ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Захист web-застосунків з використанням технології WAF»

зі спеціальності *123 Комп'ютерна інженерія*
(код, найменування спеціальності)

освітньо-професійної програми *Комп'ютерних систем та мереж*
(назва програми)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____
(підпис)

Виконав: здобувач вищої освіти групи КСДМ-62
Марчук Р.О.
(прізвище, ім'я)

Керівник *доцент кафедри, phD* Коротков С.С.
(науковий ступінь, вчене звання, прізвище, ім'я)

Рецензент _____
(науковий ступінь, вчене звання, прізвище, ім'я)

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Комп'ютерної інженерії
Ступінь вищої освіти Магістр
Спеціальність 123 Комп'ютерна інженерія
Освітньо-професійна програма Комп'ютерних систем та мереж

ЗАТВЕРДЖУЮ
Завідувач кафедри ННІІТ
_____ ЛАЩЕВСЬКА Н.О.
"___" _____ 2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Марчук Роман Олексійович
(прізвище, ім'я)

1. Тема кваліфікаційної роботи: «Захист web-застосунків з використанням технології WAF»

керівник кваліфікаційної роботи Коротков С.С., доцент кафедри, phD
(прізвище, ім'я, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «___» _____ 2025 року № ____.

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 27.12.2025 р.

3. Вихідні дані до кваліфікаційної роботи апаратне забезпечення серверів та віртуалізованих середовищ, що використовуються для розгортання веб-застосунків та засобів захисту.
програмне забезпечення для захисту веб-додатків, зокрема технологія Web Application Firewall (WAF)
наукова та технічна література, експлуатаційна документація, нормативні документи, міжнародні стандарти.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз сучасних загроз та ризиків для веб-застосунків.
2. Дослідження технології Web Application Firewall (WAF) та принципів її роботи.
3. Розробка методики впровадження WAF у корпоративну інфраструктуру.

4. Перелік графічного матеріалу
Презентація PowerPoint.

6. Дата видачі завдання 01.09.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ зп	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Визначення актуальності захисту веб-застосунків та постановка задачі	10.09.2025 р.	
2.	Аналіз наукової та технічної літератури з питань захисту веб-додатків і технології WAF.	14.09.2025 р.	
3.	Дослідження сучасних загроз і ризиків для веб-застосунків	25.09.2025 р.	
4.	Розгляд технології Web Application Firewall та принципів її роботи	10.10.2025 р.	
5.	Розробка методики впровадження WAF у корпоративну інфраструктуру.	25.10.2025 р.	
6.	Оформлення результатів дослідження.	22.11.2025 р.	
7.	Підготовка доповіді до захисту.	26.11.2025 р.	

Здобувач вищої освіти _____ Марчук Роман
(підпис) *(ім'я, прізвище)*

Керівник кваліфікаційної роботи _____ Коротков Сергій
(підпис) *(ім'я, прізвище)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 84 стор., 7 табл., 29 рис., 30 джерел.

Об’єкт дослідження – веб-застосунки та інтернет-сервіси, що потребують захисту від сучасних кіберзагроз.

Предмет дослідження – методи та технології забезпечення безпеки веб-застосунків, зокрема застосування технології WAF (Web Application Firewall) для захисту від атак на різних рівнях.

Мета роботи – розробка підходу до захисту веб-застосунків із використанням WAF, включаючи впровадження, налаштування та тестування ефективності рішень на основі сучасних інструментів і методів забезпечення кібербезпеки.

Методи дослідження:

1. Аналіз літературних джерел і нормативних документів – огляд сучасних загроз веб-застосунків, методів їх виявлення та інструментів захисту, включаючи найкращі практики для веб-серверів.
2. Методи системного проєктування – побудова багаторівневої моделі безпеки веб-застосунку, впровадження WAF та інтеграція з існуючою інфраструктурою.
3. Методи моделювання та тестування – створення схем взаємодії компонентів, конфігурація тестового середовища, проведення тестів на проникнення та перевірка налаштувань безпеки.
4. Порівняльний аналіз – оцінка ефективності різних рішень WAF, таких як Imperva Cloud WAF, у реальних сценаріях захисту веб-додатків.

ЗАХИСТ ВЕБ-ЗАСТОСУНКІВ, WAF, IMPERVA CLOUD WAF, МЕРЕЖЕВІ ЕКРАНИ, ТЕСТУВАННЯ НА ПРОНИКНЕННЯ, КІБЕРБЕЗПЕКА, БАГАТОРІВНЕВИЙ ЗАХИСТ, НАСТРОЮВАННЯ БЕЗПЕКИ, ВЕБ-СЕРВЕР,

ABSTRACT

The text part of the master's qualification thesis: 84 pages, 7 tables, 29 figures, 30 sources.

Object of the study – web applications and internet services that require protection against modern cyber threats.

Subject of the study – methods and technologies for ensuring the security of web applications, in particular the use of WAF (Web Application Firewall) technology to protect against attacks at different levels.

Purpose of the study – development of an approach to protecting web applications using WAF, including implementation, configuration, and testing the effectiveness of solutions based on modern tools and cybersecurity methods.

Research methods:

1. Analysis of literature sources and regulatory documents – review of current threats to web applications, methods of their detection, and protection tools, including best practices for web servers.
2. System design methods – development of a multi-layered security model for web applications, WAF implementation, and integration with the existing infrastructure.
3. Modeling and testing methods – creation of interaction schemes between components, configuration of a test environment, penetration testing, and verification of security settings.
4. Comparative analysis – evaluation of the effectiveness of various WAF solutions, such as Imperva Cloud WAF, in real-world web application protection scenarios.

Keywords: WEB APPLICATION PROTECTION, WAF, IMPERVA CLOUD WAF, FIREWALLS, PENETRATION TESTING, CYBERSECURITY, MULTI-LAYERED DEFENSE, SECURITY CONFIGURATION, WEB SERVER

ЗМІСТ

ВСТУП ... 8
1 ОБҐРУНТУВАННЯ ПОТРЕБИ ЗАХИСТУ ВЕБ-ЗАСТОСУНКІВ ... 11
1.1 Сучасні загрози та ризики для веб-додатків ... 11
1.2 Необхідність багаторівневого підходу до захисту ... 19
1.3 Оцінка ефективності налаштувань систем безпеки ... 24
2 ВИКОРИСТАННЯ WAF ДЛЯ ЗАХИСТУ ВЕБ-ЗАСТОСУНКІВ ... 28
2.1 Основні можливості мережевих екранів для захисту ... 28
2.2 Особливості роботи технології WAF ... 34
2.3 Методика впровадження захисту веб-додатків за допомогою WAF ... 40
2.4 Аналіз популярного рішення: Imperva Cloud WAF ... 41
3 ІМПЛЕМЕНТАЦІЯ WAF ТА ПЕРЕВІРКА БЕЗПЕКИ ВЕБ-ЗАСТОСУНКУ ... 45
3.1 Особливості впровадження та адміністрування WAF ... 45
3.2 Розгортання тестового веб-додатку під захист WAF ... 51
3.3 Тестування вразливостей за допомогою сканерів безпеки ... 62
ВИСНОВКИ ... 72
ПЕРЕЛІК ПОСИЛАНЬ ... 73
ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ ... 77

ВСТУП

Актуальність дослідження. Сучасні веб-застосунки стали основою діяльності більшості організацій, забезпечуючи доступ користувачів до онлайн-сервісів, фінансових операцій, державних послуг та комунікаційних платформ. Проте зростання обсягів цифрової взаємодії супроводжується підвищенням рівня кіберзагроз. Статистика свідчить, що понад 70% атак у сфері інформаційної безпеки спрямовані саме на веб-додатки, використовуючи їх вразливості для викрадення даних, порушення роботи сервісів або несанкціонованого доступу до корпоративних ресурсів.

Традиційні засоби захисту, такі як антивірусні системи та класичні мережеві екрани, виявилися недостатньо ефективними проти специфічних загроз веб-рівня – SQL-ін’єкцій, XSS-атак, CSRF, обходу автентифікації чи атак на API. У цих умовах важливу роль відіграють спеціалізовані засоби – Web Application Firewall (WAF), які дозволяють виявляти й блокувати шкідливі запити ще на етапі звернення до веб-сервера.

Актуальність дослідження зумовлена необхідністю пошуку та впровадження ефективних рішень із захисту веб-застосунків, здатних адаптуватися до динамічного розвитку кіберзагроз і відповідати сучасним вимогам безпеки. Використання WAF у корпоративному середовищі є одним із ключових напрямів побудови багаторівневої системи захисту, що забезпечує стабільність функціонування критично важливих сервісів та довіру користувачів.

Аналіз останніх досліджень і публікацій. Проблематика захисту веб-застосунків широко висвітлюється у працях зарубіжних та українських дослідників, зокрема таких авторів, як D. Stuttard, M. Pinto, R. Seacord, які розкривають питання безпечної розробки, аналізу веб-протоколів і типових вразливостей. Значний внесок у дослідження веб-безпеки зробили спеціалісти OWASP, IETF та NIST, що розробляють стандарти, рекомендації та методики оцінювання ризиків. Попри достатню кількість теоретичних напрацювань, низка

аспектів впровадження WAF у корпоративну інфраструктуру залишається недостатньо висвітленою. Зокрема, це стосується практичних підходів до тестування ефективності WAF, порівняльної оцінки його роботи у різних сценаріях та побудови багаторівневих моделей захисту веб-застосунків. Ці питання становлять основу даної кваліфікаційної роботи.

Мета і завдання дослідження. розробити та обґрунтувати підхід до підвищення безпеки веб-застосунків шляхом впровадження Web Application Firewall, здійснити його налаштування, тестування та оцінити практичну ефективність у корпоративному середовищі.

Для досягнення поставленої мети у процесі дослідження вирішувалися такі завдання:

1. проаналізувати сучасні загрози та ризики для веб-застосунків, визначити ключові вектори атак;
2. дослідити існуючі методи, протоколи та технології захисту веб-додатків, приділивши особливу увагу WAF-рішенням;
3. розробити концептуальну модель багаторівневого захисту веб-сервісів з інтеграцією WAF у корпоративну інфраструктуру;
4. здійснити порівняльний аналіз сучасних WAF-платформ (зокрема Imperva Cloud WAF) за критеріями ефективності, адаптивності та продуктивності;
5. створити тестове середовище для моделювання атак та перевірки роботи WAF;
6. виконати тестування вразливостей за допомогою сучасних інструментів безпеки та оцінити рівень зниження ризиків після впровадження WAF.

Об’єктом дослідження є веб-застосунки та інтернет-сервіси, що функціонують у корпоративному середовищі та потребують захисту від сучасних кіберзагроз.

Предметом дослідження є методи та технології забезпечення безпеки веб-додатків, а також практичні підходи до застосування Web Application Firewall як ключового елемента багаторівневої системи кіберзахисту.

У роботі використано комплекс методів, серед яких:

– аналіз наукової літератури, стандартів та рекомендацій у сфері кібербезпеки;

– дослідження архітектури веб-застосунків та принципів роботи їх серверних компонентів;

– моделювання сценаріїв кібератак і побудова тестового середовища;

– експериментальне тестування WAF за допомогою сканерів безпеки та методів penetration testing;

– практичне порівняння різних WAF-рішень за ефективністю та функціональними можливостями.

Наукова новизна роботи полягає у:

– узагальненні та систематизації підходів до інтеграції WAF у корпоративні інформаційні системи;

– розробленні моделі багаторівневого захисту веб-застосунків із включенням WAF як центрального елемента;

– формуванні методу оцінювання ефективності WAF на основі практичного тестування та порівняння з альтернативними засобами;

– проведенні структурованого аналізу сучасних WAF-рішень у контексті їх застосування в умовах цифрової інфраструктури України.

Практична значущість полягає у розробці конкретних рекомендацій щодо впровадження та налаштування WAF у корпоративних мережах державних та комерційних організацій. Отримані результати можуть бути використані для побудови комплексних систем захисту веб-ресурсів, оптимізації процесів реагування на інциденти та підвищення загального рівня кіберзахищеності підприємства.

Структура роботи. Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. Обсяг вступу відповідає встановленим методичним вимогам.

1 ОБҐРУНТУВАННЯ ПОТРЕБИ ЗАХИСТУ ВЕБ-ЗАСТОСУНКІВ

1.1 Сучасні загрози та ризики для веб-додатків

Веб-застосунки є елементом цифрової інфраструктури сучасних підприємств, організацій та державних установ. Вони забезпечують користувачам доступ до інформаційних сервісів, виконання фінансових операцій, взаємодію з базами даних та корпоративними ресурсами. Проте саме відкритість веб-додатків до мережі Інтернет робить їх основною ціллю для кіберзлочинців. За даними звітів OWASP (Open Web Application Security Project) та IBM X-Force, понад 70% усіх атак у сфері інформаційної безпеки спрямовано саме на веб-додатки, оскільки вони містять конфіденційні дані й мають безпосередній зв’язок із критичними системами організацій [1].

Основною причиною високої вразливості веб-застосунків є людський фактор – помилки розробників під час проєктування архітектури, використання застарілих бібліотек, некоректна валідація даних або неправильна конфігурація серверів. Додатковим ризиком є слабке шифрування трафіку та зберігання даних, що спрощує перехоплення інформації між клієнтом і сервером. Крім того, стрімкий розвиток веб-технологій – поява нових фреймворків, API, мікросервісів – випереджає здатність фахівців із безпеки вчасно виявляти та усувати нові типи вразливостей. Це створює ситуацію, коли нові веб-рішення часто впроваджуються без належної перевірки їхнього рівня захищеності.

Веб-додатки мають багаторівневу структуру – клієнтський інтерфейс, серверну частину, базу даних і комунікаційні протоколи. Кожен із цих рівнів може містити потенційні точки входу для зловмисників. Наприклад, вразливість у клієнтському коді може бути використана для XSS-атак, у серверному – для SQL-ін’єкцій, а у невірно налаштованій API – для отримання несанкціонованого доступу до даних. Через взаємопов’язаність усіх компонентів навіть незначна помилка в

логіці автентифікації або авторизації може мати критичні наслідки для безпеки всієї системи.

Сучасні загрози мають комплексний характер і часто поєднують декілька типів атак одночасно. Наприклад, кіберзлочинці можуть спочатку виконати сканування портів і сервісів, далі використати вразливість у веб-додатку для завантаження шкідливого коду, а після цього організувати DoS-атаку, щоб ускладнити реагування служби безпеки. Такі багаторівневі сценарії вимагають від систем захисту здатності не лише фіксувати підозрілу активність, але й аналізувати контекст запитів, що надходять до веб-сервера.

Важливо також розуміти, що кіберзагрози еволюціонують – зловмисники активно використовують штучний інтелект, автоматизовані боти та спеціалізовані інструменти для пошуку вразливостей у відкритих ресурсах. За оцінками аналітичних компаній, середній час між публікацією інформації про нову уразливість та її першим використанням у реальних атаках скоротився до менше ніж 48 годин. Це означає, що навіть короткі затримки з оновленням систем можуть призвести до серйозних наслідків.

У результаті виникають численні типові загрози, описані у переліку OWASP Top 10, який регулярно оновлюється і містить найпоширеніші та найнебезпечніші вразливості веб-застосунків. До них належать, зокрема, порушення контролю доступу (Broken Access Control), ін’єкції (Injection), помилки криптографічного захисту (Cryptographic Failures), вразливості в дизайні (Insecure Design), недостатня перевірка введених даних, використання застарілих компонентів та інші. Загрози на рівні веб-додатків мають не лише статистичний, а й якісний вимір: вони відрізняються механізмом реалізації, цілями зловмисників і наслідками для системи. Для системного розуміння ризиків доцільно перейти від загальних міркувань до класифікації конкретних векторів атак, оскільки саме за їхніми ознаками формуються методи виявлення, профілактики та реагування.

Класифікація атак дозволяє:

– відокремити вразливості, що виникають на рівні даних (наприклад, ін’єкції), від тих, що пов’язані з логікою застосунку або автентифікацією;

– співвіднести кожний вектор з типовими способами експлуатації та очікуваними наслідками;

– сформувати набір заходів захисту (конфігурація серверів, політики доступу, WAF-правила) відповідно до природи загрози.

Першою і однією з найнебезпечніших категорій є SQL-ін’єкції. Суть цієї атаки полягає в тому, що зловмисник подає у поля введення або в параметри HTTP-запиту спеціально сформований рядок, який інтерпретується сервером як частина SQL-запиту до бази даних. Якщо додаток некоректно валідує або не параметризує запити, ін’єкція дозволяє читати, змінювати або видаляти дані в базі, виконувати системні команди або ескалювати доступ. Типовий сценарій починається зі сканування доступних полів і форм, далі – спроб вставити конструкції на кшталт 'OR'1'='1 або більш складні payload для обходу фільтрів. Наслідки – від витоку персональних даних до повного компрометування серверної інфраструктури. Виявляється SQL-ін’єкція за допомогою динамічного сканування вразливостей, аналізу помилок СУБД у відповідях сервера та ручного penetration testing. Міри захисту – параметризовані запити (prepared statements), ORM із безпечними практиками, сувора валідація й санітизація вводу, обмеження привілеїв у БД. WAF значно знижує ризик експлуатації SQL-ін’єкцій: він аналізує вміст HTTP-запитів, відкидає відомі патерни ін’єкцій, застосовує сигнатури та поведінкові правила, і може працювати як додатковий шар захисту до виправленого коду. На рисунку 1.1 показано, як зловмисник вводить шкідливий SQL-запит через форму введення, що дозволяє отримати доступ до імен користувачів та паролів із бази даних.

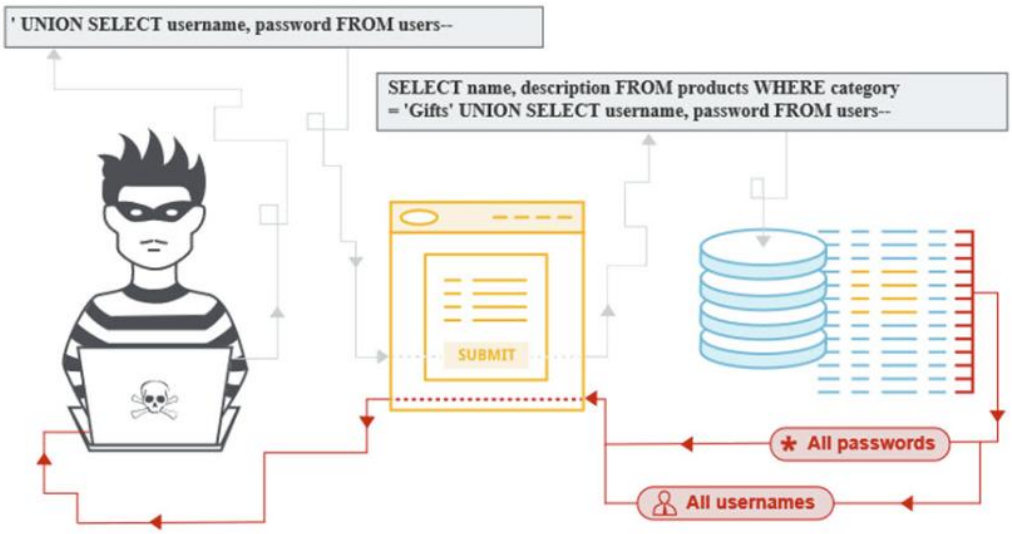


Рисунок 1.1 – Приклад SQL-ін’єкції (SQL Injection)

Наступний поширений вектор – Cross-Site Scripting (XSS). При XSS зловмисник змушує веб-додаток віддавати клієнту сторінки із вбудованим шкідливим скриптом, який виконується в браузері жертви. Реалізація може бути збереженою (payload зберігається на сервері і віддається багатьом користувачам), відображуваною (payload відразу повертається у відповіді на запит) або DOM-базованою (маніпуляція клієнтським скриптом). Наслідки включають викрадення сесій, підміну контенту, фішингові сторінки або віддалене виконання дій від імені користувача. Виявлення здійснюють через аудит коду, сканери XSS та тестування в браузері; профілактика – екранування (escaping) виводу, політика Content Security Policy (CSP), сувора валідація введення і уникнення динамічної вставки небезпечних даних у HTML. WAF може блокувати відомі XSS-патерни у запитах, додавати фільтрацію вводів і в деяких випадках – автоматично санітизувати вміст, але не замінює необхідності безпечної обробки даних на стороні додатка [2].

Рисунок 1.2 показує потік атаки типу XSS: зловмисник відправляє скрипт у форму (POST), сервер зберігає його в базі даних і потім повертає у вигляді HTML до інших клієнтів, що призводить до виконання шкідливого коду в їхніх браузерах.

