Міністерство освіти і науки України
Харківський національний університет імені В.Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Спеціальність 125 «Кібербезпека»
Освітня програма «Кібербезпека»

В.о. зав. кафедрою КІСМіТ
Марина ЄСІНА
"Допущено до захисту"

« »_____2025р.

Пояснювальна записка

до кваліфікаційної роботи бакалавра

на тему: «Дослідження, аналіз та порівняння методів і засобів забезпечення безпеки сучасних веб-застосунків»

оцінка « _____ »

Голова ЕК
Мичуда Л.З.

Керівник: к.т.н. Єсіна М.В.

Рецензент: к.т.н., професор Качко О. Г.

Виконавець: студент групи КБ-41
Григоренко Є.Б.

Харків 2025

РЕФЕРАТ

Кваліфікаційна робота бакалавра містить 52 сторінок, 2 рисунки, 5 таблиць, 22 джерела за списком посилань.

Метою роботи є дослідження, аналіз та порівняння сучасних методів і засобів забезпечення безпеки веб-застосунків. Актуальність теми зумовлена зростанням кількості кібератак та уразливостей у сучасних веб-системах, що вимагає комплексного підходу до захисту.

У роботі використано методи аналізу наукової літератури, огляду міжнародних стандартів (OWASP, NIST, ISO/IEC 27001), а також практичне тестування інструментів безпеки. Розглянуто такі засоби, як Web Application Firewall (WAF), SIEM-системи, сканери вразливостей (OWASP ZAP, Burp Suite), методи автентифікації (JWT, OAuth, SAML), а також інструменти для шифрування даних.

Отримані результати дозволили порівняти ефективність різних засобів безпеки за критеріями: рівень захисту, продуктивність, масштабованість, складність впровадження. Новизна роботи полягає у проведенні практичного аналізу і тестування обраних засобів захисту з урахуванням сучасних загроз, таких як SQL-ін’єкції, XSS, CSRF, DDoS тощо.

Рекомендації стосуються впровадження багаторівневої системи захисту, використання сучасних інструментів моніторингу, інтеграції DevSecOps у процес розробки. Результати можуть бути корисні для розробників, фахівців з кібербезпеки та IT-компаній при побудові безпечних веб-застосунків.

Подальші дослідження можуть бути спрямовані на використання штучного інтелекту в системах виявлення загроз та побудову універсальних фреймворків для захисту веб-застосунків.

Ключові слова: БЕЗПЕКА ВЕБ-ЗАСТОСУНКІВ, КІБЕРЗАГРОЗИ, SQL-ІН’ЄКЦІЯ, XSS, CSRF, ШИФРУВАННЯ, АВТЕНТИФІКАЦІЯ, FIREWALL, OWASP, SIEM, ВРАЗЛИВІСТЬ, ВЕБ-ТЕСТУВАННЯ, DEVSECOPS.

ABSTRACT

The bachelor's qualification thesis comprises 52 pages, 2 figures, 5 tables, and 22 sources in the reference list.

The purpose of this work is to study, analyze, and compare modern methods and tools for securing web applications. The relevance of the topic is driven by the increasing number of cyberattacks and vulnerabilities in modern web systems, which require a comprehensive approach to protection.

The research applies methods of scientific literature analysis, review of international standards (OWASP, NIST, ISO/IEC 27001), and practical testing of security tools. The study considers tools such as Web Application Firewalls (WAF), SIEM systems, vulnerability scanners (OWASP ZAP, Burp Suite), authentication methods (JWT, OAuth, SAML), and data encryption technologies.

The obtained results enabled the comparison of the effectiveness of various security tools based on protection level, performance, scalability, and implementation complexity. The novelty of the work lies in the practical analysis and testing of selected security measures, considering current threats such as SQL injections, XSS, CSRF, DDoS, and others.

The recommendations include the implementation of a multi-layered security system, the use of modern monitoring tools, and the integration of DevSecOps into the development process. The findings can be useful for web developers, cybersecurity specialists, and IT companies in building secure web applications.

Future research may focus on the application of artificial intelligence in threat detection systems and the development of universal frameworks for web application security.

Keywords: WEB APPLICATION SECURITY, CYBER THREATS, SQL INJECTION, XSS, CSRF, ENCRYPTION, AUTHENTICATION, FIREWALL, OWASP, SIEM, VULNERABILITY, WEB TESTING, DEVSECOPS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ ТА СИМВОЛІВ

Скорочення	Розшифрування
OWASP	Open Web Application Security Project
XSS	Cross-Site Scripting
CSRF	Cross-Site Request Forgery
DDoS	Distributed Denial of Service
JWT	JSON Web Token
OAuth	Open Authorization
SAML	Security Assertion Markup Language
WAF	Web Application Firewall
HTTPS	HyperText Transfer Protocol Secure
TLS	Transport Layer Security
AES	Advanced Encryption Standard
RSA	Rivest–Shamir–Adleman
SIEM	Security Information and Event Management
SAST	Static Application Security Testing
DAST	Dynamic Application Security Testing
CI/CD	Continuous Integration / Continuous Delivery
DevSecOps	Development, Security, and Operations
CSP	Content Security Policy
API	Application Programming Interface
PCI DSS	Payment Card Industry Data Security Standard
SSDLC	Secure Software Development Lifecycle

ВСТУП

Сучасний світ значною мірою залежить від веб-технологій, які стали невід'ємною частиною повсякденного життя. Веб-застосунки використовуються в різних сферах: від електронної комерції, банкінгу та соціальних мереж до державних послуг та медичних систем. Однак, зі зростанням популярності веб-застосунків зростає і кількість кібератак, спрямованих на них. Згідно з дослідженнями, більшість сучасних веб-застосунків мають вразливості, які можуть бути використані зловмисниками для отримання несанкціонованого доступу до конфіденційних даних, порушення роботи систем або навіть повного їх знищення. Це робить питання забезпечення безпеки веб-застосунків однією з найактуальніших проблем сучасності.

Актуальність теми дослідження підкріплюється також тим, що кіберзагрози постійно еволюціонують. Зловмисники використовують все більш складні методи атак, що вимагає від розробників і фахівців з кібербезпеки постійного вдосконалення підходів до захисту веб-застосунків. Зокрема, такі атаки, як SQL Injection, XSS (міжсайтовий скриптинг), CSRF (підробка міжсайтових запитів) та інші, залишаються серйозною загрозою для веб-застосунків, незалежно від їх складності чи масштабу. Крім того, з появою нових технологій, таких як хмарні обчислення, мікросервіси та Інтернет речей (IoT), виникають нові виклики для забезпечення безпеки, що робить цю тему ще більш важливою для вивчення.

Метою даної роботи є дослідження, аналіз та порівняння сучасних методів і засобів забезпечення безпеки веб-застосунків. Для досягнення цієї мети необхідно вирішити низку завдань: вивчити основні загрози безпеці веб-застосунків, проаналізувати існуючі методи захисту, провести порівняльний аналіз їх ефективності та розробити рекомендації щодо впровадження найбільш ефективних рішень. Об'єктом дослідження є принцип роботи сучасних веб-застосунків, а предметом – методи та засоби забезпечення їх безпеки.

Методологія дослідження базується на комплексному підході, який включає аналіз наукової літератури, вивчення міжнародних стандартів і

рекомендацій (таких як OWASP Top 10, NIST, ISO 27001), а також практичне тестування обраних методів захисту. Для проведення порівняльного аналізу використовуються такі критерії, як ефективність, простота впровадження, вартість та сумісність з різними технологіями. Крім того, у роботі розглядаються реальні приклади вразливостей та способи їх усунення, що дозволяє отримати практично орієнтовані результати.

Структура роботи включає п’ять розділів. У першому розділі розглядаються основи безпеки веб-застосунків, включаючи основні загрози, класифікацію атак та принципи побудови безпечних систем. Другий розділ присвячений методам і засобам забезпечення безпеки, таким як автентифікація, шифрування, використання Web Application Firewall (WAF) та інструменти моніторингу. У третьому розділі проводиться порівняльний аналіз ефективності різних методів захисту, а також практичне тестування обраних рішень. Четвертий розділ містить рекомендації щодо забезпечення безпеки веб-застосунків, включаючи кращі практики, автоматизовані засоби перевірки та впровадження DevSecOps. П’ятий розділ присвячений оформленню результатів дослідження, узагальненню висновків та підготовці наукових рекомендацій.

Результати даної роботи можуть бути корисними для розробників веб-застосунків, фахівців з кібербезпеки та студентів, які вивчають сучасні технології захисту інформації. Робота також може стати основою для подальших досліджень у цій галузі, зокрема щодо застосування нових технологій, таких як штучний інтелект та машинне навчання, для покращення безпеки веб-застосунків. У перспективі подальші дослідження можуть бути спрямовані на розробку універсальних фреймворків для забезпечення безпеки веб-застосунків, які враховуватимуть специфіку різних галузей та типів застосунків.

Дана робота є актуальною та важливою для розуміння сучасних викликів у сфері безпеки веб-застосунків, а також для пошуку ефективних шляхів їх вирішення. Результати дослідження можуть сприяти підвищенню рівня безпеки веб-застосунків, що є важливим кроком у забезпеченні захисту інформації в умовах сучасного цифрового світу.

1 ОСНОВИ БЕЗПЕКИ ВЕБ-ЗАСТОСУНКІВ

1.1. Основні загрози безпеці веб-застосунків

Сучасні веб-застосунки є одними з найбільш вразливих елементів інформаційної інфраструктури через їхню складність, відкритість для користувачів та постійну взаємодію з зовнішніми системами. Основні загрози безпеці веб-застосунків пов’язані з можливістю зловмисників отримати несанкціонований доступ до конфіденційних даних, порушити роботу системи або навіть повністю її знищити. Згідно з дослідженнями, проведеними OWASP (Open Web Application Security Project), більшість веб-застосунків мають вразливості, які можуть бути використані для атак [1].

Однією з найпоширеніших загроз є SQL Injection (SQL-ін’єкція). Ця атака дозволяє зловмисникам вводити SQL-запити через поля введення даних, що може призвести до викрадення, зміни або видалення даних з бази даних. Наприклад, у 2017 році компанія Equifax стала жертвою масштабної атаки через SQL Injection, що призвело до витоку даних 147 мільйонів користувачів [2].

Іншою серйозною загрозою є XSS (Cross-Site Scripting). Це одна з найпоширеніших атак на веб-застосунки, яка дозволяє зловмиснику вставляти шкідливі скрипти у веб-сторінки, що надалі виконуються у браузері користувача. Це може призвести до компрометації облікових записів, викрадення сесій або перенаправлення користувача на шкідливі ресурси. Згідно зі звітом компанії Veracode, понад 60% веб-застосунків містять XSS-вразливості [3].

CSRF (Cross-Site Request Forgery) – це ще одна поширена загроза, яка дозволяє зловмисникам виконувати дії від імені авторизованого користувача без його відома. Наприклад, якщо користувач авторизований у банківському застосунку, зловмисник може ініціювати переказ коштів на свій рахунок [4].

Також варто згадати про DDoS-атаки (Distributed Denial of Service), які спрямовані на перевантаження серверів веб-застосунків, що призводить до їхньої недоступності для законних користувачів.

Крім того, веб-застосунки часто стають жертвами ін’єкцій команд, несанкціонованого доступу до файлів та неправильної конфігурації безпеки.

Наприклад, неправильне налаштування серверів або використання застарілих версій програмного забезпечення може відкрити двері для зловмисників [6].

Для кращого розуміння класифікація загроз за типами та наслідками наведена у таблиці 1.1.

Таблиця 1.1 – Класифікація основних загроз безпеці веб-застосунків

Тип загрози	Опис	Наслідки
SQL-ін’єкції	Введення SQL-запитів через поля введення даних	Викрадення, зміна або видалення даних з бази даних
XSS (Cross-Site Scripting)	Впровадження скриптів на сторінки веб-застосунків	Викрадення сесій, перенаправлення на шкідливі сайти, крадіжка даних
CSRF (Cross-Site Request Forgery)	Виконання дій від імені авторизованого користувача	Несанкціоновані транзакції, зміна даних
DDoS-атаки	Перевантаження серверів великою кількістю запитів	Недоступність веб-застосунку для користувачів
Ін’єкції команд	Виконання команд на сервері через вразливі поля введення	Отримання повного контролю над сервером
Неправильна конфігурація	Помилки в налаштуванні серверів або програмного забезпечення	Несанкціонований доступ до системи або даних

Основні загрози безпеці веб-застосунків включають як технічні вразливості, так і помилки в конфігурації та управлінні. Для ефективного захисту необхідно враховувати всі ці аспекти та впроваджувати комплексні підходи до забезпечення безпеки [7].

