Доповідач 1

Доброго дня! Сьогодні ми зосередимо нашу увагу на безпечній реалізації та подальшій експлуатації децентралізованих додатків, або DApps, адже їхня архітектура і принципи роботи суттєво відрізняються від тих, що ми звикли бачити в традиційних веб-додатках. По-перше, DApp — це програмний продукт, який функціонує на базі розподіленої мережі блокчейну або іншої peer-to-peer інфраструктури, де замість єдиного централізованого сервера дані зберігаються на великій кількості вузлів: кожен вузол, або нод, тримає копію повного журналу транзакцій або його частину. Це гарантує високу стійкість до втрати даних і здебільшого забезпечує неможливість приховування факту певних операцій. До того ж, завдяки вбудованим криптографічним механізмам жодна записана інформація не може бути змінена заднім числом, що створює додатковий рівень довіри до цілісності мережі.

По-друге, логіка роботи DApp реалізується через смарт-контракти — спеціальні програми, що виконуються безпосередньо в рамках віртуальної машини блокчейну. Вони автоматизують бізнес-процеси й усувають потребу в посередниках, але при цьому стають критичними точками в системі: будь-яка помилка в коді чи валідації вхідних параметрів може призвести до незворотних фінансових втрат або витоку коштів. Окрім звичних для веб-додатків вразливостей — переповнення, некоректна обробка даних чи повторний виклик функцій — доводиться враховувати ризики, пов’язані з атакою на консенсус, коли зловмисник, контролюючи більше половини обчислювальних потужностей (51 % атака), може цензурувати транзакції чи навіть переписувати історію.

Третій важливий аспект — це безпека приватних ключів користувачів. У традиційних системах для аутентифікації достатньо пароля, тоді як тут приватний ключ одночасно є паролем і підписом, тож його компрометація відкриває зловмиснику повний контроль над активами. Нарешті, додаткові сервіси, зокрема децентралізоване зберігання (наприклад, IPFS), також можуть страждати від впливу DoS-атак або атак на передбачуваність контенту, якщо не впроваджені належні механізми реплікації та авторизації доступу.

Наша головна мета полягає в тому, щоб не лише ознайомитися з особливостями роботи DApps, а й систематично оцінити рівень їхньої інформаційної безпеки під час реального функціонування. Перший етап дослідження передбачає вивчення та практичне застосування **OWASP Top 10 Web Application Security Risks** для класичних веб-додатків, аби зрозуміти основні категорії вразливостей і методи їхньої виявлення. Далі ми адаптуємо ці вимоги до контексту блокчейн-середовища, формуючи набір рекомендацій і критеріїв безпеки, специфічних для смарт-контрактів, управління ключами, захищеного розгортання нод та моніторингу транзакцій. У наступних розділах ми продемонструємо конкретні приклади загроз та проаналізуємо, як традиційні ризики трансформуються у світі децентралізованих додатків.

Доповідач 2

Тепер коротко розглянемо **OWASP Top 10** і приклади загроз, які стосуються традиційних веб-додатків:

1. **A1: Broken Access Control**
   — Неправильне налаштування прав користувачів, можливість отримати доступ до чужих ресурсів.

2. **A2: Cryptographic Failures**
   — Слабке шифрування, неправильне зберігання секретів, небезпечні алгоритми.

3. **A3: Injection**
   — SQL-ін’єкції, XSS, Command Injection — коли користувацькі дані потрапляють у запит без валідації.

4. **A4: Insecure Design**
   — Недостатня увага до захисту під час проєктування архітектури.

5. **A5: Security Misconfiguration**
   — Неправильно налаштовані заголовки, сервіси або середовище виконання.

6. **A6: Vulnerable and Outdated Components**
   — Використання застарілих бібліотек із відомими уразливостями.

7. **A7: Identification and Authentication Failures**
   — Слабкі механізми входу, вразливі токени, недостатній контроль сесій.

8. **A8: Software and Data Integrity Failures**
   — Відсутність перевірки цифрових підписів, ризик підміни коду.

9. **A9: Security Logging and Monitoring Failures**
   — Неможливість вчасно виявити або відстежити інцидент.

10. **A10: Server-Side Request Forgery (SSRF)**
    — Примусове виконання запитів сервером до внутрішніх ресурсів.

Ми дослідили кожен пункт: які вразливості виникають, як їх виявляти та запобігати. Наприклад, для A3 Injection використовували **фреймворк OWASP ZAP**, протестували прості веб-форми на XSS. Для A2 Cryptographic Failures вивчили принципи безпечного зберігання паролів і секретів із використанням **bcrypt** та **KMS-сервісів**.

Але які з цих ризиків є критичними для DApp? Наприклад:

- **Injection** трансформується у вразливості смарт-контрактів (Reentrancy, Integer Overflow).

- **Authentication Failures** стають питанням безпечного зберігання приватних ключів та управління правами доступу через мульти-сигнатури.

