Звіт з виконання лабораторної роботи 3
Тип 2
"Дослідження вимог OWASP та складання аналогічних вимог для обраної системи децентралізованих додатків"

ФІ-42мн Бондар Петро
ФІ-42мн Кістаєв Матвій

20 червня 2025 р.

Мета: Дослідження безпечної реалізації та експлуатації децентралізованих додатків.

Задача: Ознайомитись із вимогами та рекомендаціями OWASP для розробки WEB-додатків, дослідити наявність стандартів OWASP, що стосуються розробки розподілених додатків, або можливість перенесення існуючих практик на розподілені додатки.

Що таке OWASP?

OWASP — це відкрита спільнота фахівців з кібербезпеки, що створює вільно доступні методики, стандарти, інструменти та рекомендації з метою підвищення рівня безпеки програмного забезпечення. Найвідомішим документом OWASP є OWASP Top 10 — список десяти найкритичніших ризиків безпеки веб-додатків, який регулярно оновлюється з урахуванням нових загроз. До цього списку, зокрема, входять такі загрози, як SQL-ін'єкції, недостатній контроль доступу, вразливості XSS, незахищена ідентифікація та автентифікація, а також небезпеки пов'язані з неправильним конфігуруванням безпеки.

Рекомендації OWASP не обмежуються лише переліком загроз — організація також пропонує практичні методи запобігання їм. Наприклад, використання вхідної валідації, принципу найменших привілеїв, шифрування чутливих даних, використання безпечних фреймворків та бібліотек, а також регулярне проведення penetration testing.

Таким чином, дотримання стандартів OWASP є критично важливим для розробників веб-додатків, адже дозволяє значно знизити ризики компрометації даних та зберегти довіру користувачів.

Smart Contract Security Verification Standard (SCSVS)

Наразі, документом OWASP, найбільш релевантним для розробки децентралізованих додатків, є стандарт SCSVS, який є списком вимог та рекомендацій для перевірки безпеки смарт-контрактів, орієнтований Solidity-контракти в мережі Ethereum.

Стандарт поділений на 11 розділів і описує вимоги до:

- Архітектури,
- Контролю доступу,
- Логіки бізнесу,
- Взаємодії з оракулами,
- Криптографії,
- Економічної безпеки.

Рівні перевірки безпеки

Стандарт SCSVS класифікує перевірку безпеки на три окремі рівні, кожен з яких орієнтований на різні рівні впевненості у безпеці під час розробки та розгортання смарт-контрактів:

1. *L1 — Базова безпека:* Цей рівень призначений для смарт-контрактів з невисокими ризиками. Він зосереджується на фундаментальних заходах безпеки, забезпечуючи базовий захист для будь-якого децентралізованого додатку.
2. *L2 — Помірна безпека:* Ідеально підходить для смарт-контрактів, які обробляють чутливі дані, фінансові транзакції або є частиною DeFi-екосистеми. Рівень 2 забезпечує збалансований підхід до безпеки, включаючи захист від поширених вразливостей, таких як атаки повторного входу (reentrancy), неефективність витрат газу та слабкий контроль доступу.

3. *L3 — Високий рівень впевненості в безпеці:* Цей рівень орієнтований на критично важливі смарт-контракти, в яких задіяні значні фінансові активи, управління або транзакції з великою цінністю. Рівень 3 передбачає широкі заходи захисту, включаючи формальну верифікацію, багатопідписні гаманці та децентралізоване управління.

Наведемо декілька прикладів вимог із стандарту SCSVS:

S4.1 — Role-Based Access Control (RBAC)

Однією з ключових вимог безпеки для смарт-контрактів є впровадження чіткого контролю доступу до функцій і даних. У специфікації OWASP SCSVS вимога S4.1 стосується впровадження механізму контролю доступу на основі ролей — Role-Based Access Control (RBAC).

RBAC передбачає, що кожному користувачеві (адресі) у системі призначається певна роль (наприклад, owner, admin, minter, pauser), а кожна роль має чітко визначені дозволи. Такий підхід дозволяє ізолювати критичні функції (наприклад, випуск токенів, заморожування контракту, оновлення конфігурації) та запобігає їх несанкціонованому використанню.

Цей підхід гарантує, що лише адреси з відповідною роллю можуть викликати критичні функції контракту. Недотримання принципу RBAC може призвести до ситуацій, коли звичайні користувачі (або зловмисники) отримують контроль над важливими функціями контракту, що, своєю чергою, може призвести до фінансових втрат або повної компрометації системи.

S4.1.B Identity Verification

Ref	Requirement	L1	L2	L3	SWE
S4.1.B1	Validate that unexpected addresses do not result in unintended behaviors, particularly when these addresses refer to contracts within the same protocol.		✓	✓	
S4.1.B2	Verify that functions like ecrecover handle all potential null addresses properly to avoid vulnerabilities arising from unexpected ecrecover outputs.		✓	✓	

S4.1.C Least Privilege Principle

Ref	Requirement	L1	L2	L3	SWE
S4.1.C1	Use msg.sender instead of tx.origin for authorization to avoid potential abuse from malicious contracts; include checks like require(tx.origin == msg.sender) to ensure the sender is an EOA.		✓	✓	
S4.1.C2	Certain addresses might be blocked or restricted from receiving tokens (e.g., LUSD). Ensure that address restrictions are properly managed and verified.		✓	✓	
S4.1.C3	Ensure that Guard's hooks (e.g., checkTransaction(), checkAfterExecution()) are executed to enforce critical security checks.		✓	✓	
S4.1.C4	Ensure that access controls are implemented correctly to determine who can use certain functions, and avoid unauthorized changes or withdrawals.		✓	✓	

Табл. 1: Вимоги S4.1 із стандарту

S7.1 — Preventing Overflow/Underflow

Вимога S7.1 стандарту OWASP SCSVS зосереджується на запобіганні помилкам переповнення (overflow) та заниження (underflow) при роботі з цілими числами.

