ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ КІБЕРБЕЗПЕКИ ТА ЗАХИСТУ ІНФОРМАЦІЇ

КАФЕДРА СИСТЕМ ТА ТЕХНОЛОГІЙ КІБЕРБЕЗПЕКИ

КВАЛІФІКАЦІЙНА РОБОТА

на тему:
«Засоби захисту веб-додатків організації»

зі спеціальності *125 Кібербезпека*
(код, найменування спеціальності)

освітньо-професійної програми *Інформаційна та кібернетична безпека*
(назва програми)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело
_____ Ярослава ТВЕРДОХЛІБ
(підпис)

Виконав: здобувач вищої освіти групи БСД-42
ТВЕРДОХЛІБ Ярослава
(прізвище, ім'я)

Керівник асистент кафедри ШУЛІМОВА Дар'я
(науковий ступінь, вчене звання, прізвище, ім'я)

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

Київ 2025

ВСТУП

Актуальність дослідження. Захист веб-додатків в організаціях є надзвичайно важливою складовою інформаційної безпеки. По-перше, в зв'язку з тим що веб-застосунки стають основним інтерфейсом для взаємодії користувачів із сервісами компанії, саме вони дедалі частіше є ціллю кібератак. Веб-додатки обробляють чутливі персональні дані, корпоративну інформацію та інтегруються з внутрішніми системами організації. Злам або експлуатація вразливостей у таких додатках може призвести до витоку даних, порушення роботи бізнес-процесів та значних фінансових втрат.

По-друге, веб-додатки зазвичай мать прямий доступ до внутрішньої бази даних та інформаційних систем організації, створюючи потенційний шлях до проникнення в корпоративну мережу. По-третє, вразливості веб-додатків можуть призвести до компрометації бренду, втрати довіри клієнтів та юридичних наслідків відповідно до законодавства про захист даних.

Згідно зі звітом CyCognito State of Web Application Security Testing 2024 [1], понад 35% опитаних організацій стикаються з серйозними інцидентами безпеки веб-додатків щонайменше раз на тиждень, а 26% ще частіше. Ці дані підкреслюють необхідність впровадження автоматизованих засобів тестування безпеки веб додатків та постійного моніторингу для своєчасного виявлення та усунення вразливостей.

Важливу роль у побудові стратегії захисту веб-додатків відіграють стандарти та ініціативи, зокрема OWASP top 10, які класифікують найбільш критичні ризики. Найбільш поширеними залишаються ін'єкції, неправильна автентифікація, витік чутливих даних, недоліки контролю доступу та неправльна конфігурація безпеки. Ці вразливості можуть призвести до серйозних інцидентів безпеки.

У контексті цієї кваліфікаційної роботи особлива роль приділяється виявленню вразливостей веб-застосунків та аналізу конфігурацій безпеки за допомогою спеціалізованих онлайн-сервісів. Такі рішення, як Security Headers і

SSL Labs, дозволяють швидко оцінити рівень захисту з точки зору політик HTTP-заголовків, SSL/TSL протоколів та інших параметрів що впливають на безпеку.

Об'єкт дослідження – захист веб-додатків організації.

Предмет дослідження – методи та засоби захисту веб-додатків на основі сучасних інструментів сканування та аналіз безпеки.

Мета роботи - аналіз наукових джерел та нормативної документації, вивчення методик виявлення вразливостей, використання онлайн-сервісів безпеки, аналіз результатів практичного сканування, розробка рекомендацій для покращення безпеки веб-додатку.

Наукові завдання:

Дослідити загальні принципи захисту веб-додатків;

Проаналізувати найпоширеніші загрози та вразливості згідно з OWASP top 10;

Провести практичне сканування сайту з використанням інструментів SSL Labs і Security Headers;

Оцініти результат аналізу та визначити шляхи покращення безпеки, формувати рекомендації щодо забезпечення безпеки веб-додатків в організаціях.

Методи дослідження – аналіз наукових джерел та нормативної документації, вивчення методик виявлення вразливостей, використання онлайн-сервісів безпеки, аналіз результатів практичного сканування .

1. ДОСЛІДЖЕННЯ ПРОБЛЕМИ ЗАХИСТУ ВЕБ-ДОДАТКІВ ОРГАНІЗАЦІЇ

1.1. Сутність веб-додатків та їх роль в інформаційній інфраструктурі організацій

Веб-додатки - це програмні системи, що функціонують через веб-браузер і забезпечують інтерактивну взаємодію користувача з серверною частиною. Веб-додатки не потребують встановлення на пристроях користувачів, завдяки цій доступності та гнучкості, веб-додатки стали невід'ємною частиною цифрової інфраструктури організації. Веб-додатки є основним інструментом для надання послуг організаціями різних напрямів діяльності. Галузі, в яких вони застосовуються, наведено в таблиці 1.1.

Таблиця 1.1

Приклади застосування веб-додатків у різних сферах діяльності

Сфера застосування	Приклад
Бізнес	CRM, ERP, аналітика, докуметнообіг
Фінансовий сектор	Інтернет-банкінг, платіжні шлюзи
Медицина	Телемедицина, електронні медичні картки
Освіта	Платформи дистанційного навчання, електронні щоденники
Державне управління	Реєстри, електронні кабінети громадян, е-послуги

Такі додатки є невід'ємною частиною інформаційної інфраструктури організації, яка включає всі технологічні засоби, що забезпечують зберігання, обробку і передачу інформації. Веб-додатки в цій системі часто виступають в якості центральної точки доступу не тільки для клієнтів, а і для співробітників.

У звіті CyCognito State of Web Application Security Testing 2024 [1] зазначено, що великі підприємства експлуатують сотні веб-додатків. Це свідчить про

зростаючу залежність організацій від веб-технологій у своїй повсякденній діяльності.

Однак, попри очевидні переваги, веб-додатки мають одну критичну особливість – вони є публічними точками входу до внутрішніх даних організації. Їх відкритість для зовнішнього доступу роботи їх привабливою ціллю для кібератак. Це свідчить про те що веб-додатки – це не лише засіб надання сервісів, а й один з найуразливіших компонентів корпоративної IT-інфраструктури.

1.2. Актуальність захисту веб-додатків

Актуальність захисту веб-додатків підтверджується численними статистичними даними та аналітичними звітами провідних міжнародних організацій у сфері кібербезпеки. Зокрема, згідно з результатами дослідження компанії CyCognito [1], понад 35% опитаних компаній стикаються з інцидентами безпеки веб-додатків щонайменше раз на тиждень, тоді як 26% повідомляють про такі інциденти ще частіше. Це свідчить про високий рівень вразливості веб-інтерфейсів та недостатність реалізованих заходів безпеки.

Ще одним підтвердженням актуальності цієї теми є рейтинг OWASP Top 10 [2], який оновлюється кожні декілька років, демонструє сталість певних типів вразливостей (наприклад, SQL-ін’єкцій, XSS, неправильного керування доступом), що свідчить про те, що багато організацій досі не впровадили базових стандартів безпеки. Таким чином, існує гостра потреба у впровадженні системних рішень – від безпечної розробки до регулярного тестування і моніторингу безпеки.

Актуальність захисту веб-додатків зростає разом із ускладненням самих додатків. Багато сучасних систем мають розподілену архітектуру, використовують сторонні API, інтеграції з хмарними сервісами, що створює додаткові точки входу для потенційних атак. Це ускладнює контроль і потребує застосування нових підходів до безпеки, таких як Zero Trust, DevSecOps, а також постійного моніторингу поведінки користувачів та підозрілої активності.

Окрему увагу слід приділити HTTPS/TLS-захисту, який є основою безпечної комунікації між клієнтом і сервером. Наявність сертифіката, правильне налаштування протоколів, сучасні шифри – усе це має безпосередній вплив на стійкість веб-додатка до атак типу “man-in-the-middle” (MITM) та перехоплення трафіку. Дослідження демонструють, що частина сайтів досі використовують застарілі версії TLS або взагалі дозволяють небезпечні протоколи, що створює критичні ризики.

Не менш актуальною є проблема використання сторонніх бібліотек і фреймворків, які можуть містити вразливості, відомі як вразливості ланцюга постачання (supply chain vulnerabilities). Оскільки веб-додатки часто збираються на основі великої кількості зовнішніх компонентів, без належного контролю залежностей організація може навіть не знати, що в її додаток уже вбудовано небезпечний код. Яскравим прикладом є інцидент з вразливістю в бібліотеці Log4j у 2021 році, яка широко використовується для ведення логів у веб-додатках і корпоративних системах. Вразливість Log4Shell, що була виявлена у бібліотеці Apache Log4j, дозволяє зловмисникам виконувати віддалений код на вразливих серверах, що робить цю уразливість особливо небезпечною через її простоту експлуатації та широкий спектр потенційних жертв. [3] Інцидент із Log4Shell продемонстрував, наскільки критичною є безпека компонентів ланцюга постачання та необхідність контролю за залежностями у процесах розробки і підтримки веб-додатків.

Не менш важливу роль відіграє людський фактор. Часто користувачі не дотримуються базових рекомендацій щодо безпечного використання веб-сервісів: обирають слабкі паролі, повторно використовують облікові дані на різних платформах, ігнорують двофакторну автентифікацію. Внаслідок цього навіть захищений веб-додаток може стати об'єктом успішної атаки через помилки кінцевих користувачів. Це ще раз підкреслює, що питання безпеки – це не тільки технічна, а й організаційна проблема.

Враховуючи ці факти, питання захисту веб-додатків є надзвичайно важливим для кожної організації. Забезпечення надійного рівня безпеки потребує не тільки

технічних знань, а й розуміння типових загроз, правильного налаштування середовища, використання сучасних інструментів перевірки, а також дотримання стандартів інформаційної безпеки. Це також стосується відповідності законодавству, наприклад, нормам щодо захисту персональних даних (GDPR) або вимогам національного регулювання.

Таким чином, захист веб-додатків – це не лише технічне, але й стратегічне завдання, яке має реалізовуватись комплексно: від етапу розробки до постійного моніторингу та оновлення. Це зумовлює важливість обраної теми та необхідність її дослідження.

