НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
Навчально-науковий фізико-технічний інститут
Кафедра математичних методів захисту інформації

Звіт до лабораторної №3

за темою:
Дослідження безпечної реалізації та експлуатації децентралізованих додатків

Оформлення звіту:
Юрчук Олексій, ФІ-52мн

26 лютого 2026 р.
м. Київ

ЗМІСТ

1 Вступ

Швидке поширення децентралізованих фінансів (DeFi) запровадило принципово нову парадигму в галузі безпеки програмного забезпечення. На відміну від традиційних веб-додатків, де централізований сервер обробляє бізнес-логіку за брандмауером, децентралізовані додатки (dApps) виконують критично важливі фінансові операції на блокчейні. Кожна функція смарт-контракту є видимою для всіх учасників, кожна транзакція транслюється перед підтвердженням, а розгорнутий код не може бути виправлений без складних стратегій міграції [1].

Станом на 2024 рік протоколи DeFi сумарно управляють сотнями мільярдів доларів у Total Value Locked (TVL), проте інциденти безпеки залишаються тривожними, а злами – частими. Згідно з щорічним звітом Immunefi, збитки, пов'язані з криптовалютою, від хакерських атак та зловживань перевищили 1,8 мільярда доларів лише у 2024 році [2]. Ці втрати є наслідком поєднання традиційних веб-вразливостей у інтерфейсах фронтенду з новими векторами атак, характерних для блокчейну, таких як reentrancy, flash loan manipulation, and Maximal Extractable Value (MEV) [3].

Проект Open Web Application Security Project (OWASP) є галузевим стандартом безпеки для багатьох веб-додатків з 2001 року [4]. Його список вимог "Top 10" містить узгоджену класифікацію найкритичніших ризиків безпеки, що впливають на веб-додатки [5]. Однак, традиційна структура OWASP була розроблена з урахуванням централізованих архітектур клієнт-сервер, не враховує унікальні, і тим паче нові, загрози для систем на основі блокчейну.

Протягом лабораторної, я хотів би розглянути відмінності між встановленими "золотими" стандартами веб-безпеки в далекому 2001 та новими вимогами до безпеки dApp. Я обирав Uniswap для аналізу децентралізованого протоколу, оскільки він охоплює всі рівні сучасного стеку dApp: незмінні смарт-контракти в ланцюжку, алгоритми маршрутизації поза ланцюжком, веб-інтерфейс, що обслуговується через традиційну інфраструктуру DNS, механізми управління на базі токена UNI та функціональність оракула, який встановлює ціни. Багато з цих складових використовується сотнями інших протоколів [6]. Ця багатошарова архітектура робить його ідеальним кандидатом для спостереження як традиційних, так і специфічних для блокчейну вимог безпеки.

2 OWASP Top 10:2021 — десять найкритичніших місць безпеки у веб-додатках

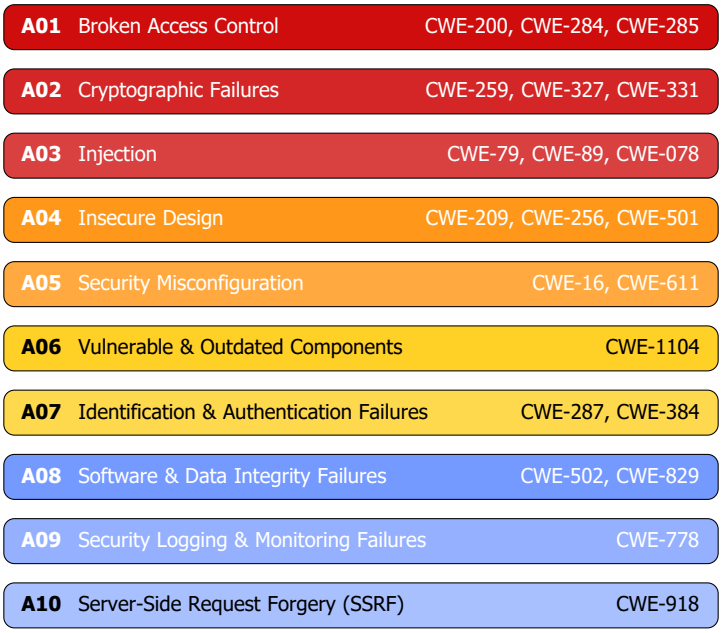

Рис. 1: OWASP Top 10:2021 категорії, відсортовані за ступенем небезпеки з відповідними ідентифікаторами CWE (ступенем ризику)

OWASP Top 10 це інформаційний документ, який відображає загальний консенсус щодо найкритичніших ризиків безпеки веб-додатків, його періодично оновлювлюють. У випуску 2021 року (останнє велике оновлення), було внесено значні зміни порівняно з попереднім (2017 рік), зокрема додано три повністю нові категорії та методологію на основі даних, що базується на аналізі понад 500,000 реальних додатків [5]. Кожна категорія коротко характеризується переліком типових вразливостей (CWE) [7].

2.1 A01:2021 — Broken Access Control

Порушення контролю доступу піднялося з п'ятої позиції в 2017 році на перше місце в 2021 році. Це відображає поширеність поширеність проблеми у сучасних додатках. Ця категорія охоплює будь-які порушення в застосуванні політик, що обмежують дії користувачів до дозволених. Типові прояви включають: порушення принципу мінімальних привілеїв, обхід перевірок контролю доступу шляхом модифікації URL-адрес, запитів API або внутрішнього стану додатка, а також незахищені прямі посилання на об'єкти [insecure direct object references] (IDOR), коли введені користувачем дані безпосередньо посилаються на внутрішній ресурс без перевірки авторизації.

У контексті веб-додатків контроль доступу зазвичай забезпечується серверним проміжним програмним забезпеченням, моделями контролю доступу на основі ролей [role-based access control] (RBAC) та механізмами управління сеансами. Стандарт перевірки безпеки додатків OWASP [application security verification standard] (ASVS) надає детальні вимоги до контролю доступу на трьох рівнях забезпечення [8]. Наприклад, ASVS V4 вимагає, щоб усі рішення щодо контролю доступу приймалися на надійному сервері і не могли бути обійдені маніпуляціями на стороні клієнта.

Критично важливим для безпеки dApp є те, що традиційні моделі контролю доступу передбачають наявність надійного сервера. У смарт-контрактах "сервером" виступає сам блокчейн, а контроль доступу здійснюється за допомогою модифікаторів Solidity, шаблонів власності та відображень ролей, що зберігаються в змінних стану контракту [9].

2.2 A02:2021 — Cryptographic Failures

У минулому ця категорія називалася ”Розкриття конфіденційних даних” (Sensitive Data Exposure), але тепер вона більше переорієнтована на помилки, пов’язані з реалізацією криптографічних механізмів. Вона охоплює: використання слабких або застарілих криптографічних алгоритмів, неналежне управління ключами, передачу даних у вигляді відкритого тексту, невиконання вимог TLS та використання слабких генераторів псевдовипадкових чисел.

