Мурсалов В. Р., здобувач
Головня О. С., к.пед.н., доцент
Державний університет «Житомирська політехніка»

ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ В ДОДАТКАХ НА БАЗІ NODE.JS

Метою даного дослідження є аналіз основних методів та технологій забезпечення безпеки у додатках на базі Node.js.

Один із ключових принципів забезпечення безпеки в Node.js — регулярне оновлення як самого Node.js, так і залежностей проєкту [3]. Застарілі версії можуть містити відомі вразливості, тому використання команд на кшталт npm audit дозволяє оперативно виявити та виправити потенційні загрози. Крім того, слід уникати використання пакетів з низькою довірою або невідомими джерелами, що зменшує ризик шкідливого коду у вашому додатку.

Важливим аспектом є також правильне управління конфігурацією. Зберігати секрети, такі як API-ключі або паролі, у змінних середовища є обов'язковою практикою. Вони не повинні зберігатися у програмному коді або конфігураційних файлах, щоб уникнути їх витоку. Для додаткового рівня безпеки варто використовувати менеджери секретів або спеціалізовані інструменти на кшталт dotenv.

Обробка введених даних є ще одним критично важливим моментом. Node.js-додатки часто вразливі до атак типу SQL-ін'єкцій, XSS та командних ін'єкцій, тому слід завжди валідувати та санітизувати всі вхідні дані. Модулі на кшталт express-validator допомагають у забезпеченні цього захисту. [3]

Захист HTTP-заголовків також має велике значення. Бібліотека helmet дозволяє легко додати стандартний набір безпечних заголовків, що захищають від атак на основі маніпуляцій заголовками [1].

Обмеження рівня доступу є ще одним дієвим методом безпеки. Принцип найменших привілеїв повинен бути застосований як до додатка загалом, так і до окремих його частин. Наприклад, використання облікового запису з мінімальними правами для виконання Node.js-сервісу зменшує потенційний збиток у разі компрометації.

Регулярне логування і моніторинг подій у додатку дозволяють швидко виявляти спроби атак. Інструменти на кшталт winston чи morgan забезпечують ефективне ведення логів та допомагають у виявленні аномальних активностей.

Окрім того, використання механізмів захисту від DDoS-атак, таких як обмеження кількості запитів (rate limiting), захищає сервер від перевантажень. Модулі на кшталт express-rate-limit є зручними засобами для цього.

Також варто подбати про безпеку сесій. У додатках на основі Express.js слід використовувати захищені кукі (secure cookies) з прапорцем HttpOnly, що унеможливлює доступ до них із JavaScript. [3]

Нарешті, слід регулярно тестувати додаток на наявність вразливостей за допомогою таких інструментів, як Snyk [3] або OWASP Dependency-Check [1], що дозволяють виявити та нейтралізувати потенційні загрози на ранніх стадіях розробки.

Дотримання цих рекомендацій дозволяє створювати надійні та захищені додатки на базі Node.js, мінімізуючи ризики як внутрішніх, так і зовнішніх атак.

У даній роботі було проаналізовано практики які дозволяють мінімізувати ризики та підвищити стійкість Node.js-додатків до атак. Регулярне оновлення залежностей, використання валідації введених даних, впровадження обмежень доступу, а також застосування інструментів моніторингу та тестування вразливостей є критично важливими заходами для безпечної роботи додатка. Комплексний підхід до безпеки допомагає захистити систему як від зовнішніх, так і від внутрішніх загроз.

Список використаних джерел:

1. Node.js Security Cheat Sheet - OWASP Cheat Sheet Series. (б. д.). OWASP Cheat Sheet Series. URL: https://cheatsheetseries.owasp.org/cheatsheets/Nodejs_Security_Cheat_Sheet.html (дата звернення: 21.03.2025).
2. Security Best Practices | Node.js. (б. д.). Node.js Documentation. URL: https://nodejs.org/en/learn/getting-started/security-best-practices (дата звернення: 21.03.2025).
3. Best Practice Security - Express. (б. д.). Express.js Documentation. URL: https://expressjs.com/en/advanced/best-practice-security.html (дата звернення: 21.03.2025).