1.3. Аналіз найбільш поширених ризиків безпеки

Для аналізу найбільш поширених ризиків розглянемо перелік із 10 найнебезпечніших векторів атаки на веб-додатки, який розробила некомерційна організація Open Web Application Security Project (OWASP), і який отримав назву OWASP top 10 [2]. Цей перелік став ключовим інструментом для компаній, які прагнуть підвищити рівень захищеності своїх програмних рішень в умовах постійно зростаючих кіберзагроз.

Важливість OWASP Top 10 полягає насамперед у його методологічному підході до класифікації вразливостей. Документ представляє структуровану ієрархію найпоширеніших проблем безпеки, які найчастіше стають причиною успішних кібератак на веб-застосунки. При складанні цього рейтингу експерти OWASP враховують декілька критичних факторів: поширеність конкретної вразливості в сучасних веб-додатках, потенційний масштаб негативного впливу при її експлуатації, технічну складність використання вразливості зловмисниками, а також ступінь складності виявлення такої вразливості під час тестування безпеки.

На зображенні нижче показані результати дослідження десять основних ризиків для безпеки веб-застосунків, далі детальніше розглянемо кожен з них.

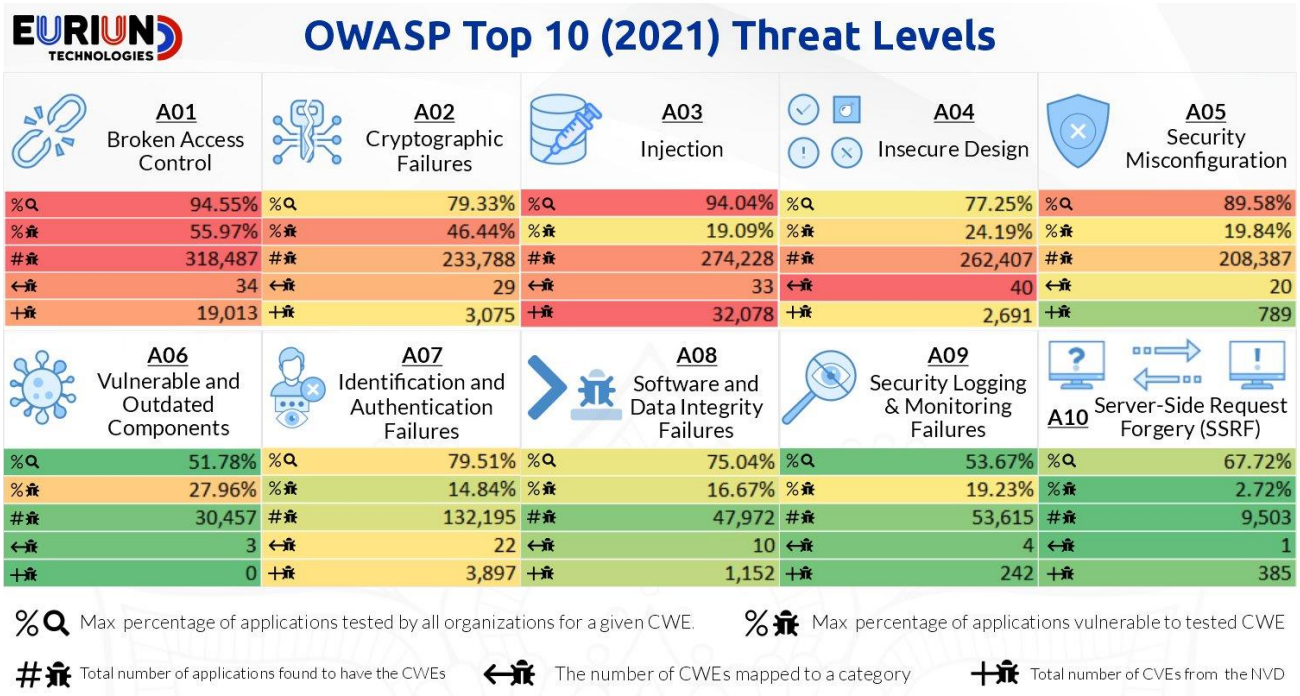

Рис.1.1. Статистика рівнів загроз OWASP top 10 [4]

A01 Broken Access Control (Порушення контролю доступів)

Ця вразливість пов’язана з неправильною реалізацією обмеженого доступу до функцій чи даних. В результаті зловмисники можуть отримати доступ до ресурсів компанії, які не призначені для них, наприклад: переглядати чужі облікові записи, виконувати дії від імені іншого користувача, редагувати конфіденційну інформацію.

A02 Cryptographic Failures (Криптографічні збої)

Загрози з цієї категорії виникають, якщо неправильно використовувати, або взагалі не використовувати шифрування даних. Нехтування цим може призвести до витоку конфіденційної інформації, особливо при передачі даних по мережі. Типовим прикладом цієї вразливості є відсутність HTTPS, зберігання паролів в вільному доступі, використання застарілих алгоритмів.

A03 Injection (Ін’єкції)

Вразливості типу ін’єкцій виникають, коли ненадійні дані надсилають інтерпретатору коду як частина команди або коду. Це дає зловмисникам можливість виконувати довільні команди, отримувати доступ і можливість

змінювати данні. Найвідоміший приклад – SQL-ін’єкції, під час якої зловмисник може виконувати довільні SQL-запити, отримуючи доступ до бази даних. [14]

A04 Insecure Design (Небезпечний дизайн)

В цьому пункті виділені саме недоліки архітектури чи логіки системи безпеки. Це може бути неправильна реалізація автентифікації, відсутність двофакторної ідентифікації або слабка реалізація управляння сесіями. Такі помилки важко виправити без втручання в архітектурні рішення.

A05 Security Misconfiguration (Некоректні налаштування безпеки)

Помилка конфігурації виникає, коли система чи її компоненти налаштовані не правильно. Як приклад можна привести відкриті адміністративні панелі, відкриті порти, дефолтні паролі, відсутність оновлень. Внаслідок цього зловмисники можуть отримати додаткову інформацію про систему чи використати вразливості для проникнення.

A06 Vulnerable and Outdated Components (Вразливості і застарілі компоненти)

Багато веб-додатків залежать від сторонніх бібліотек і фреймворків. Використання застарілих або небезпечних версій може привести до компрометації всієї системи. Важливо регулярно оновлювати програмні залежності, бібліотеки і компоненти та слідкувати за новими вразливостями в використовуваному ПЗ.

A07 Identification and Authentication Failures (Помилки ідентифікації і автентифікації)

Ці вразливості стосуються слабких місць в системі входу користувачів. Наприклад: передбачвані паролі, відсутність обмеження на кількість спроб входу, вразливості в механізмі відновлення паролю. Це полегшує зловмисникам підбір паролю і захоплення акаунтів.

A08 Software and Data Integrity Failures (Порушення цілісності програмного забезпечення і даних)

Помилки в забезпечення цілісності програмного коду та даних можуть привеста до виконання неавторизованого чи шкідливого коду. Типовим прикладом буде відсутність перевірки цифрових підписів оновлень чи використання непідтверджених зовнішніх скриптів (CDN).

A09 Security Logging and Monitoring Failures (Збої реєстрації та моніторингу безпеки)

Без ефективного журналювання та спостереження за системою адміністратори не зможуть своєчасно виявити або розслідувати атаки. Як наслідок це дає зловмисникам можливість діяти непомітно. Хороша практика – вести централізоване логування та використовувати системи виявлення вторгнень.

A10 Server-Side Request Forgery – SSRF (Підробка запитів на стороні сервера)

SSRF-атаки виникають, коли веб-додатки змушують надсилати запити від свого імені до внутрішніх або зовнішніх ресурсів. Це дає зловмисникам можливість отримати доступ до внутрішніх мереж, метаданих хмарної інфраструктури чи виконати несанкціоновані дії від імені серверу.

Враховуючи високий рівень поширеності та різноманіття вразливостей веб-додатків, визначених у OWASP Top 10, стає очевидним, що забезпечення безпеки організаційних інформаційних систем вимагає комплексного підходу. Уразливості, такі як ін’єкції, міжсайтове виконання скриптів (XSS) або порушення автентифікації, є лише частиною загальної проблеми. Для ефективного захисту необхідно інтегрувати не лише заходи на рівні програмного забезпечення, а й впроваджувати надійні механізми захисту на мережевому рівні та в налаштуваннях веб-сервера.

Одним із фундаментальних засобів забезпечення безпеки інформації при передачі даних між клієнтом і сервером є протоколи шифрування SSL/TLS. Використання цих протоколів дозволяє гарантувати конфіденційність, цілісність і автентичність інформації, що є необхідними у контексті захисту веб-додатків організацій, особливо тих, що працюють із конфіденційними або персональними даними. Відсутність або неналежна реалізація SSL/TLS значно підвищує ризики атак типу «людина посередині», що може призвести до витоку або модифікації інформації.

В той же час, надійне шифрування – це лише частина безпекової архітектури. Додатково до протоколів SSL/TLS критично важливо правильно налаштувати HTTP-заголовки безпеки, які регламентують поведінку браузера та захищають

користувача від класу атак, пов’язаних із взаємодією з веб-додатком. Заголовки, такі як Content-Security-Policy, X-Frame-Options, X-Content-Type-Options та Strict-Transport-Security, забезпечують механізми запобігання міжсайтовому скриптингу, кліковому грабежу, MIME-сніффінгу та посиленню використання захищеного з’єднання.

Таким чином можна прийти до висновку, що неможливо уявити захист веб-додатків організацій без комплексного підходу, що поєднує виправлення вразливостей на рівні програмного коду (відповідно до рекомендацій OWASP Top 10) із впровадженням криптографічних протоколів SSL/TLS і системи HTTP-заголовків безпеки. Такий багаторівневий підхід забезпечує надійний захист переданих даних, контроль поведінки клієнтських браузерів і значно знижує ймовірність успішних атак. Отже, реалізація цих механізмів є необхідною умовою для забезпечення інформаційної безпеки веб-додатків у сучасних організаціях.