1.2. Класифікація атак на веб-застосунки

Веб-застосунки є одними з найбільш популярних цілей для кібератак через їхню доступність та складність архітектури. Атаки на веб-застосунки можуть бути класифіковані за різними критеріями, такими як тип вразливості, спосіб реалізації атаки, ціль атаки та її наслідки. Розуміння цих класифікацій допомагає розробникам та фахівцям з кібербезпеки ефективніше протидіяти загрозам. Нижче наведено детальний огляд основних типів атак на веб-застосунки.

Атаки на клієнтську частину веб-застосунків спрямовані на експлуатацію вразливостей у браузері користувача або на стороні клієнта. До таких атак належать:

— XSS (Cross-Site Scripting): ця атака дозволяє зловмисникам впроваджувати скрипти на сторінки веб-застосунків, які виконуються в браузері користувача. XSS поділяється на три основні типи:

- Reflected XSS: зловмисник вводить скрипт через URL або форму, який відображається на сторінці без належного екранування.
- Stored XSS: шкідливий скрипт зберігається на сервері (наприклад, у базі даних) і виконується кожного разу, коли користувач відкриває певну сторінку.
- DOM-based XSS: вразливість виникає через неправильну обробку даних у DOM (Document Object Model) браузера [1].

— Clickjacking: ця атака полягає в тому, що зловмисник створює невидимий шар поверх легального інтерфейсу веб-застосунку, що змушує користувача виконувати дії, які він не планував (наприклад, натискання кнопок) [2].

Атаки на серверну частину веб-застосунків спрямовані на експлуатацію вразливостей у серверній логіці, базі даних або інфраструктурі. До таких атак належать:

— SQL-ін’єкції: ця атака дозволяє зловмисникам вводити SQL-запити через поля введення даних, що може призвести до викрадення, зміни або видалення даних з бази даних. Наприклад, зловмисник може отримати доступ до конфіденційної інформації, такої як паролі або платіжні дані [3].

— Command Injection: ця атака дозволяє зловмисникам виконувати команди на сервері через вразливі поля введення. Наприклад, зловмисник може отримати доступ до файлової системи сервера або запустити шкідливий код [4].

— File Inclusion: ця атака дозволяє зловмисникам включати файли на сервері через вразливі параметри веб-застосунку. Наприклад, зловмисник може завантажити та виконати шкідливий скрипт на сервері [5].

Атаки на мережу спрямовані на експлуатацію вразливостей у мережевій інфраструктурі веб-застосунків. До таких атак належать:

— DDoS (Distributed Denial of Service): ця атака полягає в перевантаженні серверів великою кількістю запитів, що призводить до їхньої недоступності для законних користувачів. Наприклад, у 2021 році компанія Cloudflare зафіксувала рекордну DDoS-атаку з потужністю 17,2 мільйона запитів за секунду [6].

— Man-in-the-Middle (MitM): ця атака дозволяє зловмиснику перехоплювати та змінювати дані, що передаються між клієнтом і сервером. Наприклад, зловмисник може перехопити конфіденційну інформацію, таку як логіни та паролі [7].

Атаки на автентифікацію та авторизацію спрямовані на обхід механізмів перевірки ідентифікації користувачів. До таких атак належать:

— Brute Force: ця атака полягає в переборі можливих комбінацій логінів та паролів для отримання доступу до системи. Наприклад, зловмисник може використовувати автоматизовані інструменти для підбору паролів [8].

— Session Hijacking: ця атака дозволяє зловмиснику викрасти сесію користувача та отримати доступ до системи від його імені. Наприклад, зловмисник може перехопити кукі сесії через незахищене з’єднання [9].

Атаки на конфігурацію спрямовані на експлуатацію помилок у налаштуванні серверів або програмного забезпечення. До таких атак належать:

— Misconfiguration: ця атака використовує помилки в налаштуванні серверів, таких як відкриті порти, застарілі версії програмного забезпечення або неправильні права доступу. Наприклад, зловмисник може отримати доступ до конфіденційних даних через відкриту директорію на сервері [10].

— Directory Traversal: ця атака дозволяє зловмиснику отримати доступ до файлів і директорій, які не повинні бути доступні через веб-інтерфейс. Наприклад, зловмисник може отримати доступ до файлу конфігурації сервера [11].

Класифікація атак на веб-застосунки, що наведена у таблиці 1.2, дозволяє краще зрозуміти природу загроз та розробити ефективні механізми захисту. Кожен тип атаки вимагає специфічних підходів до профілактики та усунення вразливостей [12].

Таблиця 1.2 – Класифікація атак на веб-застосунки

Тип атаки	Опис	Приклад
XSS (Cross-Site Scripting)	Впровадження скриптів на сторінки веб-застосунків	Викрадення сесій, перенаправлення на шкідливі сайти
SQL-ін’єкція	Введення SQL-запитів через поля введення даних	Викрадення, зміна або видалення даних з бази даних
DDoS (Distributed Denial of Service)	Перевантаження серверів великою кількістю запитів	Недоступність веб-застосунку для користувачів
Brute Force	Перебір можливих комбінацій логінів та паролів	Отримання доступу до системи через слабкі паролі
Misconfiguration	Експлуатація помилок у налаштуванні серверів	Отримання доступу до конфіденційних даних через відкриті порти

1.3. Принципи побудови безпечних веб-застосунків

Побудова безпечних веб-застосунків є критично важливим аспектом сучасної розробки програмного забезпечення, оскільки веб-застосунки часто стають мішенню для кібератак через свою доступність та складність архітектури. Забезпечення безпеки веб-застосунків вимагає комплексного підходу, який охоплює всі етапи життєвого циклу розробки – від проєктування до впровадження та підтримки. Основні принципи побудови безпечних веб-застосунків включають мінімізацію привілеїв, захист на всіх рівнях, валідацію та санацію вхідних даних, шифрування конфіденційної інформації, регулярне оновлення програмного забезпечення, журналювання та моніторинг, інтеграцію безпеки на етапі розробки та регулярне тестування на вразливості.

Одним із ключових принципів є мінімізація привілеїв. Цей принцип передбачає, що кожен компонент системи, чи то користувач, процес або модуль, повинен мати лише ті права доступу, які необхідні для виконання його функцій. Наприклад, якщо веб-застосунок взаємодіє з базою даних, він повинен мати доступ лише до тих таблиць і операцій, які безпосередньо пов’язані з його

роботою. Це значно зменшує ризик того, що зловмисник, який отримав доступ до одного компонента, зможе поширити свій вплив на всю систему. Мінімізація привілеїв також включає обмеження прав адміністраторів та використання принципу найменших привілеїв для всіх облікових записів [1].

Іншим важливим принципом є захист на всіх рівнях (Defense in Depth). Цей підхід передбачає створення багаторівневої системи захисту, де навіть якщо один рівень буде порушено, інші рівні зможуть запобігти подальшій експлуатації вразливості. Наприклад, захист може включати мережевий рівень (використання фаєрволів, VPN), рівень сервера (регулярне оновлення програмного забезпечення, налаштування прав доступу) та рівень застосунку (валідація вхідних даних, шифрування конфіденційної інформації). Такий підхід дозволяє створити надійну систему, яка може протистояти різноманітним загрозам [2].

Валідація та санація вхідних даних є ще одним критично важливим принципом. Валідація передбачає перевірку вхідних даних на відповідність очікуваним форматам (наприклад, перевірка, чи є введений текст дійсно електронною поштою), тоді як санація полягає у видаленні або екрануванні потенційно небезпечних символів (наприклад, лапок або символів HTML-тегів). Ці механізми допомагають запобігти атакам, таким як SQL-ін’єкції, XSS та Command Injection, які експлуатують недоліки у обробці вхідних даних. Наприклад, якщо користувач вводить текст у поле коментаря, веб-застосунок повинен автоматично екранувати спеціальні символи, щоб запобігти виконанню скриптів [3].

Шифрування даних є основним засобом захисту конфіденційної інформації, такої як паролі, платіжні дані та особиста інформація користувачів. Веб-застосунки повинні використовувати сучасні алгоритми шифрування, такі як AES для шифрування даних та RSA для шифрування ключів. Крім того, всі дані, що передаються між клієнтом і сервером, повинні бути захищені за допомогою протоколу HTTPS, який забезпечує шифрування трафіку за допомогою TLS/SSL. Це допомагає запобігти перехопленню даних зловмисниками, які можуть знаходитися в одній мережі з користувачем [4].

Регулярне оновлення та патчинг програмного забезпечення є обов’язковим принципом для підтримки безпеки веб-застосунків. Багато атак експлуатують вразливості, які вже були виправлені в нових версіях програмного забезпечення. Наприклад, вразливості в популярних фреймворках, таких як Django або Spring, часто виправляються у нових релізах. Тому розробники повинні оперативно впроваджувати оновлення та патчі, щоб мінімізувати ризики. Це також стосується операційних систем, серверного програмного забезпечення та бібліотек, які використовуються в проєкті [5].

Журналювання та моніторинг дозволяють оперативно виявляти та реагувати на потенційні загрози. Веб-застосунки повинні вести детальні логи всіх критичних подій, таких як спроби автентифікації, зміни конфігурації та підозріла активність. Ці логи повинні аналізуватися за допомогою спеціалізованих інструментів моніторингу, таких як Splunk або ELK Stack, що дозволяє швидко виявляти аномалії та вживати заходів для їх усунення. Наприклад, якщо система фіксує багато невдалих спроб входу, це може свідчити про спробу підбору пароля, і адміністратори повинні негайно вжити заходів для блокування атаки [6].

Інтеграція безпеки на етапі розробки (Secure by Design) – це принцип, який передбачає, що безпека повинна бути врахована на всіх етапах життєвого циклу веб-застосунку, починаючи з проєктування. Це включає використання практик, таких як Threat Modeling (моделювання загроз) та Secure Coding Guidelines (рекомендації з написання безпечного коду). Наприклад, перед початком розробки команда повинна провести аналіз потенційних загроз і визначити, які механізми захисту необхідно впровадити. Це дозволяє мінімізувати ризики на ранніх етапах та уникнути дорогих виправлень пізніше [7].

Регулярне тестування безпеки є обов’язковим етапом для підтримки безпеки веб-застосунків. До методів тестування належать статичний аналіз коду (SAST), який дозволяє виявляти вразливості на етапі написання коду, динамічний аналіз безпеки (DAST), який тестує працюючий застосунок на предмет вразливостей, та пентестування, яке імітує атаки зловмисників для

виявлення слабких місць у системі. Регулярне тестування допомагає виявляти та усувати вразливості до того, як вони будуть використані зловмисниками [8].

Побудова безпечних веб-застосунків вимагає дотримання низки ключових принципів, які забезпечують захист на всіх етапах життєвого циклу застосунку. Ці принципи допомагають мінімізувати ризики кібератак, захистити конфіденційну інформацію користувачів та забезпечити стабільну роботу веб-застосунків у довгостроковій перспективі [9].

1.4 Огляд стандартів і рекомендацій щодо безпеки веб-застосунків

Безпека веб-застосунків є критично важливим аспектом сучасного розвитку інформаційних технологій, і для її забезпечення існує низка міжнародних стандартів та рекомендацій. Ці стандарти розроблені для того, щоб допомогти організаціям та розробникам створювати безпечні веб-застосунки, мінімізуючи ризики кібератак та забезпечуючи захист конфіденційної інформації. Серед найбільш відомих та широко використовуваних стандартів і рекомендацій можна виділити OWASP Top Ten, NIST Cybersecurity Framework, ISO/IEC 27001 та PCI DSS.

