БЛОКЧЕЙН ТА ДЕЦЕНТРАЛІЗОВАНІ СИСТЕМИ

Лабораторна робота №3

“Дослідження безпечної реалізації та експлуатації децентралізованих додатків”

Недождій Максим, Буржимський Ростислав

ФІ-42мн

1 Мета роботи

Отримання навичок роботи із децентралізованими додатками та оцінка безпеки інформації при їх функціонуванні.

Для першого типу лабораторних робіт

Дослідження вимог OWASP (безпека web-додатків) та складання аналогічних вимог для обраної системи децентралізованих додатків.

2 Вимоги OWASP до безпеки веб-додатків

Open Web Application Security Project (OWASP) публікує список найпоширеніших та критичних вразливостей безпеки веб-додатків. Нашому потоку доведеться вказувати дані за 2021 рік, наступний рік вже зможе використати дані за 2025, які наразі завершили збирати, але ще не обробили до кінця. Нижче наведено основні десять загроз OWASP (OWASP Top 10) з коротким описом:

A01. Broken Access Control — Порушення контролю доступу:
Користувачі можуть отримати доступ до функцій або даних, які їм не призначені.

A02. Cryptographic Failures — Криптографічні помилки:
Недостатній захист конфіденційних даних через неправильну або слабку реалізацію криптографічних механізмів захисту.

A03. Injection — Ін’єкції:
Вразливості, що дозволяють виконувати шкідливий код, наприклад SQL або NoSQL ін’єкції.

A04. Insecure Design — Небезпечна архітектура:
Недостатній аналіз загроз при розробці, або використання шаблонів, які спрощують атаки.

A05. Security Misconfiguration — Неправильна конфігурація безпеки:
Використання стандартних паролів, відкрите адміністрування тощо.

A06. Vulnerable and Outdated Components — Уразливі та застарілі компоненти:
Застосування бібліотек із відомими вразливостями.

A07. Identification and Authentication Failures — Помилки ідентифікації та автентифікації:
Слабка реалізація механізмів входу або керування сесіями.

A08. Software and Data Integrity Failures — Порушення цілісності програмного забезпечення та даних:
Відсутність перевірки цифрових підписів на оновленнях.

A09. Security Logging and Monitoring Failures — Відсутність журналювання та моніторингу безпеки:
Ускладнює виявлення атак та розслідування інцидентів.

A10. Server-Side Request Forgery (SSRF) — Підміна серверних запитів:
Сервер може робити запити до зовнішніх або внутрішніх ресурсів без належної перевірки.

3 Адаптовані вимоги OWASP для Ethereum dApps

Стандарт OWASP для традиційних веб-додатків не повністю охоплює специфіку децентралізованих додатків (dApps), особливо таких, що працюють на блокчейні Ethereum. Нижче наведено адаптацію 10 ключових принципів OWASP Top 10 з урахуванням специфіки платформи Ethereum.

E01. Неправильне керування правами доступу (Access Control)
У Ethereum доступ до функцій визначається через змінні контролю (‘owner‘, ‘admin‘, ‘roles‘). Вразливості, такі як відсутність модифікатора `onlyOwner`, можуть дозволити зловмисникам керувати контрактом або отримати доступ до критичних функцій.

E02. Криптографічні помилки
Ethereum сам забезпечує цифрові підписи та хешування (через ‘keccak256‘, ‘ecrecover‘), однак неправильне їх використання (наприклад, передбачувані nonce або повторне використання підписів) може дозволити атаки, як-от replay або signature malleability.

E03. Ін’єкції в смарт-контрактах
Хоча в Ethereum не існує SQL, інші види ін’єкцій (наприклад, маніпуляції з вхідними даними, зовнішні виклики до нестабільних контрактів) можуть спричинити небажану поведінку. Слід уникати необмежених циклів і використовувати перевірки на типи і валідацію параметрів.

E04. Небезпечна архітектура смарт-контрактів
Використання централізованих елементів, таких як єдиний адміністратор без DAO, або монолітних контрактів без можливості оновлення — все це може створити проблеми для безпеки. Варто розділяти логіку за контрактами (наприклад, Proxy-патерн).

E05. Неправильна конфігурація безпеки при розгортанні
Смарт-контракти не мають можливості змін після деплою — тому будь-які помилки в налаштуваннях (наприклад, публічні функції без обмежень, не ініціалізований owner) створюють постійні загрози.

