Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Фізико-технічний інститут

«Блокчейн та децентралізовані системи»
Лабораторна робота №1

Тема: „Дослідження безпечної реалізації та експлуатації децентралізованих додатків”.
Мета роботи: «Отримання навичок роботи із децентралізованими додатками та оцінка безпеки інформації при їх функціонуванні».

Виконав:
студент групи ФІ-41мн
Намчук Максим

Для першого типу лабораторних робіт:

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

Список вимог OWASP Top 10 до безпеки Web-додатків:

Broken Access Control — неправильна перевірка доступу

Cryptographic Failures — слабке або відсутнє шифрування

Injection — SQL, OS або інші ін'єкції

Insecure Design — поганий проект з точки зору безпеки

Security Misconfiguration — неправильні налаштування безпеки

Vulnerable and Outdated Components — небезпечні сторонні бібліотеки

Identification and Authentication Failures — проблеми з аутентифікацією

Software and Data Integrity Failures — незахищене оновлення або відробка коду

Security Logging and Monitoring Failures — відсутність логування та моніторингу

Server-Side Request Forgery (SSRF) — сервер виконує шкідливі запити

Аналогічні вимоги для децентралізованого додатку (на прикладі Aave)

1. **Несправна перевірка доступу (Broken Access Control)**
   Необхідно забезпечити контроль прав доступу до функцій смарт-контрактів, аби користувачі не могли маніпулювати умовами позик або змінювати статуси угод без належного дозволу. Наприклад, функції, як withdraw, mint або upgrade не повинні бути доступні кожному користувачу.

   Рішення:

   - Впровадження модифікаторів `onlyOwner`, `onlyAdmin`
   - Використання бібліотек контролю доступу, таких як OpenZeppelin AccessControl

2. **Слабке або відсутнє шифрування (Cryptographic Failures)**
   Шифрування всіх транзакцій та персональних даних користувачів є критично важливим для запобігання атакам на блокчейн, особливо у випадках, пов’язаних із позиками та поверненням коштів.

   Рішення:

   - Використання стандартів шифрування даних у транзакціях та персональних даних.
   - Забезпечення належної криптографії для всіх фінансових операцій.

3. **Ін'єкції (Injection)**
   Атаки через несанкціоновані запити або введення шкідливих даних можуть призвести до небажаних змін у смарт-контракті, як-от маніпуляції з відсотками або лімітуванням позик.

   Рішення:

- Фільтрація та валідація всіх введених даних перед передачею їх до смарт-контракту.

- Використання захищених механізмів для перевірки вхідних запитів.

4. **Поганий проект з точки зору безпеки (Insecure Design)**
   Смарт-контракти повинні бути спроектовані таким чином, щоб уникати вразливостей, таких як повторне використання коштів або маніпуляції з відсотками позик.

   Рішення:

   - Дотримання принципів безпеки при проектуванні смарт-контрактів.

   - Використання архітектурних підходів для забезпечення безпеки системи.

5. **Невірні налаштування безпеки (Security Misconfiguration)**
   Виявлення та виправлення помилок у конфігураціях смарт-контрактів або інтерфейсів, щоб уникнути уразливих точок доступу до баз даних або інших важливих компонентів платформи.

   Рішення:

   - Проводити регулярні аудити конфігурацій безпеки.

   - Використовувати автоматизовані інструменти для перевірки налаштувань безпеки.

6. **Небезпечні сторонні компоненти (Vulnerable and Outdated Components)**
   Сторонні бібліотеки та компоненти можуть бути джерелом уразливостей, тому їх потрібно постійно оновлювати та перевіряти на наявність безпекових дір.

   Рішення:

   - Перевірка та оновлення всіх сторонніх бібліотек.

   - Впровадження механізмів автоматичного оновлення залежностей.

7. **Проблеми з аутентифікацією (Identification and Authentication Failures)**
   Надійна аутентифікація, включаючи багатофакторну аутентифікацію та

інтеграцію з криптовалютними гаманцями, такими як MetaMask, є необхідною для безпечного доступу до системи позик.

Рішення:

- Використання багатофакторної аутентифікації для користувачів.
- Впровадження перевірки підключення через криптогаманці.

8. **Незахищене оновлення або відробка коду (Software and Data Integrity Failures)**
   Запобігання будь-яким несанкціонованим змінам коду чи цілей, що обробляються смарт-контрактами, важливе для підтримки цілісності системи.

   Рішення:

   - Використання цифрових підписів для перевірки оновлень та змін в коді.
   - Постійний контроль за цілісністю даних та програмного забезпечення.

9. **Відсутність логування та моніторингу (Security Logging and Monitoring Failures)**
   Важливо впровадити системи моніторингу, щоб відстежувати аномальні транзакції або несанкціоновані спроби зняття коштів.

   Рішення:

   - Інтеграція з системами для моніторингу транзакцій в реальному часі.
   - Використання логування для всіх важливих дій у смарт-контрактах.

10. **Серверне підроблення запитів (Server-Side Request Forgery — SSRF)**
    Захист від атак, коли смарт-контракт може виконувати небажані запити на сторонні сервери, що загрожує безпеці платформи.

    Рішення:

    - Фільтрація запитів до сторонніх серверів, щоб уникнути SSRF атак.
    - Впровадження політик доступу для сторонніх запитів.