OWASP (Open Web Application Security Project) – це міжнародна некомерційна організація, яка займається дослідженням та популяризацією питань безпеки веб-застосунків. Одним із найвідоміших документів OWASP є OWASP Top Ten, який містить список десяти найбільш критичних ризиків для веб-застосунків. Цей документ регулярно оновлюється, щоб враховувати нові загрози та тенденції в галузі кібербезпеки. До основних ризиків, які описані в OWASP Top Ten, належать:

— Ін’єкції (Injection): атаки, такі як SQL Injection, які дозволяють зловмисникам виконувати команди на стороні сервера та отримувати доступ до конфіденційних даних.

— Несправна автентифікація (Broken Authentication): вразливості, пов’язані з механізмами входу та управління сесіями.

— Чутливі дані (Sensitive Data Exposure): неналежний захист конфіденційної інформації, такої як паролі або платіжні дані.

— XML External Entities (XXE): вразливості, пов’язані з обробкою XML-файлів.

— Несправний контроль доступу (Broken Access Control): проблеми з обмеженням доступу до ресурсів [1].

OWASP Top Ten є основним орієнтиром для розробників та фахівців з кібербезпеки, які прагнуть забезпечити безпеку своїх веб-застосунків.

NIST (National Institute of Standards and Technology) – це американська організація, яка розробляє стандарти та рекомендації у сфері інформаційної безпеки. NIST Cybersecurity Framework (CSF) є одним із найважливіших документів, який надає рекомендації щодо захисту інформаційних систем, включаючи веб-застосунки. CSF складається з п’яти основних функцій:

— Ідентифікація (Identify): визначення активів, ризиків та вимог до безпеки.

— Захист (Protect): впровадження заходів для захисту систем та даних.

— Виявлення (Detect): моніторинг та виявлення кібератак.

— Реагування (Respond): планування та впровадження заходів для реагування на інциденти.

— Відновлення (Recover): відновлення нормальної роботи після інциденту [2].

NIST CSF широко використовується як урядовими організаціями, так і приватним сектором для побудови ефективних систем кібербезпеки.

ISO/IEC 27001 – це міжнародний стандарт, який визначає вимоги до систем управління інформаційною безпекою (СУІБ). Цей стандарт передбачає створення, впровадження, підтримку та постійне вдосконалення системи управління безпекою в організації. ISO/IEC 27001 охоплює всі аспекти інформаційної безпеки, включаючи захист веб-застосунків. До ключових елементів стандарту належать:

— Оцінка ризиків: визначення потенційних загроз та вразливостей.

— Політики безпеки: розробка та впровадження політик, які регулюють доступ до інформації.

— Управління інцидентами: планування заходів для реагування на кібератаки.

— Аудит та моніторинг: регулярна перевірка ефективності заходів безпеки [3].

Сертифікація за стандартом ISO/IEC 27001 є важливим кроком для організацій, які прагнуть довести свою відповідність міжнародним вимогам до інформаційної безпеки.

PCI DSS (Payment Card Industry Data Security Standard) – це стандарт, розроблений для захисту платіжних даних. Він є обов'язковим для всіх організацій, які обробляють, зберігають або передають дані платіжних карток. PCI DSS включає 12 основних вимог, таких як:

— Захист мережі: встановлення та підтримка мережевих фаєрволів.

— Захист даних: шифрування конфіденційної інформації під час передачі та зберігання.

— Управління вразливостями: регулярне оновлення програмного забезпечення та сканування на вразливості.

— Контроль доступу: обмеження доступу до даних на основі принципу найменших привілеїв [4].

PCI DSS є критично важливим для організацій, які працюють у сфері електронної комерції або обробляють платіжні дані.

Крім вищезазначених стандартів, існує низка інших документів, які можуть бути корисними для забезпечення безпеки веб-застосунків. Наприклад:

— ISO/IEC 27034: стандарт, який спеціалізується на безпеці застосунків, включаючи веб-застосунки.

— GDPR (General Data Protection Regulation): Регламент Європейського Союзу, який регулює захист персональних даних.

— CIS Controls: набір практик, розроблений Center for Internet Security, який допомагає організаціям захистити свої системи від кібератак [5].

Стандарти та рекомендації щодо безпеки веб-застосунків є важливим інструментом для розробників та організацій, які прагнуть забезпечити захист

своїх систем та даних. Вони надають чіткі вказівки щодо впровадження ефективних заходів безпеки, що дозволяє мінімізувати ризики кібератак та забезпечити довіру користувачів [6].

Крім того, впровадження таких стандартів має позитивний вплив і на репутацію компаній: дотримання вимог міжнародних стандартів свідчить про відповідальне ставлення до безпеки та сприяє зміцненню довіри з боку клієнтів і партнерів. Це особливо важливо в умовах зростаючих вимог до захисту персональних даних та фінансової інформації. Також стандарти сприяють формуванню культури безпеки в організації, що забезпечує краще розуміння загроз і відповідальнішу поведінку працівників.

Отже, впровадження міжнародних стандартів безпеки не лише відповідає технічним і нормативним вимогам, а й створює фундамент для сталого та надійного розвитку цифрових рішень.

2 МЕТОДИ ТА ЗАСОБИ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ ВЕБ-ЗАСТОСУНКІВ

2.1 Методи автентифікації та авторизації (JWT, OAuth, SAML тощо)

Автентифікація та авторизація є ключовими елементами безпеки веб-застосунків, які забезпечують перевірку ідентифікації користувачів та контроль доступу до ресурсів. У сучасних веб-застосунках використовуються різні методи та технології для реалізації цих процесів, такі як JSON Web Token (JWT), OAuth, Security Assertion Markup Language (SAML) та інші. Кожен із цих методів має свої переваги, недоліки та сфери застосування, і правильний вибір дозволяє забезпечити ефективний захист веб-застосунків.

JSON Web Token (JWT) – це відкритий стандарт (RFC 7519), який визначає спосіб передачі інформації між сторонами у вигляді JSON-об’єкта. JWT використовується для автентифікації та авторизації, а також для обміну даними між клієнтом і сервером. Токен складається з трьох частин: заголовка (Header), який містить інформацію про тип токена та алгоритм шифрування; корисного навантаження (Payload), яке містить дані, такі як ідентифікатор користувача, ролі та термін дії токена; та підпису (Signature), який використовується для перевірки цілісності токена. Основною перевагою JWT є його статусність – токен містить усю необхідну інформацію для автентифікації, що дозволяє серверу не зберігати стан сесії. Це робить JWT ідеальним для використання в розподілених системах, де токен може бути перевірений будь-яким сервером, який має відповідний ключ. Однак JWT має і недоліки, такі як більший розмір порівняно з іншими типами токенів, що може вплинути на продуктивність, а також складність відкликання токенів до закінчення терміну їх дії [1].

OAuth – це відкритий стандарт авторизації, який дозволяє стороннім застосункам отримувати обмежений доступ до ресурсів користувача без необхідності передавати їхні облікові дані. OAuth 2.0 є найпоширенішою версією стандарту і використовується для авторизації в таких сервісах, як Google, Facebook та GitHub. Основними компонентами OAuth 2.0 є клієнт (застосунок, який запитує доступ до ресурсів), ресурсний сервер (сервер, який зберігає ресурси користувача), сервер авторизації (сервер, який видає токени доступу) та

власник ресурсу (користувач, який надає доступ до своїх ресурсів). OAuth підтримує різні типи потоків, такі як Authorization Code Flow, Implicit Flow, Client Credentials Flow та Resource Owner Password Credentials Flow, кожен із яких призначений для різних сценаріїв використання. Основною перевагою OAuth є його безпека – стандарт не вимагає передачі облікових даних користувача стороннім застосункам, що значно зменшує ризики витоків даних. Однак OAuth має і недоліки, такі як складність реалізації та ризики безпеки при неправильному налаштуванні [2].

SAML (Security Assertion Markup Language) – це стандарт для обміну даними автентифікації та авторизації між сторонами, зокрема між постачальником послуг (Service Provider) та постачальником ідентифікації (Identity Provider). SAML використовується переважно в корпоративних середовищах для реалізації Single Sign-On (SSO), що дозволяє користувачам входити в кілька застосунків за допомогою одного набору облікових даних. Основними компонентами SAML є Assertion (XML-документ, який містить інформацію про автентифікацію користувача), Protocol (визначає, як запити та відповіді передаються між сторонами) та Binding (визначає, як SAML-повідомлення передаються через HTTP, SOAP тощо). Основною перевагою SAML є підтримка SSO, що значно спрощує управління доступом у великих організаціях. Однак SAML має і недоліки, такі як складність реалізації через використання XML та обмежена підтримка в мобільних застосунках [3].

Крім JWT, OAuth та SAML, існують інші методи автентифікації та авторизації, які використовуються в веб-застосунках. Наприклад, OpenID Connect є розширенням OAuth 2.0, яке додає підтримку автентифікації, що робить його ідеальним для систем, де необхідно забезпечити як авторизацію, так і автентифікацію. Basic Authentication – це простий метод, який використовує логін та пароль, але він менш безпечний через відсутність шифрування. API Keys використовуються для авторизації API, але вони менш гнучкі порівняно з OAuth, оскільки не підтримують складні сценарії авторизації [4].

Вибір методу автентифікації та авторизації залежить від конкретних вимог веб-застосунку, таких як масштабованість, безпека та інтеграція з іншими системами. Кожен із розглянутих методів має свої переваги та недоліки, що наведені у таблиці 2.1, і правильний вибір дозволяє забезпечити ефективний захист веб-застосунків. Наприклад, JWT ідеально підходить для розподілених систем, OAuth – для інтеграції зі сторонніми сервісами, а SAML – для корпоративних середовищ з підтримкою SSO [5].

Таблиця 2.1 – Переваги та недоліки поширених методів автентифікації [21]

Метод	Переваги	Недоліки
JWT	Бездержавний, може переносити більше даних, може підтримувати кілька доменів або служб, може бути перевірений будь-ким, хто має доступ до ключа.	Вразливий до *XSS*-атак, має фіксований термін дії, не може бути легко відкликаний або оновлений.
OAuth	Дозволяє користувачеві надавати доступ до своїх ресурсів або інформації з одного сайту іншому сайту без обміну обліковими даними, може використовувати різні типи токенів, такі як JWT-, API-токени або SAML-твердження.	Складний і вимагає участі багатьох сторін та взаємодії, створює певні ризики для безпеки, такі як фішингові атаки, витік токенів або їх повторне відтворення.
SAML	Дозволяє користувачеві входити на один сайт і отримувати доступ до іншого сайту без повторного введення облікових даних, використовує твердження, що містять інформацію про особу користувача, атрибути або рішення про авторизацію, може бути підписана й зашифрована постачальником ідентифікації та перевірена постачальником послуг.	Складний і вимагає оброблення та розбору XML, має деякі проблеми з продуктивністю через розмір і кількість повідомлень, що беруть участь.
OpenID	Дозволяє користувачеві входити на один сайт і застосовувати свій ідентифікатор для доступу до іншого сайту без створення облікового запису або повторного введення облікових даних, може використовувати різні типи токенів для подання ідентифікаційної інформації.	Складний, вимагає участі багатьох сторін і взаємодії, створює певні ризики для безпеки, такі як фішингові атаки, підроблення ідентифікаторів або витік токенів.

2.2 Захист від атак типу XSS, SQL Injection, CSRF

Захист веб-застосунків від атак типу XSS (Cross-Site Scripting), SQL Injection та CSRF (Cross-Site Request Forgery) є критично важливим для забезпечення безпеки даних та стабільної роботи системи. Ці атаки є одними з найпоширеніших і найнебезпечніших, оскільки вони експлуатують вразливості у веб-застосунках, що може призвести до витоків конфіденційної інформації, порушення функціонування системи або навіть повного захоплення контролю над застосунком. Нижче розглянуто основні методи захисту від цих атак.

XSS (Cross-Site Scripting) – одна з найбільш поширених вразливостей веб-застосунків. Для ефективного протистояння цій загрозі необхідно реалізувати низку технічних заходів, спрямованих на зменшення ризику впровадження та виконання шкідливого коду в браузері користувача. Нижче розглянуто основні методи захисту від XSS:

1) Екранування вхідних даних: Усі дані, які відображаються на сторінці, повинні бути екрановані для запобігання виконання скриптів. Наприклад, спеціальні символи, такі як <, >, &, повинні бути замінені на HTML-сутності (<, >, &).

2) Валідація вхідних даних: Усі дані, які надходять від користувача, повинні перевірятися на відповідність очікуваним форматам (наприклад, електронна пошта, телефонний номер).

