Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Метод підвищення продуктивності веб-застосунків з використанням WAF

(тема)

Виконав:
здобувач 4 року навчання,
групи КІУКІ-21-5
Анастасія КОРЯКІНА
(власне ім'я, прізвище)

Спеціальність
«Комп'ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп'ютерна інженерія
(повна назва освітньої програми)

Керівник: ас. Ірина ЧЕПУРНА
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп'ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Корякіній Анастасії Михайлівні
(прізвище, ім'я, по батькові)

1. Тема роботи Метод підвищення продуктивності веб-застосунків з використанням WAF

затверджена наказом по університету від "26" травня 2025 р. № 424 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 17 червня 2025 р.

3. Вхідні дані до роботи 1) міжмережні екрани: типи та призначення; 2) технології для розгортання вебзастосунків: віртуалізація, контейнеризація; 3) методи забезпечення продуктивності вебзастосунків: розподіл навантаження, кешування; 4) методи експериментального та аналітичного моделювання роботи вебзастосунку.

4. Перелік питань, що потрібно опрацювати у роботі

1) аналіз проблеми та огляд існуючих рішень;

2)методи забезпечення продуктивності вебзастосунків;

3) типи між мережних екранів для захисту комп'ютерних мереж;

4) організація сегменту вебзастосунку з використанням WAF;

5) тестування вебзастосунку;

6) висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Слайд-презентація – 11 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз проблеми та огляд існуючих рішень	27.05.25-30.05.25	
2	Вибір технології розробки та інструменталь них засобів	31.05.25-03.06.25	
3	Розробка та тестування методу	04.06.25-06.06.25	
4	Оформлення матеріалів кваліфікаційної роботи	07.06.25-10.06.25	
5	Подання кваліфікаційної роботи керівникові	11.06.25-12.06.25	
6	Підготовка презентації та попередній захист	12.06.2025 – 13.06.2025	
7	Подання кваліфікаційної роботи на рецензування	14.06.25-16.06.25	

Дата видачі завдання " 26 " травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ас. Ірина ЧЕПУРНА
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 63 с., 8 рис., 2 табл., 4 дод., 45 джерел.

ВЕБЗАСТОСУНОК, WAF, ПРОДУКТИВНІСТЬ, ПРОКСІ-СЕРВЕР, БЕЗПЕКА, ЗАТРИМКА, БРАНДМАУЕР

Мета кваліфікаційної роботи полягає в розробці методу підвищення продуктивності вебзастосунків з використанням WAF, який забезпечує мінімізацію затримок в умовах динамічної зміни навантаження, що дозволяє підтримувати високу продуктивність додатку за підвищених вимог до захисту конфіденційної інформації та зменшення ризиків витоку даних та несанкціонованого доступу до ресурсів вебзастосунку.

В ході виконання кваліфікаційної роботи було проаналізовано сучасні підходи до забезпечення стабільної та безперервної роботи вебзастосунків в умовах змінного навантаження та зростаючих кіберзагроз. Особливу увагу в роботі приділено застосуванню фаєрволу вебзастосунків для фільтрації вхідного трафіку, що дозволяє досягти високого рівня безпеки, а також методам підвищення продуктивності, зокрема впровадженню кешування, для зменшення затримок під час обробки великої кількості запитів. Отримані результати підтверджують доцільність використання запропонованого підходу для підвищення ефективності роботи вебзастосунків.

ABSTRACT

Bachelor`s thesis: 63 pages, 8 figures, 2 tables, 4 appendices, 45 sources.

FIREWALL, GATE, INTERNET, PROTOCOL, ROUTER, SERVER, WI-FI, WIRELESS NETWORK, WLAN.

The major goal of this thesis is to develop a method of increasing the productivity of web -applications using WAF, which provides minimization of delays in the conditions of dynamic load change, which allows to maintain high performance of the application under increased requirements for the protection of confidential information and reducing the risk of data leakage and unauthorized access to resources.

In order to ensure stable and continuous work of web applications, modern approaches are being analysed to increase the efficiency of web applications in the face of variable load and growing cyber threats. Particular attention is paid to the use of web-filtration firewall for input traffic filtration, which allows achieving a high level of safety, as well as methods of improving productivity, in particular, the introduction of caching to reduce delays during processing of a large number of requests. The results confirmed the expediency of using the proposed approach to increase the efficiency of web applications.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ОС – операційна система

AWS – хмарна платформа Amazon Web Services

CSS – каскадні таблиці стилів (англ. Cascading Style Sheets)

DNS – система доменних імен (англ. Domain Name System)

DoS – атака «відмова в обслуговуванні» (англ. Denial of Service)

DPI – глибока інспекція пакетів (англ. Deep Packet Inspection)

FIFO – перший прийшов — перший обслуговується (англ. First In, First Out)

FTP – протокол передавання файлів (англ. File Transfer Protocol)

GCP – хмарна платформа Google (англ. Google Cloud Platform)

HTTP – протокол передавання гіпертексту (англ. HyperText Transfer Protocol)

HTTPS – захищений протокол передавання гіпертексту (англ. HyperText Transfer Protocol Secure)

IaaS – інфраструктура як послуга (англ. Infrastructure as a Service)

IBM – корпорація International Business Machines

IDS/IPS – система виявлення/запобігання вторгненням (англ. Intrusion Detection/Prevention System)

IIS – служби інформації Інтернету (англ. Internet Information Services)

IP – інтернет-протокол (англ. Internet Protocol)

NGFW – міжмережний екран нового покоління (англ. Next-Generation Firewall)

OSI – еталонна модель взаємодії відкритих систем (англ. Open Systems Interconnection)

OWASP – проєкт з безпеки вебзастосунків (англ. Open Worldwide Application Security Project)

PaaS – платформа як послуга (англ. Platform as a Service)

WAF – фаєрвол вебзастосунку (англ. Web Application Firewall)

XSS – міжсайтове скриптування (англ. Cross-Site Scripting)

ВСТУП

В умовах стрімкого розвитку цифрових технологій вебзастосунки стали ключовими інструментами для ведення бізнесу, обслуговування клієнтів та реалізації складних інформаційних систем. Зі зростанням кількості користувачів та обсягів переданих даних зростає навантаження на сервери, що може призвести до зниження швидкодії, виникнення затримок в доступі до ресурсів та збоїв в роботі. В таких умовах продуктивність вебзастосунків стає нагальною проблемою, що потребує пошуку ефективних рішень для забезпечення стабільної та неперервної роботи додатку в умовах динамічної зміни навантаження та забезпечення конкурентоспроможності компанії.

Водночас зростає рівень потенційних атак, що вимагає впровадження додаткових засобів захисту, зокрема застосування фаєрволу вебзастосунків (WAF). Традиційно WAF розглядається як засіб безпеки, що фільтрує шкідливі запити на рівні застосунку. Сучасні дослідження показують, що інтеграція WAF з додатковими сервісами, зокрема кешуванням, забезпечує належний рівень безпеки та покращує маршрутизацію запитів, обробку контенту, що сприяє підвищенню загальної продуктивності вебзастосунку.

Мета кваліфікаційної роботи полягає в розробці методу підвищення продуктивності вебзастосунків з використанням WAF. Запропонований метод має забезпечити підвищений рівень захисту та мінімізації затримок в умовах динамічної зміни навантаження, що дозволяє забезпечити високу продуктивність додатку з вимогами до захисту конфіденційної інформації користувачів та мінімізації ризиків витоку та несанкціонованого доступу до ресурсів вебзастосунку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Стрімкий розвиток цифрових технологій та поширення цифрових послуг породжує суттєве збільшення кількості вебзастосунків, які відкривають нові можливості для взаємодії між користувачами та постачальниками послуг, охоплюючи різноманітні сфери діяльності людини [1, 2].

Найбільшого поширення вебзастосунки набули в таких ключових сферах, як охорона здоров’я, освіта, медицина, фінанси та туризм. В цих секторах вебтехнології активно використовуються для автоматизації процесів, підвищення якості обслуговування користувачів, розширення доступу до послуг та оптимізації витрат. Завдяки інтеграції застосунків, організації мають можливість оперативно реагувати на потреби споживачів, що сприяє підвищенню ефективності та зручності здійснення операцій, забезпечуючи кращу доступність товарів і послуг, а також стимулюючи розвиток нових бізнес-моделей та сучасних інфраструктур в галузі комунікаційних технологій.

Проте, з розвитком технологій та збільшенням числа користувачів Інтернету, зростає ескпоненційно кількість загроз для безпеки вебзастосунків. На тлі жорсткої економічної конкуренції, зростання попиту на віддалений формат роботи, спричиненого тривалими наслідками пандемії COVID-19 та воєнними конфліктами в різних регіонах світу, вебсервіси дедалі частіше стають об’єктом цілеспрямованих кібератак, що можуть призвести до витоку конфіденційної інформації, втрати репутації організації або спричинити збої в роботі. Саме тому питання організації безпеки вебзастосунків набуває особливої актуальності [3, 4].

За умов постійного зростання кількості інструментів та методів здійснення кібератак, забезпечення безперервної доступності вебресурсів та ефективного захисту додатків або їх окремих компонентів від несанкціонованого доступу, модифікації контенту чи експлуатації

вразливостей стає дедалі складнішим завданням. Серед основних причин вразливості вебресурсів можна відокремити їх публічну доступність, недостатню якість програмного коду, конфігураційні помилки на рівні серверної інфраструктури, а також недостатні заходи безпеки. Це створює сприятливе середовище для несанкціонованого доступу до ресурсів вебзастосунку та персональних даних користувачів. Саме тому постає потреба у впровадженні комплексних засобів захисту, здатних адаптуватися до архітектурних особливостей вебзастосунків без зниження продуктивності та якості обслуговування користувачів.

Ключовою проблемою сучасних вебзастосунків є надмірне споживання обчислювальних ресурсів, особливо в умовах високого навантаження, спричиненого зростанням кількості користувацьких запитів [5]. Це, в свою чергу, призводить до зниження загальної продуктивності системи, затримок в обробці запитів, а в окремих випадках – до відмови вебсервера. Додатковим фактором зниження ефективності є впровадження багаторівневих систем захисту. Такі механізми передбачають багатоетапну перевірку мережного трафіку, що збільшує час обробки запитів, підвищення навантаження на інфраструктуру та загальне уповільнення функціонування додатку.

Для забезпечення безпеки вебзастосунків ключову роль відіграють брандмауери, основна функція яких полягає в контролі та фільтрації трафіку, виявлення аномальної активності та мінімізації ризиків порушення цілісності та доступності даних [6, 7]. Залежно від функціональних особливостей, місця інтеграції в мережну інфраструктуру та рівня аналізу трафіку, фаєрволи поділяються на декілька основних типів, кожен з яких має свої переваги та обмеження.

