Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(освітній рівень)

на тему: "Дослідження механізмів захисту для серверної частини на платформі Node.js"

Виконав: студент (ка)

Спеціальності:
125 «Кібербезпека та захист інформації»
(шифр і назва напряму підготовки, спеціальності)
Шиндеровський Сергій Вікторович
підпис (прізвище та ініціали)

Керівник Кульчицький Т. Р.
підпис (прізвище та ініціали)

Нормоконтроль Тимощук Д. І.
підпис (прізвище та ініціали)

Завідувач кафедри Загородна Н.В.
підпис (прізвище та ініціали)

Рецензент Луцик Н.С.
підпис (прізвище та ініціали)

м. Тернопіль – 2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра Кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В.
(підпис) (прізвище та ініціали)
«__» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека та захист інформації
(шифр і назва спеціальності)

Студенту Шиндеровському Сергію Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження механізмів захисту для серверної частини на платформі Node.js

Керівник роботи Кульчицький Тарас Русланович,
Ph.D., старший викладач кафедри КБ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «20» 11 2024 року № № 4/7-1112

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи Інтернет-джерела, технічна документація
.

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. Розділ 1. Актуальність безпеки веб-додатків у програмуванні. Розділ 2. Основні підходи до захисту веб-серверів. Розділ 3. Реалізація заходів безпеки для серверної частини на платформі Node.js. Розділ 4.Охорона праці та безпека в надзвичайних ситуаціях. Висновки. Список використаних джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титулка. 2. Актуальність, новизна та завдання дослідження. 3. Список репортів крадіжок 2023 року. 4. Візуальне представлення системи взаємодії сервера з іншими елементами. 5. Модель загроз STRIDE. 6. Захист від Spoofing. 7. Реалізація HMAC-підпису та Middleware для токена. 8. Фрагмент реалізації запису у лог-журнал. 9. Фрагмент коду для фільтрації та автоматичного налаштування для helmet. 10. Реалізація кешування запитів та обмеження кількості запитів. 11. Реалізація коду для перевірки ролей авторизованих користувачів. 12. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Осухівська Г.М., к.т.н., доцент		
Безпека в надзвичайних ситуаціях	Клепчик В.М., проректор з адміністративно-господарської роботи та будівництва		

7. Дата видачі завдання 20.11.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	20.11	Виконано
2.	Джослідження джерел щодо безпеки серверних	21.11 – 22.11	Виконано
	платформ та Node.js		
3.	Аналіз існуючих механізмів захисту серверів на	22.11 – 23.11	Виконано
	платформі Node.js		
4.	Дослідження джерел щодо методології STRIDE	23.11 – 24.11	Виконано
5.	Аналіз механізмів захисту для методології STRIDE	25.11 – 26.11	Виконано
6.	Реалізація існуючих механізмів захисту стосовно	26.11 – 28.11	Виконано
	методології STRIDE для сервера на Node.js		
7.	Оформлення першого розділу	28.11 – 29.11	Виконано
8.	Оформлення другого розділу	29.11 – 30.11	Виконано
9.	Оформлення третього розділу	30.11 – 06.12	Виконано
10.	Оформлення розділу «Безпека життєдіяльності і основи	07.12 – 08.12	Виконано
	охорони праці»		
11.	Оформлення кваліфікаційної роботи	09.12 – 12.12	Виконано
12.	Нормоконтроль	13.12 – 16.12	Виконано
13.	Перевірка на плагіат	16.12 – 17.12	Виконано
14.	Попередній захист кваліфікаційної роботи	23.12.2024	
15.	Захист кваліфікаційної роботи	27.12.2024	

Студент _____ Шиндеровський С. В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Кульчицький Т. Р.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дослідження механізмів захисту для серверної частини на платформі Node.js// Кваліфікаційна робота «Магістр» // Шиндеровський Сергій Вікторович // Тернопільський національний технічний університет імені Івана Пулюя, Факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБм-62 // Тернопіль, 2024 // С. 79, рис. - 22, табл. - 1, кресл. - 0, додат. – 3.

Ключові слова: Node.js, серверна безпека, STRIDE, MongoDB, JWT, HMAC, XSS, DoS, аутентифікація.

У магістерській роботі досліджено та впроваджено механізми захисту для серверної частини на платформі Node.js з метою забезпечення цілісності, конфіденційності та доступності даних. Основну увагу приділено аналізу загроз та реалізації відповідних заходів захисту, що відповідають методології STRIDE.

В даній роботі метою є розробка ефективних та комплексних механізмів захисту серверної частини на платформі Node.js для мінімізації ризиків кіберзагроз та забезпечення високого рівня безпеки.

В першому розділі описано теоретичні основи інформаційної безпеки, зокрема найпоширеніші загрози для веб-серверів і їх вплив на захищеність систем. В другому розділі описано аналіз загроз відповідно до методології STRIDE, визначенню потенційних вразливостей серверної частини на платформі Node.js, а також підходів до їхньої мінімізації. У третьому розділі детально описано реалізацію заходів захисту для кожного типу загроз, які зазначені в методології STRIDE. В четвертому розділі висвітлено окремі питання охорони праці та безпеки життєдіяльності.

ABSTRACT

Research on Protection Mechanisms for the Server Side on the Node.js Platform // Thesis of educational level "Master"// Shinderovskyi Serhii Victorovich // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Cybersecurity, group SBm-62 // Ternopil, 2024 // P. 79, figs. 22, tables 1, drawings 0, added. – 3.

Keywords: Node.js, server security, STRIDE, MongoDB, JWT, HMAC, XSS, DoS, authentication.

The master's thesis investigates and implements protection mechanisms for the server-side on the Node.js platform to ensure data integrity, confidentiality, and availability. The main focus is on threat analysis and the implementation of corresponding security measures based on the STRIDE methodology.

The goal of this work is to develop effective and comprehensive mechanisms for securing the server side on the Node.js platform in order to minimize the risks of cyber threats and ensure a high level of security.

In the first chapter, the theoretical foundations of information security are described, including the most common threats to web servers and their impact on the security of systems. The second chapter focuses on the analysis of threats according to the STRIDE methodology, the identification of potential vulnerabilities in the server-side platform Node.js, as well as approaches to minimizing them. The third chapter provides a detailed description of the implementation of protective measures for each type of threat mentioned in the STRIDE methodology. The fourth chapter highlights occupational safety and security considerations in the banking sector.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

DoS	—	Denial of Service
DNS	—	Domain Name System
SDLC	—	Software Development Lifecycle
CIA	—	Confidentiality, Integrity, and Availability
SQL	—	Structured Query Language
XSS	—	Cross-Site Scripting(міжсайтовий скриптинг)
JWT	—	JSON Web Token
HMAC	—	Hash-based Message Authentication Code
API	—	Application Programming Interface
CSRF	—	Cross-Site Request Forgery
IoT	—	Internet of Things
AWS	—	Amazon Web Services(Веб-сервіси від Amazon)
SWS	—	Secure Web Services
SSL	—	Secure Sockets Layer (Шар безпечних сокетів)
TLS	—	Transport Layer Security (Безпека транспортного рівня)
MITM	—	Man-in-the-Middle
DDoS	—	Distributed Denial of Service
URL	—	Uniform Resource Locator
CSP	—	Content Security Policy
bcrypt	—	алгоритм хешування паролів

ВСТУП

Актуальність теми магістерської роботи обумовлена швидким зростанням кількості кібератак та необхідністю захисту серверної частини додатків, що обробляють конфіденційні та важливі дані. З розвитком хмарних технологій і популяризацією архітектури клієнт-сервер зростає і кількість потенційних векторів атак, які зловмисники можуть використати для отримання несанкціонованого доступу, модифікації даних або порушення доступності сервісів. Платформа Node.js активно використовується у веб-розробці завдяки своїй продуктивності, масштабованості та можливостям обробки великої кількості одночасних запитів. Водночас це робить її привабливою ціллю для атак, спрямованих на порушення цілісності, конфіденційності та доступності даних.

Захист серверної частини є критичним завданням для забезпечення надійності інформаційних систем, а використання сучасних методологій, таких як STRIDE, дозволяє систематично аналізувати загрози і впроваджувати комплексні механізми захисту. У цьому контексті дослідження і реалізація надійних заходів безпеки для серверних додатків на Node.js є важливими як для розробників, так і для організацій, які мають на меті посилити безпеку своїх інформаційних систем.

Метою дослідження є розробка та впровадження комплексних механізмів захисту для серверної частини на платформі Node.js, що дозволяє мінімізувати ризики кібератак і забезпечити цілісність, конфіденційність та доступність даних. Для досягнення даної мети потрібно здійснити аналіз основних загроз для серверної частини додатків на Node.js, використовуючи методологію STRIDE, яка охоплює такі типи загроз, як підробка, зміна даних, відмова від авторства, розголошення інформації, відмова в обслуговуванні та підвищення привілеїв. На основі даного аналізу потрібно визначити відповідні заходи безпеки для кожного типу загроз та інтегрувати їх у систему.

Завданнями дослідження є:

– провести ґрунтовний аналіз загроз для серверної частини на платформі Node.js з використанням методології STRIDE, що охоплює виявлення потенційних ризиків для цілісності, конфіденційності та доступності даних;

– провести аналіз основних загроз для веб-серверів, використовуючи методологію STRIDE, із докладним описом таких атак, як DoS, SQL-ін’єкції, XSS, експлойти нульового дня, та слабка автентифікація, а також розглянуто сучасні методи їх виявлення і запобігання;

– розробити та впровадити ефективні механізми захисту, які відповідатимуть кожному з типів загроз: підробці, зміні даних, відмові від авторства, витоку інформації, відмові в обслуговуванні та підвищенню привілеїв.

Об’єктом дослідження є серверна частина веб-додатків, розроблена на платформі Node.js, яка обробляє запити користувачів, зберігає та управляє даними, а також забезпечує функціонування прикладного програмного інтерфейсу (API). Особливу увагу приділено аспектам безпеки серверної частини, яка є вразливою до різних видів атак, що можуть порушити цілісність, конфіденційність та доступність даних.

Предметом дослідження є механізми забезпечення безпеки серверної частини на платформі Node.js, спрямовані на захист від основних типів загроз, таких як підробка, модифікація даних, відмова від авторства, витік інформації, відмова в обслуговуванні та підвищення привілеїв. Дослідження зосереджене на методах та інструментах для реалізації заходів захисту, які дозволяють мінімізувати ризики атак і забезпечити стабільність та надійність роботи серверної інфраструктури.

Наукова новизна одержаних результатів кваліфікаційної роботи полягає у комплексному підході до захисту серверної частини веб-додатків на платформі Node.js з урахуванням сучасних загроз, що систематизовані за методологією STRIDE. У процесі дослідження розроблено та впроваджено ряд частково інноваційних рішень для запобігання основним видам атак, а саме: підробці, модифікації даних, відмові від авторства, витоку інформації, відмові в обслуговуванні та підвищенню привілеїв. Запропоновані механізми інтегровані

у серверну інфраструктуру та адаптовані для специфічних вимог Node.js, що підвищує їх ефективність і дозволяє забезпечити комплексний захист.

Практичне значення отриманих результатів полягає у можливості застосування розроблених механізмів захисту для підвищення безпеки серверної частини веб-додатків на платформі Node.js. Впроваджені заходи безпеки, такі як аутентифікація за допомогою JWT, аудит-логи, контроль цілісності даних через HMAC, захист від DoS-атак та розподіл прав доступу, можуть бути використані для побудови надійних і стійких до атак веб-сервісів. Отримані результати можуть стати основою для розробки політик безпеки у сучасних веб-додатках, а також для підготовки фахівців з кібербезпеки, які працюють із платформою Node.js.

Апробація результатів магістерської роботи. Основні результати проведених досліджень обговорювались на: XII науково-технічна конференція «Інформаційні моделі, системи та технології» (м.Тернопіль, Україна).

Публікації. Основні результати кваліфікаційної роботи опубліковано у працях конференції (див. Додаток А).

РОЗДІЛ 1 АКТУАЛЬНІСТЬ БЕЗПЕКИ ВЕБ-ДОДАТКІВ У ПРОГРАМУВАННІ

Безпека веб-додатків є однією з ключових проблем сучасної кібербезпеки, враховуючи стрімке зростання кількості цифрових сервісів та обсягів оброблюваних ними даних. У зв’язку з розвитком технологій і збільшенням інтеграції веб-додатків у різні сфери життя – від банківської справи до соціальних мереж – забезпечення їх захищеності стає критично важливим. Будь-яка вразливість у веб-додатку може бути використана зловмисниками для крадіжки конфіденційної інформації, порушення роботи сервісу чи навіть компрометації цілих інформаційних систем.

Особливої уваги ця проблема набуває на фоні масового впровадження хмарних технологій та мікросервісної архітектури, які збільшують складність систем і розширюють можливі вектори атак. Крім того, підвищення мобільності користувачів та їхньої залежності від цифрових рішень вимагає нових підходів до забезпечення безпеки, адже традиційні методи часто не справляються з сучасними загрозами.

Особливо, якщо враховувати, що інформаційні технології (зокрема, ІКТ — інформаційні та комунікаційні технології) стали невід'ємною частиною людського життя і, без сумніву, займуть значне місце в майбутньому суспільстві. ІКТ значною мірою змінили наше повсякденне життя: міста стають розумними, автомобілі — автономними, кол-центри обслуговуються роботами тощо. Багато гаджетів використовуються будь-де і будь-коли, щоб допомогти людям. Тепер важко знайти галузь, в якій інформаційні технології ще не застосовуються [6].

Актуальність дослідження безпеки серверної частини на платформі Node.js визначається її широким використанням у веб-розробці завдяки високій продуктивності та масштабованості. Проте динамічне середовище виконання, властиве Node.js, потребує спеціалізованих заходів захисту, які враховують специфіку цієї платформи. Тому дослідження загроз і впровадження ефективних рішень для підвищення безпеки веб-додатків не лише актуальне, а й має важливе практичне значення.

Головна різниця між веб-додатком та веб-сайтом в функціоналі та можливостях. Коли був винайдений Інтернет, веб-сайти мали значно меншу функціональність, ніж веб-додатки. Вони були здатні доставляти інформацію користувачам лише через статичний вміст. Потрібно було встановити та запустити програмне забезпечення зі складною функціональністю. Веб-програми були створені, щоб подолати розрив між програмним забезпеченням і статичними сайтами. Вони мали функціональність та інтерактивні елементи користувача, як і програмне забезпечення, але доставлялися за допомогою URL-адреси веб-браузера [1].

Однак з того часу веб-технології значно розвинулися. Більшість сучасних веб-сайтів є складними веб-додатками за своїм дизайном.

1.1 Термінологія веб-додатків, їх роль, та основні переваги

Веб-додатки — це програмне забезпечення, яке функціонує на веб-серверах і забезпечує взаємодію з користувачами через веб-браузери, незалежно від типу пристрою чи операційної системи. Вони стали невід'ємною частиною сьогодення та сучасного цифрового світу, даючи можливість виконувати найрізноманітніші завдання: від банківських операцій і покупок в інтернет-магазинах до взаємодії в соціальних мережах чи участі в онлайн-освіті [1].

Головна відмінність веб-додатків від традиційних десктопних програм полягає у тому, що вони не вимагають інсталяції на пристрої користувача, а доступ до них здійснюється через інтернет. Це забезпечує зручність використання, доступність з будь-якого місця, де є підключення до мережі, і знижує залежність користувача від конкретного обладнання. Завдяки цьому веб-додатки стали ключовим елементом глобалізації, підвищуючи ефективність бізнесу, освіти та повсякденного життя.

Роль веб-додатків у сучасному світі важко переоцінити. Вони є основою для електронної комерції, що стимулює розвиток світової економіки, дозволяючи компаніям охоплювати глобальну аудиторію. У сфері охорони здоров'я веб-додатки забезпечують швидкий доступ до медичних записів, організацію

телемедицини та моніторинг стану пацієнтів у реальному часі. В освітньому середовищі вони відкривають доступ до необмеженого обсягу знань через онлайн-курси, віртуальні класи та інтерактивні навчальні платформи [1].

Крім того, веб-додатки мають критичне значення для автоматизації бізнес-процесів і створення нових моделей управління. Завдяки інтеграції з хмарними технологіями та можливостям масштабування, вони дозволяють компаніям оптимізувати витрати, впроваджувати інновації та швидко адаптуватися до змін на ринку.

Веб-додатки також сприяють покращенню комунікації, забезпечуючи швидкий обмін даними за допомогою електронної пошти, месенджерів чи корпоративних платформ для командної роботи. Вони є інструментом для залучення клієнтів, підвищення їхньої лояльності через персоналізовані сервіси, аналітику поведінки користувачів та інтерактивні функції.

Разом із тим, їхня популярність робить веб-додатки привабливою ціллю для кіберзлочинців. Будь-яка вразливість у коді чи інфраструктурі може призвести до втрати даних, фінансових збитків або репутаційних втрат для компаній. Це підкреслює важливість розробки веб-додатків із дотриманням принципів безпеки на всіх етапах життєвого циклу, від проєктування до розгортання та підтримки [12].