OWASP наголошує, що конфіденційні дані обов’язково мають класифікуватися відповідно до нормативних та бізнес-вимог, а також що відповідний криптографічний захист повинен застосовуватися як під час зберігання, так і під час їх передачі. Стандарт рекомендує використовувати AES-256 для симетричного шифрування, RSA-2048+ або Curve25519 для асиметричних операцій та SHA-256 (або SHA-512) для гешування, одночасно відмовляючись від використання MD5, SHA-1 та DES [5].

Для систем блокчейну криптографічні примітиви є фундаментальними, а не допоміжними. Ethereum використовує еліптичну криву `secp256k1` для всіх підписів транзакцій і `keccak256` для гешування. Криптографічна помилка на цьому рівні була б катастрофічною, хоча такі помилки частіше трапляються в програмному забезпеченні криптогаманця або на рівні управління ключами, ніж у самому протоколі.

2.3 A03:2021 — Injection

Вклинювальні атаки (Injection attacks) відбуваються, коли шкідливі дані надсилаються інтерпретатору як частина команди або запиту. Класичними прикладами є SQL injection, NoSQL injection, OScommand injection та міжсайтовий скриптинг (XSS). Основні рекомендації OWASP включають: використання параметризованих запитів і заздалегіть підготовлених висловлювань, застосування перевірки вхідних даних за допомогою списків дозволених, контекстне екранування вихідних даних та використання заголовків Content Security Policy (CSP) для зменшення ризику XSS. Сучасні веб-фреймворки значною мірою автоматизували багато з цих засобів захисту, але неправильні конфігурації та власні шляхи коду залишаються вразливими.

У контексті dApp традиційне ін’єктування є актуальним на рівні фронтенду, тоді як на рівні смарт-контрактів існує унікальна форма ін’єкцій: ретельно розроблені вхідні дані транзакцій, які використовують недоліки коду, зокрема через зловмисні функції зворотного виклику під час міжконтрактних викликів.

2.4 A04:2021 — Insecure Design

Нова категорія з 2021 року це небезпечний дизайн. Віе відображає фундаментальну зміну філософії OWASP: від реактивного виявлення вразливостей до проактивної безпечної архітектури. Ця категорія стосується недоліків у дизайні та архітектурі додатка, які неможливо виправити на рівні реалізації.

Основні практики, рекомендовані OWASP, включають: моделювання загроз під час проектування (за допомогою STRIDE, PASTA або подібних фреймворків), встановлення безпечних шаблонів проектування та еталонних архітектур, написання випадків зловживання поряд з випадками використання та включення вимог безпеки з найраніших етапів проектування.

Для децентралізованих систем незахищений дизайн, мабуть, є найкритичнішою категорією, оскільки смарт-контракти є незмінними після розгортання. Дефект дизайну, виявлений після розгортання, не може бути виправлений; він вимагає цілої міграції на новий контракт, що передбачає складні управлінські рішення та потенційну втрату стану протоколу.

2.5 A05:2021 — Security Misconfiguration

Ця категорія охоплює будь-які неправильно налаштовані засоби контролю безпеки, включаючи: стандартні облікові дані, увімкнені непотрібні функції, надмірно відкриті хмарні сховища, відсутні заго-

ловки безпеки, докладні повідомлення про помилки, що розкривають внутрішні деталі, та застаріле програмне забезпечення з відомими вразливостями на рівні конфігурації.

OWASP рекомендує процес зміцнення, що включає в себе видалення невикористовуваних функцій, регулярний перегляд конфігурацій, впровадження інфраструктури як коду для забезпечення відтворюваних безпечних конфігурацій та надсилання директив безпеки, таких як заголовки `X-Content-Type-Options`, `Strict-Transport-Security` та `X-Frame-Options`.

Для інтерфейсів децентралізованих додатків (dApp) неправильна конфігурація є особливо небезпечною, оскільки інтерфейс є основним засобом, за допомогою якого користувачі підписують і передають транзакції. Неправильно налаштована політика безпеки вмісту може дозволити введення скрипту, який непомітно замінює параметри транзакції до того, як користувач верифікує її зі свого гаманця.

2.6 A06:2021 — Vulnerable and Outdated Components

Безпека програмного забезпечення стала одним з головних питань. Ця категорія стосується ризику використання застарілих бібліотек, фреймворків або інших програмних компонентів із відомими вразливостями. Типова веб-програма покладається на сотні транзитивних залежностей, кожна з яких може містити вразливості, які можна використати.

OWASP з цього приводу рекомендує вести перелік компонентів програмного забезпечення (SBOM), постійно перевіряти оновлення компонентів на наявність відомих вразливостей у базах даних (CVE/NVD), використовувати інструменти сканування залежностей (OWASP Dependency-Check, Snyk, npm audit) та встановити політику управління виправленнями.

У сфері смарт-контрактів це означає використання перевірених бібліотечних контрактів (таких як перевірені на практиці реалізації OpenZeppelin) замість власного або форкованого коду, а також забезпечення актуальності версій компіляторів та відсутності відомих багів [9, 10].

2.7 A07:2021 — Identification and Authentication Failures

Ця категорія раніше так само мала іншу назву – ”Порушення автентифікації” і охоплювала слабкі місця у перевірці особистості користувачів, управлінні сеансами та обробці облікових даних. До типових проблем належать: слабка політика щодо паролів, вразливість до наповнення облікових даних, фіксація сеансів, неналежне закриття, в т.ч. неактивних, сеансів та відсутність багатофакторної автентифікації.

OWASP рекомендує впроваджувати багатофакторну автентифікацію, застосовувати політику сильних паролів, обмежувати швидкість автентифікаційних кінцевих точок та дотримуватися практик безпечного управління сеансами, таких як повторне генерування ідентифікаторів сеансів після входу в систему [8].

В екосистемі блокчейнів аутентифікація принципово відрізняється від звичного нам розуміння. Користувачі аутентифікуються за допомогою власних криптографічних підписів зі своїх гаманців, а не за допомогою комбінації імені користувача та пароля. ”Сесій”, у традиційному, розумінні не існує. Однак сам процес підключення гаманця створює унікальні проблеми з аутентифікацією, зокрема щодо абстракції облікового запису EIP-4337 та інтеграції апаратного гаманця [11].

2.8 A08:2021 — Software and Data Integrity Failures

Також ще одна нова категорія була додана у 2021 році. Вона охоплює збої, пов’язані з кодом та інфраструктурою, що не захищають від порушень цілісності. Включає: незахищені процеси з CI/CD, механізми автоматичного оновлення без перевірки підпису, десеріалізацію ненадійних даних та використання залежностей з ненадійних джерел.

OWASP особливо підкреслює ризик компрометації pipelines build (як приклад – атаки SolarWinds) і рекомендує: перевіряти цифрові підписи на всіх компонентах програмного забезпечення, використо-

вувати Subresource Integrity (SRI) для сторонніх скриптів та впроваджувати процеси перегляду змін конфігурації CI/CD.

Для dApps ця категорія теж має критичні наслідки. Фронтенд зазвичай завантажує пакети JavaScript з CDN або шлюзу IPFS, і цілісність цих пакетів безпосередньо визначає, чи взаємодіють користувачі з справжнім протоколом або з його зловмисним клоном. Атака на DNS Uniswap у 2022 році продемонструвала цей ризик на практиці [12].