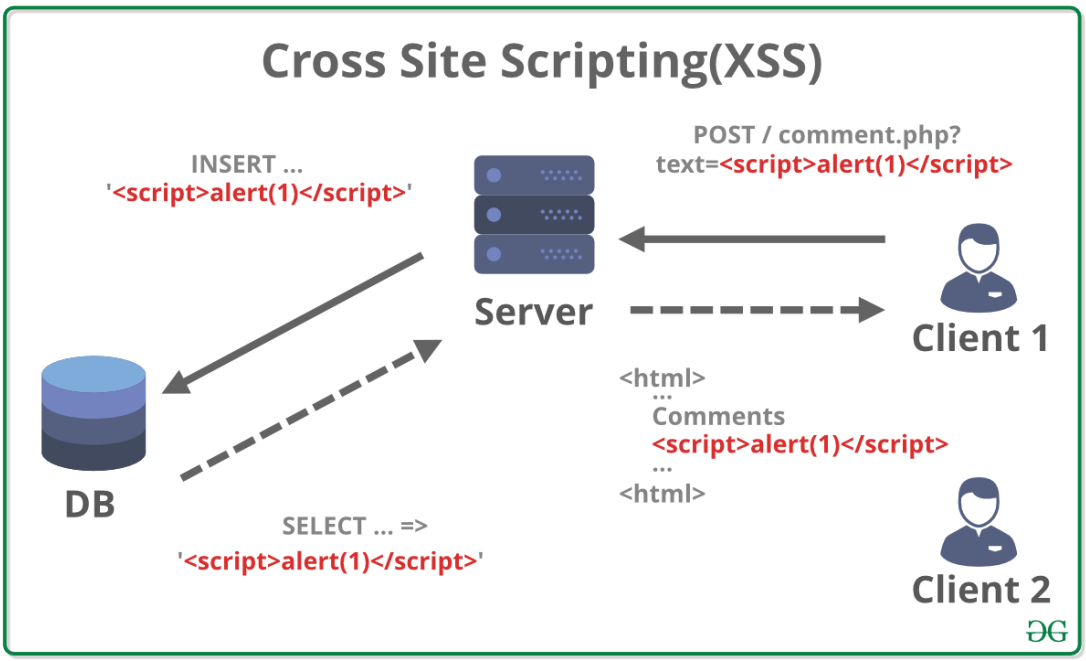


Рисунок 1.2 – Cross-Site Scripting (XSS)

Також широко поширена підробка міжсайтових запитів (CSRF) – атака, коли авторизований користувач ненавмисно виконує дію (наприклад, зміна пароля або здійснення транзакції) через сторонній запит. CSRF експлуатує те, що браузери автоматично відправляють cookie або інші креденшали до цільового сайту. Захист ґрунтується на застосуванні анти-CSRF токенів у формах і заголовках, перевірці заголовка Origin/Referer, застосуванні політик SameSite для cookie та обмеженні небезпечних HTTP-методів. WAF може виявляти аномальні патерни запитів і блокувати підозрілі cross-site звернення, проте найефективніший захист – коректна імплементація токенів у додатку.

Категорія Remote File Inclusion (RFI) і Local File Inclusion (LFI) пов’язана з можливістю завантажити або підключити на сервер зовнішній файл або звернутися до локального файлу, що може призвести до виконання шкідливого коду. RFI/LFI виникає при некоректному обробленні шляхів до файлів або недостатній валідації параметрів. Наслідками можуть бути установка бекдорів, ескалація привілеїв і повний контроль над сервером. Виявлення включає аналіз журналів, перевірку наявності підозрілих include-директив і автоматизовані сканування; пом’якшення – чітке обмеження шляхів (whitelisting), відмови від динамічного include на основі

користувацького вводу, використання механізмів sandboxing. WAF допомагає блокувати запити з підозрілими параметрами шляху або з відомими exploit-патернами.

Атаки DoS/DDoS спрямовані не на витік даних, а на виведення сервісу з ладу шляхом перевантаження його запитами або ресурсними операціями. DDoS часто використовують ботнети для генерації великого обсягу трафіку. Вплив – недоступність сервісу для клієнтів, фінансові втрати та репутаційні ризики. Виявлення потребує моніторингу трафіку, аналізу метрик (зростання RPS, аномальна сесійність) і роботи мережевого та прикладного рівнів. Захист – масштабування, rate-limiting, CDN із захистом від DDoS, фільтрація трафіку на периметрі [3]. WAF може виконувати rate-limiting для HTTP(S)-запитів, блокувати аномальні шаблони звернень і працювати спільно з мережевими рішеннями для зменшення навантаження на бекенд.

Brute-force атаки на автентифікацію полягають у масовому підборі облікових даних; часто їх автоматизують боти. Наслідки – компрометація облікових записів, подальший доступ до привілейованих ресурсів. Методи захисту – політики складності паролів, обмеження кількості спроб входу, багатофакторна автентифікація (MFA), блокування по IP/обліковому запису, використання CAPTCHA. WAF здатний пом’якшувати brute-force через правила throttling, розпізнавання бот-трафіку та тимчасове блокування джерел підозрілої активності.

Окрему увагу потребують атаки на API, які стають особливо актуальними в епоху мікросервісів і мобільних фронтендів. Уразливі API можуть допускати недостатню перевірку прав доступу, ін'єкції у параметри JSON/XML, неправильну обробку пагінації або масивів, що призводить до витоку або підміни даних. Тестування API, застосування аутентифікації й авторизації на рівні токенів (OAuth, JWT із належною валідацією), обмеження швидкості запитів і строгі схеми валідації payload-ів – ключові заходи захисту. WAF із сучасними можливостями розуміє JSON-структури та API-трафік, дозволяючи застосовувати політики для захисту REST/GraphQL інтерфейсів.

Man-in-the-Middle (MitM) – атаки, під час яких зловмисник перехоплює або змінює трафік між клієнтом і сервером. Вони реалізуються при відсутності чи неправильній конфігурації шифрування (TLS), через підміни DNS або через компрометацію мережевого обладнання. Захист – обов'язкове використання HTTPS з сучасними версіями TLS, перевірка сертифікатів, застосування HSTS, DNSSEC і захищені канали між внутрішніми компонентами. WAF не усуває проблему MitM на каналі, але він фіксує аномалії у вмісті запитів і може сигналізувати про підозрілу модифікацію трафіку.

Крім перерахованих, існують також логічні вразливості (помилки бізнес-логіки), insecure deserialization, недостатній контроль доступу та інші складніші кейси, де атакувальник використовує специфічну послідовність дій для досягнення мети. Ці проблеми часто важче автоматично виявити, оскільки вони залежать від семантики застосунку. Тому необхідна комбінація ручного аудиту коду, тестування сценаріїв використання та моніторингу в продуктивному середовищі.

Загалом, кожен тип атаки має свої характерні сигнатури і вимагає сукупності заходів: безпечного кодування, політик доступу, оновлень компонентів, постійного тестування та моніторингу. У цій системі захисту WAF виступає як прикладний рівень фільтрації та детекції, що аналізує HTTP/HTTPS-трафік, застосовує сигнатури, евристичні правила й поведінкову аналітику; однак він не є панацеєю – правильне побудоване захищене застосування й процеси DevSecOps залишаються базовими засобами безпеки.

Наслідки таких атак можуть бути вкрай серйозними як для окремого веб-додатку, так і для всієї інформаційної системи організації. Найпоширенішим результатом є викрадення персональних або фінансових даних користувачів, що створює ризики порушення законодавства про захист інформації (наприклад, GDPR або українського Закону «Про захист персональних даних»). Крім того, втручання в роботу веб-сервісу може призвести до часткової або повної його недоступності, втрати бізнес-процесів та зриву критичних операцій. У випадках, коли атака спрямована на модифікацію або знищення баз даних, підприємство може зазнати прямих фінансових збитків. Не менш важливим є і репутаційний

аспект – користувачі, дізнавшись про витік або компрометацію їхніх даних, часто втрачають довіру до компанії, що ускладнює подальше відновлення її іміджу.

Міжнародною спільнотою з кібербезпеки систематично публікуються аналітичні звіти про найпоширеніші уразливості. Одним із найавторитетніших є список OWASP Top 10, який формується Open Web Application Security Project і відображає десятку найкритичніших ризиків безпеки веб-додатків. Актуальна версія OWASP Top 10 включає такі категорії вразливостей, як:

– A01 – Broken Access Control (порушення контролю доступу);
– A02 – Cryptographic Failures (помилки шифрування);
– A03 – Injection (ін’єкції, зокрема SQL);
– A04 – Insecure Design (небезпечна архітектура);
– A05 – Security Misconfiguration (неправильна конфігурація безпеки);
– A06 – Vulnerable and Outdated Components (застарілі компоненти);
– A07 – Identification and Authentication Failures (збої автентифікації);
– A08 – Software and Data Integrity Failures (порушення цілісності ПЗ та даних);
– A09 – Security Logging and Monitoring Failures (недостатній моніторинг);
– A10 – Server-Side Request Forgery (підробка серверних запитів) [4].

Цей перелік служить орієнтиром для розробників, аналітиків і фахівців з інформаційної безпеки, допомагаючи структурувати процес виявлення та усунення найбільш критичних вразливостей.

Підсумовуючи, можна зробити висновок, що традиційні засоби захисту, такі як антивірусні системи та класичні мережеві фаєрволи, не здатні повною мірою забезпечити безпеку веб-застосунків. Їхня дія обмежується мережевим рівнем (Network Layer) і не охоплює специфічних загроз прикладного рівня (Application Layer), де й відбувається більшість сучасних атак. Тому необхідне впровадження спеціалізованих засобів захисту, що працюють безпосередньо з HTTP/HTTPS-трафіком, аналізують запити та відповіді на рівні логіки веб-додатку. До таких

засобів належать технології Web Application Firewall (WAF), які стають ключовим елементом комплексної багаторівневої стратегії безпеки.

1.2 Необхідність багаторівневого підходу до захисту

Ефективний захист веб-додатків починається з правильної організації безпеки веб-сервера – базового компонента, який обробляє запити користувачів, передає дані до додатка та відповідає за стабільність роботи системи. Навіть найсучасніші інструменти прикладного рівня не зможуть забезпечити високий рівень кіберзахисту, якщо серверна інфраструктура має помилки в конфігурації або не відповідає вимогам безпеки. Тому дотримання найкращих практик адміністрування є першим і необхідним кроком на шляху до побудови комплексного захисту веб-застосунку. На рисунку 1.3 наведено концепцію багаторівневого захисту інформаційних систем (Defense in Depth), що передбачає використання декількох послідовних механізмів безпеки для зменшення ризику успішної атаки.

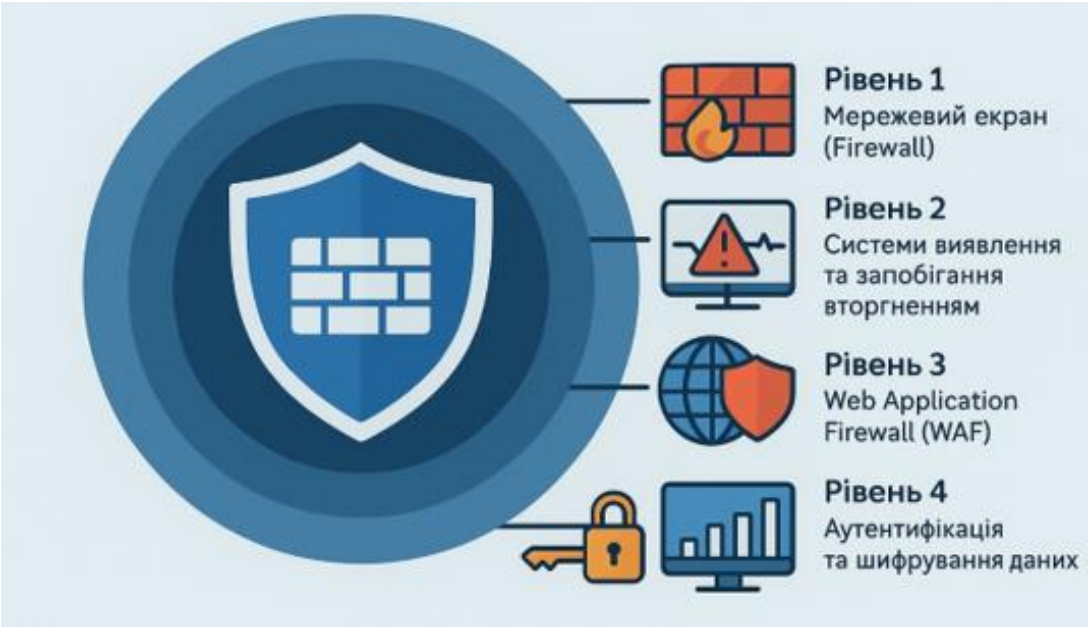


Рисунок 1.3 – Концепція багаторівневого захисту (Defense in Depth).

Рівень 1 – Мережевий екран (Firewall)

Першою лінією оборони є мережевий екран, який працює на мережевому (Network) і транспортному (Transport) рівнях моделі OSI. Його основне завдання – контролювати вхідний і вихідний трафік, дозволяючи лише ті з'єднання, які відповідають встановленим політикам безпеки.

Класичний firewall аналізує IP-адреси, порти, протоколи та напрямки трафіку, запобігаючи несанкціонованому доступу до внутрішньої мережі. Він ефективно блокує спроби сканування портів, мережеві атаки типу SYN Flood, а також обмежує доступ із ненадійних сегментів мережі.

Проте firewall не здатен аналізувати вміст запитів HTTP або SQL, тому не забезпечує повного захисту на рівні веб-додатку, що робить необхідним впровадження вищих рівнів оборони.

Рівень 2 – Системи виявлення та запобігання вторгненням (IDS/IPS)

Другим рівнем захисту виступають системи виявлення (IDS) та запобігання вторгненням (IPS). Вони аналізують мережевий трафік у реальному часі, порівнюючи його з базами сигнатур відомих атак або використовуючи поведінковий аналіз для виявлення аномалій.

IDS-системи фіксують підозрілу активність і повідомляють адміністраторів про можливу атаку, тоді як IPS мають активну функцію блокування шкідливих запитів.

Типовими представниками цих рішень є Snort, Suricata, Zeek, які здатні виявляти спроби експлуатації вразливостей у мережевих протоколах, DoS-атаки чи аномальні з'єднання.

Однак, попри високу ефективність, IDS/IPS також мають обмеження – вони не розуміють бізнес-логіку веб-додатка і не можуть виявляти складні атаки на рівні HTTP чи API, тому потребують доповнення у вигляді WAF [5].

Рівень 3 – Web Application Firewall (WAF)

Третім, прикладним рівнем є Web Application Firewall (WAF) – спеціалізований засіб, що аналізує трафік на рівні додатку (Application Layer).

На відміну від класичного фаєрвола, WAF розуміє структуру HTTP-запитів і відповіді сервера, здатен виявляти ознаки SQL-ін’єкцій, XSS, CSRF, обходу автентифікації, RFI/LFI та інших атак, які спрямовані безпосередньо на логіку веб-застосунку.

WAF може працювати у двох режимах – моніторингу (logging) і захисту (blocking), дозволяючи не лише сповіщати про загрози, а й автоматично блокувати небезпечні запити.

Додаткові можливості включають аналіз API-трафіку, захист від ботів і DDoS, впровадження правил OWASP CRS (Core Rule Set).

Таким чином, WAF є ключовим елементом багаторівневої моделі, який компенсує недоліки нижчих рівнів і забезпечує глибокий контроль прикладних запитів.

Рівень 4 – Аутентифікація та шифрування даних

Наступним рівнем є забезпечення конфіденційності й цілісності інформації. Це досягається за допомогою аутентифікації, авторизації та шифрування.

До ефективних методів аутентифікації належать мультифакторна перевірка (MFA), OAuth 2.0, OpenID Connect, а також використання токенів доступу (JWT).

Шифрування застосовується як до передавання даних (TLS 1.3), так і до їхнього зберігання (AES-256, RSA-2048).

Додатково впроваджуються захищені протоколи взаємодії з API, контроль сесій користувачів та політики автоматичного завершення сесій при бездіяльності.

Цей рівень забезпечує, щоб навіть у випадку перехоплення трафіку чи часткового проникнення в систему дані залишалися захищеними та непридатними до використання.

Рівень 5 – Логування та моніторинг подій

Останній рівень у системі Defense-in-Depth – логування, моніторинг та реагування на інциденти.

Постійний збір логів із веб-сервера, WAF, IDS/IPS, баз даних та інших компонентів дозволяє своєчасно виявляти аномальну поведінку та аналізувати джерело загроз.

У великих системах для цього застосовують SIEM-платформи (Security Information and Event Management), які забезпечують централізований збір, нормалізацію та кореляцію подій безпеки.

Завдяки таким системам можна не лише реагувати на інциденти у реальному часі, а й будувати аналітичні моделі для прогнозування можливих атак.

Регулярний моніторинг підвищує прозорість роботи системи та дає змогу зменшити час виявлення загрози (MTTD) і час реагування (MTTR).

Технологія WAF працює на прикладному рівні (Application Layer), тобто безпосередньо аналізує логіку взаємодії клієнта з веб-додатком. На відміну від традиційних фаєрволів, які контролюють лише мережевий трафік за портами, IP-адресами або протоколами, WAF розпізнає структуру HTTP/HTTPS-запитів і перевіряє вміст кожного параметра, cookie, URI чи тіла запиту [6].

Це дозволяє виявляти та блокувати атаки у реальному часі, які спрямовані безпосередньо на логіку веб-застосунку. Зокрема, WAF ефективно протидіє:

– SQL-ін’єкціям, коли зловмисник намагається змінити SQL-запит до бази даних;

– XSS-атакам (Cross-Site Scripting), що полягають у вбудовуванні шкідливого JavaScript-коду у сторінки веб-додатку;

– CSRF (Cross-Site Request Forgery) – підробці запитів, які виконуються від імені авторизованого користувача;

– RFI/LFI (Remote/Local File Inclusion) – спробам завантаження або виконання сторонніх файлів на сервері;

– атакам на API та REST-запити, які використовуються у сучасних веб-додатках.

WAF виконує глибокий аналіз вмісту запитів – перевіряє синтаксис, структуру параметрів, наявність сигнатур шкідливого коду, а також може застосовувати поведінкові алгоритми та машинне навчання для виявлення нових або модифікованих типів атак.

Сучасні WAF-системи (наприклад, Imperva Cloud WAF, AWS WAF, F5 Advanced WAF) здатні автоматично оновлювати бази сигнатур, застосовувати

політики OWASP Core Rule Set, інтегруватися з SIEM-платформами і навіть адаптивно навчатися на основі профілю нормального трафіку веб-додатку.

WAF виступає не просто “фільтром” для веб-запитів, а інтелектуальним рівнем контролю, який поєднує сигнатурні та евристичні методи аналізу, реагує на загрози в реальному часі та забезпечує активне блокування спроб експлуатації вразливостей ще до того, як вони досягнуть серверної логіки.

Багаторівневий захист діє як комплексна архітектура, де кожен компонент виконує свою роль, а їхня взаємодія створює цілісну систему кібербезпеки.

Принцип роботи можна уявити як послідовність перевірок, через які проходить кожен запит до веб-додатку.

1. На периметрі мережі запит спочатку обробляє мережевий фаєрвол, який перевіряє, чи надходить трафік із дозволених IP-адрес, через допустимі порти, та чи відповідає він визначеним протоколам. Наприклад, якщо запит надійшов із підозрілої IP-адреси, що відома як джерело ботнету, фаєрвол його одразу блокує.

2. Якщо трафік проходить перший рівень, його аналізує IDS/IPS-система, що виявляє спроби сканування, вторгнення або передачу експлойтів у тілі пакету. Вона може, наприклад, розпізнати спробу експлуатації відомої вразливості в Apache чи PHP і зупинити атаку до її реалізації.

3. Далі запит потрапляє до WAF, який перевіряє вміст HTTP/HTTPS-запиту. Тут здійснюється аналіз полів форми, cookie, параметрів URI – тобто тих елементів, через які зазвичай реалізуються SQL-ін’єкції, XSS або CSRF. Якщо у запиті виявлено підозрілі символи чи шаблони, WAF блокує його або передає на додаткову перевірку [7].

4. У випадку, коли запит є легітимним, він доходить до рівня аутентифікації та шифрування, де перевіряються облікові дані користувача, токени доступу, а дані передаються захищеним каналом через TLS. Цей рівень гарантує, що взаємодія між користувачем і сервером є конфіденційною та автентичною.

5. Всі дії на кожному з рівнів – прийняті запити, відхилені звернення, виявлені аномалії – фіксуються у журналах подій. Ці логи надходять у централізовану систему моніторингу або SIEM, де вони аналізуються для

виявлення тенденцій, складання звітів і кореляції подій між різними джерелами. Наприклад, якщо IDS виявляє сканування з певної IP-адреси, а WAF фіксує спробу SQL-ін’єкції з того ж джерела, система може автоматично внести цю адресу до чорного списку на рівні фаєрвола.

Якщо один рівень пропустить атаку, наступний її виявить або заблокує. Це суттєво підвищує стійкість корпоративної інфраструктури до сучасних багатовекторних кібератак. Тільки комбінація засобів захисту може гарантувати стійкість веб-додатку до складних багатовекторних атак.

1.3 Оцінка ефективності налаштувань систем безпеки

Будь-яка система інформаційної безпеки потребує постійного контролю й перевірки ефективності. Навіть найкраще налаштовані засоби захисту з часом можуть втратити актуальність через появу нових вразливостей, оновлення програмного забезпечення чи зміну конфігурацій мережевої інфраструктури. Тому системний підхід до оцінювання ефективності заходів безпеки є необхідною складовою управління ризиками в організації.

Перевірка працездатності систем безпеки здійснюється за допомогою кількох взаємодоповнюючих методів:

1. аудит безпеки – це комплексна перевірка відповідності політик, процедур і технічних засобів вимогам стандартів інформаційної безпеки. Аудит дозволяє виявити слабкі місця в конфігураціях серверів, у налаштуваннях мережі, у системах контролю доступу та в логуванні подій. Його метою є не лише пошук вразливостей, а й формування рекомендацій для їх усунення;
2. тестування на проникнення (Penetration Testing) – метод активної перевірки, коли фахівці з безпеки імітують дії потенційного зловмисника, намагаючись експлуатувати вразливості веб-додатку чи серверної інфраструктури.

Пентест дозволяє оцінити реальний рівень стійкості системи до атак та ефективність реагування на інциденти;

3. сканування вразливостей – використання спеціалізованих програмних засобів, таких як Nessus, Acunetix, OpenVAS, які автоматично перевіряють системи на наявність відомих уразливостей. Цей метод дає можливість регулярно контролювати стан безпеки, виявляти відхилення від належної конфігурації або появу небезпечних компонентів;

4. аналіз логів і журналів подій – один із найважливіших елементів моніторингу, що дозволяє оцінити поведінку систем у динаміці. Завдяки аналізу журналів можна виявити спроби несанкціонованого доступу, аномальні запити або збої у роботі компонентів. Для цього використовуються системи збору та кореляції подій, такі як SIEM (Security Information and Event Management), які автоматизують аналіз і надають аналітичні звіти про стан безпеки.