Таким чином, веб-додатки не лише спрощують доступ до інформації та сервісів, але й формують нову екосистему, що змінює спосіб життя, бізнес-процеси та взаємодію між людьми в цифрову епоху.

Веб-додатки мають кілька переваг, і майже всі великі підприємства використовують їх як частину своїх пропозицій для користувачів. Ось деякі з найпоширеніших переваг веб-програм: доступність, ефективний розвиток, простота для користувача та масштабованість.

Доступ до веб-програм можна отримати з усіх веб-браузерів і на різних особистих і бізнес-пристроях. Команди в різних точках доступу можуть отримати спільну інформацію до документів, систем керування вмістом та інших бізнес-служб через веб-додатки на основі підписки.

Якщо більш коротко, то процес розробки веб-додатків є відносно простим і економічно ефективним для бізнесу. Невеликі команди можуть досягти коротких циклів розробки, що робить веб-додатки ефективним і доступним методом створення комп'ютерних програм. Крім того, оскільки одна версія працює в усіх сучасних браузерах і пристроях, тому не доведеться створювати кілька різних ітерацій для кількох платформ.

В плані простоти та зручності, то веб-програми не потребують завантаження користувачами, що забезпечує зручний доступ і усуває необхідність в обслуговуванні з боку кінцевого користувача, а також економить місце на жорсткому диску. Вони автоматично отримують оновлення програмного забезпечення та засобів безпеки, гарантуючи постійну актуальність та знижуючи ризик безпекових порушень.

Стосовно масштабованості, то організації, які використовують веб-програми, можуть додавати користувачів, коли їм потрібно, без додаткової інфраструктури чи дорогого обладнання. Крім того, переважна більшість даних веб-додатків зберігається в хмарі, а це означає, що організаціям не доведеться інвестувати в додатковий обсяг пам'яті для запуску веб-додатків [1].

1.2 Найпоширеніші види кібератак та загроз для веб-серверів

Зростання кіберзагроз є однією з найсерйозніших проблем сучасного цифрового суспільства. Разом із розвитком технологій і масовим впровадженням інтернету в усі сфери життя, кіберзлочинність перетворилася на глобальну загрозу, яка завдає значних економічних, соціальних та навіть політичних збитків. Щороку кількість кібератак зростає, а методи зловмисників стають усе більш витонченими, що підкреслює важливість розробки ефективних стратегій протидії загрозам.

Успішні атаки на веб-додатки, яких з кожним роком їх стає більше, призводять до фінансових втрат компаній і шкоди їх репутації. Найчастіше користувачі стають жертвами кібератак, спрямованих на захоплення обчислювальних ресурсів, облікових записів або даних користувачів.

Користувачі зазвичай більше довіряють веб-додаткам від відомих брендів або компаній, вважаючи, що великі корпорації приділяють належну увагу безпеці своїх додатків. Однак жертвами кібератак стають і великі компанії [9].

Одним із базових чинників збільшення кіберзагроз є постійне збільшення обсягів даних, що передаються через інтернет. Кожного дня мільйони користувачів генерують терабайти інформації — від особистих даних до конфіденційної корпоративної інформації. Цей обсяг даних стає привабливим об'єктом для зловмисників, які прагнуть отримати несанкціонований доступ до фінансової, особистої чи стратегічної чи будь-якої іншої конфіденційної інформації.

Типи кіберзагроз значно урізноманітнилися. Традиційні методи, такі як віруси, трояни чи фішинг, продовжують залишатися актуальними, але останніми роками зростає частота складніших атак. До них належать атаки типу ransomware (вимагальне програмне забезпечення), які блокують доступ до даних користувача до сплати викупу, а також DDoS-атаки (розподілені атаки відмови в обслуговуванні), що спрямовані на перевантаження серверів і виведення систем з ладу. Зокрема, за даними аналітичних компаній, у 2023 році кількість DDoS-атак зросла більш ніж на 20% порівняно з попереднім роком.

Кілька коротких фактів про кібербезпеку. У 2023 році було зафіксовано 2365 кібератак із 343 338 964 жертвами. У 2023 році кількість витоків даних зросла на 72% з 2021 року, що є попереднім історичним рекордом. У 2024 році витік даних у всьому світі коштував у середньому 4,88 мільйона доларів. Електронна пошта є найпоширенішим переносником зловмисного програмного забезпечення, у 2023 році близько 35% зловмисного програмного забезпечення доставлялося електронною поштою. Дев'яносто чотири відсотки організацій повідомили про інциденти безпеки електронної пошти. У 2023 році компрометація бізнес-електронної пошти призвела до збитків на суму понад 2,9 мільярда доларів. Прогнозується, що в період з 2022 по 2032 рік кількість робочих місць у сфері інформаційної безпеки зросте на 32% [2].

Одним із найнебезпечніших векторів загроз стало зростання атак на веб-додатки, які обслуговують критично важливі сервіси, такі як електронна

комерція, онлайн-банкінг і державні системи. Наприклад, атаки типу SQL-ін'єкцій, Cross-Site Scripting (XSS) чи Cross-Site Request Forgery (CSRF) так і залишаються основними інструментами для злому веб-додатків. За даними звіту OWASP, понад 40% вразливостей у веб-додатках спричинені недостатньою обробкою користувацьких даних, що робить ці атаки відносно простими для реалізації [2].

Пристрої Інтернету речей (IoT) стають невід'ємною частиною повсякденного життя. Кількість пристроїв, у яких реалізовано цю технологію, зростає з кожним роком: від домашніх термостатів і освітлення до автомобілів і медичного обладнання. Однак проблема сумісності між різними пристроями, а особливо питання щодо безпеки зберігання та передачі даних також стають все більш актуальними [10].

Крім цього, зростання інтернету речей (IoT) створює нові можливості для кіберзлочинців. Пристрої IoT часто мають слабкі механізми захисту, що дає можливість зловмисникам використовувати їх як точки входу для атак на більш масштабні системи. Наприклад, великомасштабні бот-мережі, які використовуються для DDoS-атак, часто складаються з тисяч уразливих IoT-пристроїв.

Електронна пошта є важливим інструментом для особистого та професійного спілкування в усьому світі, що робить її привабливою мішенню для кіберзлочинців. Вона стала основним каналом для поширення шкідливих програм. У 2023 році 35% усіх кіберзагроз було доставлено саме через електронну пошту, а більше 94% організацій зафіксували випадки інцидентів, пов'язаних із безпекою електронних листів [3].

Не менш важливим аспектом є соціальна інженерія, яка використовує людський фактор як вразливість. Зловмисники створюють правдоподібні сценарії для обману користувачів і отримання доступу до їхніх даних. Це включає фішингові електронні листи, шахрайські дзвінки або підроблені веб-сайти.

Економічний вплив кіберзагроз також є вражаючим. За оцінками компанії McAfee, глобальні втрати від кіберзлочинності у 2023 році перевалили за 8

трильйонів доларів США. Ці витрати включають як прямі фінансові збитки, так і витрати на відновлення даних, репутаційні втрати та штрафи за порушення регуляцій щодо захисту даних [2].

Таким чином, зростання кіберзагроз є складним і багатогранним викликом, який вимагає комплексного підходу. Це включає розробку нових технологій захисту, навчання користувачів основам кібербезпеки, впровадження жорсткіших регуляцій та міжнародну співпрацю для протидії кіберзлочинності. У цьому контексті захист веб-додатків стає пріоритетним завданням, адже вони є критично важливими точками у сучасному цифровому ландшафті.

Наслідки кібератак мають далекосяжний і дорогий вплив. У 2024 році середня вартість порушення безпеки даних становить 4,88 мільйона доларів. У 2023 році атаки на бізнес-електронну пошту призвели до фінансових втрат понад 2,9 мільярда доларів. Ці тривожні статистичні дані наголошують на серйозних ризиках кібервразливості та акцентують увагу на необхідності посилення заходів безпеки в кваліфікованих фахівцях з кібербезпеки [2].

Найпоширенішою загрозою є саме фішинг. Фішинг — це використання текстових повідомлень, оманливих електронних листів, вебсайтів та інших форм комунікації з метою обману людей для завантаження шкідливого програмного забезпечення або розголошення конфіденційної інформації. Кіберзлочинці, видаючи себе за надійних осіб або авторитетні організації, намагаються викрасти важливі дані, такі як облікові дані, фінансова інформація та інші персональні дані [2].

Існує чотири основні види фішингу: Spear phishing, Whaling pishing, Vishing pishing та Email phishing. Кожнен з цих видів розрахований на певні аспекти:

– Spear phishing це вид фішингу спрямований на отримання конфіденційної інформації або доступу до комп’ютерних систем через персоналізовані повідомлення, надіслані електронною поштою, у текстових повідомленнях або через телефонні дзвінки. Зловмисники часто використовують дані із соціальних мереж, публічних баз або попередніх витоків інформації, щоб підвищити свою достовірність;

– Whaling pishing це метод орієнтований на високопоставлених співробітників, таких як директори або фінансові менеджери. Зловмисники створюють ретельно продумані та переконливі повідомлення, щоб викрасти конфіденційні дані організації;

– Vishing pishing це фішинг через голосові повідомлення. В даному випадку шахраї телефонують або залишають голосові повідомлення, видаючи себе за надійні джерела. Метою є отримання персональних даних, доступу до банківських рахунків або викрадення грошей;

– Email phishing це фішинг через електронну пошту. Цей вид атак спрямований на масову аудиторію. Зловмисники видають себе за легітимні організації через електронні листи, щоб викрасти конфіденційну інформацію [3].

Фішинг є одним із найпоширеніших та найефективніших видів кіберзлочинності. Близько 74% атак із захоплення облікових записів починаються саме з фішингу [4].

Найчастіше фішингові атаки спрямовані на такі компанії:

– Microsoft (57%);

– Apple (10%);

– LinkedIn (7%);

– Google (6%);

– Facebook (1,8%);

– Amazon (1,6%);

– DHL (0,9%);

– Adidas (0,8%);

– WhatsApp (0,8%);

– Instagram (0,7%) [2].

Основними цілями кіберзлочинців є компанії, які володіють великою часткою ринку користувачів електронної пошти, як-от Microsoft із сервісом Outlook та Google із Gmail.

Щоб впізнати та уникнути фішингових атак, то потрібно уважно перевіряти відправників повідомлень, уники сумнівних посилань і вкладень. Оманливі

пропозиції, які здаються занадто вигідними, відчуття терміновості або незвичні запити також є тривожними сигналами.

Захист від фішингу потребує постійної пильності, адже цей вид шахрайства щодня наражає на ризик мільйони людей у цілому світі.

Крім фішингу надзвичайно часто використовуються такі загрози як шкідливе програмне забезпечення. Атаки з використанням шкідливого програмного забезпечення зросли на 71% у період з 2016 по 2021 рік. Також у 2023 році кількість атак із використанням програм-вимагачів збільшилася на 74% у порівнянні з 2022 роком [3].

На цей момент близько 4,1 мільйона вебсайтів заражені шкідливим програмним забезпеченням. У середньому одна атака із застосуванням програм-вимагачів обходиться компанії в $4,91 мільйона. У 2023 році 7% атак програм-вимагачів призвели до фінансових втрат, а середній розмір викупу склав $10,000 [2].

Ці статистичні дані підкреслюють небезпеку кіберзлочинності та показують, наскільки серйозні наслідки можуть мати атаки з використанням шкідливого програмного забезпечення як для окремих організацій, так і для глобальної економіки.

Також атаки типу DDoS не поступаються за популярністю та ефективністю фішингу та шкідливому програмному забезпеченню. DDoS-атака виникає, коли зловмисники використовують велику кількість пристроїв для створення надмірного трафіку, спрямованого на систему, мережу або вебсайт. Це призводить до перевантаження цільового ресурсу, що унеможливлює його обробку легітимних запитів і робить його недоступним для користувачів.

У середньому компанія Microsoft щоденно нейтралізує близько 1,700 DDoS-атак. Варто зазначити, що лише у 2023 році з'явилося 20% усіх платформ, які пропонують DDoS-атаки "на замовлення" [3].

Серед відомих жертв таких атак — Amazon Web Services (AWS), GitHub та Dyn.

Зростаюча доступність інструментів та методів для проведення DDoS-атак сприяє їхній частоті й масштабу. Підвищення кількості платформ для оренди

DDoS-атак підкреслює актуальність цієї загрози для сучасного цифрового середовища.

Розуміючи серйозність загрози, багато організацій інвестують у стратегії та сервіси для захисту своїх мереж і ресурсів. Постійне вдосконалення технологій захисту стає критично важливим, адже навіть великі компанії, як-от AWS, GitHub та Dyn, зазнавали значних порушень через DDoS-атаки.

Крім вищезгаданих загроз та кібератак важливою загрозою є вітик персональних даних. У 2023 році витоки персональних даних торкнулися 349,221,481 людей. Споживча мережа Consumer Sentinel Network отримала понад 5,5 мільйонів звітів про різноманітні кіберзлочини, серед яких:

– Шахрайство – 2,606,042 звіти;

– Викрадення особистих даних – 1,036,955 звітів;

– Інші типи порушень – 1,905,717 звітів [2].

Через високий ступінь взаємопов’язаності цифрових систем і велику кількість персональної інформації, яка зберігається онлайн, кіберзлочини часто стають основною причиною крадіжок особистих даних користувачів. Зловмисники активно використовують фішингові електронні листи, шкідливе програмне забезпечення та витоки даних для отримання несанкціонованого доступу до конфіденційної інформації людей, наприклад такої як номери соціального страхування, облікові дані для входу та фінансова інформація.

Одним із найбільших витоків даних в історії був інцидент 2013 року, коли хакери зламали базу даних Yahoo, скомпрометувавши понад 3 мільярди облікових записів користувачів. У 2019 році Facebook зазнав подібного масштабного витоку, в результаті якого особиста інформація 533 мільйонів користувачів стала доступною у відкритому доступі [2]. На рисунку 1.1 зображена кількість репортів для найвідоміших крадіжок протягом 2023 року.

Most Common Types of Identity Theft

Rank	Theft Type	Number Of Reports In 2023[15]
1	Credit card fraud	416,636
2	Other identity theft	260,820
3	Loan or lease fraud	149,804
4	Bank account fraud	136,852
5	Government documents or benefits fraud	97,038

Рисунок 1.1 – Кількість репортів для найвідоміших крадіжок [2].

Крадіжка особистих даних може і відбувається у різних формах, зокрема в шахрайстві під час онлайн-шопінгу чи оформленні іпотеки. Найпоширенішими формами крадіжки особистої інформації є: шахрайство з кредитними картками, шахрайство з кредитами чи орендою, банківське шахрайство та шахрайство з урядовими документами чи пільгами.

У 2023 році шахрайство з кредитними картками було найпоширенішим видом, тоді як категорія «інші види крадіжки особистої інформації» посіла друге місце. Банківське шахрайство та шахрайство з кредитами чи орендою мали понад 130,000 звітів кожне, посівши третє і четверте місця відповідно. Шахрайство, пов’язане з урядовими документами чи пільгами, було п’ятим за поширеністю з понад 97,000 повідомлень у 2023 році [2].

Ця статистика підкреслює необхідність посилення захисту особистих даних та підвищення обізнаності щодо основних тактик зловмисників.

Водночас не менш важливим є розуміння загроз, з якими стикаються веб-сервери як ключові елементи інформаційної інфраструктури. Веб-сервери забезпечують доступ до веб-додатків і надають користувачам можливість взаємодіяти з різними сервісами, що робить їх унікальним незамінним компонентом сучасних цифрових екосистем. Однак їх популярність і відкритий доступ водночас перетворюють їх на привабливу мішень для кіберзлочинців.

Виявлення, аналіз і розуміння основних загроз для веб-серверів є критично важливими для забезпечення їхньої безпеки. Безпека веб-сервера стосується методів і заходів, які використовуються для захисту веб-сервера, його даних і

запущених на ньому програм від кіберзагроз, несанкціонованого доступу, витоків даних та інших вразливостей безпеки. Оскільки веб-сервери необхідні для розміщення веб-сайтів і веб-додатків, забезпечення їх безпеки має вирішальне значення для підтримки доступності, цілісності та конфіденційності даних, які вони обробляють.

Захищений веб-сервер (Secure Web Server – SWS) підтримує протоколи безпеки, наприклад як рівень захищених сокетів (SSL/TLS), де конфіденційна інформація, що передається з сервера та на сервер, шифрується для безпеки користувача. SWS може посилатися на веб-сервер, захищений від зовнішніх загроз і використовується лише невеликою групою працівників у локальній мережі. Серед основних загроз, з якими стикаються веб-сервери, можна виділити кілька ключових загроз [3].

Одна з таких загроз – це атаки відмови в обслуговуванні (DoS) та розподілені атаки відмови в обслуговуванні (DDoS). DoS і DDoS мають на меті перевантажувати сервери трафіком, поки вони не перестануть відповідати, що робить корпоративний веб-сайт або мережу недоступними.