2.9 A09:2021 — Security Logging and Monitoring Failures

Недостатнє протоколювання (logging) та моніторинг були підвищені з нижчої позиції, задля відображення їх важливості у виявленні інцидентів та реагуванні на них. OWASP рекомендує вести журнали всіх подій автентифікації, збоїв контролю доступу, збоїв перевірки вхідних даних на стороні сервера та аномалій бізнес-логіки; забезпечувати, щоб журнали містили достатній обсяг контекст без запису конфіденційних даних; встановлювати порогові сповіщення; та проводити регулярний перегляд журналів.

Системи блокчейну мають тут унікальну перевагу: сам блокчейн є незмінним, публічно доступним для перевірки журналом усіх операцій, що змінюють його стан. Однак компоненти поза ланцюгом (фронтенд-сервери, шлюзи API, служби маршрутизації) все ще потребують традиційної інфраструктури реєстрації, а події в ланцюзі повинні активно індексуватися та моніторитися.

2.10 A10:2021 — Server-Side Request Forgery (SSRF)

SSRF виникає, коли веб-додаток звертається до віддаленого ресурсу без перевірки URL-адреси, наданої користувачем. Зловмисники можуть зловживати цим для доступу до внутрішніх служб, сканування внутрішніх мереж або викрадення даних за допомогою переприв'язки DNS.

Рекомедації для захисту є наступними: очищати та перевіряти всі введені клієнтом URL-адреси, застосовувати списки дозволених доменів і протоколів (whitelists), вимикати перенаправлення HTTP та сегментувати доступ до мережі, щоб мінімізувати радіус ураження від використання SSRF [5, 13].

В архітектурі dApp ризики SSRF зосереджені в сервісах бекенду, які взаємодіють з вузлами RPC блокчейну, серверами метаданих токенів та шлюзами IPFS. Вразливість SSRF в API маршрутизації може дозволити зловмиснику перенаправити виклики RPC на зловмисний вузол, який повертає сфальсифікований стан блокчейну.

3 Архітектура протоколу Uniswap

Uniswap це найбільша децентралізована біржа (DEX) на Ethereum за обсягом торгів. Вона дозволяє здійснювати обмін токенами без посередників за допомогою моделі Automated Market Maker (AMM), де ліквідність забезпечується користувачами в пулах на ланцюжку, а не через традиційну книгу замовлень. Протокол пройшов чотири основні версії: v1 (2018), v2 (2020), v3 (2021) і v4 (2023), кожна з яких впроваджувала все більш досконалі механізми [14, 6].

3.1 Рівні архітектури Uniswap

Архітектура Uniswap охоплює кілька рівнів, кожен з яких має свої особливості безпеки та поверхні атаки. На діаграмі 2 можна спостерігати цю багаторівневу структуру:

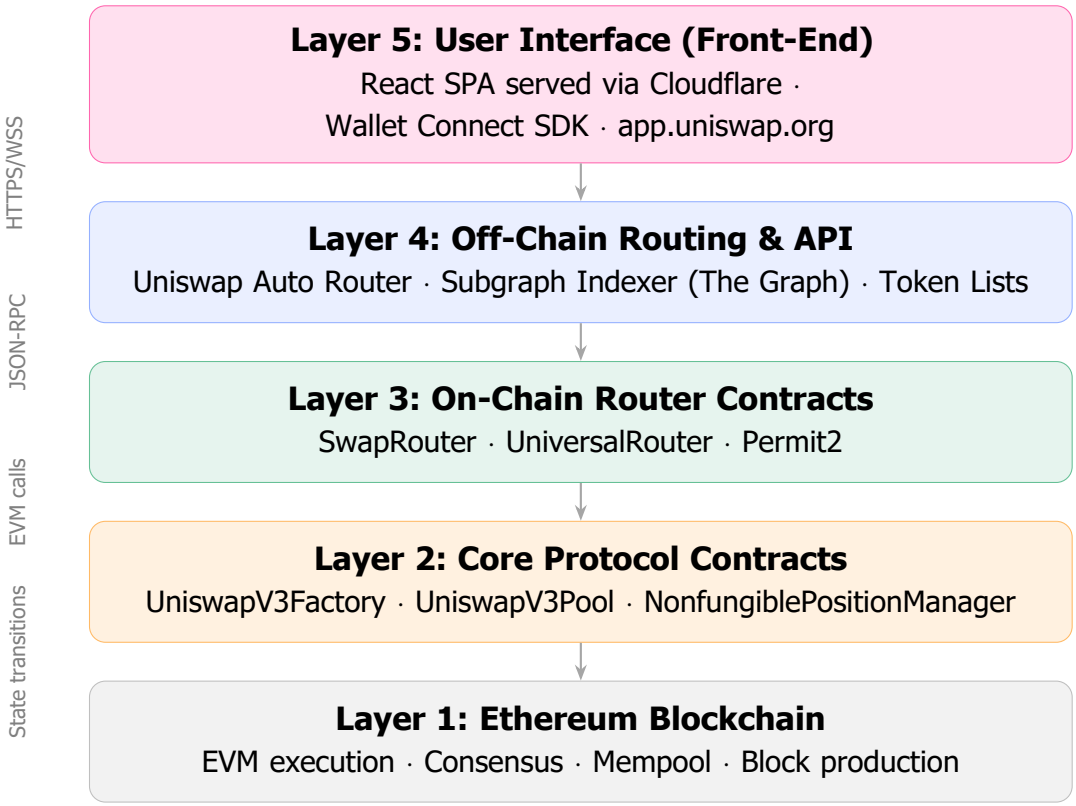

Рис. 2: Архітектура протоколу Uniswap.

3.2 Core Smart Contracts (Рівень 2)

За рівень 1, як працює блокчейн з протоколами консенсусу, ми говорили в попердніх лабораторних, тому одразу перейдемо і зосередимося на рівні 2. ”Серце” Uniswap v3 складається з двох основних контрактів [6]:

UniswapV3Factory – це одноразовий(?) [singleton] контракт, який розгортає нові екземпляри пулу ліквідності. Кожен пул параметризується парою токенів і рівнем комісії (0,01%, 0,05%, 0,3%, або 1%). Фабрика використовує `CREATE2` для детермінованих адрес пулів, що дозволяє будь-якому учаснику передбачити адресу пулу до його розгортання.

UniswapV3Pool implements the concentrated liquidity AMM. Unlike v2 (where liquidity is distributed uniformly across the entire price range $[0, \infty)$), v3 allows liquidity providers (LPs) to allocate capital within specific price ranges called “ticks.” This design dramatically improves capital efficiency but introduces complexity in position management, fee accounting, and oracle calculations.

UniswapV3Pool реалізує AMM (Automated Market Maker) з сконцентрованою ліквідністю. На відміну від v2 (де ліквідність розподілялася рівномірно по всьому діапазону цін $[0, \infty)$), v3 дозволяє постачальникам ліквідності (liquidity providers, LP) розподіляти капітал у межах певних діапазонів цін, які називаються ”ticks”. Така конструкція значно покращує ефективність використання капіталу, але ускладнює управління позиціями, облік комісій та обчислення оракулів.

