НТУУ "Київський політехнічний інститут імені Ігоря Сікорського" НН
Фізико-технічний інститут

Технологія блокчейн та розподілені системи

Лабораторна робота №3

Виконали:
Ярощук Владислав ФБ-31мн
Осінній Максим ФБ-31мн
Золотов Іван ФБ-31мп

Київ - 2024

Вступ

Веб-безпека є критично важливим аспектом розробки та підтримки веб-додатків. З ростом кількості кіберзагроз, підвищенням складності атак та розширенням використання децентралізованих додатків, виникає необхідність розробки надійних методів захисту. Одним із найвідоміших і авторитетних стандартів у цій галузі є OWASP (Open Web Application Security Project). Цей реферат спрямований на дослідження вимог OWASP для безпеки веб-додатків та розробку аналогічних вимог для обраної системи децентралізованих додатків.

1. Загальний огляд OWASP

1.1. Що таке OWASP?

OWASP (Open Web Application Security Project) – це відкрита спільнота, яка створює методики, документацію, інструменти та стандарти для покращення безпеки веб-додатків. Основна мета OWASP – надання розробникам та організаціям інформації та інструментів для забезпечення безпеки їх додатків.

1.2. OWASP Top Ten

OWASP Top Ten – це список найкритичніших ризиків безпеки веб-додатків, який оновлюється кожні три роки. У цей список входять:

- Ін'єкції
- Втрата автентифікації
- Втрата конфіденційності
- Небезпечне налаштування безпеки
- Міжсайтовий скриптинг (XSS)
- Небезпечні контрольні дії доступу
- Використання компонентів з відомими вразливостями
- Недостатня реєстрація та моніторинг
- Недостатнє керування сесіями
- Ненадійне програмне забезпечення

2. Дослідження вимог OWASP

2.1. Ін'єкції

Ін'єкційні атаки відбуваються, коли зловмисник вводить зловмисний код в запити до бази даних, системних команд або інших інтерфейсів. Для запобігання цим атакам рекомендується використовувати параметризовані запити, зберігання паролів у хешованому вигляді та регулярне оновлення бібліотек. Наприклад, SQL-ін'єкції можна запобігти шляхом використання підготовлених запитів та зберігання даних у захищеному вигляді.

2.2. Втрата автентифікації

Недостатня автентифікація може призвести до несанкціонованого доступу до системи. OWASP рекомендує використовувати багатофакторну автентифікацію, регулярну зміну паролів та сильні політики паролів. Додатково, важливо впроваджувати механізми блокування акаунтів після кількох невдалих спроб входу та регулярно оновлювати методи автентифікації.

2.3. Втрата конфіденційності

Конфіденційні дані, такі як персональна інформація користувачів, повинні бути захищені за допомогою шифрування як під час передачі, так і на зберіганні. Використання HTTPS та інших методів шифрування є обов'язковим. Наприклад, для захисту даних користувачів можна використовувати SSL/TLS для захищених з'єднань та AES для шифрування збережених даних.

2.4. Небезпечне налаштування безпеки

Небезпечні налаштування безпеки включають використання стандартних паролів, незахищені конфігурації серверів та додатків, а також відсутність оновлень безпеки. Рекомендується проводити регулярні аудити конфігурацій, оновлювати програмне забезпечення та застосовувати налаштування безпеки за замовчуванням. Наприклад, для захисту серверів можна використовувати інструменти автоматичного оновлення та управління конфігураціями.

2.5. Міжсайтовий скриптинг (XSS)

Міжсайтовий скриптинг (XSS) дозволяє зловмисникам виконувати зловмисний код у браузерах користувачів. Для запобігання XSS-атакам рекомендується використовувати методи очищення та екранування даних, а також впроваджувати політики безпеки контенту (Content Security Policy, CSP). Наприклад, можна використовувати бібліотеки для очищення введених користувачами даних та встановлювати заголовки CSP на сервері.