Як приклад у 2020 році Amazon Web Services (AWS) зазнала однієї з найбільших за всю історію DDoS-атак із максимальною швидкістю 2,3 Тбіт/с. Зловмисники використовували техніку під назвою CLDAP Reflection, щоб перевантажити сервери, хоча AWS вдалося пом’якшити атаку без істотного впливу на клієнтів [2].

Ще однією серйозною вразливістю є впровадження мови структурованих запитів (SQL). SQL-ін’єкції надсилають структуровані запити мовою запитів через веб-форми для атаки на веб-додатки та веб-сайти шляхом створення, читання, оновлення, зміни або видалення будь-яких даних, що зберігаються на сервері.

Для прикладу у 2014 році американський роздрібний продавець Target зазнав значного порушення даних через уразливості SQL-ін’єкцій. Зловмисники вводили зловмисні команди SQL через уразливі веб-форми, дозволяючи їм отримати доступ до баз даних Target, де зберігалася конфіденційна інформація,

наприклад дані кредитної картки клієнтів. Це порушення призвело до розкриття 40 мільйонів рахунків кредитних і дебетових карток [2].

Запобігти або зменшити ризики SQL-ін’єкцій можна шляхом використання сучасних баз даних, таких як MongoDB, яка є нереляційною базою даних. MongoDB не використовує традиційні SQL-запити для маніпулювання даними, що значно знижує ймовірність виникнення вразливостей, пов'язаних з SQL-ін’єкціями. Замість цього MongoDB використовує власний формат запитів на основі JSON, що унеможливлює виконання шкідливих SQL-команд.

Ще одним важливим інструментом для захисту є використання Mongoose — модуля для Node.js, який дозволяє ефективно працювати з MongoDB. Mongoose забезпечує захист від SQL-ін’єкцій через вбудовану перевірку та фільтрацію вхідних даних, а також підтримує механізми безпечного виконання запитів і взаємодії з базою даних. Використання цих засобів дає можливість значно знизити вразливості і захистити веб-додатки від потенційних атак.

Не менш важливою є проблема невиправленого програмного забезпечення. Оновлення патчів для програмного забезпечення та виправлення безпеки призначені для усунення недоліків у старіших версіях програмного забезпечення. Однак, коли виходить незахищена версія, це залишає простір для кіберзлочинців для крадіжки даних. З цієї причини необхідно запровадити авторитетні служби керування виправленнями, щоб гарантувати, що захист завжди актуальний.

Як приклад витік даних Equifax у 2017 році, який розкрив особисті дані понад 147 мільйонів людей, був спричинений невиправленою вразливістю у структурі веб-програми Apache Struts. Компанії не вдалося застосувати патч безпеки, який був доступний протягом двох місяців, що дозволило зловмисникам скористатися недоліком і отримати доступ до особистої інформації, навіть включаючи номери соціального страхування [2].

Ще однією загрозою є експлойти нульового дня. Експлойт нульового дня — це тип кібератаки, який усуває вразливості програмного забезпечення, про які постачальники програмного забезпечення чи антивірусів не знають. Хакер виявляє вразливість програмного забезпечення перед тим, як розробники

виправлять її або антивірусні компанії створять засоби захисту, і активно використовує цю вразливість для атаки.

Стосовно прикладу, то у 2021 році Microsoft Exchange Server постраждав від серії експлойтів нульового дня, відомих як ProxyLogon. Зловмисники використовували раніше невідомі вразливості, щоб отримати дані доступу до облікових записів електронної пошти, встановити бекдори та пересуватися по мережах. Тисячі організацій по всьому світу були скомпрометовані перед тим, як Microsoft випустила патчі для уразливостей [2].

Веб-сервери також схильні до атак міжсайтового сценарію (XSS). XSS — це метод, більше подібний до SQL-ін'єкцій, оскільки програмне забезпечення імплантується в сценарії на стороні сервера, щоб хакер міг збирати конфіденційні дані або виконувати сценарії.

Людська недбалість (наприклад, поганий код, паролі, які можна легко вгадати, чи невстановлення брандмауерів та інших рішень кібербезпеки) спричиняє більшість ризиків для безпеки сервера.

Наприклад у 2019 році компанія Capital One зазнала масштабного витоку даних, у результаті якого були розкриті дані понад 100 мільйонів клієнтів. Зловмисник використав неправильно налаштований брандмауер в інфраструктурі Amazon Web Services (AWS) Capital One. Це порушення значною мірою пояснюється людською помилкою під час належного налаштування брандмауера, що дозволило несанкціонований доступ до конфіденційної інформації [2].

Також слабка автентифікація та авторизація є серйозною проблемою. Неналежна безпека процесів входу, наприклад слабкі паролі або неправильно налаштовані засоби контролю доступу, здатні дозволити зловмисникам отримати неавторизований доступ до сервера.

API, які не застосовують належні механізми автентифікації чи авторизації, здатні дозволити неавторизованим користувачам отримати можливий доступ до конфіденційної інформації або виконувати певні дії, які їм не можна дозволяти. Наприклад, якщо кінцева точка API загальнодоступна, не вимагаючи дійсного токена чи облікових даних, зловмисники можуть скористатися нею.

До прикладу у 2016 році Міністерство юстиції США було зламано і хакери отримали конфіденційні дані, причиною стали слабкі заходи автентифікації. Група хакерів отримала доступ до облікових записів електронної пошти співробітників департаменту, включно з генпрокурором, підгадавши ненадійні паролі. Вони розкрили особисті дані тисяч державних службовців [2].

Успішні результати тих чи інших атак, які націлені на ті чи інші компанії приносять тим компаніям великі фінансові втрати. Загальна вартість збитків від кіберзлочинів, за прогнозами, досягне $10,5 трлн до 2025 року . США посідають перше місце у світі за середньою вартістю витоків даних, яка становить $9,36 млн за інцидент. У галузі охорони здоров'я ціна витоку даних ще вища — $9,77 млн, що перевищує середній показник у фінансовому секторі майже на $3,7 млн [2].

Кіберзлочини не лише ставлять під загрозу конфіденційність чутливої інформації та безпеку користувачів і клієнтів, а й мають значні фінансові наслідки. Вартість таких атак складається з витрат на відновлення систем, юридичних витрат, штрафів регуляторів, крадіжки інтелектуальної власності, переривання операційної діяльності та втрати репутації.

За даними звіту ФБР за 2023 рік, Каліфорнія стала штатом із найбільшими втратами від кіберзлочинів серед жертв — понад 77 тисяч зареєстрованих випадків і збитки, що перевищують $2 млрд (див. рисунок 1.2).

10 Worst States for Cybercrime by Victim Loss in 2023

Rank	State	Victim Loss In Millions[5]	Number Of Victims[5]
1	California	$2,159.5	77,271
2	Texas	$1,021.6	47,305
3	Florida	$874.7	41,061
4	New York	$750.0	26,948
5	New Jersey	$441.2	12,253
6	Pennsylvania	$360.3	16,407
7	Illinois	$335.8	15,783
8	Arizona	$324.4	16,584
9	Georgia	$301.0	13,917
10	Washington	$288.7	14,600

Рисунок 1.2 – Топ штатів США за втратою коштів через кіберзлочинність [2].

Каліфорнія, Техас, Флорида, Нью-Йорк і Нью-Джерсі увійшли до п'ятірки штатів із найбільшими втратами. На їхню частку припало значно більше половини загальних збитків у США. У свою чергу, такі штати, як Пенсильванія, Іллінойс, Аризона, Джорджія та Вашингтон, також зазнали серйозних втрат, сукупно перевищуючи $1,6 млрд [2].

Ця тенденція підкреслює важливість посилення кібербезпеки як для окремих осіб, так і для організацій, які змушені боротися з постійно зростаючими загрозами в умовах цифрової епохи.

Наразі технічне навчання є основною тенденцією у сфері освіти з кібербезпеки. Часто навчальні програми університетів не включають вимоги до циклу розробки, професійні стандарти та регламенти. Водночас сучасний спеціаліст з кібербезпеки потребує гнучкості, а технічні навички відносно легко освоїти за допомогою доступних IT-інструментів [7].

У сучасному світі, де цифрова трансформація відбувається неймовірно швидкими темпами, а віртуальний простір стає невід'ємною частиною соціального життя, питання кібербезпеки набуває безпрецедентного значення. Україна, як і велика кількість решти інших країн світу, стикається з серйозними

викликами та загрозами в цій сфері, і досягнення належного рівня кібербезпеки стає питанням життєвої важливості [8].

В Україні посилення кібребезпеки почалося ще до 2014 року, але значний акцент на це питання було зроблено після подій Революції Гідності та початку російської агресії. Після анексії Криму та початку війни на сході України, кібербезпека стала критичною складовою національної безпеки країни.

Убезпечення безпеки інформації у веб-додатках, зокрема серверної частини, регламентується низкою законодавчих актів України, що стосуються кібербезпеки та захисту цінних даних. Одним із основних документів у цій сфері є ЗУ "Про основні засади забезпечення кібербезпеки України", який прийняли в 2017 році [23]. Даний закон визначає правові та організаційні основні забезпечення кібербезпеки в Україні, спрямовані на захист інформаційної інфраструктури держави, у тому числі серверів, баз даних та інших об’єктів, що є надзвичайно важливими для забезпечення функціонування суспільства і держави.

Згідно з цим законом, до ключових вимог належать обов’язкові заходи стосовно захисту інформаційних систем від кібератак, моніторинг інцидентів безпеки та впровадження сучасних механізмів виявлення й усунення вразливостей. У контексті веб-додатків, серверна частина має відповідати стандартам кіберзахисту для забезпечення конфіденційності, цілісності та доступності оброблюваних даних.

Також важливим документом є ЗУ "Про захист персональних даних", який встановлює правила щодо збору, обробки, зберігання та передачі персональних даних [24]. Цей закон зобов’язує розробників інформаційних систем, і також веб-додатків, впроваджувати технічні та організаційні заходи для того щоб забезпечити безпеку особистих даних, особливо захисту стосовно несанкціонованого доступу, втрати, розголошення чи модифікації. Серверна частина веб-додатків, яка обробляє такі дані, повинна використовувати надійні механізми аутентифікації, шифрування даних та контроль доступу.

Таким чином, закони України встановлюють чіткі вимоги до розробників і власників інформаційних систем щодо забезпечення їхньої безпеки. Вони

слугують правовою основою для впровадження ефективних механізмів захисту, які знижують ризики кібератак і втрати даних, що є важливим аспектом у контексті дослідження методів захисту серверної частини веб-додатків на платформі Node.js.

РОЗДІЛ 2 ОСНОВНІ ПІДХОДИ ДЛЯ ЗАХИСТУ ВЕБ-СЕРВЕРІВ

2.1 Методологія STRIDE її переваги та обмеження

Моделювання загроз — це структурований процес для розуміння безпеки системи, який підтримується маніфестом для підвищення безпеки та конфіденційності під час розробки. Моделювання загроз сприяє співпраці між різними зацікавленими сторонами, об’єднуючи знання з різних областей для комплексної оцінки та усунення потенційних ризиків безпеці [6]. Цей спільний підхід не тільки покращує якість процесу моделювання загроз, але й служить потужним освітнім інструментом. Беручи активну участь у вправах з моделювання загроз, члени команд різних відділів отримують цінну інформацію про концепції кібербезпеки та найкращі практики [15]. Ця взаємодія допомагає культивувати усвідомлення безпеки в усій організації, перетворюючи кібербезпеку з відокремленої відповідальності на спільну компетенцію. Оскільки члени команди стають більш обізнаними про потенційні загрози та їхні наслідки, вони краще підготовлені для включення питань безпеки у свою повсякденну роботу, що призведе до більш надійної та стійкої системи безпеки в усій організації.

Моделювання загроз є важливим інструментом для виявлення та зменшення ризиків безпеці, зокрема у складних і взаємопов'язаних системах. Одним із таких прикладів є промислові системи управління (ICS) — це спеціальні системи, які поєднують інформаційні технології (IT) і операційні технології (OT). Завдяки їх взаємозв’язку та віддаленій доступності вони стають об’єктом кібератак. Через їх складність і неоднорідність з точки зору пристроїв і протоколів зв’язку необхідно розробити спеціальні засоби контролю безпеки та методи аналізу ризиків. Зокрема, для того, щоб зменшити зусилля, пов’язані з розгортанням аналізу ризиків у таких складних системах, необхідно забезпечити автоматизовані методи. Для подібного використовується добре відомий інструмент моделювання STRIDE, а саме Microsoft Threat Modeling Tool (MTMT), додатковим шаблоном, призначеним для ICS [16]. Для прикладу

прогнозується, що до 2035 року майже всі нові автомобілі будуть мати підключення до інтернету. Інтеграція може надати багато переваг, таких як доступ до різної інформації, оскільки транспортний засіб завжди підключений до інтернету. Однак проблема полягає в тому, що система стає вразливою до кіберзагроз з боку зацікавлених в цьому сторін. Саме для тестування на безпеку варто використовувати інструмент MTMT або подібні [17].

Наприклад у Сполучених Штатах під критичною інфраструктурою (КІ) розуміють системи та активи, які є життєво важливими для країни, і їхнє пошкодження або знищення може серйозно вплинути на безпеку, економіку, охорону здоров'я чи інші важливі сфери. Історично руйнування КІ зосереджувалося на загрозах у фізичній сфері, таких як стихійні лиха та кінетичні атаки. Однак тепер у кібердомені може виникнути збій КІ – проблема, яка ускладнюється інтеграцією мереж інформаційних технологій (IT) і операційних технологій (OT). Саме для такого і використовують різні методології включно із STRIDE [19].

STRIDE означає Spoofing, Tampering, Repudiation, Information disclosure, Denial of service and Elevation of privilege, розроблений Лореном Конфельдером і Прерітом Гаргом у 1999 році для виявлення потенційних вразливостей і загроз продуктам компанії [5]. Методологія Microsoft STRIDE має на меті гарантувати, що програма відповідає вимогам безпеки конфіденційності, цілісності та доступності (CIA), окрім авторизації, автентифікації та неспростовності. STRIDE з часом еволюціонував, щоб включити нові таблиці для конкретних загроз і варіанти STRIDE-per-Element і STRIDE-per-Interaction [18].

Модель STRIDE надає організаціям ефективний інструмент для аналізу систем і мереж, оскільки дозволяє класифікувати потенційні загрози, враховуючи їх ймовірність і можливий масштаб впливу. Це допомагає правильно визначити пріоритети у питаннях безпеки, фокусуючи увагу на найбільш критичних ризиках.

Наприклад, організація охорони здоров’я, яка використовує методологію STRIDE, може розглядати конфіденційність інформації про пацієнтів як головний пріоритет безпеки, оскільки є ризик розголошення конфіденційних

даних або несанкціонованого доступу. Оцінюючи ці загрози як потенційно серйозні, організація може розробити та впровадити контрзаходи, ефективні для їх запобігання [5]. Ця методологія особливо корисна для компаній, які чітко розуміють загрози та вразливі місця, з якими вони стикаються.

Це найстаріша методологія моделювання загроз, яка допомагає визначити методи пом'якшення. Методологія відносно проста у використанні, але може зайняти багато часу. Перегляньте таблицю 1.1 для прикладу використання [5].

Таблиця 1.1 – Перелік основних загроз моделі STRIDE

Загроза	Власність	Визначення
Підміна особистості	Аутентифікація	Видавати себе за щось чи когось іншого
Фальсифікація	Цілісність	Зміна даних або коду
Відмова	Безвідмовність	Стверджуючи, що не виконав дію
Розкриття інформації	Конфіденційність	Розкриття інформації особі, яка не має права на її перегляд
Відмова в обслуговуванні (DoS)	Доступність	Відмовляти або погіршувати обслуговування користувачів
Зловживання привілеями	Авторизація	Отримайте можливості без належного дозволу

Методологія STRIDE має кілька переваг в своєму використанні. В неї хороша гнучкість, STRIDE — це адаптована структура, яку можна застосовувати в багатьох галузях і сферах. Вона не обмежується конкретними випадками використання або секторами, що робить її універсальним інструментом. Також за її методологією можна визначати потенційні ризики. Використання STRIDE дозволяє ефективно виявляти можливі загрози. Методологія Shostack Four Question Framework, STRIDE допоможе детальніше визначити потенційні вектори атак або методи загроз. STRIDE допомагає командам оцінювати і визначати пріоритети щодо ризиків. Це дозволяє зосередитися на найбільш критичних питаннях і ефективно управляти безпекою. STRIDE дозволяє

зменшити кількість порушень, якщо методологію застосовувати правильно, то вона може значно знизити ймовірність безпекових порушень та уразливостей, що може позитивно вплинути на безпеку організації в цілому [5].

Крім переваг методологія STRIDE має ще деякі недоліки. Витратність ресурсів є одним з мінусів методу STRIDE, оскільки проведення ретельного аналізу може зайняти значний час, особливо для складних систем. Крім того, STRIDE забезпечує переважно статичний аналіз, зосереджуючись на оцінці дизайну та архітектури системи, що може призводити стосовно того, що певні або ж деякі динамічні або поточні загрози не будуть враховані.