1.4. Вимоги стандартів та законодавства у сфері захисту веб-додатків

У процесі забезпечення безпеки веб-додатків організації недостатньо лише технічного виконання рекомендацій OWASP або налаштування TLS/SSL-протоколів. Надзвичайно важливим аспектом є відповідність чинному законодавству та міжнародним стандартам у сфері інформаційної безпеки. З одного боку, такі вимоги спрямовані на підвищення загального рівня кіберзахисту, з іншого – на захист прав суб’єктів персональних даних і клієнтів. Такі нормативні документи не лише визначають мінімальні вимоги до захисту інформації, а й забезпечують правові підстави для діяльності організації в онлайн-середовищі. Недотримання вимог законодавства або ігнорування стандартів може призвести до юридичних наслідків, фінансових санкцій, втрати довіри з боку клієнтів і партнерів.

У вітчизняному правовому полі головним нормативним актом, що регламентує захист інформації в інформаційно-телекомунікаційних системах, є Закон України «Про захист інформації в інформаційно-телекомунікаційних

системах» № 80/94-ВР від 05.07.1994 року [7]. Закон встановлює основні принципи захисту інформації, у тому числі вимоги до забезпечення конфіденційності, цілісності та доступності даних. Веб-додатки, як елемент ІКТ-системи організації, повинні бути захищені технічними й програмними засобами, зокрема через автентифікацію користувачів, контроль доступу, криптографічний захист даних, реєстрацію подій доступу та інших дій у системі.

Окремі вимоги до суб'єктів фінансового сектора викладені у Постанові Національного банку України № 95 від 28.09.2017 р. «Про затвердження вимог з кіберзахисту» [8]. Згідно з цим документом, об'єкти критичної інфраструктури зобов'язані впроваджувати Web Application Firewall (WAF), проводити регулярне тестування безпеки (включаючи сканування вразливостей та penetration testing), а також забезпечувати криптографічний захист передачі даних із використанням протоколів не нижче TLS 1.2.

Міжнародна організація зі стандартизації (ISO) спільно з Міжнародною електротехнічною комісією (IEC) розробила стандарт ISO/IEC 27001:2022 «Інформаційні технології. Методи забезпечення безпеки. Системи управління інформаційною безпекою. Вимоги» [9]. Цей стандарт визначає вимоги до впровадження, підтримки та постійного вдосконалення системи управління інформаційною безпекою (СУІБ).

У контексті захисту веб-додатків ISO/IEC 27001 передбачає, зокрема:

- проведення регулярної оцінки ризиків і реалізацію відповідних заходів контролю;
- документування політик безпеки;
- впровадження процесів управління вразливостями;
- аудит безпеки веб-систем;
- навчання персоналу з питань ІБ.

Відповідно до наведених вимог, впровадження стандарту ISO/IEC 27001 [9] сприяє комплексному підходу до управління інформаційною безпекою та підвищує загальний рівень захисту веб-додатків в організації.

Також слід відзначити рекомендації Національного інституту стандартів і технологій США (NIST), зокрема документ NIST SP 800-53 Rev.5 [10], який визначає мінімальні вимоги до забезпечення безпеки інформаційних та інформаційно-телекомунікаційних систем. Документ охоплює понад 20 функціональних доменів безпеки, включаючи автентифікацію, контроль доступу, обробку інцидентів, а також захист клієнтської взаємодії у веб-середовищі. У контексті захисту веб-додатків особливу увагу приділяється правильній реалізації криптографічного захисту, автентифікації користувачів, а також процедур виявлення та реагування на вразливості.

До рекомендованих документів, що містять практичні орієнтири для забезпечення безпеки веб-додатків, належить OWASP Top 10 – перелік найбільш критичних категорій вразливостей, визначений міжнародною спільнотою Open Web Application Security Project (OWASP) [2]. Незважаючи на рекомендаційний характер, OWASP Top 10 часто слугує основою для формування політик безпечної розробки, проведення аудиту програмного коду, а також створення тестових сценаріїв при виконанні етичного хакінгу.

Для організацій, які обробляють платіжні транзакції, обов’язковим є дотримання вимог стандарту PCI DSS (Payment Card Industry Data Security Standard), зокрема щодо шифрування даних карткових транзакцій, верифікації користувачів та регулярного тестування веб-ресурсів на наявність вразливостей.

Узагальнюючи результати аналізу нормативно-правових документів та міжнародних стандартів, можна стверджувати, що захист веб-додатків організації становить собою комплексну функцію, яка потребує одночасної реалізації правових, організаційних і технічних заходів. Веб-додатки, як критично важлива складова інформаційної інфраструктури, є об’єктами підвищеного ризику з точки зору кібербезпеки, а тому підлягають суворому регуляторному контролю.

З метою забезпечення відповідності веб-додатків сучасним вимогам інформаційної безпеки, організація повинна:

- імплементувати систему управління інформаційною безпекою на основі вимог стандарту ISO/IEC 27001, що дозволяє впорядкувати процеси

виявлення, аналізу та мінімізації ризиків, пов’язаних із функціонуванням веб-додатків;

- забезпечити обробку персональних даних користувачів веб-додатків відповідно до положень Загального регламенту захисту даних (GDPR), зокрема шляхом застосування механізмів криптографічного захисту, політик доступу та процедур звітності;
- враховувати національні вимоги, зокрема положення Закону України «Про захист інформації в інформаційно-телекомунікаційних системах» та вимоги Постанови НБУ № 95 у випадку обробки фінансової або банківської інформації через веб-додатки;
- використовувати галузеві методичні напрацювання (OWASP Top 10, NIST SP 800-53, PCI DSS), які забезпечують стандартизовані підходи до виявлення і нейтралізації найпоширеніших вразливостей у веб-середовищі.

Недотримання зазначених регламентів і стандартів при розробці, розгортанні або експлуатації веб-додатків може спричинити суттєві негативні наслідки для організації. Серед таких наслідків варто виокремити втрату конфіденційної інформації, зниження рівня довіри з боку користувачів, застосування санкцій з боку наглядових органів, а також ризики припинення партнерських чи комерційних відносин з контрагентами. У зв’язку з цим, відповідність законодавчим та галузевим вимогам має розглядатися як ключовий пріоритет у контексті побудови стійкої системи безпеки веб-додатків організації.

2. АНАЛІЗ СТАНУ БЕЗПЕКИ ВЕБ-ДОДАТКУ ОРГАНІЗАЦІЇ ЗАСОБАМИ SSL LABS ТА SECURITYHEADERS.COM

2.1. Вибір тестового веб-додатка для проведення аналізу

Сайт https://demo.testfire.net/ – це публічно доступний демонстраційний веб-додаток, створений компанією IBM (раніше Watchfire) з метою тестування та демонстрації вразливостей веб-застосунків. Він імітує функціонал інтернет-банкінгу й спеціально містить різноманітні поширені уразливості, що робить його ідеальним середовищем для навчання, тестування засобів аналізу безпеки та оцінки ефективності захисних заходів.

Demo.testfire.net реалізує типовий веб-додаток фінансової установи, з функціоналом, що включає:

- авторизацію користувачів;
- роботу з особистими фінансовими даними;
- перегляд транзакцій;
- переказ коштів;
- зміну пароля;
- запити на отримання кредиту тощо.

Цей сайт побудований із типовими технологіями веб-розробки, а його структура і поведінка подібні до реальних застосунків банків чи інших організацій, які працюють з конфіденційною інформацією.

У межах кваліфікаційної роботи, присвяченої захисту веб-додатків організації, вибір саме цього ресурсу має кілька вагомих обґрунтувань:

1) Легальність і безпечність тестування

Demo.testfire.net призначений саме для етичного хакінгу, аналізу безпеки й демонстрації атак. Його використання не порушує законів чи політик компаній, на відміну від спроб тестування реальних сайтів без дозволу. Це дозволяє

безперешкодно застосовувати сервіси сканування, такі як SSL Labs та SecurityHeaders.com, у рамках практичної частини кваліфікаційної роботи.

2) Репрезентативність

Інтерфейс, логіка роботи та функціональність сайту схожі на ті, що використовуються у веб-застосунках банків, страхових компаній або інших організацій, які працюють із чутливою інформацією. Це дозволяє зробити висновки, які будуть релевантними для реального корпоративного середовища.

3) Наявність типових вразливостей

Завдяки спеціально вбудованим недолікам, demo.testfire.net чудово підходить для моделювання атак, перевірки механізмів захисту та аналізу відповідності сайту сучасним стандартам безпеки веб-додатків. Це дозволяє на практиці продемонструвати, як відсутність належного шифрування чи захисних заголовків може становити загрозу для користувачів і організації.

4) Відкрита доступність і стабільна робота

Сайт доступний без авторизації, реєстрації або ліцензій. Він стабільно працює, не блокує мережевий трафік і підтримує більшість інструментів оцінювання безпеки, зокрема SSL Labs і SecurityHeaders.com.

Оскільки тематика роботи стосується методів і засобів захисту веб-додатків, вибір веб-додатку з типовими помилками та відсутністю належного рівня захисту є вдалим рішенням для практичного дослідження. Такий об’єкт дозволяє наочно продемонструвати критичну роль, яку відіграють протоколи SSL/TLS у забезпеченні конфіденційності даних під час передавання. Крім того, аналіз дає змогу побачити, як саме недоліки або неправильна конфігурація HTTP-заголовків безпеки можуть створити можливості для експлуатації вразливостей. Проведене тестування також дає змогу чітко визначити, які налаштування та технології необхідні для досягнення відповідності сучасним вимогам безпеки, включно з рекомендаціями OWASP Top 10. Таким чином, обраний підхід дозволяє не лише провести прикладний аналіз, але й сформулювати дієві рекомендації для організацій, зацікавлених у підвищенні захищеності своїх веб-застосунків.

2.2. Огляд протоколів SSL/TLS та їх роль у захисті веб-застосунків

Забезпечення конфіденційності, цілісності та автентичності переданих даних у веб-додатках неможливе без використання криптографічних протоколів захищеного обміну інформацією. Найважливішими з них у контексті веб-застосунків є протоколи SSL (Secure Sockets Layer) та його наступник TLS (Transport Layer Security). Ці протоколи є стандартом де-факто для шифрування трафіку в мережі Інтернет і використовуються практично в усіх веб-застосунках, що обробляють конфіденційну інформацію користувачів, зокрема логіни, паролі, платіжні реквізити, персональні дані тощо.