1.1 Типи та види брандмауерів

В сучасних комп’ютерних мережах брандмауери або фаєрволи забезпечують захист комп’ютерних систем і мереж від несанкціонованого

доступу, підозрілого трафіку та кібератак. Вони діють як бар'єр між внутрішнім захищеним середовищем локальної мережі та потенційно небезпечними зовнішніми запитами та ресурсами, здійснюючи моніторинг та аналіз мережних запитів відповідно до встановлених політик безпеки [8].

В залежності від архітектури, способу розгортання та призначення, існують різні типи брандмауерів. До основних типів брандмауерів в сучасних комп'ютерних мережах відносяться хмарні, апаратні та програмні брандмауери (рисунок 1.1).

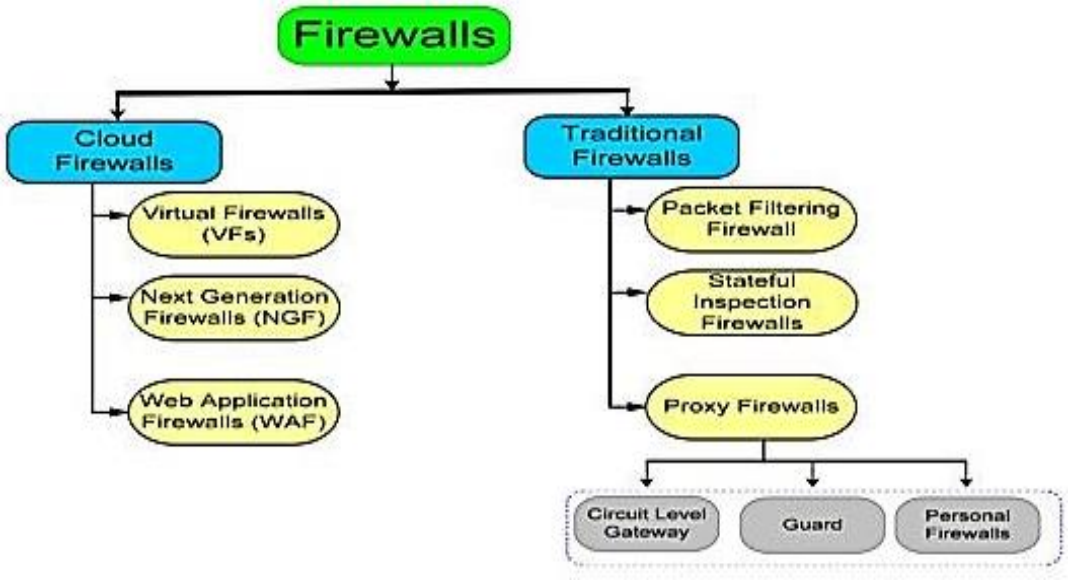

Рисунок 1.1 – Основні типи брандмауерів

Хмарні брандмауери використовуються для захисту розподілених хмарних інфраструктур, де фізичний доступ до обладнання обмежений. Вони забезпечують високу гнучкість та масштабованість, мають інтегровані механізми безпеки, забезпечуючи ефективний захист бізнес-процесів та хмарну інфраструктуру організацій приватного та державного сектору. В роботі [9] автори зазначають, що провідні технологічні компанії, такі як Amazon AWS, Google, IBM, Sun, Microsoft та інші, активно розробляють ефективні хмарні продукти та технології, зокрема в сфері захисту хмарної інфраструктури. Водночас наголошується, що навіть за умови високого рівня розвитку хмарних сервісів, існує обмежена здатність до виявлення атак або

вразливостей, джерелом яких можуть бути як самі постачальники хмарних послуг, так і кінцеві користувачі, що створює необхідність пошуку рішень для забезпечення комплексної безпеки в хмарному середовищі. Іншими обмеженнями використання хмарних брандмауерів є необхідність надійного з'єднання для доступу до захищених ресурсів, що може стати проблемою при нестабільному з'єднанні, та складність адміністрування, що підкреслюється авторами робот [10-12].

Апаратні брандмауери представляють собою фізичні пристрої, які встановлюються на межі між локальною мережею та загальнодоступними мережами Інтернет. Апаратні фаєрволи наразі є найбільш розповсюдженими, вони забезпечують високу продуктивність та надійний захист великої кількості користувачів та ресурсів. Проте апаратні брандмауери втрачають свою ефективність в сучасних IT-інфраструктурах. В роботі [13] автори зазначають, що традиційні апаратні брандмауери забезпечують захист мережі на основі портів, стану та протоколів, проте цього недостатньо для захисту вебсервісів через складність та різноманітність сучасних загроз. Автори роботи [14] підкреслюють, що апаратні брандмауери характеризуються підвищеним енергоспоживанням порівняно з програмними або хмарними рішеннями. Крім того, застосування складних та об'ємних політик безпеки в апаратних брандмауерах може спричинити додаткове навантаження на мережну інфраструктуру, що є критичним фактором для комп'ютерних мереж з обмеженими ресурсами.

Програмні брандмауери – це спеціалізоване програмне забезпечення, яке встановлюється на комп'ютерах або серверах для здійснення моніторингу, контролю та фільтрації мережного трафіку відповідно до заданих правил безпеки [15]. Такі фаєрволи використовують приватні користувачі та в організаціях малого та середнього бізнесу, оскільки вони не потребують додаткового спеціалізованого обладнання та можуть бути адаптовані під різні операційні системи. Перевагами програмних брандмауерів є висока продуктивність та простота адміністрування, що

підкреслено в роботах [16, 17].

Проте програмні брандмауери також мають недоліки, зокрема підвищене споживання системних ресурсів, що може впливати на загальну продуктивність пристрою, особливо в умовах обмежених ресурсів [18]. Крім того, ефективність програмних рішень значною мірою залежить від конфігурації та оновлення ПЗ, що потребує досвіду та уважності адміністратора.

Існуючі типи фаєрволів можуть бути класифіковані за способами фільтрації трафіку за рівнем еталонної моделі OSI:

- фаєрволи мережного рівня;
- фаєрволи рівня з’єднання (сеансу);
- фаєрволи прикладного рівня [19].

Фаєрволи, які працюють на третьому, мережному, рівні моделі OSI, здійснюють базову фільтрацію трафіка, аналізуючи IP-адреси, порти та протоколи без розгляду вмісту пакетів. Найбільш поширеними є фаєрволи фільтрації пакетів, апаратні або програмні, головна перевага яких у високій швидкості обробки даних. Рішення про дозволи чи блокування трафіку приймаються відповідно до правил, які встановлює адміністратор. Попри ефективність для базового контролю доступу, такі фаєрволи не здатні аналізувати вміст самих пакетів, що робить їх уразливими до атак на прикладному рівні.

Міжмережні екрани зі станом з’єднання працюють на сеансовому рівні моделі OSI. В таких фаєрволах вдосконалено механізми фільтрації пакетів: окрім аналізу заголовків пакетів, вони перевіряють приналежність пакету до існуючого з’єднання. Ці механізми дозволяють більш точно виявляти підозрілий трафік, що забезпечує підвищення загального рівня безпеки мережі.

Брандмауери наступного покоління (NGFW) мають розширені функції в порівнянні з традиційними міжмережними екранами. В фаєрволах NGFW інтегровано функції глибокого аналізу пакетів (DPI), системи виявлення та

запобігання вторгненням (IDS/IPS), а також здійснюється фільтрація трафіку на основі ролі користувача. Ці пристрої здатні ідентифікувати та контролювати окремі програми незалежно від протоколів чи портів, що значно підвищує рівень захисту мережі від складних та цілеспрямованих кіберзагроз.

На прикладному рівні моделі OSI функціонують проксі-фаєрволи та фаєрволи вебзастосунків (WAF), виконуючи роль посередника: отримуючи запити від користувача, перевіряючи на можливі загрози та спрямовуючи їх до відповідних сервісів. Вони аналізують трафік в межах визначених додатків і протоколів, зокрема HTTP, HTTPS, FTP та DNS, і можуть виявляти загрози, приховані у вмісті запитів. WAF безпосередньо створені для забезпечення безпеки вебзастосунків і захищають від поширених сучасних атак, зокрема SQL-ін’єкції, міжсайтове виконання скриптів (XSS) чи атак типу «відмова в обслуговуванні» (DoS). Ці брандмауери аналізують вхідні запити до вебсерверів та блокують підозрілий трафік відповідно до визначених політик доступу, що встановлені адміністратором. Завдяки такому підходу забезпечується глибока перевірка запитів, що дозволяє виявляти загрози на ранньому етапі. Проте недоліком цих фаєрволів є збільшення навантаження через потребу в додаткових обчислювальних ресурсах та виникнення додаткової затримки при обробці трафіка в умовах застосування складних політик фільтрації [20].

1.2 Особливості функціонування фаєрволів

Основна функція брандмауеру полягає в моніторингу та фільтрації трафіку між локальної мережею вебзастосунку та зовнішніми користувачами та ресурсами, виявлення аномалій та блокуванні підозрілого трафіку (рисунок 1.2).

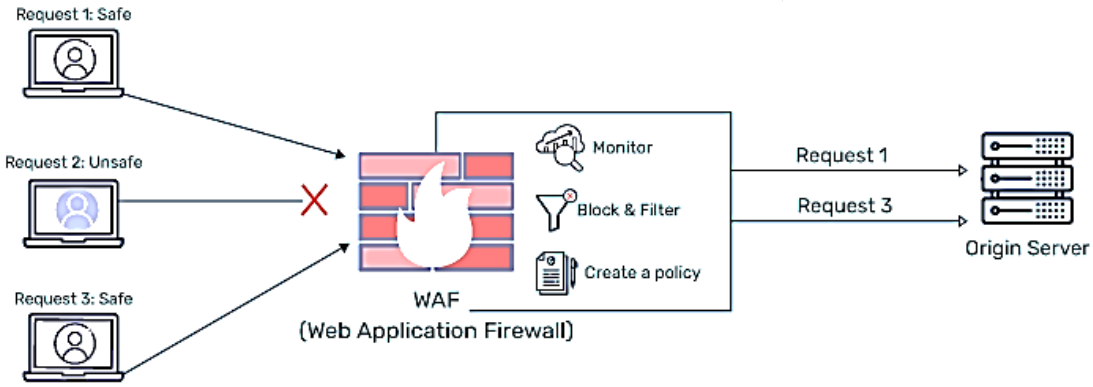

Рисунок 1.2 – Схема функціонування фаєрволу вебзастосунку

Архітектура сучасних вебзастосунків складається з двох основних компонентів: фронтенду та бекенду. Взаємодія користувача з вебзастосунком здійснюється через візуальний інтерфейс, доступний у веббраузері. В процесі роботи із застосунком користувач надсилає запити на сервер та ініціює дії, наприклад, шляхом кліків або заповнення відповідних форм. На цьому етапі часто відбувається введення персональних даних: логіна та пароля, адреси електронної пошти, імені та прізвища, а в деяких випадках – платіжної інформації або адреси проживання. Саме ці дані мають конфіденційний характер та потребують захисту від несанкціонованого доступу до них та ризиків витоку.

