Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет *інфокомунікацій*
(повна назва)

Кафедра *інформаційно-мережної інженерії*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів захисту даних у системах дистанційного тестування та опитувань
(тема)

Виконав:
здобувач 2 курсу, групи ІМІм-24-1
Сухицький Б.С.
(прізвище, ініціали)

Спеціальність 172 Електронні комунікації та радіотехніка
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма інформаційно-мережна інженерія
(повна назва освітньої програми)

Керівник доцент Ляшенко Г.Є.
(посада, прізвище, ініціали)

Допускається до захисту
Завідувач кафедри _____ _____ Микола МОСКАЛЕЦЬ
(підпис) (прізвище, ініціали)

20 25 р.

Не містить відомостей заборонених до відкритого публікування.

Студент / Сухицький Б.С. /

Керівник / Ляшенко Г.Є. /

Харківський національний університет радіоелектроніки

Факультет *інфокомунікацій*
Кафедра *інформаційно-мережної інженерії*
Рівень вищої освіти другий (мгістерський)
Спеціальність *172 Електронні комунікації та радіотехніка*
(код і повна назва)

Тип програми освітньо-професійна
Освітня програма інформаційно-мережна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві *Сухицькому Богдану Сергійовичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження методів захисту даних у системах дистанційного тестування та опитувань*

затверджена наказом університету від 24 жовтня 20 25 р. № 959Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 19 грудня 20 25 р.
3. Вихідні дані до роботи
Архітектура та функціональні можливості систем дистанційного тестування та опитувань, структура бази даних користувачів та результатів тестування.
Програмні засоби та середовище розробки: OpenServer, вебсервер Apache, PHP, система керування базами даних MySQL, засоби адміністрування баз даних phpMyAdmin.
Вихідні вимоги до системи дистанційного тестування: забезпечення автентифікації користувачів, захист персональних даних, стійкість до SQL-ін'єкцій та безпечне зберігання результатів тестування.

4. Перелік питань, що потрібно опрацювати в роботі
Вступ
1. Аналіз систем дистанційного тестування та опитувань
2. Методи та засоби захисту даних
3. Аналіз сервісів для забезпечення захисту в системах тестувань та опитувань
4. Проєктування системи дистанційного тестування та оцінка ефективності методів захисту
Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) _____
Слайди у форматі Power Point

6. Консультанти розділів роботи (п.6 включається до завдання 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
Основна частина	доцент Ляшенко Г.Є.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Ознайомлення з завданням	26.10.25	виконано
2	Пошук джерел за темою та ознайомлення	26.10.25 − 30.10.25	виконано
3	Виконання розділу 1	30.10.25 − 10.11.25	виконано
4	Виконання розділу 2	11.11.25 − 17.11.25	виконано
5	Виконання розділу 3	17.11.25 − 23.11.25	виконано
6	Виконання розділу 4	23.11.25 − 15.12.25	виконано
7	Оформлення презентаційних матеріалів матеріалу	15.12.25 − 19.12.25	виконано
8	Підготовка до захисту роботи	19.12.25 − 24.12.25	виконано

Дата видачі завдання 26 жовтня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ *доцент Ляшенко Г.Є.*

РЕФЕРАТ

Записка містить 96 сторінки, 8 рисунків, 17 таблиць, 2 додатки і 24 посилання.

Об’єкт дослідження – система тестувань та опитувань.

Мета дослідження – дослідження методів захисту даних у системах дистанційного тестування

Було досліджено комплекс заходів захисту, зокрема захищене з’єднання, фільтрацію трафіку, захист від DDoS-атак і базові механізми контролю доступу. Розроблено та реалізовано вебсистему дистанційного тестування на основі трирівневої архітектури з використанням JavaScript, CSS, PHP, MySQL та середовища OpenServer. Експериментально підтверджено ефективність використання параметризованих SQL-запитів, що дозволило підвищити рівень захисту від SQL-ін’єкцій.

ДИСТАНЦІЙНЕ ТЕСТУВАННЯ, ВЕБСИСТЕМА, SQL-ІН’ЄКЦІЯ, БАЗА ДАНИХ, MYSQL, OPENSERVER.

THE ABSTRACT

The note contains 96 pages, 8 figures, 17 tables, 2 appendices, and 24 references.

The object of the study is a system of testing and surveys.

The purpose of the study is to study data protection methods in remote testing systems

A set of protection measures was studied, including a secure connection, traffic filtering, protection against DDoS attacks and basic access control mechanisms. A remote testing web system based on a three-tier architecture using JavaScript, CSS, PHP, MySQL and the OpenServer environment was developed and implemented. The effectiveness of using parameterized SQL queries was experimentally confirmed, which allowed to increase the level of protection against SQL injections.

REMOTE TESTING, WEB SYSTEM, SQL INJECTION, DATABASE, MYSQL, OPENSERVER.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних;

SQL – Structured Query Language;

SQLI – SQL Injection;

HTML – HyperText Markup Language;

HTTP – HyperText Transfer Protocol;

HTTPS – HyperText Transfer Protocol Secure;

CSS – Cascading Style Sheets;

PHP – Hypertext Preprocessor;

SSL – Secure Sockets Layer;

TLS – Transport Layer Security;

GDPR – General Data Protection Regulation;

PBKDF2 – Password-Based Key Derivation Function 2;

AES – Advanced Encryption Standard;

NIST – National Institute of Standards and Technology;

DES – Data Encryption Standard;

3DES – Triple Data Encryption Standard;

RSA – Rivest–Shamir–Adleman;

ECC – Elliptic Curve Cryptography;

MD5 – Message Digest 5;

SHA-1 – Secure Hash Algorithm 1;

SHA-2 – Secure Hash Algorithm 2;

SHA-3 – Secure Hash Algorithm 3;

OTP – One-Time Password;

2FA – Two-Factor Authentication;

JWT – JSON Web Token;

JSON – JavaScript Object Notation;

XSS – Cross-Site Scripting;

CSRF – Cross-Site Request Forgery;

XML – eXtensible Markup Language;

URL – Uniform Resource Locator;

CSP – Content Security Policy;

IDS – Intrusion Detection System;

IPS – Intrusion Prevention System;

ECDHE – Elliptic Curve Diffie–Hellman Ephemeral;

HSM – Hardware Security Module;

HMAC – Hash-based Message Authentication Code;

PGP – Pretty Good Privacy;

AES-GCM – Advanced Encryption Standard – Galois Counter Mode;

RSA-OAEP – Rivest–Shamir–Adleman Optimal Asymmetric Encryption Padding;

ECDSA – Elliptic Curve Digital Signature Algorithm;

OWASP – Open Web Application Security Project;

WAF – Web Application Firewall;

IAM – Identity and Access Management;

KMS – Key Management Service;

API – Application Programming Interface;

DV – Domain Validation;

OV – Organization Validation;

EV – Extended Validation;

ACME – Automatic Certificate Management Environment;

SAN – Subject Alternative Name;

VPS – Virtual Private Server;

CPU – Central Processing Unit;

RAM – Random Access Memory;

SSD – Solid State Drive;

DDoS – Distributed Denial of Service;

EC2 – Elastic Compute Cloud;

RDS – Relational Database Service;

S3 – Simple Storage Service;

GCP – Google Cloud Platform;

ORM – Object-Relational Mapping;

DAST – Dynamic Application Security Testing;

ACL – Access Control List;

WAAP – Web Application and API Protection;

CDN – Content Delivery Network;

IIS – Internet Information Services;

CRS – Core Rule Set;

DOM – Document Object Model;

HSTS – HTTP Strict Transport Security;

MITM – Man-In-The-Middle;

ELB – Elastic Load Balancing;

MFA – Multi-Factor Authentication;

SaaS – Software as a Service;

DNS – Domain Name System;

SLA – Service Level Agreement;

SDK – Software Development Kit;

ВСТУП

У сучасних умовах цифрової трансформації суспільства дистанційні форми навчання, тестування та опитувань набувають особливого значення. Зокрема, у закладах освіти, на підприємствах і в державних установах активно впроваджуються системи, що дозволяють проводити оцінювання знань, анкетування та сертифікацію в онлайн-режимі. Такі системи забезпечують гнучкість, доступність і масштабованість освітніх процесів, однак одночасно створюють нові виклики у сфері захисту інформації.

У процесі функціонування систем дистанційного тестування обробляються великі обсяги персональних даних користувачів, а також результати тестів, логіни, паролі, електронні адреси, статистика активності тощо. Несанкціонований доступ або витік цієї інформації може призвести до порушення конфіденційності, фальсифікації результатів тестування чи маніпуляцій з даними. Тому питання забезпечення безпеки даних у таких системах є надзвичайно актуальним.

З розвитком інформаційних технологій зростає кількість кібератак на освітні ресурси, що підкреслює необхідність впровадження надійних механізмів автентифікації, шифрування та контролю доступу. У зв’язку з цим дослідження методів і засобів захисту даних у системах дистанційного тестування та опитувань є важливим як з теоретичної, так і з практичної точки зору.

Незважаючи на наявність сучасних криптографічних алгоритмів, протоколів захищеного з’єднання та засобів контролю доступу, на практиці значна кількість вебсистем реалізується з порушенням базових принципів безпеки. Це зумовлює необхідність не лише теоретичного аналізу методів захисту даних, але й їх практичного впровадження та експериментальної перевірки ефективності у реальних умовах функціонування системи.

Метою кваліфікаційної роботи є дослідження та оцінка ефективності методів захисту даних у системах дистанційного тестування та опитувань

шляхом проєктування, реалізації та експериментального тестування захищеної вебсистеми.

Об’єктом дослідження є системи дистанційного тестування та опитувань, що функціонують у мережі Інтернет.

Предметом дослідження є методи та засоби забезпечення захисту даних у цих системах, зокрема способи автентифікації, контролю доступу та запобігання несанкціонованим діям користувачів.

Практичне значення роботи полягає у створенні та дослідженні захищеної системи дистанційного тестування, розгорнутої в середовищі Open Server, а також у розробці методики оцінювання ефективності захисту від SQL-ін’єкцій, яка може бути використана під час проєктування та аудиту безпеки подібних вебсистем.

Результати роботи можуть бути використані під час створення або модернізації вебсистем дистанційного тестування та опитувань у навчальних закладах, на підприємствах або в організаціях, що потребують безпечної онлайн-оцінки знань.

1 АНАЛІЗ СИСТЕМ ДИСТАНЦІЙНОГО ТЕСТУВАННЯ ТА ОПИТУВАНЬ

1.1 Поняття та класифікація систем дистанційного тестування

Дистанційне тестування – це форма контролю знань або збору інформації, що здійснюється за допомогою інформаційно-комунікаційних технологій у режимі онлайн. Такі системи забезпечують проведення тестів, опитувань або анкетувань без необхідності фізичної присутності користувача у навчальному чи робочому середовищі [1].

Системи дистанційного тестування можна класифікувати за різними ознаками:

– за цільовим призначенням: навчальні (для освітніх закладів), корпоративні (для підготовки персоналу), дослідницькі (для збору статистичних даних);

– за способом доступу: відкриті (публічні вебплатформи) та закриті (локальні системи організацій);

– за типом архітектури: клієнт-серверні, хмарні, гібридні;

– за типом тестування: автоматизовані (з автоматичною перевіркою відповідей) та ручні (з участю викладача або адміністратора).

Основною метою таких систем є забезпечення зручного, масштабованого та надійного процесу оцінювання знань або збору аналітичної інформації. Проте разом із розширенням функціональності виникає проблема – забезпечення конфіденційності, цілісності та достовірності даних користувачів.

1.2 Архітектура та принципи роботи онлайн-платформ для тестування

Типова архітектура системи дистанційного тестування має трирівневу структуру. Схема цієї архітектури показана на рисунку 1.1:

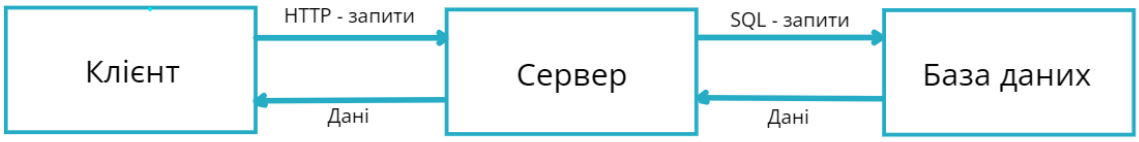

Рисунок 1 – Трирівнева архітектура програмного комплексу

Трирівнева архітектура є популярним підходом до побудови програмних систем, який забезпечує розділення відповідальності між різними компонентами та полегшує масштабування, підтримку і розвиток системи. Трирівнева архітектура складається з трьох основних рівнів: презентаційного, логіки та даних [2].

Презентаційний рівень – це інтерфейс користувача, створений за допомогою HTML, CSS та JavaScript. Він відповідає за зручність взаємодії, відображення питань, варіантів відповідей, результатів тестування.

Логічний рівень (рівень застосунку) – реалізує бізнес-логіку системи: обробку запитів, перевірку відповідей, збереження результатів, формування звітів. Як правило, створюється за допомогою мов програмування PHP, Python, Java або Node.js.

Рівень даних – це система управління базами даних (наприклад, MySQL, PostgreSQL, MongoDB), яка зберігає інформацію про користувачів, тести, відповіді та статистику.

Коли користувач взаємодіє з інтерфейсом користувача, дані відправляються на логічний рівень для обробки. Після обробки дані можуть бути збережені або оновлені в базі даних рівня даних. Відповідно, інформація може бути отримана з бази даних та відображена на презентаційному рівні для користувача.

Принцип роботи таких систем полягає у наступному:

– користувач авторизується в системі;

– отримує тест або опитування відповідно до свого рівня доступу;

– відповіді передаються на сервер;

– система автоматично обробляє результати та зберігає їх у базі даних.

Архітектурна гнучкість дозволяє інтегрувати модулі захисту – наприклад, SSL/TLS-з’єднання, токени безпеки, криптографічні механізми шифрування даних.

1.3 Аналіз сучасних систем дистанційного тестування та опитувань

На сьогодні існує велика кількість програмних платформ для онлайн-тестування, які відрізняються функціональністю, масштабованістю та рівнем захисту даних. Найпопулярніші з них :

– Moodle – відкрита система дистанційного навчання з підтримкою курсів, тестів, форумів та звітів. Має базові механізми авторизації, резервного копіювання та захисту сесій користувачів. В Європі 2/3 закладів вищої освіти використовують саме Moodle [3];

– Google Forms – онлайн-сервіс для створення анкет і тестів, інтегрований із Google Drive. Перевага – простота використання, недолік – обмежені можливості контролю безпеки [4];

– Testportal – хмарна платформа для проведення тестів із можливістю встановлення часових обмежень, запобігання копіюванню запитань і захистом від повторного проходження тесту [5];

– Kahoot! – інтерактивна система для тестування в ігровій формі. Основна увага приділяється гейміфікації, але безпекові механізми залишаються на базовому рівні [6].

Більшість популярних систем зосереджуються на функціональності та зручності користувачів, але питання безпеки даних часто залишаються недостатньо опрацьованими: зберігання паролів у незашифрованому вигляді, слабкий контроль доступу, відсутність багаторівневої автентифікації.

