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

Вимоги OWASP Top 10 для Web-додатків

OWASP (Open Web Application Security Project) — це міжнародна некомерційна організація, яка визначає найбільш критичні загрози безпеки веб-застосунків. Актуальний список OWASP Top 10 включає такі категорії:

1. Broken Access Control —

Неправильне або відсутнє обмеження прав доступу. Користувач може отримати доступ до функціональності, яка йому не призначена (наприклад, змінити чужий пароль або видалити чужі дані).

У dApp:

Смарт-контракт може дозволити звичайному користувачу виконувати дії, призначені лише для адміністратора — наприклад, змінити комісію чи заморозити активи.

Захист: Рольова модель (RBAC), модифікатори доступу (onlyOwner, require(msg.sender == admin)).

2. Cryptographic Failures

Неправильне або відсутнє обмеження прав доступу. Користувач може отримати доступ до функціональності, яка йому не призначена (наприклад, змінити чужий пароль або видалити чужі дані).

У dApp: збереження даних у відкритому вигляді на блокчейні або використання слабких хешів (MD5, SHA1).

Захист: Використання сучасних криптоалгоритмів (SHA3, AES), шифрування поза блокчейном, підписані повідомлення.

.

3. Injection — ін’єкції (SQL, Command, та інші)

Ін’єкція шкідливого коду в запит (SQL, LDAP, NoSQL, shell). Наприклад, через поле форми користувач додає шкідливу SQL-команду.

У dApp: використання небезпечних delegatecall або передача вхідних даних без валідації, що дозволяє змінювати логіку контракту.

Захист:Валідація/санітизація введення, уникнення прямого використання зовнішніх call, обмеження обробки даних у контракті.

4. Insecure Design

Недостатній захист у самій архітектурі. Наприклад, відсутність перевірок на рівні дизайну системи, які дозволяють легко обійти безпеку.

У dApp:Контракт не перевіряє повторний виклик функції, що робить його вразливим до re-entrancy.

Захист:Патерни безпечної архітектури, threat modeling, використання design-review практик ще до кодування.

5. Security Misconfiguration — помилки конфігурації безпеки

Вразливості, пов’язані з неправильним налаштуванням серверів, middleware, контейнерів тощо.

У dApp: контракт не має ліміту на операції, дозволяє змінювати стан без автентифікації або надає доступ до функцій debug.

Захист: ретельне тестування конфігурацій, виключення небезпечного коду перед продакшн-деплоєм.

6. Vulnerable and Outdated Components — уразливі або застарілі компоненти

Використання бібліотек, фреймворків чи плагінів з відомими уразливостями.

У dApp: використання застарілих бібліотек OpenZeppelin, залежність від неконтрольованих оракулів або контрактів сторонніх систем.

Захист: постійний аудит залежностей, оновлення бібліотек, перевірка сумісності зі стандартами безпеки (наприклад, EIP).

7. Identification and Authentication Failures — помилки ідентифікації та автентифікації

Недостатній захист автентифікації. Наприклад, слабкі паролі, відсутність обмежень на спроби входу.

У dApp: контракт не перевіряє, чи дійсно той, хто викликає функцію, є справжнім власником гаманця, або працює з неавтентичними підписами.

Захист:

Багатофакторна автентифікація, використання ecrecover, перевірка підписів на бекенді чи контракті.

8. Software and Data Integrity Failures — порушення цілісності ПЗ і даних

Відсутність перевірки цілісності оновлень, залежностей або конфігурацій. Атакуючий може підмінити код або вставити бэкдор.

У dApp: контракт має можливість оновлення логіки (upgradeability) без перевірки власника чи хешу коду.

Захист: контроль за оновленням через DAO, підписані хеші нових реалізацій, фіксація імплементацій у proxy.

9. Security Logging and Monitoring Failures — відсутність журналювання та моніторингу

Відсутність журналювання, логів доступу або сповіщень про атаки.

У dApp: немає подій (event) про критичні дії — переміщення коштів, зміна ролей, блокування користувачів тощо.

Захист: запис event при всіх важливих діях, підключення до аналітичних сервісів (Etherscan, Tenderly, Forta).

10. Server-Side Request Forgery (SSRF) — підробка серверних запитів

Застосунок змушується зробити HTTP-запит на внутрішні ресурси (наприклад, AWS metadata), що може дати доступ до конфіденційної інформації.

У dApp: На перший погляд здається, що смарт-контракти не можуть робити запити на сторонні ресурси (немає HTTP у Solidity). І це правда. Але проблема SSRF перемістилась у рівень:

Oracle (Chainlink, Band, Witnet тощо)

Смарт-контракти не мають доступу до зовнішніх даних напряму, тому використовують оракули, які дістають ці дані з Інтернету і передають у блокчейн. Якщо зловмисник маніпулює адресою, до якої oracle має зробити запит — можна спричинити SSRF на інфраструктурі oracle.

Захист: огортання Oracle-платформ під багаторівневу перевірку, whitelist IP/URL, використання авторизованих data-feeds.

Висновок

Проведений аналіз демонструє, що принципи OWASP Top 10 залишаються актуальними і в контексті децентралізованих додатків. Особливості Uniswap як DeFi-протоколу на Ethereum вимагають адаптації стандартів безпеки до специфіки smart-контрактів, відмовостійкої архітектури та відкритої природи блокчейн-взаємодій. Для забезпечення надійної роботи dApp необхідно впроваджувати не лише технічні механізми захисту, а й інституційні — багаторівневе управління, аудити та спільнотні перевірки коду.