2.6. Небезпечні контрольні дії доступу

Ненадійне управління доступом може призвести до несанкціонованого доступу до критичних функцій або даних. Для запобігання цим проблемам рекомендується впроваджувати політики контролю доступу на основі ролей (Role-Based Access Control, RBAC), а також регулярно перевіряти права доступу. Наприклад, можна використовувати бібліотеки RBAC для розподілу прав доступу на основі ролей користувачів.

2.7. Використання компонентів з відомими вразливостями

Використання застарілих або вразливих компонентів може призвести до компрометації системи. Рекомендується регулярно оновлювати всі компоненти та використовувати інструменти для моніторингу вразливостей. Наприклад, можна використовувати інструменти для управління залежностями, такі як Dependabot, для автоматичного оновлення бібліотек.

2.8. Недостатня реєстрація та моніторинг

Відсутність або недостатня реєстрація та моніторинг подій безпеки може призвести до запізнілого виявлення атак. Рекомендується впроваджувати системи журналювання та моніторингу, а також регулярно аналізувати журнали подій. Наприклад, можна використовувати інструменти для централізованого журналювання, такі як ELK Stack, для збирання та аналізу журналів подій.

2.9. Недостатнє керування сесіями

Недостатнє керування сесіями може призвести до захоплення сесій та несанкціонованого доступу. Рекомендується впроваджувати захищені механізми керування сесіями, такі як використання захищених кукі

(HTTPOnly, Secure), регулярне оновлення сесійних ідентифікаторів та встановлення таймаутів сесій. Наприклад, можна використовувати бібліотеки для керування сесіями, які забезпечують захист від атак на захоплення сесій.

2.10. Ненадійне програмне забезпечення

Ненадійне програмне забезпечення може містити вразливості, які можуть бути використані зловмисниками. Рекомендується проводити регулярні перевірки коду, використовувати автоматизовані інструменти для аналізу безпеки коду (SAST, DAST), а також впроваджувати безпечні методики розробки. Наприклад, можна використовувати інструменти для статичного аналізу коду, такі як SonarQube, для виявлення вразливостей у коді.

3. Аналіз вимог до безпеки децентралізованих додатків

3.1. Що таке децентралізовані додатки?

Децентралізовані додатки (dApps) працюють на основі блокчейна і забезпечують прозорість, безпеку та децентралізоване управління без єдиної точки відмови. Вони використовують смарт-контракти для автоматизації бізнес-процесів.

3.2. Вимоги до безпеки децентралізованих додатків

Розробка вимог безпеки для децентралізованих додатків включає:

- Захист смарт-контрактів від вразливостей, таких як переповнення цілого числа, атаки повторного входу та інші.
- Забезпечення конфіденційності даних на блокчейні через використання методів шифрування та анонімізації.
- Використання надійних методів автентифікації та авторизації, зокрема багатофакторної автентифікації.
- Регулярні аудити безпеки смарт-контрактів та оновлення їх у випадку виявлення вразливостей.

4. Складання вимог безпеки для обраної системи децентралізованих додатків

4.1. Аналіз обраної системи

Для складання вимог безпеки було обрано систему децентралізованих фінансових додатків (DeFi), яка дозволяє користувачам проводити фінансові операції без посередників. Основними компонентами системи є смарт-контракти, криптовалютні гаманці та інтерфейс користувача.

4.2. Вимоги до безпеки

На основі аналізу вимог OWASP та специфіки обраної системи, були сформульовані наступні вимоги:

- Безпека смарт-контрактів: використання перевірених бібліотек, проведення аудиту безпеки, тестування на вразливості.
- Захист користувацьких даних: шифрування даних на блокчейні, забезпечення конфіденційності транзакцій.
- Автентифікація та авторизація: багатофакторна автентифікація, використання надійних методів авторизації.
- Управління доступом: розробка політик доступу, обмеження прав доступу до критичних компонентів системи.