1.4 Вимоги до безпеки та збереження даних у вебсистемах тестування

Згідно з міжнародними стандартами (ISO/IEC 27001, GDPR, NIST SP 800-53), вебсистеми, які обробляють персональні або навчальні дані, повинні відповідати таким вимогам [7]:

– конфіденційність – захист інформації від несанкціонованого доступу (наприклад, через шифрування або контроль доступу);

– цілісність – забезпечення незмінності даних під час передачі та зберігання (використання хешування, цифрових підписів);

– доступність – забезпечення безперервної роботи системи навіть у разі збоїв або атак;

– автентифікація та авторизація – підтвердження особи користувача й розмежування рівнів доступу;

– аудит та журналювання – ведення логів дій користувачів і адміністраторів для подальшого аналізу.

Недотримання цих вимог може призвести до витоку конфіденційної інформації, фальсифікації результатів тестування або несанкціонованих змін у базі даних.

Більшість існуючих рішень забезпечують базову функціональність, проте не гарантують належного рівня захисту персональних даних користувачів і результатів тестування.

Отже, необхідним є розроблення удосконалених методів захисту даних, які враховують специфіку вебдодатків, характер оброблюваної інформації та потенційні ризики безпеки. Це визначає напрям подальших досліджень, що розглядаються в наступних розділах роботи.

2 МЕТОДИ ТА ЗАСОБИ ЗАХИСТУ ДАНИХ

2.1 Загальні принципи інформаційної безпеки

Інформаційна безпека – це стан захищеності інформації, який забезпечує її конфіденційність, цілісність та доступність.

У контексті систем дистанційного тестування та опитувань вона означає гарантування того, що дані користувачів, результати тестів і службова інформація не будуть розголошені, змінені або знищені сторонніми особами [8].

Основні принципи забезпечення безпеки інформації:

– конфіденційність – захист даних від несанкціонованого доступу (шифрування, аутентифікація, розмежування прав);

– цілісність – гарантія незмінності даних під час обробки та зберігання (контроль цілісності, цифрові підписи);

– доступність – забезпечення безперебійної роботи системи навіть у разі збоїв чи атак;

– автентичність – підтвердження справжності джерела інформації;

– невідмовність – запобігання можливості заперечення факту виконання певних дій (використання цифрових підписів, журналів аудиту).

Дотримання цих принципів є ключовим для побудови захищених систем тестування, які обробляють персональні дані та результати оцінювання.

2.2 Методи шифрування та криптографічного захисту даних

Одним з основних напрямів захисту інформації є криптографія, що забезпечує перетворення даних у вигляд, недоступний для сторонніх осіб.

У вебсистемах тестування застосовуються три основні категорії криптографічних методів: симетричне, асиметричне шифрування та хешування.

Симетричне шифрування – криптографічний підхід, при якому один спільний секретний ключ використовується для шифрування й дешифрування даних. Цей метод забезпечує високу швидкість і ефективність, тож застосовується для шифрування дисків та великих обсягів даних [9].

Принцип роботи симетричного шифрування:

– генерація ключів – передбачає підбір приватного ключа. Захищений ключ генерується за допомогою таких алгоритмів, як PBKDF2 (функція виведення ключів на основі пароля 2) або апаратних генераторів випадкових чисел. Цей ключ має бути надійно спільним або переданий через мережу для подальшого використання;

– шифрування – оригінальне повідомлення (відкритий текст) перетворюється на нечитабельний текст (шифротекст), а відкритий текст обробляється блоками або потоками за допомогою алгоритму шифрування та секретного ключа;

– передача тексту шифру – зашифроване повідомлення (зашифрований текст) надсилається через мережу. Навіть у разі перехоплення він залишається нечитабельним для зловмисника, якщо він не має доступу до спільного секретного ключа та алгоритму, який використовується для шифрування;

– розшифровка – одержувач використовує той самий секретний ключ і алгоритм зворотного шифрування для перетворення тексту шифру назад у вихідне повідомлення (звичайний текст).

Симетричне шифрування має різні види алгоритмів шифрування, залежно від їх особливостей, сильних сторін та ефективності.

У таблиці 2.1 описані найпопулярніші алгоритми симетричного шифрування.

Перевага симетричних алгоритмів – швидкість роботи, що робить їх ефективними для шифрування великих обсягів даних (наприклад, результатів тестів).

Таблиця 2.1 – Типи симетричних алгоритмів шифрування

Алгоритм	Опис	Ключові особливості
AES (Advanced Encryption Standard)	Широко прийнятий стандарт симетричного шифрування, схвалений NIST для національного та промислового використання.	Доступні з розмірами 128-бітних, 192-бітних і 256-бітних ключів. Висока продуктивність і безпека.
DES (Data Encryption Standard)	Раніше популярний, а тепер застарів через вразливість до brute-force attacks.	Розмір ключа 56 біт. На зміну їм приходять більш безпечні альтернативи, такі як AES і 3DES.
Triple DES (3DES)	Покращена версія DES, застосовується DES тричі до кожного блоку даних.	Сильніший, ніж DES, але повільніший і менш ефективний, ніж AES.
Blowfish	Блочний шифр розроблений як альтернатива DES, відомий своєю швидкістю та ефективністю в багатьох додатках.	Розмір блоку 64 біти. Гнучка довжина ключа до 448 біт.
Twofish	Наступник Blowfish, забезпечує надійну безпеку та гнучкість.	Розмір блоку 128 біт. Довжина ключа до 256 біт.

Асиметричні методи використовують пару ключів: публічний (для шифрування) та приватний (для розшифрування) [10].

Принцип роботи асиметричного шифрування:

– генерація пари ключів – першим кроком в асиметричному шифруванні є генерація пари ключів: відкритого ключа та приватного ключа (відкритий ключ надається відкрито, а приватний ключ тримається в таємниці);

– шифрування – відправник використовує відкритий ключ одержувача для шифрування повідомлення. Відкритий ключ використовується тому, що він знаходиться у вільному доступі і може бути використаний для перетворення повідомлення в нечитабельний формат (текст шифру);

– відправка – зашифроване повідомлення (текст шифру) відправляється по мережі одержувачу. Навіть якщо його кимось перехопить, він залишається нечитабельним без відповідного закритого ключа;

– розшифровка – отримавши зашифроване повідомлення, одержувач використовує свій приватний ключ для його розшифровки. Приватний ключ зберігається в безпеці і ніколи не розголошується, гарантуючи, що лише призначений одержувач може розшифрувати та прочитати повідомлення;

– верифікація – у деяких випадках відправник також може підписати повідомлення за допомогою свого приватного ключа, щоб переконатися в його автентичності. Одержувач може перевірити підпис за допомогою відкритого ключа відправника, підтвердивши, що повідомлення не було підроблено та дійсно було надіслано призначеним відправником;

– результат – після розшифровки повідомлення повертається до початкового вигляду (звичайний текст), і одержувач може його прочитати. Цей метод забезпечує як конфіденційність повідомлення (за рахунок використання відкритого ключа одержувача), так і цілісність і автентичність (за допомогою цифрових підписів з використанням закритого ключа відправника).

Основні алгоритми:

– RSA – найпоширеніший метод для захисту вебз’єднань і цифрових підписів;

– ECC (Elliptic Curve Cryptography) – забезпечує такий самий рівень безпеки, але з меншими ключами, що підвищує продуктивність системи.

Ці методи часто використовуються в системах дистанційного тестування для забезпечення безпечного обміну ключами та автентифікації користувачів.

Хешування – це перетворення вхідного масиву даних довільної довжини у вихідний бітовий рядок фіксованої довжини.

Хешування застосовується для побудови асоціативних масивів, пошуку дублікатів в серіях наборів даних, побудови унікальних ідентифікаторів для наборів даних, контрольного підсумовування з метою виявлення випадкових або навмисних помилок при зберіганні або передачі [11].

Існують алгоритми хешування з різними властивостями (розрядність, обчислювальна складність, криптостійкість тощо). Вибір тієї чи іншої хеш-функції визначається специфікою розв'язуваної задачі.

Основні алгоритми:

– MD5 (Message Digest 5) – 128-бітний алгоритм хешування, призначений для створення «відбитків» або дайджестів повідомлення довільної довжини і подальшої перевірки їх автентичності;

– SHA-1 – реалізує хеш-функцію, побудовану на основі функції стиснення. Входами функції стиснення є блок повідомлення довжиною 512 біт і вихід попереднього блоку повідомлення.;

– SHA-2 (SHA-256 та SHA-512): SHA-2 – це надійний, перевірений часом стандарт хешування, який залишається основним у більшості сучасних систем безпеки. Хоча вже існує новіша версія SHA-3, SHA-2 досі залишається золотим стандартом» криптографічних хеш-функцій завдяки своїй стабільності, ефективності й широкій підтримці.її чинний стандарт у безпеці та блокчейні. Вони генерують 256- або 512-бітні хеші та не були порушені відповідними зіткненнями досі.

– SHA-3 – це еволюційний крок у розвитку хешування, який забезпечує підвищений рівень безпеки завдяки новій конструкції та гнучкості. Попри те, що SHA-2 залишається основним стандартом у більшості систем, SHA-3 вважається алгоритмом майбутнього, що підходить для нових протоколів, блокчейн-рішень і високо захищених середовищ. SHA-3, на відміну від попередників (SHA-1, SHA-2), побудований на спонж-конструкції (sponge construction). Це означає, що алгоритм поглинає (absorb) вхідні дані в стан фіксованого розміру, а потім видає (squeeze) вихід будь-якої довжини. Тобто SHA-3 може генерувати хеш довільної довжини, що робить його більш гнучким у порівнянні з SHA-2.

Цифровий підпис – це криптографічний механізм, який дозволяє перевірити справжність джерела даних і запобігти їх підробці. У системах тестування цифрові підписи можуть застосовуватись для підтвердження достовірності результатів або відповідей користувачів.

2.3 Механізми автентифікації та авторизації користувачів

Автентифікація та авторизація – це ключові механізми контролю доступу до ресурсів системи.

Автентифікація – це процес перевірки особи користувача або системи, щоб переконатися, що вони є тими, за кого себе видають. Зазвичай це включає облікові дані, такі як імена користувачів, паролі, одноразові паролі (OTP) або біометричні методи, такі як відбитки пальців і розпізнавання обличчя. Перевіряючи ці облікові дані, автентифікація запобігає несанкціонованому доступу та допомагає захистити конфіденційні системи та дані від порушень безпеки [12].

Авторизація – це процес визначення та надання прав доступу аутентифікованому користувачеві або системі. Він визначає, до яких ресурсів користувач має доступ і які дії йому дозволено виконувати. Авторизація завжди відбувається після аутентифікації і гарантує, що тільки дозволені користувачі можуть виконувати конкретні завдання, тим самим забезпечуючи дотримання політик безпеки і захищаючи конфіденційні ресурси. Найпоширенішим способом є перевірка логіна та пароля.

Проте для підвищення безпеки паролі мають зберігатися у вигляді хешів із додаванням солі (salt), що унеможливлює їх відновлення навіть у разі витоку бази даних.

Двофакторна автентифікація (2FA) – це спосіб перевірки особи користувача шляхом запиту рівно двох доказів, таких як пароль до облікового запису в Інтернеті (перший фактор) і одноразовий пароль від програми-автентифікатора (другий фактор) [13].

У системах тестування це дозволяє запобігти несанкціонованому доступу навіть у разі викрадення пароля. Двофакторна автентифікація допомагає посилити безпеку облікового запису, вимагаючи другого фактора. Мало того, що хакерам потрібно викрасти два облікові дані, щоб зламати систему, але й другий фактор часто є чимось, що важко зламати. Поширені другі фактори включають

відбитки пальців і біометрію, фізичні ключі безпеки та коди доступу, термін дії яких закінчується.

Для сучасних вебдодатків часто використовується OAuth 2.0 – протокол авторизації, який дозволяє входити через облікові записи Google, Facebook тощо.

JWT (JSON Web Token) використовується для передачі безпечної інформації між клієнтом і сервером у зашифрованому вигляді, що робить його ефективним засобом авторизації у SPA-додатках.

Реалізація OAuth за допомогою JWT (JSON Web Tokens) [14]:

– клієнт: програма, яка намагається отримати доступ до даних користувача. Це може бути веб або мобільний додаток;

– власник ресурсу: користувач, який володіє даними, до яких клієнт намагається отримати доступ. Власник ресурсу надає клієнту дозвіл на доступ до даних;

– сервер авторизації: сервер авторизації, який аутентифікує власника ресурсу та видає токени доступу після отримання належної авторизації;

– сервер ресурсів: сервер ресурсів, на якому розміщуються захищені ресурси. Він перевіряє токен доступу та обслуговує запитуваний ресурс, якщо токен дійсний;

– грант авторизації: Грант авторизації – це облікові дані, що представляють авторизацію власника ресурсу (пароль, облікові дані клієнта, код авторизації);

– токен: маркер доступу представляє дозвіл на авторизацію для клієнта;

– JWT: JWT визначається як вебтокен JSON, який може бути безпечним для URL-адрес і представляє вимоги, які мають бути передані між двома сторонами. JWT можна використовувати як токен доступу в OAuth 2.0;

– твердження JWT – фрагменти інформації, які передаються в JWT. Це можуть бути такі речі, як ідентифікація користувача, ролі користувачів, час закінчення терміну дії тощо;

– область застосування: визначаються дозволи, які запитує клієнт. Наприклад, доступ лише для читання або повний доступ.

Як тільки користувач реєструється, дані про користувача та пароль зберігаються в базі даних MongoDB. Паролі можуть зберігатися закодовані форми, і в той же час вони можуть генерувати JWT. Отже, користувачі можуть увійти в систему та отримати доступ до бази даних, але вони повинні мати облікові дані користувача разом із токеном JWT. Після того, як облікові дані та токени зіставляються, вони можуть увійти до бази даних.

2.4 Методи захисту вебдодатків від типових атак

Вебсистеми дистанційного тестування є потенційною мішенню для численних атак. Найпоширеніші з них – SQL-ін’єкції, XSS, CSRF, brute-force та інші.

SQL-ін’єкції дозволяють зловмиснику втручатися у запити до бази даних. SQL-ін'єкція, часто відома як SQLI, є типовим вектором атаки, який використовує шкідливий SQL-код для маніпулювання базами даних Backend з метою отримання інформації, яка не була призначена для показу. Ця інформація може містити конфіденційні корпоративні дані, списки користувачів або конфіденційну інформацію про споживачів.

Існують прості методи запобігання вразливостей SQL-ін'єкцій, і їх можна використовувати практично з будь-яким видом мови програмування і будь-яким типом баз даних. Хоча бази даних XML можуть мати схожі проблеми (наприклад, XPath та впровадження XQuery), ці методи також можуть бути використані для їх захисту.

Захист включає [15]:

– використування розширенної дезінфекції даних: усі введені користувачами дані повинні бути відфільтровані вебсайтами. В ідеалі дані користувачів повинні бути відфільтровані за контекстом;

– використання брандмауер вебдодатків: Mod Security, безкоштовний модуль з відкритим вихідним кодом для вебсерверів Apache, Microsoft IIS і

Nginx, є яскравим прикладом. Mod Security має складний і постійно мінливий набір правил для фільтрації потенційно небезпечних онлайн-запитів;

– регулярно оновлювати програмне забезпечення: оскільки вразливості SQL-ін'єкцій часто виявляються в комерційному програмному забезпеченні, дуже важливо йти в ногу з оновленням;

– обмеження прав користувачів бази даних.