Переповнення виникає тоді, коли значення типу uint перевищує максимальне допустиме значення і «обертається» до нуля. Аналогічно, underflow трапляється при відніманні від нуля, що призводить до переходу до максимального значення. У контексті блокчейн-додатків такі помилки можуть призвести до серйозних фінансових наслідків: неправильного підрахунку токенів, втрати балансу або навіть виведення надмірних коштів.

Запобігання непередбаченим помилкам забезпечується використанням спеціальної бібліотеки арифметики SafeMath.

S7.1.A Use of Safe Math Libraries

Ref	Requirement	L1	L2	L3	SWE
S7.1.A1	Verify that explicit type casting does not lead to overflow or underflow issues.		✓	✓	
S7.1.A2	Verify that integer arithmetic operations do not exceed their bounds to prevent integer overflow or underflow vulnerabilities.		✓	✓	
S7.1.A3	Ensure that operations involving time units and other expressions handle potential overflows correctly.		✓	✓	
S7.1.A4	Verify that division by zero is correctly handled and causes a transaction revert to prevent unexpected behavior.		✓	✓	
S7.1.A5	Ensure that variables are managed within their bounds to prevent reverts due to exceeding limits.		✓	✓	
S7.1.A6	Ensure that arithmetic operations within the unchecked{} block are carefully managed to prevent unintentional overflow or underflow.		✓	✓	
S7.1.A7	Confirm that inline assembly operations handle division by zero and overflow/underflow according to desired behavior and revert appropriately.		✓	✓	
S7.1.A8	Implement checks to handle cases where operations might introduce unintended precision issues or rounding errors.		✓	✓	

Табл. 2: Вимоги S7.1 із стандарту

S6.3 — Secure Random Number Generation

Генерація випадкових чисел у смарт-контрактах має критичне значення в таких застосунках, як лотереї, NFT-мінтинг, ігри, механізми вибору переможця тощо. Вимога S6.3 стандарту OWASP SCSVS вимагає забезпечення криптографічно стійкого і надійного способу генерації випадкових значень у децентралізованих умовах.

Ethereum середовище виконання є детермінованим, а отже — не підтримує справжню ентропію. Тому будь-яка "локальна" генерація, як-от на основі `block.timestamp`, `block.number` або `msg.sender`, є небезпечною, оскільки може бути передбачена або змінена майнерами чи валідаторами.

Для безпечної генерації випадкових чисел необхідно використовувати спеціалізовані оракули, які постачають ентропію ззовні. Найпопулярніший варіант — Chainlink VRF (Verifiable Random Function), який забезпечує криптографічно перевірювану випадковість.

S6.3.A Best Practices for Randomness

Ref	Requirement	L1	L2	L3	SWE
S6.3.A1	Ensure that random number generation follows best practices and uses secure sources of entropy.		✓	✓	
S6.3.A2	Verify that any random number generation is resistant to manipulation and prediction.		✓	✓	

Табл. 3: Вимоги S6.3A із стандарту

Chainlink VRF

Chainlink VRF (Verifiable Random Function) — це децентралізований сервіс, який надає смарт-контрактам безпечні та криптографічно перевірені випадкові числа. Він поєднує генерацію випадковості поза блокчейном із перевіркою достовірності на блокчейні.

1. *Контракт надсилає запит*
   Функція смарт-контракту викликає функцію `requestRandomWords(seed)` у допоміжному смарт-контракті VRFCoordinator, щоб отримати випадкове число.
2. *Запит потрапляє в мережу Chainlink*
   Вузли Chainlink відстежують подію запиту та бачать, що контракт запитує випадкове значення.
3. *Генерація випадкового числа (off-chain)*
   Один із вузлів Chainlink генерує випадкове число за допомогою криптографічної функції VRF, використовуючи свій приватний ключ і `seed` запиту. Також формується доказ для доведення без розголошення, що число було згенеровано правильно.
4. *Повернення результату на блокчейн*
   Вузол надсилає число та доказ до VRFCoordinator на блокчейні у вигляді виклику відповідної функції.
5. *Перевірка доказу (on-chain)*
   Контракт VRFCoordinator перевіряє доказ і, якщо він дійсний, викликає функцію `fulfillRandomWords()` у початковому смарт-контракті.
6. *Отримання випадкового числа*
   Контракт отримує перевірене випадкове число, та може продовжувати виконання логіки викликаної функції.

Висновки

У результаті дослідження було встановлено, що безпечна розробка децентралізованих додатків вимагає, в першу чергу, забезпечення захисту смарт-контрактів, на яких працює додаток. Стандарт OWASP SCSVS пропонує багаторівневу систему перевірки безпеки, яка охоплює ключові аспекти, такі як контроль доступу (RBAC), перевірка арифметичних операцій, генерація випадковості (наприклад, через Chainlink VRF), криптографія, захист від DoS-атак та забезпечення прозорого логування. Дотримання цих вимог дозволяє зменшити ризики вразливостей, зберегти довіру користувачів і створювати надійні децентралізовані додатки.