3) Використання Content Security Policy (CSP): CSP дозволяє обмежити джерела, з яких можна завантажувати скрипти, стилі та інші ресурси. Наприклад, CSP може заборонити виконання вбудованих скриптів, що значно зменшує ризик XSS.

4) Використання бібліотек для санації даних: Такі бібліотеки, як DOMPurify для JavaScript, автоматично видаляють небезпечні елементи з HTML-коду [1].

SQL-ін’єкція – це атака, яка дозволяє зловмисникам вводити SQL-запити через поля введення даних, що може призвести до викрадення, зміни або

видалення даних з бази даних. Наприклад, зловмисник може отримати доступ до конфіденційної інформації, такої як паролі або платіжні дані.

Методи захисту від SQL-ін’єкцій:

1) Використання параметризованих запитів (Prepared Statements): Параметризовані запити дозволяють відокремити дані від коду SQL, що робить неможливим введення команд. Наприклад, замість того, щоб вставляти дані безпосередньо в SQL-запит, використовуються параметри, які автоматично екрануються.

2) ORM (Object-Relational Mapping): ORM-бібліотеки, такі як Hibernate або Entity Framework, автоматично генерують безпечні SQL-запити, що значно зменшує ризик SQL-ін’єкції.

3) Валідація та санація вхідних даних: Усі дані, які надходять від користувача, повинні перевірятися на відповідність очікуваним форматам та очищатися від небезпечних символів.

4) Обмеження прав доступу до бази даних: Користувач веб-застосунку повинен мати лише ті права доступу до бази даних, які необхідні для його роботи. Наприклад, якщо застосунок не потребує зміни даних, він повинен мати доступ лише для читання [2].

CSRF (Cross-Site Request Forgery) – це атака, яка дозволяє зловмисникам виконувати дії від імені авторизованого користувача без його відома. Наприклад, якщо користувач авторизований у банківському застосунку, зловмисник може ініціювати переказ коштів на свій рахунок.

Методи захисту від CSRF:

1) Використання CSRF-токенів: CSRF-токен – це унікальний випадковий рядок, який генерується сервером і додається до кожного запиту, що змінює стан системи (наприклад, POST-запити). Сервер перевіряє наявність та валідність токена перед обробкою запиту.

2) SameSite Cookies: атрибут SameSite для cookies дозволяє обмежити відправку cookies лише для запитів, які надходять з того самого домену. Це запобігає використанню cookies у CSRF-атаках.

3) Перевірка заголовка Origin або Referer: сервер може перевіряти заголовки Origin або Referer для підтвердження того, що запит надійшов з легального джерела.

4) Додаткові шари авторизації: для критичних операцій (наприклад, переказ коштів) можна вимагати додаткову авторизацію, таку як введення одноразового пароля (OTP) [3].

Крім специфічних методів захисту від XSS, SQL-ін’єкцій та CSRF, існують загальні рекомендації, які допомагають підвищити безпеку веб-застосунків:

1) Регулярне оновлення програмного забезпечення: усі компоненти веб-застосунку, включаючи фреймворки, бібліотеки та серверне програмне забезпечення, повинні регулярно оновлюватися для усунення вразливостей.

2) Використання Web Application Firewall (WAF): WAF дозволяє блокувати запити на рівні мережі, що допомагає запобігти багатьом типам атак, включаючи XSS та SQL-ін’єкції.

3) Регулярне тестування на вразливості: веб-застосунки повинні регулярно тестуватися на наявність вразливостей за допомогою інструментів, таких як OWASP ZAP або Burp Suite [4].

Захист від атак типу XSS, SQL-ін’єкцій та CSRF вимагає комплексного підходу, який включає як технічні засоби (екранування даних, параметризовані запити, CSRF-токени), так і організаційні заходи (регулярне оновлення, тестування). Дотримання цих рекомендацій дозволяє значно зменшити ризики кібератак та забезпечити безпеку веб-застосунків [5].

2.3 Шифрування даних та безпечне зберігання інформації

Шифрування даних є одним із ключових методів забезпечення безпеки веб-застосунків, оскільки воно дозволяє захистити інформацію від несанкціонованого доступу навіть у разі компрометації системи. Дані можуть шифруватися як у стані спокою (at rest), так і під час передачі (in transit).

Для шифрування даних у стані спокою зазвичай використовуються алгоритми AES (Advanced Encryption Standard), RSA (Rivest-Shamir-Adleman) або ECC (Elliptic Curve Cryptography). AES є стандартом для симетричного

шифрування і широко застосовується для збереження конфіденційної інформації в базах даних. RSA та ECC, які належать до асиметричних алгоритмів, використовуються для безпечного зберігання ключів та шифрування меншого обсягу даних.

При збереженні паролів необхідно використовувати криптографічні геш-функції, такі як bcrypt, Argon2 або PBKDF2, які включають механізм сольового гешування (salting), що запобігає атакам типу «rainbow table». Просте гешування (наприклад, MD5 або SHA-1) не є безпечним через його вразливість до швидких атак перебором (brute force).

Для забезпечення безпеки даних під час передавання використовується TLS (Transport Layer Security), який шифрує трафік між клієнтом і сервером. TLS 1.2 і 1.3 є стандартами, що забезпечують шифрування переданих даних за допомогою алгоритмів AES-GCM, ChaCha20-Poly1305 та інших стійких криптографічних методів. Використання застарілих версій, таких як SSL або TLS 1.0, є небезпечним через вразливості, наприклад, BEAST та POODLE.

Крім шифрування, важливим аспектом є безпечне зберігання ключів. Ключі не повинні зберігатися у відкритому вигляді в коді або конфігураційних файлах. Для їх захисту використовуються апаратні модулі безпеки (HSM), сховища секретів (наприклад, AWS Secrets Manager, HashiCorp Vault) або механізми захисту на рівні операційної системи, такі як Windows DPAPI або Linux Secure Enclave.

Також необхідно застосовувати контроль доступу до конфіденційних даних, зокрема принцип мінімальних привілеїв (least privilege), щоб користувачі та сервіси мали доступ лише до необхідної інформації. Регулярний аудит безпеки та моніторинг потенційних загроз допомагають виявляти можливі порушення безпеки та запобігати витоку даних.

Використання сучасних методів шифрування, безпечного зберігання паролів, захищених каналів зв’язку та механізмів керування доступом є важливими елементами забезпечення конфіденційності та цілісності даних у веб-застосунках.

2.4 Використання Web Application Firewall (WAF) та інших засобів захисту

Web Application Firewall (WAF) є одним із ключових засобів захисту веб-застосунків, оскільки дозволяє фільтрувати та аналізувати HTTP-запити з метою виявлення та блокування шкідливого трафіку. WAF працює за принципом аналізу вхідних та вихідних запитів на основі визначених правил або з використанням методів машинного навчання для виявлення аномальної активності. Існують два основні типи WAF: мережеві (апаратні), які встановлюються перед сервером веб-застосунку, та хмарні, що надаються як послуга (наприклад, AWS WAF, Cloudflare WAF).

Основні функції WAF включають захист від атак SQL-ін’єкцій, XSS, CSRF, DDoS, а також контроль доступу до ресурсів. WAF може використовувати як чорні списки (блокуючи відомі шкідливі IP-адреси), так і білі списки (дозволяючи доступ лише перевіреним клієнтам). Деякі WAF також підтримують поведінковий аналіз для виявлення підозрілих дій, які не потрапляють під стандартні сигнатурні фільтри.

Окрім WAF, для захисту веб-застосунків застосовуються й інші засоби безпеки. Системи виявлення та запобігання вторгненням (IDS/IPS) аналізують трафік у реальному часі та можуть блокувати підозрілі запити. Наприклад, Snort та Suricata є популярними інструментами IDS/IPS для моніторингу мережевої активності.

Також важливим засобом захисту є засоби моніторингу та логування, такі як SIEM-системи (Security Information and Event Management), що допомагають збирати, аналізувати та виявляти загрози на основі логів та подій у системі. Наприклад, такі рішення, як Splunk, ELK Stack або Wazuh, дозволяють проводити детальний аналіз атак та відповідати на інциденти безпеки.

Додатково використовується захист на рівні Content Security Policy (CSP) для запобігання XSS-атакам, обмеження доступу до API за допомогою CORS (Cross-Origin Resource Sharing) та впровадження багатофакторної автентифікації (MFA) для підвищення рівня безпеки доступу до адміністративних панелей та важливих даних.

Ефективне забезпечення безпеки веб-застосунків потребує комплексного підходу, що включає використання WAF, систем моніторингу, IDS/IPS, захисних політик та механізмів контролю доступу для запобігання загрозам і швидкого реагування на потенційні атаки.

2.5 Інструменти моніторингу безпеки веб-застосунків

Моніторинг безпеки веб-застосунків є важливим компонентом стратегії кіберзахисту, оскільки дозволяє виявляти, аналізувати та реагувати на загрози у режимі реального часу. Для цього використовуються спеціалізовані інструменти, що забезпечують збір логів, аналіз мережевого трафіку, виявлення аномалій та автоматизацію заходів реагування.

Одним із найпоширеніших класів інструментів є SIEM-системи (Security Information and Event Management), які збирають, корелюють та аналізують дані з різних джерел, включаючи журнали веб-серверів, баз даних, WAF, IDS/IPS та мережевого обладнання. Популярні SIEM-рішення включають Splunk, IBM QRadar, Elastic Stack (ELK), ArcSight та Graylog. Вони дозволяють виявляти підозрілі активності, автоматизувати сповіщення про інциденти та створювати детальні звіти про стан безпеки веб-застосунку.

Ще одним важливим класом інструментів є IDS/IPS (Intrusion Detection and Prevention Systems), які аналізують вхідний та вихідний трафік веб-застосунку, шукаючи відомі сигнатури атак або аномалії в поведінці користувачів. Приклади таких систем включають Snort, Suricata, Zeek (раніше Bro), які можуть працювати як пасивно (виявлення атак), так і активно (блокування загроз).

Для моніторингу безпеки веб-застосунків також використовуються сканери вразливостей, які перевіряють веб-застосунки на наявність відомих проблем безпеки, таких як SQL-ін’єкції, XSS, CSRF, небезпечні конфігурації серверів та витоки даних. Популярні інструменти в цій категорії: OWASP ZAP, Burp Suite, Acunetix, Nessus, Nikto. Вони дозволяють проводити як автоматизоване, так і ручне тестування веб-застосунків.

Додатково можуть використовуватись системи моніторингу активності користувачів та аналізу поведінки (UEBA – User and Entity Behavior Analytics),

такі як Exabeam, Varonis, які аналізують поведінку користувачів і можуть виявляти аномалії, наприклад, підозрілу зміну прав доступу або спроби несанкціонованого входу.

Не менш важливими є серверні та мережеві журнали (логування), які можна аналізувати за допомогою таких платформ, як Elastic Stack (ELK), Grafana Loki, Fluentd, Prometheus. Вони дозволяють відстежувати стан серверів, баз даних та веб-застосунків, виявляючи потенційні загрози, такі як часті помилки автентифікації або нетипові запити до API.

Для активного моніторингу також застосовуються хмарні засоби безпеки, такі як AWS Security Hub, Azure Security Center, Google Chronicle, які інтегруються з хмарними інфраструктурами та надають розширені можливості аналізу кіберзагроз.

Загалом, ефективний моніторинг безпеки веб-застосунків потребує комплексного підходу, що включає використання SIEM-систем, IDS/IPS, сканерів вразливостей, логування та аналізу поведінки користувачів, що дозволяє оперативно реагувати на потенційні загрози та забезпечувати належний рівень кібербезпеки.

3 ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕТОДІВ І ЗАСОБІВ БЕЗПЕКИ

3.1 Критерії оцінки методів та засобів безпеки

Порівняльний аналіз методів та засобів забезпечення безпеки веб-застосунків потребує визначення чітких критеріїв оцінки, які дозволять об’єктивно порівняти їх ефективність. Основні критерії включають наступні аспекти.

Рівень захисту. Цей критерій визначає, наскільки ефективно метод чи засіб може протидіяти різним загрозам безпеці. Важливо оцінити, чи забезпечує він захист від основних атак (SQL-ін’єкції, XSS, CSRF, DDoS, brute force тощо) та наскільки добре справляється з багатовекторними атаками (наприклад, комбінованими атаками на мережевому і прикладному рівнях).