Безпечна передача даних між користувачем і веб-додатком є одним із ключових елементів побудови надійної інформаційної інфраструкури. З огляду на поширеність кіберзагроз, таких як перехоплення трафіку, підміна даних та атаки типу «людина посередині» (Man-in-the-Middle), особливу роль відіграють протоколи, що забезпечують захищений канал зв’язку –зокрема SSL (Secure Sockets Layer) та TLS (Transport Layer Security). Протоколи SSL/TLS виконують низку критично важливих функцій:

a) Шифрування даних – забезпечує, щоб передана інформація не могла бути прочитана сторонніми особами.

b) Аутентифікація сервера (та клієнта, за потреби) – з використанням цифрових сертифікатів, виданих довіреним центром сертифікації (CA).

c) Перевірка цілісності – гарантує, що дані не були змінені під час передачі (застосовуються MAC-алгоритми або AEAD-режими).

d) Установлення сеансового ключа – в процесі рукостискання (handshake) сторони погоджують алгоритми та генерують спільний ключ.

Робота протоколу TLS базується на складному процесі, відомому як TLS Handshake – рукостискання між клієнтом і сервером, під час якого сторони домовляються про криптографічні алгоритми, автентифікують один одного (зазвичай за допомогою цифрового сертифіката сервера) та обмінюються ключами шифрування. Як зазначено в технічному огляді Cloudflare, процес TLS Handshake є

критичним для встановлення зашифрованого каналу зв'язку та забезпечення конфіденційності даних, що передаються між сторонами [12]. У більшості сучасних випадків застосовуються комбінації алгоритмів симетричного шифрування (AES, ChaCha20), обчислення хеш-функцій (SHA-2, SHA-3) та обміну ключами (ECDHE, DHE). Важливим кроком у цьому процесі є перевірка цифрового сертифіката сервера, що засвідчує його приналежність до певної організації та справжність доменного імені. Сертифікати видаються довіреними центрами сертифікації (Certificate Authorities), які перевіряють запити на сертифікацію.

Безпека TLS безпосередньо залежить від правильності його конфігурації. Застарілі версії протоколу (TLS 1.0, TLS 1.1) більше не вважаються безпечними, а їх використання може дозволити зловмисникам здійснити атаки на трафік (наприклад, BEAST, CRIME або RC4-базовані атаки). Відповідно, рекомендується використовувати лише TLS 1.2 або, ще краще, TLS 1.3 – найновішу версію, яка включає низку покращень: скорочений handshake, обов'язкове шифрування forward secrecy, усунення вразливих алгоритмів тощо.

Особливу роль у забезпеченні надійної реалізації протоколу відіграє набір підтримуваних шифрів (cipher suites). Це комбінації криптографічних алгоритмів, які використовуються у TLS-сеансі. Слабкі cipher suites, зокрема ті, що використовують алгоритм RC4 або короткі ключі (менше 128 біт), не повинні бути дозволеними в сучасних конфігураціях. Натомість варто використовувати лише ті, що гарантують forward secrecy, тобто можливість зберігати конфіденційність навіть у разі компрометації приватного ключа в майбутньому.

Не менш критичною є процедура оновлення SSL/TLS-сертифікатів. Сертифікати мають обмежений термін дії, зазвичай від 90 днів (у випадку Let's Encrypt) до 1–2 років. Використання прострочених або самопідписаних сертифікатів значно знижує довіру до ресурсу та може призвести до втрати клієнтів. З цієї причини все частіше використовуються автоматизовані рішення для генерації та оновлення сертифікатів – наприклад, Certbot або ACME-клієнти.

Використання SSL/TLS у веб-додатках дозволяє реалізувати базові принципи інформаційної безпеки – конфіденційність, цілісність і доступність даних, за

рахунок надійного шифрування трафіку, автентифікації сторін та верифікації переданої інформації. У корпоративному середовищі ці протоколи є не просто бажаним елементом захисту, а вимогою з боку галузевих стандарті та регуляторів, таких як ISO/IEC 27001, NIST та GDPR.

Отже, зважаючи на високі ризики, пов’язані з незахищеними або некоректно налаштованими з’єднаннями, впровадження та правильна конфігурація SSL/TLS є критично важливими завданнями для будь-якої організації, що використовує веб-додатки для обробки чутливої інформації чи взаємодії з користувачами.

2.3 Аналіз безпеки SSL/TLS за допомогою SSL Labs

Захист каналу передачі даних між клієнтом і сервером є одним із ключових аспектів безпеки веб-додатків. Застосування криптографічних протоколів SSL/TLS забезпечує конфіденційність та цілісність даних, що передаються на сайті. Проте лише наявність TLS не гарантує належного рівню безпеки – важливими є параметри конфігурацій версії протоколів, підтримувані шрифти, наявність захисту від атак та інші аспекти. Для оцінки цих параметрів можна використати інструмент від SSL Labs – SSL Server Test [5], який розробила компанія Qualys.

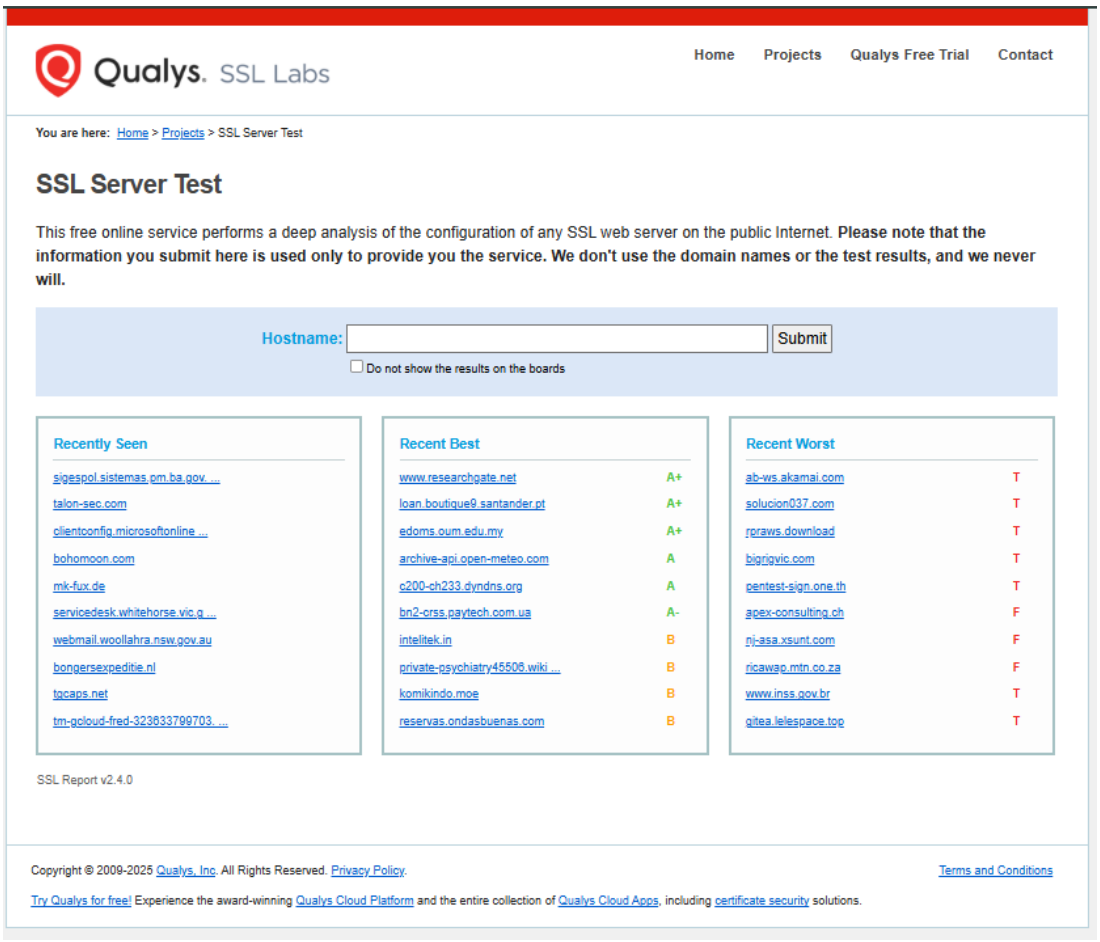

Рис. 2.1. Інтерфейс SSL Labs

Сервіс SSL Labs дозволяє протестувати публічно доступний веб-сервер на предмет коректності SSL/TSL. Він проводить глибокий аналіз і надає деталізований звіт про:

- Підтримувані версії SSL і TSL;
- Список шифрувальних алгоритмів (cipher suites) для кожної доступної версії протоколу;
- Наявність сертифікату його коректність, довіру та термін дії;
- Захист від відомих атак (Heartbleed, ROBOT, LOGJAM тощо);
- Підтримку функції безпеки, таких як HTTP Strict Transport Security (HSTS);
- Оцінку безпечного повторного використання сесій (Session Resumplion);

- Загальний рейтинг безпеки від A+ до F або T (для сайтів з критичними помилками)

Для аналізу конфігурації TLS-протоколів, сертифікатів та криптографічної стійкості веб-застосунку було використано сервіс SSL Labs SSL Server Test. Об’єктом дослідження став демонстраційний сайт demo.testfire.net [6], що широко використовується в навчальних та тестових цілях для перевірки вразливостей та налаштувань безпеки.