XSS (Cross-Site Scripting) – це вразливість веббезпеки, яка дозволяє зловмиснику скомпрометувати взаємодію користувачів із вразливою програмою. Це дозволяє зловмиснику обійти ту саму політику походження, яка призначена для відокремлення різних вебсайтів один від одного. Уразливості міжсайтового скриптингу зазвичай дозволяють зловмиснику маскуватися під користувача-жертву, виконувати будь-які дії, які користувач здатний виконати, і отримувати доступ до будь-яких даних користувача. Якщо користувач-жертва має привілейований доступ у програмі, то зловмисник може отримати повний контроль над усіма функціями та даними програми.

Основні методи забезпечення захисту від XSS [16]:

– фільтрування даних, вхідні дані користувача, вони фільтруються якомога суворіше на основі очікуваних або дійсних вхідних даних;

– кодування даних, керовані користувачем дані виводяться у відповідях HTTP, кодуються вихідні дані, щоб запобігти їх інтерпретації як активного вмісту. Залежно від контексту виводу, це може вимагати застосування комбінацій кодування HTML, URL, JavaScript і CSS;

– використання відповідних заголовків відповідей, запобігти XSS у відповідях HTTP, які не призначені для зберігання HTML або JavaScript, використовуються заголовки and, щоб переконатися, що браузери інтерпретують відповіді правильно;

– політика безпеки контенту, використовується політика безпеки контенту (CSP), щоб зменшити серйозність будь-яких XSS-вразливостей, які все ще виникають.

CSRF (Cross-Site Request Forgery) змушує користувача виконати небажану дію. Під час атаки зловмисник обманом змушує користувача або браузер зробити HTTP-запит до цільового сайту зі шкідливого сайту. Запит включає облікові дані користувача і змушує сервер виконати шкідливу дію [17].

Захист реалізується за допомогою токенів (CSRF Token), які генеруються для кожної сесії. У цьому захисті, коли сервер обслуговує сторінку, він вбудовує в сторінку непередбачуване значення, яке називається CSRF-токеном. Потім, коли законна сторінка надсилає запит на зміну стану на сервер, вона включає CSRF-токен у HTTP-запит. Потім сервер може перевірити значення токена і виконати запит тільки в тому випадку, якщо воно збігається. Навіть якщо зловмисник виявить токен після його використання, запит не може бути повторно відтворений, якщо токен змінюється щоразу.

2.5 Системи моніторингу та виявлення вторгнень

Зі зростанням кількості кібератак на вебзастосунки, зокрема системи дистанційного тестування та опитувань, особливого значення набувають системи моніторингу та виявлення вторгнень (Intrusion Detection Systems, IDS), а також системи запобігання вторгненням (Intrusion Prevention Systems, IPS). Їх основне призначення полягає у своєчасному виявленні підозрілої або шкідливої активності та реагуванні на потенційні загрози безпеці.

Системи виявлення вторгнень (IDS/IPS) аналізують мережевий трафік і виявляють підозрілу активність.

Для вебдодатків найчастіше застосовуються:

– Snort – система аналізу трафіку на основі сигнатур атак;

– Suricata – інструмент із підтримкою протоколів TLS і HTTP/2;

– Fail2Ban – система блокування IP-адрес після багаторазових невдалих спроб входу.

Також важливо використовувати системи журналювання (логування), які фіксують усі події в системі – це дозволяє проводити аудит безпеки й виявляти спроби несанкціонованого доступу.

3 АНАЛІЗ СЕРВІСІВ ДЛЯ ЗАБЕЗПЕЧЕННЯ ЗАХИСТУ В СИСТЕМАХ ТЕСТУВАНЬ ТА ОПИТУВАНЬ

3.1 Принципи захисту в існуючих системах тестувань та опитувань

Криптографічні методи захисту є основним інструментом забезпечення конфіденційності, цілісності та автентичності даних у системах дистанційного тестування та опитувань. Їх реалізація охоплює всі етапи життєвого циклу інформації: від моменту введення користувачем до зберігання та обробки результатів.

3.1.1 Захист даних під час передавання

Принцип реалізації заключається в передаванні даних між клієнтською частиною (браузером користувача) та сервером через захищений канал зв’язку з використанням криптографічних протоколів.

Технічна реалізація:

– використання протоколу HTTPS;

– застосування TLS 1.2 / TLS 1.3;

– асиметричне шифрування (RSA або ECDHE) для обміну ключами;

– симетричне шифрування (AES-128 або AES-256) для передавання даних.

Практичний приклад:

– Google Forms: усі опитування доступні виключно через HTTPS, передавання відповідей користувачів захищене TLS, унеможливлюється перехоплення результатів опитування;

– Moodle: використовує HTTPS та TLS для передавання тестових відповідей, забезпечує захист під час складання іспитів у режимі онлайн.

3.1.2 Захист даних під час зберігання

Принцип реалізації заключається в тому, що дані, які зберігаються у базі даних (результати тестів, персональна інформація), шифруються з використанням стійких криптографічних алгоритмів.

Технічна реалізація:

– симетричне шифрування AES-256;

– зберігання ключів шифрування окремо від бази даних;

– застосування серверних модулів безпеки (HSM) або конфігурацій середовища.

Практичний приклад:

– Blackboard Learn: результати тестування студентів зберігаються у зашифрованому вигляді, використовується шифрування даних на рівні бази та серверного сховища;

– Canvas LMS: персональні дані та оцінки користувачів зберігаються з використанням криптографічних методів, застосовується шифрування резервних копій.

3.1.3 Криптографічний захист облікових даних

Паролі користувачів не зберігаються у відкритому вигляді, а перетворюються за допомогою криптографічних хеш-функцій.

Технічна реалізація:

– хешування з використанням bcrypt, Argon2, PBKDF2;

– застосування випадкової солі (salt);

– захист від атак словником і brute-force.

Практичний приклад:

– Moodle: використовує хешування паролів, підтримує сучасні алгоритми захисту облікових даних;

– Testportal: зберігає паролі виключно у вигляді криптографічних хешів, реалізує обмеження кількості спроб входу.

3.1.4 Криптографічні механізми автентифікації та сесій

Для підтвердження особи користувача та керування сесіями використовуються токени, захищені криптографічними алгоритмами.

Технічна реалізація:

– застосування JWT (JSON Web Token);

– цифровий підпис токена (HMAC або RSA);

– обмежений термін дії токена.

Практичний приклад:

– Microsoft Forms: використовує токенну автентифікацію через Microsoft Account, застосовується цифровий підпис токенів доступу.

3.1.5 Забезпечення цілісності результатів тестування

Для захисту результатів тестування від підміни використовуються криптографічні механізми контролю цілісності.

Технічна реалізація:

– хешування результатів тестів (SHA-256);

– цифрові підписи;

– часові мітки виконання тестів.

Практичний приклад:

– ProctorU: використовує криптографічні механізми для підтвердження цілісності екзаменаційних даних, застосовує цифрові підписи та захищені журнали подій.

Бюджет на реалізацію криптографічних методів захисту значною мірою залежить від масштабу системи, кількості користувачів, вимог до рівня безпеки та обраних технологічних рішень. У сучасних вебсистемах значна частина криптографічних механізмів може бути реалізована з використанням відкритих стандартів і бібліотек, що суттєво знижує загальні витрати.

3.2 Аналіз існуючих сервісів для забезпечення захисту системи тестувань та опитувань

3.2.1 Криптографічні бібліотеки

Криптографічна бібліотека – це програмний набір інструментів, який реалізує алгоритми шифрування, хешування, електронного підпису та керування ключами для забезпечення конфіденційності, цілісності й автентичності даних.

Класифікація криптографічних бібліотек для вебсерверів та бекенду:

– OpenSSL: використовує мови: C, C++, PHP, Python, Node.js (через обгортки); алгоритми: AES, RSA, ECC, SHA-2, SHA-3, TLS; призначений для: HTTPS, SSL/TLS, шифрування даних; використовується у: Apache, Nginx, OpenVPN, Moodle; безкоштовний;

– libsodium: використовує мови: C, PHP, Python, JavaScript, Rust; алгоритми: XChaCha20-Poly1305, Ed25519, BLAKE2; призначення: сучасне симетричне та асиметричне шифрування; простий, безпечні налаштування за замовчуванням; безкоштовний;

– Bouncy Castle: використовує мови: Java, C#; алгоритми: AES, RSA, ECC, SHA-3, PGP; призначення: корпоративні Java-додатки; використовується у Java-системах дистанційного навчання; безкоштовний;

– Crypto++: використовує мову C++; алгоритми: AES, RSA, ECC, Twofish, SHA-2; призначення: високопродуктивні системи; безкоштовний.

Класифікація криптографічних бібліотек для для вебклієнта (JavaScript):

– Web Crypto API: середовище: браузер; алгоритми: AES-GCM, RSA-OAEP, ECDSA, SHA-256; призначення: шифрування даних на стороні клієнта; нативна підтримка браузерами; безкоштовний;

– CryptoJS: мова: JavaScript; алгоритми: AES, DES, SHA-1, SHA-256; призначення: базове клієнтське шифрування; не рекомендується для високих вимог безпеки; безкоштовний.

Класифікація криптографічних бібліотек для керування паролями:

– bcrypt: призначення: хешування паролів; стійкий до brute-force

– Argon2: призначення: сучасний алгоритм хешування паролів; рекомендований стандарт (OWASP);

– PBKDF2: призначення: деривація ключів із паролів; стандарт: NIST.

Класифікація криптографічних бібліотек для керування ключами та секретами:

– HashiCorp Vault: зберігання ключів шифрування; підтримка HSM, аудит доступу; план Community – безкоштовно, Enterprise – платно.

– AWS KMS / Azure Key Vault / Google Cloud KMS: керування криптографічними ключами у хмарі, інтеграція з WAF та IAM, коштує від ~$1–3 за ключ/місяць.

В таблиці 3.1 наведено узагальнене порівняння криптографічних бібліотек.

Таблиця 3.1 – Порівняння криптографічних бібліотек

Бібліотека	Середовище	Алгоритми	Призначення	Вартість
OpenSSL	Сервер	AES, RSA, TLS	HTTPS, шифрування	0
libsodium	Сервер	XChaCha20, ECC	Захищені дані	0
Bouncy Castle	Java	AES, ECC	Корпоративні системи	0
Crypto++	C++	AES, RSA	Висока продуктивність	0
Web Crypto API	Браузер	AES-GCM, RSA	Клієнтське шифрування	0
Argon2	Сервер	Password hashing	Захист паролів	0
HashiCorp Vault	Сервер	KMS	Управління ключами	0 / платно

Аналіз даних, наведених у таблиці 3.1, показує, що сучасні криптографічні бібліотеки забезпечують широкий набір засобів для реалізації захисту інформації на різних рівнях вебсистеми. Більшість розглянутих бібліотек є безкоштовними та мають відкритий вихідний код, що робить їх доступними для використання у навчальних і комерційних системах дистанційного тестування та опитувань.

Зокрема, бібліотеки OpenSSL та libsodium орієнтовані на серверний рівень і забезпечують шифрування даних та захищені канали зв’язку, що є критично важливим для передавання персональної інформації користувачів. Bouncy Castle та Crypto++ доцільно використовувати у високопродуктивних або корпоративних середовищах, де необхідна підтримка широкого набору криптографічних алгоритмів.

Web Crypto API дозволяє реалізувати клієнтське шифрування безпосередньо у браузері, що підвищує рівень безпеки даних ще на етапі їх формування. Окрему увагу заслуговує алгоритм Argon2, який забезпечує надійний захист паролів і відповідає сучасним рекомендаціям щодо хешування облікових даних. HashiCorp Vault розширює можливості системи за рахунок централізованого управління криптографічними ключами.

Таким чином, результати порівняння свідчать про доцільність комплексного використання кількох криптографічних бібліотек для забезпечення конфіденційності, цілісності та доступності даних у системах дистанційного тестування та опитувань.

3.2.2 SSL/TLS-сертифікати

SSL/TLS-сертифікат – це цифровий сертифікат, який забезпечує захищений канал передавання даних між клієнтом (браузером користувача) та вебсервером шляхом застосування криптографічних алгоритмів.

SSL/TLS-сертифікати забезпечують виконання таких функцій:

– конфіденційність – шифрування даних, що передаються мережею;

– цілісність – захист від підміни або спотворення інформації;

– автентифікація – підтвердження достовірності сервера;

– захист від MITM-атак (Man-in-the-Middle).

Види SSL/TLS-сертифікатів за рівнем перевірки:

Таблиця 3.2 – Порівняння SSL/TLS-сертифікатів за рівнем перевірки

Вид сертифіката	Захист	Призначення	Рівень перевірки
DV (Domain Validation)	Базовий	Особисті / стартапи	Перевірка домену
OV (Organization Validation)	Середній	Компанії	Перевірка організації
EV (Extended Validation)	Найвищий	Комерційні / критичні сервіси	Повна перевірка компанії
Wildcard	DV/OV/EV	Піддомени (*.domain.com)	Захист усіх піддоменів
Multi-Domain (SAN)	DV/OV/EV	Багато доменів	Для кількох доменів одночасно

Основні параметри SSL/TLS-сертифікатів наведені в таблиці 3.3.

Таблиця 3.3 – Параметри SSL/TLS-сертифікатів

Параметр	Призначення
Ціна	Від безкоштовно до декількох тисяч грн/рік
Рівень довіри	DV → OV → EV
Перевірка компанії	Є лише для OV та EV
Підтримка піддоменів	Wildcard
Кількість доменів	SAN/Multi-Domain
Установка на сервер	Стандартна для всіх

Основні сервіси, що надають SSL/TLS-сертифікати, та їх характеристики:

– Let’s Encrypt: безкоштовний, використовує ACME-протокол, підтримує DV, широко поширений у вебсервісах та хостингах, ідеальний для освітніх, тестових і невеликих проєктів. Переваги: безкоштовний, автоматичне

оновлення, широка підтримка (cPanel, Plesk, Cloudflare, Certbot). Недоліки: лише базовий рівень (DV), немає перевірки організації;

– Cloudflare SSL: безкоштовний план (DV / Universal SSL), план Pro/Business – розширені функції безпеки, автоматична конфігурація. Переваги: просте налаштування, автоматичне HTTPS, Web Application Firewall (платний). Недоліки: безкоштовний SSL лише DV, деякі функції – платні. Безкоштовний лише базовий SSL, план Pro: ~25–35 USD/місяць (додаткові захисні функції), план Business: ~250 USD/місяць;

– Comodo (Sectigo): підтримка DV, OV, EV, Wildcard, SAN; підтримка у більшості панелей керування. Переваги: повний спектр сертифікатів, прийнятний бюджет для OV/EV. Недоліки: платний, складніше налаштування для початківців. Орієнтовна ціна / рік: DV ~ 10 – 50 USD/міс; OV ~ 20 – 80 USD/міс; EV ~ 27 – 60 USD/міс; Wildcard ~ 50 – 80 USD/міс;

– DigiCert: преміум-сертифікати (EV/OV), професійна підтримка, часто використовується в корпоративних рішеннях. Переваги: високий рівень довіри, автоматичне сканування безпеки. Недоліки: дорожчі за інші. Орієнтовна ціна / рік: OV ~ 27 – 112 USD/міс, EV ~ 40 – 160 USD/міс;

– GeoTrust: DV / EV сертифікати, хороший баланс ціни-якість. Орієнтовна ціна / рік: DV ~ 14 – 57 USD/міс, EV ~ 40 USD/міс;

Нижче наведена узагальнена таблиця основних постачальників SSL/TLS-сертифікатів (табл. 3.5).