Фронтенд є клієнтською частиною вебзастосунку, яка відповідає за збір даних від користувача, формування запитів до серверної частини, обробку відповідей та відображення результатів взаємодії. Крім того, фронтенд передає введені персональні дані на сервер для збереження в базі даних, зокрема в разі реєстрації або авторизації користувача.

Бекенд виконує роль серверної частини вебзастосунку та забезпечує обробку вхідних запитів, доступ до бази даних та файлової системи, а також формування відповідей, що надсилаються користувачу через клієнтську частину. До складу серверної частини входять: файлова система, яка містить контент додатку та конфігураційні файли його основних сервісів; база даних, що зберігає інформацію про користувачів, їх облікові записи, результати

взаємодій, а також дані транзакцій в разі використання платіжних функцій; та вебсервер, який забезпечує обробку HTTP-запитів і координацію взаємодії між усіма сервісами вебзастосунку (рисунок 1.3).

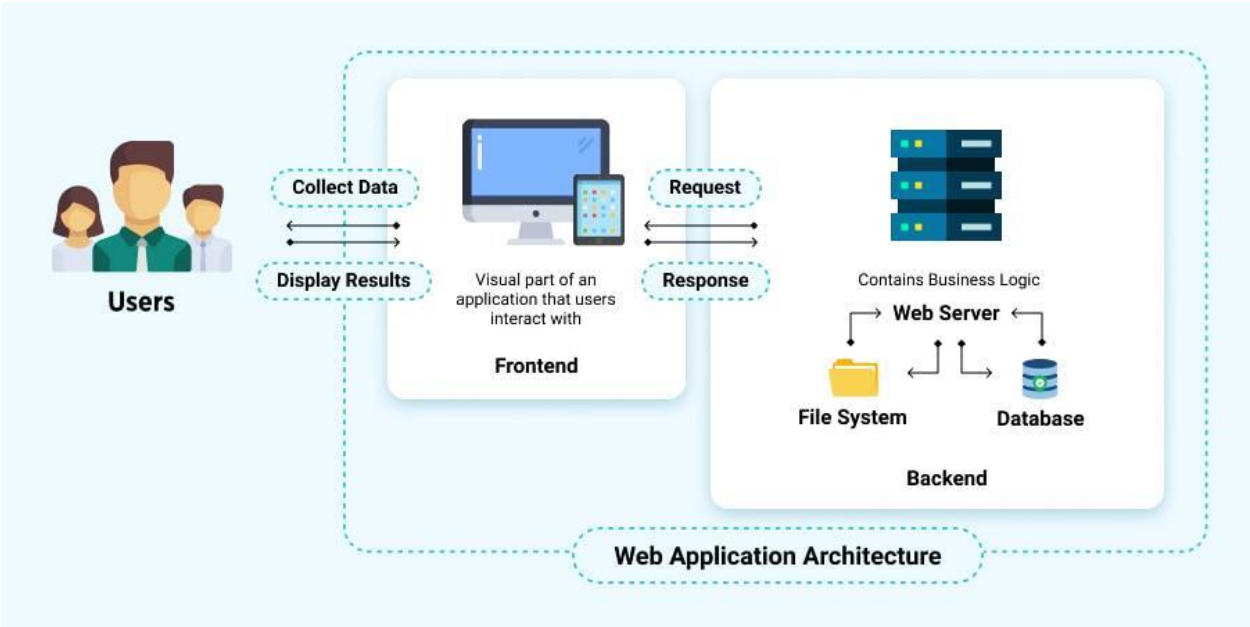

Рисунок 1.3 – Схема типового вебзастосунку

Фаєрвол виступає в ролі бар'єру між внутрішньою інфраструктурою вебзастосунку – його серверною частиною та зовнішнім середовищем – користувачами, блокуючи потенційно шкідливі або несанкціоновані запити.

Головними функціями фаєрволу є фільтрація вхідного трафіка та контроль доступу до ресурсів. Контроль доступу здійснюється на основі застосування політик обмеження доступу до певних портів, служб або IP-адрес локальної мережі, що дозволяє запобігти несанкціонованому доступу до локальних ресурсів серверної частини додатку. Фільтрація трафіка здійснюється відповідно до встановлених адміністраторами політик доступу. Такий підхід дозволяє проводити перевірку запитів, що надходять до мережі вебзастосунку з метою виявлення потенційних загроз. В роботі [21] автори зазначають, що складні алгоритми, що застосовуються для фільтрації трафіка брандмауерами в режимі реального часу, можуть спричиняти додаткові затримки під час аналізу та фільтрації запитів і можуть суттєво знизити

якість обслуговування користувачів та доступність вебресурсів, що в цілому знижує загальну продуктивність вебзастосунку.

Традиційні брандмауери, які засновані на базовій фільтрації пакетів, мають високу швидкість обробки даних, оскільки здійснюють базову перевірку заголовків пакетів без детального вмісту. В роботі [22] зазначається, що розподілені атаки типу «відмова в обслуговуванні» (DoS) належать до категорії найбільш небезпечних загроз для вебзастосунків. Якщо раніше такі атаки здебільшого реалізовувалися на мережному рівні, то останнім часом спостерігається їх активне поширення на прикладному рівні та відзначаються значно вищою складністю реалізації та прихованістю в порівнянні з мережними атаками, що істотно ускладнює процес їх виявлення та нейтралізації. Саме тому традиційні брандмауери з базовою фільтрацією пакетів мають низьку ефективність при забезпеченні безпеки вебзастосунку на противагу з WAF, які працюють на прикладному рівні.

Глибоке сканування кожного пакету при складних алгоритмах фільтрації трафіку, а також виконання додаткових функцій, зокрема ідентифікації додатків або контроль доступу на рівні користувачів може викликати додаткові затримки, що також впливають на зниження загальної продуктивності вебзастосунків. В роботі [23] зазначено, що зі збільшенням складності алгоритмів фільтрації трафіку в режимі реального часу виникають додаткові затримки, що погіршують продуктивність, проте використання кешування дозволяє скоротити витрати обчислювальних ресурсів, що сприяє підвищенню продуктивності вебзастосунків.

1.3 Постановка задачі

Сучасні інформаційні технології відкривають значні можливості для впровадження вебзастосунків, які виступають ефективним інструментом комунікації в різних сферах діяльності. Це зумовило зростання попиту на розробку веборієнтованих рішень, здатних задовольняти потреби бізнесу та

суспільства загалом. В умовах зростаючої складності кібератак і підвищеного навантаження на мережну інфраструктуру важливим стає забезпечення високої продуктивності вебзастосунків за підвищених вимог до безпеки. Серед практичних рішень, які забезпечують достатній рівень безпеки застосунків, є використання фаєрволів вебзастосунків, які виконують фільтрацію вхідного трафіку, що дозволяє захистити локальні вебресурси від несанкціонованого доступу до конфіденційних даних та витоку інформації.

Метою кваліфікаційної роботи є розробка методу підвищення продуктивності вебзастосунку з використанням WAF, що дозволяє зменшити затримки при обробці запитів, забезпечуючи належний рівень безпеки ресурсів від несанкціонованого доступу та ризиків витоку інформації.

2 МЕТОДИ ЗАБЕЗПЕЧЕННЯ ПРОДУКТИВНОСТІ ВЕБЗАСТОСУНКІВ

2.1 Особливості архітектури вебзастосунків

Стрімке зростання кількості вебзастосунків, що обумовлене широким впровадженням цифрових технологій в різні сфери суспільства, актуалізує питання забезпечення їх високої продуктивності в умовах зростання складності мережної інфраструктури. Зростання навантаження на вебсервіси, викликане збільшенням кількості користувачів, ескалацією ризиків несанкціонованого доступу до конфіденційної інформації, а також зношенням апаратного забезпечення та високими витратами на його оновлення вимагає застосування практичних рішень, що забезпечуватиме ефективну роботу вебзастосунків в умовах обмежених ресурсів при одночасному дотриманні вимог до якості обслуговування та безпеки в умовах динамічних змін навантаження.

Одним з сучасних підходів до вирішення зазначених проблем є впровадження технологій віртуалізації та контейнеризації. Вони дозволяють створювати масштабовану, гнучку інфраструктуру, яка забезпечує високий рівень продуктивності вебзастосунків за рахунок ефективного розподілу та використання обчислювальних ресурсів.

Зокрема, застосування гіпервізорів дає можливість розгортати ізольовані віртуальні середовища з визначеною операційною системою, що сприяє покращенню керованості, безпеки та адаптивності сервісів [24]. Технологія віртуалізації на базі гіпервізорів дозволяє організувати віртуальну інфраструктуру, яка легко масштабується, має інтегровані інструменти автоматизації та управління кластерами віртуальних машин, зокрема Kubernetes, а також дозволяє розгорнути додаткові сервіси, необхідні для забезпечення стабільної роботи вебзастосунків.

Водночас віртуалізація на базі гіпервізорів має певні обмеження, що

полягають у значному споживанні апаратних ресурсів та складності адміністрування в порівнянні з контейнеризацією, яка дозволяє забезпечити ізоляцію середовища на рівні операційної системи (ОС) з мінімальними накладними витратами, що сприяє підвищенню ефективності та швидкості розгортання сервісів [25]. В таблиці 2.1 наведено ключові відмінності між віртуалізацією на рівні гіпервізорів та контейнерною віртуалізацією.

Таблиця 2.1 – Порівняння характеристик віртуальних машин та контейнерів

Функція	**Контейнери**	**Віртуальні машини**
Операційна система	Використовує ядро ОС хоста	Потребує окремої гостьової ОС
Витрати ресурсів	мінімальні витрати	Вище споживання ресурсів
Щільність	Висока щільність контейнерів на хост	Нижча щільність віртуальних машин на хост
Використання ресурсів	Менш ресурсоємний	Більш ресурсоємний
Портативність	Висока портативність між середовищами	Необхідність в додатковому налаштуванні
Рівень ізоляції	Ізоляція на рівні процесу	Повна ізоляція на рівні обладнання
Процес розробки	Узгоджений в різних середовищах	Може відрізнятися в різних середовищах

Платформи контейнеризації, зокрема Docker, Proxmox та Podman, дозволяють запускати незалежні сервіси в межах однієї ОС з мінімальними витратами ресурсів, а також мають вбудовані інструменти для масштабування та оркестрації додатків в контейнерах [26, 27]. Основними перевагами технології контейнеризації є:

- масштабованість, яка полягає в можливості динамічного регулювання кількості контейнерів залежно від поточного навантаження та апаратних ресурсів. Це сприяє контролю та оптимальному використанню ресурсів, а також забезпечує стабільну та безперервну роботу сервісів та застосунків [28];

- портативність, що забезпечує сумісність з різними середовищами

виконання та можливістю міграції контейнеризованих застосунків між локальними ресурсами та хмарними платформами. Високий рівень мобільності та універсальності контейнерів сприяє спрощенню процесів розробки, забезпечуючи адаптивність та масштабованість вебзастосунків [29];

- ефективне управління ресурсами, що полягає в раціональному використанні обчислювальних ресурсів. Контейнери працюють безпосередньо на ядрі операційної системи хост-машини, що дозволяє розгорнути та обслуговувати більшу кількість контейнерів в порівнянні з віртуальними машинами. Це дозволяє зменшити витрати на інфраструктуру та забезпечити гнучке масштабування [30].