- **Data Integrity Failures** набувають значення при оновленні смарт-контрактів — необхідно передбачити механізм **upgradable contracts** з контролем версій.

Доповідач 3

Тепер запропонуємо **аналогічні вимоги безпеки для DApps** на основі виявлених ризиків:

1. **DBC1: Control of Smart-Contract Access**
   — Забезпечити чіткі ролі (owner, admin, user), обмежити виклики адмін-функцій через мультисигнатури та timelock.

2. **DBC2: Secure Key Management**
   — Використовувати апаратні гаманці (Ledger, Trezor) або дедиковані KMS для захисту приватних ключів; запровадити політики ротації та бекапу.

3. **DBC3: Integer Safety & Overflow Protection**
   — Застосовувати бібліотеки SafeMath, валідувати вхідні параметри смарт-контрактів та використовувати останні версії Solidity.

4. **DBC4: Upgradability & Code Review**
   — Використовувати проксі-контракти для оновлення логіки й одночасно зберігати дані; організовувати **дефайд тестування** (audit) та рев'ю сторонніми аудиторами.

5. **DBC5: Consensus & Node Security**
   — Захищати приватні RPC-ендпоінти, слідкувати, щоб ноди працювали на актуальному ПЗ, обмежити доступ через firewall.

6. **DBC6: Transaction Monitoring & Alerts**
   — Налаштувати моніторинг незвичайних транзакцій із підозрілими обсягами або частими зняттями; інтегрувати webhook-сповіщення та SIEM.

7. **DBC7: Data Availability & DoS Protection**
   — Використовувати декілька провайдерів RPC, забезпечити кешування, балансування, децентралізоване зберігання через IPFS із pinning-сервісами, встановити ліміти gas price/gas limit.

8. **DBC8: User-Facing Security UX**
   — Інформувати користувача про ризики транзакцій (зміна прав, великий gas fee), попереджати про phishing-сайти через white-label доменні списки.

Підсумовуючи, ми:

- Вивчили **OWASP Top 10**, протестували класичні веб-додатки на вразливості.
- Систематизували та адаптували ці вимоги для **децентралізованих додатків**.
- Розробили рекомендації, що охоплюють безпеку смарт-контрактів, управління ключами, моніторинг і захист нод.

Дякуємо за увагу!

Код	Назва ризику	Короткий опис	Приклад тестування / інструмент
A1	Broken Access Control	Неправильні права доступу до ресурсів	OWASP ZAP, ручна перевірка ACL
A2	Cryptographic Failures	Слабке або відсутнє шифрування, небезпечне зберігання секретів	Аналіз налаштувань TLS, bcrypt
A3	Injection	Впровадження небезпечного коду (SQL, XSS, Command Injection)	SQLMap, Burp Suite
A4	Insecure Design	Недостатній захист на рівні архітектури	Архітектурний рев'ю
A5	Security Misconfiguration	Неправильні налаштування серверів, заголовків, середовища	OWASP Benchmark, автоматизовані сканери
A6	Vulnerable and Outdated Components	Використання застарілих бібліотек із відомими вразливостями	Dependency-сканери (Snyk, Dependabot)
A7	Identification and Authentication Failures	Ненадійні механізми аутентифікації, слабкі сесійні токени	OWASP Auth Cheat Sheet, ручний аудит
A8	Software and Data Integrity Failures	Відсутність перевірки цілісності коду й даних	Перевірка цифрових підписів, CI/CD
A9	Security Logging and Monitoring Failures	Недостатнє логування та моніторинг подій безпеки	SIEM, ELK-стек
A10	Server-Side Request Forgery (SSRF)	Примусове виконання запитів до внутрішніх чи захищених ресурсів	OWASP ZAP, Burp Collaborator

Код	Вимога	Основні заходи
DBC1	Control of Smart-Contract Access	Використання ролей (owner/admin/user), мульти-підписи, timelock-функції
DBC2	Secure Key Management	Апаратні гаманці (Ledger/Trezor), KMS, політики ротації та резервного копіювання ключів
DBC3	Integer Safety & Overflow Protection	Бібліотеки SafeMath, перевірка діапазонів параметрів, оновлення до нових версій Solidity
DBC4	Upgradability & Code Review	Проксі-контракти для оновлень, аудит сторонніми фахівцями, автоматизовані тести (unit, integration)
DBC5	Consensus & Node Security	Захист RPC-ендпоінтів (firewall, IP-whitelisting), регулярні оновлення нод, моніторинг стану мережі
DBC6	Transaction Monitoring & Alerts	Налаштування сповіщень про підозрілі транзакції, інтеграція із SIEM/Webhook, аналіз аномалій
DBC7	Data Availability & DoS Protection	Використання декількох провайдерів RPC, балансування навантаження, кешування, реплікація через IPFS із pinning-сервісами
DBC8	User-Facing Security UX	Інформування користувача про ризики (зміна прав, великий gas-fee), попередження про фішингові сайти, підписування транзакцій через перевірені UI-елементи