Таблиця 3.4 – Порівняння сервісів для постачання SSL/TLS-сертифікатів

Сервіс	Типи сертифікатів	Рівень перевірки	Бюджет / міс	Підтримка Wildcard
1	2	3	4	5
Let’s Encrypt	DV	перевірка по домену	0 USD	тільки через платні плани
Cloudflare	DV	перевірка по домену	~ 0 – 250 USD	тільки через платні плани

Продовження таблиці 3.4

Сервіс	Типи сертифікатів	Рівень перевірки	Бюджет / міс	Підтримка Wildcard
1	2	3	4	5
Comodo (Sectigo)	DV, OV, EV	перевірка по домену, організації	~ 10 – 80 USD	тільки через платні плани
DigiCert	OV, EV	перевірка організації	~ 27 – 160 USD	тільки через платні плани
GeoTrust	DV, EV	перевірка організації	~ 14 – 57 USD	тільки через платні плани

Ринок сервісів для отримання SSL-сертифікатів пропонує рішення для будь-яких потреб – від базового захисту особистих сайтів до комплексних корпоративних рішень.

3.2.3 Сервіси для надання серверних ресурсів (хостинги, VPS, cloud), їх порівняння за характеристиками

Сервер – це комп’ютер/сервіс, що постійно працює і надає ресурси для вебсервісів:

– розміщення коду;

– база даних;

– SSL/TLS;

– обробка запитів користувачів.

Сучасні рішення поділяють на:

– VPS (Virtual Private Server) – виділений віртуальний сервер;

– Cloud (хмарні сервіси) – масштабовані ресурси, оплата за використання;

– хостинг – прості рішення для невеликих сайтів.

Критерії вибору сервера наведені в таблиці 3.5.

Таблиця 3.5 – Критерії вибору сервера

Критерій	Що важливо
Продуктивність	CPU, RAM, SSD
Масштабованість	горизонтальне/вертикальне розширення
Надійність	SLA, резервування
Безпека	брандмауер, DDoS-захист
Ціна	щомісячна / погодинна
Локація	сервери ближче до користувачів

Основні сервіси для надання серверних ресурсів:

– DigitalOcean: легкий старт для проєктів, проста панель управління, швидке розгортання серверів. Переваги: зручність, масштабованість. Недоліки: додаткові платні послуги (резервне копіювання, DDoS Protection) [18].

В таблиці 3.6 розглянуті плани VPS «Droplet».

Таблиця 3.6 – Плани VPS («Droplet»)

План	CPU	RAM	SSD	Орієнтовна ціна/міс
Basic S	1 vCPU	1 GB	25 GB	~5 USD
Standard	2 vCPU	2 GB	50 GB	~10 USD
Optimized	4 vCPU	8 GB	160 GB	~48 USD

– Amazon Web Services (AWS): потужна хмарна платформа, величезний набір сервісів (EC2, RDS, S3 тощо). Переваги: потужність, екосистема. Недоліки: складніша конфігурація, непередбачувана ціна при навантаженні [19].

В таблиці 3.7 наведено типи інстансів для екземпляру EC2, які надає AWS.

Таблиця 3.7 – Екземпляр EC2 (Elastic Compute Cloud)

Тип інстансу	CPU	RAM	Орієнтована ціна/місяць
t3.micro	2 vCPU	1 GB	~Free
t3.small	2 vCPU	2 GB	~20–25 USD
t3.medium	2 vCPU	4 GB	~40–50 USD

– Google Cloud Platform (GCP): Хороший варіант для стартапів, Free Tier + кредит для нових акаунтів [20].

В таблиці 3.8 наведено типи інстансів для екземпляру Compute Engine від Google Cloud Platform.

Таблиця 3.8 – Екземпляр Compute Engine

Тип інстансу	CPU	RAM	Орієнтовна ціна
e2-micro (Free)	2 vCPU	1 GB	безкоштовно (обмеження)
e2-small	2 vCPU	2 GB	~25–30 USD
e2-medium	2 vCPU	4 GB	~50–60 USD

– Microsoft Azure: інтеграція з Active Directory, добре підходить корпоративним проєктам. Переваги: корпоративні сервіси. Недоліки: дорожче за інші [21].

В таблиці 3.9 наведено екземпляр Azure Virtual Machines від Microsoft Azure.

Таблиця 3.9 – Екземпляр Azure Virtual Machines

Тип	CPU / RAM	Орієнтовна ціна/міс
B1s	1 vCPU / 1 GB	~15–20 USD
B2s	2 vCPU / 4 GB	~50–60 USD

– Hetzner Cloud: низькі ціни, європейський хостинг. Переваги: дуже вигідно за співвідношенням. Недоліки: підтримка інколи повільніша [22].

В таблиці 3.10 наведено плани для екземпляру компанії Hetzner.

Таблиця 3.10 – Екземпляр компанії Hetzner

План	CPU	RAM	SSD	Ціна/міс
CX11	1 vCPU	2 GB	20 GB	~4 USD
CX21	2 vCPU	4 GB	40 GB	~8 USD
CX31	2 vCPU	8 GB	80 GB	~16 USD

В таблиці 3.11 проведено порівняння сервісів за бюджетом та призначенням.

Таблиця 3.11 – Порівняння сервісів

Сервіс	Масштабованість	Легкість налаштування	Базова ціна	Підходить для
DigitalOcean	середня	висока	~5–10 USD	SMB, освітніх систем
AWS	дуже висока	середня	~0–50+ USD	корпоративних систем
GCP	дуже висока	середня	~0–60+ USD	стартапів
Azure	висока	середня	~15–60+ USD	корпоративних систем
Hetzner	середня	середня	~4–16+ USD	малий/середній бізнес

Порівняльний аналіз сервісів хмарної інфраструктури, наведених у таблиці 3.11, свідчить про наявність широкого спектра рішень, що відрізняються рівнем масштабованості, складністю налаштування та вартістю використання. Це дозволяє обрати оптимальний сервіс залежно від вимог до функціональності, рівня безпеки та фінансових обмежень.

Сервіс DigitalOcean характеризується простотою налаштування та помірною вартістю, що робить його доцільним для використання у навчальних проєктах та системах для малого й середнього бізнесу. Hetzner також є економічно вигідним варіантом для невеликих і середніх систем, забезпечуючи достатній рівень масштабованості за відносно низької вартості.

Хмарні платформи AWS, Google Cloud Platform та Microsoft Azure забезпечують дуже високий рівень масштабованості та надійності, а також широкий набір вбудованих сервісів безпеки. Такі платформи є найбільш доцільними для корпоративних систем та стартапів, орієнтованих на швидке зростання, однак потребують вищих фінансових витрат і більшої кваліфікації для адміністрування.

Таким чином, вибір хмарного сервісу для систем дистанційного тестування та опитувань повинен базуватися на балансі між вимогами до масштабованості, рівнем захисту даних і доступним бюджетом, що підтверджує доцільність використання різних інфраструктурних рішень залежно від масштабу та призначення системи.

3.3 Захист від типових вебзагроз

3.3.1 Захист від SQL-ін’єкцій

Основні способи захисту від SQL-ін’єкцій:

– параметризовані запити / ORM (перший і найефективніший захід). Фреймворки: Laravel Eloquent, Django ORM, Hibernate – безкоштовно;

– WAF / правилова фільтрація;

– сканування коду / DAST – виявлення проблем до релізу.

Сервіси для захисту від SQL-ін’єкцій:

– Cloudflare WAF – гнучкі правила, rate limiting; має безкоштовний тариф і платні рівні (Pro/Business) для розширених правил;

– AWS WAF – підтримує веб-ACL, rate-based правила; добре інтегрується з AWS-інфраструктурою;

– Imperva WAF – enterprise WAF з керованими правилами (потужний, дорожчий варіант);

– Akamai Kona / App & API Protector – WAF/WAAP на CDN-ребрі для промислових навантажень;

– ModSecurity (open-source) – вбудовується в Apache/Nginx/IIS; чудовий для on-prem або VPS рішень з додатковими правилами (OWASP CRS).

Орієнтовний бюджет (рік):

– Let’s Encrypt + ModSecurity на VPS: 0–200 USD / рік (сервер) – мінімум;

– AWS WAF: залежить від правил/трафіку; орієнтовно від 10–200 USD/міс у типових сценаріях (варіюється);

– Imperva / Akamai (enterprise): від кількох тисяч доларів на рік (контрактна ціна);

3.3.2 Захист від міжсайтового скриптингу (XSS)

Основні способи від міжсайтового скриптингу (XSS):

– екранування/самітапізація на сервері та клієнті (output encoding);

– санітайзинг HTML на клієнті (наприклад, DOMPurify);

– Content Security Policy (CSP) – обмежує джерела виконуваного JS/CSS;

– WAF як додатковий шар.

Сервіси для захисту від міжсайтового скриптингу (XSS):

– DOMPurify – широко рекомендована JS-бібліотека для очищення HTML перед вставкою у DOM. (клієнтський бранд);

– CSP (Content Security Policy) – конфігурація заголовків на сервері (можна через CDN/WAF задавати політику). Cloudflare, Akamai і т.д. підтримують управління заголовками;

– WAF (Cloudflare, AWS WAF, Imperva, ModSecurity) – блокують скрипти/патерни XSS на рівні запитів.

Орієнтовний бюджет

– DOMPurify – 0 USD (OSS);

– CSP – реалізація на сервері безкоштовна; управління через Cloudflare/WAF – див. їхні тарифи (Cloudflare Pro/Business);

– Enterprise WAF (Imperva/Akamai) – контрактні ціни, тисячі $/рік.

3.3.3 Захист від CSRF (Cross-Site Request Forgery)

Основні способи захисту від CSRF (Cross-Site Request Forgery):

– CSRF-токени (synchronizer token) у формах; SameSite cookies; перевірка Origin/Referer;

– багато вебфреймворків мають вбудований механізм CSRF (Django, Laravel, Ruby on Rails);

– для Java-середовищ – бібліотеки, наприклад OWASP CSRFGuard.

Сервіси для захисту від CSRF:

– вбудовані механізми фреймворків (Django, Laravel тощо) – безкоштовно;

– OWASP CSRFGuard – проєкт для JavaEE (реалізація synchronizer token pattern);

– WAF – другий шар перевірки, але не замінює токени; тимчасово блокує підозрілі запити.

Орієнтовний бюджет:

– CSRF-токени у фреймворку – 0 USD;

– OWASP CSRFGuard – 0 USD (OSS);

3.3.4 Захист від brute-force атак

Основні способи від brute-force атак:

– Rate limiting / rate-based rules (на вхідних маршрутах, login endpoints);

– блокування IP (Fail2Ban або edge-rate limiting);

– CAPTCHA на повторні спроби;

– сервіси автентифікації з вбудованим захистом (Auth0, Okta, Duo).

Сервіси для захисту від brute-force атак:

– Fail2Ban – локальний демон для блокування IP за логами (open-source);

– Cloudflare Rate Limiting / WAF rate rules – налаштовуються для login endpoints; має безкоштовні та платні опції;

– AWS WAF rate-based rules – приклади налаштування rate rules для логіну;

– Auth0 / Okta / Duo / Authy(Twilio) – мають вбудований захист від brute-force і anomaly detection (Auth0 Attack Protection). Auth0 має тарифні плани (від безкоштовного до платних з розширеним захистом).

Орієнтовний бюджет:

– Fail2Ban – 0 USD (встановлення на VPS);

– Cloudflare rate limiting – базова тарифа 0, Pro ~25 USD/міс, Business ~250 USD/міс (ціни залежать від правил/трафіку);

– Auth0 – є безкоштовний план; платні починаються ~35 USD/міс для Essentials і вище. Для великих проєктів від ~240 USD/міс згідно референсів.

3.3.5 Захист від перехоплення даних (MITM)

Основні способи захисту від перехоплення даних:

– TLS (HTTPS) – обов’язково (TLS 1.2/1.3);

– HSTS, правильна конфігурація серверів;

– використання перевірених сертифікаційних центрів або ACM/Let’s Encrypt;

Сервіси для захисту від MITM:

– Let’s Encrypt (Certbot) – безкоштовні TLS-сертифікати, автоматичне оновлення;

– AWS Certificate Manager (ACM) – безкоштовні сертифікати для ресурсів у AWS (ELB, CloudFront);

– Cloudflare SSL / Edge Certificates – просте включення HTTPS + опції (Universal SSL, full/strict mode).

Орієнтовний бюджет:

– Let’s Encrypt – 0 USD (сертифікати) + оплата сервера;

– ACM – 0 USD для сертифікатів у AWS (але інфраструктура AWS платна);

– Cloudflare – безкоштовний SSL у Free плані; платні плани дають додаткові гарантії/функції;

Узагальнена таблиця, щодо захисту від розглянутих атак наведена нижче в таблиці 3.12.

Таблиця 3.12 – Порівняння сервісів для захисту вебсистеми від різних атак

Сервіс / клас	SQL-ін'єкції	XSS	CSRF	Brute-force	MITM	Примітки
1	2	3	4	5	6	7
Cloudflare (WAF + CDN + Rate Limiting)	+ високий (через правила)	+ високий	частковий (WAF правила)	+ (через rate limiting, bot management)	+ (через edge TLS)	Просте для впровадження, має безкоштовний план
AWS WAF + AWS Shield + ALB	+ дуже високий	+ дуже високий	тільки через правила	+ (AWS WAF + Cognito/IP block)	+ (ACM TLS)	Глибока інтеграція з AWS, оплата за правила/запити
Imperva / Akamai (enterprise WAF)	+ дуже високий	+ дуже високий	+	+	+	Enterprise-рішення з повною підтримкою, дорожче
ModSecurity (open source WAF)	+ високий, з правилами CRS	+	частковий	–	–	Встановлюється на сервер, безкоштовний,
Sucuri (WAF + CDN)	+ високий	+ високий	частковий	+	+	Хороший для сайтів, простий у використанні
Fail2Ban / Rate limiting (NGINX)	–	–	–	(блокування IP)	–	Локальний захист від brute-force

Продовження таблиці 3.12

1	2	3	4	5	6	7
Auth0 / Okta / Duo (MFA)	–	–	–	(MFA суттєво знижує brute-force)	–	Identity/MFA – не WAF, але критично для захисту логінів
Let’s Encrypt / ACM / DigiCert	–	–	–	–	+	Захищає канал (MITM), не блокує атаки на аплікацію
OWASP ZAP / Burp / Acunetix (тестування)	виявляє	виявляє	виявляє	виявляє	виявляє	Інструменти для знаходження вразливостей – не блокують загрози
Snyk / SonarQube (SAST)	виявляє в коді	виявляє	виявляє	виявляє	виявляє	Інтегруються в CI для раннього виявлення вразливостей

Порівняльний аналіз сервісів захисту вебсистеми від типових атак, наведений у таблиці 3.12, свідчить про те, що жоден окремий інструмент не забезпечує повного захисту від усіх класів загроз. Найвищий рівень протидії SQL-ін’єкціям, XSS- та CSRF-атакам забезпечують комплексні рішення класу Web Application Firewall (WAF), зокрема Cloudflare, AWS WAF та enterprise-рішення Imperva і Akamai.