Ще однією проблемою є суб'єктивність результатів, що може бути як перевагою, так і недоліком. Завдяки адаптивності STRIDE оцінка ймовірності та впливу загроз може варіюватися залежно від конкретної команди або окремої особи, що проводить аналіз, і це може призвести до різних результатів [20].

Також, використання STRIDE вимагає постійного обслуговування, оскільки метод не є одноразовим. Для того, щоб врахувати зміни в дизайні системи, архітектурі або нові загрози, також аналіз слід проводити регулярно.

Нарешті, варто зазначити, що STRIDE був розроблений переважно для моделювання загроз у розробці програмного забезпечення і тому може бути менш застосовним у інших галузях чи сферах, де його методи можуть не мати такої ж ефективності.

2.2 Основні вразливості методології STRIDE

STRIDE був розроблений наприкінці 1990-х років двома інженерами Microsoft, Кореном Конфельдером і Прерітом Гаргом. У своєму листі під назвою «The Threats To Our Products» вони розглянули нові загрози безпеці систем, спричинені прогресивними технологіями, і визначили, що потрібен спосіб визначення місця потенційних загроз. Модель загроз STRIDE враховує шість різних категорій загроз [5].

До першої категорії загроз відводиться Spoofing identity. Spoofing identity — це метод, коли хакер видає себе за іншу особу, використовуючи її особисті дані

чи іншу інформацію, щоб здійснити шахрайство. Одним із найпоширеніших прикладів цього виду загрози є надсилання електронного листа з фальшивої адреси, яка виглядає як адреса справжньої особи (цей метод часто використовується в фішингових атаках). У таких листах зазвичай запитують конфіденційні дані, такі як паролі, номери кредитних карток чи іншу особисту інформацію. Вразливий або необізнаний одержувач може надати ці дані, і хакер, отримавши їх, може використати для того, щоб видавати себе за жертву або здійснити подальші шахрайські дії.

Підроблені особи можуть бути як людськими, так і технічними. Завдяки спуфінгу (підміна особи) хакери можуть отримати можливий доступ до системи через одну вразливу точку, що дозволяє їм здійснити більш масштабну кібератаку [6]. Швидкий розвиток штучного інтелекту (ШІ) значною мірою покращив ефективність фішингових атак, які тепер виглядають переконливіше, ніж будь-коли раніше. Є кілька способів, як ШІ може бути використаний у фішингових атаках. Наприклад надсилання електронних листів або повідомлень, які намагаються змусити користувачів натискати на шкідливі посилання чи вкладення. Використання соціальних мереж для створення опозиційних тролів, що мають на меті зруйнувати репутацію брендів або осіб. Або ж створення фейкових новинних сайтів або підроблених сторінок у соціальних мережах, щоб вводити користувачів в оману.

Стосовно другої категорії загроз, то це Tampering with data (Підробка даних). Підробка даних відбувається, коли інформація змінюється або маніпулюється без дозволу, що може призвести до змін у роботі системи або компрометації її безпеки. Зловмисники можуть здійснювати втручання різними способами. Наприклад зміна файлів конфігурації для отримання контролю над системою. Чи вставка шкідливих файлів або програм, що можуть привести до порушення роботи системи. Або ж видалення або модифікація файлів журналів для приховування своїх дій або змін.

Моніторинг змін, або моніторинг цілісності файлів (FIM), є важливим інструментом для виявлення змін у даних і визначення моменту, коли відбувається підробка чи несанкціоноване редагування файлів. Цей процес

полягає у порівнянні поточного стану файлів з базовою лінією, яка визначає, як виглядає «нормальний» або «безпечний» файл. Важливим аспектом є належне ведення журналів та їх зберігання, оскільки вони необхідні для коректного моніторингу змін. Це дозволяє відслідковувати всі події і забезпечувати належний рівень безпеки системи. Рекомендується ознайомитися з посібником із безпеки, щоб зрозуміти, як недостатнє або надмірне журналювання може вплинути на ефективність моніторингу та забезпечення належного аудиту.

На рисунку 2.1 показано приклад дерева атаки для моделювання втручань у роботу конкретної 3D-системи друку. Це ще один приклад діяльності з моделювання загроз. Зображення надано з публікації "Threat Modeling in Construction: An example of a 3D Concrete Printing System" [5].

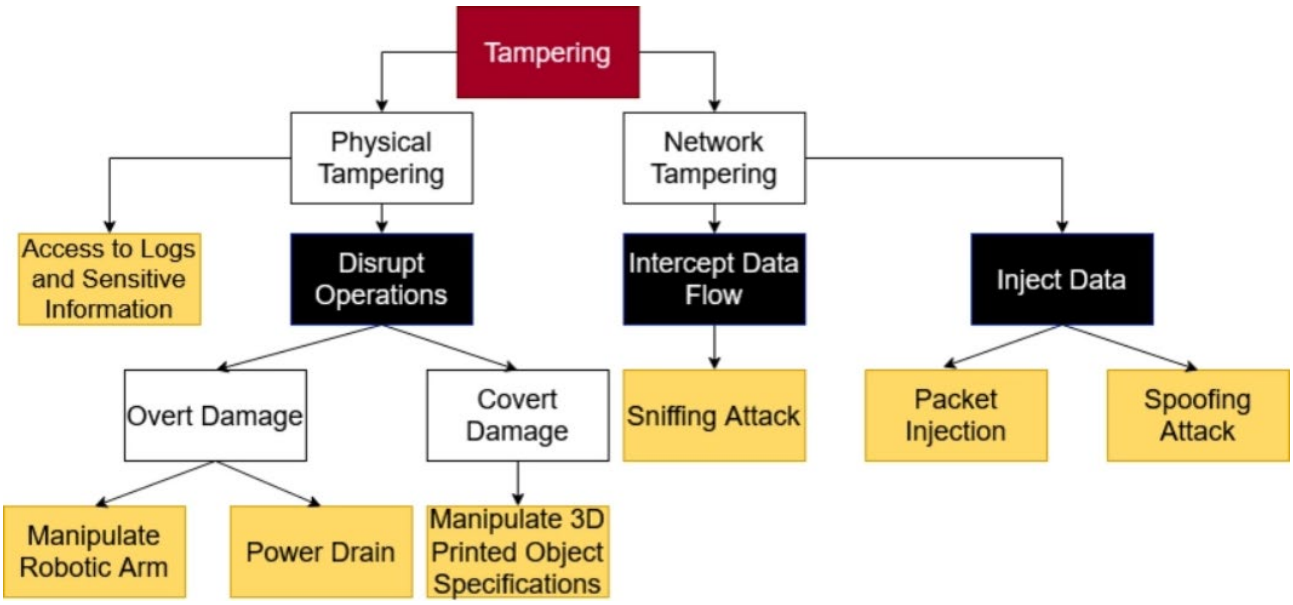

Рисунок 2.1 – Приклад дерева атаки

В третій категорії загроз знаходиться Repudiation (Відмова). Атаки відмови від причетності (denial of origin) виникають, коли зловмисник здійснює незаконну або зловмисну операцію в системі і потім заперечує свою причетність до цієї атаки. В результаті таких атак система не може чітко ідентифікувати або відстежити джерело загрози, що ускладнює виявлення хакера.

Особливо ці атаки є поширеними в системах електронної пошти, оскільки багато з них не перевіряють наявність аутентифікації вихідних повідомлень. Це створює можливості для хакерів приховувати свою особу. Більшість атак

відмови від причетності починаються з атак на доступ до системи, де зловмисник отримує контроль над акаунтами або даними.

Стосовно четвертої категорії загроз, то до неї відноситься Information disclosure. Information disclosure (витік інформації) — це ситуація, коли програма або веб-сайт ненавмисно розкриває дані неавторизованим користувачам. Така загроза може вплинути на різні аспекти роботи програми, включаючи обробку, потік і зберігання даних.

До прикладів розкриття інформації можна віднести ненавмисний доступ до файлів вихідного коду через тимчасові резервні копії, ненавмисне розкриття конфіденційної інформації, такої як номери кредитних карток, або розкриття даних бази даних через повідомлення про помилки.

Ці проблеми є досить поширеними і можуть виникати через невірно налаштовані або незахищені конфігурації програм, а також через помилкові або недостатньо обережні відповіді на помилки в дизайні програм.

Наступною, п’ятою категорією загрози є Denial of service (Dos). Атаки на відмову в обслуговуванні (DoS) обмежують доступ авторизованих користувачів до ресурсів, до яких вони повинні мати доступ. Це може впливати на процеси, потоки даних і зберігання інформації в системах. Зокрема, в результаті DoS-атак сервери або інші критичні компоненти можуть стати недоступними для користувачів.

За останні роки DoS-атаки стали масовішими й частішими. За оцінками, у 2020 році було зафіксовано 12,5 мільйона інструментів для проведення DDoS-атак. У “звіті про стан тестування на проникнення за 2022 рік” повідомляється, що кількість атак типу DoS збільшилася на 133% порівняно з попереднім роком [2].

Одна з найбільш відомих атак сталася на Google в 2017 році. У своїй заяві компанія зазначила: «Зловмисник використав кілька мереж щоб підробити 167 Mpps (мільйонів пакетів на секунду) до 180 000 відкритих серверів CLDAP, DNS і SMTP, що спричиняло велику кількість відповідей, які надходили до нас. Це демонструє масштаби, яких може досягти добре оснащений зловмисник: ця атака

була вчетверо більшою за рекордну атаку в 623 Гбіт/с, здійснену ботнетом Mirai роком раніше.» [2].

Попри зростання кількості DoS-атак, захисні інструменти, як AWS Shield і CloudFlare, продовжують залишатися ефективними.

Останньою, шостою катерогією загроз є Elevation of privileges (Підвищення привілеїв). Завдяки підвищенню привілеїв авторизований або навіть неавторизований користувач може отримати доступ до інформації, до якої не має права доступу. Така атака може проявлятися, наприклад, у вигляді пропущеної перевірки авторизації або підвищення привілеїв через маніпуляцію даними. В останньому випадку зловмисник може змінювати вміст диска або пам'яті, щоб виконати несанкціоновані команди або отримати доступ до захищених ресурсів.

2.3 STRIDE для покращення безпеки сервера на платформі Node.js

Методологія STRIDE була обрана для оцінки безпеки серверної частини на платформі Node.js через її комплексний і структурований підхід до аналізу загроз. STRIDE, розроблена компанією Microsoft, дозволяє ефективно ідентифікувати та класифікувати потенційні загрози на основі їхнього впливу на систему. Цей підхід особливо важливий для веб-додатків, які працюють у швидко змінюваному та високодинамічному середовищі.

Node.js — популярна платформа для розробки сучасних веб-додатків, відома своєю продуктивністю, масштабованістю та активним використанням в реальному часі [11]. Node.js переносить мову JavaScript на сторону сервера, дозволяючи розробникам створювати масштабовані, високопродуктивні та керовані подіями програми [13]. Однак, її відкритий код, популярність серед розробників і взаємодія з клієнтськими системами створюють додаткові ризики безпеки. STRIDE дозволяє врахувати специфіку Node.js і системно оцінити її вразливості.

Крім того досить таки важливими аспектами для Node.js є Патерни проектування. Патерни проектування — це перевірені рішення для поширених проблем, з якими розробники програмного забезпечення стикаються під час

свого шляху кодування. Вони надають структурований підхід до вирішення задач і сприяють впровадженню кращих практик в архітектурі програмного забезпечення. Використовуючи шаблони проектування, розробники можуть створювати більш надійні, масштабовані та прості для підтримки кодові бази [14].

Ключовою перевагою методології STRIDE є її здатність класифікувати загрози в шість основних категорій, що дає можливість комплексно підійти до вирішення певних проблем в безпеці.

Підроблення (Spoofing). Ця загроза особливо актуальна для систем аутентифікації на Node.js, де зловмисники можуть підробити запити або користувачів для отримання несанкціонованого доступу. Для захисту від даної загрози використовуються механізми автентифікації та верифікації, такі як двофакторна аутентифікація (2FA) і перевірка токенів. Цей тип загрози стосується можливості підробки особи користувача або сервісу. Методологія STRIDE допомогла виявити потенційні проблеми в аутентифікації. Зокрема, це використання JWT токенів (access та refresh) для аутентифікації, що забезпечує унікальність та перевірку особи користувача. Та аудит-лог з записами IP-адрес та часових міток для моніторингу дій користувачів і знаходження підозрілої активності.

Підробка даних (Tampering). Захист даних під час передачі і зберігання є критичним для Node.js-додатків, оскільки їхня архітектура часто передбачає обробку великих обсягів даних через мережу. Стратегічно важливо використовувати шифрування (SSL/TLS) і перевірку цілісності даних для безпеки від несанкціонованих змін. Цей тип загрози охоплює ризики несанкціонованої зміни даних під час їх зберігання чи передачі. Для запобігання цій загрозі в сервері реалізовано механізм підпису даних (HMAC) для перевірки їх цілісності, та обмежено доступ до операцій з модифікації даних, дозволяючи редагувати або видаляти пости лише авторам або адміністраторам.

Відмова від відповідальності (Repudiation). Ця загроза означає можливість для користувачів або систем відмовитись від своїх дій, що ускладнює аудит і забезпечення відповідальності. Для протидії цьому в Node.js застосовуються

механізми логування всіх дій користувачів і перевірки доступу до критичних ресурсів, що дозволяє відстежувати та підтверджувати дії в системі. Цей тип загрози стосується ситуацій, коли користувач може заперечувати виконання певних дій. Щоб цього уникнути, то впроваджено аудит-лог, що фіксує всі дії користувачів з точними мітками часу і автором дії, що забезпечує прозорість та підтвердження авторства.

Розголошення інформації (Information Disclosure). Уразливості, які дозволяють витік конфіденційної інформації, є одними з найбільш серйозних для будь-якої платформи. Враховуючи, що багато додатків на Node.js обробляють персональні або фінансові дані, STRIDE допомагає виявити можливі шляхи витоку через неналежне зберігання чи обробку даних. Для захисту використовуються техніки шифрування, обмеження доступу і аудит доступу до конфіденційних даних. Загроза розголошення конфіденційної інформації може виникнути через некоректні налаштування сервера або вразливості в обробці запитів. Для зменшення таких ризиків було налаштовано захист від XSS-атак, щоб запобігти впровадженню шкідливого коду в поля вводу (наприклад, title чи body). І використано helmet, що додає додаткові заголовки безпеки до HTTP-відповідей, обмежуючи доступ до потенційно небезпечних даних.

Відмова в обслуговуванні (Denial of Service). Атаки типу DoS/DDoS можуть призвести до значних втрат у доступності сервісів. STRIDE дозволяє оцінити ризики таких атак, зокрема через надмірне навантаження на сервери або уразливості в механізмах обробки запитів. Для мінімізації таких загроз на платформі Node.js використовуються різні стратегії, зокрема лімітування запитів і використання кешування. Загроза перевантаження сервера величезною кількістю від можливих запитів може бути запобігнута завдяки таким заходам, як реалізацією rate limiting для обмеження частоти запитів, зокрема до маршруту логіну. І оптимізовано роботу з базою даних шляхом кешування запитів для постів, що знижує навантаження на сервер.

Покращення привілеїв (Elevation of Privilege). Ця загроза виникає, коли зловмисник отримує доступ до привілеїв, які зазвичай не належать йому. В Node.js це може статися через помилки в коді або неправильне налаштування

прав доступу до серверних ресурсів. Для попередження таких атак застосовуються принципи найменших привілеїв і ретельний контроль доступу. Цей тип загрози пов'язаний з можливістю підвищення привілеїв, коли звичайний користувач може отримати можливості стосовно адміністративного функціоналу. Для запобігання цього ризику впроваджено чіткий розподіл прав доступу між адміністраторами та звичайними користувачами, і також контролери перевіряють ролі користувачів, дозволяючи виконувати адміністративні операції лише тим, хто має відповідну роль.

Методологія STRIDE не лише дозволяє виявити вразливості, але й допомагає впровадити конкретні заходи для їхнього усунення. Це включає використання шифрування, двофакторної аутентифікації, обмеження прав доступу, захист від атак DoS/DDoS та впровадження логування для забезпечення прозорості дій користувачів.

Завдяки використанню методології STRIDE, вдалося не лише ідентифікувати вразливості серверної частини на платформі Node.js, а й оцінити потенційні наслідки атак. Це забезпечує більш обґрунтовані рішення для вибору методів захисту, що гарантують високий рівень безпеки та відповідність сучасним вимогам захисту інформації.

2.4 Механізми та принципи захисту інформаційних систем

Аутентифікація — це процес перевірки особи користувача, що намагається отримати доступ до системи. Метою аутентифікації є підтвердити, що користувач є тим, за кого себе видає. Найпоширеніший спосіб аутентифікації — введення логіна та пароля. Для посилення безпеки часто використовують багатофакторну аутентифікацію (2FA), яка вимагає додаткового кроку підтвердження, наприклад, через мобільний код, додатки для генерації кодів або електронну пошту.