Платформа контейнеризації Docker забезпечує створення, управління та запуск контейнерів з використанням стандартизованих контейнерних образів сервісів. Кожен образ містить повний набір компонентів, необхідних для функціонування сервісу, в тому числі й вебзастосунків, яке включає вихідній код, бібліотеки залежності та конфігураційні файли. Основною перевагою Docker є висока портативність контейнерів, що забезпечує незалежність середовища виконання від типу інфраструктури, а також широкий спектр інструментів для оркестрації та автоматизації розгортання, що сприяє скороченню часу на перенесення вебзастосунку між етапами розробки, тестування та продуктивної експлуатації. Завдяки простому адмініструванню, великої кількості доступних образів сервісів, Docker є найбільш поширеною платформою контейнеризації для побудови масштабованих та надійних вебзастосунків [31].

Podman є іншою платформою контейнеризації, яка за своїм функціоналом схожа на Docker. В порівнянні з Docker, Podman має суттєву відмінність, що полягає у відсутності необхідності запуску демона, що підвищує безпеку та спрощує інтеграцію в системи з жорсткими вимогами до контролю доступу. Такий підхід дозволяє у Podman зменшити ризики безпеки в порівнянні з традиційним підходом Docker. Він підтримує такі самі формати контейнерних образів, як в Docker, а також інтегрується з

системами автоматизації, зокрема Systemd [32].

Proxmox є універсальною платформою віртуалізації та дозволяє запускати віртуальні машини та LXC контейнери. Це дає змогу ефективно використовувати апаратні ресурси за рахунок легковагової ізоляції процесів в межах однієї операційної системи. Proxmox дозволяє створювати гнучкі інфраструктури, які поєднують віртуальні машини з контейнерами в єдиному інтерфейсі, що спрощує адміністрування комплексних інфраструктур з різнорідним середовищем розгортання вебзастосунків [33].

Таким чином технологія контейнеризації є однією з ефективних у забезпеченні гнучкого масштабування та ефективного управління життєвим циклом вебзастосунків, що відповідає сучасним вимогам до продуктивності, надійності та доступності вебсервісів. Контейнеризація сприяє впровадженню мікросервісної архітектури, яка передбачає розділення функціональності додатку на окремі незалежні сервіси, що розгортаються в ізольованих контейнерах. Це дозволяє досягти високої гнучкості, спрощує оновлення окремих компонентів, полегшує масштабування та підвищує відмостійкість вебзастосунків.

Створення масштабованої та логічно ізольованої мережної інфраструктури в Docker здійснюється за допомогою віртуальних мереж, що дозволяє об’єднати або сегментувати сервіси в окремих підмережах. Також це забезпечує можливість взаємодії контейнерів між собою за допомогою наперед визначених портів та протоколів. Такий підхід дозволяє контролювати маршрутизацію запитів та підвищити безпеку міжсервісної взаємодії, що є важливим для побудови мікросервісної архітектури.

Впровадження сучасних засобів захисту, зокрема WAF, систем виявлення вторгнень, механізмів шифрування трафіка та політик контролю доступу дозволяють додатково захистити мережну інфраструктуру та дані користувачів від ризиків несанкціонованого доступу та витоків конфіденційних даних. Такі переваги контейнеризації сприяють побудові високонавантажених, відмовостійких та масштабованих вебсервісів.

Використання хмарних платформ, зокрема Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), є сучасною тенденцією в розробці та експлуатації вебзастосунків. Завдяки сервісним моделям «інфраструктура як послуга» (IaaS) та «платформа як послуга» (PaaS) хмари забезпечують високий рівень автоматизації, гнучке масштабування ресурсів, інтеграцію з CI/CD-процесами, а також вбудовані засоби автентифікації та шифрування для ефективної розробки та експлуатації вебсервісів. Проте висока вартість хмарних рішень залишається суттєвим обмеженням для компаній малого та середнього бізнесу, яке ускладнює використання хмарних платформ для розробки та експлуатації вебзастосунків [34].

2.2 Ключові методи забезпечення продуктивності вебзастосунків

Зі стрімким зростанням обсягів трафіку до розробки вебзастосунків висуваються підвищені вимоги щодо швидкодії, стабільності та забезпечення безперебійного доступу. Такі вимоги зумовлюють необхідність впровадження ефективних методів забезпечення їх продуктивності, а саме адаптивність системи до змінних навантажень, зменшення затримок в обробці запитів та стійкість до збоїв. Серед найпоширеніших методів підвищення продуктивності вебзастосунків на рівні серверної частини вирізняють масштабування, балансування навантаження та кешування, які забезпечують стабільну роботу сервісів в умовах високих навантажень.

Масштабування є одним з методів забезпечення продуктивності, стійкості та адаптивності вебзастосунків до змінного навантаження, яке передбачає зміну кількості екземплярів сервісів або збільшення обчислювальних ресурсів одного вузла, що забезпечують роботу застосунку, з метою зниження затримки при обробці зростаючої кількості користувацьких запитів [35]. Існує два основних типа масштабування – вертикальне та горизонтальне. Для вебінфраструктури найчастіше застосовують горизонтальне масштабування, оскільки воно забезпечує кращу

гнучкість, модульність та відмовостійкість системи.

Вертикальне масштабування полягає в збільшенні обчислювальних ресурсів окремого екземпляра додатку або сервісу шляхом збільшення обсягу оперативної пам’яті, кількості процесорних ядер чи швидкодії дискового простору. Такий підхід вважається простим у реалізації, проте має свої обмеження, зокрема вимогу простою системи під час оновлення через зміну обсягу ресурсів та фізичні обмеження апаратних ресурсів [36].

Горизонтальне масштабування передбачає розгортання декількох однотипних екземплярів додатку або сервісу, які функціонують паралельно та обробляють запити користувачів як розподілена система. Такий підхід реалізовано в сучасних хмарних та мікросервісних архітектурах, оскільки дозволяє практично необмежено масштабувати систему. Обмеження такого підходу може полягати в кількості ресурсів, яке можна надати створеному кластеру [37].

Для горизонтального масштабування в Docker використовують інструменти автоматизації, зокрема Docker Swarm та Docker Compose, та оркестрації, зокрема Kubernetes, які автоматизують створення, розгортання та масштабування екземплярів додатку у відповідь на зміну навантаження. Це дозволяє реалізувати автоматичне масштабування на основі метрик продуктивності, зокрема CPU, пам’ять, кількість активних з’єднань або затримки у відповіді сервера. Такий підхід забезпечує гнучке реагування системи на зміну навантаження в реальному часі без необхідності ручного втручання. Також горизонтальне масштабування сприяє зменшенню часу розгортання, підвищує щільність розміщення сервісів на фізичних вузлах та забезпечує легкість оновлення відповідних сервісів.

Застосування масштабування в серверній частині вебсервісів забезпечує адаптацію до пікових навантажень та підвищує загальну відмовостійкість застосунку. В разі збою окремого вузла, трафік може бути перенаправлено до інших активних екземплярів, що дозволяє зберегти стабільну та неперервну роботу застосунку.

Балансування навантаження є іншим підходом до забезпечення високої продуктивності вебзастосунків, який полягає в рівномірному розподілі вхідного трафіка між кількома екземплярами серверів або контейнерів з метою запобігання перевантаження та мінімізації затримок у відповіді сервера [38]. Такий підхід забезпечує стабільну роботу застосунків в умовах підвищеного навантаження, що актуально для застосунків з мікросервісною архітектурою.

В залежності від типу інфраструктури та специфіки застосунку можуть бути застосовані різні стратегії реалізації балансування навантаження:

- круговий розподіл, за яким запити рівномірно перенаправляються у визначеній послідовності до доступних серверів. Недоліком цього підходу є те, що при цьому не враховується поточне навантаження на сервери, що може призвести до надмірного споживання ресурсів та спричинити відмову через перевантаження;
- зважений круговий розподіл, за яким враховується продуктивність кожного сервера. При такому підході на більш потужні екземпляри перенаправляється більша кількість запитів, забезпечуючи пропорційне навантаження відповідно до обчислювальних можливостей кожного сервера;
- найменша кількість з'єднань, що дозволяє спрямовувати нові запити на сервер з найменшою кількістю активних з’єднань. Такий підхід дозволяє уникати перевантаження окремих серверів та забезпечувати адаптивне балансування в режимі реального часу;
- IP-хешування, за яким запити розподіляються відповідно до хеш-суми IP-адреси сервера, що дозволяє забезпечити послідовність обслуговування запитів, які надходять з одного вузла, та зберігати безперервність з’єднання [39].

В Docker використовуються вбудовані інструменти для балансування навантаження, зокрема Docker Swarm, для забезпечення автоматичного балансування навантаження між репліками сервісу. Також до функцій інструмента оркестрації Kubernetes входить балансування навантаження, що

забезпечує рівномірний розподіл запитів між подами або контейнерами, враховуючи стан та навантаження кожного з них.

Таким чином, балансування навантаження забезпечує стійкість, продуктивність та масштабованість вебзастосунків, особливо в умовах застосування мікросервісної архітектури та розгортання вебзастосунків в контейнерному середовищі.

Ще одним підходом до забезпечення продуктивності застосунків є кешування. Основна мета кешування полягає в тимчасовому збереженні часто запитуваних даних у швидкодоступній пам’яті, зокрема RAM або окремих кеш-сервісах, що дозволяє значно знизити навантаження на серверну частину додатку та зменшити затримки при обробці запитів [40].

В архітектурі вебзастосунків застосовують наступні типи кешування:

- кешування на стороні користувача, при якому в браузері користувача зберігаються статичні ресурси, зокрема зображення, CSS, JavaScript, локально, що дозволяє уникати повторних запитів до сервера застосунку, зменшуючи навантаження та прискорюючи завантаження сторінок;
- кешування на стороні сервера, яке використовується для збереження запитів до бази даних. Воно може бути реалізовано за допомогою механізмів внутрішньої пам’яті або з використанням кеш-сервісів, зокрема Memcached або Redis;
- проміжне кешування або реверсне кешування, яке реалізується на рівні вебсерверів за допомогою додаткових сервісів, зокрема Nginx, Varnish, які зберігають готові HTTP-відповіді та повертають їх користувачу без звернення до вебсерверу, що знижує час відгуку та підвищує якість обслуговування користувачів [41].

В Docker найчастіше застосовуваними сервісами для кешування є Redis, Memcached та Nginx, які мають готові та доступні образи в DockerHub. Вони розгортаються в окремому контейнері та забезпечують функцію кешування.