Сервіси Cloudflare та Sucuri є найбільш універсальними з точки зору співвідношення ефективності та простоти впровадження, оскільки поєднують WAF, CDN та механізми обмеження запитів, що дозволяє ефективно протидіяти brute-force атакам і частково захищати систему від міжсайтових атак. Водночас enterprise-рішення Imperva та Akamai забезпечують максимальний рівень

захисту, однак їх використання доцільне переважно у великих комерційних або корпоративних системах через високу вартість.

Засоби типу ModSecurity та Fail2Ban орієнтовані переважно на локальний серверний рівень і вимагають додаткового налаштування та комбінування з іншими інструментами для досягнення комплексного захисту. Сервіси керування ідентифікацією та багатофакторною автентифікацією (Auth0, Okta, Duo) не блокують вебатаки безпосередньо, проте істотно знижують ризик компрометації облікових записів користувачів.

Окрему групу становлять інструменти тестування безпеки та статичного аналізу коду (OWASP ZAP, Burp Suite, Snyk, SonarQube), які не запобігають атакам у режимі реального часу, але є важливими для виявлення вразливостей на етапах розробки та експлуатації системи.

Таким чином, результати аналізу підтверджують доцільність комплексного підходу до захисту систем дистанційного тестування та опитувань, який передбачає поєднання WAF, SSL/TLS, механізмів автентифікації, локальних засобів захисту та регулярного тестування безпеки.

Порівняння бюджету та функціоналу розглянутих сервісів наведено в таблиці 3.13.

Таблиця 3.13 – Порівняння бюджету та функціоналу для захисту вебсистеми

Сервіс	Ефективність захисту	Простота впровадження	Масштабованість	Орієнтовний бюджет
1	2	3	4	5
Cloudflare (Free, Pro, Business)	високий, в планах Pro та Business дуже високий	дуже просто	дуже висока	Free, $20–200+/міс
AWS WAF	дуже високий	середня, через налаштування	дуже висока	$5–$30+/міс + правила/запити

Продовження таблиці 3.13

1	2	3	4	5
Imperva / Akamai	дуже високий	середня	дуже висока	від $2000/рік (частіше значно більше)
ModSecurity (w/OWASP CRS)	високий з налаштуванням	складніше через налаштування	залежить від інфраструктури	0 (OSS) + витрати на адміністрування
Sucuri WAF	високий	дуже просто	висока	~$200–500/рік (плани варіюють)
Fail2Ban / NGINX rate limit	низький, середній для brute-force	просто	низька	0 (OSS)
Auth0 / Duo (MFA)	не блокує SQL/XSS, але захищає логіни	дуже просто	висока	від $0 до $3–8/користувач/міс
Burp Suite Pro (pentest)	– (інструмент тестування)	середня	–	~400 USD/рік
OWASP ZAP	– (тестування)	середня	–	0 (OSS)

Отже, вибір інструментів безпеки залежить від трьох ключових факторів: бюджету, технічної експертизи команди та масштабу проєкту. На ринку є рішення від повністю безкоштовних (Open Source) до дорогих Enterprise-систем.

3.4 Вибір оптимального стеку для захисту систем тестувань та опитувань

За результатом проведеного аналізу запропоновано 3 стеки – від «дуже бюджетного» до «корпоративного»:

– Стек А – Навчальний стек (мінімальний бюджет):

Склад: Let’s Encrypt, ORM (Laravel/Django), Cloudflare Free, Fail2Ban, VPS.

Даний стек забезпечує базовий рівень інформаційної безпеки, достатній для навчальних або демонстраційних вебсистем. Використання ORM на рівні

бекенду значно знижує ризик SQL-ін’єкцій за рахунок параметризованих запитів. CSRF-атаки блокуються стандартними механізмами фреймворків.

Застосування безкоштовного SSL/TLS-сертифіката Let’s Encrypt гарантує захищений канал передавання даних та ефективний захист від MITM-атак. Fail2Ban забезпечує мінімальний захист від brute-force атак шляхом блокування IP-адрес з підозрілою активністю.

У приведеному стеку Cloudflare Free надає лише базову фільтрацію запитів і не включає повноцінний WAF, що обмежує захист від складних XSS- та SQL-атак. Відсутні розширені механізми моніторингу та аудиту.

Отже, стек А є найбільш економічно вигідним, проте призначений виключно для навчальних або низькоризикових проєктів. Його застосування доцільне для прототипів систем дистанційного тестування, де обсяг даних і кількість користувачів є обмеженими.

– Стек В – Стек для малого бізнесу (оптимальне співвідношення ціни та захисту):

Склад: Let’s Encrypt, Cloudflare Pro (WAF), Authy/Auth0 (2FA), Burp Suite Professional, VPS.

Цей стек забезпечує високий рівень захисту для вебдодатків із середнім рівнем ризику. Cloudflare Pro включає повноцінний Web Application Firewall, який ефективно блокує SQL-ін’єкції, XSS-атаки, CSRF-спроби та brute-force атаки на рівні мережі.

Інтеграція двофакторної автентифікації (2FA) через Authy або Auth0 суттєво підвищує захищеність облікових записів користувачів. Це особливо важливо для систем тестування, де обробляються персональні дані та результати оцінювання.

Burp Suite Professional дозволяє проводити періодичне ручне тестування безпеки та виявляти логічні вразливості, які можуть бути пропущені автоматичними засобами.

Зростання кількості користувачів або трафіку може призвести до збільшення витрат (особливо на 2FA та WAF). Також стек потребує базової технічної підтримки.

Отже, стек B є найбільш збалансованим рішенням, що поєднує помірний бюджет із високим рівнем захисту. Він є оптимальним для впровадження у комерційні або університетські системи дистанційного тестування.

– Стек C – SaaS-орієнтований стек (професійний рівень):

Склад: AWS WAF, Okta/Auth0 (IAM + MFA), AWS KMS, Nessus, хмарна інфраструктура AWS/GCP.

Даний стек забезпечує корпоративний рівень інформаційної безпеки та відповідає вимогам масштабованих SaaS-рішень. AWS WAF дозволяє створювати кастомні правила фільтрації та адаптувати захист до специфіки навантаження.

Системи керування ідентифікацією та доступом (IAM) у поєднанні з MFA (Okta/Auth0) забезпечують централізований контроль доступу та значно зменшують ризик компрометації облікових записів.

AWS KMS відповідає за захищене зберігання та ротацію криптографічних ключів, що є критично важливим для відповідності вимогам GDPR та інших стандартів. Nessus забезпечує регулярний аудит вразливостей інфраструктури.

Основним недоліком є висока вартість впровадження та експлуатації, а також потреба у кваліфікованому персоналі для адміністрування.

Отже, стек C доцільний для великих комерційних платформ дистанційного тестування з високими вимогами до надійності, масштабованості та відповідності стандартам інформаційної безпеки.

У таблиці 3.14 наведено узагальнену таблицю порівняних стеків для захисту системи тестувань та опитувань.

Таблиця 3.14 – Порівняння запропонованих стеків для захисту системи тестувань та опитувань

Стек	Компонент захисту	Сервіс	Тип загрози, від якої захищає	Орієнтовна вартість на рік (USD)	Примітки
1	2	3	4	5	6
A – Навчальний (мінімальний бюджет)	HTTPS	Let’s Encrypt	MITM	0	Безкоштовний TLS-сертифікат
	Захист БД	ORM (Laravel / Django)	SQL-ін’єкції, CSRF	0	Вбудовано у фреймворк
	WAF (базовий)	Cloudflare Free	XSS, SQLi (частково)	0	Базова фільтрація
	Захист від brute-force	Fail2Ban	Brute-force	0	Open-source
	Хостинг	VPS (Hetzner /DigitalOcean)	–	~120	1–2 GB RAM
Разом для стеку A				~120	Найнижчий бюджет
B – Малий бізнес (оптимальний варіант)	HTTPS	Let’s Encrypt	MITM	0	Автоматичне оновлення
	WAF	Cloudflare Pro	SQLi, XSS, CSRF, brute-force	~300	~25 USD/міс
	2FA	Authy / Auth0	Brute-force, крадіжка акаунтів	~500	Залежить від кількості користувачів
	Тестування безпеки	Burp Suite Professional	SQLi, XSS	~400	Ручне тестування
	Хостинг	DigitalOcean VPS	–	~360	2 GB RAM

Продовження таблиці 3.14

1	2	3	4	5	6
Разом для стеку B				~1 560	Найкраще співвідношення ціна/захист
C – SaaS (професійний рівень)	WAF (Enterprise)	AWS WAF	Повний спектр вебатак	~3 000	Залежить від трафіку
	IAM + MFA	Okta / Auth0	Brute-force, викрадення доступу	~5 000	100–300 користувачів
	Керування ключами	AWS KMS	Компрометація ключів	~600	Key Management
	Аудит безпеки	Nessus	Комплексний аудит	~3 000	Enterprise-рівень
	Хмарна інфраструктура	AWS / GCP	–	~5 000	Середнє навантаження
Разом для стеку C				~16 600	Для комерційних SaaS

Запропоновані стеки захисту систем дистанційного тестування та опитувань істотно відрізняються за рівнем безпеки, функціональними можливостями та фінансовими витратами. Це дозволяє обрати оптимальний варіант залежно від масштабу системи, кількості користувачів та вимог до захисту даних.

Стек A, орієнтований на мінімальний бюджет, забезпечує базовий рівень захисту за рахунок використання безкоштовних інструментів і сервісів. Його доцільно застосовувати у навчальних або демонстраційних системах, де ризики компрометації даних є відносно низькими, а фінансові ресурси обмежені.

Стек B характеризується оптимальним співвідношенням між вартістю та рівнем захисту. Використання WAF, двофакторної автентифікації та засобів тестування безпеки дозволяє ефективно протидіяти основним вебзагрозам, що

робить цей стек найбільш придатним для університетських та комерційних систем дистанційного тестування середнього масштабу.

Стек С забезпечує найвищий рівень захисту та орієнтований на комерційні SaaS-рішення з великим навантаженням і підвищеними вимогами до безпеки. Водночас його впровадження потребує значних фінансових витрат і залучення кваліфікованих спеціалістів, що обмежує доцільність використання у невеликих проєктах.

Таким чином, результати порівняння підтверджують доцільність застосування диференційованого підходу до вибору стеку захисту, при якому рівень безпеки та бюджет узгоджуються з функціональними вимогами та масштабом системи дистанційного тестування та опитувань.

3.5 Оцінка ефективності сервісів захисту даних у системі тестування та опитувань

Для оцінки ефективності сервісів було впроваджено шкалу оцінювання рівня безпеки сервісів від 1 до 5.

Характеристика балів для шкали оцінювання рівня безпеки:

– 1 бал – дуже низький рівень безпеки, характеризується практично повною відсутністю засобів захисту інформації або їх формальною наявністю без реального функціонування. Передавання даних здійснюється без шифрування, автентифікація користувачів є примітивною або відсутньою, а система вразлива до більшості відомих вебатак (SQL-ін’єкцій, XSS, CSRF, brute-force);

– 2 бали – низький рівень безпеки, передбачає часткову реалізацію заходів захисту, однак без комплексного підходу. Наприклад, може бути використано шифрування каналу зв’язку без належного захисту бази даних або без механізмів протидії типовим атакам. Засоби захисту застосовуються фрагментарно, не забезпечують достатнього рівня надійності та не відповідають сучасним рекомендаціям з інформаційної безпеки. Система залишається вразливою до

автоматизованих атак і може бути використана лише у низькоризикових середовищах;

– 3 бали – середній рівень безпеки, забезпечує базовий захист інформації та відповідає мінімальним вимогам до вебсистем. На цьому рівні реалізуються основні механізми захисту: SSL/TLS-шифрування, хешування паролів, захист від SQL-ін’єкцій за допомогою підготовлених запитів, базові фільтри для запобігання XSS-атакам. Проте відсутній розширений контроль доступу, багатофакторна автентифікація та систематичний аудит безпеки;

– 4 бали – високий рівень безпеки, характеризується впровадженням комплексних заходів захисту, спрямованих на протидію основним сучасним вебзагрозам. Система використовує надійне шифрування даних, захист від SQL-ін’єкцій, XSS та CSRF-атак, механізми обмеження кількості спроб автентифікації, а також регулярне оновлення програмних компонентів. Реалізується розмежування прав доступу та базові елементи моніторингу безпеки;

– 5 балів – дуже високий рівень безпеки, передбачає застосування сучасних, комплексних і взаємодоповнюючих методів захисту інформації. Окрім стандартних механізмів, система використовує багатофакторну автентифікацію, WAF-рішення, автоматизований моніторинг подій безпеки, регулярне тестування на проникнення та відповідність міжнародним стандартам (OWASP, ISO/IEC 27001). Забезпечується захист як даних у процесі передавання, так і даних у стані зберігання.

Обрані критерії оцінювання сервісів безпеки:

– захист каналу передавання даних – характеризує здатність сервісу забезпечувати конфіденційність та цілісність інформації під час її передавання між клієнтом і сервером. Оцінюється наявність та коректність використання криптографічних протоколів SSL/TLS, підтримка сучасних алгоритмів шифрування (AES, RSA, ECC), автоматичне оновлення сертифікатів та захист від прослуховування мережевого трафіку;

– захист від SQL-ін’єкцій – визначає ефективність сервісу або платформи у запобіганні атакам, спрямованим на несанкціоноване виконання SQL-запитів. Оцінюється підтримка механізмів фільтрації вхідних даних, використання prepared statements, ORM, а також наявність Web Application Firewall (WAF) з відповідними правилами захисту;

– захист від XSS-атак (Cross-Site Scripting) – відображає можливості сервісу щодо запобігання впровадженню шкідливих сценаріїв у веб-сторінки. Оцінюється наявність механізмів екранування даних, Content Security Policy (CSP), HTTP-заголовків безпеки, а також можливість фільтрації небезпечного JavaScript-коду;

– захист від CSRF-атак (Cross-Site Request Forgery) – характеризує здатність сервісу запобігати підробці запитів від імені автентифікованого користувача. Оцінюється реалізація CSRF-токенів, перевірка заголовків Origin і Referer, а також підтримка механізмів SameSite cookies;

– захист від brute-force атак – даний критерій визначає ефективність протидії багаторазовим спробам підбору паролів або облікових даних. Оцінюється наявність механізмів обмеження кількості спроб входу, тимчасового блокування облікових записів, CAPTCHA, а також підтримка багатофакторної автентифікації (2FA/MFA);

– захист від MITM-атак (Man-in-the-Middle) – відображає здатність сервісу протидіяти атакам типу «людина посередині», при яких зловмисник намагається перехопити або змінити дані під час передавання. Оцінюється коректна реалізація TLS, використання перевірених сертифікатів, HSTS, certificate pinning та захист від підміни DNS;

– масштабованість та надійність – даний критерій оцінює здатність сервісу стабільно функціонувати при зростанні кількості користувачів та навантаження. Враховується наявність резервування, балансування навантаження, захисту від DDoS-атак, а також гарантований рівень доступності (SLA);

– простота інтеграції – характеризується зручністю впровадження сервісу в існуючу систему тестування та опитувань. Оцінюється наявність API, SDK,

детальної документації, прикладів інтеграції, а також сумісність з популярними мовами програмування та фреймворками.

Ступінь безпеки сервіса, за вище вказаними критеріями, можно розрахувати за формулою:

$$G = \sqrt[n]{x_1 \cdot x_2 \cdot \ldots \cdot x_n} \quad (3.1)$$

– де: G – ступінь безпеки сервіса, x_n – оцінка захисту від певної загрози, n – ступінь кількості факторів.