E06. Використання застарілих або вразливих бібліотек
Solidity-контракти можуть використовувати сторонні бібліотеки (наприклад, OpenZeppelin). Якщо їх не оновлювати, можуть виникати вразливості. Важливо використовувати перевірені версії та аудитований код.

E07. Недоліки ідентифікації та автентифікації
В Ethereum автентифікація здійснюється через `msg.sender`. Неправильне використання (наприклад, через делеговані виклики або зовнішні проксі-контракти) може призвести до викривлення ідентичності. Потрібно уникати викликів через ‘delegatecall‘, якщо це не потрібно.

E08. Порушення цілісності коду та даних

DApp повинен гарантувати, що зовнішні залежності, такі як бібліотеки або API, залишаються незмінними. Для цього використовують механізми фіксації хешів, контроль коду через IPFS або ENS, а також верифікацію контрактів у публічних репозиторіях (наприклад, Etherscan).

E09. Відсутність журналювання та моніторингу

Ethereum не має централізованого журналювання, але події (‘event‘) у контрактах дозволяють створити систему моніторингу. Важливо логувати всі важливі дії (торги, зміни власності, адміністративні дії) та інтегрувати моніторингові сервіси (наприклад, Tenderly, Forta).

E10. Виклики до зовнішніх контрактів (Reentrancy)

Класична вразливість, яка дозволяє зловмиснику багаторазово викликати контракт до завершення основної логіки. Для запобігання слід дотримуватись патерну `Checks-Effects-Interactions` та використовувати ‘reentrancyGuard‘.

4 Безпекові вимоги до CryptoKitties на Ethereum

CryptoKitties — це популярний децентралізований додаток (dApp), який функціонує на блокчейні Ethereum (і не тільки) і дозволяє користувачам купувати, розводити та продавати віртуальних котів за криптовалюту. Розробники концепту позиціонують свій dApp, як оригінальний NFT за рахунок унікальності кожного кота і прив’язки певних ідентифікаторів до них, пов’язаних з власником у даний момент часу і творцем. При адаптації вимог OWASP до такого dApp необхідно враховувати особливості блокчейн-архітектури, оскільки у OWASP в основному розглядаються саме централізовані додатки, відповідно за відсутністю цих загроз безпеки відповідає центральний орган управління.

C01. Контроль доступу до функцій смарт-контракту:

Всі дії (наприклад, розведення котів, передача токенів) повинні бути дозволені лише авторизованим власникам.

C02. Криптографічний захист ідентифікації користувачів:

Ідентифікація базується на адресах Ethereum — необхідно перевіряти цифрові підписи транзакцій і уникати можливості підробки.

C03. Захист від ін’єкцій у смарт-контрактах:

Оскільки контракти будуються на основі блокчейну, необхідно уникати використання зовнішніх викликів і динамічного виконання коду у Solidity.

C04. Надійна архітектура контрактів:

Розмежування логіки за контрактами (наприклад, власність, функціональність, зберігання даних) і верифікація за допомогою сторонніх аудитів.

C05. Безпечна ініціалізація та конфігурація:

Контракти не повинні мати функцій, які можна викликати повторно або змінювати після розгортання без належних перевірок.

C06. Оновлення контрактів:

Якщо передбачена можливість оновлення, вона має бути захищена механізмами довіри (наприклад, через DAO або багатопідписні контракти).

C07. Ідентифікація користувачів і перевірка прав:

Автентифікація транзакцій повинна покладатися на механізми блокчейну (наприклад, ‘msg.sender‘ у Solidity).

C08. Цілісність даних:
Усі дані повинні зберігатись у блокчейні або у перевірених децентралізованих сховищах (наприклад, IPFS для зображень).

C09. Моніторинг та аудит транзакцій:
Всі транзакції мають бути публічними — але також варто інтегрувати сервіси аналітики для виявлення підозрілої активності.

C10. Захист від зовнішніх викликів (Reentrancy):
Контракт має бути стійким до атак повторного виклику (наприклад, використовуючи шаблон Checks-Effects-Interactions).

5 Висновки

Ethereum dApps мають унікальні архітектурні риси, які вимагають особливої уваги до безпеки на всіх етапах життєвого циклу. Адаптація OWASP дозволяє створити перевірений підхід до оцінки ризиків і проектування захищених смарт-контрактів.

У випадку CryptoKitties, забезпечення контролю доступу, перевірки автентичності, цілісності даних і стійкості до відомих вразливостей є критично важливим для безпечної роботи системи.