МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
імені ІГОРЯ СІКОРСЬКОГО
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ ЗАХИСТУ ІНФОРМАЦІЇ

ЗВІТ
з лабораторної роботи № 3

Виконав:
студент ФБ-41мн
Дрик Владислав
Олександрович
«8» Квітня 2025

КИЇВ 2024

Тема: Дослідження безпечної реалізації та експлуатації децентралізованих додатків.

Мета роботи: отримання навичок роботи із децентралізованими додатками та оцінка безпеки інформації при їх функціонуванні

Вимоги OWASP Top 10 до безпеки Web-додатків

1. Broken Access Control — неправильна перевірка доступу
2. Cryptographic Failures — слабке або відсутнє шифрування
3. Injection — SQL, OS або інші ін’єкції
4. Insecure Design — поганий проєкт з точки зору безпеки
5. Security Misconfiguration — неправильні налаштування безпеки
6. Vulnerable and Outdated Components — небезпечні сторонні бібліотеки
7. Identification and Authentication Failures — проблеми з автентифікацією
8. Software and Data Integrity Failures — незахищене оновлення або підробка коду
9. Security Logging and Monitoring Failures — відсутність логування та моніторингу
10. Server-Side Request Forgery (SSRF) — сервер виконує шкідливі зап

Аналогічні вимоги для децентралізованого додатку (на прикладі Uniswap)

1) **Контроль доступу у смарт-контрактах**

У блокчейн-системах немає традиційних ролей "користувач/адмін", але контроль доступу до функцій смарт-контракту є критично важливим. Наприклад, функції типу withdraw, mint, або upgrade не повинні бути доступні кожному.

Хакер може викликати withdraw() функцію і вивести кошти, якщо немає require(msg.sender == owner).

Рішення:

- Впровадити модифікатори onlyOwner, onlyAdmin
- Використовувати бібліотеки контролю доступу, як OpenZeppelin AccessControl

2) **Криптографічна цілісність транзакцій**

Хоча транзакції в Ethereum вже підписані, інші компоненти (наприклад, позасмугові повідомлення, офчейн-підписи) повинні перевірятися на достовірність.

Ризик - Підробка повідомлень або транзакцій через replay-атаки чи інші механізми.

Рішення:

- Підпис усіх даних користувачем (ecrecover)
- Захист від повторного використання (nonce, expiry)

3) **Ін’єкції у смарт-контрактах**

Смарт-контракти можуть містити логічні уразливості, як-от повторний вхід (reentrancy), при якому зовнішній контракт викликає ваш до завершення транзакції.

Приклад - DAO Hack (2016) — повторний виклик дозволив виводити кошти нескінченно.

Рішення:

- Використання шаблону "Checks-Effects-Interactions"
- Використання ReentrancyGuard з OpenZeppelin
- Бібліотеки для безпечної арифметики (SafeMath)

4) **Безпечне проєктування смарт-контракту**

Контракт повинен бути модульним, розділяти логіку, мати fail-safe механізми. Уникати надто складної логіки або “все в одному”.

Рішення:

- Поділ контрактів: логіка, зберігання, доступ
- Захист від фатальних помилок (fallback, selfdestruct)

5) **Конфігурація блокчейн-інфраструктури**
Зловмисник може скористатися неправильними RPC-налаштуваннями, підключенням до тестової мережі або публічним API.

Рішення:

- Власні ноди або VPN-захист RPC
- Обмеження IP/ключів
- Аудит конфігурації перед деплоєм

6) **Безпека ораклів**

Оракли — зовнішні джерела даних (ціни токенів, події тощо). Якщо вони неавторитетні або зламані — можна маніпулювати логікою смарт-контракту.

Приклад:
Атаки на DeFi платформи, які залежать від даних ціни токенів (flash loan + підробка ціни = вивід активів).

Рішення:

- Chainlink або інші надійні оракли
- Перевірка середнього значення з кількох джерел
- Обмеження на зміни даних

7) **Децентралізована автентифікація**

Користувачі автентифікуються через гаманці (MetaMask, WalletConnect). Не можна покладатися на традиційні логіни/паролі.

