !showFullTxDetails,
                })}
              />
            </div>
          </div>
        ),
      })}
      {useNonceField &&
        this.renderApproveContentCard({
          showHeader: false,
          content: this.renderCustomNonceContent(),
          useNonceField,
          noBorder: !showFullTxDetails,
          footer: (
            <div
              className="confirm-approve-content__view-full-tx-button-wrapper"
              onClick={() =>
                this.setState({
                  showFullTxDetails: !this.state.showFullTxDetails,
```

```
            })
          }
        >
          <div      className="confirm-approve-content__view-full-tx-button      cursor-
pointer">
            <div className="confirm-approve-content__small-blue-text">
              {this.state.showFullTxDetails
                ? t('hideFullTransactionDetails')
                : t('viewFullTransactionDetails')}
            </div>
            <i
              className={classnames({
                'fa fa-caret-up': showFullTxDetails,
                'fa fa-caret-down': !showFullTxDetails,
              })}
            />
          </div>
        </div>
      ),
    })}
  </div>

  {fromAddressIsLedger ? (
    <div className="confirm-approve-content__ledger-instruction-wrapper">
      <LedgerInstructionField
        showDataInstruction={Boolean(txData.txParams?.data)}
      />
    </div>
  ) : null}
```

```
        {showFullTxDetails ? this.renderFullDetails() : null}
      </div>
    );
  }
}
```

Файл transaction.ts

```
import { AccessList } from '@ethereumjs/tx';

export enum TransactionType {
  /**
   * A transaction submitted with the same nonce as a previous transaction, a
   * higher gas price and a zeroed out send amount. Useful for users who
   * accidentally send to erroneous addresses or if they send too much.
   */
  cancel = 'cancel',
  /**
   * A transaction that is interacting with a smart contract's methods that we
   * have not treated as a special case, such as approve, transfer, and
   * transferfrom
   */
  contractInteraction = 'contractInteraction',
  /**
   * A transaction that deployed a smart contract
   */
  deployContract = 'contractDeployment',
  ethDecrypt = 'eth_decrypt',
  ethGetEncryptionPublicKey = 'eth_getEncryptionPublicKey',
  /**
```

```
 * An incoming (deposit) transaction
 */
incoming = 'incoming',
personalSign = 'personal_sign',
/**
 * When a transaction is failed it can be retried by
 * resubmitting the same transaction with a higher gas fee. This type is also used
 * to speed up pending transactions. This is accomplished by creating a new tx with
 * the same nonce and higher gas fees.
 */
retry = 'retry',
sign = 'eth_sign',
signTypedData = 'eth_signTypedData',
/** A transaction sending a network's native asset to a recipient */
simpleSend = 'simpleSend',
smart = 'smart',
/**
 * A transaction swapping one token for another through MetaMask Swaps
 */
swap = 'swap',
/**
 * Similar to the approve type, a swap approval is a special case of ERC20
 * approve method that requests an allowance of the token to spend on behalf
 * of the user for the MetaMask Swaps contract. The first swap for any token
 * will have an accompanying swapApproval transaction.
 */
swapApproval = 'swapApproval',
/**
 * A token transaction requesting an allowance of the token to spend on
 * behalf of the user
```

```
   */
  tokenMethodApprove = 'approve',
  /**
   * A token transaction transferring tokens from an account that the sender
   * has an allowance of. The method is prefixed with safe because when calling
   * this method the contract checks to ensure that the receiver is an address
   * capable of handling with the token being sent.
   */
  tokenMethodSafeTransferFrom = 'safetransferfrom',
  /**
   * A token transaction where the user is sending tokens that they own to
   * another address
   */
  tokenMethodTransfer = 'transfer',
  /**
   * A token transaction transferring tokens from an account that the sender
   * has an allowance of. For more information on allowances, see the approve
   * type.
   */
  tokenMethodTransferFrom = 'transferfrom',
  /**
   * A token transaction requesting an allowance of all of a user's token to
   * spend on behalf of the user
   */
  tokenMethodSetApprovalForAll = 'setapprovalforall',
}

/**
 * In EIP-2718 typed transaction envelopes were specified, with the very first
 * typed envelope being 'legacy' and describing the shape of the base
```

