Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Навчально-науковий фізико технічний інститут
Кафедра інформаційної безпеки

Звіт

З практичного завдання №3
із дисципліни «Технологія блокчейн та розподілені системи»
Тема: «Дослідження безпечної реалізації та експлуатації децентралізованих додатків»

Виконав:
Студент групи ФБ-41мн
Шерстюк А. В.
Варіант 13

Київ-2025

Axie Infinity

Архітектура децентралізованих додатків суттєво відрізняється від класичних веб-додатків, що зумовлює необхідність переосмислення принципів забезпечення їх безпеки. Смарт-контракти, які виступають ядром логіки dApp, після розгортання стають незмінними, що значно ускладнює процес оновлення або виправлення виявлених вразливостей.

Відсутність централізованого бекенду означає, що вся логіка повинна бути закладена безпосередньо в контракт або підтримуватися off-chain компонентами з чітким дотриманням моделі довіри. У зв'язку з цим критично важливо забезпечити правильну ініціалізацію контрактів, контроль доступу на рівні функцій, а також захист від повторних транзакцій, маніпуляцій із даними та зовнішніх викликів.

Взаємодія клієнта з dApp відбувається через децентралізовану мережу, де дані не контролюються єдиною організацією, а передаються через peer-to-peer мережу й зберігаються в блоках ланцюга. Це накладає специфічні вимоги до побудови механізмів автентифікації, протоколів передачі даних, логування та аудиту подій.

OWASP Top 10- це список з десяти найкритичніших загроз безпеці вебзастосунків. Його створено для підвищення обізнаності розробників і архітекторів щодо типових вразливостей. Розглянемо кожен пункт на прикладі гри Axie Infinity.

OWASP Top 10

A01 Broken Access Control

Недостатньо суворий або повністю відсутній контроль доступу до функціональності або даних, що дозволяє неавторизованим користувачам отримати привілеї або здійснювати заборонені дії.

Контроль доступу забезпечує дотримання політики таким чином, щоб користувачі не могли діяти поза межами своїх призначених дозволів. Збої зазвичай призводять до несанкціонованого розголошення інформації, зміни або знищення всіх даних або виконання бізнес-функцій поза межами можливостей користувача.

У Axie Infinity контроль доступу реалізується через розмежування прав у смарт-контрактах, зокрема щодо керування ігровими активами, SLP-токенами та доступом до marketplace. Рекомендується ретельно перевіряти авторизацію користувачів при виклику функцій, які змінюють стан NFT або токенів, а також впровадити сувору рольову модель (наприклад, гільдії, адміни, гравці) із логічною валідацією їхніх прав. Застосування модифікаторів onlyOwner, require(msg.sender == ...), делегування прав із чіткою політикою RBAC. Важливо також перевіряти стан контракту на момент виклику (наприклад, шляхом впровадження логічних прапорців).

A02 Cryptographic Failures

Неправильне або небезпечне впровадження криптографічних алгоритмів, яке може призвести до витоку конфіденційних даних, порушення їх цілісності або скомпрометованості процесу аутентифікації.

Перше, що потрібно зробити, це визначити потреби в захисті даних під час передачі та зберігання. Наприклад, паролі, номери кредитних карток, медичні записи, особиста інформація та комерційна таємниця потребують додаткового захисту, головним чином, якщо ці дані підпадають під дію законів про конфіденційність, наприклад, Загального регламенту ЄС про захист даних (GDPR), або нормативних актів, наприклад, щодо захисту фінансових даних, таких як стандарт безпеки даних PCI (PCI DSS).

У контексті Axie Infinity важливо забезпечити криптографічну цілісність транзакцій, які пов’язані з обміном токенів AXS і SLP, а також взаємодією з NFT. Використання Keccak256 гарантує хешування критичних структур даних у контрактах. Варто уникати залежності від block.timestamp або blockhash, особливо при реалізації ігрової механіки з елементами випадковості (наприклад, створення нових Axies), оскільки це може бути використано зловмисником для маніпуляцій. Рекомендується впровадити зовнішні верифіковані oracle-сервіси для генерації випадкових значень у випадках, коли це критично для геймплею. Забезпечення криптостійкої ентропії, використання Keccak256 для хешування, уникнення block.timestamp, block.number та інших псевдовипадкових джерел у логіці визначення переможців або вибору.

A03 Injection

Ін’єкційні вразливості дозволяють зловмисникам інжектувати шкідливий код (наприклад, SQL, NoSQL, OS-команди), що виконується інтерпретатором на стороні сервера.

У Axie Infinity ін’єкційні атаки можуть виникати переважно у front-end частині (наприклад, у web-інтерфейсі marketplace) або у випадках інтеграції з централізованими API. У смарт-контрактах важливо уникати необмеженого виклику функцій через delegatecall або call, які можуть бути використані зловмисниками для виконання небезпечної логіки. Рекомендується використовувати сувору типізацію аргументів, обов’язково перевіряти їх на відповідність очікуваним діапазонам значень, а також унеможливити взаємодію з зовнішніми контрактами без ретельної перевірки адреси та коду. На фронтенді слід реалізувати фільтрацію та екранування вхідних даних для захисту від XSS. Уникнення динамічних викликів на кшталт call.value() без перевірки, жорстке типування параметрів, перевірка вхідних даних на коректність у зовнішніх інтерфейсах (інтерфейси взаємодії користувача).