Авторизація — це процес визначення, які ресурси або функції доступні конкретному користувачеві після того, як його особа була підтверджена. Авторизація визначає, що користувач може або не може робити в системі, на

основі прав, ролей чи атрибутів. Зазвичай застосовуються такі підходи, як рольова модель доступу (RBAC), де кожному користувачу надається певна роль з відповідними правами. Та модель на основі атрибутів (ABAC), яка враховує додаткові фактори, такі як час, місце чи інші характеристики.

Аутентифікація та авторизація працюють разом: перша підтверджує особу користувача, а друга — визначає доступ до конкретних ресурсів. Наприклад, у веб-додатку аутентифікація може здійснюватися через пароль, а авторизація визначає, чи має користувач право редагувати свій профіль або доступати обмежені ресурси.

Забезпечення безпеки за допомогою ефективної аутентифікації та авторизації допомагає мінімізувати ризики несанкціонованого доступу та зловживань у системі.

Для забезпечення безпеки даних, які передаються між клієнтом і сервером, використовують шифрування, шифрування дає захсит від перехоплення чи змін. Основним методом є TLS (Transport Layer Security), який убезпечує захищене з’єднання через HTTPS. Це шифрує всю інформацію, що передається, запобігаючи атакам типу "man-in-the-middle" (MITM). TLS використовує симетричне і асиметричне шифрування для забезпечення конфіденційності та цілісності.

Хешування — це процес перетворення даних (наприклад, паролів, повідомлень) у фіксовану довжину строку (хеш), яка виглядає випадковою. Хеш-функція одностороння, тобто неможливо відновити оригінальні дані з хешу. Хешування зазвичай застосовується для збереження паролів і перевірки цілісності даних.

Ключові характеристики хешування: паролі, функції хешування та перевірка цілісності. Замість того, щоб зберігати паролі у відкритому вигляді, сервер зберігає їх хеш. При кожному введенні пароля він знову хешується і порівнюється з збереженим хешем. В хешуванні для паролів зазвичай використовуються bcrypt, scrypt або Argon2, які спеціально розроблені для захисту паролів і містять механізми уповільнення процесу хешування, що робить атаку методом підбору пароля більш складною. Хешування також

використовують для перевірки цілісності, чи не було змінено файли чи дані. Наприклад, для завантаження файлів користувач може перевірити хеш файлу з його оригінальним хешем, щоб переконатися, що файл не був змінений.

Шифрування даних на сервері захищає інформацію, що зберігається на сервері. Використовуються симетричні алгоритми, такі як AES, і асиметричні (наприклад, RSA) для захисту чутливих даних. Шифрування забезпечує, що навіть при компрометації серверу, дані залишаються недоступними без ключа для дешифрування. Важливо правильно управляти ключами шифрування, використовуючи інструменти типу AWS KMS чи HashiCorp Vault.

Однією з найбільш поширених атак є SQL-ін’єкція, коли зловмисник вставляє шкідливий SQL-код у запит до БД. Це призводить до витоку особистої інформації або її модифікації. Щоб захиститися від SQL-ін’єкцій важливо використовувати підготовлені запити, які автоматично екранують входи користувачів, а також застосовувати об'єктно-реляційні мапери (ORM), що додатково захищають від таких вразливостей.

Ще однією популярною атакою є кросс-сайтові скрипти (XSS). У цьому випадку зловмисник вставляє шкідливий JavaScript-код у веб-сторінку, що може викрасти сесії користувачів або виконати небажані дії від їхнього імені. Захист від XSS включає в себе екранування вихідних даних та використання політик безпеки контенту (CSP), які обмежують, які скрипти можуть бути виконані на веб-сторінці.

Крім вищезгаданих атак є атаки типу відмова в обслуговуванні (DoS/DDoS), котрі спрямовані на те, щоб перевантажити сервер великою кількістю запитів, зробивши його недоступним для законних користувачів. Для запобігання таким атакам використовують різні методи, зокрема розподілені мережі доставки контенту (CDN), що дозволяють розподіляти навантаження, а також обмеження швидкості запитів (rate limiting) та спеціалізовані анти-DDoS сервіси, які можуть блокувати підозрілі запити.

На додачу до DoS атак, HTTP-запити приймаються HTTP-сервером Node.js і передаються коду програми через зареєстрований обробник запитів. Сервер не аналізує вміст тіла запиту. Таким чином, будь-який DoS, викликаний вмістом

тіла після того, як його було передано обробнику запитів, не є вразливістю в самому Node.js, оскільки за правильну обробку цього відповідає код програми [11].

Міжсайтова підробка запитів (CSRF) — це атака, в якій хакер примушує користувача виконати несанкціоновані дії на веб-сайті, де він вже автентифікований. Щоб захиститися від CSRF використовуються CSRF-токени, що додаються до форм і запитів, а також перевірка реферера, щоб переконатися, що запит надійшов з легітимного джерела.

Перехоплення сесії (Session Hijacking) є ще одним серйозним ризиком. Атакувальник може перехопити сесію користувача та одержати доступ до його акаунту. Для захисту від подібних атак необхідно використовувати HTTPS для шифрування трафіку, а також налаштувати додаткові параметри безпеки для cookies, наприклад, прапорці HttpOnly та Secure, щоб запобігти доступу до них через JavaScript.

Ще один важливий аспект захисту — це мінімізація прав доступу. Користувачам повинні надаватися лише ті права, які необхідні для виконання їхніх завдань, згідно з принципом найменших привілеїв. Це зменшує ризики, якщо облікові дані користувача потрапляють саме до зловмисника.

Захист даних включає також шифрування чутливої інформації, як в момент її передачі через мережу (наприклад, за допомогою TLS/SSL), так і при її зберіганні на сервері. Шифрування допомагає зберегти конфіденційність даних навіть у разі компрометації серверу.

Невід’ємною частиною забезпечення безпеки є також аудит і моніторинг системи. Постійний моніторинг дозволяє виявити підозрілу активність або атаки в реальному часі, що дає змогу оперативно реагувати на інциденти.

Моніторинг і логування є невід'ємними елементами системи безпеки, оскільки дозволяють виявляти підозрілі дії та аномалії в реальному часі. Логування передбачає запис усіх значущих подій, що відбуваються в системі, таких як входи користувачів, зміни в базах даних, помилки та спроби доступу до заборонених ресурсів. Логи дозволяють не лише відновити історію дій, але й допомагають виявляти аномалії або несанкціоновані спроби доступу. Для

забезпечення конфіденційності та цілісності даних логів важливо зберігати їх у захищених місцях і обмежити доступ до них тільки для авторизованих осіб.

Моніторинг полягає в постійному спостереженні за станом системи в реальному часі. Це включає перевірку працездатності серверів, наявності шкідливих запитів, зловмисної активності, навантаження на ресурси тощо. Використовуються спеціалізовані інструменти для моніторингу, які автоматично відслідковують критичні параметри, сповіщають про аномалії та дають змогу вчасно вжити заходів у разі загрози. Одним із прикладів таких інструментів є Prometheus, ELK Stack (Elasticsearch, Logstash, Kibana), або Splunk.

Системи логування і моніторингу можуть допомогти у своєчасному виявленні спроб атак (наприклад, SQL-ін’єкцій, XSS чи DDoS), виявленні вразливостей у програмному забезпеченні та розпізнаванні зловмисних дій у реальному часі. Вони також важливі для проведення аудиту безпеки і збереження доказів у разі інцидентів.

Важливою практикою є налаштування оповіщень про критичні події. Це дозволяє миттєво отримати інформацію про підозрілі або небезпечні ситуації, такі як спроби входу з незнайомих IP-адрес чи масові запити до сервера, що можуть свідчити про DDoS-атаку.

Також важливо, щоб логування та моніторинг не лише виконувалися в реальному часі, а й зберігалися для подальшого аналізу. Це допомагає виявити тренди і закономірності, що можуть свідчити про потенційні загрози або слабкі місця в системі.

Для підвищення безпеки в додаткових механізмах захисту важливо враховувати управління вразливостями, ізоляцію та контейнеризацію додатків, а також резервне копіювання і відновлення даних. Управління вразливостями включає регулярне сканування для виявлення уразливостей у коді, залежностях та конфігураціях за допомогою інструментів, таких як OWASP Dependency-Check або Snyk. Важливим аспектом є своєчасне оновлення залежностей, бібліотек і серверного програмного забезпечення для усунення вразливостей. Крім того, необхідно впроваджувати стратегію безпечного управління

залежностями, використовуючи інструменти для перевірки вразливостей у бібліотеках, наприклад, npm audit для Node.js.

Ізоляція та контейнеризація є ключовими аспектами забезпечення безпеки серверних додатків. Використання Docker дозволяє ізолювати додатки в окремих контейнерах, створюючи безпечне середовище для їх запуску, що знижує ризики атак. Мікросервісна архітектура, в свою чергу, розподіляє додатки на незалежні компоненти, обмежуючи масштаби потенційної атаки. Для додаткового підвищення безпеки важливо налаштувати політики безпеки для контейнерів, зокрема, застосовувати Docker security profiles та AppArmor, що допомагає контролювати доступ до ресурсів і захищати систему від небажаних втручань.

Резервне копіювання та відновлення даних включає регулярне створення бекапів для забезпечення відновлення інформації в разі атак або системних збоїв. Важливим елементом є також розробка плану відновлення після катастроф, що визначає стратегію оперативного відновлення функціонування системи після великих інцидентів безпеки, таких як атаки на сервери чи фізичні пошкодження інфраструктури.

Принципи безпеки включають принцип найменших привілеїв, що передбачає надання кожному користувачу та процесу лише тих прав доступу, які необхідні для виконання конкретних завдань; принцип оборонної стратегії (defense in depth), що полягає в застосуванні багаторівневих засобів захисту для підвищення стійкості системи до атак; та принцип безпеки через невизначеність, що знижує ймовірність успішної атаки шляхом ускладнення доступу до конфіденційної інформації, наприклад, через замасковані налаштування серверів або генерацію складних API-ключів.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ЗАХОДІВ БЕЗПЕКИ ДЛЯ СЕРВЕРНОЇ ЧАСТИНИ НА ПЛАТФОРМІ NODE.JS

3.1 Імплементація захисту від Spoofing та Tampering

Для ефективного захисту серверної частини системи на платформі Node.js було обрано системний підхід, оснований на методології STRIDE. Цей підхід дозволив системно виявити й класифікувати потенційні загрози безпеці. На основі отриманих результатів були впроваджені конкретні механізми захисту, які нейтралізують загрози, ідентифіковані на попередніх етапах розробки. Всі ці механізми спрямовані на забезпечення цілісності, конфіденційності та доступності даних, а також на запобігання спробам зловмисників обійти чи зламати систему.

Розглянемо детальніше кожен з типів загроз, визначених методологією STRIDE, та відповідні впроваджені заходи захисту. Захист від Spoofing, для запобігання несанкціонованому доступу до системи реалізовано механізми автентифікації та авторизації, особливо через використання токенів доступу та протоколів безпеки, таких як OAuth2.

Стосовно запобіганню зміні даних (Tampering), то для забезпечення цілісності даних використовується механізм підпису даних (HMAC), що дозволяє запобігти несанкціонованим змінам. Крім того, для захисту від несанкціонованих дій реалізовано механізм контролю доступу на базі ролей, що дозволяє лише адміністраторам або власникам змінювати та видаляти свої дані.

Гарантія авторства (Repudiation) дає забезпечення аудиту та запобігання відмовам у діях було впроваджено систему журналювання всіх критичних дій користувачів, що дозволяє визначити, хто і коли виконував певні операції в системі.

Для захисту від витоків конфіденційних даних (Information Disclosure) застосовано шифрування на всіх рівнях, включаючи зберігання чутливої інформації в зашифрованому вигляді. Також введено жорстке управління доступом до баз даних та інших чутливих ресурсів.

Для запобігання атак типу DoS були налаштовані механізми обмеження кількості запитів на сервер, а також впроваджено моніторинг та автоматичне виявлення аномальних навантажень.

Для запобігання підвищенню привілеїв (Elevation of Privilege) у системі реалізовані багаторівневі механізми контролю доступу, а також регулярне оновлення системи безпеки для усунення потенційних вразливостей, які могли б бути використані для несанкціонованого підвищення прав.

Кожен з цих механізмів відповідає конкретним типам загроз у рамках методології STRIDE та був реалізований через поєднання налаштувань, фрагментів коду та пояснень, що описують їх застосування для забезпечення безпеки серверної частини системи.

Отже, першою загрозою виступає «Підробка» (Spoofing). Spoofing — це загроза, пов’язана з можливістю зловмисника видавати себе за іншого користувача або систему, отримуючи доступ до захищених ресурсів [4]. Підробка ідентичності користувача є однією з основних загроз для серверних систем. Щоб запобігти цій загрозі, було реалізовано механізм аутентифікації, заснований на використанні JWT-токенів (JSON Web Tokens). Цей підхід забезпечує надійний метод перевірки автентичності користувача під час кожного запиту до сервера.

Аутентифікація за допомогою JWT. Користувач отримує пару токенів (access і refresh) після успішного входу в систему. Access-токен має короткий термін дії (15 хвилин), тоді як refresh-токен дозволяє отримувати нові access-токени протягом 7 днів. Це знижує ризики компрометації даних у разі викрадення токена.

Перевірка токена в middleware. Middleware authMiddleware перевіряє наявність і дійсність токена в cookie клієнта. У разі відсутності токена або якщо він недійсний, сервер повертає помилку автентифікації. Це запобігає доступу зловмисників, які не мають валідного токена.

Перевірка токена на термін дії. За допомогою jwt.verify токен розшифровується, і система перевіряє, чи не минув його термін дії (decoded.exp). Якщо токен застарів, користувач отримує відповідне повідомлення.

Перевірка існування користувача. Після розшифрування токена система здійснює додаткову перевірку, щоб переконатися, що користувач з відповідним userId все ще існує в базі даних. Це запобігає доступу до системи, якщо користувача було видалено.

Включення ролей у токен. У токені зберігається роль користувача (admin або user), що дозволяє обмежити доступ до різних ресурсів системи залежно від прав доступу.

Генерація нових токенів через refresh-токен. Якщо access-токен закінчився, користувач може отримати новий за допомогою refresh-токена через middleware refreshTokenMiddleware. Refresh-токен також перевіряється на дійсність. На рисунку 3.1 зображено фрагмент коду з перевірки токена, на рисунку 3.2 зображено функцію генерації токенів, а на рисунку 3.3 зображено код з обробки оновлення токена.

```
const authMiddleware = async (req, res, next) => {
    const token = req.cookies.token; // Токен з cookie
    if (!token) {
        return res.status(401).json({ message: "Authorization denied" });
    }
    try {
        const decoded = jwt.verify(token, process.env.JWT_SECRET);
        if (decoded.exp * 1000 < Date.now()) {
            return res.status(401).json({ message: 'Token expired' });
        }
        const user = await User.findById(decoded.userId);
        if (!user) {
            return res.status(401).json({ message: 'User not found' });
        }
        req.user = { userId: decoded.userId, role: decoded.role }; // Додаємо інформацію про користувача
        next();
    } catch (error) {
        res.status(401).json({ message: "Invalid token" });
    }
};
```

Рисунок 3.1 – Фрагмент перевірки токена

```
const generateTokens = (userId, role) => {
    const accessToken = jwt.sign({ userId, role }, jwtSecret, { expiresIn: '15m' }); // Access токен на 15 хвилин
    const refreshToken = jwt.sign({ userId, role }, jwtRefreshSecret, { expiresIn: '7d' }); // Refresh токен на 7 днів
    return { accessToken, refreshToken };
};
```

Рисунок 3.2 – Генерація токенів

```
const refreshTokenMiddleware = (req, res) => {
    const refreshToken = req.header('x-refresh-token');
    if (!refreshToken) {
        return res.status(401).json({ message: "Refresh token not provided" });
    }
    try {
        const decoded = jwt.verify(refreshToken, jwtRefreshSecret);
        const newTokens = generateTokens(decoded.userId, decoded.role); // Генеруємо нові токени
        res.json(newTokens);
    } catch (error) {
        res.status(403).json({ message: "Invalid refresh token" });
    }
};
```

Рисунок 3.3 – Обробка оновлення токена

Завдяки цьому підходу сервер запобігає підробці ідентичності, перевіряючи валідність і дійсність кожного токена. Токен містить унікальний ідентифікатор користувача (userId) і роль, що дає можливість розмежувати доступ до ресурсів. Зловмисники не можуть отримати доступ до системи без валідного токена. Таким чином, цей захід захисту ефективно протидіє загрозам Spoofing і забезпечує надійну автентифікацію користувачів.

Tampering передбачає несанкціоновану зміну даних під час їх передачі чи зберігання. Щоб протидіяти цій загрозі, було реалізовано низку заходів, що забезпечують цілісність даних і їх захист від несанкціонованих змін.