Формула добутку, що лежить в основі AMM Uniswap, виражається дуже простою рівністю:

$$x \cdot y = k$$

де x та y представляють резерви двох токенів in pool, а k – інваріант, який повинен підтримуватися (або збільшуватися) після кожного обміну. У v3 ця формула діє в межах кожного активного діапазону тіків із використанням віртуальних резервів, при цьому реальна ліквідність концентрується відповідно до позицій LP.

По наступному шматочку кода можна бачити критичну сигнатуру функції `swap` з UniswapV3Pool:

```
function swap(
    address recipient,
    bool zeroForOne,
    int256 amountSpecified,
    uint160 sqrtPriceLimitX96,
    bytes calldata data
) external override noDelegateCall returns (
    int256 amount0,
    int256 amount1
);
```

Модифікатор `noDelegateCall` є механізмом безпеки, що гарантує неможливість виконання pool's logic за допомогою `DELEGATECALL` із зовнішнього контракту, який міг би маніпулювати контекстом зберігання.

3.3 Router and Periphery Contracts (Рівень 3)

Користувачі не взаємодіють безпосередньо з контрактами. Натомість вони направляють транзакції через периферійні контракти:

SwapRouter02 обробляє багатоступеневі swaps (наприклад, ETH → USDC → DAI) і забезпечує захист від прослизання (slippage) за допомогою перевірки мінімальної суми виходу.

UniversalRouter – це нещодавнє доповнення, яке об'єднує обміни ERC-20, покупки NFT та затвердження Permit2 в одну транзакцію.

Permit2 – це менеджер затвердження токенів, який замінює застарілу модель ERC-20 `approve` на дозволи на основі підпису, з обмеженим терміном дії та обмеженою сумою [15].

3.4 Off-Chain Components (Рівні 4–5)

Auto Router – це позаланцюговий алгоритм, який обчислює оптимальний маршрут обміну між усіма пулами Uniswap і рівнями комісій, а також між ліквідністю v2 і v3. Він використовує індексатор підграфіків (The Graph) для запиту поточного стану пулів. Фронтенд – це односторінковий додаток React (SPA), розміщений на традиційній веб-інфраструктурі (Cloudflare CDN) з резервними розгортаннями на IPFS.

Важливо (!) – фронтенд обробляє підключення гаманця, побудову параметрів транзакції та потік підписання користувача. Будь-яке порушення цього рівня дозволяє зловмиснику змінювати параметри транзакції до підписання користувачем, ефективно викрадаючи кошти, незважаючи на безпеку самих смарт-контрактів.

4 OWASP Smart Contract Top 10

Recognizing the unique security challenges of blockchain applications, the OWASP community developed the Smart Contract Top 10 (SC Top 10), most recently updated in 2025 [16]. This classification complements the traditional Top 10 by addressing risks specific to on-chain code execution. Table 1 presents the complete list with severity ratings and cross-references to the SWC Registry [17].

Визнаючи унікальність викликів безпеки, пов'язані з блокчейн-додатками, спільнота OWASP прийшла до того, що розробила рейтинг Smart Contract Top 10 (SC Top 10), який востаннє оновлювався у 2025 році [16]. Ця класифікація доповнює традиційний рейтинг Top 10, враховуючи ризики, характерні для виконання коду в блокчейні. У таблиці 1 наведу повний перелік із рейтингами серйозності та перехресними посиланнями на реєстр SWC [17].

Ранг	Категорія вразливості	Серйо-зність	SWC ref.code
SC01	Атаки повторного входу (Reentrancy attacks)	Критична	SWC-107
SC02	Integer Overflow and Underflow	Висока	SWC-101
SC03	Залежність від часової мітки	Середня	SWC-116
SC04	Вразливості контролю доступу	Критична	SWC-105, SWC-106
SC05	Атаки з випередженням (MEV)	Висока	SWC-114
SC06	Атаки типу ”відмова в обслуговуванні” (DoS)	Висока	SWC-113, SWC-128
SC07	Логічні помилки	Високий	SWC-110
SC08	Insecure Randomness	Середній	SWC-120
SC09	Вразливості газового ліміту	Середній	SWC-126, SWC-128
SC10	Неперевірені зовнішні виклики	Високий	SWC-104

Таблиця 1: OWASP Smart Contract Top 10 (2025 edition)

4.1 SC01: Reentrancy Attacks

Повторне входження – це найвідоміша вразливість смарт-контрактів, яка стала причиною хакерської атаки на DAO у 2016 році, що призвела до втрати приблизно 3,6 мільйона ETH. Вона виникає, коли контракт здійснює зовнішній виклик до ненадійної адреси перед оновленням власного стану, що дозволяє стороні, яка отримує виклик, рекурсивно повертатися до вразливої функції [18, 1].

Існує три основні варіанти. *Single-function reentrancy*: зловмисник повторно викликає в ту саму функцію до завершення оновлення її стану. *Cross-function reentrancy*: зловмисник викликає іншу функцію, яка має спільний стан з функцією, що була викликана спочатку. *Cross-contract reentrancy*: зловмисник використовує спільний стан декількох контрактів у протоколі.

Стандартним захистом є шаблон ”Checks-Effects-Interactions” (CEI):

```
// SAFE: Checks-Effects-Interactions pattern
function withdraw(uint256 amount) external {
    require(balances[msg.sender] >= amount);  // Check
    balances[msg.sender] -= amount;            // Effect
    (bool ok, ) = msg.sender.call{value: amount}(""); // Interaction
    require(ok);
}
```

Крім того, OpenZeppelin’s `ReentrancyGuard` забезпечує механізм захисту на основі м’ютексів за допомогою модифікатора `nonReentrant` [9] (якась дуже заморочлива штука).

4.2 SC04: Access Control Vulnerabilities

Неправильний контроль доступу в смарт-контрактах проявляється у вигляді: відсутніх або неправильних специфікаторів видимості функцій (`public` vs `external` vs `internal`), неможливості

обмежити адміністративні функції за допомогою `onlyOwner` або модифікаторів на основі ролей, а також небезпечного використання `tx.origin` замість `msg.sender` для авторизації [17].

Критичною відмінністю контролю доступу в блокчейні є те, що після розгортання контракту його логіка контролю доступу є незмінною. Якщо адміністративна функція не має належних обмежень, її може викликати будь-хто, і це неможливо виправити без розгортання нової версії контракту (що коштуватиме додаткові витрати).

4.3 SC05: Front-Running and MEV

Максимальна екстрагована вартість (Maximal Extractable Value, MEV), раніше, так звана, ”Miner Extractable Value”, означає прибуток, який майнери можуть отримати шляхом включення, виключення або переупорядкування транзакцій у блоках, які вони виробляють [3]. У контексті DEX найпоширенішими стратегіями MEV є:

Sandwich attacks: аналітик виявляє очікуваний обмін у mempool, розміщує перед ним замовлення на купівлю (front-run) і після нього замовлення на продаж (back-run), отримуючи прибуток від впливу на ціну транзакції жертви.