Продуктивність та вплив на систему. Деякі методи захисту можуть суттєво впливати на продуктивність веб-застосунку, викликаючи затримки в обробці запитів або створюючи високе навантаження на сервер. Наприклад, шифрування даних підвищує безпеку, але може споживати значні ресурси. Тому важливо оцінити баланс між рівнем безпеки та впливом на продуктивність.

Зручність інтеграції та використання. Оцінюється, наскільки легко впровадити певний метод або засіб у веб-застосунок. Наприклад, Web Application Firewall (WAF) може вимагати налаштування правил, тоді як вбудовані засоби захисту в фреймворках (наприклад, автоматичний захист від CSRF у Django) можуть бути простішими у використанні.

Масштабованість. Важливо оцінити, чи підходить метод або засіб для застосування у великих проєктах із високим трафіком. Деякі засоби (наприклад, апаратні WAF або хмарні рішення) можуть легко адаптуватися до зростаючих навантажень, тоді як локальні інструменти можуть мати обмеження за продуктивністю.

Надійність і актуальність. Безпекові методи мають оновлюватися відповідно до нових загроз. Оцінюється, чи активно підтримується технологія або інструмент, чи отримує регулярні оновлення, наскільки велика спільнота розробників та експертів підтримує цей підхід.

Вартість впровадження та підтримки. Деякі засоби безпеки є безкоштовними (наприклад, OWASP ZAP, ModSecurity), тоді як комерційні рішення (як-от Cloudflare WAF або Imperva) можуть мати високу вартість ліцензії. Також потрібно враховувати витрати на налаштування, підтримку та оновлення.

Автоматизація та можливості моніторингу. Оцінюється, чи підтримує метод або інструмент механізми автоматичного виявлення загроз, чи є можливість інтеграції з системами моніторингу (SIEM, IDS/IPS), чи доступні механізми сповіщень про інциденти.

Гнучкість налаштувань. Деякі інструменти безпеки дозволяють гнучко налаштовувати рівні захисту, інші – працюють за принципом «чорного ящика» без можливості тонкої конфігурації. Наприклад, WAF може мати кастомні правила блокування, а деякі хмарні сервіси мають лише стандартні механізми захисту.

Юридичні та нормативні відповідності. Методи безпеки повинні відповідати міжнародним стандартам та регуляціям, таким як GDPR, ISO/IEC 27001, NIST, PCI DSS. Деякі підходи можуть не відповідати вимогам щодо зберігання та обробки даних у різних юрисдикціях.

Вибір методів та засобів безпеки залежить від конкретних потреб веб-застосунку, балансу між безпекою, продуктивністю та зручністю впровадження. Використання цих критеріїв дозволяє об’єктивно порівняти різні рішення та вибрати найоптимальніші для конкретного випадку.

3.2 Аналіз ефективності різних методів захисту веб-застосунків

Ефективність методів забезпечення безпеки веб-застосунків залежить від їх здатності протидіяти основним загрозам, мінімізувати вразливості та забезпечувати відповідність сучасним стандартам кібербезпеки. Для оцінки ефективності різних методів аналізуються такі аспекти, як рівень захисту від поширених атак, продуктивність, складність реалізації та відповідність стандартам безпеки (наприклад, OWASP, NIST, ISO/IEC 27001) [1-3].

Захист від атак типу XSS, SQL-ін’єкції та CSRF є одними з основних елементів безпеки сучасних веб-застосунків. Кожен з цих типів атак має свою специфіку і потребує використання різних методів захисту.

Захист від XSS-атак залишається важливим компонентом загальної стратегії безпеки веб-застосунків. У цьому розділі розглянуто ефективність застосування різних методів захисту, таких як екранування вхідних даних та впровадження політики Content Security Policy (CSP), з погляду їх практичної доцільності, впливу на продуктивність і відповідності міжнародним стандартам. Детальний аналіз представлено у табл. 3.1, що дозволяє порівняти методи захисту за ключовими критеріями.

Атаки типу SQL-ін’єкції мають на меті виконати шкідливі SQL-запити на сервері бази даних, що може призвести до несанкціонованого доступу до даних або їх втрати. Для запобігання таким атакам важливо використовувати параметризовані запити, при яких вхідні дані не включаються безпосередньо в SQL-запит, а передаються як параметри. Це виключає можливість виконання шкідливих SQL-команд. Окрім того, застосування ORM (Object-Relational Mapping) дозволяє абстрагувати SQL-запити від конкретного коду програми, що знижує ризик виникнення SQL-ін’єкцій у випадку помилок у коді або невірному форматуванні запитів [5].

Атаки CSRF (Cross-Site Request Forgery) спрямовані на те, щоб змусити користувача виконати непотрібну або небажану операцію без його відома (наприклад, змінити пароль чи виконати фінансові транзакції). Для захисту від CSRF використовуються токени автентифікації (CSRF-токени), які генеруються на сервері при кожному запиті і включаються у форму чи запит на клієнтській стороні. При обробці запиту сервер перевіряє, чи токен відповідає тому, що був створений раніше. Це дозволяє переконатися, що запит дійсно був ініційований законним користувачем, а не зловмисником [6].

Шифрування даних та безпечне зберігання інформації є важливими аспектами забезпечення конфіденційності та цілісності даних у веб-застосунках.

Для захисту даних на різних етапах їх зберігання та передачі використовуються ефективні методи шифрування та гешування.

Шифрування даних за допомогою сучасних алгоритмів забезпечує високий рівень захисту конфіденційної інформації. Наприклад, алгоритм AES-256 є одним з найпоширеніших і надійних методів шифрування для збереження даних. Цей алгоритм забезпечує сильний захист даних при зберіганні, оскільки навіть при великих обсягах даних забезпечує ефективну та надійну конфіденційність [7]. Для захисту переданих даних під час комунікацій між клієнтом і сервером використовують протокол TLS 1.3 (Transport Layer Security), що забезпечує шифрування та захист від атак типу "man-in-the-middle". Це дозволяє гарантувати, що передана інформація не буде доступною для перехоплення або зміни зловмисниками під час її передачі через мережу [7].

Гешування паролів є важливим етапом у забезпеченні безпеки облікових даних користувачів. Замість того, щоб зберігати паролі в їх оригінальному вигляді, вони гешуються за допомогою спеціальних алгоритмів. Для цього використовують PBKDF2 (Password-Based Key Derivation Function 2), bcrypt або Argon2 – сучасні криптографічні алгоритми, що застосовуються для гешування паролів. Ці алгоритми додають складність до процесу гешування, здійснюючи численні ітерації та використовуючи додаткові параметри, що ускладнює процес вгадування паролів або злому гешів. Такий підхід дозволяє значно знизити ризик компрометації облікових даних у разі витоку бази даних, оскільки навіть отримавши гешовані паролі, зловмисники не зможуть легко відновити їх оригінальні значення [8].

Окрім того, для забезпечення більшого рівня безпеки, зберігання конфіденційних даних повинно передбачати використання додаткових засобів, таких як криптографічні ключі, сертифікати SSL/TLS, а також використання багатофакторної автентифікації для підвищення захисту доступу до чутливої інформації. Усі ці методи в сукупності дозволяють забезпечити належний рівень захисту даних від несанкціонованого доступу, крадіжки або їх компрометації.

Використання Web Application Firewall (WAF) є важливою складовою стратегії захисту веб-застосунків від різноманітних шкідливих дій. WAF-фільтри аналізують вхідний трафік, перевіряючи його на наявність аномальних запитів, таких як SQL-ін’єкції, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF) та інші типи атак. Вони можуть блокувати або фільтрувати шкідливі запити, тим самим знижуючи ризик успішних атак на веб-застосунки.

Сучасні WAF-рішення можуть бути реалізовані як хмарні (наприклад, Cloudflare WAF) або локальні (наприклад, ModSecurity). Хмарні WAF-рішення забезпечують більш масштабований підхід до безпеки, оскільки вони здатні обробляти великий обсяг трафіку без додаткових витрат на інфраструктуру та зручні для інтеграції з іншими хмарними сервісами. Наприклад, Cloudflare WAF застосовує постійно оновлювані бази даних сигнатур атак і використовує машинне навчання для виявлення нових загроз, що дозволяє забезпечити високий рівень захисту від нових векторів атак [10].

Локальні WAF-рішення, такі як ModSecurity, можуть бути інтегровані безпосередньо на сервері веб-застосунку та надають більший контроль над конфігурацією та налаштуваннями фільтрів. Вони добре підходять для використання у приватних мережах або на локальних серверах, де є потреба у більш детальному налаштуванні політик безпеки. Однак налаштування і підтримка таких рішень вимагає додаткових зусиль для управління та оновлення.

Хоча WAF може забезпечити ефективний захист від багатьох видів атак, його продуктивність і ефективність залежатимуть від кількох факторів, таких як частота оновлення сигнатур загроз, налаштування фільтрів і взаємодія з іншими системами безпеки. Важливо регулярно оновлювати сигнатури для збереження високої ефективності захисту та уникати "хибних спрацьовувань", коли WAF блокує легітимний трафік, що може вплинути на зручність використання веб-застосунку.

Системи моніторингу, такі як Splunk, Burp Suite, OWASP ZAP, дозволяють виявляти вразливості в реальному часі та аналізувати потенційні загрози [11-13].

Вони також можуть інтегруватися з SIEM-системами для комплексного аналізу подій безпеки.

Методи забезпечення безпеки мають різну ефективність залежно від сценарію використання.

За результатом порівняння ефективності, що наведене у таблиці 3.1, найкращі результати дає комплексний підхід, що поєднує кілька методів захисту, зокрема WAF, SQL-захист, CSP, шифрування та моніторинг безпеки.

Таблиця 3.1 – Порівняння ефективності методів

Метод захисту	Захист від атак	Простота впровадження	Вплив на продуктивність	Відповідність стандартам
XSS-фільтрація + CSP	Високий	Середній	Низький	OWASP, NIST
Параметризовані SQL-запити	Високий	Високий	Низький	OWASP, ISO/IEC 27001
CSRF-токени	Високий	Високий	Низький	OWASP, NIST
Шифрування AES-256	Високий	Середній	Середній	ISO/IEC 27001
WAF	Високий	Середній	Високий	OWASP, NIST
SIEM-моніторинг	Високий	Низький	Високий	ISO/IEC 27001

3.3 Практичне тестування обраних методів безпеки

Практичне тестування є важливим етапом оцінки ефективності методів захисту веб-застосунків. Воно дозволяє виявити вразливості, перевірити працездатність механізмів захисту та оцінити їхню стійкість до атак. У цьому розділі описується процес тестування найбільш критичних методів безпеки, таких як захист від XSS, SQL-ін’єкцій, CSRF, використання WAF, а також перевірка шифрування даних.

Тестування проводилося відповідно до підходів OWASP Testing Guide [1], що включає:

1) Статичний аналіз коду – аналіз вихідного коду веб-застосунку на наявність вразливостей.

2) Динамічний аналіз – тестування в реальному середовищі із застосуванням автоматизованих та ручних методів.

3) Імітація атак (penetration testing) – перевірка ефективності механізмів захисту шляхом спроб експлуатації вразливостей.

4) Тестування захисту від XSS
   - Використано інструменти Burp Suite та OWASP ZAP для виявлення можливих XSS-вразливостей.
   - Тестові скрипти вставлялися у форми введення та URL-параметри.
   - Результат: якщо браузер не виконує вставлений JavaScript-код, отже, захист працює коректно.
   - Найефективніші механізми захисту: екранування (escaping), Content Security Policy (CSP) [2].
5) Перевірка захисту від SQL-ін’єкцій
   - Використано інструменти SQLMap та ручне тестування за допомогою спеціально сформованих запитів (' OR '1'='1).
   - Тестові атаки проводилися на формах автентифікації та пошукових запитах.
   - Результат: у випадках, коли сервер повертав помилки або видавав несанкціонований доступ, вразливість була присутня.
   - Найефективніший спосіб захисту: використання параметризованих запитів [3].
6) Перевірка захисту від CSRF
   - Використано OWASP CSRF Tester для автоматизації атак.
   - Створено підроблений HTML-формуляр, що імітує запит від імені авторизованого користувача.
   - Результат: якщо сервер приймав запит без CSRF-токена – вразливість існувала.
   - Найефективніший метод захисту: впровадження CSRF-токенів та перевірка заголовків клієнта [4].
7) Тестування Web Application Firewall (WAF)
   - Випробувано Cloudflare WAF та ModSecurity.
   - Симульовано DDoS-атаку та SQL Injection через автоматизовані інструменти.