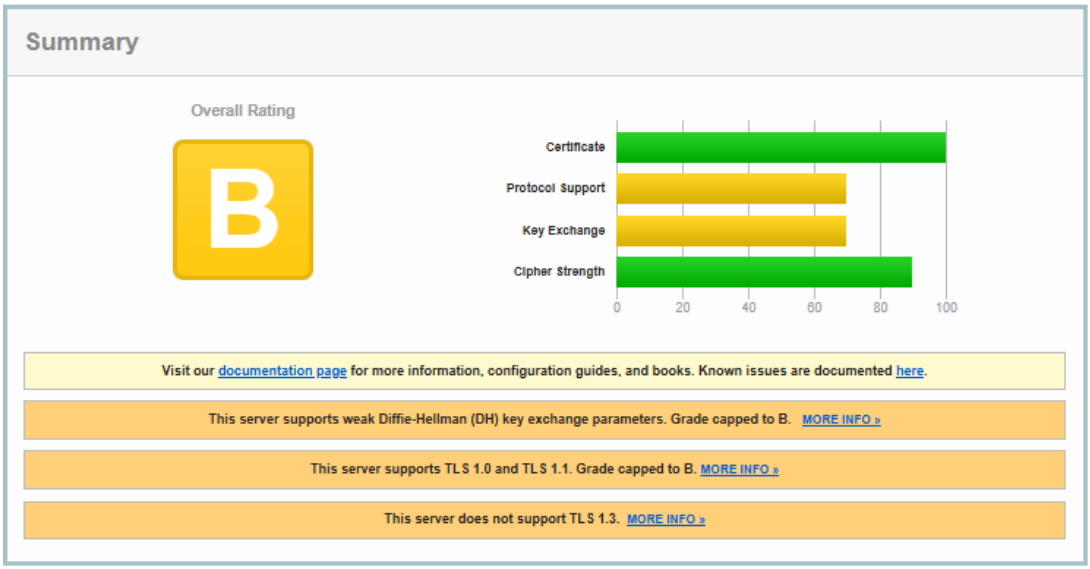

Рис. 2.2. Загальна оцінка після тестування за допомогою SSL Labs

За результатами сканування ресурс отримав загальну оцінку B, що свідчить про наявність ряду важливих недоліків у реалізації захисту TLS-з’єднання. Незважаючи на те, що сайт підтримує базові механізми шифрування, його конфігурація не відповідає актуальним критеріям безпеки. Оцінка була знижена з таких основних причин:

По-перше, виявлено підтримку застарілих криптографічних протоколів TLS 1.0 та TLS 1.1, які вже вважаються небезпечними та не рекомендованими до використання. Ці протоколи мають відомі вразливості, зокрема до атак типу BEAST (Browser Exploit Against SSL/TLS), а також не забезпечують належного рівня криптографічного захисту. Основні сучасні браузери та операційні системи вже відмовилися від їх підтримки, тому їх наявність у конфігурації веб-сервера

негативно впливає як на оцінку безпеки, так і на практичну захищеність користувачів.

По-друге, сервер використовує слабкі параметри обміну ключами, зокрема алгоритм Diffie-Hellman з довжиною ключа лише 1024 біти. Такий рівень криптографічної стійкості є недостатнім за сучасними стандартами. Для надійного захисту даних рекомендується використовувати щонайменше 2048-бітні ключі, або ж перейти на більш ефективні механізми, зокрема на основі еліптичних кривих (ECDHE), які забезпечують вищу безпеку при меншому обсязі обчислень. Наявність слабких параметрів обміну ключами значно спрощує проведення атак типу "forward secrecy compromise" у випадку компрометації сесійного ключа.

По-третє, відсутня підтримка сучасного протоколу TLS 1.3, який наразі є рекомендованим стандартом для забезпечення безпечного шифрування. TLS 1.3 суттєво підвищує рівень безпеки завдяки усуненню застарілих функцій, спрощеному процесу встановлення з’єднання, обов’язковому використанню захищених шифрів та зниженій латентності. Його впровадження не лише підвищує стійкість до атак, але й покращує продуктивність з’єднання. Відсутність TLS 1.3 вказує на застаріле або неналежно оновлене програмне забезпечення на сервері, що також негативно впливає на загальну оцінку безпеки.

В результаті тестування було виявлено як позитивні, так і проблемні аспекти реалізації криптографічного захисту. Основний сертифікат, яким підписано TLS-з’єднання, має такі характеристики:

Загальне ім’я (Common Name, CN): demo.testfire.net

Алгоритм: RSA з довжиною ключа 2048 біт; підписано за допомогою SHA256withRSA

Період дії: з 3 червня 2024 року до 4 липня 2025 року

Центр сертифікації (CA): Sectigo RSA Domain Validation Secure Server CA

Сертифікат є чинним, не відкликаним, та розпізнається основними браузерами, що свідчить про коректну реалізацію базових механізмів автентифікації сервера (рис. 2.3).

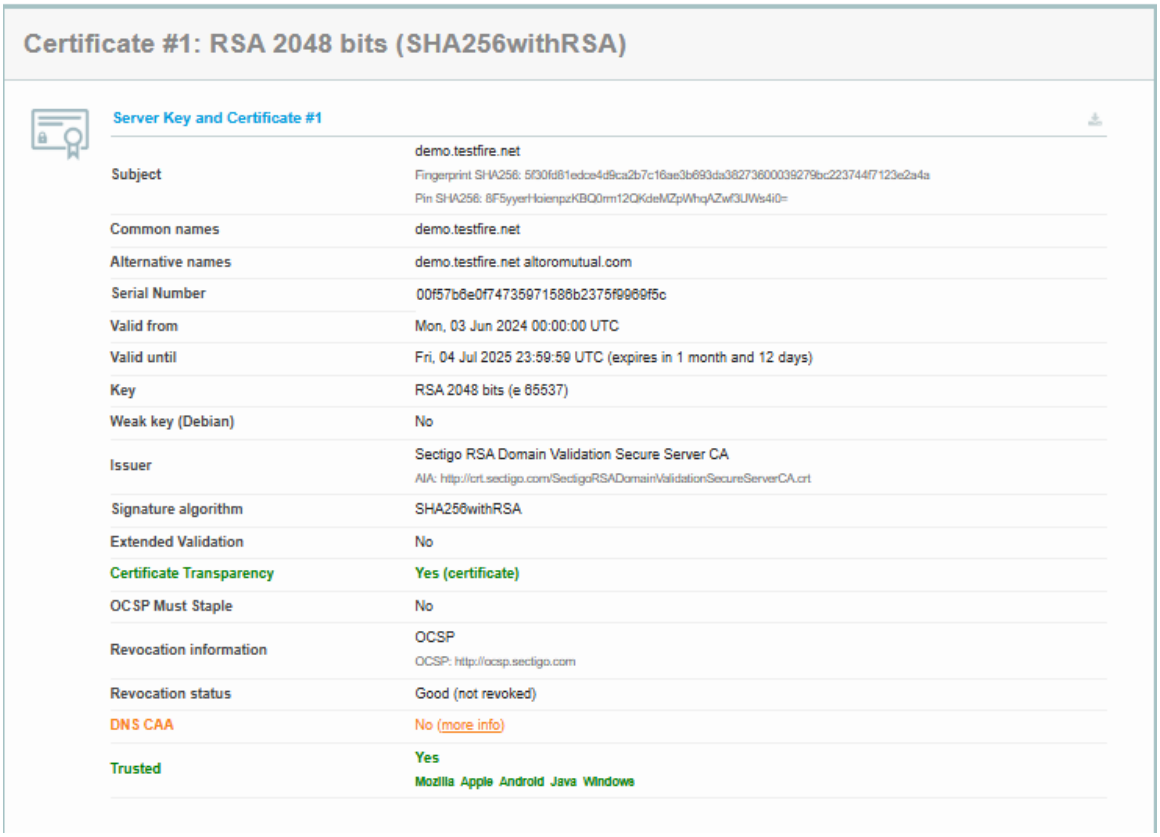

Рис. 2.3 Серверний сертифікат, отриманий за допомогою сервісу SSL Labs

Водночас виявлено кілька недоліків, які знижують загальний рівень довіри:

По-перше, відсутня підтримка розширеної перевірки (Extended Validation, EV), яка надає користувачеві візуальне підтвердження автентичності сайту в браузері (наприклад, зелена адресна стрічка або відображення назви організації). Наявність EV-сертифіката особливо важлива для банківських, фінансових та державних порталів, де ризик фішингових атак надзвичайно високий.

По-друге, не реалізовано механізм OCSP Must-Staple, який дозволяє браузеру запитувати актуальний статус відкликання сертифіката безпосередньо з сервера під час кожного запиту. У разі його відсутності браузер покладається на зовнішній OCSP-сервер, доступність якого не гарантована. Це може призвести до атаки типу "людина посередині" (MITM), коли зловмисник перехоплює з'єднання та використовує відкликаний або підроблений сертифікат, якщо перевірка відкликання не спрацювала.

По-третє, сертифікаційний ланцюг складається з чотирьох сертифікатів, зокрема одного кореневого (self-signed). Було виявлено конфігураційну проблему типу "Contains anchor", що зображена на рис. 2.4, яка полягає в тому, що сервер включає root-сертифікат у ланцюг передачі. Така практика не рекомендована, оскільки root-сертифікати вже містяться у сховищі довіри клієнтських систем. Повторне їх передавання лише створює надмірне навантаження на мережу без суттєвої користі. Це може вказувати на некоректну конфігурацію або використання застарілого середовища.