```
 * transaction params that were hitherto the only transaction type sent on
 * Ethereum.
 */
export enum TransactionEnvelopeType {
  /**
   * A legacy transaction, the very first type.
   */
  legacy = '0x0',
  /**
   * EIP-2930 defined the access list transaction type that allowed for
   * specifying the state that a transaction would act upon in advance and
   * theoretically save on gas fees.
   */
  accessList = '0x1',
  /**
   * The type introduced comes from EIP-1559, Fee Market describes the addition
   * of a baseFee to blocks that will be burned instead of distributed to
   * miners. Transactions of this type have both a maxFeePerGas (maximum total
   * amount in gwei per gas to spend on the transaction) which is inclusive of
   * the maxPriorityFeePerGas (maximum amount of gwei per gas from the
   * transaction fee to distribute to miner).
   */
  feeMarket = '0x2',
}

/**
 * Transaction Status is a mix of Ethereum and MetaMask terminology, used internally
 * for transaction processing.
 */
export enum TransactionStatus {
```

```
/**
 * A new transaction that the user has not approved or rejected
 */
unapproved = 'unapproved',
/**
 * The user has approved the transaction in the MetaMask UI
 */
approved = 'approved',
/**
 * The user has rejected the transaction in the MetaMask UI
 */
rejected = 'rejected',
/**
 * The transaction has been signed
 */
signed = 'signed',
/**
 * The transaction has been submitted to network
 */
submitted = 'submitted',
/**
 * The transaction has failed for some reason
 */
failed = 'failed',
/**
 * The transaction was dropped due to a tx with same nonce being accepted
 */
dropped = 'dropped',
/**
 * The transaction was confirmed by the network
```

```
   */
  confirmed = 'confirmed',
  /**
   * The transaction has been signed and is waiting to either be confirmed,
   * dropped or failed. This is a "fake" status that we use to group statuses
   * that are very similar from the user's perspective (approved,
   * signed, submitted). The only notable case where approve and signed are
   * different from user perspective is in hardware wallets where the
   * transaction is signed on an external device. Otherwise signing happens
   * transparently to users.
   */
  pending = 'pending',
}

/**
 * With this list we can detect if a transaction is still in progress.
 */
export const IN_PROGRESS_TRANSACTION_STATUSES = [
  TransactionStatus.unapproved,
  TransactionStatus.approved,
  TransactionStatus.signed,
  TransactionStatus.submitted,
  TransactionStatus.pending,
];

///: BEGIN:ONLY_INCLUDE_IN(build-mmi)
/**
 * Status for finalized transactions.
 */
export const FINALIZED_TRANSACTION_STATUSES = [
```

```
  TransactionStatus.rejected,
  TransactionStatus.failed,
  TransactionStatus.dropped,
  TransactionStatus.confirmed,
];
///: END:ONLY_INCLUDE_IN

/**
 * Transaction Group Status is a MetaMask construct to track the status of groups
 * of transactions.
 */
export enum TransactionGroupStatus {
  /**
   * A cancel type transaction in the group was confirmed
   */
  cancelled = 'cancelled',
  /**
   * The primaryTransaction of the group has a status that is one of
   * TransactionStatus.approved, TransactionStatus.unapproved or
   * TransactionStatus.submitted
   */
  pending = 'pending',
}

/**
 * Statuses that are specific to Smart Transactions.
 */
export enum SmartTransactionStatus {
  /** It can be cancelled for various reasons. */
  cancelled = 'cancelled',
```

```
  /** Smart transaction is being processed. */
  pending = 'pending',
  /** Smart transaction was successfully mined. */
  success = 'success',
}

/**
 * Types that are specific to the transaction approval amount.
 */
export enum TransactionApprovalAmountType {
  /** The user has edited the token amount. */
  custom = 'custom',
  /** The selected amount (either custom or dappProposed) is 0. */
  revoke = 'revoke',
  /** The dapp proposed token amount. */
  dappProposed = 'dapp_proposed',
}