— Результат: Cloudflare WAF ефективно блокував більшість атак, а ModSecurity вимагав додаткового налаштування правил [5].

8) Перевірка безпечного зберігання даних
   — Використано Hashcat для перевірки стійкості гешованих паролів.
   — Тестувалося шифрування AES-256 на збережених даних.
   — Результат: паролі, гешовані за допомогою bcrypt, виявилися найстійкішими [6].

Результати тестування підтвердили, що комплексний підхід із використанням декількох методів забезпечує найкращий рівень безпеки. Найефективнішими виявилися:

— Параметризовані SQL-запити – повністю усувають SQL-ін’єкції.
— Content Security Policy (CSP) – значно знижує ризик XSS-атак.
— CSRF-токени – ефективно запобігають підробленим запитам.
— Web Application Firewall (WAF) – забезпечує загальний захист від атак на рівні мережі.
— Шифрування AES-256 та bcrypt – забезпечують надійне зберігання даних.

Ці результати можуть бути використані для рекомендацій щодо підвищення безпеки веб-застосунків.

3.4 Висновки щодо доцільності використання конкретних рішень

Аналіз застосування різних методів і засобів безпеки веб-застосунків показав, що для досягнення високого рівня захисту необхідно впроваджувати комплексні стратегії. Кожен метод має свої переваги та обмеження, тому їх ефективність значно підвищується при поєднанні різних підходів. Використання параметризованих SQL-запитів є одним з найбільш ефективних способів запобігання SQL-ін’єкціям. Це дозволяє мінімізувати ризик виконання небажаних SQL команд, які можуть бути використані для доступу до конфіденційної інформації. Важливим аспектом є також екранування вхідних даних і використання політики Content Security Policy (CSP) для захисту від атак

типу XSS, оскільки ці методи обмежують виконання небажаних скриптів, що можуть бути введені через вразливі поля вводу на сайті.

Для захисту від атак CSRF необхідно впроваджувати CSRF-токени, які перевіряються на сервері при виконанні важливих операцій, таких як зміна пароля або транзакції. Це дає змогу гарантувати, що запити надходять від справжніх користувачів, а не зловмисників, які намагаються виконати небажані дії на стороні жертви. Перевірка заголовків клієнта також може бути корисним додатковим засобом для ідентифікації небажаних запитів.

У випадках, коли застосування окремих засобів безпеки може бути недостатнім, Web Application Firewall (WAF) є важливим доповненням. Він забезпечує додатковий рівень захисту шляхом фільтрації шкідливих запитів на рівні сервера, блокуючи потенційно небезпечний трафік. Цей підхід є особливо корисним у випадках, коли традиційні методи захисту, такі як інкапсуляція запитів або фільтрація вхідних даних, не здатні виявити складні або нові типи атак.

Для забезпечення конфіденційності та цілісності даних необхідно використовувати сучасні криптографічні методи. Наприклад, шифрування даних за допомогою AES-256 гарантує високий рівень безпеки під час зберігання конфіденційної інформації, а гешування паролів за допомогою алгоритмів bcrypt чи Argon2 дозволяє мінімізувати ризик компрометації облікових даних. Важливим є також використання TLS 1.3 для забезпечення захищених з'єднань під час передачі даних через мережу.

Крім того, для ефективного моніторингу і виявлення потенційних загроз важливо впроваджувати системи моніторингу, такі як Splunk або OWASP ZAP, що дозволяють оперативно реагувати на зміни в системі безпеки і на загрози. Інструменти моніторингу дозволяють аналізувати трафік і виявляти підозрілі дії, що можуть свідчити про спроби атаки, забезпечуючи своєчасне виявлення вразливих місць і мінімізуючи можливі наслідки.

У результаті, доцільно впроваджувати багатошаровий підхід до безпеки веб-застосунків, який включає як превентивні заходи (шляхом використання

різноманітних методів захисту), так і активний моніторинг і реагування на загрози. Це дозволить забезпечити високий рівень безпеки в умовах постійних змін у загрозливому середовищі, а також зробити веб-застосунки стійкими до нових і більш складних атак.

Багаторівнева стратегія передбачає, що навіть якщо один із захисних механізмів буде подолано зловмисником, інші шари зможуть зупинити атаку або суттєво зменшити її наслідки. Такий підхід дозволяє створити не лише технологічно стійкі системи, але й зменшити ризик людського фактора, оскільки автоматизовані рішення доповнюють ручний контроль.
Крім того, безперервне оновлення безпекових механізмів відповідно до актуальних загроз та адаптація до специфіки конкретного веб-застосунку дозволяє уникнути типових помилок та вчасно виявити нові вектори атак. Усе це разом створює фундамент для розробки надійних, масштабованих і довгостроково захищених веб-рішень.

4 РЕКОМЕНДАЦІЇ ЩОДО ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ ВЕБ-ЗАСТОСУНКІВ

4.1 Кращі практики безпеки для веб-розробників

Забезпечення безпеки веб-застосунків є важливою частиною процесу розробки, і веб-розробники повинні бути обізнані з кращими практиками, які допомагають захистити застосунки від різних загроз. Нижче наведені основні кращі практики безпеки для веб-розробників, які допоможуть значно знизити ймовірність вразливостей у веб-застосунках.

1) Використання перевірених бібліотек і фреймворків

Веб-розробники повинні використовувати надійні й перевірені бібліотеки та фреймворки, такі як Django, Spring, Laravel тощо, для створення веб-застосунків. Ці платформи мають вбудовані механізми захисту від найбільш поширених вразливостей, таких як SQL-ін’єкції, XSS, CSRF та інші. Оновлення бібліотек до останніх версій також є важливим для усунення відомих вразливостей.

2) Принцип мінімальних привілеїв

Розробники повинні застосовувати принцип мінімальних привілеїв (Least Privilege), що означає надання користувачам, службам і процесам тільки тих прав доступу, які необхідні для виконання їхніх завдань. Це дозволяє обмежити можливості зловмисників у разі компрометації системи.

3) Захист від SQL-ін’єкцій

Одним із найбільш поширених видів атак є SQL-ін’єкції. Для запобігання цьому необхідно використовувати параметризовані запити та ORM (Object-Relational Mapping) бібліотеки, які дозволяють автоматично обробляти введені дані, усуваючи можливість внесення шкідливих SQL-операторів.

4) Реалізація захисту від XSS (Cross-Site Scripting)

Розробники повинні обробляти й екранувати всі вхідні дані, зокрема з форм та URL-параметрів, для запобігання виконанню шкідливих скриптів у браузері користувача. Окрім того, необхідно використовувати політики Content Security Policy (CSP), що дозволяють обмежити джерела контенту, які можуть бути виконані на веб-сторінці.

5) Захист від CSRF (Cross-Site Request Forgery)

Для запобігання атакам CSRF необхідно використовувати токени для перевірки кожного запиту, який змінює стан на сервері. Такі токени повинні бути унікальними для кожного користувача та сесії, і їх слід перевіряти при кожному зміненому запиті.

6) Шифрування та безпечне зберігання даних

Всі конфіденційні дані, зокрема паролі, повинні бути зашифровані за допомогою надійних алгоритмів, таких як AES для шифрування та bcrypt або Argon2 для гешування паролів. Це забезпечить їх захист від несанкціонованого доступу.

7) Безпечне управління сесіями

Сесії повинні бути налаштовані таким чином, щоб унеможливити крадіжку сесійних даних. Для цього слід використовувати надійні механізми управління сесіями, такі як HttpOnly і Secure флаги для cookies, та регулярну зміну ідентифікаторів сесій. Також слід обмежувати термін дії сесії.

8) Перевірка та моніторинг безпеки

Веб-розробники повинні регулярно перевіряти свої застосунки на наявність вразливостей за допомогою автоматизованих інструментів, таких як OWASP ZAP або Burp Suite. Окрім того, важливо постійно моніторити веб-застосунок за допомогою таких інструментів, як Splunk чи ELK, щоб виявляти потенційні загрози на ранній стадії.

9) Регулярні оновлення та патчі

Веб-застосунки та сервери повинні регулярно оновлюватися, щоб забезпечити захист від нових вразливостей, про які стає відомо після публікації патчів або оновлень безпеки. Це включає оновлення операційних систем, бібліотек, фреймворків та серверного програмного забезпечення.

10) Освічення команди та забезпечення культури безпеки

Важливим аспектом безпеки є освічення розробників щодо безпечних практик програмування. Проведення регулярних тренінгів з безпеки допомагає

зменшити ймовірність помилок, що можуть призвести до вразливостей у коді. Важливо, щоб безпека стала частиною культурної практики розробки.

Виконання цих кращих практик дозволить веб-розробникам значно знизити ризики атак і підвищити загальний рівень безпеки веб-застосунків. Кожен етап розробки, від проєктування до тестування, повинен враховувати принципи безпеки, щоб створити надійний та захищений продукт.

4.2 Автоматизовані засоби перевірки безпеки

Автоматизовані засоби перевірки безпеки є важливим елементом у забезпеченні захищеності веб-застосунків, оскільки вони дозволяють швидко і ефективно виявляти вразливості, що можуть бути пропущені під час ручного тестування. Використання таких інструментів дозволяє автоматизувати процеси сканування, тестування та моніторингу, що значно скорочує час і зусилля, необхідні для перевірки безпеки.

Одним із найбільш ефективних методів є використання інструментів для статичного аналізу коду (SAST). Ці інструменти перевіряють вихідний код або двійкові файли веб-застосунків на наявність вразливостей, таких як неправильне шифрування, некоректно реалізовані перевірки вводу або можливості для SQL Injection. Популярні інструменти цього типу включають SonarQube, який допомагає знаходити помилки в коді, Veracode для виявлення вразливостей на ранніх етапах розробки, а також Checkmarx, який пропонує інтеграцію в процеси розробки для забезпечення постійного контролю.

Іншим важливим інструментом є динамічний аналіз (DAST), який здійснюється під час роботи веб-застосунку. Цей тип інструментів взаємодіє з працюючими веб-застосунками, тестуючи їх на наявність вразливостей, таких як XSS, SQL-ін’єкції і CSRF. До таких інструментів належать OWASP ZAP, безкоштовний інструмент для автоматизованого сканування вразливостей, Burp Suite, що забезпечує глибокий аналіз і виявлення вразливостей, а також Acunetix, який підтримує понад 7 000 різних вразливостей.

Тестування на проникнення, або penetration testing, є ще одним важливим методом для виявлення реальних загроз. Інструменти, такі як Metasploit, Kali

Linux і Nessus, дозволяють симулювати атаки, виявляючи вразливості, які можуть бути використані зловмисниками для компрометації веб-застосунку. Вони дають змогу тестувати не тільки на наявність вразливостей, але й можливість їх експлуатації у реальних умовах.

Ще одним важливим напрямом є моніторинг безпеки. Інструменти для моніторингу допомагають відслідковувати діяльність на веб-серверах і веб-застосунках, дозволяючи оперативно виявляти аномалії та зловмисні дії. До таких інструментів належать Splunk, що використовується для моніторингу безпеки та управління подіями, а також ELK Stack (Elasticsearch, Logstash, Kibana), який дозволяє збирати й аналізувати логи для виявлення потенційних загроз.

Оскільки веб-застосунки часто використовують сторонні бібліотеки, важливо також перевіряти безпеку цих залежностей. Інструменти, як Snyk і OWASP Dependency-Check, допомагають виявляти вразливості в сторонніх бібліотеках та бібліотеках з відкритим кодом, дозволяючи своєчасно оновлювати їх до більш безпечних версій.

Інтеграція автоматизованих інструментів безпеки в процеси безперервної інтеграції та доставки (CI/CD) дозволяє виявляти вразливості ще на етапах розробки. Такі інструменти, як SonarCloud і Snyk, допомагають забезпечити автоматичний аналіз і виявлення вразливостей безпосередньо в процесі розробки коду, що дозволяє знизити ризики ще до випуску продукту.