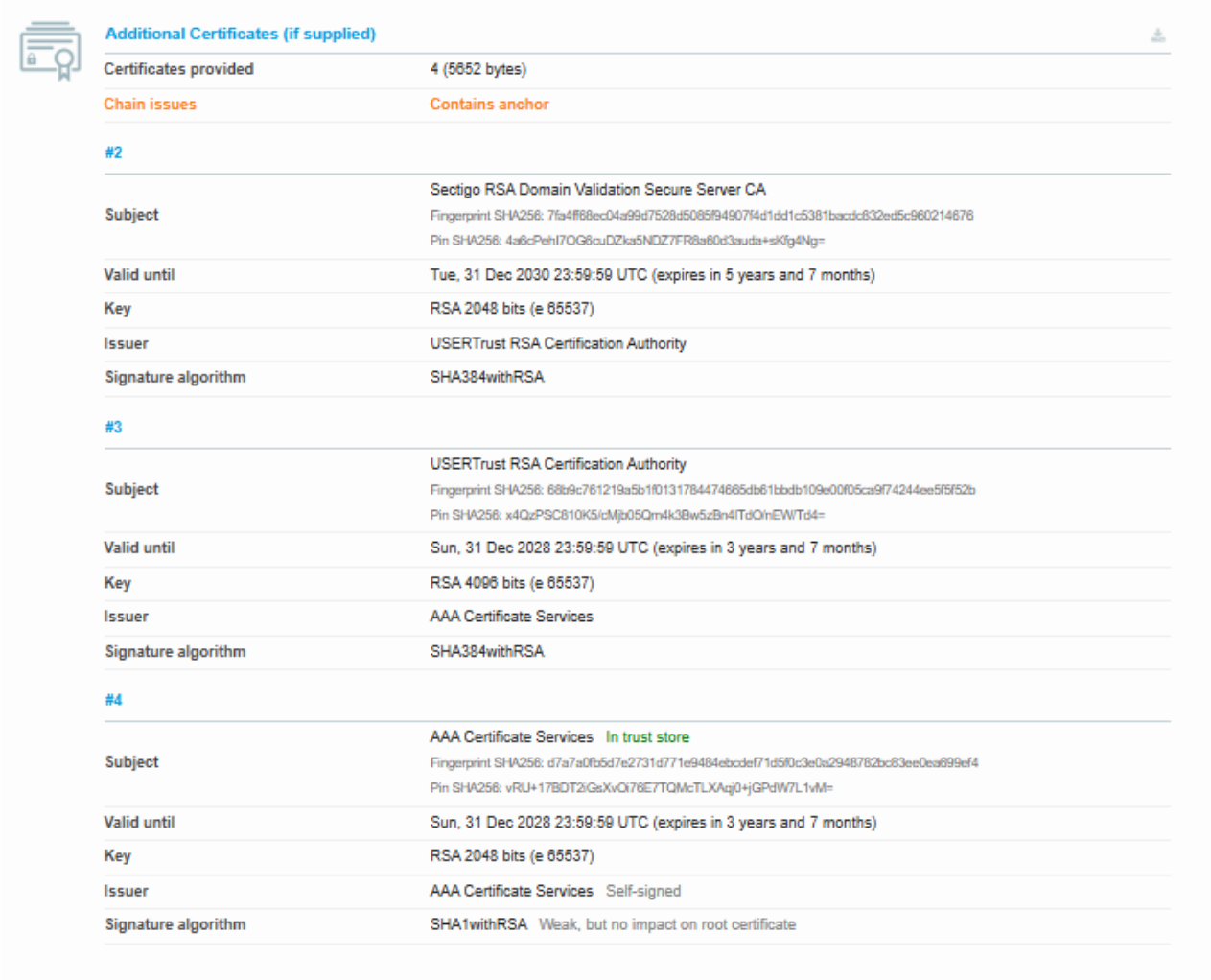

Рис. 2.4. Ланцюжок додаткових сертифікатів TLS із позначенням конфігураційної помилки

Крім того, Станом на час перевірки сервер підтримує такі версії протоколу TLS:

- ✓ TLS 1.0 – підтримується (застарілий)
- ✓ TLS 1.1 – підтримується (застарілий)
- ✓ TLS 1.2 – підтримується (рекомендовано)
- ☒ TLS 1.3 – не підтримується

Підтримка застарілих протоколів TLS 1.0 та TLS 1.1 становить значну загрозу для інформаційної безпеки. Ці версії протоколу давно визнані застарілими, оскільки містять низку добре задокументованих криптографічних вразливостей, зокрема атаку типу BEAST (Browser Exploit Against SSL/TLS), а також не підтримують низку сучасних механізмів захисту, таких як AEAD-шифрування (Authenticated Encryption with Associated Data). Застосування цих протоколів у виробничому середовищі створює потенційні ризики для цілісності, конфіденційності та автентичності даних, які передаються через зашифроване з'єднання. Крім того, їх використання негативно впливає на загальну оцінку ресурсу з точки зору відповідності сучасним стандартам безпеки.

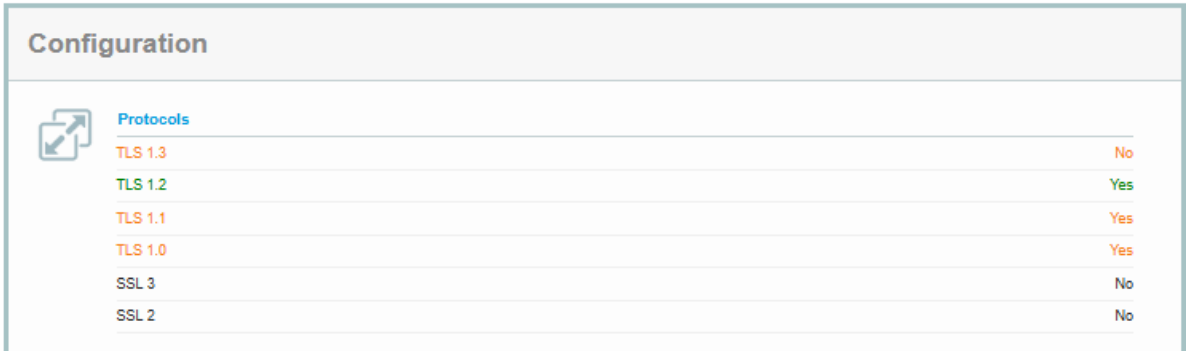

Рис. 2.5 Підтримувані протоколи SSL/TLS

Відсутність підтримки TLS 1.3 також свідчить про недостатній рівень оновлення програмного забезпечення сервера або відсутність належної уваги до сучасних вимог кібербезпеки. Протокол TLS 1.3 є актуальним галузевим стандартом, рекомендованим провідними організаціями, такими як IETF і OWASP. Його впровадження значно підвищує рівень безпеки з'єднань за рахунок усунення

низки вразливих та застарілих механізмів, притаманних попереднім версіям. Зокрема, TLS 1.3 виключає підтримку слабких шифрів, застарілих алгоритмів хешування і уразливих схем обміну ключами. Також він забезпечує поліпшену продуктивність завдяки спрощеному процесу встановлення з'єднання та реалізує обов'язкову підтримку forward secrecy – механізму, що унеможливлює розшифрування перехопленого трафіку навіть у разі компрометації довготривалих ключів.

Під час аналізу конфігурації TLS-з'єднання було виявлено, що сервер підтримує широкий набір шифрувальних пакетів (cipher suites), серед яких присутні як сучасні та рекомендовані до використання варіанти, так і застарілі або потенційно небезпечні, результат аналізу зображено на Рис.2.6.

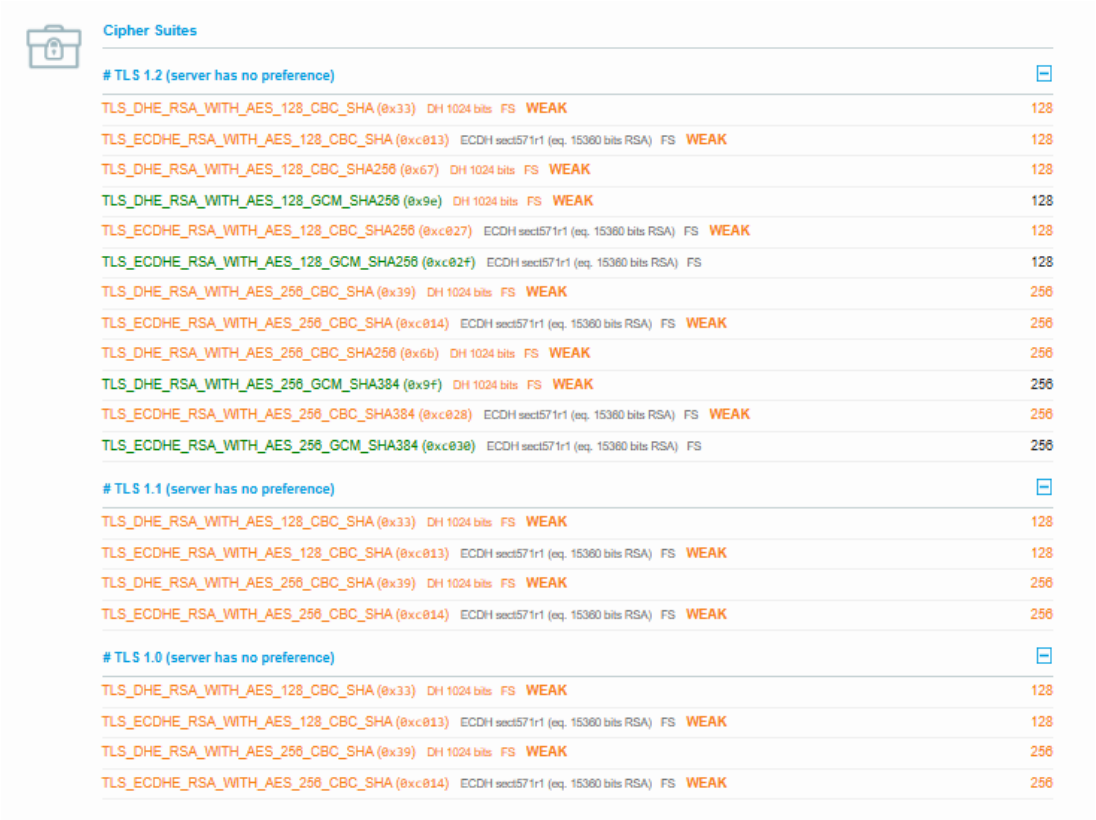

Рис. 2.6 Перелік шифрів TLS із виявленими слабкими алгоритмами

Серед рекомендованих шифрів використовуються такі алгоритми, як TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 та TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384. Ці пакети відповідають сучасним вимогам до

інформаційної безпеки, оскільки реалізують шифрування з автентифікацією (GCM-режим) та використовують еліптичну криптографію (ECDHE), яка забезпечує forward secrecy – тобто захист трафіку навіть у випадку компрометації довготривалих ключів.

Утім, одночасно з цим сервер дозволяє використання слабших шифрів, таких як TLS_DHE_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_RSA_WITH_AES_256_CBC_SHA і TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA. Ці шифрувальні пакети мають низку серйозних недоліків:

1) Використання алгоритму SHA-1, який більше не вважається криптостійким. З моменту виявлення ефективних колізій для SHA-1, даний алгоритм не рекомендований для використання в будь-яких криптографічних процесах, пов’язаних з автентифікацією чи хешуванням. Відомі атаки, такі як SHAttered, підтверджують практичну можливість компрометації безпеки при його використанні.

2) Застосування CBC (Cipher Block Chaining) режиму шифрування є небажаним через можливість реалізації атак типу padding oracle. Такі атаки дозволяють зловмиснику поступово розшифровувати передані дані, скориставшись особливостями обробки помилок при розпаковуванні блоків CBC. На практиці ці атаки були продемонстровані проти TLS 1.0 і 1.1, і їх складність значно нижча, ніж для сучасних режимів шифрування (наприклад, GCM або ChaCha20-Poly1305).