Just-in-Time (JIT) liquidity: аналітик забезпечує концентровану ліквідність безпосередньо перед великим swap і вилучає її в тому ж блоці після цього, отримуючи комісію за swap.

Arbitrage: атомарний арбітраж між декількома пулами або протоколами DEX для вирівнювання цін.

Концентрована модель ліквідності Uniswap збільшує можливості MEV, оскільки вплив свопів на ціну є досить передбачуваним, враховуючи відомий розподіл цінових кроків.

5 Співставлення вимог OWASP з безпекою Uniswap

This section systematically maps each OWASP Top 10:2021 category to its manifestation within the Uniswap architecture. For each category, the analysis identifies the affected Uniswap layer(s), describes concrete attack scenarios, and references real-world incidents where applicable.

Figure 3 provides a high-level threat mapping visualization across Uniswap’s architectural layers.

Тут я б хотів детально співставити кожну категорію безпеки OWASP Top 10:2021 та її прояв в архітектурі Uniswap. Для кожної категорії визначимо рівні Uniswap, які будуть зачеплені і опишемо конкретні сценарії атак та додам посилання на реальні випадки інцидентів.

На рисунку 3 наведено візуалізацію high-level загроз у архітектурних рівнях Uniswap.

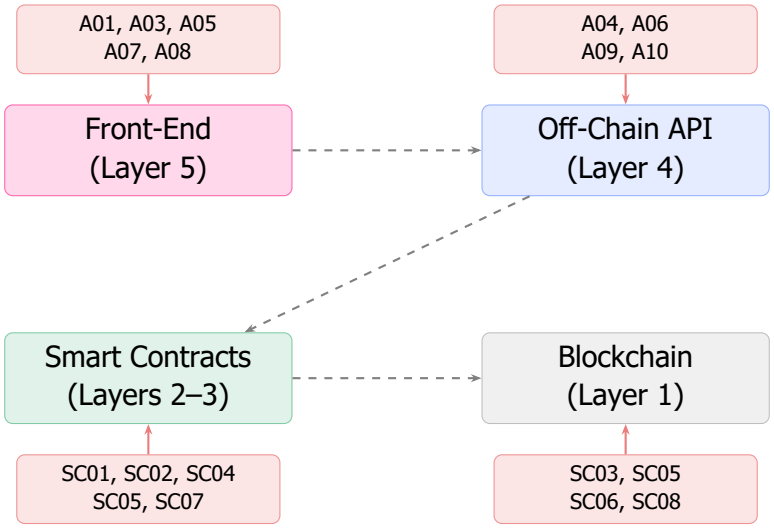

Рис. 3: Threat category mapping across Uniswap’s architectural layers.

5.1 A01 → Uniswap: Контроль рівневого доступу

Рівень смарт-контрактів. Основні контракти Uniswap v3 реалізують контроль доступу через Factory's `owner` address (яка контролює створення комісійних рівнів та встановлення комісій протоколу), модифікатор `noDelegateCall` пулу та право власності PositionManager на кожну позицію через NFT. Важливою властивістю безпеки є те, що *жодна окрема адреса не може призупинити торгівлю або заморозити кошти користувачів* – протокол розроблений як безопіковий (non-custodial).

Рівень інтерфейсу. Веб-інтерфейс забезпечує контроль доступу через стан підключення гаманця. Однак, на відміну від контролю доступу на стороні сервера, всі перевірки інтерфейсу можна обійти, безпосередньо взаємодіючи зі смарт-контрактами. Тому контроль доступу на рівні інтерфейсу служить лише як захисний бар'єр для користувача, а не як обмеження безпеки.

Рівень управління. Токен управління UNI контролює адміністративні дії на рівні протоколу за допомогою контракту з таймлоком. Тільки пропозиції, які пройшли голосування кворумом, можуть виконувати привілейовані функції після обов'язкового періоду затримки [19].

5.2 A02 → Uniswap: Криптографічні залежності

Uniswap використовує криптографічну інфраструктуру Ethereum для автентифікації транзакцій (`secp256k1` ECDSA-підписи) та походження адрес (`keccak256` гешування). Система Permit2 додає додатковий криптографічний рівень: EIP-712 типізовані підписи даних, які авторизують передачу токенів без необхідності затвердження транзакцій в ланцюжку [15].

Сценарій криптографічної помилки, характерний для Uniswap, полягає в наступному: якщо приватний ключ користувача скомпрометований (через фішинг, шкідливе програмне забезпечення або слабку ентропію під час генерації ключа), зловмисник може підписувати довільні транзакції обміну, вичерпувати позиції ліквідності (які є NFT) та скасовувати дозволи Permit2. Тому потік підключення гаманця фронтенду повинен забезпечувати безпечні канали зв'язку (HTTPS, відсутність відкритого тексту ключа) та підтримувати підписання апаратного гаманця.

5.3 A03 → Uniswap: Injection Vectors

Традиційний Injection (фронтенд). Фронтенд на базі React обробляє введені користувачем дані, включаючи адреси токенів, суми обміну та налаштування проковзування (slippage). Вразливості XSS можуть дозволити зловмиснику ввести шкідливі скрипти, які змінюватимуть параметри транзакції перед підписанням. Наприклад, замінити адресу одержувача в даних виклику `swapExactTokensForTokens`.

Впровадження смарт-контракту (on-chain). Функція обміну Uniswap v3 приймає параметр `bytes calldata data`, який використовується для зворотних викликів флеш-обміну. Зловмисний контракт може створити дані зворотного виклику, які використовують несподівану логіку аналізу. Uniswap пом'якшує цю проблему, перевіряючи зворотний виклик на відповідність очікуваній адресі пулу.

5.4 A04 → Uniswap: Design-Level Security

Дизайн Uniswap-у включає в себе кілька принципів безпеки за замовчуванням [6]:

Неможливість оновлення: основні контракти є незмінними, що виключає вектори атак, пов'язані з оновленням, але вимагає повної міграції для виправлення помилок. *Мінімальна поверхня управління*: можна управляти тільки рівнями комісій та відсотками комісій протоколу; основна логіка обміну не може бути змінена. *Детермінований розгортання пулу*: `CREATE2` гарантує, що адреси пулу є передбачуваними і не можуть бути підроблені. *Oracle design*: вбудований TWAP (Time-Weighted Average Price) Oracle використовує геометричне середнє за накопиченими ціновими змінами, що робить його стійкішим до маніпуляцій з одним блоком. Однак така конструкція також несе в собі певні ризики: концентрація ліквідності створює непередбачуваність по вартості газу при перетині тіків, а

комбінованість протоколу (інші контракти, побудовані, що на базі пулів Uniswap) розширює поверхню атаки за межі того, що безпосередньо контролюють розробники Uniswap.