Також важливим механізмом захисту служить механізм підпису HMAC. Він забезпечує цілісність даних і дозволяє виявити будь-які зміни в них. У даному випадку HMAC використовується для створення JWT-токенів, які гарантують, що їх вміст не був змінений під час передачі між клієнтом і сервером.

Коли генерується токен, він підписується секретним ключем (JWT_SECRET та JWT_REFRESH_SECRET). Під час перевірки токена сервер використовує той самий секретний ключ для валідації токена. Якщо підпис не відповідає, сервер відхиляє запит як неавторизований.

Код на рисунку 3.4 генерує два JWT токени: access token (на 15 хвилин) і refresh token (на 7 днів) для користувача. Middleware перевіряє наявність токена в запиті, перевіряє його дійсність, і якщо токен правильний, додає дані користувача до запиту, інакше повертає помилку 401.

```
const generateTokens = (userId, role) => {
    const accessToken = jwt.sign({ userId, role }, jwtSecret, { expiresIn: '15m' }); // Access токен на 15 хвилин
    const refreshToken = jwt.sign({ userId, role }, jwtRefreshSecret, { expiresIn: '7d' }); // Refresh токен на 7 днів
    return { accessToken, refreshToken };
};

const authMiddleware = (req, res, next) => {
    const token = req.cookies.token;
    if (!token) return res.status(401).json({ message: "Authorization denied" });
    try {
        const decoded = jwt.verify(token, jwtSecret); // Перевірка HMAC-підпису
        req.user = { userId: decoded.userId, role: decoded.role };
        next();
    } catch (error) {
        res.status(401).json({ message: "Invalid token" });
    }
};
```

Рисунок 3.4 – Фрагмент кодової реалізації

Даний механізм запобігає змінам в токені або його даних (наприклад, ролей чи ідентифікаторів), оскільки будь-яка модифікація призведе до порушення HMAC-підпису.

І разом з механізмом підпису HMAC було реалізовано middleware. У middleware реалізовано перевірку ролі користувача (admin чи user), що обмежує можливість змінювати чи видаляти дані лише до авторизованих осіб.

На рисунку 3.5 реалізація коду, що для виконання чутливих дій, таких як редагування або видалення постів, перевіряється роль користувача та відповідність його ідентифікатора (userId) автору посту. Ця логіка запобігає несанкціонованим змінам даних, обмежуючи доступ до них лише адміністраторам або авторам постів. Це гарантує, що лише адміністратор може виконувати ці дії над будь-якими постами, а користувач може редагувати або видаляти лише свої власні пости.

```
router.put('/edit-post/:id', authMiddleware, async (req, res) => {
    const { id } = req.params;
    if (!mongoose.Types.ObjectId.isValid(id)) {
        return res.status(400).json({ message: "Invalid ID format" });
    }

    try {
        const post = await Post.findById(id);

        if (!post) {
            return res.status(404).json({ message: "Post not found" });
        }

        // Перевірка прав доступу
        if (req.user.role !== 'admin' && post.author.toString() !== req.user.userId) {
            return res.status(403).json({ message: "You do not have permission to edit this post" });
        }

        // Оновлюємо пост
        post.title = title;
        post.body = body;
        post.updatedAt = Date.now();
        await post.save();

        res.json( post );
    } catch (error) {
        console.log(error);
        res.status(500).json({message: "Server Error"})
    }
})
```

Рисунок 3.5 – Реалізація коду перевірки прав доступу користувача та оновлення поста

Також важливим аспектом є логування та аудит. Для забезпечення додаткового захисту та моніторингу система автоматично фіксує дії користувачів (створення, редагування, видалення постів), зберігаючи при цьому їх IP-адреси та часові мітки. Це дає змогу виявляти підозрілі дії та реагувати на них у разі потреби. На рисунку 3.6 зображено фрагмент коду реалізації логування.

```
const logUserAction = (action, userId, ipAddress) => {
    console.log(`[${new Date().toISOString()}] Action: ${action}, UserID: ${userId}, IP: ${ipAddress}`);
};
```

Рисунок 3.6 – Фрагмент кодової реалізація прикладу логування

Комбінація HMAC-підпису токенів, перевірки ролей користувачів і реалізації системи аудиту дозволяє ефективно протидіяти загрозам Tampering.

Дані захищені від несанкціонованих змін, а всі дії користувачів ретельно контролюються.

3.2 Реалізація захисту від Repudation та від Information Disclosure

Repudiation (Відмова) виникає, коли користувач може заперечувати виконання певної дії в системі, наприклад, створення, редагування або видалення даних. Щоб запобігти цьому, у даній системі реалізовано механізм аудит-логу, який дозволяє відслідковувати дії користувачів із зазначенням автора, часу дії та додаткових даних.

Реалізація аудит-логу. Аудит-лог — це механізм запису дій користувачів, що включає ідентифікатор користувача, тип дії, наприклад, створення, редагування, видалення, час виконання (точна часова мітка), а також IP-адресу клієнта для додаткової ідентифікації джерела дії.

Цей механізм працює наступним чином: кожна критична дія, наприклад, створення, редагування або видалення посту, записується в аудит-лог із зазначенням ключової інформації про дію. Це дозволяє забезпечити прозорість, довести факт виконання дії конкретним користувачем у разі спорів та виявляти підозрілі дії. На рисунку 3.7 та 3.8 відображається взаємодія стосовно бази даних MongoDB.

```
const mongoose = require('mongoose');

const AuditLogSchema = new mongoose.Schema({
    userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
    action: { type: String, required: true },
    ipAddress: { type: String, required: true },
    timestamp: { type: Date, default: Date.now }
});

const AuditLog = mongoose.model('AuditLog', AuditLogSchema);

const logUserAction = async (userId, action, ipAddress) => {
    try {
        await AuditLog.create({ userId, action, ipAddress });
    } catch (error) {
        console.error('Error logging user action:', error);
    }
};

module.exports = { logUserAction, AuditLog };
```

Рисунок 3.7 – Код запису у лог

```
router.post('/add-new-post', authMiddleware, async (req, res) => {
    try {
        const { title, body } = req.body;

        // Перевірка наявності ролі та визначення доступу
        if (!req.user || !req.user.role) {
            return res.status(403).json({ message: 'Forbidden' });
        }

        // Створення нового поста
        const newPost = new Post({
            title: sanitizedTitle,
            body: sanitizedBody,
            author: req.user.userId
        });
        const post = await newPost.save();

        // Логування дії користувача
        const userId = req.userId;  // Передбачається, що у вас є користувач після middleware authMiddleware
        const action = 'Created a new post';
        const ipAddress = req.ip;  // IP-адреса користувача

        await logUserAction(userId, action, ipAddress);  // Логуємо дію

        res.status(200).json({ message: 'Post created successfully', post });
    } catch (error) {
        res.status(500).json({message: "Post create error", error: error.message});
    }
})
```

Рисунок 3.8 – Фрагмент коду реалізації запису у лог в контролері

Результати реалізації забезпечують прозорість, оскільки кожна дія користувача записується в лог, що унеможливлює її заперечення; контроль, адже у випадку підозрілих дій чи порушень легко відслідкувати, хто та коли їх виконав; та відповідальність, оскільки використання IP-адреси та унікального ідентифікатора користувача додає додатковий рівень захисту.

Механізм аудит-логу ефективно усуває загрозу Repudiation, забезпечуючи можливість перевірки кожної дії в системі. Це є ключовим елементом для побудови довіри до системи й доказової бази у випадках конфліктних ситуацій.

На відміну від Відмови, “Information Disclosure” (Розголошення інформації) виникає, коли конфіденційна інформація стає доступною несанкціонованим користувачам. Це може відбутися через вразливості в коді, незахищені маршрути, некоректні повідомлення про помилки або недостатньо захищені поля вводу.

Захист від XSS-атак. Було впроваджено ефективні механізми перевірки та фільтрації вхідних даних, що дозволяє запобігти впровадженню шкідливого коду в такі поля, як title і body. Це забезпечує захист від виконання небажаного JavaScript-коду, який може бути використаний для викрадення особистих даних користувачів або здійснення атак на сайт, що, в свою чергу, підвищує безпеку веб-ресурсу. На рисунку 3.9 зображено код для фільтрації та перевірки.

```
const sanitizeHtml = require('sanitize-html');  // Для санітизації HTML в body

router.post('/add-new-post', authMiddleware, async (req, res) => {
    try {
        const { title, body } = req.body;

        // Перевірка наявності ролі та визначення доступу
        if (!req.user || !req.user.role) {
            return res.status(403).json({ message: 'Forbidden' });
        }

        // Санітизація body: дозволяємо теги <b>, <i>, <a> і т.д., за потреби налаштуйте список дозволених тегів
        const sanitizedBody = sanitizeHtml(body, {
            allowedTags: [ 'b', 'i', 'em', 'strong', 'a', 'ul', 'ol', 'li', 'p', 'br' ],  // Дозволяємо базові теги
            allowedAttributes: {
                'a': ['href', 'target']  // Дозволяємо тільки атрибути href та target для <a>
            },
        });
        // Створення нового поста
        const newPost = new Post({
            title: sanitizedTitle,
            body: sanitizedBody,
            author: req.user.userId
        });
        const post = await newPost.save();=

        res.status(200).json({ message: 'Post created successfully', post });
    } catch (error) {
        res.status(500).json({message: "Post create error", error: error.message});
    }
})
```

Рисунок 3.9 – Фрагмент коду для фільтрації та перевірки в контролері

Також важливими аспектами є використання helmet та обмеження повідомлень стосовно помилок на сервері. Бібліотека Helmet інтегрована в даному сервері для автоматичного налаштування критичних заголовків безпеки. Це суттєво зменшує ризик витоку вразливої інформації та захищає веб-додатки від потенційних атак, таких як XSS, крос-сайт скриптинг і інші види експлуатації вразливостей. На рисунках 3.10 та 3.11 зображено фрагменти коду налаштування helmet та його приклад детальнішої конфігурації.

```
const helmet = require('helmet');

// Використовуємо Helmet для налаштування заголовків безпеки
app.use(helmet());
```

Рисунок 3.10 – Код автоматичнооого налаштування для helmet

```
const helmet = require('helmet');

// Додавання заголовків безпеки
app.use(helmet({
    contentSecurityPolicy: {
        directives: {
            defaultSrc: ["'self'"],
            scriptSrc: ["'self'", "'unsafe-inline'"],
            styleSrc: ["'self'", "'unsafe-inline'"],
            imgSrc: ["'self'", "data:"]
        }
    },
    referrerPolicy: { policy: 'no-referrer' }
}));
```

Рисунок 3.11 – Приклад коду більш детальної конфігурації helmet

Вимкнення надмірно докладних повідомлень про помилки є важливим кроком для забезпечення безпеки системи, оскільки такі повідомлення можуть розкрити її внутрішню структуру. У даній системі конфіденційні дані, зокрема стеки помилок, не передаються користувачам, а відповіді сервера містять лише загальні повідомлення про помилки, що зменшує ризик витоку інформації. На рисунку 3.12 зображено загальну обробку помилок без вказання будь-яких деталей.

```
router.post('/admin/login', loginLimiter, async (req, res) => {
    try {
        const { username, password } = req.body;
        const user = await User.findOne({ username: sanitizedUsername });
        if (!user) {
            res.status(401).json({ message: 'Invalid credentials' });
        }
        // Check password
        const isPasswordValid = await bcrypt.compare(password, user.password)
        if (!isPasswordValid) {
            user.loginAttempts = (user.loginAttempts || 0) + 1;
            // Блокуємо користувача, якщо досягнуто максимальної кількості спроб
            if (user.loginAttempts >= MAX_LOGIN_ATTEMPTS) {
                user.lockUntil = Date.now() + LOCK_TIME;
            }
            await user.save();
            res.status(401).json({ message: 'Invalid credentials' });
        }
        // Token generation
        const token = jwt.sign({ userId: user._id, username: sanitizedUsername, role: user.role }, jwtSecret, options);
        res.cookie('token', token, { httpOnly: false, path: '/' });
        res.status(200).json({ message: "Login was succesessful" });
    } catch (error) {
        console.log("Some error with admin:", error);
        res.status(500).json({ message: 'Server error' });
    }
})
```

Рисунок 3.12 – Обробка помилок в контролері

Результати реалізації захисту достатньо прості, але в той же ж момент достатньо ефективні.

Захист від XSS. Завдяки впровадженню заходів безпеки, зловмисники не можуть впроваджувати шкідливий код у ваші форми вводу, зокрема у поля, як-от title та body. Це дає надійний захист стосовно атак, які спричиняють зміну або викрадення даних користувачів.

Безпечні HTTP-заголовки. Використання бібліотеки helmet гарантує додатковий рівень захисту для вашої системи. Вона блокує багато можливих атак, зокрема ті, що виникають через некоректну обробку HTTP-заголовків. Це допомагає уникнути вразливостей, які можуть бути використані для компрометації серверу.

Контроль помилок. Впровадження загальних повідомлень про помилки значно знижує ризик розголошення важливої інформації про внутрішню структуру сервера. Такий підхід дозволяє захистити систему від потенційних зловмисників, які можуть скористатися цією інформацією для подальших атак.

Реалізація заходів захисту від Information Disclosure забезпечує захист конфіденційності даних, мінімізує ризик витоку інформації та підвищує загальний рівень безпеки системи. Завдяки цьому користувачі системи можуть бути впевнені, що їхні дані надійно захищені.

3.3 Механізми захисту від Denial of Service та Elevation of Privilege

Denial of Service (DoS) — це атака, що спрямована на недоступність сервера для користувачів шляхом перевантаження його великою кількістю запитів. Це може призвести до виведення сервера з ладу або значного зниження його продуктивності, що в результаті порушує нормальну роботу системи.

Для забезпечення захисту від надмірної кількості запитів впроваджено механізм обмеження частоти на окремих маршрутах. Наприклад, кількість спроб входу до системи обмежено впродовж певного проміжку часу, що дозволяє

запобігти атакам методом грубої сили (brute force) та захистити сервер від перевантаження через надмірну кількість запитів.

Створено та налаштовано middleware для обмеження кількості запитів при певних діях. На рисунку 3.13 зображено кодову реалізацію middleware, а на рисунку 3.14 його застосування.

```
const rateLimit = require('express-rate-limit');

// Обмеження для логіну (5 запитів на 15 хвилин)
const loginLimiter = rateLimit({
    windowMs: 15 * 60 * 1000, // 15 хвилин
    max: 5, // максимальна кількість запитів
    message: { message: 'Too many login attempts, please try again later' },
    standardHeaders: true, // відображає інформацію про обмеження в заголовках
    legacyHeaders: false, // відключає старі заголовки
});

module.exports = loginLimiter;
```

Рисунок 3.13 – Реалізація middleware для обмеження запитів

```
const loginLimiter = require('../middleware/loginLimiter');

router.post('/admin/login', loginLimiter, async (req, res) => {
    // Логіка контролера
})
```

Рисунок 3.14 – Використання middleware до маршруту логіну

Потрібним функціоналом для зменшення навантаження на сервер та базу даних при отриманні постів реалізовано механізм кешування. Кешування дає змогу тимчасово зберігати результати часто виконуваних запитів, що значно знижує кількість звернень до бази даних та покращує продуктивність системи. На рисунку 3.15 зображено кешування запитів в контролері.

```
const NodeCache = require("node-cache");
const cache = new NodeCache({ stdTTL: 600 }); // Кешування на 10 хвилин

router.get('/posts', async (req, res) => {
    const page = parseInt(req.query.page);
    const limitOnPage = parseInt(req.query.limitOnPage);
    const startIndex = (page - 1) * limitOnPage;
    const endIndex = page * limitOnPage;
    // Генерація унікального ключа для кешу, що враховує сторінку і ліміт
    const cacheKey = `posts_page_${page}_limit_${limitOnPage}`;
    // Перевірка, чи є кешовані дані
    const cachedPosts = cache.get(cacheKey);
    if (cachedPosts) {
        return res.json(cachedPosts);
    }
    try {
        const posts = await Post.find()
            .sort({ createdAt: -1 })
            .skip(startIndex)
            .limit(limitOnPage);
        const paginatedPosts = posts.slice(startIndex, endIndex);
        const totalPages = Math.ceil(posts.length / limitOnPage);
        // Кешуємо отримані дані
        const cacheData = { posts: paginatedPosts, totalPages };
        cache.set(cacheKey, cacheData);
        res.json({ posts: paginatedPosts, totalPages });
    } catch (error) {
        console.log(error);
        res.status(500).json({message: "Server Error"})
    }
})
```

Рисунок 3.15 – Код контролера в якому кешуються запити

Реалізація механізмів rate limiting та кешування сприяє підвищенню безпеки та продуктивності системи. Rate limiting захищає від атак як метод перебору (brute force) на маршруті логіну, а також запобігає перевантаженню серверу через надмірну кількість запитів з одного джерела. Кешування, в свою чергу, знижує навантаження на базу даних, забезпечуючи швидку відповідь на запити користувачів та покращує загальну продуктивність, особливо при роботі з часто запитуваними даними, такими як пости.