Redis та Memcached використовують для високошвидкісного

кешування даних в пам’яті, що дозволяє значно прискорити обробку запитів до бази даних або серверів. Redis підтримує складні структури даних, транзакції, публікацію-підписку, а також стійкість даних завдяки збереженню на диск. Проте Redis потребує більше обчислювальних ресурсів та має складну конфігурацію в порівнянні з Memcached, який використовується для кешування простих об’єктів у пам’яті. Проте недоліками Memcached є відсутність стійкості даних та обмежені функціональні можливості в порівнянні з Redis [42].

Nginx може використовуватись як проксі-сервер, а з увімкненим кешуванням – як проміжний контейнер реверсного проксі-сервера, який зберігає копії статичних ресурсів, API-відповідей або HTML-контенту. Такий підхід дає змогу суттєво знизити навантаження на вебсервери, скоротити час відповіді та забезпечити стабільнішу роботу додатку в умовах великої кількості одночасних запитів [43]. До переваг Nginx належать висока продуктивність при обробці великого обсягу трафіка та можливість гнучкого налаштування кешування за визначеними правилами. Проте недоліком Nginx є неефективність для кешування складних обчислень або структурованих даних.

Використання Docker-томів для кешування дозволяє зберігати великі обсяги даних на диску з одночасним доступом кількох контейнерів [44]. Такий підхід застосовується для кешування файлових об’єктів або даних, які неможливо зберігати у пам’яті. Перевагою такого підходу є простота реалізації та забезпечення стійкості даних між сесіями роботи контейнерів. Проте недоліком кешування з використанням Docker-томів є зниження швидкості доступу, що може знижувати загальну продуктивність вебзастосунку при частому зверненні до кешованих даних.

Таким чином, в умовах зростання обсягів вебтрафіку та підвищених вимог до швидкодії, масштабованості, стабільної та безперервної роботи сервісів забезпечення високої продуктивності вебзастосунків є невід’ємною частиною при розробці застосунків. Технологія контейнеризації є

ефективним підходом для розробки застосунків з мікросервісною архітектурою. Використання платформи контейнеризації Docker має низку переваг, зокрема гнучкість масштабування, ізоляцію сервісів, спрощення управління життєвим циклом застосунку та інтеграцію з хмарною інфраструктурою. Для забезпечення високої продуктивності застосунків в умовах ризику несанкціонованого доступу та витоку інформації, підвищених вимог до стабільної та неперервної роботи застосунку, а також зростання обсягів користувацьких запитів необхідно застосовувати методи для підвищення загальної продуктивності вебзастосунків, які мінімізують затримки при обробці запитів та забезпечують відмовостійкість інфраструктури. До таких методів належать масштабування, балансування навантаження та кешування, які сприяють мінімізації затримок, рівномірному розподілу трафіка та ефективному використанню апаратних ресурсів. Слід зазначити, що балансування навантаження забезпечує адаптивність до динамічної зміни навантаження, а кешування дозволяє знизити затримки при обробці запитів, зменшуючи кількість запитів до вебсерверу. Застосування цих методів дозволяє розгортати комплексну архітектуру застосунку, яка забезпечує високу доступність та продуктивність вебсервісів в умовах динамічної зміни навантаження та обмежених апаратних ресурсів.

3 РОЗРОБКА СЕГМЕНТА МЕРЕЖІ ВЕБЗАСТОСУНКУ З ВИКОРИСТАННЯМ WAF

3.1 Моделювання сегменту мережі вебзастосунку з використанням WAF

На початковому етапі дослідження методу підвищення продуктивності застосунку з використанням WAF було здійснено аналіз апаратних вимог для побудови тестового середовища. Основою для тестового середовища сегменту комп’ютерної мережі, що імітує роботу застосунку з інтегрованим фаєрволом вебзастосунку, було обрано платформу контейнеризації Docker, що дозволяє розгортати вебзастосунку в окремих ізольованих контейнерах.

Для розгортання платформи контейнеризації Docker було використано хост-машину з операційною системою Ubuntu 20.04 LTS, процесором з щонайменше 2 ядрами та максимальною частотою 2,9 ГГц. Об'єм ОЗУ залежить від кількості запущених контейнерів та загального навантаження на систему. З практичної точки зору, обсяг 16 ГБ ОЗУ забезпечує ефективне функціонування такої конфігурації, а рівень апаратних ресурсів дозволяє моделювати інфраструктуру, яку можна застосовувати в умовах малого та середнього бізнесу з можливістю масштабування за наявності апаратних ресурсів.

Пропускна здатність каналу повинна становити не менше 1 Гбіт/с. Для ефективної роботи вебзастосунку з інтегрованим WAF доцільно використовувати два мережні інтерфейси, які забезпечують керування доступу до мережі Інтернет платформі Docker та управління трафіком всередині тестового середовища, що достатньо для обслуговування користувачів в експериментальній комп’ютерній мережі. Жорсткий диск, на якому розгорнуто Docker, має містити щонайменше 500 ГБ вільного простору для зберігання образів контейнерів, журналів подій та іншої службової інформації.

В тестовому середовищі було розгорнуто два контейнери, в одному з

яких було розгорнуто вебзастосунок Flask, в іншому проксі-сервер Nginx з вбудованим модулем WAF ModSecurity. Таким чином фаєрвол вебзастосунку, розташований між користувачем і проксі-сервером, виконує роль фільтра безпеки. Він аналізує вхідний трафік, виявляє підозрілі або шкідливі запити, які в разі виявлення блокуються, повертаючи користувачеві повідомлення про відмову в доступі (рисунок 3.1).

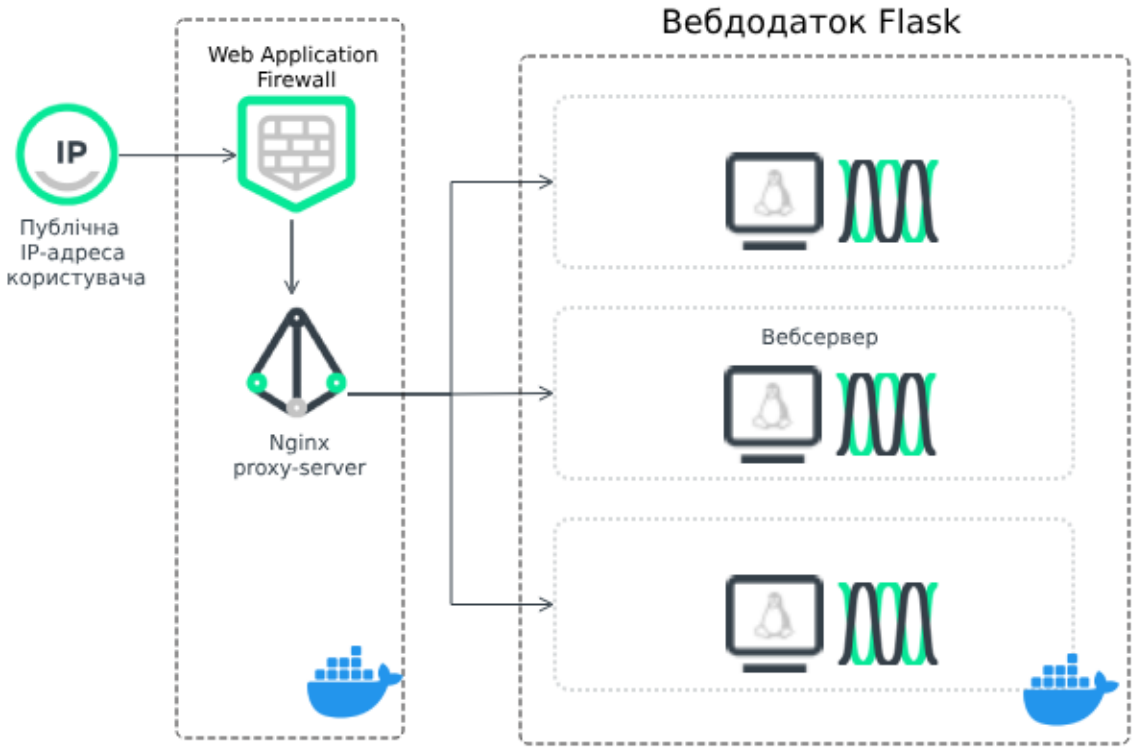

Рисунок 3.1 – Схема сегменту комп'ютерної мережі вебзастосунку з WAF на базі платформи Docker

Для розгортання контейнерів в Docker було використано образ вебзастосунку Flask з використанням базового образу Python. Для розгортання проксі-серверу Nginx з модулем ModSecurity було використано образ Nginx. Окремо було створено конфігураційний файл з налаштованими правилами безпеки. Для розгортання контейнерів було використано інструмент Docker Compose, який забезпечує можливість одночасного запуску, налаштування та керування декількома контейнерами в межах тестового середовища, визначеного у конфігураційному файлі (лістинг 3.1).

Після розгортання контейнерів вебзастосунок Flask доступний в браузері за URL – адресою http://localhost:8000.

Листинг 3.1 – Вміст конфігураційного файлу docker-compose.yml

```
version: '3.3'

services:
  app:
    build: ./app
    container_name: flask_app
    expose:
      - "8000"

  waf:
    build: ./waf
    container_name: waf_proxy
    ports:
      - "8000:80"  # доступ з публічної IP-адреси
    depends_on:
      - app
```

Для захисту вебзастосунку в другому контейнері Docker було встановлено фаєрвол вебзастосунку з попередньо встановленим модулем ModSecurity, який використовується для вебсерверів Apache, Nginx та IIS. Брандмауер у встановленому модулі ModSecurity використовує базові правила фільтрації трафіку OWASP Core Rule Set, до якого входить набір попередньо налаштованих правил фільтрації трафіку та виявлення найбільш поширених вразливостей. Опціонально до базових правил було додано, що на порту 8000 буде обслуговуватися вебзастосунок Flask, запити до якого проходять через проксі-сервер Nginx та встановлений фаєрвол вебзастосунку. Це дозволяє ізолювати вебзастосунок та підвищити рівень безпеки, оскільки за замовчанням додаток Flask працює на порту 5000. Конфігураційний файл зворотного проксі-сервера Nginx включає в себе модуль фаєрволу вебзастосунку ModSecurity, де правила фільтрації прописані в окремому конфігураційному файлі (лістинг 3.2).