Автоматизовані засоби перевірки безпеки є важливим інструментом для забезпечення надійного захисту веб-застосунків. Вони допомагають зменшити ймовірність людських помилок, прискорюють процес тестування та дозволяють вчасно виявляти загрози на різних етапах розробки.

4.3 Впровадження DevSecOps у процес розробки веб-застосунків

Впровадження DevSecOps у процес розробки веб-застосунків є важливим кроком для інтеграції безпеки на всіх етапах життєвого циклу розробки. DevSecOps (Development, Security, and Operations) передбачає тісну взаємодію між командами розробників, операційними командами та фахівцями з безпеки,

щоб забезпечити безпечне створення, тестування та експлуатацію веб-застосунків. Інтеграція безпеки у життєвий цикл ПЗ (Secure SDLC) – це ключова концепція сучасної розробки. Безпека повинна враховуватися не лише на етапі тестування, а бути присутньою на кожному етапі: від планування до супроводу. Для кращого розуміння на рис. 4.1 наведено типовий життєвий цикл ПЗ з фокусом на безпеку:

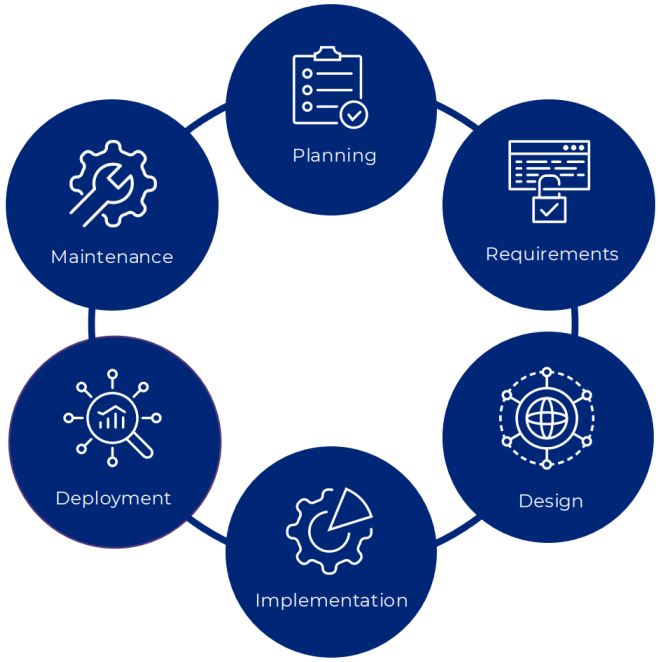

Рисунок 4.1 – Схема життєвого циклу розробки з фокусом на безпеку [22]

Незважаючи на теоретичну завершеність моделі Secure SDLC, у реальному середовищі розробки безпека часто впроваджується фрагментарно або запізно. Багато команд зосереджуються на функціональності, а не на безпеці, що призводить до накопичення вразливостей, які залишаються непоміченими аж до релізу або й довше. Звіт Veracode наочно ілюструє, як нестача безпекових перевірок упродовж життєвого циклу ПЗ призводить до довготривалого існування критичних вразливостей у додатках. [18]

Саме ці дані ми використовуємо для обґрунтування необхідності вбудованого контролю безпеки на всіх етапах SDLC — не як опціонального кроку, а як стандарту сучасної розробки. На рис. 4.2 представлено ключові

показники, які підтверджують, чому інтеграція безпеки у процес розробки — це не просто найкраща практика, а вимога часу.

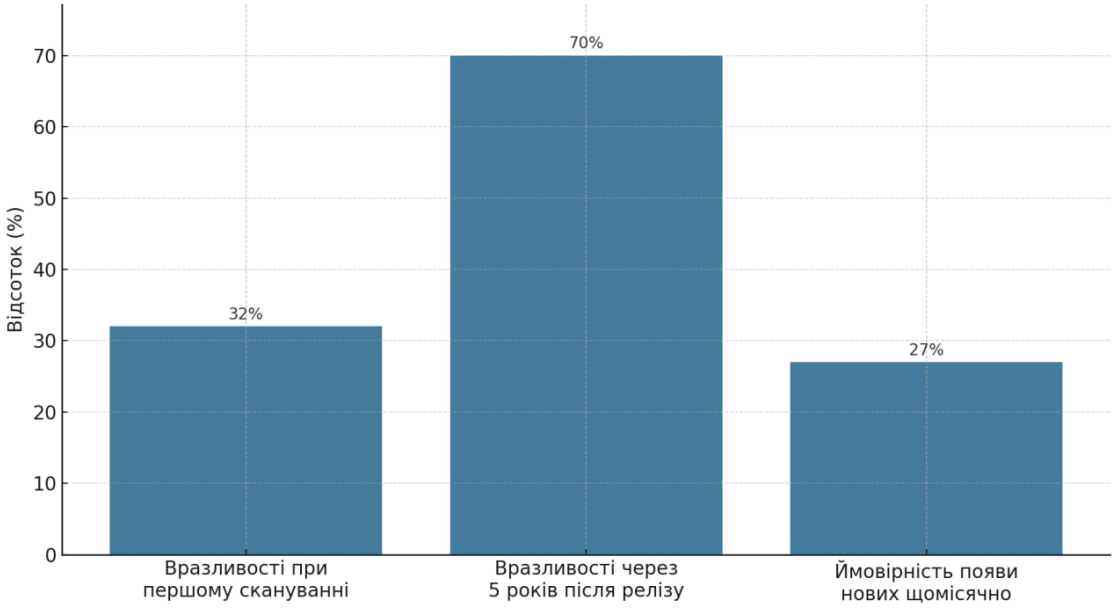

Рисунок 4.2 – Основні показники поширеності вразливостей (за Veracode, 2023)

Одним із основних принципів DevSecOps є автоматизація процесів безпеки, щоб зменшити людський фактор та підвищити ефективність тестування на вразливості. На кожному етапі розробки, починаючи від планування і закінчуючи розгортанням, безпека стає невід'ємною частиною. Це дозволяє виявляти вразливості раніше та уникати їх на етапі запуску або під час експлуатації застосунку.

Одним із ключових аспектів впровадження DevSecOps є автоматизовані тести безпеки, які запускаються на кожному етапі процесу розробки. Це включає використання інструментів для статичного аналізу коду (SAST), які перевіряють код на наявність вразливостей до того, як він буде запущений у середовищі розробки або продуктивному середовищі. Інструменти, як SonarQube, Checkmarx та Fortify, дозволяють знаходити потенційні проблеми безпеки в коді, що зменшує ризики ще до того, як код буде переданий на етап тестування чи впровадження. Детальнішу інформацію про ці інструменти наведено у таблиці 4.1.

Іншим важливим аспектом є інтеграція динамічного тестування безпеки (DAST) у процес розробки. Це дозволяє перевіряти безпеку веб-застосунків в реальному часі під час їхнього функціонування. Інструменти, як OWASP ZAP або Burp Suite, можна налаштувати так, щоб вони автоматично запускали тести під час збірки або розгортання програми. Це дозволяє швидко виявляти вразливості, такі як XSS, CSRF або SQL-ін’єкції, що виникають під час виконання застосунку.

Ще одним важливим аспектом є використання контейнеризації та оркестрації для забезпечення безпеки на рівні інфраструктури. Використання таких технологій, як Docker і Kubernetes, дозволяє ізолювати застосунки в контейнерах, що забезпечує кращу безпеку, а також полегшує моніторинг та оновлення залежностей без ризику впливу на інші частини системи. За допомогою таких інструментів, як Aqua Security або Sysdig, можна забезпечити безпеку контейнеризованих застосунків на всіх етапах розробки та експлуатації.

Важливою складовою є також використання управління доступом і автентифікації через сучасні протоколи, такі як OAuth або OpenID Connect, а також забезпечення належної конфігурації інфраструктури через інструменти для автоматизації, як Terraform або Ansible. Це дає змогу зменшити ймовірність помилок, які можуть виникнути через неправильну конфігурацію або недотримання політик безпеки.

Таблиця 4.1 – Інструменти забезпечення безпеки на етапах DevSecOps

Інструмент	Призначення	Етап застосування	Очікуваний результат
SonarQube	Статичний аналіз коду (SAST)	На етапі написання коду	Виявлення потенційних вразливостей у коді
OWASP ZAP	Динамічне тестування безпеки (DAST)	Після збірки / перед релізом	Виявлення XSS, CSRF, SQL-ін’єкцій тощо
Burp Suite	Аналіз вразливостей під час виконання	Тестування та впровадження	Поглиблене ручне та автоматичне тестування
Checkmarx	Глибокий аналіз коду на вразливості	На етапі розробки	Забезпечення відповідності безпеки внутрішнім політикам
Fortify	Комплексний статичний аналіз	На етапі CI	Попередження критичних помилок до тестування

Інтеграція DevSecOps також вимагає постійної оцінки та аналізу ризиків, щоб постійно вдосконалювати стратегії захисту. Це включає використання засобів моніторингу та аналізу подій безпеки (SIEM-системи, як Splunk або ELK Stack) для оперативного виявлення загроз і реагування на інциденти. Постійний моніторинг допомагає швидко виявляти вразливості та загрози, що дозволяє оперативно вжити заходів.

Окрім того, DevSecOps передбачає важливість навчання команди щодо безпеки на всіх етапах розробки. Це включає не тільки впровадження інструментів, а й зміни у корпоративній культурі, орієнтуючи співробітників на важливість безпеки, в тому числі через тренінги, семінари та обговорення найкращих практик.

Впровадження DevSecOps дозволяє інтегрувати безпеку у всі етапи життєвого циклу веб-застосунку, від розробки до експлуатації. Це підвищує якість програмного продукту, знижує ризики та забезпечує більш надійний захист від сучасних загроз.

4.4 Рекомендовані інструменти та фреймворки для захисту веб-застосунків

Для забезпечення безпеки веб-застосунків існує широкий спектр інструментів і фреймворків, які допомагають виявляти вразливості, захищати дані, моніторити загрози та автоматизувати безпеку на різних етапах розробки та експлуатації застосунків. Далі розглянемо деякі з найбільш рекомендованих інструментів і фреймворків.

OWASP ZAP є одним з найпопулярніших інструментів для динамічного тестування безпеки веб-застосунків. Це відкритий інструмент, який допомагає автоматично виявляти вразливості, такі як XSS, SQL-ін’єкції, CSRF та інші. ZAP можна використовувати як для ручного тестування, так і для автоматизованих перевірок у рамках CI/CD процесів.

Burp Suite – набір інструментів для тестування безпеки веб-застосунків, який дозволяє здійснювати динамічне тестування, аналіз трафіку, сканування на наявність вразливостей, а також маніпуляції з веб-застосунками в реальному

часі. Burp Suite є популярним серед фахівців з безпеки завдяки своїй потужності та універсальності.

SonarQube – інструмент для статичного аналізу коду, який дозволяє знаходити помилки безпеки в коді ще до його виконання. SonarQube підтримує багато мов програмування і допомагає виявляти не тільки вразливості, а й проблеми з якістю коду, що можуть бути джерелом потенційних загроз.

Fortify від Micro Focus – потужний інструмент для статичного і динамічного аналізу безпеки, який допомагає автоматично виявляти вразливості у вихідному коді та на етапі тестування. Fortify підтримує різні платформи і мовні середовища, дозволяючи інтегрувати його в процес розробки та тестування програмного забезпечення.

IBM AppScan є інструментом для тестування безпеки веб-застосунків, який включає статичний, динамічний та інтерактивний аналіз. Він допомагає виявляти і виправляти вразливості в коді та інфраструктурі веб-застосунків.

Snyk є популярним інструментом для безпеки в DevSecOps середовищах, який дозволяє автоматизувати процеси перевірки на вразливості в залежностях, контейнерах та коді. Snyk надає можливості інтеграції з CI/CD, що робить його зручним для команд, що працюють за принципом Agile та DevOps.

Cloudflare WAF – один з найбільш потужних та ефективних інструментів для захисту веб-застосунків від різних атак, таких як DDoS, SQL Injection, XSS, CSRF та інші. Cloudflare WAF автоматично аналізує та блокує шкідливий трафік, забезпечуючи додатковий рівень захисту веб-сайтів і веб-застосунків.