Для розрахунку було обрано вище наведені критерії. В таблиці 3.15 представлена оцінка безпеки сервісі за обраними критеріями.

Таблиця 3.15 – Оцінка безпеки сервісів за обраними критеріями

Сервіс	Параметр								G
	Канал	SQL	XSS	CSRF	Brute-force	MITM	Масштаб	Інтеграція	
Let’s Encrypt	5	1	1	1	1	5	5	5	2.23
Cloudflare Free	3	3	3	3	3	4	4	5	3.43
Cloudflare Pro	4	5	5	4	5	4	5	5	4.59
Authy	1	1	1	1	5	4	4	4	2.05
Auth0	1	1	1	1	5	5	5	4	2.17
Hetzner	3	3	3	3	3	3	3	4	3.1
DigitalOcean	4	4	4	4	4	4	4	4	4
AWS	5	5	5	5	5	5	5	3	4.66
Burp Suite	1	5	5	5	4	1	3	3	2.85
Nessus	1	5	5	5	5	1	4	3	3.06

Сервіси мають суттєві відмінності в рівнях безпеки та захищеності, що відображено у фінальному показнику ступеню безпеки. Найвищий рівень комплексної безпеки демонструють хмарні платформи та платні рішення, тоді як спеціалізовані сервіси аутентифікації та безкоштовні плани мають вужчу спрямованість.

На рисунку 3.1 наведена гістограма результатів розрахунків ступеню безпеки для сервісів.

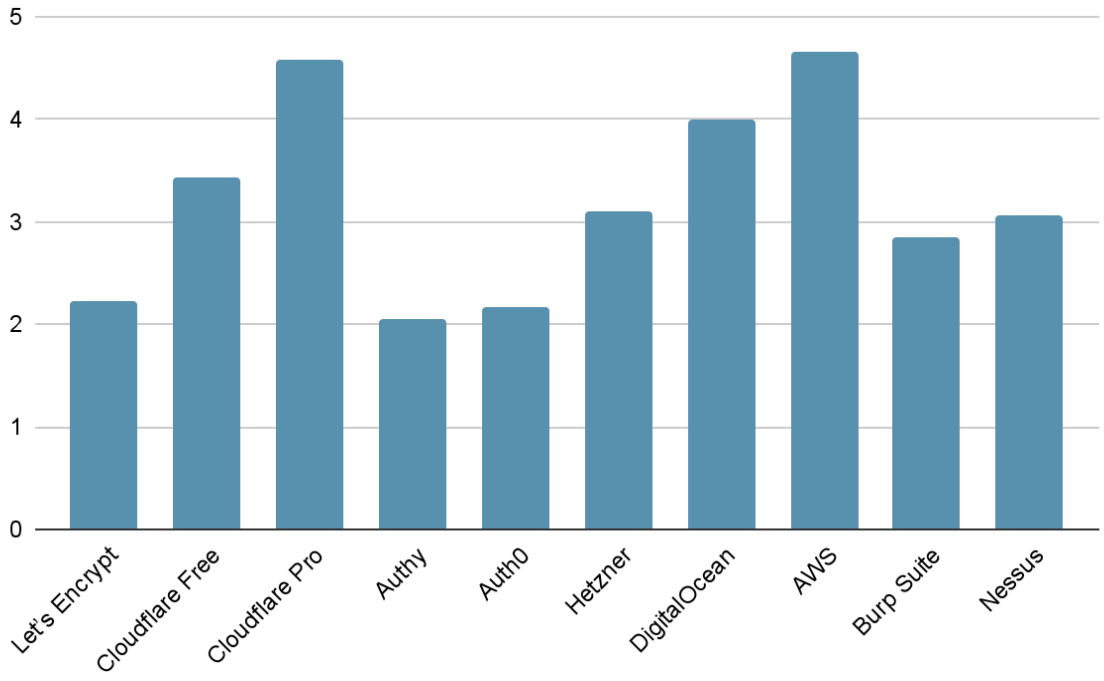

Рисунок 3.1 – Гістограма розрахунків ступеню безпеки розглянутих сервісів

При аналізі безпеки сервісів за обраними критеріями лідером за рівнем безпеки виявився AWS 93.2% – найвищий інтегральний рівень безпеки, на 11.4% безпечніший за DigitalOcean та 49% безпечніший за Let’s Encrypt. Cloudflare Pro має 91.8% лише на 1.4% поступається AWS та на 23.2% перевищує Cloudflare Free. AWS і Cloudflare Pro формують професійний рівень захисту, придатний для критичних систем тестування (іспити, сертифікація).

Середній рівень захисту має DigitalOcean (80.0%), на 17.4% безпечніший за Cloudflare Free та на 18% перевищує Hetzner. Cloudflare Free (68.6%), на 24%

безпечніший за Let’s Encrypt, забезпечує базовий захист без фінансових витрат. Ці сервіси оптимальні для навчальних або малих систем тестування.

Низький рівень інтегральної безпеки (спеціалізовані сервіси) Authy (41.0%) та Auth0 (43.4%) мають низький показник безпеки, але це не означає слабку безпеку вони захищають лише один аспект – автентифікацію. Let’s Encrypt (44.6%) забезпечує лише захист каналу (TLS), на 48.6% слабший за Cloudflare Pro. Ці сервіси не є комплексними, але критично важливі як частина загальної системи захисту.

Аналіз показника безпеки G показав, що найвищий рівень захищеності забезпечує платформа AWS із показником 93,2%, що на 52,2% перевищує рівень спеціалізованих сервісів автентифікації. Cloudflare Pro продемонстрував порівнянний рівень безпеки (91,8%), забезпечуючи ефективний захист від типових вебатак. Сервіси середнього рівня, зокрема DigitalOcean (80,0%) та Cloudflare Free (68,6%), можуть бути використані в навчальних системах дистанційного тестування. Отримані результати підтверджують доцільність використання комбінованого підходу до захисту вебсистем.

4 ПРОЄКТУВАННЯ СИСТЕМИ ДИСТАНЦІЙНОГО ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ МЕТОДІВ ЗАХИСТУ

4.1 Архітектура та програмне середовище реалізації системи

Об’єктом експериментальних досліджень у даному розділі є вебсистема дистанційного тестування та опитувань, з використанням трирівневої архітектури. Система призначена для проведення онлайн-тестування користувачів, збереження результатів та обробки персональних даних.

Архітектура системи включає в себе 3 рівні:

– презентаційний рівень – вебінтерфейс користувача, реалізований з використанням HTML, CSS та JavaScript;

– рівень прикладної логіки – серверна частина на основі PHP, що відповідає за бізнес-логіку, автентифікацію та обробку запитів;

– рівень даних – база даних для зберігання облікових записів, тестів та результатів.

4.2 Середовище розгортання системи тестувань

Для проведення експериментальних досліджень та оцінки ефективності методів захисту даних у системі дистанційного тестування було обрано локальне програмне середовище OpenServer. Вибір даного середовища зумовлений необхідністю створення контрольованих умов для моделювання роботи веб-системи, реалізації механізмів захисту та відтворення типових веб-атак без ризику для реальних користувацьких даних.

OpenServer є популярним локальним серверним пакетом для операційної системи Windows, який дозволяє швидко розгорнути повноцінне середовище для розробки та тестування веб-застосунків. Він об’єднує в собі всі необхідні компоненти для функціонування системи дистанційного тестування, зокрема

веб-сервер, інтерпретатор серверної мови програмування та систему керування базами даних [23].

У межах даного дослідження використовувалися такі компоненти OpenServer:

– Apache – вебсервер, який забезпечує обробку HTTP/HTTPS-запитів клієнтів, підтримує механізми конфігурації доступу, логування подій, а також інтеграцію з TLS для захищеного передавання даних;

– PHP – серверна мова програмування, яка використовується для реалізації бізнес-логіки системи тестування, обробки запитів користувачів, автентифікації, авторизації та взаємодії з базою даних;

– MySQL – система керування реляційними базами даних, призначена для зберігання інформації про користувачів, результати тестування, облікові дані та налаштування системи.

Комбінація зазначених компонентів дозволяє реалізувати класичну трирівневу архітектуру вебзастосунку, що відповідає сучасним вимогам до побудови систем дистанційного тестування та опитувань.

Застосування OpenServer у межах кваліфікаційної роботи має низку суттєвих переваг:

– контрольоване середовище – усі експерименти проводяться локально, що дозволяє точно фіксувати параметри системи та повторювати результати;

– безпека тестування – відсутність підключення до публічної мережі мінімізує ризики витоку даних під час моделювання атак;

– можливість відтворення атак – середовище OpenServer дозволяє безпечно тестувати SQL-ін’єкції, XSS-атаки, brute-force спроби та інші загрози без порушення законодавства чи етичних норм;

– гнучкість налаштувань – можливість змінювати конфігурацію Apache, PHP та MySQL для дослідження впливу різних параметрів безпеки;

– швидкість розгортання – мінімальні часові витрати на встановлення та підготовку середовища.

Таким чином, використання OpenServer є доцільним і обґрунтованим вибором для реалізації практичної частини дослідження, оскільки воно

забезпечує необхідні умови для повноцінного аналізу ефективності методів захисту даних у системі дистанційного тестування та опитувань.

4.3 Опис бази даних системи дистанційного тестування та опитувань

Для зберігання та обробки інформації у системі дистанційного тестування та опитувань використовується реляційна база даних MySQL. База даних призначена для централізованого зберігання відомостей про користувачів системи, тестові матеріали, питання, варіанти відповідей, а також результати проходження тестів.

Проєктування бази даних здійснювалося з урахуванням принципів нормалізації, що дозволяє уникнути дублювання даних, забезпечити цілісність інформації та підвищити продуктивність системи. Структура бази даних відповідає логіці трирівневої архітектури та забезпечує ефективну взаємодію з серверною частиною системи. Для взаємодії з базою даних на сервер було встановлено програмне середовище phpMyAdmin.

На рисунку 4.1 наведена база даних test для системи дистанційного тестування, яка складається з 3 взаємопов’язаних таблиць, кожна з яких виконує окрему функціональну роль.

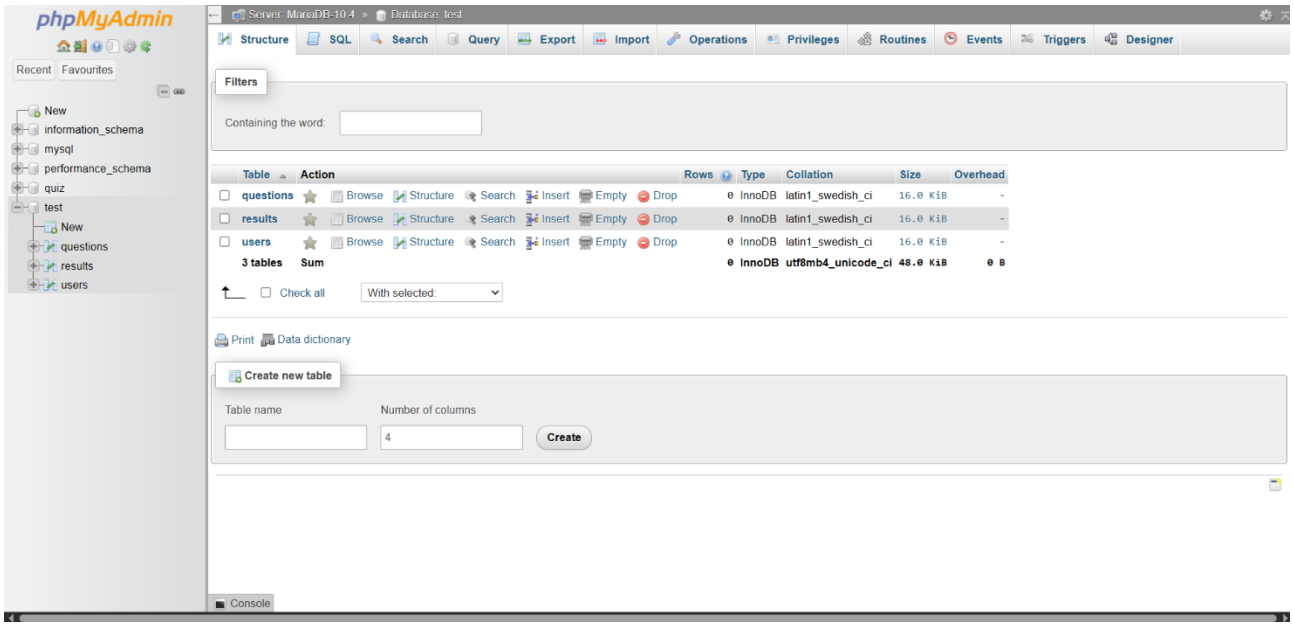

Рисунок 4.1 – База даних test

Таблиця questions містить інформацію про тестові завдання, доступні в системі. Таблиця забезпечує можливість проходження тестів та їх відображення користувачам. На рисунку 4.2 наведено структуру таблиці questions.

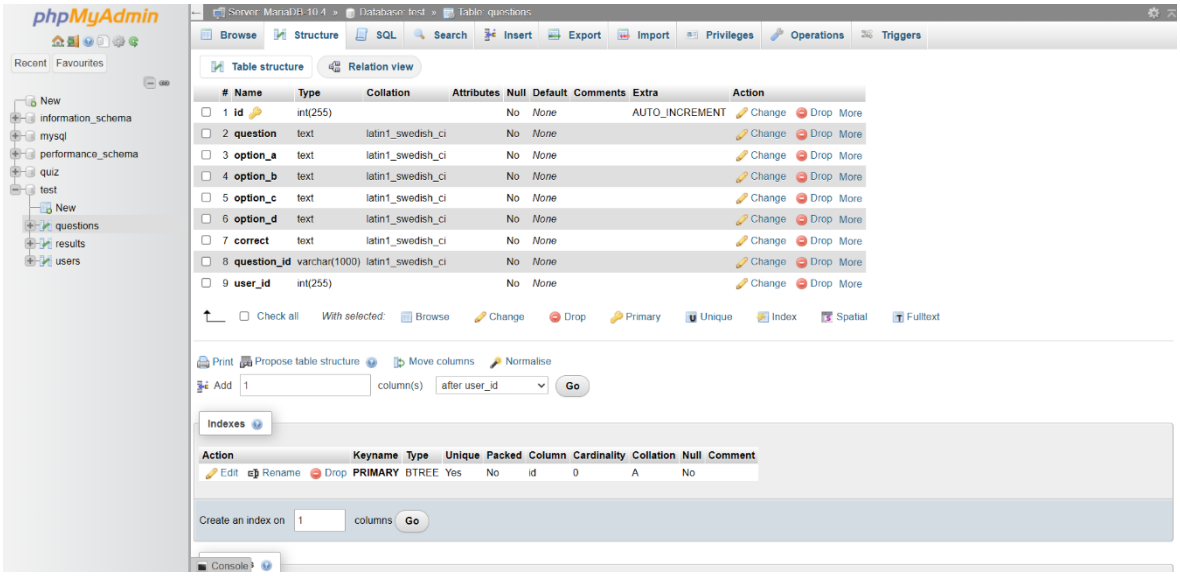

Рисунок 4.2 – Структура таблиці questions

Таблиця results призначена для зберігання результатів проходження тестів користувачами. Таблиця забезпечує збереження історії тестування та можливість аналізу результатів. На рисунку 4.3 наведена структура таблиці result.