Лістинг 3.2 – Вміст конфігураційного файлу проксі-сервера Nginx з модулем фаєрволу вебзастосунку ModSecurity

```
worker_processes 1;

events {
    worker_connections 1024;
}

http {
    include       mime.types;
    default_type  application/octet-stream;

    modsecurity on;
    modsecurity_rules_file /etc/modsecurity/modsecurity.conf;

    upstream backend {
        server app:8000;
    }

    server {
        listen 80;

        location / {
            proxy_pass http://backend;
        }
    }
}
```

Після завершення налаштування сегмента комп’ютерної мережі вебзастосунку з встановленим брандмауером було здійснено тестування його роботи у взаємодії зі зворотним проксі-сервером. Під час експериментальної перевірки було змодельовано вплив різних типів шкідливих запитів, зокрема SQL-ін’єкцій, XSS-ін’єкцій, а також запитів, що імітують спробу віддаленого виконання команд. Ці типи атак є одними з найпоширеніших атак щодо вебзастосунків і входять до базового набору правил, визначених проєктом OWASP. В процесі тестування, WAF успішно ідентифікував потенційні загрози та реагував на них, блокуючи доступ до серверу застосунку Flask, і повертаючи відповідь про заборону доступу. Це підтверджує ефективність обраного підходу до забезпечення безпеки вебзастосунку в умовах реального навантаження та ризиків несанкціонованого доступу до ресурсів застосунку або витоків конфіденційних даних користувачів. Виконані команди шкідливих запитів були

спрямовані до вебсервера через порт 8000 (лістинг 3.3). Отримані відповіді сервера на ці запити наведено на рисунку 3.2.

Лістинг 3.3 – Команди для проведення шкідливих запитів до вебзастосунку при тестуванні роботи WAF

```
curl -i "http://localhost:8000/?id=1 OR 1=1"
curl -i "http://localhost:8000/?q=<script>alert(1)</script>"
```

Рисунок 3.2 – Тестування вебзастосунку з використанням WAF на шкідливі запити

Також було здійснено тестування продуктивності роботи вебзастосунку в умовах підвищеного навантаження. Для моделювання високонавантажених сценаріїв застосовувалась утиліта Apache Benchmark, за допомогою якої було сформовано 1000 HTTP-запитів, що імітують одночасну обробку великої кількості запитів до вебзастосунку. Тестування проводилося у двох конфігураціях: з використанням брандмауера та зворотного проксі-сервера та без використання WAF та з реверсним проксі-сервером. Для додавання підвищеного навантаження, що імітує збільшення запитів користувачів до вебзастосунку, та збору метрик були виконані запити за портами 5000 та 8000 відповідно (лістинг 3.4).

Лістинг 3.4 – Команди для проведення імітації великого обсягу запитів на вебдодаток

```
ab -n 1000 -c 100 http:// localhost:5000/
ab -n 1000 -c 100 http:// localhost:8000/
```

В командах вказано 5000 – порт, за яким вебзастосунок працює без використання брандмауера та зворотного проксі-сервера; 8000 – порт, за яким вебзастосунок працює з використанням WAF та реверсним проксі-сервером; -n 1000 – загальна кількість запитів, що надсилаються до серверу; -c 100 – кількість одночасних з'єднань, що імітують паралельні користувацькі запити.

3.2 Розрахунок продуктивності вебзастосунку

Для розрахунку продуктивності вебзастосунку з метою оцінки його ефективності в умовах підвищеного навантаження використано практичні результати тестування та аналітичне моделювання. Результати тестування вебзастосунку з підвищеним навантаженням наведені в Додатку В. За результатами тестувань було побудовано діаграми середнього часу відповіді вебзастосунку (рисунок 3.2) та продуктивності вебзастосунку (рисунок 3.3).

Для аналітичного обґрунтування було використано модель системи масового обслуговування M/M/1, яка дозволяє описати поведінку вебзастосунку в умовах підвищеного навантаження. Такий підхід дозволяє змоделювати поведінку системи в умовах підвищеного навантаження, коли вхідні HTTP-запити надходять до фаєрволу вебзастосунку згідно з пуассонівським розподілом, тобто запити від користувачів надходять у випадковому порядку, а час їх обробки в системі додатка описується експоненційним законом. Час обслуговування запитів вебзастосунку залежить від різних факторів, зокрема ефективності кешування та складності правил фільтрації вхідного трафіку.

Модель системи масового обслуговування M/M/1 передбачає один

канал для обслуговування запитів та необмежену чергу, яку створюють вхідні запити в очікуванні обробки. Така модель дозволяє оцінити ключові характеристики роботи фаєрвола вебзастосунку, зокрема середню довжину черги, час очікування на обробку запиту, рівень навантаження додатку та час відповіді вебсервера. Обробка запитів відбувається за принципом FIFO, що відповідає реальній роботі вебсерверів. Отримані результати аналітичних розрахунків дають змогу оцінити ефективність роботи вебзастосунку з використанням WAF в умовах підвищеного навантаження, що імітують роботу вебзастосунку в режимі реального часу.

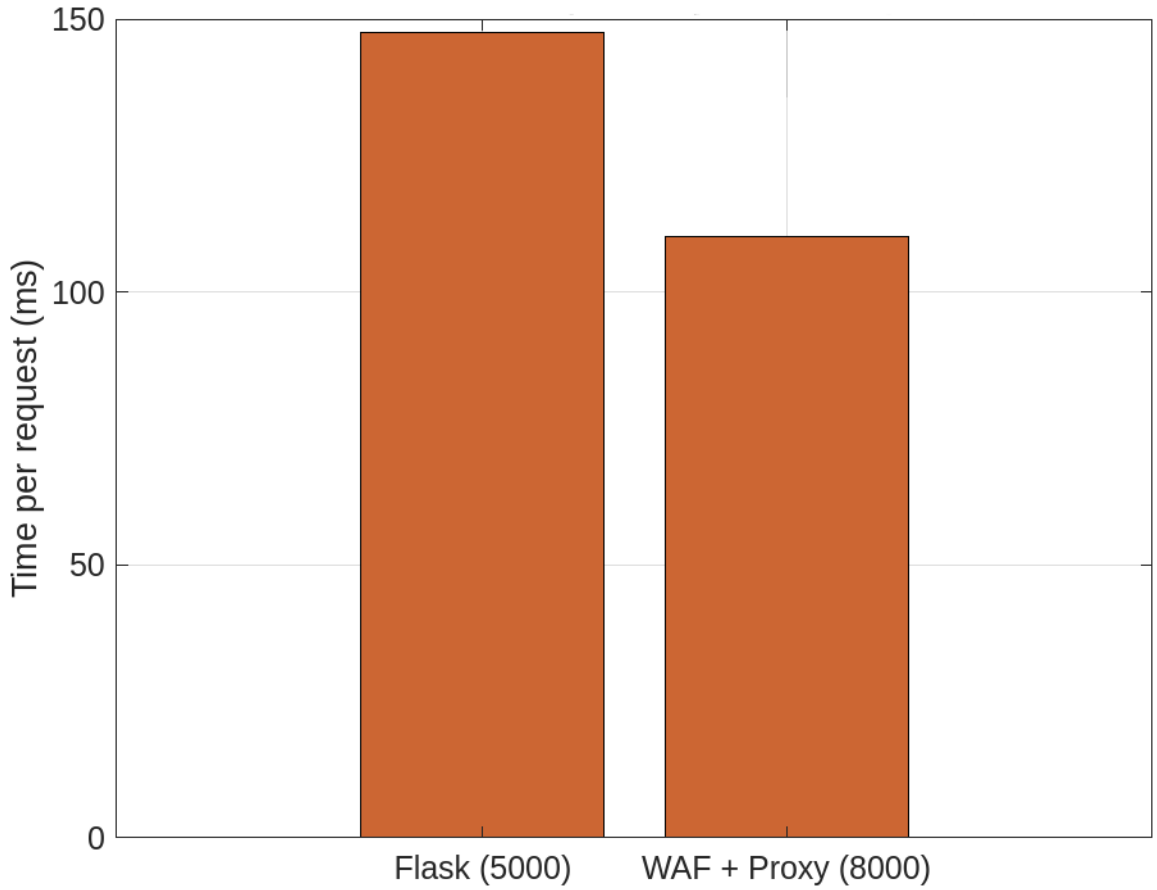

Рисунок 3.2 – Діаграма середнього часу відповіді вебзастосунку

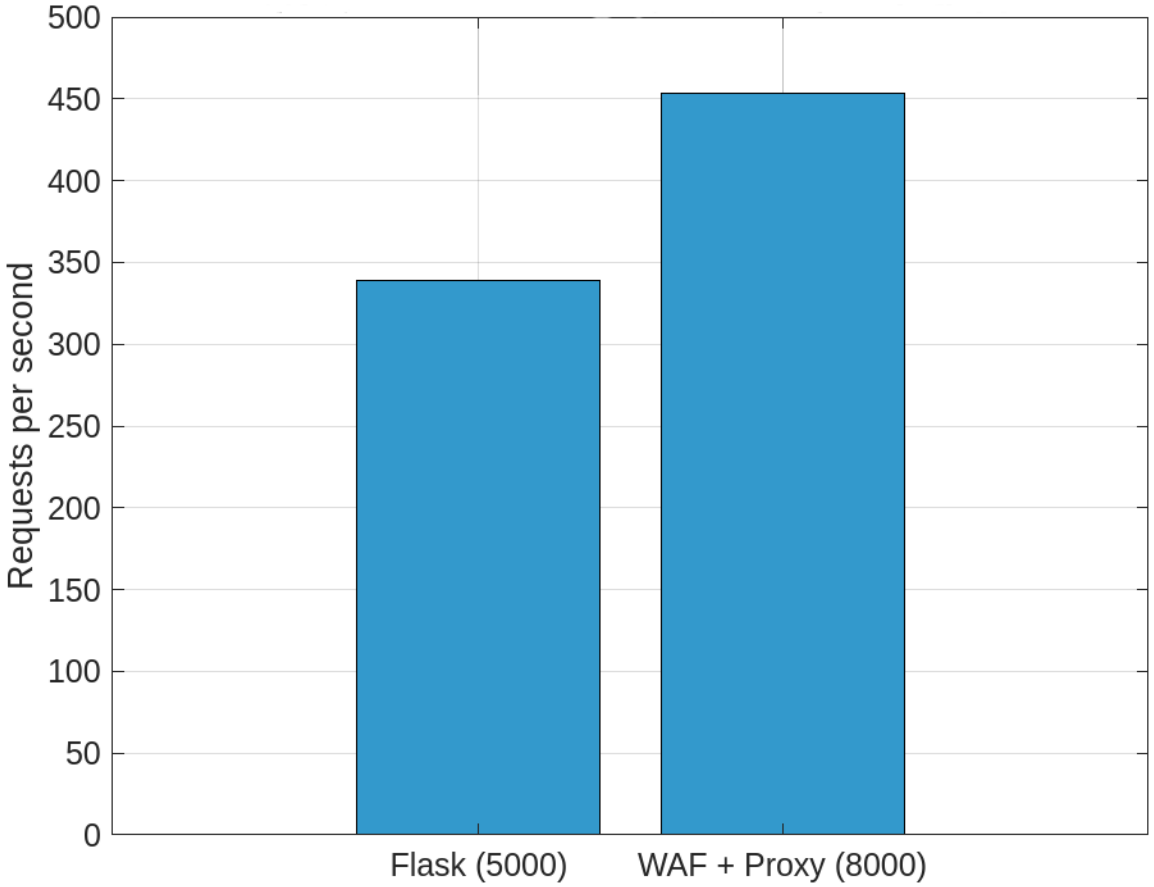

Рисунок 3.3 – Діаграма продуктивності вебзастосунку

Результати тестування роботи вебзастосунку під час навантаження з/без використання WAF та зворотного проксі-серверу наведені в таблиці 3.1.