Реалізація механізмів захисту від атак типу Denial of Service (DoS) є критично важливою для забезпечення стабільної роботи серверів навіть під час високих навантажень. Використання обмеження частоти запитів та ефективне кешування дозволяє уникнути перевантаження ресурсів, значно підвищує

продуктивність системи та забезпечує безперебійний доступ для законних користувачів.

На сервері на платформі Node.js реалізовано заходи для запобігання атакам типу "підвищення привілеїв", які можуть виникати, коли класичний користувач намагається отримати доступ до функцій адміністратора або інших захищених ресурсів. Для захисту системи від цього типу загроз впроваджено такі механізми: розподіл прав доступу, middleware для перевірки ролей, обмеження доступу та певний захист від несанкціонованого доступу.

Надзвичайно важливою деталлю є система, яка забезпечує чітке розмежування ролей між адміністраторами та звичайними користувачами. Це досягається через додавання атрибута role до токена користувача, який зберігається в базі даних та передається із допомогою JWT (JSON Web Token). Даний підхід дозволяє ефективно контролювати доступ до ресурсів та функціоналу системи залежно від ролі користувача. На рисунку 3.16 зображено дані, які знаходяться в токені, включаючи роль користувача.

Рисунок 3.16 – Дані, які містяться в токені

Структура токена дає змогу серверу ідентифікувати користувача та перевіряти його права доступу на основі ролі, що забезпечує належний рівень безпеки та контролю доступу.

У деяких контролерах реалізовано логіку декодування токена та визначення ролі користувача. Роль користувача передається через об'єкт req.user, що

дозволяє виконувати подальші перевірки доступу на основі ролі. Цей підхід забезпечує централізовану перевірку аутентифікації та авторизації для зручного керування доступом до ресурсів. На рисунку 3.17 зображено логіку редагування вмісту в одному з контролерів.

```
// Перевірка прав доступу
if (req.user.role !== 'admin' && post.author.toString() !== req.user.userId) {
    return res.status(403).json({ message: "You do not have permission to edit this post" });
}
```

Рисунок 3.17 – Фрагмент коду контролера для редагування вмісту

Така перевірка гарантує, що можливість змінювати вміст мають тільки користувачі в яких є адміністративні можливості оскільки вони мають таку роль або ж ті хто є творцями такого вмісту.

У кожному контролері реалізована логіка перевірки прав доступу, що визначає рівень доступу різних категорій користувачів. Адміністратори мають повний доступ до всього функціоналу, особливо можуть редагувати та видаляти пости, створені будь-якими користувачами. Звичайні користувачі обмежені в своїх діях і можуть виконувати операції редагування та видалення лише щодо власних постів. На рисунку 3.18 зображено логіку для видалення поста в певному контролері.

```
// Перевірка прав доступу
if (req.user.role !== 'admin' && post.author.toString() !== req.user.userId) {
    return res.status(403).json({ message: "You do not have permission to delete this post" });
}
```

Рисунок 3.18 – Фрагмент коду з контролера для видалення поста

Окрім перевірки ролей, було реалізовано додатковий захист від несанкціонованого доступу до захищених ресурсів. Маршрути, що призначені для адміністратора, захищені спеціальним middleware, яке здійснює перевірку наявності ролі "admin" у користувача. Звичайні ж маршрути доступні для всіх користувачів, які мають дійсні токени, що підтверджують їхню автентичність. На рисунку 3.19 зображено код маршруту який призначений для користувачів з

роллю адміністратор, а для звичайних авторизованих користувачів призначений абсолютно інший маршрут.

```
router.get('/admin/dashboard', authMiddleware, async (req, res) => {

    const page = parseInt(req.query.page) || 1;
    const limitOnPage = parseInt(req.query.limitOnPage) || 10;
    const startIndex = (page - 1) * limitOnPage;

    try {
        let query = {};

        // Якщо користувач є звичайним, фільтруємо за його userId
        if (req.user.role !== 'admin') {
            query.author = req.user.userId;  // Використовуємо req.user.userId замість req.userId
        }
        // Знаходимо пости з врахуванням ролі та пагінації
        const posts = await Post.find(query)
            .sort({ createdAt: -1 })
            .skip(startIndex)
            .limit(limitOnPage);
        // Підрахунок загальної кількості постів для пагінації
        const totalPosts = await Post.countDocuments(query);
        const totalPages = Math.ceil(totalPosts / limitOnPage);

        res.json({ posts, totalPages });
    } catch (error) {
        console.log(error);
        res.status(500).json({message: "Server Error"})
    }
})
```

Рисунок 3.19 – Код маршруту для адміністративного доступу

Завдяки впровадженню механізму ролей та перевірок у middleware і контролерах, система надійно захищена від атак типу "підвищення привілеїв". Такий підхід гарантує, що кожен користувач має доступ виключно до тих функцій, які відповідають його ролі. Це значно підвищує рівень безпеки серверної частини системи та ефективно запобігає компрометації адміністративних прав і функцій.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

Дослідження механізмів захисту для серверної частини на платформі Node.js із використанням методології STRIDE має важливе значення не лише для забезпечення безпеки даних, але й для дотримання вимог охорони праці, техніки безпеки та протипожежної безпеки. Створені механізми повинні враховувати аспекти безпеки як для користувачів веб-додатків, так і для розробників, які впроваджують і підтримують ці механізми на своїх робочих місцях.

При розробці механізмів захисту для методології STRIDE для серверної частини Node.js особливу увагу приділено дотриманню вимог нормативних документів з охорони праці, зокрема положень Закону України "Про охорону праці" (стаття 13), Кодексу законів про працю України та НПАОП 0.00-7.11-12 "Загальні вимоги стосовно забезпечення роботодавцями охорони праці працівників". Охорона праці, згідно із Законом України №2694-XII, передбачає створення безпечних умов праці для усіх осіб, залучених до розробки та експлуатації інформаційних систем. У процесі розробки серверної частини на платформі Node.js враховувались основні принципи безпеки праці для IT-працівників, зокрема запобігання ризикам, пов’язаним із надмірним навантаженням, стресом та потенційними аваріями (наприклад, збій у роботі серверів, втрата даних, кібератаки, аварії в системах енергопостачання і т.д.).

Ергономічні аспекти безпеки розроблених механізмів захисту передбачають мінімізацію ризиків для здоров'я працівників при тривалій роботі з інформаційними системами. Відповідно до ДСанПіН 3.3.2-007-98 "Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин", враховано норми тривалості безперервної роботи, регламентовані перерви та оптимальні параметри робочого місця. Зокрема, розроблені механізми захисту мають інтерфейси та налаштування, які дозволяють:

– Зменшувати навантаження на зоровий апарат шляхом оптимізації контрастності та яскравості відображення інформації;

– Забезпечувати швидкість реакції системи, що знижує психоемоційне напруження користувача;

– Мати вбудовані механізми сповіщення про необхідність перерви або зміни робочої пози.

Електробезпека при роботі з серверною інфраструктурою на базі Node.js є критичним аспектом охорони праці. Відповідно до Правил технічної експлуатації електроустановок споживачів, затверджених Наказом Міністерства енергетики України від 24.03.2022 № 204, розроблені механізми захисту враховують вимоги до:

– Заземлення обладнання;

– Захисту від електростатичних розрядів;

– Використання сертифікованого електрообладнання;

– Захисту від несанкціонованого доступу до електричних комунікацій.

Пожежна безпека при експлуатації серверної інфраструктури також має бути на найвищому рівні. НАПБ А.01.001-2014 "Правила пожежної безпеки в Україні" висувають жорсткі вимоги до приміщень, де розташовується серверне обладнання [22]. Механізми захисту серверної частини Node.js повинні передбачати:

– Моніторинг температурного режиму;

– Автоматичні системи охолодження;

– Наявність первинних засобів пожежогасіння;

– Резервування критично важливих систем.

– Монтаж пожежних сигналізацій із системами автоматичного реагування на займання;

– Використання негорючих матеріалів для корпусів обладнання;

– Встановлення систем вентиляції та кондиціонування для запобігання перегріву серверів.

Додатково враховано вимоги Наказу МНС України № 74 від 19.03.2018 "Про затвердження Правил пожежної безпеки в інформаційно-

телекомунікаційних системах" щодо забезпечення максимальної безпеки під час роботи з електронними системами.

Крім того, рекомендується застосовувати принципи резервування серверів і мережевих з’єднань, щоб уникнути збоїв у разі аварійної ситуації. Це не лише підвищує загальну надійність системи, але й мінімізує ризики для працівників, які можуть виникнути у разі аварійних втручань.

Захист працівників також охоплює контроль за рівнем стресу і навантаженням у процесі роботи з серверними системами. Підтримка безперебійного функціонування системи знижує ризик виникнення екстрених ситуацій, які могли б вимагати понаднормової роботи чи оперативного втручання. Завдяки цьому розроблені механізми не тільки гарантують безпеку даних, але й сприяють створенню безпечних і комфортних умов праці для IT-фахівців.

Розроблені механізми захисту для серверної частини на платформі Node.js відповідають вимогам охорони праці, техніки безпеки та протипожежної безпеки. Їх впровадження забезпечує захист працівників від потенційних ризиків, пов’язаних із кіберзагрозами, технічними збоями та аварійними ситуаціями, а також сприяє створенню безпечного та ефективного робочого середовища.

4.2 Безпека в надзвичайних ситуаціях

Воєнний час стає серйозним випробуванням для економіки будь-якої держави, зокрема для промислових підприємств. Приладобудівна галузь, яка забезпечує технологічні основи для функціонування різних секторів економіки та оборони, є однією з найважливіших складових промислового комплексу країни. Її стабільна робота в умовах воєнних загроз є не лише питанням економічної безпеки, але й стратегічним пріоритетом для держави.

Сучасні реалії воєнного конфлікту формують принципово нове середовище функціонування промислових підприємств, де традиційні системи безпеки та менеджменту виявляються недостатньо ефективними. Специфіка

приладобудівної галузі передбачає наявність складного високотехнологічного обладнання, чутливого до зовнішніх впливів, що додатково ускладнює завдання забезпечення безпеки.

У нових умовах підприємства приладобудівної галузі стикаються з низкою викликів, які суттєво ускладнюють їх функціонування. Це перебої у постачанні сировини, руйнування інфраструктури, ризики кібератак, втрата кадрів через мобілізацію та міграцію, а також загальний економічний спад. Водночас потреба у продукції приладобудівної галузі зростає через важливість її внеску у військові та оборонні програми.

Проблеми безпеки в умовах надзвичайних ситуацій, особливо у воєнний час, набувають особливого значення для підприємств приладобудівної галузі. Високий рівень автоматизації, складні технологічні процеси та критичність продукції, яку вони виробляють, ставлять такі підприємства у зону підвищеного ризику. У випадку загрозливих подій, зокрема військових дій, кібератак чи техногенних катастроф, підприємства повинні швидко адаптуватися до нових умов та забезпечити безперервність своєї діяльності.

У зв’язку з цим постає необхідність розробки та впровадження комплексних заходів для підвищення стійкості підприємств галузі. Це включає технічні, організаційні та економічні рішення, спрямовані на мінімізацію ризиків та забезпечення безперебійної роботи у критичних умовах.

У воєнний час підприємства приладобудівної галузі стикаються з низкою проблем, які впливають на їхню здатність стабільно функціонувати. Ці виклики можна умовно поділити на кілька основних категорій: економічні, технічні, кадрові та безпекові.

Економічні виклики. Воєнні дії призводять до скорочення економічної активності, зниження попиту на продукцію та втрати ключових ринків збуту. Багато підприємств зазнають фінансових труднощів через нестачу обігових коштів, зростання вартості сировини та матеріалів, а також інфляцію. Додатково ситуацію ускладнює зростання витрат на логістику через зруйновану інфраструктуру та зміну ланцюгів постачання.

Економічна безпека підприємства потребує розроблення гнучкої стратегії адаптації до мінливих умов. Диверсифікація постачальників, формування фінансових резервів, впровадження адаптивної цінової політики дозволять мінімізувати економічні ризики. Кадрова політика має бути спрямована на максимальне збереження кваліфікованого персоналу, створення системи соціального захисту та підтримки колективу.

Технічні виклики. Підприємства приладобудівної галузі, залежні від імпортних компонентів та обладнання, стикаються з перебоями в їхньому постачанні через порушення логістичних ланцюгів, міжнародні санкції або блокування експорту з певних країн. Недостатність запасів сировини й матеріалів змушує підприємства шукати альтернативні джерела, що може вплинути на якість та продуктивність.

Кадрові виклики. Наприклад в Україні через мобілізацію працівників до лав Збройних Сил України, а також міграцію населення до інших регіонів чи за кордон, підприємства зіштовхуються з нестачею кваліфікованих кадрів. Заміна висококваліфікованих спеціалістів новими співробітниками або автоматизація окремих процесів потребує часу та ресурсів, що ускладнює оперативне реагування на виклики.

Безпекові виклики. Воєнний стан створює підвищений ризик для фізичної та кібернетичної безпеки підприємств. Інфраструктура приладобудівних підприємств може стати ціллю ракетних або диверсійних атак, що загрожує не лише матеріальними збитками, але й втратою виробничих потужностей. Одночасно зростає кількість кібератак на інформаційні системи, які використовуються для управління виробництвом, фінансами та логістикою.

Соціально-політичні виклики. Соціальна нестабільність, зміна пріоритетів державного фінансування та зростання рівня невизначеності значно ускладнюють довгострокове планування. Це негативно позначається на реалізації стратегічних проектів і впровадженні інновацій.

Таким чином, підприємства приладобудівної галузі опинилися в умовах багатовекторного впливу негативних факторів. Для забезпечення їхньої стійкості

потрібен комплексний підхід, що передбачає як внутрішні організаційні заходи, так і підтримку з боку держави та міжнародних партнерів.

Важливим аспектом є кібербезпека, адже цифрові системи управління, що активно використовуються у приладобудуванні, стають потенційними мішенями для атак. Забезпечення надійного захисту даних, які обробляються серверною частиною систем, відіграє ключову роль у стійкості підприємств до зовнішніх загроз. Методологія STRIDE, застосована у цій роботі, пропонує системний підхід до виявлення загроз і запобігання можливим вразливостям. Реалізовані механізми захисту, такі як авторизація через JWT, протидія XSS-атакам і запобігання SQL-ін’єкціям, можуть бути використані не лише для захисту окремих веб-додатків, але й для забезпечення кібербезпеки підприємств у цілому.

У воєнний час зростає ризик фізичних атак на підприємства. До таких загроз можна віднести ракетні удари, диверсії чи інші дії, що спрямовані на знищення виробничих потужностей або порушення ланцюгів постачання. Для мінімізації таких ризиків необхідно забезпечити фізичний захист об’єктів, включаючи укриття, посилення периметра, встановлення відеоспостереження та систем раннього оповіщення. Важливим є створення резервних виробничих майданчиків у регіонах, менш схильних до ризиків, що дозволяє уникнути повного зупинення діяльності у разі знищення основного підприємства.

Ще одним важливим завданням є організація роботи персоналу в умовах надзвичайних ситуацій. Це включає розробку планів евакуації, навчання співробітників діям у випадку аварій, забезпечення засобів індивідуального захисту та першої медичної допомоги. Особлива увага приділяється захисту інформаційних систем, адже компрометація даних може призвести до катастрофічних наслідків як для підприємства, так і для національної безпеки.

Соціально-психологічний вимір надзвичайних ситуацій також є критично важливим. Стресовий стан працівників, постійна психологічна напруга, невизначеність та страх за власне життя та життя близьких значно ускладнюють нормальне функціонування підприємства. Необхідність екстреної евакуації,

кардинальна зміна режимів роботи персоналу вимагають запровадження спеціальних антикризових психологічних програм.

Ефективна система захисту передбачає комплекс заходів для збереження життя та здоров'я працівників:

Колективний захист:

– Обладнання захисних споруд (бомбосховищ, укриттів);

– Створення системи швидкої евакуації;

– Забезпечення медичних пунктів;

– Організація харчування та водопостачання.

Індивідуальний захист:

– Забезпечення працівників засобами індивідуального захисту;

– Навчання правилам користування захисним обладнанням;

– Медичне страхування;

– Психологічна підтримка.

В умовах сучасної гібридної війни однією з ключових загроз стають кібератаки, спрямовані на критичну інфраструктуру. Зокрема, на підприємства приладобудівної галузі, які виготовляють компоненти для військової техніки. Використання механізмів моніторингу активності серверів і веб-додатків дозволяє виявляти підозрілі дії у реальному часі та запобігати порушенням.

Не менш важливим є питання зберігання даних у віддалених базах даних чи хмарних сховищах. У випадку фізичного знищення локальних серверів резервне копіювання даних дозволяє відновити роботу підприємства у найкоротший термін. Використання методів шифрування на рівні бази даних забезпечує конфіденційність даних навіть у випадку їх компрометації.