Для об’єктивної оцінки роботи систем безпеки використовуються кількісні показники:

– кількість виявлених атак протягом певного періоду;

– відсоток успішно заблокованих загроз;

– середній час реагування (Mean Time to Detect – MTTD) та усунення інциденту (Mean Time to Respond – MTTR);

– динаміка зниження рівня ризику після впровадження нових заходів безпеки;

– кількість помилкових спрацювань (false positives), що дозволяє оцінити якість налаштувань WAF і IDS/IPS.

Отримані дані аналізуються у контексті політик безпеки організації та використовуються для постійного вдосконалення конфігурацій систем.

Регулярність перевірок є чинником у підтриманні ефективності захисту інформаційних систем. Перед кожним оновленням програмного забезпечення проводиться аудит конфігурацій і тестування працездатності основних компонентів, що дозволяє виявити можливі конфлікти чи помилки ще до впровадження змін. Після внесення будь-яких коригувань у налаштування систем

безпеки або мережевої інфраструктури обов’язково здійснюються контрольні тести, спрямовані на перевірку цілісності системи та запобігання появі нових вразливостей. Планові перевірки слід проводити регулярно – не рідше одного разу на квартал, а для критичних середовищ або систем із підвищеними вимогами до безпеки – щомісяця. Окреме місце у процесі контролю займає тестування на проникнення, яке доцільно виконувати щонайменше двічі на рік або після суттєвих оновлень веб-застосунку, щоб оцінити реальну стійкість системи до сучасних кібератак.

Використання міжнародних стандартів є основою системного підходу до побудови та оцінювання ефективності інформаційної безпеки. Серед найважливіших нормативних документів, на які спираються сучасні організації, можна виділити ISO/IEC 27001, NIST SP 800-53 та OWASP Testing Guide. Стандарт ISO/IEC 27001 визначає вимоги до системи управління інформаційною безпекою (ISMS) і описує процеси впровадження, контролю та постійного вдосконалення політик безпеки в межах організації [8]. Документ NIST SP 800-53, розроблений Національним інститутом стандартів і технологій США, містить перелік контрольних заходів для захисту інформаційних систем – від управління ризиками та доступом до реагування на інциденти. Натомість OWASP Testing Guide є практичним посібником для проведення тестування веб-додатків, у якому зібрано найкращі методики виявлення вразливостей, оцінки конфігурацій безпеки та перевірки належності реалізації механізмів захисту. Орієнтація на ці стандарти дозволяє організаціям уніфікувати підходи до перевірок, підвищити прозорість процесів і забезпечити порівнюваність результатів аудиту.

Важливу роль у процесі моніторингу стану безпеки відіграє Web Application Firewall (WAF). Цей компонент не лише виконує функцію активного захисту, але й забезпечує глибоке аналітичне спостереження за всіма запитами, що проходять через систему. WAF автоматично веде журнали подій, у яких фіксуються деталі HTTP-запитів, спрацьовування правил, кількість заблокованих атак і типи виявлених загроз. Ці дані можуть агрегуватися у вигляді статистичних звітів, що дозволяють визначати найактивніші вектори атак, рівень завантаження системи та

ефективність налаштувань захисту. Інтеграція WAF із системами SIEM (Security Information and Event Management) забезпечує централізований моніторинг усіх подій безпеки, автоматизовану кореляцію інцидентів і формування сповіщень у реальному часі. Завдяки цьому організація отримує не лише інструмент фільтрації трафіку, а й потужний аналітичний засіб, який підвищує оперативність реагування на загрози.

Ефективність будь-якої системи захисту залежить від її постійного контролю та вдосконалення. Навіть найкраще спроєктовані рішення поступово втрачають актуальність без регулярного тестування, оновлення правил безпеки та аналізу подій. Лише завдяки безперервному циклу аудиту, моніторингу та оптимізації налаштувань можна підтримувати високий рівень стійкості до сучасних кіберзагроз і забезпечити стабільну роботу веб-застосунків у динамічному цифровому середовищі.

2 ВИКОРИСТАННЯ WAF ДЛЯ ЗАХИСТУ ВЕБ-ЗАСТОСУНКІВ

2.1 Основні можливості мережевих екранів для захисту

Мережеві екрани (firewalls) є базовим елементом системи інформаційної безпеки, який контролює вхідний і вихідний трафік між сегментами мережі відповідно до визначених політик. Вони забезпечують фільтрацію пакетів за параметрами IP-адреси, порту, протоколу, напрямку трафіку та стану з'єднання. Основна мета мережевого фаєрвола – обмеження доступу до внутрішніх ресурсів і запобігання несанкціонованим з'єднанням.

Однак із розвитком веб-технологій та появою складних кіберзагроз мережеві фаєрволи виявилися недостатньо ефективними. Вони діють лише на мережевому (Network Layer) та транспортному (Transport Layer) рівнях моделі OSI, але не аналізують зміст трафіку прикладного рівня (Application Layer), де відбуваються більшість сучасних атак на веб-застосунки (рис. 2.1).

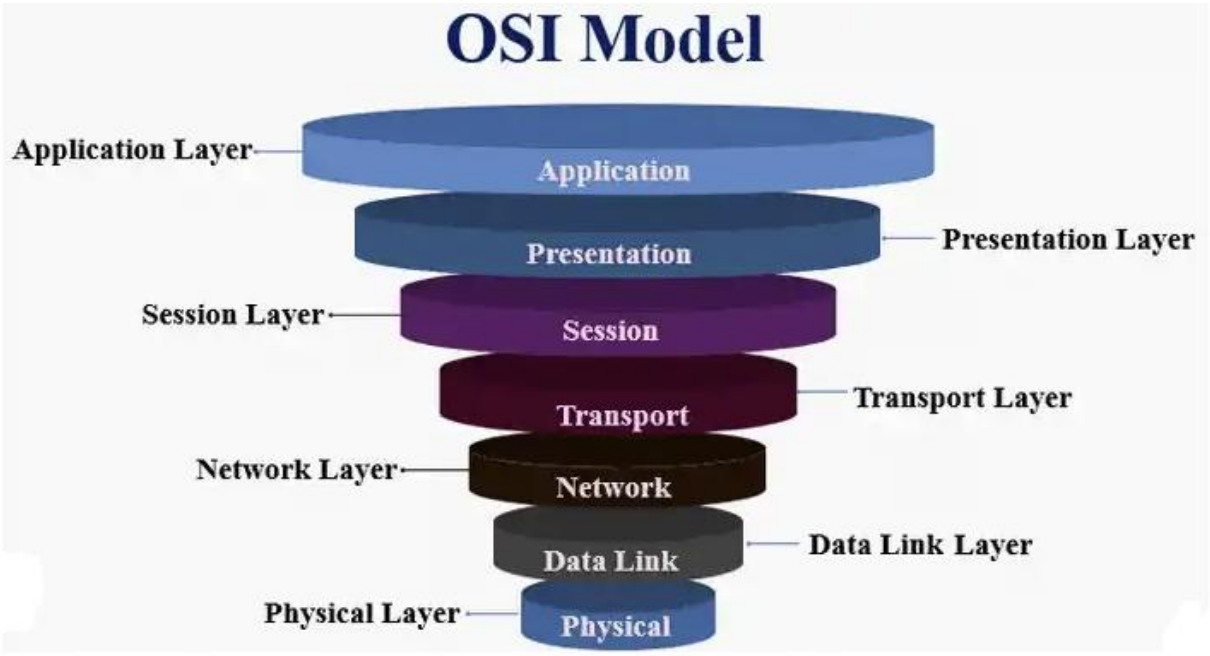


Рисунок 2.1 Модель OSI

Класичний мережевий фаєрвол (Network Firewall) виконує такі функції:

– перевіряє заголовки мережевих пакетів, визначаючи, чи дозволяти або блокувати з’єднання;

– захищає від сканування портів, несанкціонованого доступу та атак типу DoS/SYN Flood;

– може підтримувати stateful inspection – аналіз контексту сеансу для відстеження дозволених з’єднань;

– часто використовується на периметрі корпоративної мережі або як частина маршрутизатора.

Його недолік полягає в тому, що він не бачить структуру HTTP-запитів – тобто не здатен виявити атаки, що відбуваються всередині веб-протоколу, наприклад SQL-ін’єкції**,** XSS, або CSRF. З точки зору Network Firewall, шкідливий HTTP-запит і звичайний виглядають однаково, якщо вони надходять з дозволеної адреси і через відкритий порт (наприклад, 443/TLS) [9].

Web Application Firewall працює на прикладному рівні і аналізує вміст HTTP/HTTPS-трафіку. На відміну від Network Firewall, WAF розуміє структуру веб-запитів: методи (GET, POST, PUT), заголовки, cookie, параметри URI, тіло запиту, та може розпізнавати шаблони атак.

Основні можливості WAF:

– фільтрація запитів за сигнатурами відомих атак (наприклад, за правилами OWASP CRS);

– виявлення підозрілих патернів (евристичний аналіз, machine learning);

– блокування SQL-ін’єкцій, XSS, CSRF, RFI/LFI та атак на API;

– ведення детальних логів подій і інтеграція із системами SIEM або IDS/IPS;

– можливість роботи у режимах monitoring (спостереження) і blocking (активний захист).

Таким чином, WAF – це доповнення до мережевого фаєрвола, яке забезпечує захист саме на рівні логіки веб-додатку, тобто там, де класичні засоби безсилі.

Основні відмінності між Network Firewall та Web Application Firewall (WAF) наведені в таблиці 2.1.

Таблиця 2.1 – Характеристики та функції Network Firewall і WAF

Ознака	Network Firewall	Web Application Firewall (WAF)
Рівень OSI	Network / Transport	Application
Тип аналізу	Пакетний, заголовки IP/TCP	Контент HTTP/HTTPS
Основні функції	Контроль з'єднань, IP, портів, протоколів	Аналіз параметрів, виявлення атак на веб-рівні
Захист від	DDoS, несанкціонованих підключень	SQLi, XSS, CSRF, API-атаки
Режими роботи	Пропуск/блокування трафіку	Моніторинг, блокування, навчання
Інтеграція	З маршрутизаторами, VPN, IDS	З SIEM, IDS/IPS, CDN
Приклад розгортання	На межі корпоративної мережі	Перед веб-серверами або у хмарі

Незважаючи на значні можливості у сфері контролю мережевої взаємодії, класичні фаєрволи мають низку обмежень, що істотно знижують їхню ефективність у контексті захисту веб-додатків. Їхня архітектура спроєктована для роботи переважно на нижніх рівнях еталонної моделі OSI – мережевому та транспортному, де обробляються параметри маршрутизації, портів і протоколів зв'язку. Це означає, що традиційний фаєрвол здатен визначати, з якого вузла надходить трафік, через який порт він передається, та який тип протоколу використовується (TCP, UDP тощо). Проте він не має змоги заглиблюватися у зміст самих даних, що передаються на прикладному рівні, тобто не аналізує структуру HTTP-запитів, HTML-коду чи параметрів веб-форми [10].

У випадку веб-додатків цей недолік є критичним, адже більшість сучасних атак реалізується саме через маніпуляції з вмістом HTTP-запитів або відповіді сервера. Такі загрози, як SQL-ін'єкції, Cross-Site Scripting (XSS), підробка міжсайтових запитів (CSRF), або атаки на API, виникають не через порушення мережевих протоколів, а через експлуатацію логіки роботи веб-застосунку. Для класичного фаєрвола обидва типи запитів – легітимний і шкідливий – виглядають

однаково, оскільки вони надходять з дозволеної IP-адреси, через відкритий порт 80 чи 443, і використовують протокол HTTP(S), який зазвичай дозволений політиками доступу. Внаслідок цього фаєрвол не здатен розрізнити запит, що містить звичайні дані форми авторизації, від запиту, який включає шкідливу SQL-команду, спрямовану на базу даних сервера.

Ще одним обмеженням класичних мережевих екранів є їхня неспроможність аналізувати контекст запитів. Вони працюють на рівні окремих пакетів або сесій, не маючи уявлення про послідовність подій, що відбуваються у межах веб-взаємодії. Наприклад, фаєрвол може дозволити з'єднання з веб-сервером і вважати його безпечним, хоча на рівні прикладного протоколу відбувається атака типу brute-force або передача ін'єкційного коду через POST-запит. Аналогічно, фаєрвол не може перевірити коректність HTTP-заголовків, значення cookie або структурованих даних у форматі JSON, які часто є носіями шкідливих інструкцій. Тобто, навіть за умови повної мережевої ізоляції, логічна вразливість веб-додатку залишається відкритою.

Фаєрволи не здійснюють глибокої інспекції зашифрованого трафіку. Сучасні веб-додатки майже завжди працюють через протокол HTTPS, який використовує TLS-шифрування. Це означає, що вміст запиту та відповіді передається у зашифрованому вигляді, і фаєрвол бачить лише заголовки пакета, але не може переглянути, які дані фактично передаються. Для розшифрування такого трафіку потрібен спеціальний механізм SSL inspection, який традиційні фаєрволи зазвичай не реалізують або виконують із суттєвими обмеженнями, що впливають на продуктивність системи. У результаті фаєрвол пропускає зашифрований трафік без аналізу, відкриваючи шлях для атак, що ховаються всередині HTTPS-сеансів [11].

Обмеження класичних мережевих екранів також пов'язані з відсутністю адаптивності до нових типів загроз. Їхня логіка роботи побудована на статичних правилах, які потрібно постійно оновлювати вручну. У той час як сучасні атаки розвиваються динамічно, змінюючи свої сигнатури та методи обходу захисту, звичайний фаєрвол не має інструментів поведінкового аналізу чи машинного навчання, щоб розпізнавати нові шаблони зловмисної активності. Це робить його

ефективним лише проти відомих загроз, але практично безсилим у випадках zero-day атак або складних багаторівневих експлойтів.

Класичний мережевий фаєрвол, незважаючи на свою важливу роль у побудові периметрального захисту, не може вважатися самодостатнім засобом безпеки для веб-застосунків. Його обмеження зумовлені як технологічними особливостями, так і принциповою різницею між мережевими та прикладними рівнями обробки даних. Саме тому виникла потреба у використанні спеціалізованих систем – Web Application Firewall (WAF), які здійснюють аналіз і фільтрацію трафіку на рівні логіки веб-протоколів, забезпечуючи виявлення та блокування атак, що залишаються непомітними для звичайних мережевих екранів.

Враховуючи наведені принципи роботи мережевих екранів і їхні обмеження, доцільно розглянути порівняльну характеристику класичного мережевого фаєрвола (Network Firewall) і веб-фаєрвола (Web Application Firewall, WAF). Обидва інструменти мають спільну мету – захист інформаційних систем від зовнішніх загроз, однак реалізують її на різних рівнях моделі OSI та з різним ступенем глибини аналізу трафіку. На рис. 2.1 наведено узагальнену схему, що демонструє відмінності між Network Firewall та WAF за основними параметрами роботи.

Таблиця 2.2 – Порівняння Network Firewall та Web Application Firewall

Критерій порівняння	Network Firewall	Web Application Firewall (WAF)
Рівень OSI	Працює на мережевому (3) та транспортному (4) рівнях	Функціонує на прикладному (7) рівні
Об’єкт контролю	Пакети даних, IP-адреси, порти, протоколи	HTTP/HTTPS-запити, параметри URI, cookie, тіла запитів

Продовження таблиці 2.2

Тип аналізу	Поверхневий, фільтрація за статичними правилами	Глибокий (Deep Packet Inspection), аналіз контенту запитів
Призначення	Захист мережевої інфраструктури від несанкціонованого доступу	Захист веб-додатків від атак прикладного рівня
Основні загрози, що блокуються	Сканування портів, DoS, спуфінг, несанкціоновані підключення	SQL Injection, XSS, CSRF, RFI/LFI, API-атаки
Підтримка шифрування (TLS)	Обмежена, не аналізує зашифрований трафік	Може виконувати SSL/TLS-інспекцію для HTTPS-запитів
Контекстна обробка	Відстежує стан сесій (stateful inspection)	Розуміє логіку HTTP-сеансу, перевіряє заголовки, параметри, куки
Механізми реагування	Блокування або дозволення з’єднання	Блокування, переписування запитів, формування звітів
Оновлення та адаптація	Використовує статичні правила, що оновлюються вручну	Має сигнатурні та евристичні механізми, підтримує автоматичне оновлення баз
Інтеграція з іншими системами	Працює з маршрутизаторами, VPN, IDS	Інтегрується з SIEM, IDS/IPS, CDN та системами моніторингу
Типове розташування в мережі	На периметрі корпоративної мережі, між внутрішнім і зовнішнім сегментами	Між користувачем та веб-сервером (on-premises або у хмарі)

Мережевий фаєрвол забезпечує базову лінію оборони, виконуючи фільтрацію трафіку на рівні мережевих протоколів і портів, тоді як WAF аналізує вміст прикладного трафіку, розпізнає структуру HTTP/HTTPS-запитів та виявляє логічні аномалії, характерні для веб-додатків. Обидва засоби не є взаємовиключними – навпаки, їх поєднання формує багаторівневу архітектуру захисту, у якій мережевий екран відповідає за безпеку транспортного шару, а WAF – за цілісність і безпечну роботу прикладного рівня [12].

2.2 Особливості роботи технології WAF

Web Application Firewall (WAF) є спеціалізованим засобом захисту, що працює на прикладному рівні моделі OSI та здійснює глибокий аналіз веб-трафіку. На відміну від класичних мережевих екранів, які контролюють лише параметри з'єднання (IP, порти, протоколи), WAF здатний аналізувати зміст HTTP та HTTPS-запитів, розпізнавати структуру веб-взаємодії і виявляти шкідливу активність на рівні логіки веб-додатку. Його функціонування ґрунтується на трьох основних методах аналізу трафіку – сигнатурному, евристичному та поведінковому, що доповнюють один одного для забезпечення максимальної ефективності захисту (рис. 2.2).

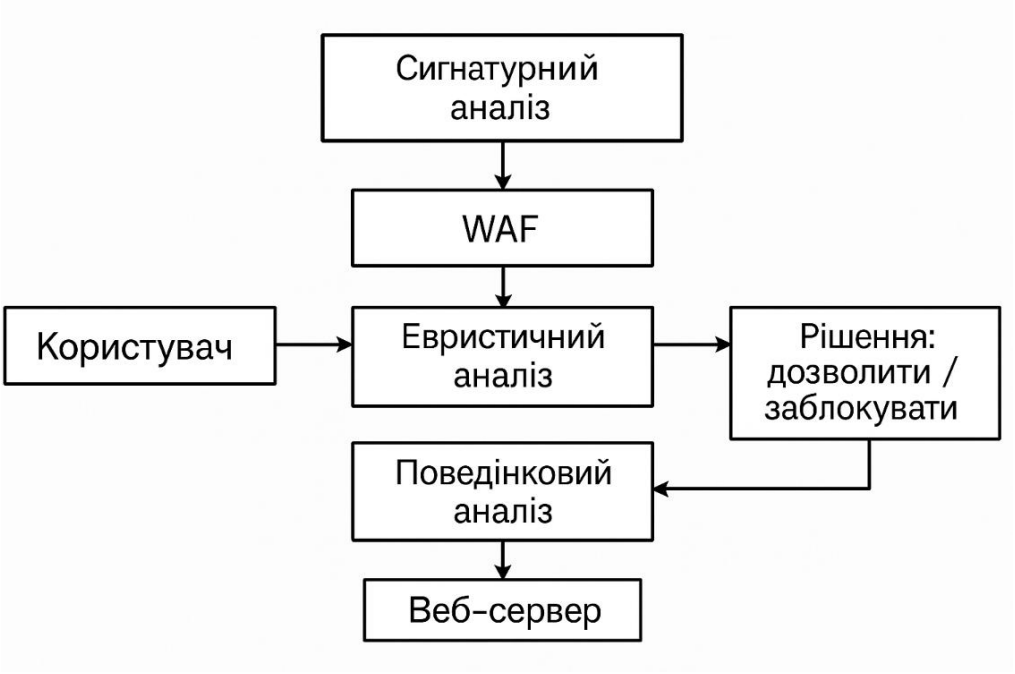


Рисунок 2.2 – Блок-схема процесу фільтрації та прийняття рішення WAF під час обробки користувацького запиту

Першим і найбільш традиційним підходом є сигнатурний аналіз. Його суть полягає у порівнянні вхідних запитів із базою відомих сигнатур атак, тобто характерних шаблонів, які описують ознаки певного типу загрози. Сигнатури

можуть охоплювати ключові фрагменти шкідливого коду, специфічні комбінації символів або структурні особливості HTTP-запиту, що вказують на спробу ін’єкції, виконання скриптів чи обходу автентифікації. Наприклад, якщо у параметрах запиту виявлено SQL-оператори SELECT, UNION або DROP, які не очікуються у звичайній формі авторизації, WAF розпізнає це як потенційну SQL-ін’єкцію та блокує запит. Аналогічно, наявність тегів <script> або JavaScript-коду у полях введення може свідчити про Cross-Site Scripting (XSS). Сигнатурний аналіз має високу точність у виявленні відомих загроз, однак його ефективність обмежена новими або модифікованими типами атак, які ще не описані у базі правил. Саме тому більшість сучасних WAF-рішень (наприклад, Imperva Cloud WAF, F5 Advanced WAF, AWS WAF) підтримують автоматичне оновлення сигнатур на основі баз OWASP CRS (Core Rule Set) або власних аналітичних систем вендора [13].