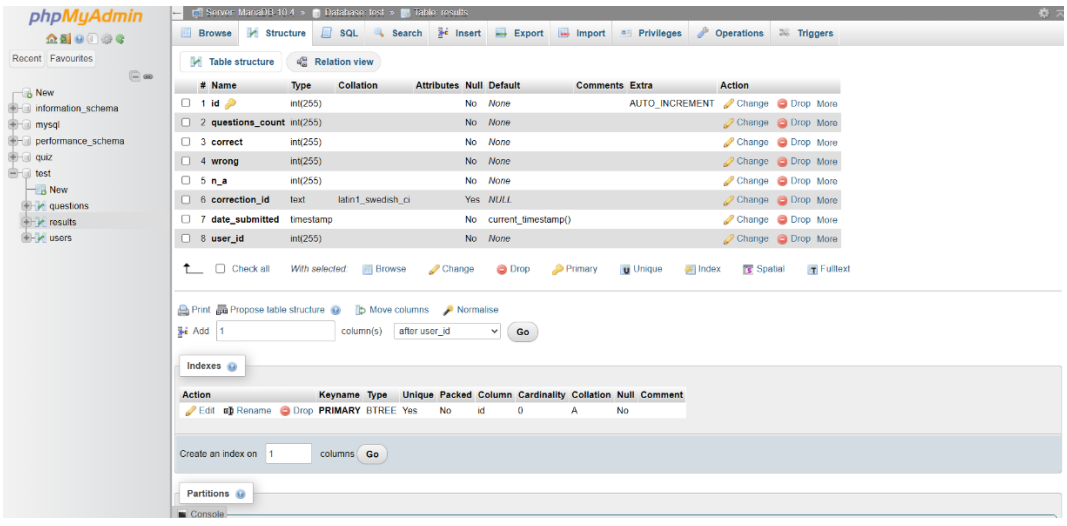

Рисунок 4.3 – Структура таблиці result

Таблиця users призначена для зберігання облікових записів користувачів системи. На рисунку 4.4 наведена таблиця user, яка використовується для реалізації механізмів автентифікації та авторизації користувачів.

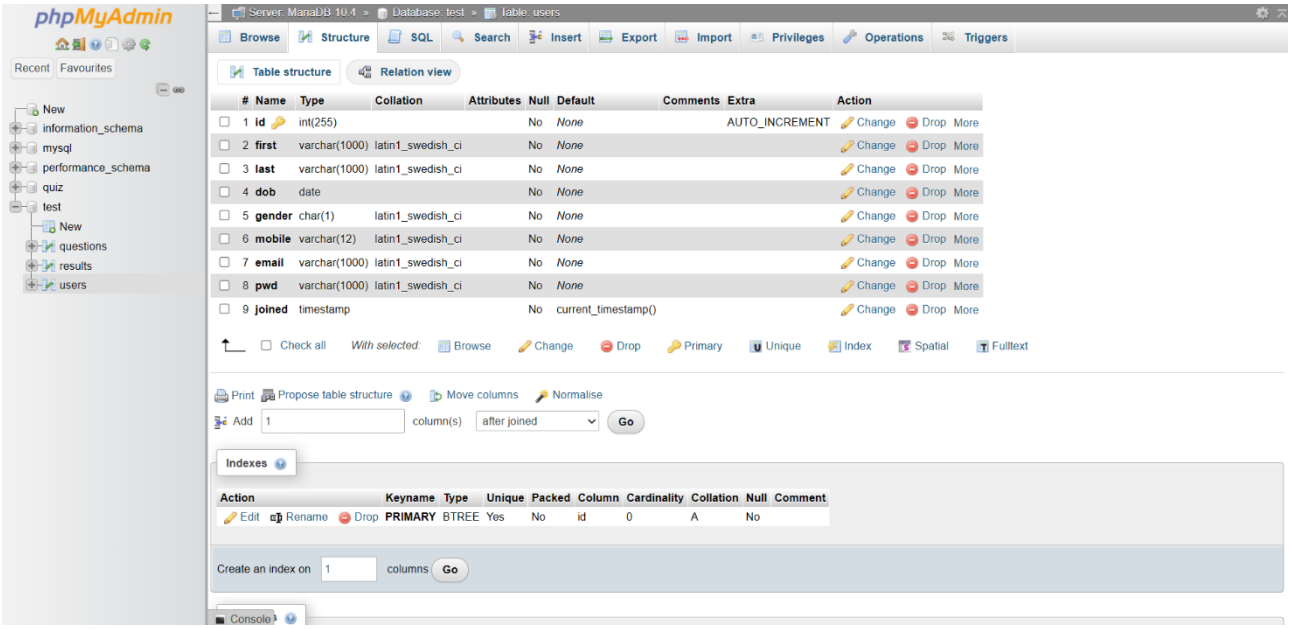

Рисунок 4.4 – Структура таблиці users

Розроблена структура бази даних забезпечує надійне зберігання інформації, масштабованість системи та можливість реалізації сучасних методів захисту даних. Використання MySQL у поєднанні з серверною логікою на PHP дозволяє ефективно інтегрувати базу даних у систему дистанційного тестування та забезпечити її безпечну роботу.

4.4 Вибір атаки для експериментального дослідження

У межах експериментального дослідження ефективності методів захисту даних у системі дистанційного тестування та опитувань необхідно було обрати типовий і водночас найбільш критичний вид вебатаки, який дозволить на практиці оцінити рівень захищеності реалізованої системи. З урахуванням особливостей досліджуваної системи, а також аналізу сучасних загроз

інформаційній безпеці, для подальшого експериментального аналізу було обрано атаку типу SQL-ін’єкція (SQL Injection).

SQL-ін’єкція є одним із найпоширеніших та найбільш небезпечних класів атак на вебзастосунки, які використовують реляційні бази даних. Суть атаки полягає у впровадженні зловмисником шкідливих SQL-конструкцій у вхідні параметри веб-застосунку з метою зміни логіки виконання запитів до бази даних.

Згідно з класифікацією OWASP Top 10, атаки, пов’язані з ін’єкціями, протягом багатьох років займають провідні позиції серед найкритичніших загроз веб-безпеці. У версіях OWASP Top 10 різних років SQL-ін’єкції стабільно входять до першої трійки загроз, що свідчить про їхню актуальність та масовість застосування [24].

Системи дистанційного тестування та опитувань мають низку особливостей, які роблять їх особливо вразливими до SQL-ін’єкцій:

– активне використання форм введення даних (авторизація, реєстрація, проходження тестів, збереження відповідей);

– наявність баз даних, що містять персональні дані користувачів, результати тестування, облікові записи та паролі;

– постійна взаємодія між клієнтською частиною та сервером у режимі реального часу;

– можливість автоматизованих атак через відкриті HTTP-запити.

У разі успішної SQL-ін’єкції зловмисник може отримати несанкціонований доступ до бази даних, змінити або знищити результати тестування, отримати облікові дані користувачів або повністю порушити працездатність системи. Для освітніх систем це може призвести до спотворення результатів оцінювання, втрати довіри до системи та порушення принципів академічної доброчесності.

Вибір SQL-ін’єкції для експериментального дослідження у межах кваліфікаційної роботи є доцільним з таких причин:

– SQL-ін’єкція є репрезентативною атакою, яка дозволяє комплексно оцінити ефективність серверних механізмів захисту;

– для даного типу атак існують чітко визначені методи протидії, що дозволяє порівняти різні підходи до захисту;

– атака легко відтворюється в контрольованому середовищі OpenServer без ризику для реальних даних;

– результати експерименту можуть бути кількісно оцінені за заздалегідь визначеними критеріями безпеки.

Таким чином, SQL-ін’єкція є обґрунтованим вибором для проведення експериментального дослідження, оскільки вона дозволяє наочно продемонструвати рівень захищеності системи дистанційного тестування та оцінити ефективність реалізованих методів захисту даних.

4.5 Реалізація методів захисту в системі

У межах практичної частини кваліфікаційної роботи в досліджуваній системі дистанційного тестування було реалізовано комплекс базових та прикладних методів захисту даних. Основна увага приділялася захисту каналу передавання інформації, запобіганню SQL-ін’єкціям, а також підвищенню безпеки механізмів автентифікації користувачів.

У системі було реалізовано такі методи захисту:

– для захисту каналу передавання інформації було створено локальний SSL-сертифікат. OpenServer автоматично створює локальний сертифікат;

– налаштування віртуального хоста HTTPS;

Після налаштування сайт відкривається в браузері (рис. 4.5).

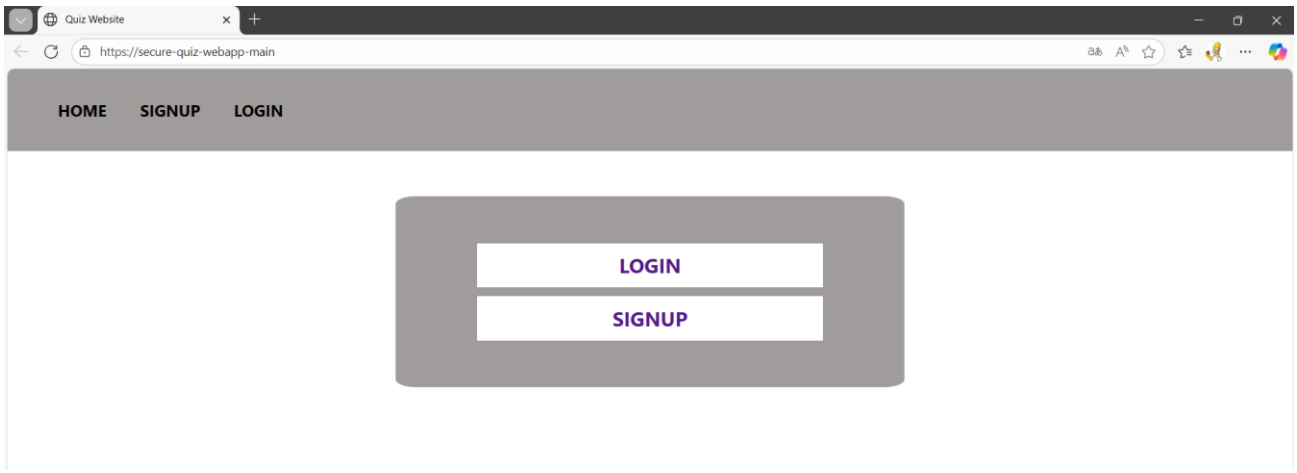

Рисунок 4.5 –Інтерфейс системи тестувань та опитувань на сервері

У середовищі OpenServer захищене передавання даних було реалізовано шляхом використання протоколу HTTPS із застосуванням SSL/TLS. Для цього було задіяно вбудований механізм локальних сертифікатів OpenServer та налаштовано веб-сервер Apache. У результаті всі дані, що передаються між клієнтом і сервером, зокрема облікові дані користувачів і результати тестування, передаються у зашифрованому вигляді, що унеможливлює їх перехоплення.

4.6 Сценарії тестування SQL-ін’єкцій

4.6.1 SQL-ін’єкція у формі авторизації

Форма авторизації є одним із найбільш критичних елементів системи дистанційного тестування, оскільки забезпечує доступ користувачів до персональних даних, результатів тестування та функцій керування. Саме тому вона часто стає об’єктом атак типу SQL-ін’єкція.

SQL-ін’єкція полягає у введенні зловмисником спеціально сформованих SQL-фрагментів у поля введення (логін або пароль) з метою зміни логіки SQL-запиту, що виконується сервером.

У вразливій версії системи авторизація реалізована шляхом безпосередньої конкатенації введених користувачем даних у SQL-запит:

```
$query = "SELECT * FROM users WHERE login = '$login' AND password = '$password'";
```

За такої реалізації введені дані не перевіряються та не екрануються, що дозволяє виконувати довільні SQL-операції.

Тестові вхідні дані (payload):

– поле логін: `' OR 1=1 -- ;`

– поле пароль: будь-який текст.

Сформований SQL-запит виглядає так:

```
SELECT * FROM users WHERE login = '' OR 1=1 -- ' AND password = '...';
```

У результаті умова OR 1=1 завжди істинна, а частина запиту після -- ігнорується як коментар.

Результат атаки на вразливу систему:

– відбувається обхід автентифікації;

– користувач отримує доступ без введення коректних облікових даних;

– можливий вхід під обліковим записом адміністратора;

– порушується конфіденційність та цілісність даних.

Таким чином, вразлива система не забезпечує базового рівня захисту від SQL-ін’єкцій.

У захищеній системі, розгорнутій у середовищі Open Server, авторизація реалізована із застосуванням параметризованих SQL-запитів (Prepared Statements):

```
$stmt = $pdo->prepare("SELECT * FROM users WHERE login = :login");
$stmt->execute(['login' => $login]);
```

Паролі зберігаються у базі даних у вигляді хешів (Argon2), а перевірка здійснюється за допомогою функції password_verify().

Спроба атаки на захищену систему при введенні того ж payload:

– поле логін: ' OR 1=1 -- ;

– поле пароль: будь-який текст.

Як результат він сприймається виключно як рядкове значення, а не як частина SQL-коду.

Результат:

– SQL-запит не змінює свою логіку;

– доступ до системи не надається;

– система коректно повертає повідомлення про помилку входу;

– структура бази даних не розкривається.

4.6.2 SQL-ін’єкція через GET-параметри

Отримання несанкціонованого доступу до даних шляхом модифікації логіки SQL-запиту через GET-параметри.

Передумови для вразливої системи:

– використання динамічних SQL-запитів;

– відсутність параметризованих запитів;

– відсутність валідації вхідних даних.

Імітований запит має вигляд:

```
test.php?id=1 OR 1=1
```

Переданий параметр id без перевірки включається до SQL-запиту:

```
SELECT * FROM users WHERE id = 1 OR 1=1;
```

Результат:

– умова OR 1=1 завжди істинна;

– сервер повертає всі записи таблиці;

– механізми контролю доступу обходяться;

– можливий витік конфіденційних даних.

Вразлива система дозволяє експлуатацію SQL-ін’єкцій через GET-параметри, що становить критичну загрозу інформаційній безпеці.

У захищеній системі реалізовано:

– параметризовані запити (Prepared Statements) – SQL-запит будується з використанням плейсхолдерів замість вставки значень напряму:

```
SELECT * FROM users WHERE id = ?
```

Значення GET-параметра id передається як окремий параметр, а не вставляється у рядок запиту. Це унеможливлює зміну логіки SQL-запиту навіть при спробі ввести 1 OR 1=1 ;

– валідація та перевірка типу даних, параметр id перевіряється на відповідність очікуваному типу (ціле число). Будь-які символи, що не відповідають числу, автоматично відхиляються або замінюються на безпечні значення;

– фільтрація спеціальних символів, забороняються символи, що можуть вплинути на SQL-запит (', ", ;, --, OR, AND). Це працює як додатковий рівень безпеки, якщо запит випадково не параметризований;

– обмеження повідомлень про помилки, сервер не повертає детальні SQL-помилки користувачу, щоб зловмисник не отримав підказки для атаки.

Як результат спроба атаки test.php?id=1 OR 1=1 не змінює логіку запиту, і доступ до несанкціонованих даних відсутній.