3) Довжина параметрів Diffie-Hellman, яка у наведених слабких пакетах становить лише 1024 біти, є вразливою до дешифрування при використанні сучасних обчислювальних ресурсів. Поточні рекомендації, зокрема від NIST, вимагають довжину не менше 2048 біт. Відомі випадки, коли ключі довжиною 1024 біти зламуються за допомогою попередньо обчислених параметрів та кластеризованих обчислювальних систем.

Ще однією проблемною особливістю є те, що сервер не примушує клієнта використовувати конкретні пріоритетні шифри, а дозволяє клієнтській стороні самостійно обирати шифрувальний пакет. Така поведінка суперечить

загальновизнаним практикам побудови безпечної TLS-конфігурації, оскільки створює можливість для даунгрейд-атак, при яких клієнт може нав’язати менш захищений шифр. Щоб уникнути цього, сервер повинен самостійно встановлювати пріоритети шифрування, гарантуючи застосування найсильніших доступних алгоритмів незалежно від налаштувань клієнта.

Механізм обміну ключами є критично важливим етапом встановлення безпечного TLS-з’єднання. Аналіз показав, що сервер використовує параметри алгоритму обміну ключами Diffie-Hellman (DHE) довжиною лише 1024 біти. Цей рівень криптографічної стійкості є неприйнятним у сучасних умовах, оскільки він піддається атакам із застосуванням обчислювальних потужностей, доступних не лише державним структурам, а й добре ресурсованим приватним організаціям.

Сервер демонструє високий рівень сумісності з більшістю сучасних клієнтських рішень, включаючи браузери Google Chrome, Mozilla Firefox, Microsoft Edge, Safari, а також мобільні платформи. Це означає, що користувачі, які використовують актуальне програмне забезпечення, з великою ймовірністю зможуть встановити захищене з’єднання без додаткових проблем.

Водночас виникають складнощі при роботі зі старими клієнтами, такими як Internet Explorer 8 або Java 6u45. У таких випадках можливе виникнення кількох проблем: або взагалі не вдається встановити TLS-з’єднання, або ж встановлюється сесія із використанням застарілих та небезпечних протоколів (наприклад, TLS 1.0) або слабких шифрів. Це створює небажаний компроміс між підтримкою зворотної сумісності та забезпеченням безпеки.

У сучасних умовах доцільно орієнтуватися на безпеку, навіть якщо це потребує відмови від підтримки застарілого ПЗ. Враховуючи високу частку оновлених браузерів серед користувачів, відключення підтримки небезпечних протоколів і шифрів не створить значних труднощів для більшості цільової аудиторії, але суттєво підвищить загальний рівень безпеки веб-додатку.

Проведений аналіз параметрів TLS-з’єднання веб-ресурсу https://demo.testfire.net засвідчив наявність низки критичних недоліків, що негативно впливають на рівень криптографічного захисту каналу передачі даних.

Зокрема, підтримка застарілих версій протоколу TLS (1.0 та 1.1), відсутність реалізації сучасного стандарту TLS 1.3, використання слабких параметрів обміну ключами (Diffie-Hellman з довжиною 1024 біти), а також підтримка шифрувальних пакетів, що базуються на алгоритмі SHA-1 та режимі CBC, свідчать про невідповідність конфігурації чинним рекомендаціям з безпеки.

Крім того, відсутність жорсткого контролю за вибором шифрувальних алгоритмів на стороні сервера суперечить кращим практикам і потенційно уможливлює встановлення з'єднань із використанням менш захищених параметрів. Подібна конфігурація створює сприятливі умови для реалізації ряду криптографічних атак, зокрема тих, що спрямовані на послаблення шифрування або перехоплення сесійних ключів.

Таким чином, поточна реалізація TLS-захисту не може вважатися достатньою з погляду сучасних вимог до безпеки веб-застосунків.

2.3 SecurityHeaders.com, як інструмент оцінки захищеності веб-додатків організаціїї

Як зазначалося раніше, HTTP-заголовки безпеки відіграють важливу роль в забезпеченні захисту клієнтської сторони веб-додатків. З їх допомогою розробники можуть обмежувати дії браузера, заборонити виконання потенційно небезпечних скриптів, запобігти вбудованого в iframe контенту і зменшити ризик атаки MITM. Для організацій чия діяльність напряму зв'язана з обробкою даних користувачів, фінансових операцій чи критичної бізнес інформації, ігнорувати такі методи захисту може призвести до серйозних наслідків: від компрометації клієнтської інформації до репутаційних і юридичних втрат. При тому що більшість HTTP-заголовків налаштовуються досить просто і не потребують значних ресурсів, практика показує, що багато корпоративних веб-додатків і внутрішніх систем все ще не використовують ці базові методи захисту і залишаються вразливими до широкого спектру атак.

Для оцінки корекності налаштувань HTTP-заголовків використовуються спеціальні інструменти, один з найпопулярніших і зручніших сервісів є SecurityHeaders.com [4]. Цей безкоштовний веб-сервіс розроблений британським спеціалістом по інформаційній безпеці Скотом Хелмом (Scott Helme).

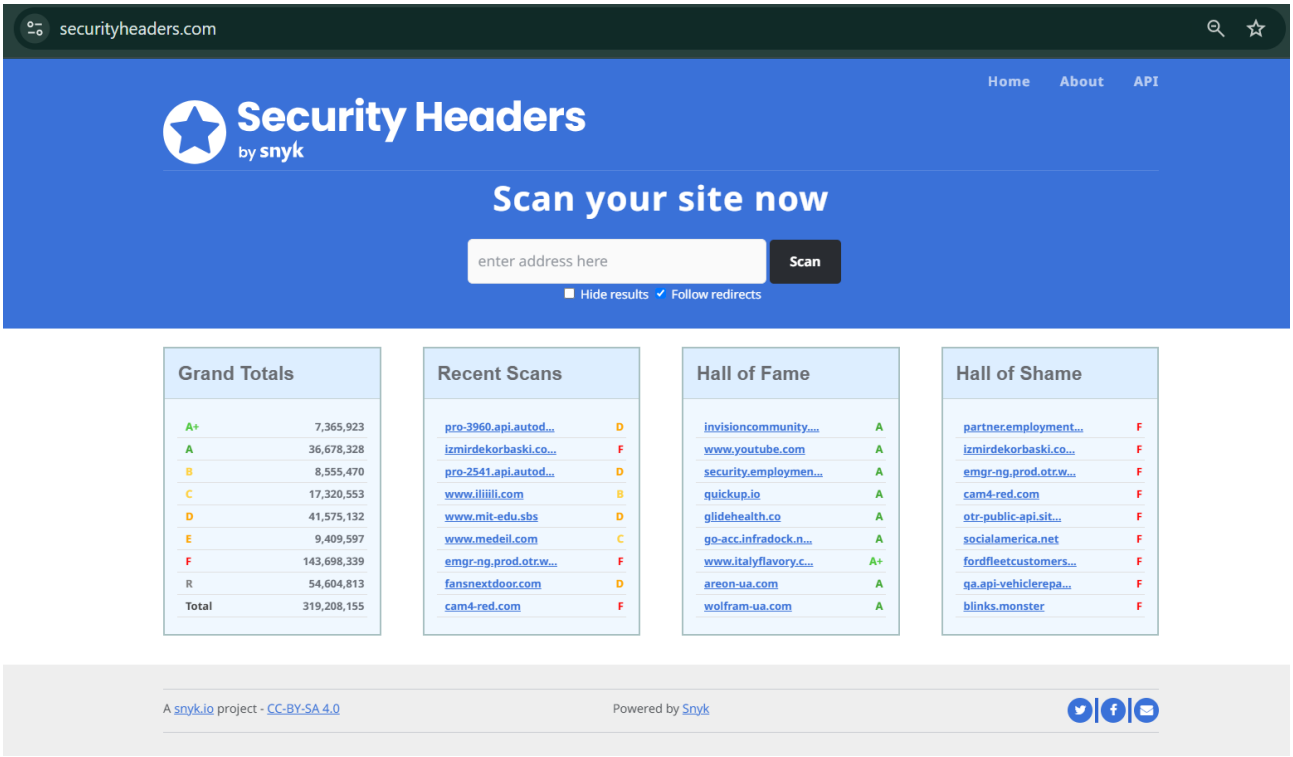

Рис.2.7 Інтерфейс ресурсу SecurityHeaders.com

SecurityHeaders.com проводить швидку та наочну перевірку того які HTTP-заголовки безпеки використовуються на веб-сайті і наскільки вони відповідають сучасним стандартам і рекомендаціям по захисту. Під час використання сервісу, вводиться URL-адреса і SecurityHeaders.com в режимі реального часу виконує HTTP-запит до цільового сайту і аналізує:

- які заголовки включені в HTTP-відповідь;
- які з них відсутні;
- який їх вміст і відповідність рекомендаціям;
- чи не містить вона небезпечних чи застарілих параметрів.

Заголовки, які присутні в відповіді, підсвічуються кольором в залежності від степені їх важливості і коректності налаштувань (зелений – оптимальні, жовтий –

частково, червоний – відсутній або небезпечні) Після сканування веб-додатку ви отримуєте оцінку від A+ до F, засновану на наявності або відсутності критично важливих заголовків. Серед яких:

- Content-Security-Policy – захист від XSS-атак;
- Strict-Transport-Security – примусове використання HTTPS
- X-Frame-Options – захист від clickjacking;
- X-Content-Type-Options – захист від MIME-ін’єкцій;
- Referrer-Policy – управління передачею заголовків посиланнь;
- Permissions-Policy – керування доступом до API браузера.

2.4 Аналіз заголовків безпеки за допомогою SecurityHeaders.com

Оцінювання безпеки веб-застосунку слід проводити комплексно, оскільки сучасні атаки охоплюють різні шари архітектури – від транспортного рівня до рівня додатка. Проведене раніше сканування за допомогою сервісу SSL Labs дозволило проаналізувати налаштування TLS-протоколів, шифрувальних пакетів і сертифікатів, тобто аспектів, що відповідають за захищене з'єднання між клієнтом і сервером. Однак захист веб-застосунку не обмежується лише коректною конфігурацією TLS. Важливим доповненням до цієї оцінки є аналіз HTTP-заголовків безпеки, які реалізуються на рівні веб-сервера або додатка і дозволяють браузерам підвищити стійкість до атак з боку користувацького інтерфейсу.