4.3. Конкретні заходи безпеки для DeFi

4.3.1. Безпека смарт-контрактів

Смарт-контракти є серцем DeFi додатків, тому їх безпека є критично важливою. Необхідно впроваджувати наступні заходи:

- Використання перевірених бібліотек, таких як OpenZeppelin, для зменшення ризиків вразливостей.
- Проведення аудиту безпеки сторонніми компаніями для виявлення потенційних вразливостей.
- Тестування смарт-контрактів з використанням інструментів, таких як MythX та Slither, для автоматичного аналізу коду на наявність помилок.

4.3.2. Захист користувацьких даних

Конфіденційність та цілісність даних є важливими аспектами безпеки DeFi додатків:

- Використання шифрування даних на блокчейні для забезпечення конфіденційності транзакцій.
- Впровадження механізмів анонімізації, таких як використання zk-SNARKs, для захисту приватності користувачів.
- Зберігання мінімального обсягу конфіденційної інформації на блокчейні та використання off-chain методів зберігання для додаткового захисту.

4.3.3. Автентифікація та авторизація

Автентифікація та авторизація в DeFi додатках повинні бути надійними:

- Використання багатофакторної автентифікації (2FA) для підвищення рівня безпеки.
- Впровадження системи авторизації на основі ролей (RBAC) для обмеження доступу до критичних функцій.
- Використання криптографічних методів для підтвердження транзакцій, таких як цифрові підписи.

4.3.4. Управління доступом

Ефективне управління доступом є ключовим для забезпечення безпеки DeFi додатків:

- Розробка чітких політик доступу для кожного рівня системи.
- Встановлення обмежень на доступ до критичних компонентів, таких як смарт-контракти та бази даних.
- Регулярний перегляд та оновлення прав доступу для всіх користувачів.

5. Розширений аналіз вразливостей та їх виправлення

5.1. Переповнення цілого числа

Однією з найбільш розповсюджених вразливостей у смарт-контрактах є переповнення цілого числа. Це відбувається, коли значення змінної перевищує її максимальне значення, що може призвести до непередбачуваних результатів.

Виправлення:

- Використання бібліотек для безпечної арифметики, таких як SafeMath.
- Проведення ретельного тестування всіх арифметичних операцій.

5.2. Атаки повторного входу

Атака повторного входу відбувається, коли зловмисник може викликати функцію смарт-контракту повторно до завершення попереднього виклику.

Виправлення:

- Використання шаблону Checks-Effects-Interactions.
- Блокування повторних викликів за допомогою перемикачів (mutex).

5.3. Відсутність перевірки прав доступу

Відсутність перевірки прав доступу може дозволити зловмисникам виконувати дії, на які вони не мають права.

Виправлення:

- Впровадження ролей та прав доступу.
- Регулярний аудит прав доступу.

6. Практичні приклади та реальні випадки

6.1. DAO Hack

У 2016 році відбулася одна з найбільших атак на децентралізовані додатки – DAO Hack, в результаті якої було викрадено приблизно 60 мільйонів доларів в криптовалюті.

Аналіз:

- Вразливість виникла через помилку в смарт-контракті, яка дозволила атаці повторного входу.

Виправлення:

- Проведення ретельних аудитів безпеки перед впровадженням смарт-контрактів у продуктивне середовище.
- Впровадження шаблону Checks-Effects-Interactions.

6.2. Parity Wallet Hack

У 2017 році відбулася ще одна велика атака, відома як Parity Wallet Hack, в результаті якої було заморожено криптовалюту на суму понад 150 мільйонів доларів.

Аналіз:

- Вразливість виникла через помилку в бібліотеці, яка використовувалася для управління гаманцями.

Виправлення:

- Використання перевірених бібліотек та проведення аудитів безпеки.
- Регулярне оновлення бібліотек та компонентів.