4.6.3 Спроба отримання структури бази даних (UNION-ін’єкція)

Метою атаки є отримання технічної інформації про базу даних за допомогою UNION-ін’єкції.

Передумови для вразливої системи:

– динамічна побудова SQL-запитів;

– відсутність контролю кількості стовпців;

– відображення повідомлень SQL-помилок.

Імітація атаки:

```
1 UNION SELECT null, database()
```

У ході атаки SQL-запит набуває вигляду:

```
SELECT id, name FROM products WHERE id = 1
UNION SELECT null, database();
```

Результат:

– сервер повертає назву поточної бази даних;

– зловмисник отримує інформацію про структуру БД;

– створюються передумови для подальших атак.

Отже, вразлива система дозволяє виконання UNION-ін’єкцій та розкриття службової інформації, що значно підвищує ризик компрометації даних.

У захищеній системі реалізовано:

– параметризовані запити – UNION-запити будуються з використанням параметрів, а не конкатенацією рядків:

```
SELECT id, name FROM products WHERE id = ?
```

Значення параметра id із запиту GET або POST передається окремо і не може містити частину SQL-коду (UNION SELECT);

– валідація кількості та типів стовпців, сервер перевіряє, що запит містить очікувану кількість стовпців з правильними типами. Спроба вставити додаткові стовпці через UNION призводить до помилки на етапі валідації, не на рівні бази даних;

– приховування помилок SQL, будь-які помилки запиту (синтаксису, типів даних) не відображаються користувачу. Це унеможливлює отримання технічної інформації про структуру бази даних;

– обмеження структури запитів, сервер дозволяє лише запити із жорстко визначеною логікою (SELECT id, name FROM products WHERE id = ?). Будь-які спроби додати додаткові SELECT-вирази через UNION блокується.

Як результат спроба атаки 1 UNION SELECT null, database() не виконується, сервер повертає повідомлення про некоректні дані без розкриття будь-якої інформації про БД.

4.7 Результати тестування захисту системи від SQL-ін’єкцій

У межах дослідження було проведено тестування веб-застосунку на наявність вразливостей типу SQL-ін’єкція. Тестування виконувалося шляхом імітації типових атак через форму авторизації, GET-параметри та UNION-запити. Отримані результати дозволили оцінити рівень захищеності системи. Також використовувалась версія системи без впроваджених методів захисту, для порівняння.

Для перевірки блокування SQL-ін’єкцій було проведено 10 тестових ін’єкцій (' OR 1=1 --, UNION SELECT, вкладені запити). Стабільність системи була розрахована на результатах 20 навантажувальних та ін’єкційних запитів. Для перевірки захисту бази даних було проведено 10 спроб виконання шкідливих SQL-команд (DROP, SELECT *, UNION), а для методу приховування SQL-помилок 10 помилкових запитів.

У таблиці 4.1 показано порівняння ефективності захисту від SQL-ін’єкцій у вразливій та захищеній системі.

Таблиця 4.1 – Порівняння ефективності захисту від SQL-ін’єкцій

Критерій оцінювання	Вразлива система, %	Захищена система, %
Блокування SQL-ін’єкцій	10	90
Стабільність системи під атакою	40	95
Захист бази даних	20	90
Приховування SQL-помилок	20	90

Середній рівень ефективності	21.25	91.25

Кількісний аналіз показав, що блокування SQL-ін’єкцій у захищеній системі ефективніше на 80% порівняно з вразливою. Це підтверджує доцільність реалізації механізмів захисту на рівні SQL-запитів.

Стабільність системи під атакою на 55% краще у захищеної системи, оскільки система не завершує аварійно роботу під час спроб ін’єкцій, зберігаючи працездатність та цілісність даних.

Рівень захисту бази даних у захищеній системі на 70% вище ніж у вразливій, що підтверджує усунення ризику несанкціонованого доступу та витоку інформації.

Використання методу приховування SQL-помилок підвищив захист на 70%, що унеможливлює отримання зловмисником службової інформації про структуру бази даних або логіку запитів.

На рисунку 4.6 зображена гістограма цього порівняння.

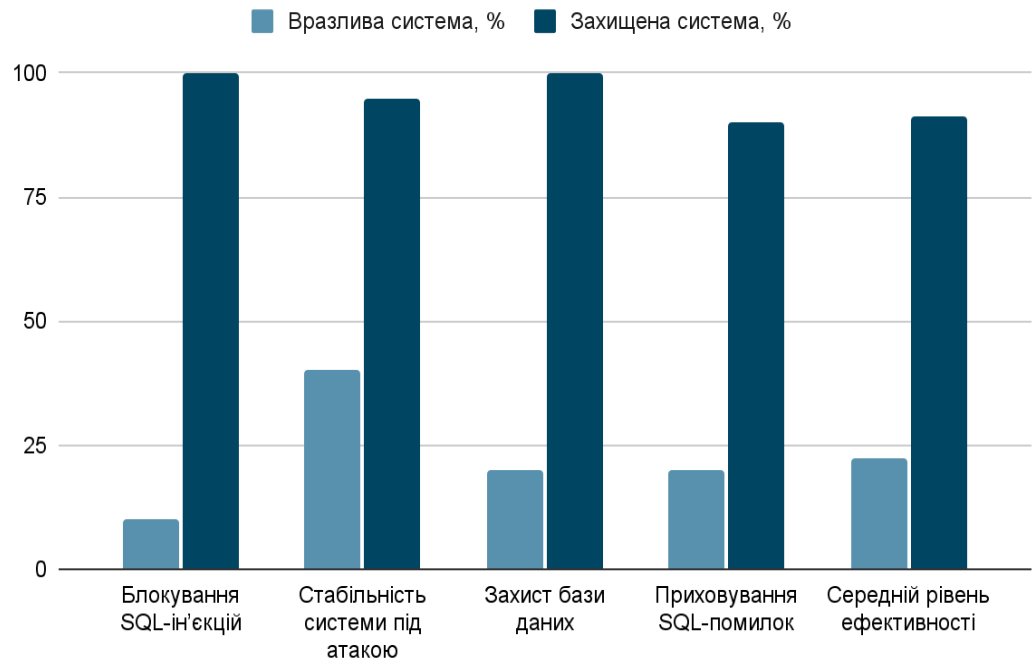

Рисунок 4.6 – Гістограма порівняння ефективності захисту від SQL-ін’єкцій

Проведене дослідження показало, що впровадження параметризованих SQL-запитів та супутніх механізмів захисту підвищує рівень безпеки системи більш ніж у 4 рази (з 22.5% до 91.25%). Це підтверджує, що навіть базові, але коректно реалізовані методи захисту дозволяють ефективно протидіяти одній з найпоширеніших атак на вебсистеми дистанційного тестування.

ВИСНОВКИ

У кваліфікаційній роботі було досліджено сучасні системи дистанційного тестування та опитувань, визначено основні принципи їх побудови, функціонування та проблеми, пов’язані із забезпеченням захисту даних користувачів.

У першому розділі розглянуто концептуальні основи систем дистанційного тестування, їх архітектурні особливості, типи реалізацій і функціональні можливості. Проаналізовано найбільш поширені платформи – такі як Moodle, Google Forms, ClassMarker, а також спеціалізовані корпоративні рішення. На основі проведеного аналізу виявлено, що більшість сучасних систем орієнтовані на зручність використання та масштабованість, проте питання захисту персональних даних і забезпечення цілісності результатів тестування не завжди реалізовані належним чином.

Особливу увагу було приділено типам інформаційних загроз, серед яких – несанкціонований доступ, модифікація даних, фальсифікація результатів, витік персональної інформації тощо. Це підтвердило актуальність дослідження методів захисту даних саме в контексті освітніх вебсистем.

У другому розділі здійснено комплексний огляд методів і засобів захисту даних, що застосовуються в сучасних інформаційних системах. Детально розглянуто принципи інформаційної безпеки (конфіденційність, цілісність, доступність), а також криптографічні методи захисту – симетричне та асиметричне шифрування, хешування й цифрові підписи.

В третьому розділі було проведено оцінку ефективності засобів захисту та виконано порівняльний аналіз кількох стеків безпеки з різним рівнем фінансових витрат. Найбільш доцільним для впровадження у навчальних та університетських системах дистанційного тестування є стек B, оскільки він дозволяє продемонструвати сучасні підходи до захисту даних без надмірних витрат.

У четвертому розділі було спроєктовано та реалізовано вебсистему дистанційного тестування, побудовану на основі трирівневої архітектури, що включає рівень представлення, серверну логіку та рівень доступу до бази даних. Система була розгорнута у середовищі Open Server та інтегрована з реляційною базою даних для зберігання користувацьких даних, результатів тестування та службової інформації. Було обрано та обґрунтовано вибір досліджуваної атаки, проведено тестування впроваджених методів захисту від SQL-ін’єкцій. Результати були порівнянні з тією ж систему, але без методів захисту.

Експериментальні дослідження в локальному середовищі OpenServer підтвердили високу ефективність впроваджених методів. Зокрема, використання параметризованих SQL-запитів, що дозволило підвищити рівень захисту від ін'єкцій більш ніж у 4 рази (з 22.5% до 91.25%).

Запропоновані теоретичні та практичні рекомендації можуть бути використані при розробці та модернізації систем дистанційного тестування та опитувань з метою підвищення рівня захисту персональних даних та результатів тестування.

ПЕРЕЛІК ПОСИЛАНЬ

1. Пєтухова І. О. Тестування навчальних досягнень здобувачів вищої освіти в умовах дистанційного навчання [Електронний ресурс] / І. О. Пєтухова // Інноваційна педагогіка. - 2020. - Вип. 30(2). - С. 159-162. – Режим доступу до ресурсу: https://ir.dpu.edu.ua/handle/123456789/3887.

2. Сучасні підходи при розробці архітектури програмного забезпечення / Д. О. Соболєв, Б. В. Федишен, І. К. Денисов // Вінницький національний технічний університет. – 13.11.25. – 3 с. – Режим доступу до ресурсу: https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/41816/20648.pdf.

3. Moodle [Електронний ресурс] // Сайт компанії «Moodle» – 14.11.2025. – Режим доступу до ресурсу: https://moodle.org/.

4. From paper to pixels: Using Google Forms for collaboration and assessment [Електронний ресурс]/ Mireille Djenno; Glenda M. Insua; Annie Pho // Library Hi Tech News. – 2015. – Вип.32(4) – С. 9–13. – Режим доступу до ресурсу: https://doi.org/10.1108/LHTN-12-2014-0105.

5. Testportal [Електронний ресурс] // Сайт компанії «Testportal» – 14.11.2025. – Режим доступу до ресурсу: https://www.testportal.com/.

6. Kahot! [Електронний ресурс] // Сайт компанії «Kahot!» – 15.11.2025. – Режим доступу до ресурсу: https://kahoot.com/.

7. Analysis of cybersecurity standard and framework components / Syafrizal M., Selamat S. R., Zakaria N. A. // International Journal of Communication Networks and Information Security (IJCNIS). – Вип. 12(3), – 2020. – С. 417–432.

8. Загальні принципи проведення тестування інформаційної безпеки підприємства / О. А. Курченко, М. В. Бржевський, А. Б. Гребенніков, В. І. Корсун // Сучасний захист інформації. – 2018. – № 4. – С.27–34 – Режим доступу до ресурсу: DOI: 10.31673/2409-7292.2018.042229.

9. Механізми забезпечення захисту даних у піринговых мережах / В. В. Лужецький, Л. М. Савицька // Herald of Khmelnytskyi National University.

Technical sciences. – 2024. – Том 339, № 4. – С. 503–508. – Режим доступу до ресурсу: https://doi.org/10.31891/2307-5732-2024-339-4-74.

10. Аналіз існуючих методів захисту та шифрування інформації / Р. С. Дробиш, М. В. Люта // Всеукраїнська конференція здобувачів вищої освіти і молодих учених. – 2021. – С. 250-253. – Режим доступу до ресурсу: https://er.knutd.edu.ua/bitstream/123456789/19525/1/Innovatyka2021_V1_P250-253.pdf.

11. Прикладні та методичні аспекти застосування хеш-функцій в інформаційній безпеці / Ю. Жданова, С. Спасітєлєва, С. Шевченко, К. Кравчук // Кібербезпека: освіта, наука, техніка. – 2020. – Том 4, № 8. – С. 85–96. – Режим доступу до ресурсу: https://oaji.net/articles/2020/8096-1594932565.pdf.

12. Authentication and authorization infrastructures (AAIs): a comparative survey [Електронний ресурс] / J. Lopez, R. Oppliger, G. Pernul // Computers & Security. – 2004. – Вип. 23, No. 7. – С. 578–590. – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/abs/pii/S0167404804001828?via%3Dihub.

13. Two-Factor Authentication / B. Lakshmiraghavan // Pro ASP.NET Web API Security. – 2013. – С. 319–343. – Режим доступу до ресурсу: https://link.springer.com/chapter/10.1007/978-1-4302-5783-7_14.

14. Applying OAuth2 and JWT protocols in securing distributed API gateways: best practices and case review / T. P. Gbenle, A. A. Abayomi, A. C. Uzoka, J. C. Ogeawuchi, O. S. Adanigbo, O. T. Odofin // International Journal of Multidisciplinary Research and Growth Evaluation. – 2022. – С. 628–634. – Режим доступу до ресурсу: DOI: 10.54660/.IJMRGE.2022.3.5.628-634.

15. Research on SQL Injection Attack and Prevention Technology Based on Web / L. Ma, D. Zhao, Y. Gao, C. Zhao // 2019 International Conference on Computer Network, Electronic and Automation (ICCNEA). – 2019. – С. 176–179. – Режим доступу до ресурсу: DOI: 10.1109/ICCNEA.2019.00042.

16. Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art / S. Gupta, B. B. Gupta // International Journal of

System Assurance Engineering and Management. – 2017. – Том 8, – С. 512–530. – Режим доступу до ресурсу: DOI: 10.1007/s13198-015-0376-0.

17. Research on SQL injection attack and prevention technology based on Web / L. Ma, D. Zhao, Y. Gao, C. Zhao // 2019 International Conference on Computer Network, Electronic and Automation (ICCNEA). – 2019. – С. 176–179. – DOI: 10.1109/ICCNEA.2019.00042.

18. DigitalOcean [Електронний ресурс] // Сайт компанії «DigitalOcean» – 07.12.2025. – Режим доступу до ресурсу: https://www.digitalocean.com/.

19. AWS [Електронний ресурс] // Сайт компанії «AWS» – 21.11.2025. – Режим доступу до ресурсу: https://aws.amazon.com/.

20. Google Cloud [Електронний ресурс] // Сайт компанії «Google» – 25.11.2025. – Режим доступу до ресурсу: https://cloud.google.com/.

21. Microsoft Azure [Електронний ресурс] // Сайт компанії «Microsoft» – 08.11.2025. – Режим доступу до ресурсу: https://azure.microsoft.com/.

22. Hetzner Cloud [Електронний ресурс] // Сайт компанії «Hetzner» – 13.11.2025. – Режим доступу до ресурсу: https://www.hetzner.com/.

23. Open Server Panel [Електронний ресурс] // Сайт компанії «OSPanel» – 12.11.2025. – Режим доступу до ресурсу: https://ospanel.io/.

24. OWASP Top 10 [Електронний ресурс] // Сайт компанії «OWASP» – 09.12.2025. – Режим доступу до ресурсу: https://owasp.org/Top10/2025/.