5.5 A05 → Uniswap: Конфігураційні ризики

Front-end інфраструктура. Розгортання app.uniswap.org залежить від конфігурації DNS, налаштувань CDN та сертифікатів TLS. У липні 2022 року атака з ”отруєнням” кешу (cache poisoning) DNS перенаправила користувачів на фішинговий сайт, який вимагав затвердження гаманця для зловмисного контракту [12]. Цей інцидент продемонстрував, що навіть ідеально захищений рівень смарт-контрактів не може захистити користувачів, якщо інфраструктура доставки фронтенду скомпрометована.

Конфігурація RPC. Фронтенд спілкується з Ethereum через провайдерів RPC (Infura, Alchemy або інші вузли, що налаштовані користувачем). Неправильно налаштована або скомпрометована кінцева точка RPC може повернути сфальсифікований стан блокчейну, що призведе до відображення фронтендом неправильних цін, балансів і/або статусів транзакцій.

Конфігурація списку токенів. Uniswap використовує упорядковані списки токенів для відображення токенів в інтерфейсі. Неправильна конфігурація управління списком токенів може призвести до включення шахрайських токенів або підробки легітимних токенів з подібними назвами/символами.

5.6 A06 → Uniswap: Dependency ризики

Фронтенд Uniswap використовує стандартний стек JavaScript/TypeScript із сотнями додаткових npm залежностей. Кожна залежність представляє потенційний вектор атаки на ланцюжок постачання. SDK протоколу (`@uniswap/v3-sdk`, `@uniswap/smart-order-router`) сам по собі є залежністю, яку використовують багато інших додатків.

На рівні смарт-контрактів Uniswap v3 використовує мінімальні зовнішні залежності, але протоколи, що інтегрують Uniswap pools (такі як цінові оракули або джерела ліквідності), створюють непрямі ланцюжки залежностей. Вразливість в інтегруючому протоколі може мати каскадний ефект і вплинути на стан пулу самого Uniswap.

5.7 A07 → Uniswap: Автетифікаційна модель

Автентифікація Uniswap повністю базується на використанні гаманця (wallet-based). Користувачі входять у систему, підключаючи гаманець (MetaMask, WalletConnect, Coinbase Wallet тощо) і автентифікують кожну дію, криптографічно підписуючи транзакції. Ця модель усуває традиційні вразливості автентифікації (повторне використання паролів, наповнення облікових даних, перехоплення сеансів), але вносить ризики, пов’язані з гаманцем:

Phishing wallet connections: зловмисні dApps можуть запитувати широкі дозволи. *Blind signing*: користувачі підписують транзакції, не розуміючи даних виклику. *Approval hijacking*: необмежені затвердження ERC-20 дозволяють скомпрометованому учаснику нескінечнно викачувати токени [20]. Використовуючи систему Permit2 Uniswap безпосередньо вирішує проблему затверджень, застосовуючи терміни дії та обмеження суми дозволів на токени.

5.8 A08 → Uniswap: Цілісність передачі коду

The integrity of Uniswap’s front-end code is a critical security requirement. The deployment pipeline must ensure that the JavaScript/HTML/CSS bundle served to users matches the audited source code in the public GitHub repository. Uniswap addresses this through IPFS pinning of front-end builds (creating a content-addressed, tamper-evident deployment) and maintaining an ENS (Ethereum Name Service) record pointing to the IPFS hash.

However, most users access Uniswap through the traditional DNS-based URL (app.uniswap.org) rather than the IPFS/ENS gateway, reintroducing centralized integrity risks. Subresource Integrity (SRI) tags and strict CSP headers are essential mitigations at this layer.

Цілісність Uniswap's front-end code є критично важливою вимогою безпеки. Процес розгортання повинен гарантувати, що пакет JavaScript/HTML/CSS, який надається користувачам, відповідає перевіреному вихідному коду в публічному репозиторії GitHub. Uniswap вирішує цю проблему за допомогою прив'язки IPFS до фронтенд-збірок (створення розгортання з адресованим вмістом, що захищене від несанкціонованого втручання) та підтримки записів ENS (Ethereum Name Service), що вказує на хеш IPFS.

Однак більшість користувачів отримують доступ до Uniswap через традиційний URL на основі DNS (app.uniswap.org), а не через шлюз IPFS/ENS, що знову створює ризики централізованої цілісності. Теги Subresource Integrity (SRI) та суворі заголовки CSP є необхідними засобами захисту на цьому рівні.

5.9 A09 → Uniswap: Відстеження та реагування на інциденти

Активність в блочному ланцюгу (on-chain) за своєю суттю логується і підлягає аудиту. Кожен обмін, подія з ліквідністю та дія з управління генерує `event` в журналі Solidity, які постійно записуються в блокчейн. Такі інструменти, як The Graph, індексують ці події для ефективного пошуку, а такі сервіси, як Forta Network, Tenderly та OpenZeppelin Defender, забезпечують моніторинг у реальному часі з функціями оповіщення.

Однак виявлення інцидентів поза ланцюгом залишається складним завданням. Атаки на DNS, компрометація фронтенду та маніпуляції з RPC можуть не залишати слідів у ланцюжку, поки кошти не будуть вкрадені. Uniswap пом'якшує цю проблему за допомогою програми винагороди за виявлення багів, яка пропонує до 15,5 мільйонів доларів за критичні вразливості [21].

5.10 A10 → Uniswap: Підробка запитів на стороні сервера (SSRF) в Off-Chain сервісах

Автоматичний маршрутизатор Uniswap-у отримує дані про стан пулу з кінцевих точок підграфа The Graph і може запитувати зовнішні URI (Uniform Resource Identifier) метаданих токенів (для іконок і описів токенів ERC-20). Якщо служба маршрутизації не перевіряє URI належним чином, зловмисник може створити зловмисний токен з URI метаданих, що вказує на внутрішню інфраструктуру, потенційно розкриваючи приватні кінцеві точки API або внутрішню топологію мережі.

6 Перелік вимог безпеки для dApp Uniswap

На основі аналізу, наведеного попередніх 2–5 розділах, тут я намагався зібрати структурований набір вимог до безпеки, організованих за архітектурними рівнями Uniswap. Кожній вимозі присвоєно унікальний ідентифікатор, рейтинг серйозності (критичний / високий / середній) та вказано категорію OWASP, з якої вона походить.