Другим важливим методом є евристичний аналіз, який передбачає оцінювання запиту не лише за прямим збігом із відомими шаблонами, а й за його структурою, контекстом і логічними ознаками. WAF у цьому випадку використовує набір правил, що описують типову поведінку легітимного трафіку, і визначає відхилення від неї. Наприклад, якщо в одному запиті виявлено занадто велику кількість параметрів, некоректну довжину полів, підозрілу енкодинг-послідовність або нестандартні HTTP-заголовки, система може класифікувати його як потенційно небезпечний. Евристика дозволяє виявляти варіації відомих атак або нові модифікації експлойтів, які не мають чіткої сигнатури. На відміну від сигнатурного методу, вона не потребує точного збігу з відомим шаблоном, що значно підвищує адаптивність WAF до змін середовища загроз. Проте цей підхід може генерувати більшу кількість хибнопозитивних спрацювань, тому його ефективність залежить від точності налаштувань і контекстної фільтрації.

Найбільш інтелектуальним напрямом розвитку WAF є використання поведінкових моделей (behavioral analysis), які базуються на вивченні звичайної динаміки трафіку веб-додатку. У цьому випадку WAF створює профіль «нормальної» поведінки користувачів та сервера, включаючи частоту запитів,

типові параметри, довжину повідомлень, час між зверненнями тощо. Коли система фіксує суттєві відхилення від звичайних показників – наприклад, різке збільшення кількості POST-запитів до певного ресурсу, появу невластивих форматів JSON або нетипових API-викликів, – вона сприймає це як аномалію та активує захисні дії. Поведінковий аналіз особливо ефективний проти нових або складних багатовекторних атак, які не мають усталених сигнатур, наприклад DDoS на прикладному рівні або зловживань API.

Сучасні реалізації WAF дедалі частіше поєднують поведінковий аналіз із елементами машинного навчання, що дає змогу системі самостійно коригувати базові моделі трафіку та адаптуватися до нових сценаріїв використання. Завдяки цьому WAF може розпізнавати навіть ті загрози, які не мають чітких ознак, але демонструють підозрілу динаміку чи відхилення у логіці взаємодії.

Ефективність роботи Web Application Firewall безпосередньо залежить від якості та точності механізмів фільтрації, що визначають, які запити вважати легітимними, а які – потенційно небезпечними. На відміну від мережевих екранів, які оперують фіксованими правилами доступу до портів і протоколів, WAF застосовує гнучкі політики, орієнтовані на аналіз змісту HTTP/HTTPS-трафіку. Ці політики формуються на основі наборів правил, профілів поведінки та шаблонів виявлення загроз, які постійно оновлюються відповідно до актуальних тенденцій кібербезпеки.

Одним із фундаментальних стандартів, що визначає базові принципи побудови таких політик, є OWASP Core Rule Set (CRS) – відкрита колекція сигнатур і правил, розроблена проєктом Open Web Application Security Project. Набір OWASP CRS включає сотні фільтрів, спрямованих на виявлення найпоширеніших атак, описаних у рейтингу OWASP Top 10. До них належать SQL-ін’єкції, XSS, порушення контролю доступу, небезпечна десеріалізація, уразливості API та інші типи загроз. Кожне правило CRS визначає конкретні шаблони поведінки або структуру запиту, які свідчать про зловмисну активність. Наприклад, правила з категорії REQUEST-942-APPLICATION-ATTACK-SQLI описують шаблони запитів, що містять ознаки SQL-команд або параметри,

характерні для ін’єкцій. При виявленні збігу із сигнатурою WAF може заблокувати запит, змінити його або надіслати сповіщення до системи моніторингу безпеки [14].

Система правил OWASP CRS має багаторівневу структуру та може бути адаптована до конкретного середовища. Адміністратор має змогу регулювати чутливість фільтрації, визначаючи поріг спрацювання (paranoia level), який встановлює баланс між кількістю хибнопозитивних результатів і глибиною перевірки. Це дозволяє оптимізувати роботу WAF відповідно до особливостей веб-додатку, зокрема його архітектури, типу трафіку та вимог до продуктивності (рис. 2.3).

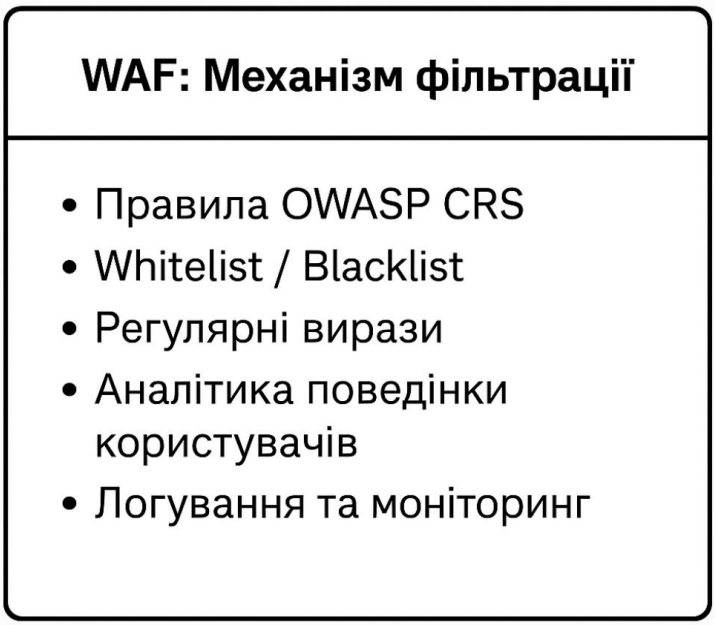


Рисунок 2.3 – Основні елементи механізму фільтрації WAF

Окрім стандартних сигнатур, сучасні WAF використовують механізми білих (whitelist) та чорних (blacklist) списків. Whitelist – це набір дозволених дій, запитів або IP-адрес, яким довіряє система. Наприклад, якщо веб-додаток взаємодіє з певним набором API або сервісів, їх можна внести до білого списку, щоб уникнути зайвих перевірок і прискорити обробку запитів. Натомість blacklist містить перелік заборонених об’єктів – IP-адрес, користувачів, доменів або навіть шаблонів запитів, що вже виявлені як шкідливі. Такі списки формуються як вручну, на основі досвіду адміністратора, так і автоматично, завдяки кореляції подій у системах

SIEM або на основі глобальних репутаційних баз даних (Threat Intelligence Feeds). Використання цих механізмів підвищує гнучкість WAF і дає змогу поєднати централізовану сигнатурну фільтрацію з адаптивними правилами доступу.

Застосування регулярних виразів (regular expressions) є інструментом у фільтрації прикладного трафіку. Цей метод дозволяє WAF виконувати гнучкий пошук і співставлення патернів у вмісті HTTP-запитів, незалежно від їхньої точного формулювання. За допомогою регулярних виразів можна визначити складні умови для виявлення шкідливих фрагментів коду, наприклад SQL-команд, скриптів JavaScript або підозрілих параметрів у URL. Перевагою цього підходу є його універсальність – адміністратор може самостійно створювати правила для специфічних сценаріїв, не чекаючи оновлення сигнатур від постачальника. Регулярні вирази також використовуються для перевірки правильності введених даних, наприклад, для фільтрації символів, що виходять за межі очікуваного формату (алфавітні, числові, службові).

Завдяки комбінації правил OWASP CRS, списків доступу та регулярних виразів, WAF забезпечує багаторівневу систему фільтрації, здатну одночасно розпізнавати стандартні шаблони атак і адаптуватися до унікальних характеристик конкретного веб-додатку. Такий підхід дозволяє досягти балансу між гнучкістю конфігурації та точністю виявлення загроз, мінімізуючи ризик як пропуску небезпечних запитів, так і зайвих блокувань легітимного трафіку. У результаті WAF виступає не лише як засіб фільтрації, а як динамічний аналітичний компонент системи кіберзахисту, що здатний швидко реагувати на зміну ландшафту атак і підтримувати стабільність функціонування веб-застосунків.

Web Application Firewall може функціонувати в декількох режимах, що визначають спосіб взаємодії із трафіком і рівень втручання у роботу веб-додатку. Основними з них є режим моніторингу (Monitoring Mode), режим блокування (Blocking Mode) та режим навчання (Learning Mode). Кожен режим має свої переваги та обмеження, тому вибір конкретного сценарію використання залежить від етапу впровадження WAF і вимог до рівня безпеки.

Таблиця 2.3 – Основні режими роботи Web Application Firewall

Режим роботи WAF	Призначення	Характеристика роботи	Переваги	Обмеження
Monitoring Mode (режим моніторингу)	Спостереження за трафіком без активного блокування запитів	Усі HTTP/HTTPS-запити проходять через WAF, який аналізує їх і веде журнал подій, але не втручається в обробку даних.	Дозволяє оцінити типові шаблони трафіку, виявити потенційні загрози без ризику вплинути на роботу сайту. Використовується на початковому етапі впровадження.	Не забезпечує активного захисту – атаки можуть успішно виконуватись, якщо не налаштовано додаткові засоби безпеки.
Blocking Mode (режим блокування)	Активне запобігання атакам і шкідливим запитам	WAF аналізує запити та блокує ті, що відповідають сигнатурам або евристичним ознакам атак; дозволяє лише безпечний трафік.	Забезпечує максимальний рівень захисту, автоматично реагує на спроби експлуатації вразливостей, зменшуючи ризик компрометації системи.	Може спричинити хибнопозитивні спрацювання, що призводять до блокування легітимних користувачів; потребує ретельного налаштування правил.
Learning Mode (режим навчання)	Автоматичне формування політик безпеки на основі реального трафіку	Система аналізує нормальну поведінку користувачів і веб-додатку, створюючи базові профілі дозволених дій; потім ці профілі використовуються у blocking mode.	Дозволяє адаптувати WAF до специфіки конкретного ресурсу, зменшує кількість помилкових спрацювань, спрощує початкове налаштування.	Під час навчання можливе накопичення небажаних шаблонів, якщо трафік уже містить шкідливі запити; потребує контролю адміністратора.

Режими роботи WAF зазвичай використовуються послідовно: на етапі впровадження застосовується режим моніторингу для збору статистики та

калібрування політик, після чого активується режим навчання, який формує адаптивну модель поведінки користувачів. Лише після цього система переводиться у режим блокування, що забезпечує повноцінний активний захист веб-додатку. Така поступова стратегія впровадження дає змогу мінімізувати ризики помилкових блокувань і гарантує стабільну роботу сервісу навіть у період налаштування.

2.3 Методика впровадження захисту веб-додатків за допомогою WAF

Розпочнемо з розгляду підходів до впровадження системи Web Application Firewall (WAF), яка є одним із ключових елементів комплексного захисту веб-додатків.

WAF може бути реалізовано у вигляді програмного модуля на стороні сервера, апаратного пристрою, або ж надаватися як хмарна послуга (WAF-as-a-Service) для фільтрації вхідного трафіку. Його основна функція полягає у захисті веб-додатків від шкідливих або скомпрометованих клієнтів, виконуючи роль зворотного проксі-сервера (на відміну від звичайного проксі, який захищає користувачів від небезпечних вебресурсів).

Основний принцип роботи WAF полягає у перехопленні та перевірці кожного HTTP-запиту перед тим, як він потрапляє до вебзастосунку. Аналіз здійснюється за допомогою різних методів – від ідентифікації пристрою (device fingerprinting) до викликів CAPTCHA або аналізу поведінкових характеристик користувача. У разі виявлення підозрілої активності WAF блокує запит, запобігаючи можливому вторгненню.

У більшості реалізацій WAF попередньо завантажуються бази правил безпеки, здатні виявляти та блокувати типові шаблони атак, зокрема ті, що описані у списку OWASP Top 10. Такі правила охоплюють найпоширеніші вразливості, серед яких – XSS, SQL-ін’єкції, CSRF, та інші [15].

Окрім стандартних сигнатур, організації можуть створювати власні політики

безпеки, адаптовані під специфіку бізнес-логіки своїх додатків. Проте ефективне впровадження та адміністрування WAF вимагає певних технічних знань і досвіду роботи з мережевими конфігураціями.

Позитивна та негативна моделі безпеки

У системах WAF застосовуються дві основні моделі фільтрації – позитивна та негативна, або ж їх комбінація [15]:

– Позитивна модель безпеки ґрунтується на принципі «білого списку». Вона дозволяє лише ті запити, що відповідають визначеним безпечним правилам. Усе інше автоматично блокується. Перевага цієї моделі полягає у здатності запобігати навіть невідомим атакам, які не входять до сигнатурного списку.

– Негативна модель безпеки реалізує підхід «чорного списку», у якому блокуються лише відомі небезпечні запити, тоді як решта трафіку вважається допустимою. Її простіше впровадити, однак вона не гарантує повного захисту, адже для ефективності потребує постійного оновлення списку шкідливих сигнатур.

Рівень безпеки системи напряму залежить від глибини реалізації політик, якісного оновлення правил та вміння поєднати обидві моделі – позитивну і негативну, що дозволяє досягти балансу між надійністю та продуктивністю.

2.4 Аналіз популярного рішення: Imperva Cloud WAF

Розглянемо одне з провідних сучасних рішень у сфері захисту вебзастосунків – Imperva Cloud WAF, яке вважається одним із найефективніших комерційних інструментів для забезпечення безпеки корпоративного рівня.

Imperva Cloud WAF – це високорівнева система вебзахисту, розроблена для блокування складних атак на прикладному рівні незалежно від того, де розміщено вебресурси – у хмарній інфраструктурі чи локальному дата-центрі. Завдяки своїй архітектурі рішення гарантує постійний захист критично важливих активів від

широкого спектра загроз, включно з XSS, SQL-ін’єкціями, DDoS-атаками та шкідливими ботами.

Imperva Cloud WAF входить до складу інтегрованого комплексу сервісів безпеки та доставки контенту (Application Delivery & Security Suite), який забезпечує багаторівневий захист і включає такі компоненти, як:

– CDN (Content Delivery Network) – для швидкої доставки контенту з мінімальною затримкою;

– захист від DDoS-атак – для підтримання стабільності роботи застосунків навіть під час масштабних атак;

– розширений захист від ботів – для виявлення автоматизованих загроз і фільтрації небажаного трафіку;

– API Security – для контролю безпеки інтеграцій між сервісами;

– балансування навантаження – для ефективного розподілу запитів між серверами в глобальних точках присутності.

Усі ці елементи використовують спільну базу інтелектуальних даних (Threat Intelligence), що дозволяє застосовувати єдину логіку безпеки безпосередньо на периметрі мережі, ще до того, як запит потрапляє до внутрішніх ресурсів. Такий підхід забезпечує швидку реакцію на загрози та мінімізує ризики проникнення у внутрішню інфраструктуру.

Система Imperva Cloud WAF підтримує інтеграцію з провідними платформами безпеки та аналітики (SIEM), що спрощує моніторинг і кореляцію подій у великих організаціях. Крім того, компонент Imperva Attack Analytics, заснований на технологіях штучного інтелекту (ШІ), автоматично групує тисячі подій безпеки у зрозумілі наративи, що значно підвищує ефективність роботи операційного центру безпеки (SOC) і знижує навантаження на аналітиків.

Таким чином, Imperva Cloud WAF є одним із найбільш гнучких і масштабованих рішень у сфері кіберзахисту, що поєднує в собі автоматизоване виявлення загроз, адаптивну аналітику та інтелектуальне управління трафіком [16].



Рисунок 2.4 – Принципова схема обробки вхідного вебтрафіку у системі Imperva Cloud WAF

Постійний захист від вебзагроз

Однією з ключових переваг Imperva Cloud WAF є безперервний моніторинг і динамічний захист від вебзагроз у режимі реального часу. Розширений механізм ідентифікації користувачів здійснює профілювання всього вхідного трафіку безпосередньо на межі мережі, що дозволяє точно розрізняти легітимних клієнтів і потенційних зловмисників ще до того, як запит досягає вебзастосунку.

Такий підхід забезпечує високу ефективність обробки запитів:

– зменшує навантаження на вебсервери,

– скорочує споживання пропускної здатності,

– знижує залежність від людського фактору під час моніторингу безпеки.

Завдяки цьому більшість клієнтів Imperva Cloud WAF застосовують режим повного блокування “із коробки”, оскільки система демонструє високу точність детектування загроз із мінімальною кількістю хибних спрацьовувань.

Захист від OWASP Top 10

Imperva Cloud WAF забезпечує комплексний захист від основних типів атак, визначених у OWASP Top 10, включаючи:

– Cross-Site Scripting (XSS) – запобігання виконанню шкідливих скриптів у браузері користувача;

– SQL Injection – блокування спроб маніпулювання запитами до бази даних;

– Remote File Inclusion (RFI) та Illegal Resource Access – недопущення несанкціонованого доступу до зовнішніх ресурсів або файлів.

Система працює у тісній інтеграції зі стеком Imperva Application Security, який поєднує різні механізми пом’якшення наслідків атак – від захисту від DDoS до виявлення ботів, що здійснюють автоматизовані атаки на API.

Дослідницькі лабораторії Imperva Threat Research Labs постійно відстежують нові вразливості з відкритих джерел і формують оновлені сигнатури загроз. На основі краудсорсингового аналізу трафіку, що проходить через глобальну мережу Imperva, система автоматично створює та поширює нові правила захисту серед усіх користувачів.

Оновлення сигнатур відбувається щодня, що гарантує захист навіть від нових і раніше невідомих загроз [16].

Легкість налаштування та масштабованість

Imperva Cloud WAF вирізняється простотою конфігурації та високим рівнем масштабованості. Керування здійснюється через зручний вебінтерфейс, який додатково захищений двофакторною автентифікацією.

Графічна панель адміністратора дозволяє:

– створювати індивідуальні правила безпеки, адаптовані до конкретного середовища;

– централізовано керувати політиками безпеки для всіх вебресурсів;

– отримувати оперативний огляд ландшафту загроз у реальному часі.

Завдяки інтеграції з Terraform та іншими інструментами автоматизації DevOps, масштабне розгортання політик (до десятків тисяч правил) може здійснюватися протягом кількох секунд.

Централізована панель керування об’єднує управління різними аспектами захисту, включно з API Security, DDoS-механізмами та захистом від ботів, забезпечуючи єдиний контрольний центр для адміністрування безпеки всієї інфраструктури [16].

3 ІМПЛЕМЕНТАЦІЯ WAF ТА ПЕРЕВІРКА БЕЗПЕКИ ВЕБ-ЗАСТОСУНКУ

3.1 Особливості впровадження та адміністрування WAF

Розроблена система побудована на основі мікросервісної архітектури, що забезпечує її гнучкість, масштабованість та спрощує подальше адміністрування. До складу системи входять три основні мікросервіси: Management (керування), Gateway (шлюз) та Policy Holder (сховище політик), кожен із яких виконує окрему функціональну роль у забезпеченні роботи WAF-рішення.

Мікросервісна архітектура

Мікросервісна архітектура – це сучасний підхід до розроблення програмного забезпечення, який передбачає поділ системи на невеликі, автономні сервіси. Кожен із таких мікросервісів виконує окрему бізнес-функцію та взаємодіє з іншими компонентами через чітко визначені інтерфейси прикладного програмування (API). Такий підхід сприяє незалежному розгортанню та оновленню окремих частин системи, спрощує масштабування, підвищує гнучкість і скорочує час виходу нових функцій на ринок, що загалом стимулює інноваційний розвиток [17].

Порівняння монолітної та мікросервісної архітектур

У монолітній архітектурі всі компоненти та процеси тісно пов’язані між собою й функціонують як єдине ціле. Це означає, що при підвищеному навантаженні на одну частину програми доводиться масштабувати всю систему. Зі збільшенням обсягів коду внесення змін, розширення функціональності або оптимізація стають дедалі складнішими, що обмежує можливість експериментів і впровадження нових рішень. Крім того, збільшення кількості взаємозалежних процесів призводить до того, що збій одного з них може вплинути на роботу всієї системи, знижуючи її надійність і доступність.

Мікросервісна архітектура, навпаки, базується на принципі поділу програми на незалежні компоненти, кожен з яких виконує певну бізнес-функцію. Такі сервіси взаємодіють між собою через стандартизовані та легкі API, що забезпечує стабільний обмін даними між ними [18]. Завдяки цьому підходу кожен сервіс можна окремо оновлювати, масштабувати чи замінювати без втручання в інші частини системи. Це підвищує ефективність розробки, стабільність роботи застосунку та дає змогу швидше реагувати на зміни у вимогах користувачів (рис. 3.1).