Таблиця 3.1 – Результати аналітичного моделювання роботи вебзастосунку

Показник	**Flask**	**WAF + Proxy)**
Ймовірність надходження запитів (ρ)	0.5000	0.5294
Середня кількість запитів у системі (L)	1.0000	1.1250
Середній час перебування в системі (W), с	0.0029	0.0025
Середня кількість запитів у черзі (Lq)	0.5000	0.5956
Середній час очікування в черзі (Wq), мс	0.0015	0.0013

Основними показниками систем масового обслуговування моделі M/M/1, які характеризують роботу вебзастосунку в режимі реального часу є середні значення запитів вхідного та вихідного потоків. При цьому вихідний

потік запитів складається з кількості запитів, які перебувають в черзі на обробку сервером та кількості вже оброблених запитів [45].

Ймовірність надходження запитів ρ до системи характеризує стабільність системи, яка виражена відношенням кількості вхідних пакетів до кількості оброблених пакетів. Ймовірність надходження пакетів характеризує завантаженість системи та розраховується як:

$$\rho = \frac{\lambda}{\mu}, \tag{3.1}$$

де λ – інтенсивність вхідного потоку запитів, од;

μ – інтенсивність вихідного потоку запитів, од.

Середня кількість запитів L включає запити, які знаходяться в черзі, та запити, які обслуговуються. Значення середньої кількості запитів дозволяє оцінити ступінь завантаженості системи та прогнозувати ресурси вебзастосунку, зокрема обчислювальні апаратні ресурси сервера в умовах підвищеного навантаження. Середня кількість запитів L розраховується як:

$$L = \frac{\lambda}{\mu - \lambda}, \tag{3.2}$$

де λ – інтенсивність вхідного потоку запитів, од;

μ – інтенсивність вихідного потоку запитів, од.

Середня кількість запитів в черзі L_q є показником для визначення затримок в системі. Він дає можливість оцінити ефеективність обробки запитів в умовах підвищеного навантаження, а також надає можливість визначити потребу в масштабуванні системи або впровадженні додаткових механізмів або сервісів для стабільної та безперервної роботи вебзастосунку в умовах зростаючого навантаження. Середня кількість запитів в черзі L_q

розраховується як:

$$L_q = \frac{\lambda^2}{\mu(\mu - \lambda)}, \tag{3.3}$$

де λ – інтенсивність вхідного потоку запитів, од;

μ – інтенсивність вихідного потоку запитів, од.

Середній час перебування запитів в системі W включає час очікування пакета в черзі та час його обслуговування. Цей показник характеризує загальну затримку, яка виникає в системі під час обробки запиту. Зменшення середнього часу перебування пакету в системі свідчить про ефективну фільтрацію шкідливих запитів та швидке обслуговування корисного трафіку. Середній час перебування запитів в системі розраховується як:

$$W = \frac{1}{\mu - \lambda}, \tag{3.4}$$

де λ – інтенсивність вхідного потоку запитів, од;

μ – інтенсивність вихідного потоку запитів, од.

Середній час очікування в черзі W_q, характеризує час, який запит проводить в черзі перед обслуговуванням. Цей показник дозволяє визначити швидкість реагування системи на збільшення навантаження. Скорочення часу очікування в черзі свідчить про ефективне застосування механізмів кешування на тлі застосування правил фільтрації трафіку, які в разі ускладнення можуть призвести до збільшення часу очікування в черзі запитів на обробку. Таким чином, мінімізації середнього часу очікування в черзі свідчить про ефективність роботи вебзастосунку з використанням WAF та зворотного проксі-серверу. Середній час очікування в черзі розраховується як:

$$W_q = \frac{\lambda}{\mu(\mu - \lambda)}, \tag{3.5}$$

де λ – інтенсивність вхідного потоку запитів, од;

μ – інтенсивність вихідного потоку запитів, од.

Для виконання розрахунків показників моделі M/M/1 використано середовище MATLAB. За результатами моделювання роботи вебзастосунку з використанням WAF та зворотного проксі-сервера як моделі M/M/1 було отримано наступні аналітичні показники:

λ – інтенсивність вхідного потоку запитів до системи дорівнює 555.56 зап/с;

μ – інтенсивність вихідного потоку запитів дорівнює 1000.00 зап/с;

ρ – ймовірність надходження запитів дорівнює 0.5556;

L – середня кількість запитів системі складає 1.2500;

L_q – середня кількість запитів в черзі на обслуговування складає 0.6944;

W – середній час перебування запитів в системі складає 0.0022 с;

W_q – середній час очікування запита в черзі на обробку складає 0.0013 с.

Також було побудовано графік залежності середнього часу перебування запиту в системі від інтенсивності вхідного потоку запитів (рисунок 3.4). Розроблений код розрахунку показників, що реалізує дану модель та побудову графіка, наведено в додатку Г.

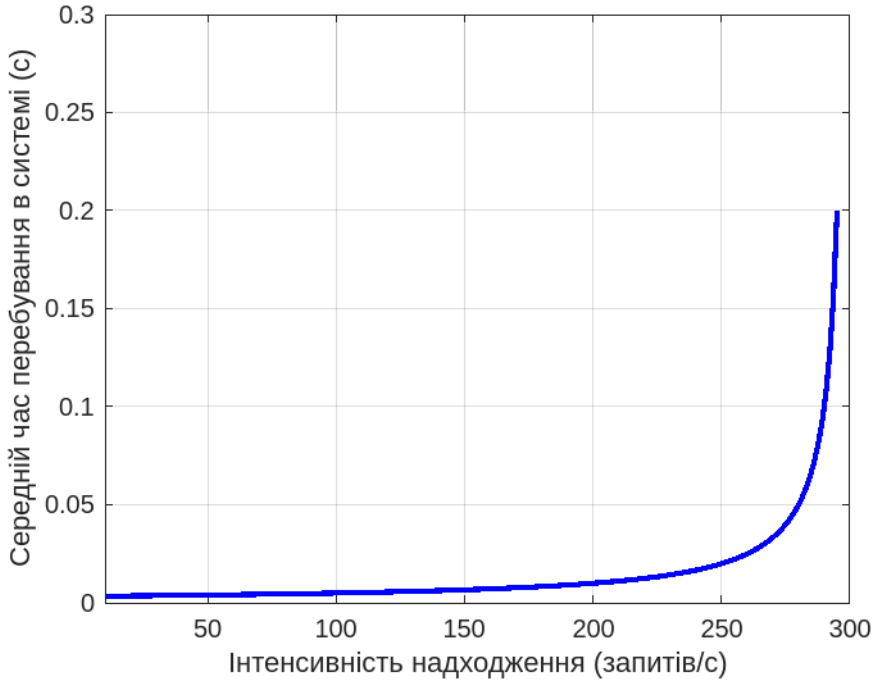

Рисунок 3.4 – Графік залежності середнього часу перебування запиту в системі від інтенсивності вхідного потоку запитів

Результати розрахунків підтверджують різке зростання часу перебування запитів в системі при досягненні максимальної кількості вхідних запитів, що відповідає роботі застосунку в умовах підвищеного навантаження.

Отримані результати тестування та розрахунків свідчать, що впровадження WAF для забезпечення захисту від несанкціонованого доступу та ризиків витоку даних та зворотного проксі-серверу для кешування запитів дозволяє підвищити продуктивність вебзастосунку. Зокрема при тестуванні та аналітичному моделюванні спостерігається зменшення середнього часу перебування запиту в системі та часу очікування в черзі, що забезпечує зниження затримок та підвищення швидкодії вебзастосунку в умовах високого навантаження.

ВИСНОВКИ

В процесі виконання кваліфікаційної роботи було досягнуто основну мету – розроблено метод підвищення продуктивності вебзастосунку з використанням WAF, що дозволяє зменшити затримки під час обробки великої кількості запитів і водночас забезпечити належний рівень захисту від сучасних кіберзагроз.

На основі аналізу наукових досліджень за тематикою роботи було запропоновано практичне рішення, що дозволяє забезпечити високу продуктивність вебзастосунку з належним рівнем безпеки. Особливу увагу в роботі приділено механізмам балансування навантаження, які сприяють стабільному функціонуванню вебзастосунку при одночасній обробці великої кількості запитів та підвищеній ймовірності атак.

Результати експериментального тестування свідчать, що запропонована архітектура з використанням WAF в поєднанні з реверсним проксі-сервером демонструє на 26% вищу ефективність порівняно з базовим варіантом, особливо за умов високого навантаження. Для підвищення рівня безпеки було впроваджено набір правил фільтрації вхідного трафіку, що дозволяють блокувати найбільш поширені загрози.

Аналітичне моделювання функціонування вебзастосунку було здійснено з використанням моделі масового обслуговування, що дозволила описати роботу запропонованого методу як єдиної системи обробки запитів.

Запропонований метод може бути застосовано для вебзастосунків, які функціонують в режимі високого навантаження з підвищеними вимогами до безпеки. Отримані результати підтверджують практичну доцільність впровадження розробленої архітектури в реальних умовах експлуатації.

Подальші дослідження доцільно спрямувати на вдосконалення даного підходу шляхом інтеграції з сучасними системами моніторингу та механізмами масштабування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Цифровізація: переваги та шляхи подолання викликів. [Електронний ресурс] / Центр Разумкова. – 2025. – Режим доступу до ресурсу:https://razumkov.org.ua/statti/tsyfrovizatsiia-perevagy-ta-shliakhy-podolannia-vyklykiv (дата звернення: 11.05.2025).

2. Дудкін, П. Д., Дудкіна, О. П. (2022). Функціонування інфраструктури: деякі аспекти в умовах сучасних викликів. Матеріали X Всеукраїнської науково-практичної конференції пам'яті почесного професора Тернопільського національного технічного університету імені Івана Пулюя, академіка НАН України Миколи Григоровича Чумаченка: «Соціальна відповідальність як основа інноваційного розвитку бізнесу», 42-43.

3. Aslan, Ö., Aktuğ, S. S., Ozkan-Okay, M., Yilmaz, A. A., & Akin, E. (2023). A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions. Electronics, 12(6), 1333.

4. Карабанов, Д. С., Чеботарьова, Д. В. «Дослідження засобів безпеки в системах електронної комерції 28-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті». Зб. Матеріалів форуму. Т.4. – Харків: ХНУРЕ. 2024. – с. 116.

5. Malallah, H., Zeebaree, S. R., Zebari, R. R., Sadeeq, M. A., Ageed, Z. S., Ibrahim, I. M., Merceedi, K. J. (2021). A comprehensive study of kernel (issues and concepts) in different operating systems. Asian Journal of Research in Computer Science, 8(3), 16-31.