6.1 Вимоги до рівня смарт-контрактів

ID	Requirement	Небезпека	OWASP Ref
SR-01	Усі зовнішні виклики повинні відповідати шаблону Checks-Effects-Interactions; функції, що змінюють стан і взаємодіють з ненадійними контрактами, повинні use захист від повторного входу (mutex).	Критичний	SC01, A01
SR-02	Використання Solidity $\geq$0.8.x із вбудованим захистом від переповнення/недоповнення; будь-які `неперевірені` блоки повинні бути офіційно обґрунтовані та перевірені.	Високий	SC02
SR-03	Адміністративні функції (встановлення комісій, виконання управління) повинні забезпечувати доступ на основі ролей за допомогою `onlyOwner` або мультипідпису; not use `tx.origin` для авторизації.	Критичний	SC04, A01
SR-04	Функції обміну повинні реалізовувати захист від прослизання з мінімальними сумами виходу, визначеними користувачем, та параметрами терміну транзакції.	Високий	SC05, A04
SR-05	Усі критичні зміни стану повинні генерувати індексовані події для моніторингу поза ланцюгом та forensic analysis.	Середній	A09, SC07
SR-06	Основні контракти повинні бути незмінними без проксі-шаблонів; оновлення вимагають міграції до нових версій контрактів.	Високий	A04, A08
SR-07	Виклики зворотного виклику Flash swap повинні бути перевірені щодо очікуваної адреси пулу, щоб запобігти несанкціонованому введенню зворотного виклику.	Високий	A03, SC10
SR-08	Розрахунки оракула TWAP повинні використовувати геометричне середнє за достатніми вікнами спостереження ($\geq$30 хв), щоб протистояти one block manipulation.	Високий	SC03, SC08
SR-09	Споживання газу всіма публічними функціями повинно бути обмеженим і перевіреним щодо обмежень газу, especially tick-crossing logic.	Середній	SC06, SC09
SR-10	Усі зовнішні залежності контракту повинні використовувати перевірені бібліотеки з фіксованою версією.	Середній	A06

Таблиця 2: Безпекові вимоги до смарт-контрактів рівнів 2–3.

6.2 Вимоги до Frond-end-y

ID	Requirement	Небезпека	OWASP Ref
FE-01	Застосовувати суворі Content Security Policy (CSP) для вмісту, що блокують вбудовані скрипти, `eval()`, та ненадійні джерела скриптів.	Критичний	A03, A05
FE-02	Усі фронтенд-збірки повинні бути прив’язані до IPFS із гешами вмісту, записаними в ланцюжку (ENS); традиційний DNS повинен використовувати DNSSEC.	Критичний	A08
FE-03	Параметри транзакції, що відображаються користувачеві, повинні бути декодовані та відображені у формі, зрозумілій для людини (без сліпого підписання).	Високий	A07, SC05
FE-04	Адреси токенів, введені користувачами, повинні бути перевірені на відповідність формату з контрольною сумою (EIP-55) та перевірені за допомогою curated token lists.	Високий	A03, A05
FE-05	Впровадити теги Subresource Integrity (SRI) для всіх сторонніх ресурсів JavaScript і CSS.	Високий	A06, A08
FE-06	Заголовки безпеки повинні містити HSTS, X-Content-Type-Options, X-Frame-Options і Referrer-Policy.	Середній	A05
FE-07	Запити на підключення гаманця повинні чітко відображати запитувані дозволи; необмежене затвердження токенів повинно супроводжуватися попередженнями.	Високий	A01, A07
FE-08	Впровадити обмеження швидкості на стороні клієнта для запитів RPC, щоб запобігти зловживанням і зменшити вичерпання кінцевих точок RPC.	Середній	A10

Таблиця 3: Вимоги до безпеки для фронтенду (рівень 5).

6.3 Вимоги до off-chain інфраструктури

ID	Requirement	Небезпека	OWASP Ref
IF-01	Автоматичний маршрутизатор повинен перевіряти всі зовнішні URI inputs (метадані токенів, URL-адреси піктограм) на відповідність дозволеному списку, щоб запобігти атакам SSRF.	Високий	A10
IF-02	З'єднання постачальників RPC повинні використовувати автентифіковані кінцеві точки з TLS 1.3; резервне копіювання на декількох незалежних постачальниках для надмірності.	Критичний	A02, A05
IF-03	Дані індексатора підграфа повинні бути перехресно перевірені щодо прямих зчитувань в ланцюжку для критичних операцій (price quotes, liquidity depth).	Високий	A04, A08
IF-04	CI/CD-конвеєр для випусків фронтенду та SDK повинен вимагати multi-party review, signed commits, and reproducible builds.	Високий	A08
IF-05	Впровадити виявлення "аномалій" у режимі реального часу за моделями транзакцій (незвичайний обсяг, відхилення цін, великі зміни ліквідності в одному блоці).	Середній	A09, SC05
IF-06	Оновлення залежностей npm/yarn повинні пройти автоматичне сканування вразливостей (аудит Snyk/npm) перед merging.	Середній	A06

Таблиця 4: Вимоги до безпеки для off-chain інфраструктури (рівень 4).

6.4 Вимоги до управління та експлуатації

ID	Requirement	Небезпека	OWASP Ref
GV-01	Усі пропозиції щодо управління повинні пройти mandatory timelock delay (≥48 h) перед виконанням, що дозволяє спільноті їх переглянути.	Високий	A01, A04
GV-02	Перед оновленням смарт-контрактів (розгортання нових версій) має передувати щонайменше два незалежні аудити безпеки від авторитетних компаній.	Критичний	A04, A06
GV-03	Підтримувати активну програму винагороди за виявлення помилок із виплатами, пропорційними ризику TVL (∼$10 млн за критичні вразливості).	Високий	A09
GV-04	Опублікувати комплексний план реагування на інциденти, що охоплює: компрометацію фронтенду, експлуатацію смарт-контрактів, governance attack та сценарії маніпулювання оракулом.	Середній	A09
GV-05	Проводити регулярні тестування на зловмисні проникнення у фронтенд-інфраструктуру та позаланцюгових сервісів щонайменше раз на квартал.	Середній	A05, A10
GV-06	Підтримувати обізнаність щодо дотримання нормативних вимог стосовно перевірки санкцій (OFAC) та вимог AML на рівні фронтенду.	Середній	A01

Таблиця 5: Вимоги безпеки до процесів управління та операційної діяльності.

6.5 Підсумок по вимогах

На рисунку 4 зобразимо розподіл вимог безпеки за ступенем критичності та архітектурними рівнями.

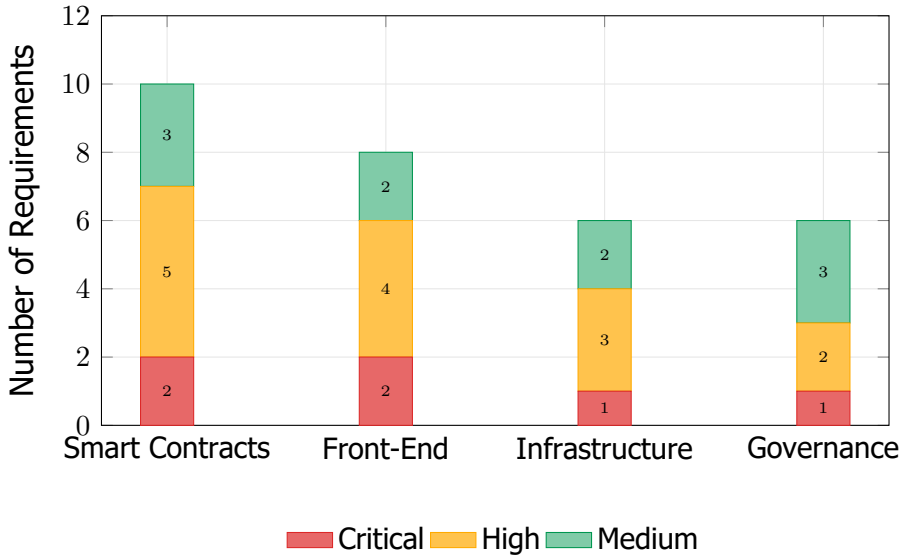

Рис. 4: Розподіл складених вимог безпеки.