7. Приклад списку вимог

Складання списку вимог до безпеки для децентралізованого обмінного сервісу (DEX) на основі аналогічних принципів OWASP може допомогти забезпечити надійність і захист від різноманітних загроз. Ось такі вимоги можуть бути застосовані:

Категорія	Вимога	Опис
Аутентифікація та авторизація	Надійна аутентифікація	Використання двофакторної аутентифікації (2FA) для забезпечення додаткового рівня захисту.
	Контроль доступу	Використання ролей і дозволів для обмеження доступу до критичних функцій та даних.
	Криптографічні токени	Використання захищених токенів для сесій користувачів.
Безпека смарт-контрактів	Аудит смарт-контрактів	Проведення регулярних аудиторських перевірок смарт-контрактів для виявлення вразливостей.
	Використання перевірених бібліотек	Використання перевірених і надійних бібліотек та фреймворків для розробки смарт-контрактів.

	Обмеження прав доступу	Обмеження доступу до функцій смарт-контрактів для мінімізації ризиків.
Захист від ін'єкційних атак	Підготовлені запити	Використання підготовлених запитів або ORM для взаємодії з базою даних, щоб уникнути SQL-ін'єкцій.
	Валідація вхідних даних	Фільтрація та валідація всіх вхідних даних для запобігання ін'єкційним атакам.
Безпека інтерфейсів API	Використання HTTPS	Використання протоколу HTTPS для захисту даних під час передачі між клієнтом і сервером.
	Валідація та фільтрація запитів	Валідація та фільтрація всіх запитів до API для запобігання атакам через API.
	Обмеження швидкості запитів	Впровадження rate limiting для запобігання DDoS-атакам.
Захист від атак на ланцюжок поставок	Використання перевірених джерел	Використання надійних і перевірених джерел для всіх залежностей.

	Регулярні перевірки оновлень	Регулярні перевірки на наявність оновлень та виправлення вразливостей у використовуваних бібліотеках.
Конфіденційність даних	Шифрування даних	Шифрування чутливих даних як у стані спокою, так і під час передачі.
	Мінімізація збору даних	Збір лише необхідних для роботи персональних даних користувачів.
Моніторинг та логування	Моніторинг аномальної активності	Впровадження системи моніторингу для виявлення аномальної активності.
	Логування важливих подій	Логування всіх важливих подій та спроб доступу для подальшого аналізу.
Захист від атак на клієнтські додатки	Використання Content Security Policy (CSP)	Використання політики безпеки вмісту для захисту від XSS-атак.
	Регулярні перевірки та оновлення	Регулярні перевірки та оновлення клієнтського ПЗ для захисту від нових вразливостей.

Безперервність бізнесу та відновлення після аварій	План резервного копіювання та відновлення	Створення та регулярне тестування плану резервного копіювання та відновлення.
	Тестування процедур відновлення	Регулярне тестування процедур відновлення після аварій для забезпечення готовності.
Освіта та навчання	Регулярні тренінги з безпеки	Проведення регулярних тренінгів з безпеки для розробників та співробітників.
	Інформування користувачів	Інформування користувачів про основи безпеки та рекомендації з захисту їхніх акаунтів.

Цей список вимог допоможе забезпечити комплексний підхід до безпеки для децентралізованого обмінного сервісу, зменшуючи ризики та забезпечуючи захист користувачів і їх активів.

Висновок

Забезпечення безпеки веб-додатків та децентралізованих додатків є надзвичайно важливим для запобігання кіберзагрозам. Вимоги OWASP надають чіткі керівництва для розробки безпечних веб-додатків, які можуть бути адаптовані для децентралізованих систем з урахуванням їх специфіки. Виконання цих вимог допоможе забезпечити високий рівень безпеки та захисту даних користувачів.

Список джерел

1. https://owasp.org/www-project-top-ten/
2. https://qagroup.com.ua/publications/tpo-10-vulnerability-owasp/
3. https://academy.binance.com/uk/articles/what-are-decentralized-applications-dapps
4. https://coinmarketcap.com/academy/uk/glossary/decentralized-applications-dapps
5. https://www.coinbase.com/ru/learn/crypto-basics/what-is-a-dex
6. https://www.kraken.com/uk-ua/learn/what-is-decentralized-finance-defi