ModSecurity – відкритий фаєрвол веб-застосунку, який може бути використаний для захисту веб-серверів від різних атак. ModSecurity підтримує багато правил для запобігання атакам, таких як SQL-ін’єкції, XSS, CSRF і багатьох інших. Його можна використовувати разом із популярними веб-серверами Apache, Nginx, IIS тощо.

OWASP Dependency-Check є інструментом для перевірки бібліотек і залежностей, що використовуються в проєктах, на наявність вразливих версій.

Це дозволяє знизити ризики, пов’язані з використанням застарілих або вразливих компонентів.

Kali Linux – операційна система для тестування безпеки, яка містить великий набір інструментів для аналізу та перевірки вразливостей в веб-застосунках. Вона включає інструменти для тестування на проникнення, сканування портів, атак на веб-застосунки та багато інших.

Nessus від Tenable – інструмент для сканування на вразливості, який дозволяє знаходити та усувати проблеми безпеки на рівні мережі, серверів, веб-застосунків та інфраструктури. Nessus допомагає виявляти вразливості в веб-застосунках і забезпечувати їх своєчасне виправлення.

Terraform і Ansible – інструменти для автоматизації інфраструктури, які допомагають забезпечити належну конфігурацію безпеки на рівні серверів та контейнерів. Вони дозволяють автоматизувати процеси безпечної конфігурації та управління інфраструктурою, знижуючи ризики помилок у налаштуваннях.

GitLab надає потужні можливості для безпеки в процесах CI/CD, включаючи інтеграцію статичного аналізу коду, перевірки на вразливості в залежностях та інші механізми безпеки. Інструменти безпеки в GitLab дозволяють легко автоматизувати тестування безпеки і дати розробникам зворотний зв'язок щодо безпеки їхнього коду.

Ці інструменти і фреймворки допомагають знизити ризики і покращити безпеку веб-застосунків, надаючи потужні механізми для перевірки, моніторингу, автоматизації і захисту від загроз. Вибір конкретного інструменту залежить від типу застосунку, вимог безпеки та інфраструктури, в якій він працює.

5 АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ ДОСЛІДЖЕНЬ

5.1 Узагальнення отриманих результатів

У цьому розділі здійснюється узагальнення результатів досліджень, проведених у рамках роботи, з метою оцінки ефективності застосованих методів та інструментів безпеки веб-застосунків.

Під час дослідження було вивчено широкий спектр методів захисту веб-застосунків, зокрема, захист від атак типу XSS, SQL-ін’єкції, CSRF, а також інші аспекти безпеки, такі як шифрування даних та безпечне зберігання інформації. Було проведено детальний аналіз різних засобів безпеки, включаючи Web Application Firewall (WAF), автоматизовані засоби перевірки безпеки та інші технології.

Одним із важливих результатів є підтвердження ефективності використання таких інструментів, як OWASP ZAP, Burp Suite, Snyk, а також статичного та динамічного аналізу коду через інструменти, як SonarQube та Fortify. Ці інструменти довели свою здатність виявляти вразливості на різних етапах розробки та експлуатації застосунків, що дозволяє своєчасно реагувати на потенційні загрози і забезпечити високий рівень безпеки.

Додатково було підтверджено важливість інтеграції безпеки в процес розробки програмного забезпечення через практики DevSecOps, що включають автоматизацію тестування безпеки на етапах CI/CD. Це забезпечує більш ефективне виявлення вразливостей на ранніх етапах розробки, знижуючи загрози і витрати на виправлення вразливостей у кінцевому продукті.

Також були проаналізовані методи захисту даних, зокрема, використання протоколів шифрування (SSL/TLS, AES) для забезпечення конфіденційності даних та захисту від несанкціонованого доступу. Шифрування стало важливим елементом для збереження цілісності та конфіденційності даних у сучасних веб-застосунках.

Насамкінець, розглянуто важливість інтеграції інструментів моніторингу та виявлення загроз у процеси управління безпекою, що дозволяє знизити ризики та забезпечити стабільну роботу веб-застосунків в умовах постійно змінюваних

загроз. Використання таких засобів, як Cloudflare WAF та ModSecurity, допомогло значно знизити кількість атак на веб-застосунки, підвищивши рівень безпеки.

Таким чином, узагальнені результати дослідження підтверджують, що поєднання різних методів та інструментів безпеки є ефективним способом забезпечення належного рівня захисту веб-застосунків від сучасних кіберзагроз. Важливим елементом є також постійна еволюція підходів до безпеки та оновлення інструментів відповідно до нових загроз і технологічних змін.

5.2 Висновки за 5 розділ

На основі проведеного дослідження та аналізу методів і засобів забезпечення безпеки сучасних веб-застосунків, можна зробити кілька важливих висновків:

1) Комплексність підходів до безпеки веб-застосунків. Різноманітність загроз для веб-застосунків вимагає застосування комплексних методів захисту, що включають захист від атак типу XSS, SQL-ін’єкції, CSRF, шифрування даних, використання WAF та інструментів моніторингу. Поєднання цих підходів дозволяє досягти високого рівня безпеки веб-застосунків, ефективно протистояти як зовнішнім, так і внутрішнім загрозам.

2) Автоматизація тестування безпеки. Використання інструментів для автоматизованого тестування, таких як OWASP ZAP, Burp Suite, SonarQube, значно спрощує процес виявлення вразливостей на різних етапах розробки та експлуатації застосунків. Інтеграція цих інструментів у процес CI/CD дозволяє вчасно виявляти і усувати проблеми безпеки, знижуючи ймовірність експлуатації вразливостей.

3) Значення практик DevSecOps. Інтеграція безпеки в процеси розробки через практики DevSecOps є важливим напрямом у забезпеченні безпеки веб-застосунків. Постійне тестування та перевірка безпеки під час розробки, а також автоматизація процесів дозволяють значно знизити ризики від вразливостей, забезпечуючи швидкість розробки без компромісів у безпеці.

4) Захист даних та конфіденційності. Шифрування даних стало основним інструментом для захисту конфіденційної інформації в сучасних веб-застосунках. Використання SSL/TLS для захисту трафіку, а також AES для шифрування збережених даних, дозволяє забезпечити високу конфіденційність та цілісність інформації.

5) Моніторинг та виявлення загроз. Інструменти моніторингу, такі як Cloudflare WAF та Splunk, дозволяють виявляти і запобігати атакам в реальному часі. Впровадження таких систем в організаціях допомагає забезпечити стабільну роботу веб-застосунків, знижуючи ризики атак на етапах експлуатації.

6) Рекомендації для подальших досліджень. Оскільки кіберзагрози постійно еволюціонують, важливо проводити подальші дослідження в напрямку розвитку нових методів захисту, зокрема в сфері машинного навчання та штучного інтелекту для виявлення загроз. Також слід вдосконалювати методи інтеграції безпеки в процеси DevSecOps та автоматизації тестування безпеки.

Дослідження показало, що для забезпечення належного рівня безпеки веб-застосунків необхідно використовувати комплексний підхід, який поєднує різні методи та інструменти безпеки, інтегровані в процеси розробки та експлуатації застосунків.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи були розглянуті основні аспекти безпеки веб-застосунків, проаналізовані методи захисту від найпоширеніших атак, а також оцінена ефективність сучасних засобів та інструментів для забезпечення безпеки.

Аналіз сучасних методів захисту дозволив виявити, що більшість атак на веб-застосунки можна попередити за допомогою комплексного підходу до безпеки, що включає захист від атак типу XSS, SQL-ін’єкції та CSRF, шифрування даних, використання WAF та регулярний моніторинг безпеки. Оцінка ефективності різних методів захисту показала, що поєднання традиційних методів безпеки (наприклад, фільтрація введених даних, валідація на стороні сервера) з новими технологіями (такими як DevSecOps) дозволяє значно підвищити рівень безпеки веб-застосунків. Важливим є також впровадження багаторівневих захисних механізмів, що забезпечують безпеку на різних етапах розробки та експлуатації.

Практичне тестування обраних методів довело доцільність застосування автоматизованих інструментів для тестування безпеки, таких як OWASP ZAP, Burp Suite та інші рішення для виявлення вразливостей і слабких місць веб-застосунків. Рекомендації щодо забезпечення безпеки включають необхідність інтеграції найкращих практик безпеки на етапі розробки та експлуатації веб-застосунків, використання шифрування для захисту конфіденційних даних, а також активне впровадження технології DevSecOps для зниження ризиків безпеки в процесі розробки.

Загалом, дослідження підтвердило, що ефективний захист веб-застосунків вимагає постійного удосконалення механізмів безпеки та уважного підходу до кожного етапу життєвого циклу застосунку. Застосування сучасних технологій безпеки та інтеграція інструментів для моніторингу і тестування допомагають зменшити ризики та підвищити надійність веб-застосунків у сучасному інформаційному середовищі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. OWASP Foundation. OWASP Top Ten Web Application Security Risks [Електронний ресурс]. – Режим доступу: https://owasp.org/www-project-top-ten/.

2. NIST. Cybersecurity Framework [Електронний ресурс]. – Режим доступу: https://www.nist.gov/cyberframework.

3. ISO/IEC 27001:2022. Information security, cybersecurity and privacy protection – Information security management systems – Requirements. – Geneva : ISO, 2022. – 42 с.

4. Stuttard, D., Pinto, M. The Web Application Hacker’s Handbook: Finding and Exploiting Security Flaws. – 2nd ed. – Wiley, 2011. – 912 с.

5. Howard, M., LeBlanc, D. Writing Secure Code. – 2nd ed. – Microsoft Press, 2003. – 768 с.

6. Kizza, J. M. Guide to Computer Network Security. – 5th ed. – Springer, 2020. – 592 с.

7. OWASP Foundation. OWASP Cheat Sheet Series [Електронний ресурс]. – Режим доступу: https://cheatsheetseries.owasp.org/.

8. Shostack, A. Threat Modeling: Designing for Security. – Wiley, 2014. – 624 с.

9. Vacca, J. R. Computer and Information Security Handbook. – 3rd ed. – Morgan Kaufmann, 2017. – 1248 с.

10. Cloudflare. What is a Web Application Firewall (WAF)? [Електронний ресурс]. – Режим доступу: https://www.cloudflare.com/learning/ddos/glossary/web-application-firewall-waf/.

11. Splunk Inc. Splunk Enterprise Security [Електронний ресурс]. – Режим доступу: https://www.splunk.com/.

12. Burp Suite. PortSwigger Web Security [Електронний ресурс]. – Режим доступу: https://portswigger.net/burp.

13. OWASP Foundation. OWASP ZAP Project [Електронний ресурс]. – Режим доступу: https://www.zaproxy.org/.

14. Sharma, A. DevSecOps: A Leader’s Guide to Producing Secure Software Without Compromising Speed, Agility, and Innovation. – Apress, 2021. – 320 с.

15. Kim, G., Humble, J., Debois, P., Willis, J. The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. – IT Revolution Press, 2021. – 480 с.

16. Django Software Foundation. Django Documentation [Електронний ресурс]. – Режим доступу: https://docs.djangoproject.com/.

17. Spring Framework. Spring Security [Електронний ресурс]. – Режим доступу: https://spring.io/projects/spring-security.

18. Veracode. State of Software Security Report [Електронний ресурс]. – Режим доступу: https://www.veracode.com/state-software-security-2023-report.

19. Symantec Corporation. Internet Security Threat Report [Електронний ресурс]. – Режим доступу: https://www.symantec.com/.

20. Gartner, Inc. Magic Quadrant for Application Security Testing [Електронний рГАРЕГІПЕРесурс]. – Режим доступу: https://www.gartner.com/.

21. Духонченко В. С. Побудова та дослідження моделі безпечного способу авторизації на основі протоколу OAuth 2.0 для веб-застосунків, що використовують API: дипломна робота бакалавра / НТУУ «КПІ ім. Ігоря Сікорського» – [Електронний ресурс]. – Режим доступу: https://is.ipt.kpi.ua/pdf/Duhonchenko.pdf

22. Kondakova R. Secure Software Development: Overview and practical examples [Електронний ресурс] // OWASP Sofia Chapter. – 2020. – Режим доступу: https://owasp.org/www-chapter-sofia/assets/presentations/202503%20-%20Secure%20Software%20Development:%20Overview%20and%20practical%20examples%20by%20Radostina%20Kondakova.pdf