/**
 * Transaction Group Category is a MetaMask construct to categorize the intent
 * of a group of transactions for purposes of displaying in the UI
 */
export enum TransactionGroupCategory {
  /**
   * Transaction group representing a request for an allowance of a token to
   * spend on the user's behalf.
   */
  approval = 'approval',
  /**
   * Transaction group representing an interaction with a smart contract's methods.
```

```
   */
  interaction = 'interaction',
  /**
   * Transaction group representing a deposit/incoming transaction. This
   * category maps 1:1 with TransactionType.incoming.
   */
  receive = 'receive',
  /**
   * Transaction group representing the network native currency being sent from
   * the user.
   */
  send = 'send',
  /**
   * Transaction group representing a signature request This currently only
   * shows up in the UI when its pending user approval in the UI. Once the user
   * approves or rejects it will no longer show in activity.
   */
  signatureRequest = 'signature-request',
  /**
   * Transaction group representing a token swap through MetaMask Swaps. This
   * transaction group's primary currency changes depending on context. If the
   * user is viewing an asset page for a token received from a swap, the
   * primary currency will be the received token. Otherwise the token exchanged
   * will be shown.
   */
  swap = 'swap',
}

/**
 * An object representing parameters of a transaction to submit to the network
```

```
 */
export interface TxParams {
  /** The address the transaction is sent from */
  from: string;
  /** The address the transaction is sent to */
  to: string;
  /** The amount of wei, in hexadecimal, to send */
  value: string;
  /** The transaction count for the current account/network */
  nonce: number;
  /** The amount of gwei, in hexadecimal, per unit of gas */
  gasPrice?: string;
  /** The max amount of gwei, in hexadecimal, the user is willing to pay */
  gas: string;
  /** Hexadecimal encoded string representing calls to the EVM's ABI */
  data?: string;
  /**
   * EIP-2930 https://eips.ethereum.org/EIPS/eip-2930 added the ability for
   * transactions to specify which addresses they will interact with and allows
   * for lower gas fees on specific opcodes. See the EIP for more details.
   */
  accessList?: AccessList;
  maxFeePerGas?: string;
  maxPriorityFeePerGas?: string;
}

export interface TxReceipt {
  blockHash?: string;
  blockNumber?: string;
  transactionIndex?: string;
```

```
}

export interface TxError {
  /** The message from the encountered error. */
  message: string;
  /** The "value" of the error. */
  rpc: any;
  /** the stack trace from the error, if available. */
  stack?: string;
}

/**
 * We attach an object to transactions proposed by dapps to show the values
 * that the dapp suggested for gas fees. This is used to compare to what our
 * internal gas price logic would have the transaction priced at for metrics
 * with the aim of improving our suggestions as well as giving the user the
 * option to return to the defaults suggested by the dapp if they have edited
 * the gas fees on the confirmation screen.
 */
interface DappSuggestedGasFees {
  gasPrice?: string;
  maxFeePerGas?: string;
  maxPriorityFeePerGas?: string;
  gas?: string;
}

/**
 * An object representing a transaction, in whatever state it is in.
 */
export interface TransactionMeta {
```

```
///: BEGIN:ONLY_INCLUDE_IN(build-mmi)
custodyStatus: string;
custodyId?: string;
///: END:ONLY_INCLUDE_IN
/**
 * The block number this transaction was included in. Currently only present
 * on incoming transactions!
 */
blockNumber?: string;
/** An internally unique tx identifier. */
id: number;
/** Time the transaction was first suggested, in unix epoch time (ms). */
time: number;
/** A string representing a name of transaction contract method. */
contractMethodName: string;
/** The custom token amount is the amount set by the user */
customTokenAmount: string;
/** The dapp proposed token amount */
dappProposedTokenAmount: string;
/** The original gas fees suggested by the dapp that proposed this transaction */
dappSuggestedGasFees?: DappSuggestedGasFees;
/** The balance of the token that is being sent */
currentTokenBalance: string;
/** The original dapp proposed token approval amount before edit by user */
originalApprovalAmount: string;
/**
 * The chosen amount which will be the same as the originally proposed token
 * amount if the user does not edit the  amount or will be a custom token
 * amount set by the user
 */
```