A04 Insecure Design

Відсутність фундаментальних принципів безпеки на етапі архітектурного проєктування системи, що унеможливлює належну протидію сучасним векторним загрозам.

Небезпечний дизайн- це широка категорія, що представляє різні слабкі місця, що виражаються як «відсутній або неефективний дизайн контролю». Небезпечний дизайн не є джерелом для всіх інших 10 категорій ризиків. Існує різниця між небезпечним дизайном та небезпечною реалізацією. Недоліки дизайну та дефекти реалізації відрізняються не просто так, вони мають різні першопричини та способи усунення. Безпечний дизайн все ще може мати дефекти реалізації, що призводять до вразливостей, які можна використовувати. Небезпечний дизайн не може бути виправлений ідеальною реалізацією, оскільки за визначенням необхідні засоби контролю безпеки ніколи не створювалися для захисту від конкретних атак. Одним із факторів, що сприяють небезпечному дизайну, є відсутність профілювання бізнес-ризиків, властивого програмному забезпеченню або системі, що розробляється, і, отже, нездатність визначити, який рівень дизайну безпеки потрібен.

У Axie Infinity проєктування логіки має критичне значення, оскільки неправильне визначення правил бою, розмноження або нагород може бути використане для експлуатації механіки гри. Необхідно впроваджувати формальну верифікацію бойових правил, обмежень на клонування Axies, а також економічних моделей (механізми розподілу винагород, емісія токенів). Використання інструментів типу MythX, Slither або Certora дозволяє виявити логічні помилки та провести символьне тестування. Крім того, доцільним є впровадження шаблонів безпечної архітектури, таких як separation of concerns, щоб розділити геймплейну логіку, управління токенами та адміністративні

функції.Формалізація логіки смарт-контрактів з використанням верифікаційних інструментів (MythX, Manticore, Certora), а також використання відомих шаблонів безпеки (Design Patterns).

A05 Security Misconfiguration

Використання небезпечних або стандартних конфігурацій, увімкнені службові інтерфейси, неправильні дозволи- все це підвищує ризики експлуатації через конфігураційні помилки.

У Axie Infinity критично важливо правильно налаштувати контракти, щоб унеможливити зловживання адміністративними функціями або функціями знищення (selfdestruct). Наприклад, контракти, що керують токенами AXS чи SLP, повинні бути налаштовані з обмеженням на доступ до функцій оновлення або викликів через delegatecall. Рекомендується використовувати проксі-патерни з контролем ініціалізації (наприклад, initializer у OpenZeppelin), уникати надлишкових дозволів для зовнішніх викликів та забезпечити обмеження на обсяг gas, який може бути спожитий у межах однієї транзакції. Також доцільно обмежити доступ до функцій конфігурації контракту лише авторизованим адміністраторам з багатофакторною перевіркою.Заборона виклику selfdestruct, обмеження delegatecall, контроль gas consumption на рівні функцій, перевірка коректної ініціалізації проксі-контрактів (UUPS, Transparent proxy).

A06 Vulnerable and Outdated Components

Інтеграція компонентів із відомими вразливостями або використання застарілих версій бібліотек та фреймворків, що вже не підтримуються або мають публічно відомі експлойти.

У випадку Axie Infinity необхідно постійно відстежувати стан безпеки використовуваних зовнішніх бібліотек, зокрема OpenZeppelin, бібліотек обробки транзакцій, а також фронтенд-компонентів. Зловмисники можуть експлуатувати відомі вразливості, якщо контракти чи бібліотеки не оновлюються регулярно. Рекомендується застосовувати автоматизовані засоби безпеки (Snyk, npm audit, GitHub Dependabot), проводити періодичні аудити коду і уникати залежностей, що не мають належної підтримки або аудиту спільнотою.Постійне оновлення залежностей, відстеження вразливостей через агрегатори (Snyk, CVE), уникнення маловідомих бібліотек, що не мають аудиту.

A07 Identification and Authentication Failures

Недостатній захист механізмів автентифікації (паролі, токени, сеанси), включаючи відсутність захисту від атак перебору, фішингу або викрадення сесій. Підтвердження ідентифікації користувача, автентифікація та керування сеансом є критично важливими для захисту від атак, пов'язаних з автентифікацією.

У Axie Infinity гравці автентифікуються шляхом підпису повідомлень через криптогаманці (наприклад, MetaMask). Цей підхід виключає збереження паролів, але вимагає додаткових перевірок для уникнення фішингових атак. Рекомендується використовувати стандарт EIP-712 для структурованих повідомлень, щоб унеможливити неочікувану інтерпретацію вмісту повідомлення. Також доцільно реалізовувати перевірку автентичності клієнтів через challenge-response механізм.Підпис повідомлень користувачем (eth_signTypedData_v4), верифікація власності через EIP-712, без використання централізованих токенів або cookie.