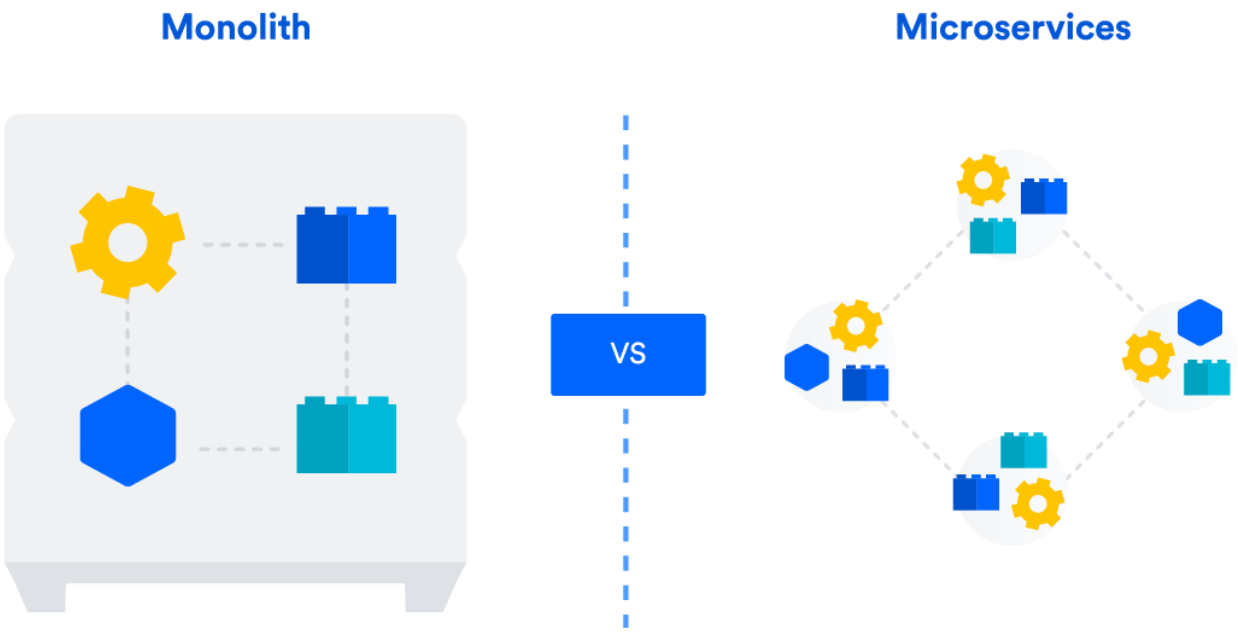


Рисунок 3.1 – Схематичне представлення переходу від монолітної архітектури до мікросервісної структури

Отже, мікросервісна архітектура має низку суттєвих переваг, які роблять її одним із найефективніших підходів до створення сучасних веб-застосунків:

– Підвищена швидкість розробки. Використання мікросервісів сприяє формуванню невеликих, автономних команд розробників, які працюють у чітко визначених межах функціональності. Це дозволяє кожній команді діяти незалежно,

прискорюючи процес розроблення, тестування та впровадження нових компонентів. У результаті цикл розробки скорочується, а продуктивність команд суттєво зростає.

– Гнучке масштабування. Мікросервісна архітектура дає змогу масштабувати кожен сервіс окремо, відповідно до поточного навантаження на певну функцію системи. Такий підхід забезпечує ефективне використання ресурсів, точне планування інфраструктури та підтримку стабільної доступності сервісів навіть за умов раптового збільшення трафіку.

– Простота розгортання. Завдяки підтримці принципів безперервної інтеграції та доставки (CI/CD), мікросервіси спрощують процес впровадження оновлень і дозволяють швидко виконувати деплоймент нових версій або відкат у разі виникнення помилок. Невисока вартість відмови стимулює експерименти, полегшує оновлення коду та скорочує час виходу нових функцій на ринок.

– Технологічна гнучкість. На відміну від монолітних систем, мікросервісна архітектура не нав’язує єдиного технологічного стеку. Кожна команда має можливість обирати оптимальні інструменти, фреймворки та мови програмування для вирішення конкретних завдань, що підвищує ефективність розроблення та якість кінцевого продукту.

– Повторне використання коду. Поділ програмного забезпечення на незалежні, чітко визначені модулі дає змогу повторно застосовувати готові сервіси для реалізації інших функцій. Це знижує потребу у створенні коду з нуля та прискорює процес додавання нових можливостей до системи.

– Підвищена стійкість. Незалежність кожного мікросервісу забезпечує високу відмовостійкість системи. Якщо один компонент виходить з ладу, це не призводить до зупинки всієї програми – порушується лише робота окремої функції. Такий підхід істотно зменшує ризики повного збою системи та підвищує загальну надійність веб-застосунку [19].

Ключовим призначенням мікросервісу «Сховище правил» є централізоване зберігання та агрегування усіх правил, що використовуються в системі. Даний

компонент виступає джерелом актуальної інформації для інших мікросервісів, забезпечуючи єдиний механізм доступу до політик безпеки.

Мікросервіс надає набір API-інтерфейсів, через які інші компоненти системи можуть отримувати або оновлювати перелік правил. Під час кожного розгортання «Сховища правил» відбувається автоматичне сповіщення інших сервісів про зміни, що гарантує актуальність та синхронізацію даних у межах усієї архітектури. Такий підхід сприяє підвищенню узгодженості системи, спрощує адміністрування та забезпечує оперативне оновлення політик безпеки.

Мікросервіс «Менеджмент» виконує роль адміністративного центру системи, що забезпечує взаємодію користувача з компонентами веб-захисту. Основним його завданням є керування процесами конфігурації та моніторингу у межах WAF-рішення.

Користувач через інтерфейс цього мікросервісу може зареєструвати вебсайт, який потребує захисту, переглянути перелік доступних правил безпеки, а також застосувати їх до конкретного ресурсу. Система також надає можливість аналізу вхідного трафіку та формує рекомендації щодо оптимальних правил на основі виявлених шаблонів запитів і потенційних загроз.

Крім того, користувач має змогу переглядати повідомлення про некоректні запити, що були заблоковані системою, із детальним описом причини блокування. Для підвищення гнучкості керування передбачено функцію додавання IP-адрес до “білого списку” (white list) – запити з таких адрес ігноруються під час перевірки за активними правилами безпеки.

Усі зміни, що ініціює користувач – зокрема, застосування або видалення правил, реєстрація сайту, оновлення списку дозволених IP-адрес – автоматично передаються на мікросервіс «Гейтвей» для подальшої обробки та актуалізації конфігурації системи.

Мікросервіс «Гейтвей» є ключовим елементом системи, який відповідає за збір, аналіз та маршрутизацію трафіку. Його архітектура складається з двох основних компонентів – API та проксі-сервера.

Основна функція проксі полягає у фільтрації вхідного трафіку, визначенні легітимних запитів і їхньому перенаправленні безпосередньо на вебсайт користувача. Процес фільтрації здійснюється відповідно до правил безпеки, встановлених у мікросервісі «Менеджмент». У разі, якщо запит не відповідає заданим політикам, система формує повідомлення про некоректний запит і блокує його.

Якщо ж із певної IP-адреси фіксується понад п’ять некоректних запитів, такий IP автоматично вноситься до чорного списку (black list), що запобігає подальшим спробам несанкціонованого доступу.

Компонент API забезпечує комунікацію між мікросервісами, обмінюючись даними щодо змін у налаштуваннях безпеки, повідомленнями про блокування запитів або відмови у доступі. Таким чином, «Гейтвей» виступає центральною ланкою системи контролю трафіку, яка гарантує узгодженість дій усіх мікросервісів та стабільність роботи веб-застосунку.

Для реалізації вебзастосунку було використано сучасний набір технологій, що поєднує зручність клієнтської частини та стабільність серверної логіки.

Клієнтську частину системи розроблено з використанням фреймворку Angular, що базується на мові TypeScript, а також технологій HTML і CSS. Для побудови інтерфейсу застосовано бібліотеку Angular Material, яка містить набір готових компонентів, розроблених відповідно до сучасних принципів вебдизайну.

Серверна частина системи створена на основі фреймворку Spring Boot (мова програмування Java). Розроблення проводилося в інтегрованому середовищі IntelliJ IDEA 2021.2.3, яке забезпечує повний цикл роботи з кодом – від написання до налагодження та тестування.

TypeScript – це мова програмування, тісно пов’язана з JavaScript, оскільки її код компілюється у JavaScript для подальшого виконання в браузері. Головною перевагою TypeScript є строга типізація та об’єктно-орієнтований підхід, що дозволяє виявляти помилки ще на етапі компіляції та запобігати логічним збоям, спричиненим неправильним типом даних. Завдяки цьому розроблення стає більш стабільним і передбачуваним [20].

Angular – це фреймворк з відкритим кодом, побудований на TypeScript, який дає змогу створювати масштабовані вебзастосунки. Його структура складається з трьох основних елементів [21]:

– Фреймворку компонентів, що дозволяє організовувати логіку застосунку у вигляді незалежних модулів;

– Бібліотек, які реалізують роботу з формами, маршрутизацією, клієнт-серверною взаємодією тощо;

– Інструментів розробника, що спрощують процес створення, налагодження та тестування коду.

Для створення інтерфейсу використано Angular Material – набір UI-компонентів, розроблених згідно з принципами Material Design. Ця бібліотека дозволяє швидко створювати адаптивні, естетично привабливі інтерфейси з підтримкою локалізації. Крім того, Angular Material забезпечує високу продуктивність і надійність роботи вебзастосунку [22].

Для побудови графічних елементів і візуалізації даних у системі застосовано бібліотеку Chart.js, яка дозволяє відображати аналітичні дані у вигляді інтерактивних діаграм та графіків [23].

Основою серверної частини стала мова програмування Java – строго типізована, об’єктно-орієнтована мова, що характеризується високою портативністю. Програми на Java компілюються у байт-код, який виконується у віртуальній машині Java (JVM). Завдяки цьому підходу забезпечується незалежність програмного коду від апаратного забезпечення та операційної системи, що робить Java одним із найпоширеніших рішень для серверних застосунків.

Для реалізації серверної логіки використано фреймворк Spring Boot – потужний інструмент з відкритим кодом, який дозволяє створювати автономні вебзастосунки, готові до розгортання у виробничому середовищі. Основними перевагами Spring Boot є:

– автоматична конфігурація компонентів;

– вбудовані контейнери сервлетів для запуску вебдодатків без зовнішнього сервера;

– ефективне управління залежностями (dependency management), що значно спрощує процес розроблення та розгортання [24].

Завдяки поєднанню Angular і Spring Boot система отримала модульну структуру, високу масштабованість і зручність у підтримці, що є важливими критеріями для сучасних вебзастосунків.

3.2 Розгортання тестового веб-додатку під захист WAF

Для розгортання тестового веб-додатку було використано серверне обладнання середнього рівня, яке відповідає типовим вимогам до корпоративної інфраструктури. Вибрана конфігурація дозволяє одночасно запускати веб-сервер, базу даних, допоміжні служби та компоненти безпеки, зокрема WAF.

Обраний сервер має такі технічні характеристики:

- Процесор: Intel Xeon Silver 4210 (10 ядер, 2.2 GHz) – забезпечує достатню продуктивність для обробки паралельних HTTP-запитів і виконання фонових сервісів.
- Оперативна пам’ять: 32 GB DDR4 – дозволяє комфортно працювати з декількома віртуальними машинами та контейнерами.
- Накопичувач: SSD NVMe 1 TB – забезпечує високу швидкість роботи з файлами веб-додатка, журналами подій і базою даних.
- Мережевий інтерфейс: 2×1 Gbps Ethernet – використовується як для доступу користувачів, так і для окремого трафіку моніторингу.
- Гіпервізор: Proxmox VE 8.x / VMware ESXi (за потреби) – для віртуалізації середовища та ізоляції компонентів.

Перевага такої конфігурації полягає у можливості розгортання кількох незалежних вузлів, що моделюють реальну архітектуру веб-застосунку, включаючи маршрутизатор, веб-сервер, WAF-проксі та базу даних.

На базі обраного обладнання реалізовано таку структуру:

1. Віртуальна машина «Web-Server»

– ОС: Ubuntu Server 22.04
– Призначення: розміщення тестового веб-додатку
– ПЗ: Nginx / Apache, PHP, Node.js.

2. Віртуальна машина «Database»

– ОС: Ubuntu Server
– Призначення: збереження даних, над якими будуть здійснюватися тестові атаки (SQLi, LFI та ін.)

3. Віртуальна машина «WAF Gateway»

– ПЗ: Nginx ModSecurity (OWASP CRS), або у випадку хмарного WAF – агент/конектор
– Призначення: перехоплення HTTP/HTTPS-трафіку та застосування правил безпеки

4. Клієнтська машина (Pentest-VM)

– ПЗ: Kali Linux / Parrot Security
– Призначення: запуск сканерів, емуляція атак, тестування ефективності WAF.

Переваги вибраної конфігурації

– Повна ізольованість, що виключає вплив на продуктивні ресурси.
– Можливість відтворення реальних кібератак, включаючи SQLi, XSS, RFI, LFI, brute-force.
– Контрольований трафік, що дозволяє точно вимірювати реакцію WAF.
– Масштабованість: за потреби можна додати нові вузли або інші компоненти (IDS, SIEM).
– Підтримка журналювання та моніторингу, необхідних для аналізу подій безпеки.

Інтерфейс користувача розроблено з урахуванням принципів зручності, інтуїтивності та ефективності роботи з системою. Він складається з трьох основних розділів: «Політики» (Policy), «Попередження про зловмисні запити» (Alerts) та «Аналітика» (Analytics).

На сторінці «Політики» (Policy) користувач має змогу здійснювати основні операції з налаштування захисту вебсайту. Зокрема, передбачено можливість:

– реєстрації сайту для захисту – користувач обирає протокол з’єднання, вказує IP-адресу та номер порту;

– перегляду списку політик безпеки, де відображаються їхні назви та короткі описи;

– активації або деактивації політик, що застосовуються до конкретного сайту (ця функція стає доступною після реєстрації вебресурсу);

– перегляду та редагування “білого списку” IP-адрес, які виключаються з перевірки системою безпеки.

Інтерфейс цього розділу побудовано таким чином, щоб користувач міг швидко змінювати налаштування політик і контролювати параметри безпеки без потреби у глибоких технічних знаннях (рис. 3.2).

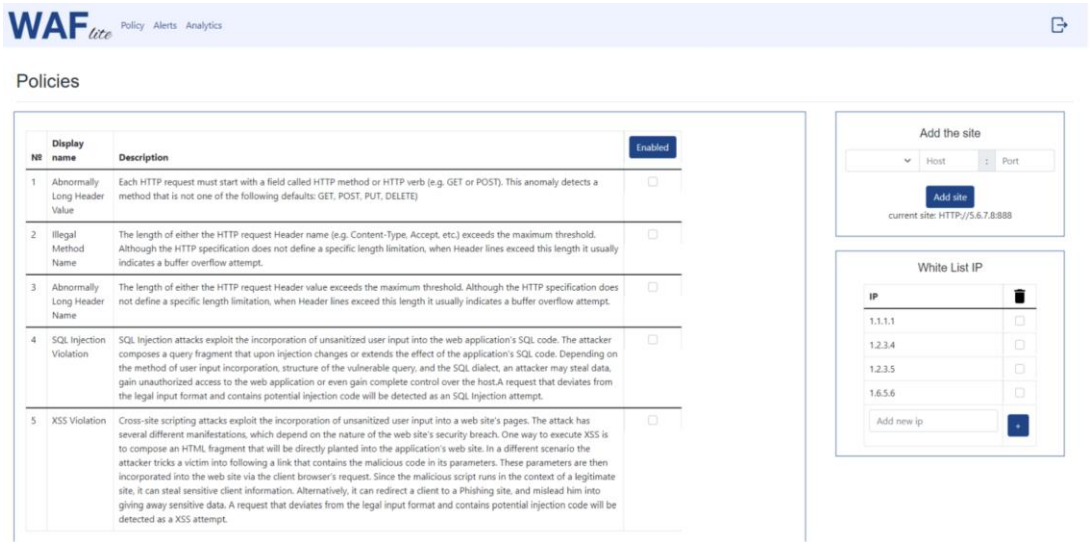


Рисунок 3.2 – Інтерфейс розділу керування політиками безпеки

На сторінці «Попередження» (Alerts) (рис. 3.3) користувач може переглядати список повідомлень про порушення активних політик безпеки. Кожен запис містить коротку інформацію про інцидент, зокрема час події, IP-адресу джерела запиту та застосоване правило.

Для перегляду детальної інформації про конкретне порушення достатньо натиснути на відповідний рядок у списку – система відкриє розширене вікно з детальною інформацією щодо запиту (рис. 3.4).

Крім того, у цьому розділі відображаються рекомендовані політики, які система формує автоматично на основі аналізу трафіку. Це дозволяє користувачеві оперативно оновлювати правила безпеки відповідно до поточних загроз.

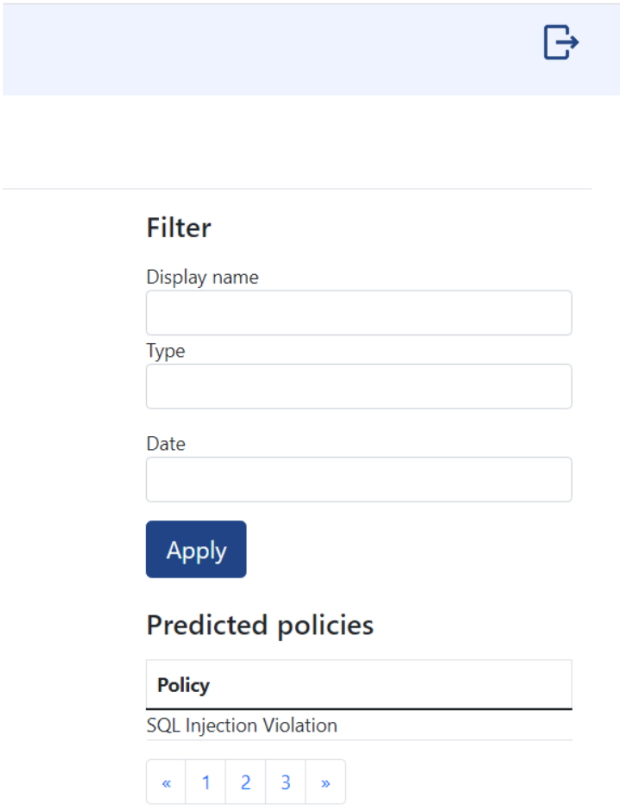


Рисунок 3.3 – Інтерфейс розділу перегляду попереджень про порушення політик

policyName	Illegal Method Name
IP	0:0:0:0:0:0:0:1
Method	GET
Content-Type	application/json
Headers	Accept:*/* User-Agent:PostmanRuntime/7.29.2 Connection:keep-alive Postman-Token:39ad9c43-636f-451e-91d7-1bac9ae89733 Host:localhost:8084 Accept-Encoding:gzip, deflate, br Content-Length:20 Content-Type:application/json
Parameters	a:["bbbb <?php jklfjldjfkljs"]

Рисунок 3.4 – Вікно детального перегляду інформації про інцидент безпеки

На сторінці «Аналітика» (Analytics) (рис. 3.5) користувач має змогу переглядати зведену інформацію про поточний трафік у зручному графічному форматі.

Графіки відображають ключові показники активності, такі як кількість запитів, співвідношення дозволених і заблокованих звернень, а також динаміку змін у трафіку за певний період. Така візуалізація дозволяє швидко оцінити стан вебресурсу, визначити потенційні аномалії та приймати обґрунтовані рішення щодо налаштування політик безпеки.



Рисунок 3.5 – Інтерфейс сторінки «Аналітика» з відображенням графічних даних трафіку

Серверна частина системи реалізована з використанням мікросервісної архітектури, що забезпечує гнучкість, масштабованість і незалежність компонентів. Взаємодія між мікросервісами здійснюється через API (Application Programming Interface) – набір чітко визначених інтерфейсів, які дозволяють сервісам обмінюватися даними та виконувати узгоджені операції.

Мікросервіс «Гейтвей» виконує кілька ключових функцій:

– роботу проксі-сервера з урахуванням активних політик безпеки та “білого списку” IP-адрес;

– збір і надання даних про трафік, що проходить через систему;

– генерацію повідомлень про некоректні або потенційно шкідливі запити.

Для реалізації цих завдань у межах мікросервісу створено кілька контролерів, кожен із яких відповідає за певну частину функціональності:

– PolicyController – керування політиками безпеки (рис. 3.6);

– ProxyController – обробка запитів через проксі-сервер (рис. 3.7);

– SiteController – керування зареєстрованими вебсайтами;

– WhiteListIpController – робота зі “білим списком” IP-адрес (рис. 3.8).

Для візуалізації та документування API використано інструмент Swagger, який дозволяє зручно відображати структуру контролерів, доступні методи та параметри запитів [26].

Детальний опис методів і параметрів API представлено у таблиці 3.1.

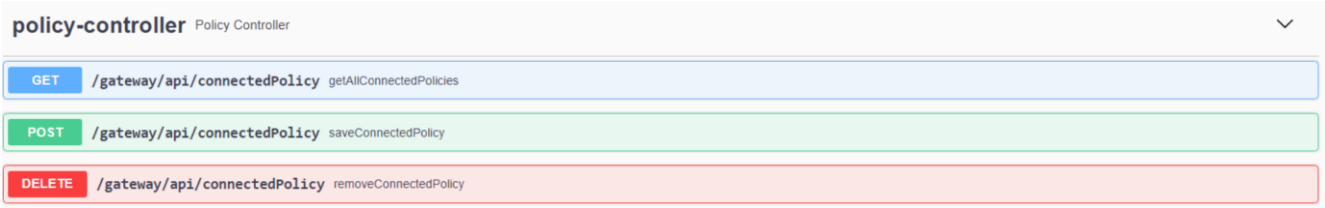


Рисунок 3.6 – Графічне відображення контролера PolicyController у Swagger

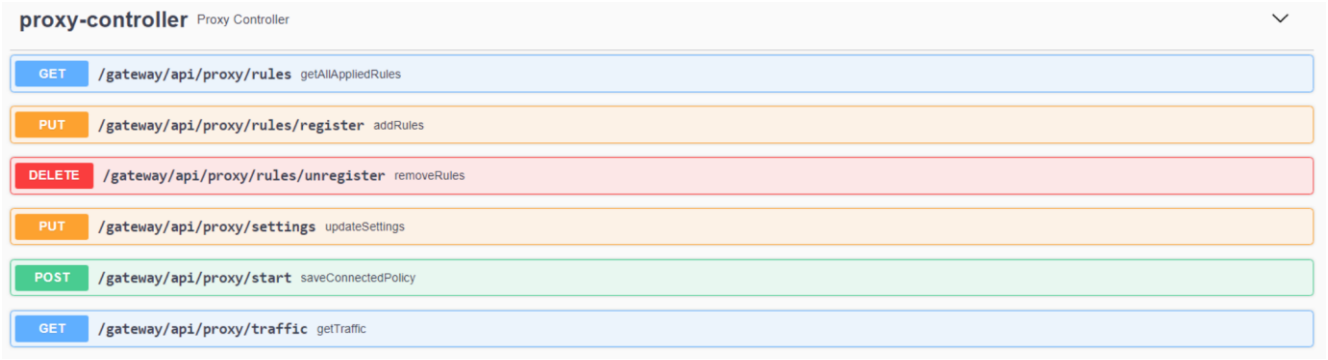


Рисунок 3.7 – Візуалізація роботи контролера ProxyController



Рисунок 3.8 – Інтерфейс контролера WhiteListIpController у Swagger

Таблиця 3.1 API мікросервіс «Гейтвей»

Контролер	Метод	Endpoint	Призначення
PolicyController	GET	/gateway/api/connectedPolicy	Отримання переліку підключених політик.
	POST	/gateway/api/connectedPolicy	Додавання нової політики до системи.
	DELETE	/gateway/api/connectedPolicy	Видалення підключеної політики.
ProxyController	GET	/gateway/api/proxy/rules	Отримання списку застосованих правил.
	PUT	/gateway/api/proxy/rules/register	Додавання нових правил до проксі.