Рішення:

- Підписування nonce-повідомлень (Sign-In with Ethereum)
- Використання ENS або DID для авторитетного представлення користувача

8) **Цілісність і оновлення контракту**

Контракти зазвичай незмінні після деплою. Але якщо використовується проксі-архітектура — може бути ризик підміни логіки.

Ризик - Адміністратор змінює логіку на шкідливу або небезпечну.

Рішення:

- Відкрита документація всіх апгрейдів
- Проксі-патерни (UUPS, Transparent Proxy) з контрольованим доступом
- Аудити після кожного оновлення

9) **Моніторинг і аудит (On-Chain Monitoring and Alerts)**

У традиційних системах є лог-файли. У DApp — це журнал транзакцій у блокчейні. Але важливо моніторити та аналізувати в реальному часі.

Рішення:

- Інтеграція Forta, Tenderly або The Graph
- Повідомлення при підозрілих діях (великі виводи, незвичні call-и)
- Візуалізація транзакцій

10) **Безпека міжланцюгових взаємодій**

Мости — найбільш вразлива частина багатьох систем. Через них передаються токени, і дані можна фальсифікувати.

Атака на Ronin Bridge (Axie Infinity) — втрачено понад $600 млн

Рішення:

- Мультисиг перевірка транзакцій
- Механізми затримки та валідації
- Багатоступенева перевірка повідомлень

Для другого типу лабораторних робіт:

Назва проєкту: WhoTF? (“Who the f*** did this?”)

Мета: надати авторам цифрового контенту можливість зафіксувати авторство, передавати права, підтверджувати власність і монетизувати твори за допомогою смарт-контрактів.

Функції -

1) Реєстрація цифрового контенту - Захист авторства через timestamp і публічний хеш)

Користувач завантажує файл (зображення, відео, документ). Файл зберігається на IPFS Створюється NFT / запис у смарт-контракті з IPFS-хешем і метаданими. Автоматичний запис: автор, час, опис

*** IPFS** (InterPlanetary File System) — це децентралізована система зберігання файлів, схожа на блокчейн, але для файлів, а не транзакцій. Вона дозволяє зберігати файли в мережі так, щоб їх неможливо підробити, втратити або контролювати з одного місця.

2) Підтвердження власності - Непідробна історія авторства

Інші користувачі можуть бачити, хто перший зареєстрував файл. Автор може показати свій NFT як доказ володіння

3) Передача прав - Трасування прав власності в блокчейні

Автор може передати NFT іншому користувачу**.** Це еквівалент юридичного переоформлення авторських прав

Спрощенна логіка

```
contract WhoTF is ERC721, Ownable {
    uint public nextTokenId = 1;

    struct Content {
        string ipfsHash;
        string title;
        address author;
        uint timestamp;
    }

    mapping(uint => Content) public contents;

    constructor() ERC721("WhoTF", "WTF") {}

    function registerContent(string memory ipfsHash, string memory title) public {
        uint tokenId = nextTokenId++;
        _mint(msg.sender, tokenId);
        contents[tokenId] = Content(ipfsHash, title, msg.sender, block.timestamp);
    }

    function getContent(uint tokenId) public view returns (Content memory) {
        return contents[tokenId];
    }
}
```

ERC-721 — стандарт для унікальних **токенів** (як цифрові предмети, мистецтво, ідентифікатори)

ВИСНОВОК

У результаті дослідження та розробки було створено базу, фундамент, грунт для децентралізованого додаток для захисту інтелектуальної власності цифрового контенту з використанням сучасних технологій Web3 — Ethereum, IPFS та смарт-контрактів стандарту ERC-721. Такий підхід дозволяє авторам фіксувати авторство, підтверджувати володіння та безпечно передавати права на цифрові об’єкти без участі централізованих посередників. Система забезпечує прозорість, незмінність даних та високий рівень довіри, що є важливими умовами у сучасному цифровому середовищі. Подальші кроки можуть включати впровадження механізмів ліцензування, монетизації та інтеграцію з платформами для перевірки оригінальності контенту.