Всього було сформульовано 30 вимог до безпеки: 6 критичних, 14 високих і 10 середніх за рівнем серйозності. Найбільша кількість вимог припадає на рівень смарт-контрактів, що відображає незворотний характер вразливостей в ланцюжку. За ним слідує рівень інтерфейсу, що підкреслює важливість забезпечення безпеки механізму доставки, орієнтованого на користувача.

7 Зіставлення: OWASP Top 10 ↔ Uniswap вимог

У таблиці 6 наведено зв'язок категорій OWASP Top 10:2021 із конкретними вимогами безпеки Uniswap.

OWASP	Category Name	Uniswap Requirements
A01	Broken Access Control	SR-01, SR-03, FE-07, GV-01, GV-06
A02	Cryptographic Failures	IF-02
A03	Injection	SR-07, FE-01, FE-04
A04	Insecure Design	SR-04, SR-06, IF-03, GV-01, GV-02
A05	Security Misconfiguration	FE-01, FE-04, FE-06, IF-02, GV-05
A06	Vulnerable Components	SR-10, FE-05, IF-06, GV-02
A07	Auth Failures	FE-03, FE-07
A08	Integrity Failures	SR-06, FE-02, FE-05, IF-03, IF-04
A09	Logging & Monitoring	SR-05, IF-05, GV-03, GV-04
A10	SSRF	IF-01, FE-08, GV-05

Таблиця 6: 10 основних категорій OWASP щодо вимог безпеки Uniswap.

8 Висновки

Протягом лабораторної я досліджував можливість застосування стандартів безпеки веб-додатків OWASP до децентралізованої екосистеми додатків на прикладі протоколу Uniswap. Аналіз показав, що правила OWASP Top 10:2021 залишаються дуже актуальними для безпеки dApp, але вимагає істотного переосмислення в контексті блокчейну. Традиційні категорії, такі як порушений контроль доступу (A01), ін'єкція (A03) та порушення налаштувань безпеки (A05), проявляються по-різному в децентралізованих архітектурах – контроль доступу кодується в незмінній логіці смарт-контракту, а не в серверному проміжному програмному забезпеченні, ін'єктивні вклинювання можуть включають шкідливі дані зворотного виклику поряд із традиційним XSS, а ризики порушення налаштувань зосереджуються на інфраструктурі DNS/CDN, а не на налаштуваннях сервера.

З'ясувалося, що вимоги до смаркт-контрактів (OWASP Smart Contract Top 10) забезпечують необхідне додаткове покриття ризиків в ланцюжку, включаючи повторний вхід, MEV/front-running та маніпуляції оракулом, які не мають прямого еквівалента в традиційній веб-безпеці.

Вимоги до безпеки для Uniswap (30 вимог у чотирьох рівнях) демонструють, що безпека dApp вимагає комплексного підходу, охоплює смарт-контракти, фронтенд, позаланцюгову інфраструктуру та процеси управління. Жоден рівень не може бути захищений і існувати ізольовано – як продемонструвала атака на DNS Uniswap у 2022 році, ідеально перевірений смарт-контракт не забезпечує захисту, якщо механізм фронтенд-доставки скомпрометований.

Основні висновки включають: (1) незмінність є як найбільшою сильною стороною безпеки (запобігання несанкціонованим змінам), так і найбільшим викликом (запобігання правоміному виправленню помилок) систем смарт-контрактів; (2) публічний характер транзакцій блокчейну створює вектори атак, пов'язані з MEV, які не мають аналогів у традиційній веб-безпеці; (3) аутентифікація на основі гаманця усуває класичні атаки на основі облікових даних, але створює нові ризики, пов'язані з сліпим підписом і необмеженим затвердженням токенів.

References

[1] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. «A Survey of Attacks on Ethereum Smart Contracts (SoK)». In: *Principles of Security and Trust* (2017), pp. 164–186. DOI: 10.1007/978-3-662-54455-6_8.

[2] Immunefi. *Crypto Losses 2024 Annual Report*. 2024. (URL).

[3] Philip Daian et al. «Flash Boys 2.0: Frontrunning in Decentralized Exchanges, Miner Extractable Value, and Consensus Instability». In: *IEEE Symposium on Security and Privacy* (May 2020), pp. 910–927. DOI: 10.1109/SP40000.2020.00040.

[4] OWASP Foundation. *About the OWASP Foundation*. 2024. (URL).

[5] OWASP Foundation. *OWASP Top 10:2021*. 2021. (URL).

[6] Hayden Adams et al. *Uniswap v3 Core*. 2021. (URL) (visited on 05/11/2025).

[7] MITRE Corporation. *CWE — Common Weakness Enumeration*. (URL).

[8] OWASP Foundation. *OWASP Application Security Verification Standard (ASVS) 4.0*. 2021. (URL).

[9] OpenZeppelin. *OpenZeppelin Contracts — Secure Smart Contract Library*. (URL).

[10] Ethereum Foundation. *Solidity Documentation v0.8.x*. Tech. rep. 2024. (URL).

[11] Vitalik Buterin et al. *EIP-4337: Account Abstraction Using Alt Mempool*. 2023. (URL).

[12] Blockworks. «Uniswap Front-End Hit by DNS Attack». In: (2022). (URL).

[13] NIST. *Security and Privacy Controls for Information Systems and Organizations (SP 800-53 Rev. 5)*. 2020. (URL).

[14] Hayden Adams, Noah Zinsmeister, and Dan Robinson. «Uniswap v2 Core». In: (2020). (URL).

[15] Uniswap Labs. *Uniswap Protocol Documentation*. Tech. rep. 2025. (URL).

[16] OWASP Foundation. *OWASP Smart Contract Top 10 (2025)*. 2025. (URL).

[17] SmartContractSecurity. *SWC Registry — Smart Contract Weakness Classification*. 2020. (URL).

[18] Ethereum Foundation. «Ethereum Smart Contract Security Best Practices». In: (2025). (URL).

[19] Uniswap Foundation. *Uniswap Governance*. 2024. (URL).

[20] Fabian Vogelsteller and Vitalik Buterin. *EIP-20: Token Standard*. Tech. rep. 2015. (URL).

[21] Uniswap Labs. *Uniswap Bug Bounty Program*. 2024. (URL).

[22] Trail of Bits. *Slither — Static Analysis Framework for Solidity*. 2025. (URL).

[23] ConsensysDiligence. *Mythril — Security Analysis Tool for EVM Bytecode*. 2024. (URL).

[24] Trail of Bits. *Uniswap v3 Core Security Assessment*. Tech. rep. 2021. (URL).

[25] Chainlink Labs. *Chainlink Price Feeds Documentation*. 2024. (URL).

[26] U.S. Department of the Treasury. «Treasury Sanctions Tornado Cash». In: (2022). (URL).

[27] CertiK. *The State of DeFi Security 2024*. 2024. (URL).