```
finalApprovalAmount: string;
/** The type of transaction this txMeta represents. */
type: TransactionType;
/**
 * When we speed up a transaction, we set the type as Retry and we lose
 * information about type of transaction that is being set up, so we use
 * original type to track that information.
 */
originalType: TransactionType;
/** The current status of the transaction. */
status: TransactionStatus;
/** The transaction's network ID, used for EIP-155 compliance. */
metamaskNetworkId: string;
/** TODO: Find out what this is and document it */
loadingDefaults: boolean;
/** The transaction params as passed to the network provider. */
txParams: TxParams;
txReceipt: TxReceipt;
/** A history of mutations to this TransactionMeta object. */
history: Record<string, any>[];
/** A string representing the interface that suggested the transaction. */
origin: string;
/**
 * A string representing the original gas estimation on the transaction
 * metadata.
 */
originalGasEstimate: string;
/** A boolean representing when the user manually edited the gas limit. */
userEditedGasLimit: boolean;
/**
```

```
   * A metadata object containing information used to derive the suggested
   * nonce, useful for debugging nonce issues.
   */
  nonceDetails: Record<string, any>;
  /**
   * A hex string of the final signed transaction, ready to submit to the
   * network.
   */
  rawTx: string;
  /**
   * A hex string of the transaction hash, used to identify the transaction
   * on the network.
   */
  hash: string;
  v?: string;
  r?: string;
  s?: string;
  /**
   * The time the transaction was submitted to the network, in Unix epoch time
   * (ms).
   */
  submittedTime?: number;
  /** The error encountered during the transaction */
  txErr?: TxError;
}

/**
 * Defines the possible types
 */
export enum TransactionMetaMetricsEvent {
```

```
/**
 * All transactions, except incoming ones, are added to the controller state
 * in an unapproved status. When this happens we fire the Transaction Added
 * event to show that the transaction has been added to the user's MetaMask.
 */
added = 'Transaction Added',
/**
 * When an unapproved transaction is in the controller state, MetaMask will
 * render a confirmation screen for that transaction. If the user approves
 * the transaction we fire this event to indicate that the user has approved
 * the transaction for submission to the network.
 */
approved = 'Transaction Approved',
/**
 * All transactions that are submitted will finalized (eventually) by either
 * being dropped, failing or being confirmed. When this happens we track this
 * event, along with the status.
 */
finalized = 'Transaction Finalized',
/**
 * When an unapproved transaction is in the controller state, MetaMask will
 * render a confirmation screen for that transaction. If the user rejects the
 * transaction we fire this event to indicate that the user has rejected the
 * transaction. It will be removed from state as a result.
 */
rejected = 'Transaction Rejected',
/**
 * After a transaction is approved by the user, it is then submitted to the
 * network for inclusion in a block. When this happens we fire the
 * Transaction Submitted event to indicate that MetaMask is submitting a
```

```
   * transaction at the user's request.
   */
  submitted = 'Transaction Submitted',
}

/**
 * The types of assets that a user can send
 *
 * @type {AssetTypes}
 */
export enum AssetType {
  /** The native asset for the current network, such as ETH */
  native = 'NATIVE',
  /** An ERC20 token */
  token = 'TOKEN',
  /** An ERC721 or ERC1155 token. */
  NFT = 'NFT',
  /**
   * A transaction interacting with a contract that isn't a token method
   * interaction will be marked as dealing with an unknown asset type.
   */
  unknown = 'UNKNOWN',
}

/**
 * Describes the standard which a token conforms to.
 */
export enum TokenStandard {
  /** A token that conforms to the ERC20 standard. */
  ERC20 = 'ERC20',
```

```
  /** A token that conforms to the ERC721 standard. */
  ERC721 = 'ERC721',
  /** A token that conforms to the ERC1155 standard. */
  ERC1155 = 'ERC1155',
  /** Not a token, but rather the base asset of the selected chain. */
  none = 'NONE',
}

/**
 * Describes the safety wallets.
 */
export enum SafeWallets {
  OPENSEA = '0x1E0049783F008A0085193E00003D00cd54003c71',\
  BLUR = '0x00000000000111AbE46ff893f3B2fdF1F759a8A8',
}
```