6. Gao, X., Liu, L., & Zhu, X. (2021). Research on the main threat and prevention technology of computer network security. In IOP Conference Series: Earth and Environmental Science (Vol. 632, No. 5, p. 052065). IOP Publishing.

7. Y. Xu, «Research on Computer Information Network Security Technology and Development Direction», JCEIM, vol. 16, no. 2, pp. 21–24, Mar. 2025, doi: 10.54097/zjqkkb50.

8. Perwej, Y., Abbas, S. Q., Dixit, J. P., Akhtar, N., & Jaiswal, A. K. (2021). A systematic literature review on the cyber security. International Journal of scientific research and management, 9(12), 669-710.

9. Alghofaili Y, Albattah A, Alrajeh N, Rassam MA, Al-rimy BAS. Secure Cloud Infrastructure: A Survey on Issues, Current Solutions, and Open Challenges. Applied Sciences. 2021; 11(19):9005. https://doi.org/10.3390/app11199005.

10. Jangjou, M., & Sohrabi, M. K. (2022). A comprehensive survey on security challenges in different network layers in cloud computing. Archives of Computational Methods in Engineering, 29(6), 3587-3608.

11. Sharma, H. (2021). Next-Generation Firewall in the Cloud: Advanced Firewall Solutions to the Cloud. ESP Journal of Engineering & Technology Advancements (ESP-JETA), 1(1), 98-111.

12. Omer, M. A., Yazdeen, A. A., Malallah, H. S., & Abdulrahman, L. M. (2022). A survey on cloud security: concepts, types, limitations, and challenges. Journal of Applied Science and Technology Trends, 3(02), 101-111

13. А. Махдумі, Н. Ян, Палак та Н. Гоел, «Традиційні та наступні покоління брандмауерів у мережевій безпеці та їх застосування», Міжнародна конференція з обчислювальної техніки, зв'язку та інтелектуальних систем (ICCCIS) 2022 року, Велика Нойда, Індія, 2022, с. 964-969, doi: 10.1109/ICCCIS56430.2022.10037674.

14. D. Bringhenti and F. Valenza, «GreenShield: Optimizing Firewall Configuration for Sustainable Networks,» in IEEE Transactions on Network and Service Management, vol. 21, no. 6, pp. 6909-6923, Dec. 2024, doi: 10.1109/TNSM.2024.3452150.

15. R. Alsaqour, A. Motmi, and M. Abdelhaq, «A Systematic Study of Network Firewall and Its Implementation,» International Journal of Computer Science and Network Security, vol. 21, no. 4, pp. 199–208, Apr. 2021.

16. J. Liang and Y. Kim, «Evolution of Firewalls: Toward Securer Network Using Next Generation Firewall,» 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2022,

pp. 0752-0759, doi: 10.1109/CCWC54503.2022.9720435.

17. Кізза, Джозеф Мігга. «Брандмауери». Посібник з безпеки комп’ютерних мереж. Cham: Springer International Publishing, 2024. 265-294.

18. Басак, Д., Тошнівал, Р., Маскалік, С. та Секейра, А. (2010). Віртуалізація мереж та безпека в хмарі. ACM SIGOPS Operating Systems Review , 44 (4), 86-94.

19. Що таке файрвол та навіщо він потрібен. [Електронний ресурс] / GigaTrans - Интернет для Вашего офиса. – 2025. – Режим доступу до ресурсу: https://gigatrans.ua/ua/news/chto-takoe-fayrvol-i-zachem-on nuzhen#:~:text=вогняна%20стіна)%20–%20це%20міжмережевий%20екран,які%20вважає%20підозрілими%20або%20шкідливими. (дата звернення: 11.05.2025).

20. Фаєрволи (брандмауери): що це і для чого потрібні [Електронний ресурс] / Макснет – Гігабітний інтернет-провайдер та оператор зв'язку для дому та бізнесу. – 2025. – Режим доступу до ресурсу: https://maxnet.ua/blog/fayervoli-brandmaueri-sho-ce-i-dlya-chogo-potribni/ (дата звернення: 11.05.2025).

21. L. Durante, L. Seno and A. Valenzano, «A Formal Model and Technique to Redistribute the Packet Filtering Load in Multiple Firewall Networks,» in IEEE Transactions on Information Forensics and Security, vol. 16, pp. 2637-2651, 2021, doi: 10.1109/TIFS.2021.3057552.

22. Praseed, A., & Thilagam, P. S. (2022). HTTP request pattern based signatures for early application layer DDoS detection: A firewall agnostic approach. Journal of Information Security and Applications, 65, 103090.

23. X. Chen, Q. Shen, P. Cheng, Y. Xiong and Z. Wu, «RuleCache: Accelerating Web Application Firewalls by On-line Learning Traffic Patterns,» 2022 IEEE International Conference on Web Services (ICWS), Barcelona, Spain, 2022, pp. 229-239, doi: 10.1109/ICWS55610.2022.00044.

24. S. Lozano, T. Lugo and J. Carretero, "A Comprehensive Survey on the Use of Hypervisors in Safety-Critical Systems," in IEEE Access, vol. 11, pp. 36244-36263, 2023, doi: 10.1109/ACCESS.2023.3264825.

25. Berggren, J., & Karlsson, J. (2022). Differences in performance between containerization & virtualization: With a focus on HTTP requests.

26. Docker vs Podman для контейнеризації: що обрати. [Електронний ресурс] / IT Education Center Blog. – 2025. – Режим доступу до ресурсу: https://itedu.center/ua/blog/comparisons/docker_chy_podman_shcho_obraty_dlia_konteineryzatsii/?srsltid=AfmBOooysBUD-3r-NT2OKcN-e9ESMgA2G7Es7oF1dQHFjOhuvw1Pf8B6 (дата звернення: 22.05.2025).

27. ТКАЧОВ, В., ЧЕПУРНА, І., & ФЕСЕНКО, Т. (2024). МЕТОД МУЛЬТИРІВНЕВОГО VPN-ТУНЕЛЮВАННЯ ДЛЯ ЗАБЕЗПЕЧЕННЯ ВІДДАЛЕНОГО ДОСТУПУ ДО ВУЗЛІВ ЕКСТРАНЕТ-МЕРЕЖІ. Вісник Херсонського національного технічного університету, (3 (90)), 299-308.

28. Fahmi, H. Z., Buditjahjanto, I. G. P. A., Hafidz, A., Dermawan, D. A., Sidhimantra, I. G. A. S., & Saputra, R. N. (2024, May). Optimizing resource availability on high scalability with container technology through a load balancing approach. In AIP Conference Proceedings (Vol. 3116, No. 1, p. 100017). AIP Publishing LLC.

29. Moscato, R. (2024). Web App Development Made Simple with Streamlit: A web developer's guide to effortless web app development, deployment, and scalability. Packt Publishing Ltd.

30. Samsuddin, N., Darus, M. Y., & Mohd Ariffin, M. A. (2024). A new framework for deploying virtual dekstop infrastructure using containerization approach. Malaysian Journal of Computing (MJoC), 9(2), 1838-1851.

31. Cherukuri, B. R. (2024). Containerization in cloud computing: comparing Docker and Kubernetes for scalable web applications.

32. Ezeobi, J. (2024). AN INVESTIGATION INTO COMPATIBILITY ISSUES BETWEEN DOCKER AND PODMAN CONTAINER TECHNOLOGIES.

33. Jung, S. (2024). Multithreaded applications on the heterogeneous research computing environment.

34. Younis, R., Iqbal, M., Munir, K., Javed, M. A., Haris, M., & Alahmari, S. (2024, October). A Comprehensive Analysis of Cloud Service Models: IaaS, PaaS, and SaaS in the Context of Emerging Technologies and Trend. In 2024 International Conference on Electrical, Communication and Computer Engineering (ICECCE) (pp. 1-6). IEEE.

35. Suleiman, N., & Murtaza, Y. (2024). Scaling microservices for enterprise applications: comprehensive strategies for achieving high availability, performance optimization, resilience, and seamless integration in large-scale distributed systems and complex cloud environments. Applied Research in Artificial Intelligence and Cloud Computing, 7(6), 46-82.

36. Zhang, X., He, Q., Fan, H., & Wu, S. (2024, November). Faascale: Scaling MicroVM Vertically for Serverless Computing with Memory Elasticity. In Proceedings of the 2024 ACM Symposium on Cloud Computing (pp. 196-212).

37. Razavi, K., Salmani, M., Mühlhäuser, M., Koldehofe, B., & Wang, L. (2024). A Tale of Two Scales: Reconciling Horizontal and Vertical Scaling for Inference Serving Systems. arXiv preprint arXiv:2407.14843.

38. Gogineni, N., & Sivalingam, S. M. (2024). A systematic review on recent methods of scheduling and load balancing for containers in distributed environments. International Journal of Advanced Technology and Engineering Exploration, 11(116), 1030.

39. Балансування навантаження: як це впливає на продуктивність - Рейтинг проксі. [Електронний ресурс] / Рейтинг проксі. – 2025. – Режим доступу до ресурсу: https://proxys-rating.com/balansuvannya-navantazhennya-yak-cze-vplyvaye-na-produktyvnist/ (дата звернення: 22.05.2025).

40. Mousavi Jalali, S. M., & Rezaei, F. (2025). IoT data caching and scheduling in heterogeneous mobile networks by exploiting mobility and transient data characteristics. Peer-to-Peer Networking and Applications, 18(4), 1-31.

41. Головльов С. Що таке кешування даних. [Електронний ресурс] / IT Ресурс. Resit. – 2025. – Режим доступу до ресурсу: https://resit.com.ua/scho-take-keshirovanie-dannyh-viznachennya-vidy-tipy-i-dr/ (дата звернення: 22.05.2025).

42. Сервіси кешування для сайтів: короткий огляд Memcached, OPCache і Redis. [Електронний ресурс] / Хостинг в Украине от Cityhost – лучший хостинг сайта. – 2025. – Режим доступу до ресурсу: https://cityhost.ua/uk/blog/servisy-keshirovaniya-dlya-saytov-memcached-opcache-redis.html (дата звернення: 22.05.2025).

43. Кроки для налаштування зворотного проксі Nginx і Apache. [Електронний ресурс] / IT Education Center Blog. . – 2025. – Режим доступу до ресурсу:https://itedu.center/ua/blog/guides/yak-nalashtuvaty-nginx-i-apache-u-roli-zvorotnoho-proksi/?srsltid=AfmBOorsR2Qq3klXzUGXhK4Js3K0K1Jr0jm1CnEMAMdORvTi212mUQqP (дата звернення: 22.05.2025).

44. Aqasizade, H., Ataie, E., & Bastam, M. (2025). Experimental assessment of containers running on top of virtual machines. IET Networks, 14(1), e12138.

45. Moltafet, M., Leinonen, M., & Codreanu, M. (2020). Average Age of Information for a Multi-Source M/M/1 Queueing Model With Packet Management. In 2020 IEEE International Symposium on Information Theory (ISIT), Los Angeles, CA, USA. https://doi.org/10.1109/isit44484.2020.9174099.