Таким чином, підвищення стійкості роботи підприємств приладобудівної галузі в умовах надзвичайних ситуацій базується на інтеграції фізичних, кібернетичних та організаційних заходів. Комплексний підхід, який включає кіберзахист, фізичну безпеку та готовність персоналу до надзвичайних ситуацій, дозволяє знизити ризики, пов’язані як із військовими діями, так і з іншими кризовими подіями. Результати дослідження, зокрема впровадження механізмів захисту на базі методології STRIDE, можуть бути адаптовані для використання

у системах захисту підприємств приладобудівної галузі, підвищуючи їхню стійкість до сучасних загроз.

Таким чином, підвищення стійкості підприємств приладобудівної галузі в умовах воєнного часу є можливим завдяки скоординованим діям на рівні самих підприємств, держави та міжнародної спільноти. Реалізація запропонованих заходів дозволить не лише зберегти ключові виробничі потужності, але й створити основу для відновлення та подальшого розвитку галузі у післявоєнний період.

ВИСНОВКИ

У процесі роботи над системою захисту серверної частини веб-додатка на платформі Node.js було реалізовано комплексний підхід до забезпечення безпеки. Використовуючи методологію STRIDE, вдалося систематично ідентифікувати потенційні загрози, визначити їхній вплив на систему та впровадити відповідні заходи захисту.

Реалізовано ряд основних досягнень для сервера:

– Запобігання підробці ідентифікаційної інформації (Spoofing). Реалізовано аутентифікацію та авторизацію за допомогою JWT-токенів, які забезпечують надійне підтвердження особи користувачів. Крім того, впроваджено перевірку ролей (адміністратор або звичайний користувач) для розмежування доступу до різних функцій;

– Захист від втручання в дані (Tampering). Використано HMAC для підпису даних, що гарантує їхню цілісність під час передачі між клієнтом і сервером. Крім того, обмежено дії користувачів у контролерах відповідно до їхніх ролей: лише автори можуть змінювати або видаляти свої пости, а адміністратори мають повний доступ до всіх постів;

– Запобігання відмові від авторства (Repudiation). Впроваджено аудит-лог, який фіксує всі ключові дії користувачів, включаючи час виконання та IP-адресу. Це забезпечує прозорість і підтвердження авторства дій;

– Запобігання розголошенню інформації (Information Disclosure). Реалізовано захист від XSS-атак, використовуючи бібліотеки для екранування та санітизації вхідних даних. Налаштовано заголовки безпеки за допомогою helmet, що мінімізує ризик розголошення чутливої інформації через відповіді сервера;

– Запобігання відмові в обслуговуванні (Denial of Service). Впроваджено “rate limiting”, що обмежує кількість запитів до сервера за певний період, а також кешування для оптимізації запитів до бази даних. Це знижує ризик перевантаження сервера та покращує його продуктивність;

– Контроль привілеїв (Elevation of Privilege). Реалізовано чіткий поділ ролей у системі. Адміністратори мають доступ до всіх функцій, включаючи управління

постами інших користувачів, тоді як звичайні користувачі можуть редагувати та видаляти лише свої власні пости.

Наукова новизна роботи полягає у використанні методології STRIDE для систематичного аналізу та запобігання загрозам у серверних додатках, реалізованих на платформі Node.js. Інтеграція класичних підходів безпеки, таких як контроль привілеїв і аудит-логи, із основними методами, включаючи JWT та HMAC, дозволила створити універсальну модель безпеки. Окрім того, застосування механізмів кешування та rate limiting забезпечує не лише підвищення рівня безпеки, а й оптимізацію для продуктивності системи.

Для практичної значущості реалізовано механізми захисту, які суттєво підвищують безпеку серверної частини додатка, забезпечуючи конфіденційність, цілісність та доступність даних. Використання методології STRIDE дозволило структуровано підходити до аналізу загроз і впроваджувати рішення, що відповідають пріоритетним вимогам безпеки.

Таким чином, впроваджені заходи забезпечують високий рівень безпеки серверної частини системи, роблячи її стійкою до більшості поширених типів загроз.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is a Web Application?[електронний ресурс] / AWS Amazon – Режим доступу до ресурсу: [https://aws.amazon.com/what-is/web-application/] (дата звернення: 11.10.2024)
2. Cybersecurity Stats: Facts And Figures You Should Know[електронний ресурс] / Mariah St. John, Brenna Swanston, Ayona Chatterjee, Ph.D. – Режим доступу до ресурсу: [https://www.forbes.com/advisor/education/it-and-tech/cybersecurity-statistics/] (дата звернення: 12.10.2024)
3. In-depth Guide to Web Server Security with 6 Real-Life Examples[електронний ресурс] / Cem Dilmegani – Режим доступу до ресурсу: [https://research.aimultiple.com/web-server-security/] (дата звернення: 16.10.2024)
4. Tymoshchuk, V., Pakhoda, V., Dolinskyi, A., Karnaukhov, A., & Tymoshchuk, D. (2024). MODELLING CYBER THREATS AND EVALUATING THE PERFORMANCE OF INTRUSION DETECTION SYSTEMS. Grail of Science, (46), 636–641. https://doi.org/10.36074/grail-of-science.29.11.2024.081
5. Threat Modeling Methodology: STRIDE[електронний ресурс] / Claire Allen-Addy – Режим доступу до ресурсу: [https://www.iriusrisk.com/resources-blog/threat-modeling-methodology-stride] (дата звернення: 17.10.2024)
6. Tymoshchuk, D., Yasniy, O., Mytnyk, M., Zagorodna, N., Tymoshchuk, V., (2024). Detection and classification of DDoS flooding attacks by machine learning methods. CEUR Workshop Proceedings, 3842, pp. 184 - 195.
7. Lypa, B., Horyn, I., Zagorodna, N., Tymoshchuk, D., Lechachenko T., (2024). Comparison of feature extraction tools for network traffic data. CEUR Workshop Proceedings, 3896, pp. 1-11.
8. Kulchytskyi, T., Rezvorovych, K., Povalena, M., Dutchak, S., & Kramar, R. (2024). LEGAL REGULATION OF CYBERSECURITY IN THE CONTEXT OF THE DIGITAL TRANSFORMATION OF UKRAINIAN SOCIETY. Lex Humana (ISSN 2175-0947).

9. The improvement of web-application SDL process to prevent Insecure Design vulnerabilities / O. A. Revniuk та ін. Applied Aspects of Information Technology. 2024. Т. 7, № 2. С. 162–174. URL: https://doi.org/10.15276/aait.07.2024.12 (дата звернення: 19.11.2024).
10. Mishko O., Matiuk D., Derkach M. Security of remote iot system management by integrating firewall configuration into tunneled traffic. Scientific journal of the Ternopil national technical university. 2024. Т. 115, № 3. С. 122–129. URL: https://doi.org/10.33108/visnyk_tntu2024.03.122 (дата звернення: 02.12.2024).
11. Derkach, M., Skarga-Bandurova, I., Matiuk, D., & Zagorodna, N. (2022, December). Autonomous quadrotor flight stabilisation based on a complementary filter and a PID controller. In 2022 12th International Conference on Dependable Systems, Services and Technologies (DESSERT) (pp. 1-7). IEEE.
12. ТИМОЩУК, Д., & ЯЦКІВ, В. (2024). USING HYPERVISORS TO CREATE A CYBER POLYGON. MEASURING AND COMPUTING DEVICES IN TECHNOLOGICAL PROCESSES, (3), 52-56.
13. Wilson D. Mastering Node. Js - Best Practices to Server-Side Development. Independently Published, 2019.
14. ТИМОЩУК, Д., ЯЦКІВ, В., ТИМОЩУК, В., & ЯЦКІВ, Н. (2024). INTERACTIVE CYBERSECURITY TRAINING SYSTEM BASED ON SIMULATION ENVIRONMENTS. MEASURING AND COMPUTING DEVICES IN TECHNOLOGICAL PROCESSES, (4), 215-220.
15. Van Landuyt D., Joosen W. A descriptive study of assumptions in STRIDE security threat modeling. Software and Systems Modeling. 2021. URL: https://doi.org/10.1007/s10270-021-00941-7 (дата звернення: 19.11.2024).
16. Derkach, M., Lysak, V., Skarga-Bandurova, I., & Kotsiuba, I. (2019, September). Parking Guide Service for Large Urban Areas. In 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS) (Vol. 1, pp. 567-571). IEEE.

17. STRIDE-Based Cybersecurity Threat Modeling, Risk Assessment and Treatment of an In-Vehicle Infotainment System / P. Das et al. Vehicles. 2024. Vol. 6, no. 3. P. 1140–1163. URL: https://doi.org/10.3390/vehicles6030054 (дата звернення: 22.11.2024).
18. Muzh, V., & Lechachenko, T. (2024). Computer technologies as an object and source of forensic knowledge: challenges and prospects of development. Вісник Тернопільського національного технічного університету, 115(3), 17-22.
19. Bomba, A., Lechachenko, T., & Nazaruk, M. (2021). Modeling the Dynamics of "Knowledge Potentials" of Agents Including the Stakeholder Requests. In Advances in Computer Science for Engineering and Education IV (pp. 75-88). Springer International Publishing.
20. Tymoshchuk, D., & Yatskiv, V. (2024). Slowloris ddos detection and prevention in real-time. Collection of scientific papers «ΛΌГОΣ», (August 16, 2024; Oxford, UK), 171-176.
21. Про затвердження Правил технічної експлуатації електроустановок споживачів : Наказ М-ва палива та енергетики України від 25.07.2006 № 258 : станом на 21 лют. 2017 р. URL: https://zakon.rada.gov.ua/laws/show/z1143-06#Text (дата звернення: 09.12.2024).
22. Про затвердження Правил пожежної безпеки в Україні : Наказ М-ва внутр. справ України від 30.12.2014 № 1417 : станом на 14 серп. 2024 р. URL: https://zakon.rada.gov.ua/laws/show/z0252-15#Text (дата звернення: 09.12.2024).
23. Про основні засади забезпечення кібербезпеки України: Закон України від 05.10.2017 № 2163-VIII: станом на 28 черв. 2024 р. URL: https://zakon.rada.gov.ua/laws/show/2163-19#Text (дата звернення: 03.12.2024).
24. Про захист персональних даних: Закон України від 01.06.2010 № 2297-VI: станом на 27 квіт. 2024 р. URL: https://zakon.rada.gov.ua/laws/show/2297-17#Text (дата звернення: 06.12.2024).

Додаток А Публікація

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
імені ІВАНА ПУЛЮЯ
НАУКОВЕ ТОВАРИСТВО ім. ШЕВЧЕНКА

М А Т Е Р І А Л И

XII НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

18-19 грудня 2024 року

ТЕРНОПІЛЬ
2024

УДК 001
М34

ПРОГРАМНИЙ КОМІТЕТ

Голова: Приймак Микола – професор кафедри комп'ютерних систем та мереж, д.т.н., професо**р.**
Співголови: Марущак Павло – проректор з наукової роботи, докт. техн. наук, професор.
Баран Ігор – канд. техн. наук, доцент, декан факультету ФІС.
Науковий секретар: Надія Крива– старший викладач.

Члени: Василь Кривень - завідувач кафедри математичних методів в інженерії д.ф.-м.н., професор; Галина Осухівська – завідувач кафедри комп'ютерних систем та мереж, к.т.н., доцент; Микола Карпінський - професор кафедри кібербезпеки, д.т.н., професор; Жанна Баб'як - завідувач кафедри української та іноземних мов, к.пед. н., доцент; Ярослав Литвиненко – професор кафедри комп'ютерних наук, д.т.н., професор; Михайло Петрик - завідувач кафедри програмної іженерії, д.ф.-м.н., професор; Наталія Загородна – завідувач кафедри кібербезпеки, к.т.н., доцент.

ОРГАНІЗАЦІЙНИЙ КОМІТЕТ

Голова: Скоренький Юрій Любомирович – канд. техн. наук, доцент кафедри фізики.
Члени: доцент кафедри комп'ютерних наук, к.т.н. В. Никитюк; доцент кафедри програмної інженерії, к.т.н. Д. Михалик; доцент кафедри кібербезпеки, к.т.н. М. Стадник; доцент кафедри комп'ютерних систем та мереж, к.т.н. Є. Тиш; ст. викладач Л. Джиджора.

М34 Матеріали XII науково-технічної конфції «Інформаційні моделі, системи та технології» Тернопільського національного технічного університету імені Івана Пулюя, (Тернопіль, 18–19 грудня 2024 р.). – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2024.

Адреса оргкомітету: ТНТУ ім. І. Пулюя, м. Тернопіль, вул. Руська, 56, 46001,
тел. (0352) 52-41-33, факс (0352) 25-49-83.
E-mail: conffis2024@gmail.com
Редагування, оформлення та верстка: Надія Крива

СЕКЦІЇ КОНФЕРЕНЦІЇ, ЯКІ ПРЕДСТВЛЕНІ В ЗБІРНИКУ

- Математичне моделювання
- Інформаційні системи та технології, кібербезпека
- Комп'ютерні системи та мережі
- Програмна інженерія та моделювання складних розподілених систем
- Новітні фізико-технічні та освітні технології

В збірнику надруковано тези доповідей XII науково-технічної конференції «Інформаційні моделі, системи та технології» (Тернопіль, 18–19 грудня 2024 р.) за такими науковими напрямками: математичне моделювання; інформаційні системи та технології, кібербезпека; комп'ютерні системи та мережі; програмна інженерія та моделювання складних розподілених систем; новітні фізико-технічні та освітні технології.

Розрахований на науковців, викладачів та студентів вузів.

За зміст тез та дотримання норм академічної доброчесності відповідальність несе автор.

УДК 004.42

Сергій Шиндеровський, студент гр.СБм-62
Тернопільський національний технічний університет імені І. Пулюя, Україна

ДОСЛІДЖЕННЯ МЕХАНІЗМІВ ЗАХИСТУ ДЛЯ СЕРВЕРНОЇ ЧАСТИНИ НА NODE.JS

Serhii Shinderovskyi, student of gr.SBm-62

RESEARCH ON PROTECTION MECHANISMS FOR THE SERVER SIDE ON THE NODE.JS

Безпека веб-додатків є однією з ключових проблем сучасної кібербезпеки, враховуючи стрімке зростання кількості цифрових сервісів та обсягів оброблюваних ними даних. У зв'язку з розвитком технологій і збільшенням інтеграції веб-додатків у різні сфери життя – від банківської справи до соціальних мереж – забезпечення їх захищеності стає критично важливим. Будь-яка вразливість у веб-додатку може бути використана зловмисниками для крадіжки конфіденційної інформації, порушення роботи сервісу чи навіть компрометації цілих інформаційних систем [1].

Особливу увагу приділено вивченню загроз для серверної частини веб-додатків, які систематизовано за методологією STRIDE [3]. На основі цієї методології визначено шість основних категорій загроз: підробка, втручання в дані, відмова від авторства, розголошення інформації, відмова в обслуговуванні та підвищення привілеїв. Кожна із загроз детально проаналізована з огляду на її особливості, реальні приклади атак та сучасні методи запобігання [2].

Було проаналізувано основні загрози для серверної частини веб-додатків, використовуючи методологію STRIDE, яка охоплює підробку, втручання в дані, відмову від авторства, розголошення інформації, відмову в обслуговуванні та підвищення привілеїв. Для кожного типу загроз були запропоновані ефективні механізми захисту, такі як впровадження аутентифікації через JWT-токен та бібліотеки хешування, зокрема bcrypt, захист від XSS-атак, використання rate limiting та розмежування прав доступу на основі ролей. Також було додано журналювання дій користувачів, що забезпечує прозорість операцій на сервері. Інтеграція цих механізмів у серверну частину забезпечила багаторівневий підхід до мінімізації ризиків.

Практична значущість дослідження полягає у створенні універсального підходу до захисту серверної частини веб-додатків, який може бути використаний для реалізації надійних систем, стійких до сучасних загроз. Запропоновані механізми та методи дозволяють підвищити рівень безпеки даних, які передаються між клієнтом і сервером, зберігаються у базі даних або обробляються сервером.

У результаті проведеного дослідження доведено, що інтеграція заходів безпеки у процесі реалізації серверної частини дозволяє знизити ризики витоку даних, перехоплення запитів та несанкціонованого доступу до інформації. Це забезпечує надійність веб-додатків та підвищує їхню стійкість до кібератак. Таким чином, результати дослідження є цінним внеском у розвиток безпеки веб-технологій та можуть бути використані у подальших дослідженнях у сфері інформаційної безпеки.

Література:

1. What is a Web Application?[електронний ресурс] / AWS Amazon – URL: https://aws.amazon.com/what-is/web-application/

2. STRIDE Threat Modelling: What You Need to Know[електронний ресурс] / Alex Hewko – URL: https://www.softwaresecured.com/post/stride-threat-modelling

3. Threat Modeling Methodology: STRIDE[електронний ресурс] / Claire Allen-Addy – URL: https://www.iriusrisk.com/resources-blog/threat-modeling-methodology-stride

Додаток Б