A08 Software and Data Integrity Failures

Нездатність гарантувати незмінність і достовірність даних або програмного коду, зокрема при використанні неавтентифікованих каналів доставки або механізмів оновлення.

Порушення цілісності програмного забезпечення та даних пов'язані з кодом та інфраструктурою, які не захищають від порушень цілісності. Прикладом цього є ситуація, коли програма покладається на плагіни, бібліотеки або модулі з ненадійних джерел, репозиторіїв та мереж доставки контенту (CDN). Незахищений конвеєр CI/CD може створити потенціал для несанкціонованого доступу, шкідливого коду або компрометації системи. Нарешті, багато програм тепер включають функцію автоматичного оновлення, коли оновлення завантажуються без достатньої перевірки цілісності та застосовуються до раніше надійної програми. Зловмисники потенційно можуть завантажувати власні оновлення для розповсюдження та запуску на всіх інсталяціях. Іншим прикладом є ситуація, коли об'єкти або дані кодуються або серіалізуються в структуру, яку зловмисник може бачити та змінювати, що є вразливим до незахищеної десеріалізації.

Axie Infinity покладається на правильну обробку транзакцій у блокчейні, тому перевірка nonce, хешування структур та верифікація підписів мають вирішальне значення. Особливо важливо забезпечити захист від повторного використання підписаних повідомлень у разі зміни логіки контракту або інтерфейсів. Рекомендується вводити TTL (time-to-live) для підписів, додаткові поля автентифікації (наприклад, chainId) та контроль за відповідністю формату й змісту транзакцій.Захист транзакцій через контроль nonce, впровадження механізмів перевірки цифрових підписів і хешованих структур даних для уникнення маніпуляцій.

A09 Security Logging and Monitoring Failures

Без реєстрації та моніторингу порушення неможливо виявити. Відсутність ефективного моніторингу інцидентів безпеки та системного логування, що значно ускладнює виявлення та реагування на атаки.

У Axie Infinity важливо логувати усі ключові події (наприклад, minting, breeding, battling, торгові операції) за допомогою подій Solidity (emit). Це дозволяє off-chain

аналітиці (наприклад, The Graph або Dune Analytics) ефективно обробляти дані. Крім того, рекомендовано дублювати критичні події в додаткові сховища або кеші для підвищення доступності і цілісності, а також впровадити систему виявлення аномалій у транзакційній активності гравців.Використання подій (emit) для фіксації транзакцій, подальша агрегація та аналіз log-записів через off-chain аналітичні інструменти (наприклад, The Graph, Dune Analytics).

A10 Server-Side Request Forgery (SSRF)

Експлуатація серверної інфраструктури для здійснення запитів до внутрішніх або зовнішніх ресурсів, часто з метою обхід внутрішніх обмежень доступу.

Недоліки SSRF виникають щоразу, коли веб-застосунок отримує віддалений ресурс без перевірки наданої користувачем URL-адреси. Це дозволяє зловмиснику змусити застосунок надіслати спеціально створений запит до неочікуваного місця призначення, навіть якщо він захищений брандмауером, VPN або іншим типом списку контролю доступу до мережі (ACL).

Оскільки сучасні веб-застосунки надають кінцевим користувачам зручні функції, отримання URL-адреси стає поширеним сценарієм. Як наслідок, частота виникнення SSRF зростає. Крім того, серйозність SSRF зростає через хмарні сервіси та складність архітектур.

У Axie Infinity взаємодія зі сторонніми контрактами (наприклад, при інтеграції з DeFi або NFT-маркетплейсами) несе ризики повторного входу (reentrancy) та маніпуляцій. Використання патерну "checks-effects-interactions" повинно бути обов’язковим для кожної функції, що взаємодіє з адресами за межами контракту. Також слід впроваджувати guard-механізми, такі як ReentrancyGuard, обмеження на кількість зовнішніх викликів у рамках транзакції, і перевірку повернених значень на наявність аномалій.Упровадження шаблону "checks-effects-interactions" для запобігання reentrancy, а також використання бібліотек (наприклад, OpenZeppelin ReentrancyGuard) для структурного захисту.

Висновки

Попри те, що архітектура децентралізованих додатків значною мірою відрізняється від класичних централізованих моделей, фундаментальні принципи безпеки, окреслені OWASP, залишаються надзвичайно цінними для аналізу загроз і побудови безпечних рішень у Web3-середовищі.

Ключовими елементами, що потребують адаптації, є іммутативність смарт-контрактів, довіра до on-chain логіки, відсутність централізованої автентифікації та складність контролю зовнішніх викликів. Успішна імплементація безпечного dApp потребує міждисциплінарного підходу- поєднання знань у галузях криптографії, безпечного проєктування, формальної верифікації й розподілених обчислень.

Формалізовані вимоги, що сформульовані в цій роботі, можуть слугувати базою для створення подальших стандартів безпеки у сфері децентралізованих технологій, а також основою для проведення аудиту, оцінки ризиків і розробки захисних механізмів для майбутніх поколінь dApp.