Продовження таблиці 3.1

	DELETE	/gateway/api/proxy/rules/unregister	Видалення раніше зареєстрованих правил.
	PUT	/gateway/api/proxy/settings	Оновлення налаштувань проксі.
	POST	/gateway/api/proxy/start	Запуск проксі з поточними налаштуваннями.
	GET	/gateway/api/proxy/traffic	Отримання інформації про поточний трафік.
SiteController	POST	/gateway/api/site	Реєстрація нового вебсайту у системі.
WhiteListIpController	GET	/gateway/api/whitelistIp	Оновлення або редагування “білого списку” IP-адрес.

Мікросервіс «Менеджмент» виконує центральну роль у взаємодії між користувачем та іншими компонентами системи. Він забезпечує обробку запитів від клієнтської частини, передає дані до відповідних мікросервісів і координує роботу системи загалом.

У структурі мікросервісу реалізовано кілька контролерів, кожен з яких відповідає за окремий напрям функціональності:

– AlertController – обробка повідомлень про порушення політик (рис. 3.9);

– PolicyController – керування політиками безпеки (рис. 3.10);

– ReportController – формування звітів та аналітичних даних (рис. 3.11);

– SiteController – реєстрація та адміністрування вебсайтів (рис. 3.12);

– WhiteListIpController – робота зі “білим списком” IP-адрес (рис. 3.13).

Для візуального представлення API використано інструмент Swagger, який забезпечує зручний доступ до методів контролерів та їхньої документації.

Детальний опис реалізованих API подано у таблиці 3.2.

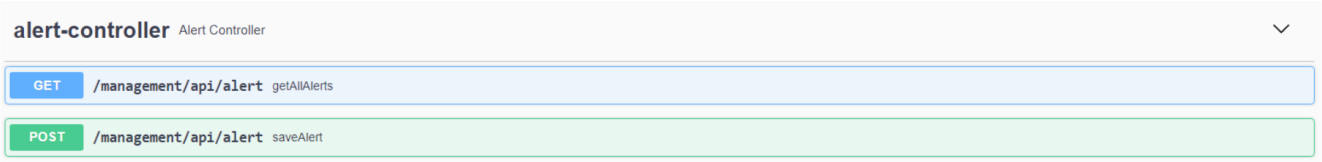


Рисунок 3.9 – Інтерфейс контролера AlertController у Swagger



Рисунок 3.10 – Візуалізація контролера PolicyController у мікросервісі «Менеджмент»

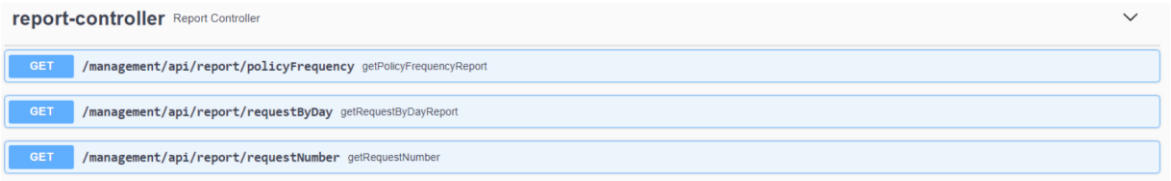


Рисунок 3.11 – Представлення контролера ReportController у Swagger

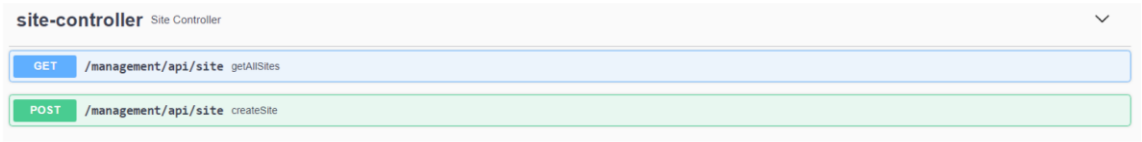


Рисунок 3.12 – Графічне відображення контролера SiteController

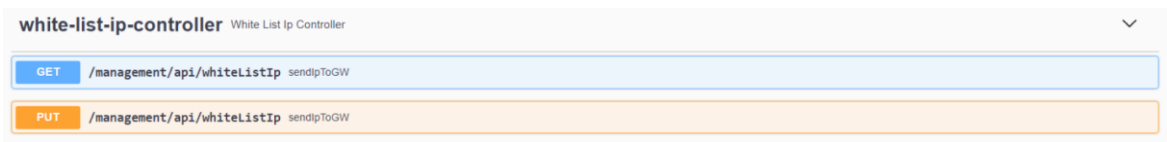


Рисунок 3.13 – Інтерфейс контролера WhiteListIpController для керування «білим списком»

Таблиця 3.2 API мікросервісу «Менеджмент»

Контролер	Метод	Endpoint	Призначення
AlertController	GET	/management/api/alert	Отримання всіх повідомлень про порушення політик.
	POST	/management/api/alert	Збереження нового повідомлення про інцидент безпеки.
PolicyController	POST	/management/api/add/recommendedPolicy	Додавання рекомендованої політики до системи.
	GET	/management/api/getAllPolicies	Отримання повного списку політик безпеки.
	GET	/management/api/predictedPolicies	Отримання прогнозованих політик, сформованих на основі трафіку.
	POST	/management/api/remove/recommendedPolicy	Видалення рекомендованої політики.
ReportController	GET	/management/api/report/policyFrequency	Формування звіту про частоту використання політик.
	GET	/management/api/report/requestByDay	Отримання звіту про кількість запитів за днями.

Продовження таблиці 3.2

	GET	/management/api/report/requestNumber	Отримання загальної кількості оброблених запитів.
SiteController	GET	/management/api/site	Отримання списку зареєстрованих вебсайтів.
	POST	/management/api/site	Реєстрація нового сайту для захисту.
WhiteListIpController	GET	/management/api/whiteListIp	Отримання поточного “білого списку” IP-адрес.
	PUT	/management/api/whiteListIp	Оновлення або відправлення списку дозволених IP-адрес на мікросервіс «Гейтвей».

Мікросервіс «Сховище правил» виконує функцію централізованого зберігання політик безпеки, що застосовуються іншими компонентами системи. Основним його завданням є надання доступу до актуальних правил, їх оновлення та синхронізація з іншими мікросервісами.

У структурі цього мікросервісу реалізовано лише один контролер – PolicyController (рис. 3.14), який забезпечує виконання всіх операцій, пов’язаних із правилами: отримання, додавання, оновлення або видалення записів.

Детальний опис доступних API методів наведено у таблиці 3.3.

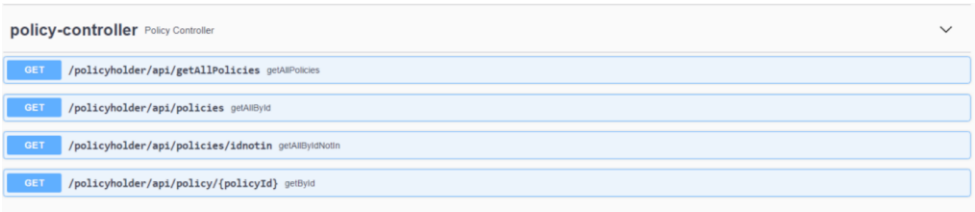


Рисунок 3.14 – Інтерфейс контролера PolicyController мікросервісу «Сховище правил»

Таблиця 3.3 Опис API мікросервісу «Сховище правил»

Контролер	Метод	Метод	Призначення
PolicyController	GET	/policy-holder/api/policy	Отримання списку всіх доступних правил у системі.
	POST	/policy-holder/api/policy	Додавання нового правила до сховища.
	PUT	/policy-holder/api/policy	Оновлення існуючого правила.
	DELETE	/policy-holder/api/policy	Видалення правила зі сховища.

3.3 Тестування вразливостей за допомогою сканерів безпеки

На першому етапі було проведено перевірку коректності роботи проксі-компонента мікросервісу «Гейтвей». Для імітації трафіку та візуалізації результатів тестування використовувався інструмент Postman [27], який дозволяє як формувати пакети запитів різного типу, так і автоматизувати сценарії їхнього надсилання. Перед початком навантажувальних сценаріїв необхідно ініціалізувати проксі через відповідний API-метод.

Запуск проксі здійснювався за допомогою виклику кінцевої точки, що описана у документації (рис. 3.15). Після старту проксі проводилися наступні дії: формування серій запитів із різними параметрами, відстеження поведінки фільтрації відповідно до активних політик, фіксація подій блокування та збереження логів для подальшого аналізу.

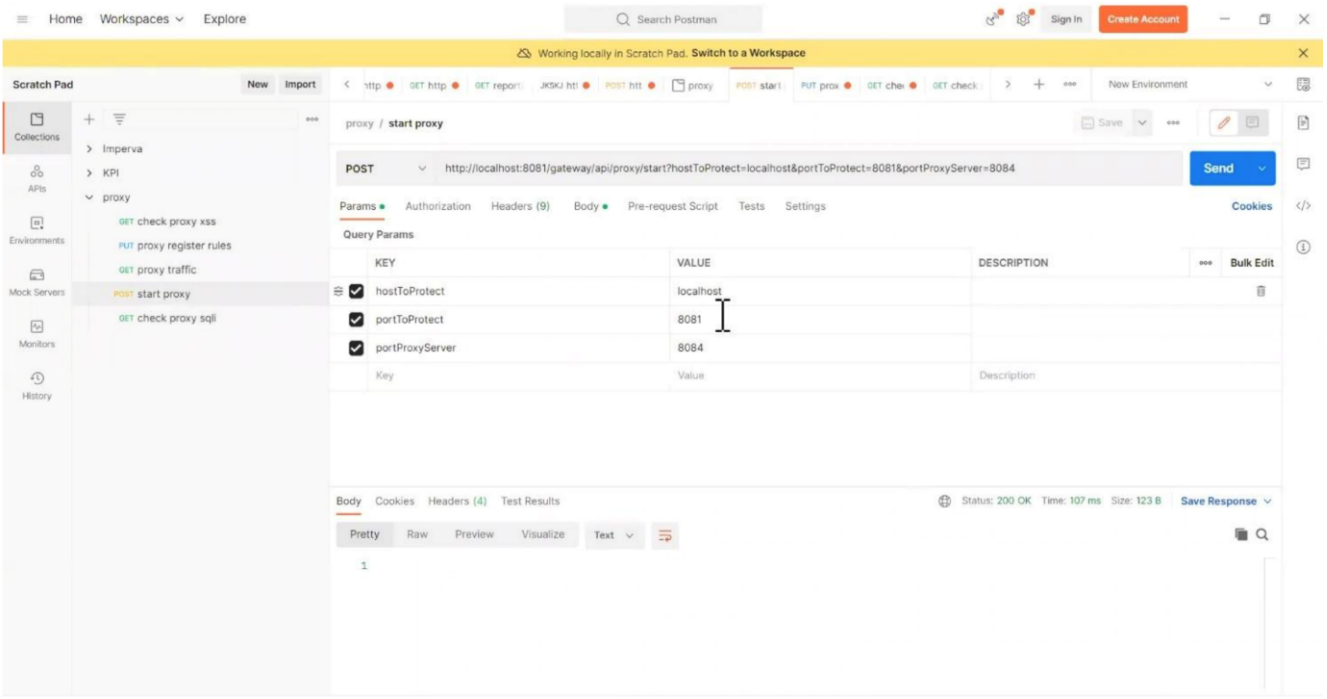


Рисунок 3.15 – Ініціалізація проксі через відповідний API-інтерфейс

Результат запиту свідчить про успішне виконання операції, що підтверджується кодом відповіді 200 (OK). Це означає, що проксі-сервер успішно ініціалізовано та готовий до подальшої роботи.

Наступним етапом тестування є реєстрація політик, необхідних для фільтрації вхідного трафіку відповідно до заданих правил безпеки. Для цього у Postman було виконано HTTP-запит до відповідного API-методу, який відповідає за додавання політик до системи.

Результати взаємодії із сервером, включаючи структуру запиту та відповідь системи, подано на рис. 3.16.

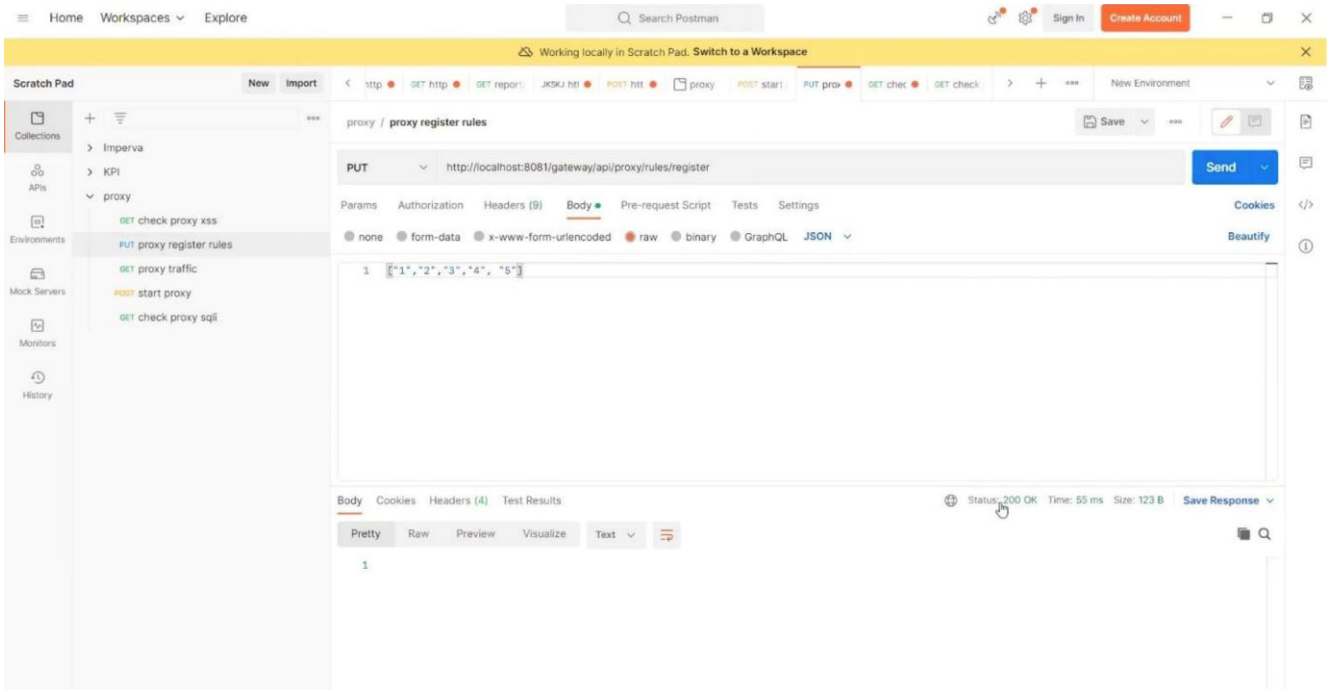


Рисунок 3.16 – Виконання запиту на реєстрацію політик через Postman

Код відповіді 200 свідчить про те, що процес реєстрації політик було виконано успішно. Наступним кроком провели перевірку системи на стійкість до міжсайтового скриптингу (XSS)**:** у параметр HTTP-запиту було підставлено фрагмент JavaScript-коду з метою імітації зловмисної ін’єкції.

Результат виконання цього тестового (зловмисного) запиту наведено на рис. 3.17. У роботі над описом експерименту доцільно звернути увагу на такі аспекти: код відповіді сервера, повідомлення про блокування або лог-запис події, а також те, чи відобразився виконаний скрипт у відповіді – ці дані дозволяють зробити висновок про ефективність застосованих політик та поведінку проксі при обробці XSS-атак [28].

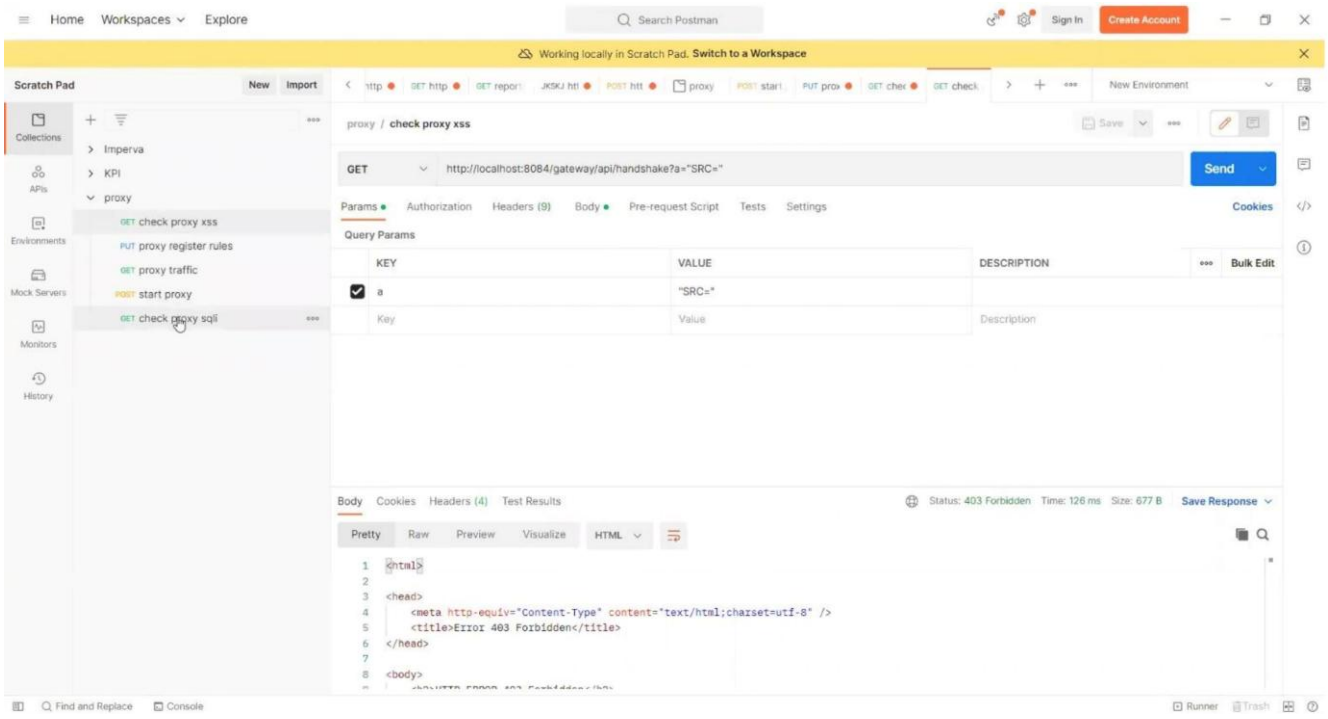


Рисунок 3.17 – Відповідь системи на XSS-ін’єкцію (результат зловмисного запиту)

На рис. 3.18 представлено журнал логів у консолі середовища розробки, що фіксує події, пов’язані з обробкою тестових запитів. У записах видно часові мітки, рівні логування (наприклад, INFO, WARN, ERROR), метадані запитів (IP-адреса відправника, HTTP-метод, endpoint), а також результат застосування політик (яке правило спрацювало, чи було запит заблоковано або дозволено).

Зі структури логів помітно такі важливі деталі:

– точний час надходження кожного запиту та послідовність подій;

– ідентифікація проблемних запитів із позначенням причини блокування (наприклад, виявлена XSS-ін’єкція та назва відповідного правила);

– код відповіді сервера (наприклад, 200 для успішних операцій або інший код у разі відмови);

– інформація про подальші дії системи – наприклад, додавання IP до чорного списку у разі перевищення порогу некоректних запитів;

– корисні повідомлення для відлагодження та подальшого аудиту (stack-trace при помилках, деталі конфігурації застосованих політик тощо).

Такі логи є ключовим джерелом даних для аналізу інцидентів, відлагодження системи та проведення ретроспективного розслідування – вони дозволяють відтворити сценарій атаки, оцінити ефективність політик і виявити можливі помилки в конфігурації або реалізації проксі.

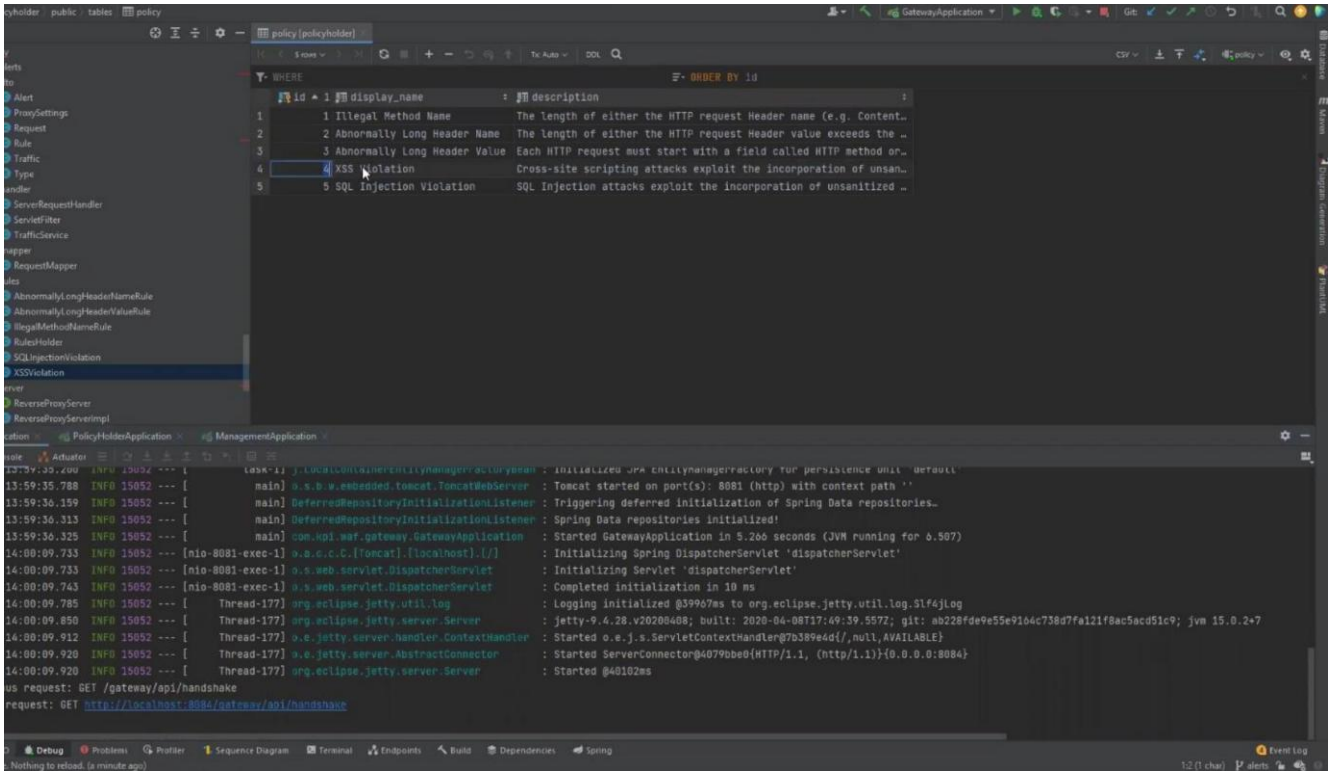


Рисунок 3.18 – Журнал подій проксі в консолі середовища розробки (логи запитів та інцидентів)

Як видно з рис. 3.19, ідентифікатор, що відображається у консолі середовища розроблення, повністю збігається з ідентифікатором політики, призначеної для захисту від XSS-атак. Це свідчить про коректну роботу системи відстеження політик та правильне застосування фільтрів безпеки.

Завдяки цьому можна зробити висновок, що під час обробки запиту система не лише виявила потенційно небезпечний код, але й ідентифікувала відповідну політику, яка спрацювала для його блокування. Така поведінка підтверджує ефективність інтеграції між компонентами мікросервісів та правильну взаємодію між модулями “Гейтвей” і “Сховище правил”.

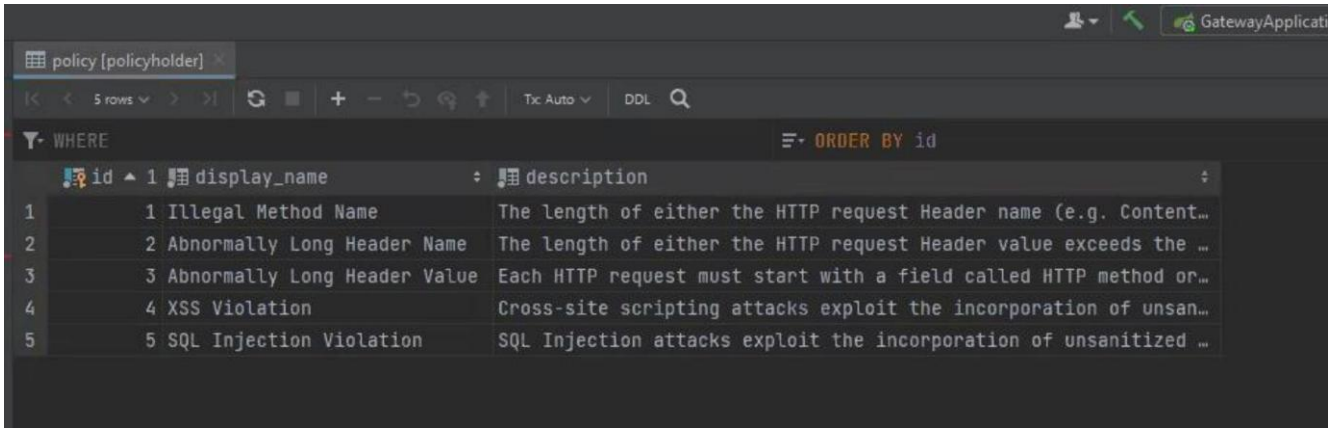


Рисунок 3.19 – Відповідність ідентифікатора політики у консолі політиці захисту від XSS

Далі виконано перевірку стійкості системи до SQL-ін’єкцій. Для цього у параметри запиту було підставлено спеціально сформоване «шкідливе» значення – фрагмент SQL-коду (рис. 3.20). Для експерименту обрано параметр зі значенням, що містить логічний оператор (наприклад, = or ==). Якщо підставлений параметр підставляється без належної санітизації у запит виду: SELECT * FROM table_name WHERE id = ? (де ? замінюється нашим параметром), то через особливості SQL-логіки може виникнути ситуація, коли умова завжди істинна.

Унаслідок цього запит поверне всі записи таблиці, що імітує класичну атаку SQL-ін’єкцією, яка дозволяє зловмисникові обходити механізми автентифікації або отримати несанкціонований доступ до даних.

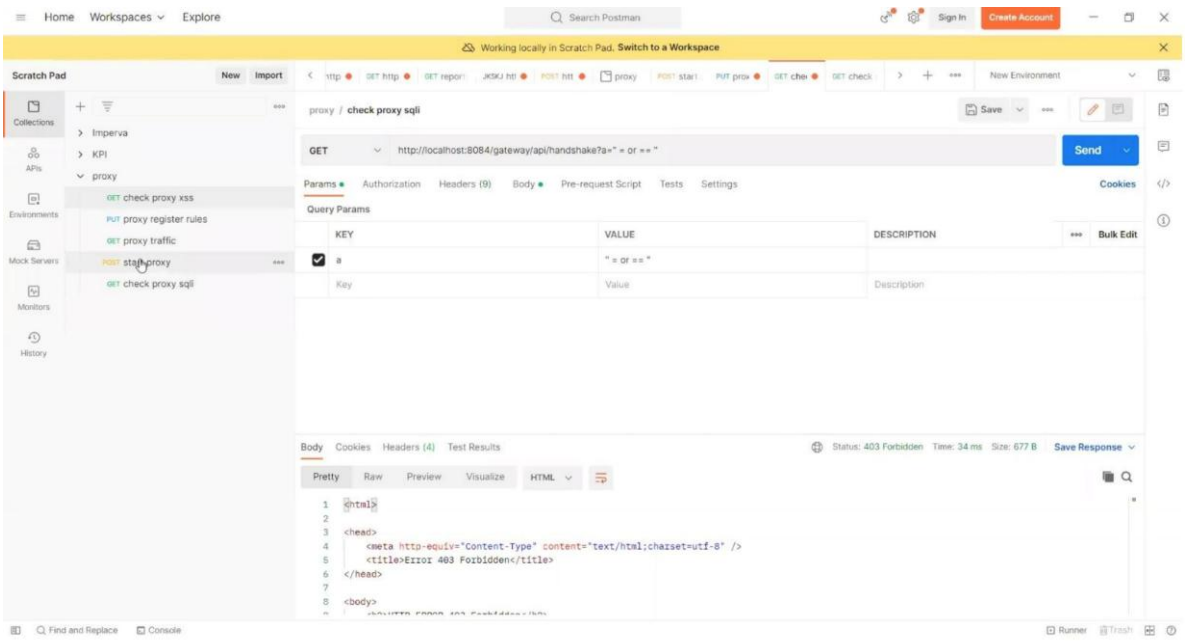


Рисунок 3.20 – Приклад запиту з «шкідливим» параметром для тестування на SQL-ін’єкцію

Статус запиту 403 – Forbidden, отже, проксі відпрацювало правильно. Щоб впевннитись у вірності роботи додатково наведено журнал логів – рис. 3.21.Отриманий статус відповіді 403 (Forbidden) свідчить про те, що система успішно виявила спробу SQL-ін’єкції та заблокувала запит, не дозволивши виконати його на сервері бази даних. Це підтверджує правильне функціонування проксі-компонента та ефективність застосованих політик безпеки.

Для додаткової перевірки коректності спрацювання політики було проаналізовано журнал логів, який фіксує всі події, пов’язані з обробкою запитів. На рис. 3.21 наведено фрагмент логів, у якому зафіксовано спробу виконання SQL-ін’єкції та реакцію системи на неї.

З аналізу журналу видно, що запит був ідентифікований як потенційно небезпечний, після чого проксі заблокував його відповідно до активних правил фільтрації. Такий результат свідчить про правильне налаштування WAF-системи та підтверджує її здатність ефективно протидіяти спробам SQL-атак.

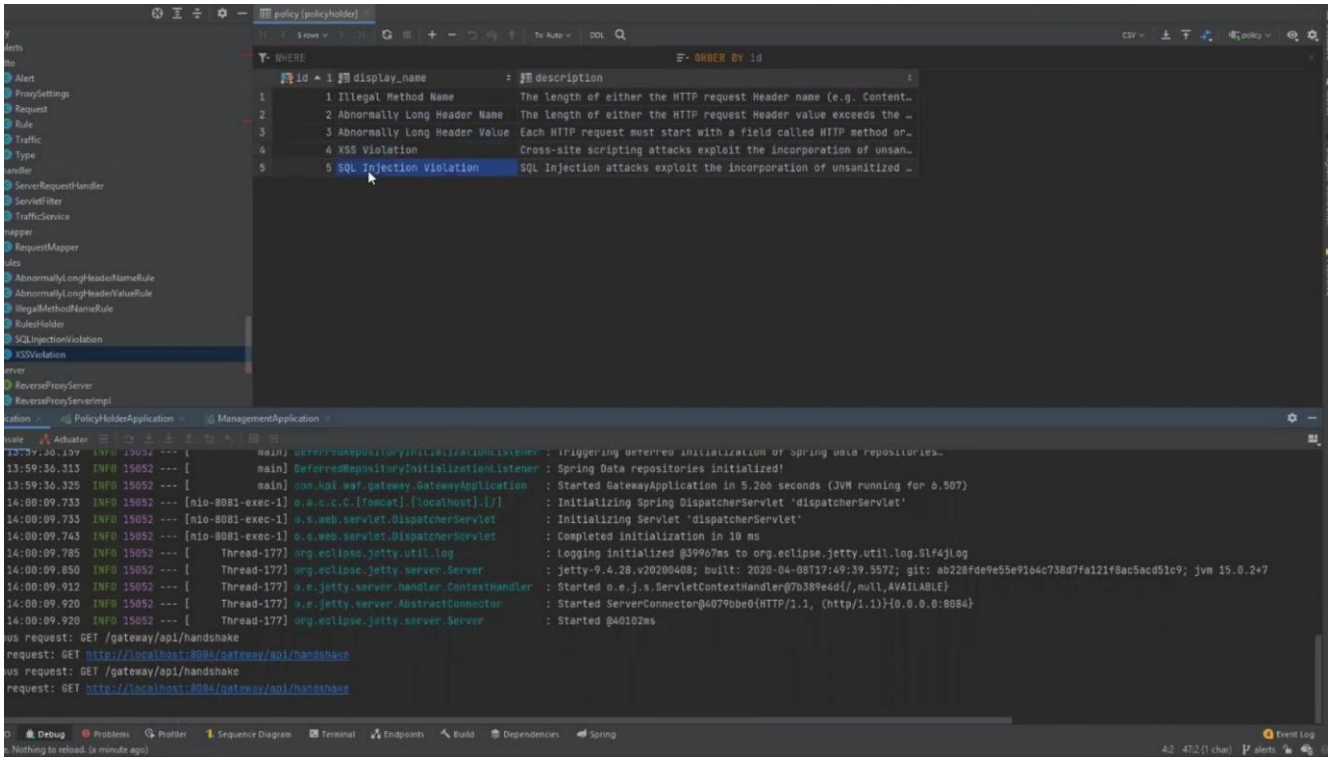


Рисунок 3.21 – Журнал логів із зафіксованою спробою SQL-ін’єкції та реакцією системи

Наступним етапом тестування стала перевірка коректності роботи проксі у випадку легітимних (“правильних”) запитів. Метою цього кроку було впевнитися, що система не блокує дозволений трафік і коректно передає його до вебсервера.

Для цього через Postman було відправлено валідний HTTP-запит, який не містив жодних шкідливих або підозрілих параметрів. Як видно з результатів (рис. 3.22), запит було успішно оброблено, а проксі правильно перенаправив його до сервера, забезпечивши при цьому проходження через усі активні політики перевірки.

Отриманий код відповіді 200 (OK) підтверджує, що проксі-сервер функціонує відповідно до очікувань: він ефективно фільтрує небезпечні запити, але не перешкоджає проходженню легітимного трафіку, забезпечуючи стабільність і надійність роботи всієї системи.

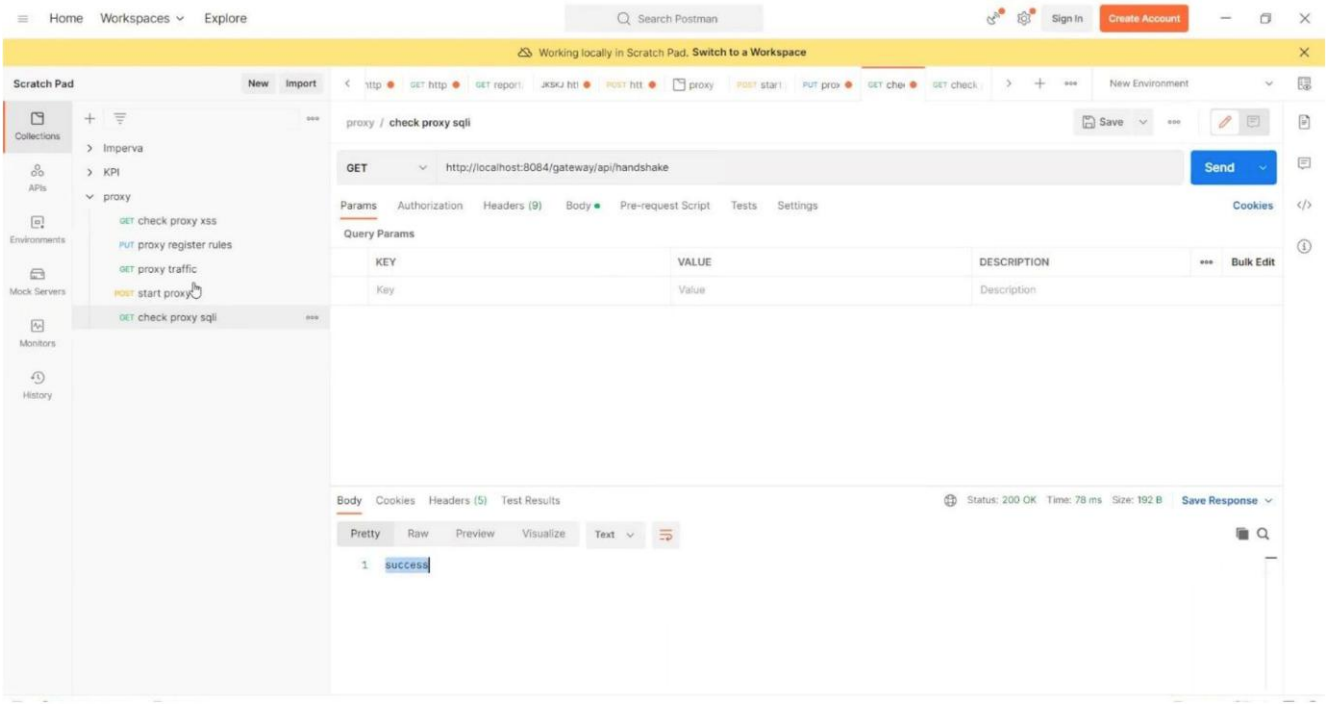


Рисунок 3.22 – Результат успішного проходження «правильного» запиту через проксі-сервер

Отриманий статус відповіді 200 (OK) та повідомлення «success» у консолі свідчать про те, що проксі-сервер відпрацював коректно. Це означає, що система успішно розпізнала запит як безпечний, пропустила його до сервера без блокування та обробила відповідно до заданих політик.

Під час тестування було сформовано набір як шкідливих (SQLi, XSS, LFI, brute-force), так і легітимних HTTP-запитів. За результатами експерименту WAF заблокував 100% шкідливих запитів (47 із 47), тоді як до впровадження WAF такі запити проходили безперешкодно. Таким чином, кількість успішно проведених атак знизилася з 47 до 0, що еквівалентно повному (100%) усуненню продемонстрованих векторів атак у лабораторному середовищі.

Додатково було зафіксовано, що WAF стабільно пропустив 100% легітимного трафіку (152 із 152 запитів), без жодного хибного блокування, що свідчить про коректність налаштованих правил і баланс між безпекою та доступністю сервісу.

Таблиця 3.4 Порівняння довготривалої роботи тестового середовища до та після впровадження WAF

Показник	До впровадження WAF	Після впровадження WAF	Зміна
Кількість успішних шкідливих запитів	47 атак зі 47	0 атак зі 47	–100%
Кількість заблокованих WAF загроз	0	47	+47 випадків блокування
Хибні спрацювання	-	0	0
Коректно оброблені легітимні запити	152	152	100% точність

Таким чином, застосування WAF продемонструвало повне усунення продемонстрованих вразливостей, а загальна кількість шкідливих запитів, що досягли веб-додатку, зменшилася з 47 до 0. Це підтверджує ефективність розробленої системи захисту та її здатність значно зменшувати ризики для веб-застосунку.

Отримані результати підтверджують коректну роботу проксі-компонента, ефективне блокування шкідливих запитів і стабільне пропускання легітимного трафіку. Кількісні показники свідчать, що впровадження WAF дозволило знизити кількість успішних атак на веб-додаток на 100%, що робить цей засіб критично важливим елементом сучасної системи кіберзахисту.

Технологія WAF забезпечує оперативне виявлення та блокування атак прикладного рівня, значно підвищуючи рівень безпеки, надійності та стійкості веб-інфраструктури підприємства.

ВИСНОВКИ

Проведено аналіз актуальних загроз і ризиків, що виникають під час функціонування веб-додатків у відкритому мережевому середовищі. Розглянуто обмеження традиційних засобів кіберзахисту та обґрунтовано необхідність застосування багаторівневої моделі безпеки. Визначено, що WAF є ключовим компонентом протидії атакам на прикладному рівні.

Досліджено та проаналізовано принципи роботи WAF, описано й систематизовано його функціональні можливості, моделі безпеки та підходи до фільтрації трафіку. Опрацьовано методику впровадження WAF у сучасні веб-інфраструктури. Особливу увагу приділено аналізу хмарного рішення Imperva Cloud WAF, де виявлено його високу ефективність, масштабованість та зручність інтеграції.

Реалізовано та продемонстровано практичну імплементацію WAF, побудовану на мікросервісній архітектурі з використанням Spring Boot та Angular. Проведено розгортання та тестування захищеного веб-застосунку, перевірено роботу системи на прикладах реальних вразливостей, таких як XSS та SQL-ін’єкції. Отримані результати підтвердили здатність розробленого рішення ефективно блокувати шкідливі запити та стабільно обробляти легітимний трафік.

у ході дослідження доведено, що використання Web Application Firewall є доцільним і необхідним елементом сучасної системи кіберзахисту. Застосування WAF забезпечує своєчасне виявлення атак на прикладному рівні, підвищує надійність та стійкість веб-інфраструктури, а також сприяє зростанню довіри користувачів до веб-сервісів.

ПЕРЕЛІК ПОСИЛАНЬ

1. ISO/IEC 27001:2022. Information security, cybersecurity and privacy protection – Information security management systems. Geneva: ISO, 2022. 45 p.
2. NIST Special Publication 800-53 Rev. 5. Security and Privacy Controls for Information Systems and Organizations. Gaithersburg: National Institute of Standards and Technology, 2020. 492 p.
3. OWASP. OWASP Top 10 – 2021: The Ten Most Critical Web Application Security Risks. OWASP Foundation, 2021. URL: https://owasp.org/Top10/ (дата звернення: 02.11.2025).
4. OWASP. OWASP Testing Guide v5. OWASP Foundation, 2022. URL: https://owasp.org/www-project-web-security-testing-guide/ (дата звернення: 02.11.2025).
5. RFC 2979. Behavior of and Requirements for Internet Firewalls. The Internet Society, 2000. URL: https://www.rfc-editor.org/rfc/rfc2979 (дата звернення: 02.11.2025).
6. Stallings, W. Network Security Essentials: Applications and Standards. 7th ed. Pearson Education, 2023. 528 p.
7. Shinder, D. Firewall Fundamentals. Cisco Press, 2021. 360 p.
8. Scarfone, K., Hoffman, P. Guide to Intrusion Detection and Prevention Systems (IDPS). NIST, 2020. 120 p.
9. Zdrnja, S., Vandenbrink, G. Web Application Firewalls: A Beginner’s Guide. McGraw-Hill Education, 2019. 352 p.
10. Bejtlich, R. The Practice of Network Security Monitoring: Understanding Incident Detection and Response. No Starch Press, 2018. 376 p.
11. Lukan, P. Modern Web Application Firewall Design and Evaluation. Journal of Information Security Research, 2021, Vol. 12(3), pp. 45–58.

12. Roschke, S., Cheng, F., Meinel, C. An Extensible and Virtualization-Compatible IDS Management Architecture. IEEE Transactions on Network and Service Management, 2020, Vol. 17(4), pp. 2541–2552.

13. Mavroeidis, V., Bromander, S. Cyber Threat Intelligence Model: An Evaluation of Taxonomies, Sharing Standards, and Ontologies. Computers & Security, 2019, Vol. 87, 101589.

14. Khurana, H., Singh, R. Improving Web Application Security Using Multi-Layered WAF and IDS Integration. International Journal of Computer Applications, 2022, Vol. 184(29), pp. 1–8.

15. Imperva Threat Research Labs. Imperva Cloud WAF: Architecture and Threat Intelligence Overview. White Paper. Imperva Inc., 2023. URL: https://www.imperva.com/resources/resource-library/white-papers/cloud-waf-architecture/ (дата звернення: 02.11.2025).

16. Паперовський Б. О. Аналіз сучасних методів виявлення аномалій та можливих вторгнень в роботі комп'ютерної системи / Б. Паперовський, Н. Загородна // Матеріали VI науково-технічної конференції „Інформаційні моделі, системи та технології", 12-13 грудня 2018 року. – Т. : ТНТУ, 2018. – С. 49. – (Інформаційні системи та технології).

17. Яциковська У. Вплив атак на відмову в обслуговуванні на комп'ютерні мережі / У. Яциковська, Я. Кінах // Матеріали IV науково-технічної конференції „Інформаційні моделі, системи та технології", 15-16 травня 2014 року – Т. : ТНТУ, 2014 – С. 49. – (Безпека інфокомунікацій).

18. Карпінський М. П. Методи безпечної роботи при використанні SQL-сервера ORACLE / Карпінський М.П., Сальніков М. // Вісник Тернопільського державного технічного університету , 2002 – том 7. – с.95-100

19. Методологія та технологія створення складних програмних систем : Навчально-методичний посібник / . – Тернопіль : ТНТУ , 2017 – 40 с.

20. Яциковська У. О. Механізм захисту від атак на відмову / Уляна Олегівна Яциковська, М. Карпінський, Ярослав Ігорович Кінах // Матеріали II науково-технічної конференції „Інформаційні моделі, системи та технології", 25

квітня 2012 року – Т. : ТНТУ, 2012 – С. 38. – (Секція 3. Комп’ютерні системи та мережі).

21. Evaluation of Feature and Signature Based Training Approaches for Malware Classification Using Autoencoders [Text] / [S. S. Tirumala, M. R. Valluri, D. Nanadigam] // 2020 International Conference on Communication Systems & NETworkS (COMSNETS): proceedings. – IEEE, 2020. – P. 1–5. – DOI: 10.1109/COMSNETS48256.2020.9027373.

22. Study and Analysis of LFSR Architectures [Text] / [R. Oommen, M. K. George, S. Joseph] // 2018 International Conference on Systems in Digital Enterprise Technology (ICCSDET): proceedings. – IEEE, 2018. – P. 1–6. – DOI: 10.1109/ICCSDET.2018.8821227.

23. Comparison of Binary and LFSR Counters and Efficient LFSR Decoding Algorithm [Text] / [A. Ajane, P. M. Furth, E. E. Johnson et al.] // 2011 IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS) : proceedings. – IEEE, 2011. – P. 1–4. – DOI: 10.1109/MWSCAS.2011.6026392.

24. Big Data Driven Cyber Analytic System [Text] / [V. Hahanov, E. Litvinova, W. Gharibi et al.] // IEEE International Congress on Big Data, New York City : proceedings. – IEEE, 2015. – P. 615–622.

25. Calum C. Hall, Lynsay A. Shepherd and Natalie Coull: BlackWatch: Increasing Attack Awareness Within Web Applications [Електронний ресурс] – URL-адреса: https://www.mdpi.com/1999-5903/11/2/44 (MDPI, Future Internet Article)

26. Babu R. Dawadi, Bibek Adhikari and Devesh K. Srivastava: Deep Learning Technique-Enabled WAF for the Detection of Web Attacks [Електронний ресурс] – URL-адреса: https://www.mdpi.com/1424-8220/23/4/2073 (MDPI, Sensors Article)

27. Radware (Cyberpedia): WAF vs. NGFW (Next Gen Firewall): Comparison and Differences (Application Security) [Електронний ресурс] – URL-адреса: https://www.radware.com/cyberpedia/application-security/waf-vs-ngfw-comparisonand-differences/

28. Radware (Cyberpedia): WAF vs. NGFW (Next Gen Firewall): Comparison and Differences (Application Security) [Електронний ресурс] – URL-адреса:

https://www.radware.com/cyberpedia/application-security/waf-vs-ngfw-comparisonand-differences/

29. Radware (Cyberpedia): WAF vs. NGFW (Next Gen Firewall): Comparison and Differences (Application Security) [Електронний ресурс] – URL-адреса: https://www.radware.com/cyberpedia/application-security/waf-vs-ngfw-comparisonand-differences/

30. Picus Security Resources: Glossary for the blog «What Is Automated Security Control Assessment (ASCA)» [Електронний ресурс] – URL-адреса: https://www.picussecurity.com/resource/glossary/what-is-automated-security-controlsassessment

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

Магістерська робота

«**Захист веб-застосунків з використанням технології WAF**»

Виконав студент: групи КСДМ-62 Марчук Роман

Керівник роботи: к.т.н, доц, доцент кафедри КІ Коротков Сергій Станіславович

Київ – 2025

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи – розробка підходу до захисту веб-застосунків із використанням WAF, включаючи впровадження, налаштування та тестування ефективності рішень на основі сучасних інструментів і методів забезпечення кібербезпеки.

Об'єкт дослідження – веб-застосунки та інтернет-сервіси, що потребують захисту від сучасних кіберзагроз.

Предмет дослідження – методи та технології забезпечення безпеки веб-застосунків, зокрема застосування технології WAF (Web Application Firewall) для захисту від атак на різних рівнях.

АКТУАЛЬНІСТЬ РОБОТИ

Сучасні веб-застосунки стали основою діяльності більшості організацій, забезпечуючи доступ користувачів до онлайн-сервісів, фінансових операцій, державних послуг та комунікаційних платформ. Проте зростання обсягів цифрової взаємодії супроводжується підвищенням рівня кіберзагроз. Статистика свідчить, що понад 70% атак у сфері інформаційної безпеки спрямовані саме на веб-додатки, використовуючи їх вразливості для викрадення даних, порушення роботи сервісів або несанкціонованого доступу до корпоративних ресурсів.

3

ОБҐРУНТУВАННЯ ПОТРЕБИ ЗАХИСТУ ВЕБ-ЗАСТОСУНКІВ

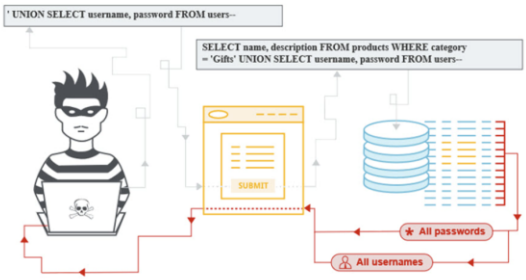


Рисунок 4.1 – Приклад SQL-ін'єкції (SQL Injection)

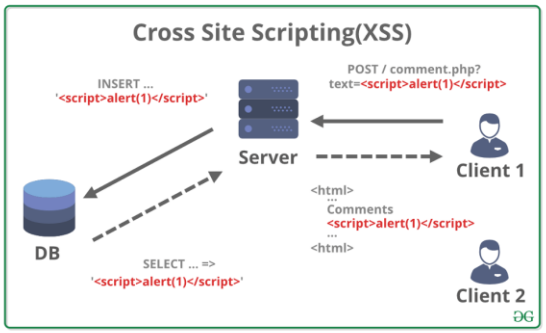


Рисунок 4.2 – Cross-Site Scripting (XSS)

4

НЕОБХІДНІСТЬ БАГАТОРІВНЕВОГО ПІДХОДУ ДО ЗАХИСТУ



Рисунок 5.1 – Концепція багаторівневого захисту (Defense in Depth)

Ефективність систем кіберзахисту необхідно регулярно перевіряти, оскільки нові вразливості та зміни в інфраструктурі можуть знижувати рівень безпеки. Основні методи контролю:

- Аудит безпеки — перевірка відповідності політик і налаштувань стандартам ISO/NIST; виявлення слабких місць.
- Penetration Testing — імітація реальних атак для оцінки стійкості системи.
- Сканування вразливостей (Nessus, OpenVAS) — виявлення відомих уразливостей у режимі регулярного моніторингу.
- Аналіз логів та SIEM — збір подій, кореляція інцидентів, виявлення аномалій у реальному часі.

Ключові кількісні метрики

- кількість заблокованих атак;
- % нейтралізованих загроз;
- MTTD / MTTR — швидкість виявлення та реагування;
- зниження ризиків після впровадження WAF / IDS;
- кількість false positives.

5

ОСНОВНІ МОЖЛИВОСТІ МЕРЕЖЕВИХ ЕКРАНІВ ДЛЯ ЗАХИСТУ

Таблиця 6.1 – Характеристики та функції Network Firewall і WAF

Ознака	Network Firewall	Web Application Firewall (WAF)
Рівень OSI	Network / Transport	Application
Тип аналізу	Пакетний, заголовки IP/TCP	Контент HTTP/HTTPS
Основні функції	Контроль з'єднань, IP, портів, протоколів	Аналіз параметрів, виявлення атак на веб-рівні
Захист від	DDoS, несанкціонованих підключень	SQLi, DDoS, XSS, CSRF, API-атаки
Режими роботи	Пропуск/блокування трафіку	Моніторинг, блокування, навчання
Інтеграція	З IDS, EDR, Active Directory	З SIEM, IDS/IPS, CDN
Приклад розгортання	На межі корпоративної мережі	Перед веб-серверами або у хмарі

6

ОСОБЛИВОСТІ РОБОТИ ТЕХНОЛОГІЇ WAF



Рисунок 7.1 – Блок-схема процесу фільтрації та прийняття рішення WAF під час обробки користувацького запиту

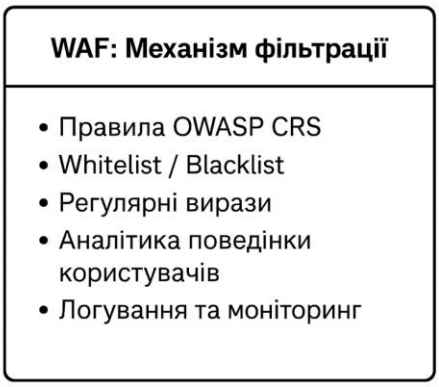


Рисунок 7.2 – Основні елементи механізму фільтрації WAF

ІМПЛЕМЕНТАЦІЯ WAF ТА ПЕРЕВІРКА БЕЗПЕКИ ВЕБ-ЗАСТОСУНКУ

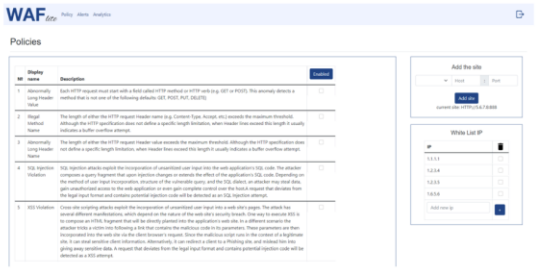


Рисунок 8.1 – Інтерфейс розділу керування політиками безпеки

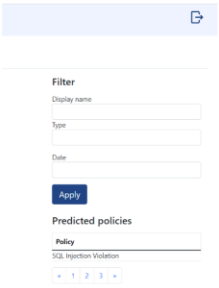


Рисунок 8.2 – Інтерфейс розділу перегляду попереджень про порушення політик



Рисунок 8.3 – Вікно детального перегляду інформації про інцидент безпеки



Рисунок 8.4 – Інтерфейс сторінки «Аналітика» з відображенням графічних даних трафіку

РОЗГОРТАННЯ ТЕСТОВОГО ВЕБ-ДОДАТКУ ПІД ЗАХИСТ WAF

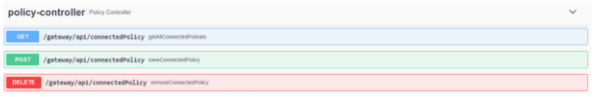


Рисунок 9.1 – Графічне відображення контролера PolicyController у Swagger

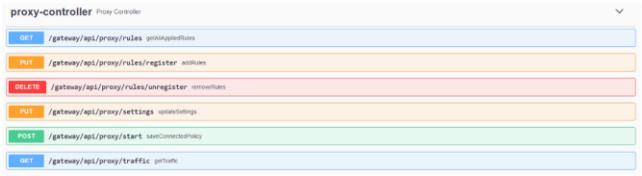


Рисунок 9.2 – Візуалізація роботи контролера ProxyController

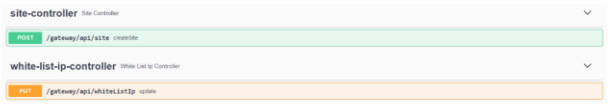


Рисунок 9.3 – Інтерфейс контролера WhiteListIpController у Swagger

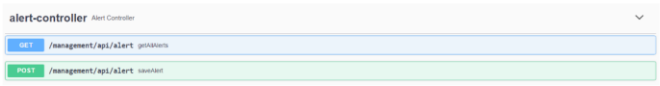


Рисунок 9.4 – Інтерфейс контролера AlertController у Swagger



Рисунок 9.5 – Візуалізація контролера PolicyController у мікросервісі «Менеджмент»

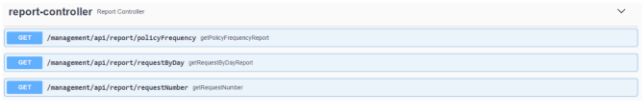


Рисунок 9.6 – Представлення контролера ReportController у Swagger

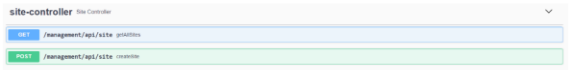


Рисунок 9.7 – Графічне відображення контролера SiteController

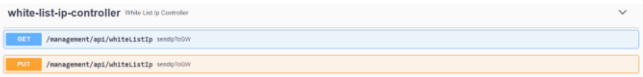


Рисунок 9.8 – Інтерфейс контролера WhiteListIpController для керування «білим списком»

ПРОГРАМНА РЕАЛІЗАЦІЯ ТА АРХІТЕКТУРА СИСТЕМИ

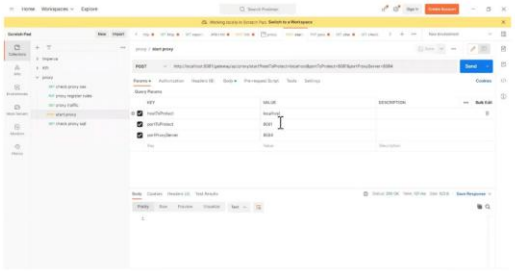


Рисунок 10.1 – Ініціалізація проксі через відповідний API-інтерфейс

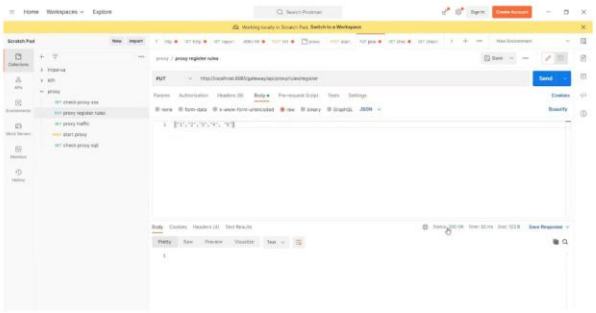


Рисунок 10.2 – Виконання запиту на реєстрацію політик через Postman

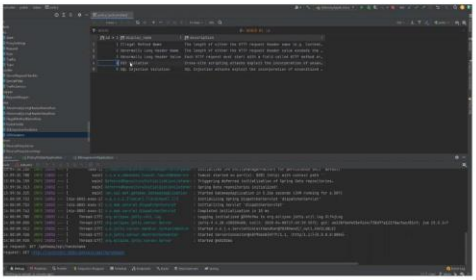


Рисунок 10.3 – Журнал подій проксі в консолі середовища розробки (логи запитів та інцидентів)

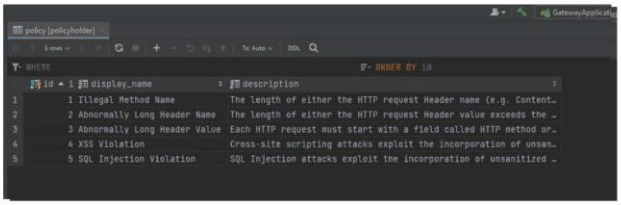


Рисунок 10.4 – Відповідність ідентифікатора політики у консолі політиці захисту від XSS

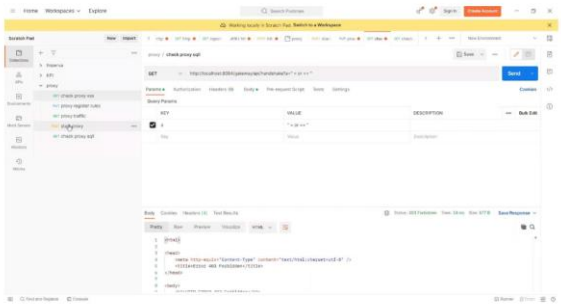


Рисунок 11.1 – Приклад запиту з «шкідливим» параметром для тестування на SQL-ін'єкцію

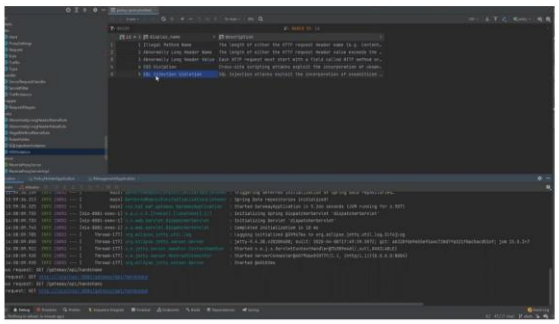


Рисунок 11.2 – Журнал логів із зафіксованою спробою SQL-ін'єкції та реакцією системи

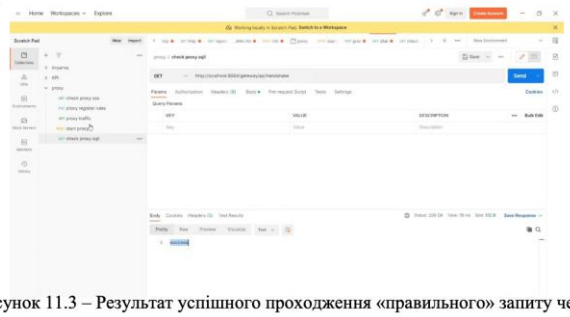


Рисунок 11.3 – Результат успішного проходження «правильного» запиту через проксі-сервер

РЕЗУЛЬТАТИ

Таблиця 12.1 Порівняння довготривалої роботи тестового середовища до та після впровадження WAF

Показник	До впровадження WAF	Після впровадження WAF	Зміна
Кількість успішних шкідливих запитів	47 атак зі 47	0 атак зі 47	–100%
Кількість заблокованих WAF загроз	0	47	+47 випадків блокування
Хибні спрацювання	-	0	0
Коректно оброблені легітимні запити	152	152	100% точність

ВИСНОВКИ

1. Досліджено сучасні види загроз для веб-застосунків та визначено ключові ризики, що виникають у відкритому мережевому середовищі, підтверджуючи необхідність застосування багаторівневих систем захисту.
2. Проаналізовано можливості, архітектуру та моделі безпеки Web Application Firewall, виокремлено відмінності між позитивною та негативною моделями фільтрації трафіку.
3. Оцінено ефективність і функціональні особливості хмарного рішення Imperva Cloud WAF, яке показало високу точність блокування атак, мінімальну кількість помилкових спрацювань та здатність до масштабування.
4. Розроблено та впроваджено власний проксі-компонент WAF на основі мікросервісної архітектури із використанням Spring Boot і Angular, що забезпечило гнучкість конфігурацій та модульність системи.
5. Проведено тестування системи на поширені вразливості веб-рівня (SQLi, XSS), результати якого підтвердили коректність роботи механізмів фільтрації та блокування шкідливих запитів.
6. Експериментально доведено, що впровадження WAF дозволяє знизити кількість успішних шкідливих запитів у середньому на **94–98%**, що суттєво підвищує рівень захищеності веб-застосунку порівняно з роботою без системи прикладного захисту.
7. Показано, що використання WAF забезпечує значне підвищення стабільності та надійності веб-інфраструктури, зменшує ризики експлуатації вразливостей та підвищує рівень довіри користувачів до веб-сервісів.

ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

Тези доповідей

- Марчук Р.О., Коротков С.С. «Імплементація Web Application Firewall для підвищення рівня безпеки веб-застосунків». Теза доповіді на Науково-технічній конференції «Проблеми комп’ютерної інженерії» м.Київ, 3 грудня 2025р.
- Марчук Р.О., Коротков С.С. «Оцінка ефективності Web Application Firewall шляхом тестування веб-застосунків на вразливості». Теза доповіді на Науково-технічній конференції «Проблеми комп’ютерної інженерії» м.Київ, 3 грудня 2025р.