Саме для перевірки цього рівня було використано сервіс SecurityHeaders.com, який оцінює наявність і правильність конфігурації таких ключових механізмів, як політики Content Security Policy (CSP), HTTP Strict Transport Security (HSTS), X-Frame-Options, X-Content-Type-Options та інших. Ці заголовки, хоч і не шифрують трафік, забезпечують критичний рівень контролю над поведінкою браузера, тим самим знижуючи ризик реалізації XSS, Clickjacking, CSRF і подібних атак. Результати цього сканування наведено нижче.

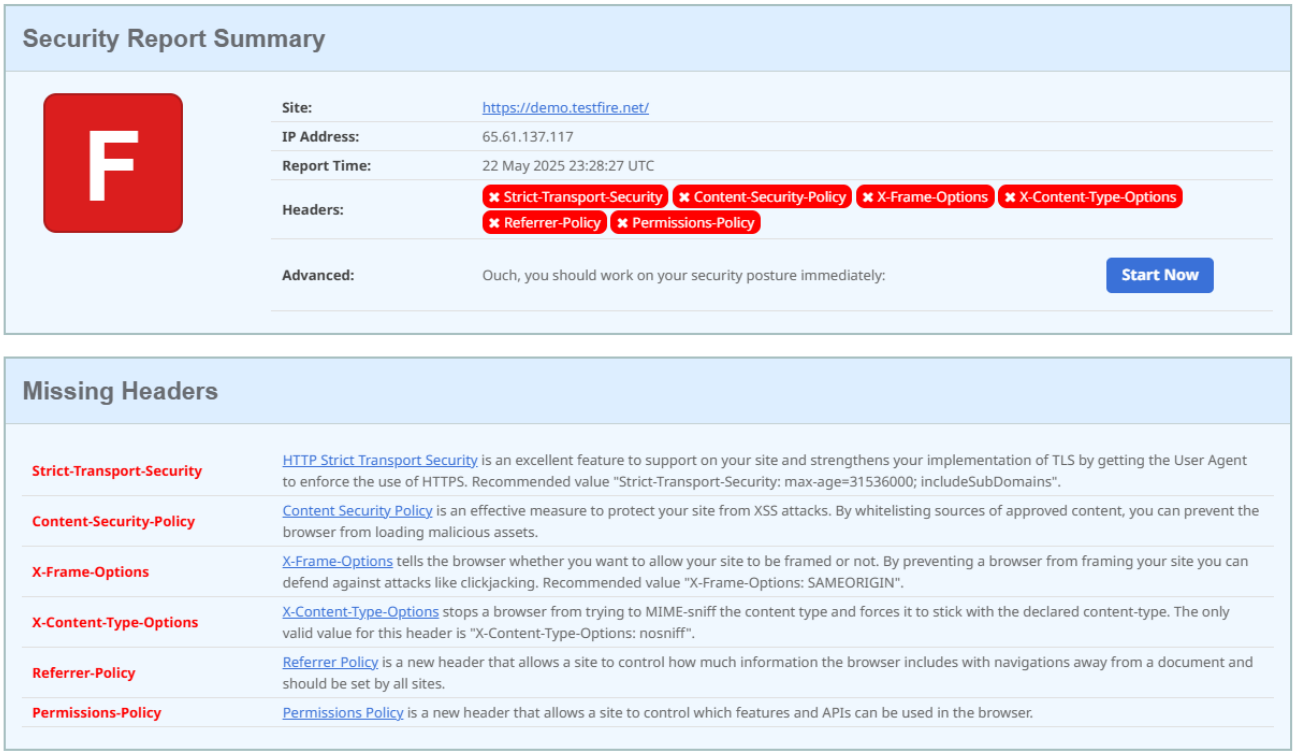

Рис.2.8 Звіт SecurityHeaders.com для demo.testfire.net

За результатами сканування вебресурсу https://demo.testfire.net/ за допомогою сервісу SecurityHeaders.com встановлено, що його конфігурація у сфері HTTP-заголовків безпеки є критично недостатньою. Загальна оцінка безпеки становить F, що свідчить про відсутність базових механізмів захисту на рівні заголовків HTTP-відповіді. Така ситуація становить серйозну загрозу для безпеки веб-застосунку, оскільки залишає його відкритим до широкого спектра атак із боку кінцевого користувача або проміжних компонентів мережі.

Серед ключових негативних аспектів конфігурації:

1) Відсутність заголовка Strict-Transport-Security (HSTS).

Цей заголовок є фундаментальним для реалізації політики захищеного з'єднання. У його відсутності браузер не фіксує вимогу використовувати виключно HTTPS, що дозволяє потенційним зловмисникам проводити атаки типу SSL stripping. Це означає, що навіть за наявності TLS-сертифіката користувач може бути примусово переадресований на незахищену версію ресурсу.

2) Відсутність Content-Security-Policy (CSP).

Політика CSP дозволяє обмежити джерела завантаження контенту (JavaScript, CSS, зображень тощо), що значно ускладнює реалізацію атак типу XSS (Cross-Site Scripting). Її відсутність означає, що браузер не має інструкцій, якому контенту довіряти, і є схильним до ін’єкцій зловмисного коду.

3) Відсутність X-Frame-Options.

Цей заголовок забороняє або обмежує відображення сайту в iframe-елементах на інших сторінках. Його відсутність відкриває можливість атакам типу clickjacking, коли користувача вводять в оману, змушуючи натискати невидимі елементи інтерфейсу.

4) Відсутність X-Content-Type-Options.

У випадку відсутності цього заголовка браузер може спробувати "вгадати" MIME-тип файлу, навіть якщо сервер його чітко вказав. Це створює умови для атак типу MIME-sniffing, коли, наприклад, текстовий файл може бути інтерпретований як виконуваний скрипт.

5) Відсутність Referrer-Policy.

Веб-застосунок не контролює обсяг метаінформації, яку браузер передає при переході на сторонні ресурси. Це потенційно може призводити до витоку конфіденційних даних (наприклад, параметрів запиту або ідентифікаторів сесій).

6) Відсутність Permissions-Policy.

Цей заголовок дозволяє обмежити використання браузерних API та функцій (наприклад, доступ до камери, мікрофона або геолокації). Його відсутність не є критичною для всіх застосунків, але вказує на відсутність налаштувань захисту від потенційного зловживання сторонніми компонентами браузера.

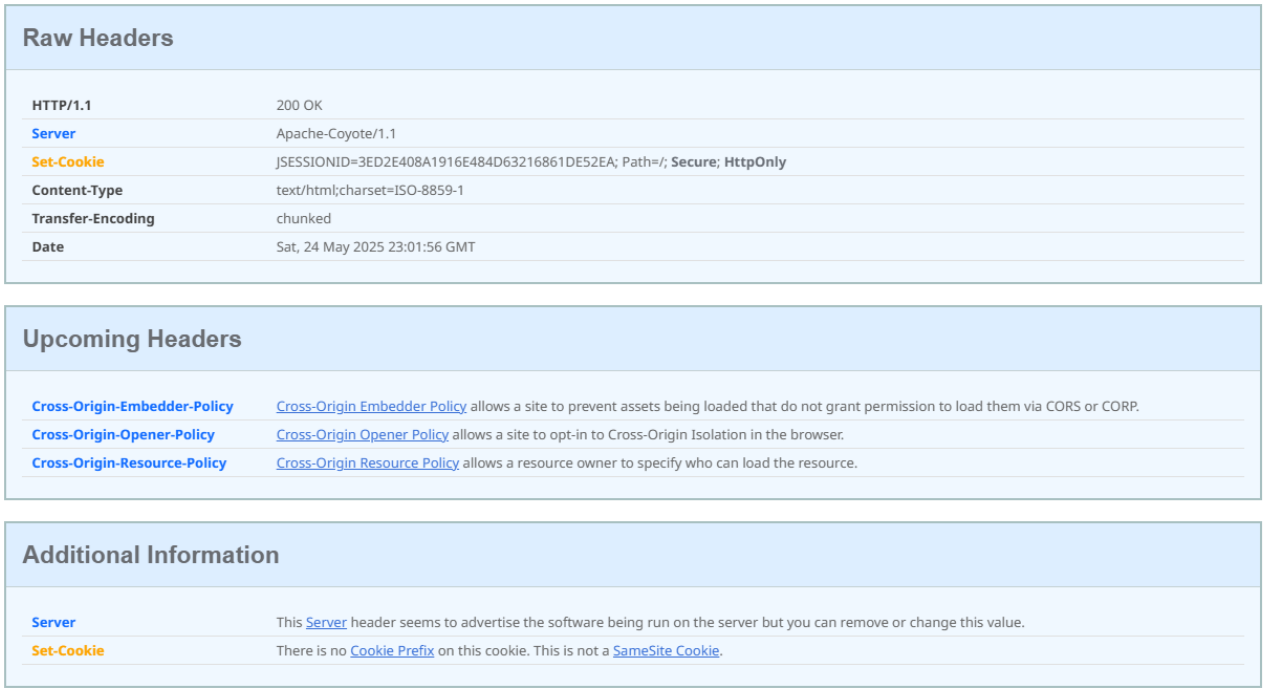

Рис. 2.9 Аналіз HTTP-заголовків вебзастосунку

Окрім цього, у відповідях сервера було виявлено відкриту інформацію про програмне забезпечення, зокрема заголовок Server: Apache-Coyote/1.1, що розкриває потенційну ціль для зловмисника у випадку виявлення відомих вразливостей цієї технології. Також файл cookie JSESSIONID встановлюється без атрибута SameSite, що може дозволити атаки типу Cross-Site Request Forgery (CSRF) у певних сценаріях, хоча він має атрибути Secure і HttpOnly, що є позитивним моментом.

Загалом ситуація свідчить про повну відсутність реалізації заголовків безпеки на рівні веб-сервера або застосунку. Усі перелічені механізми є стандартною практикою для сучасних веб-ресурсів, зокрема для тих, що обробляють автентифікаційні дані, персональну або фінансову інформацію. Їх ігнорування демонструє не лише технічні недоліки, а й організаційні прогалини в політиці безпеки веб-застосунку.

Результати сканування ресурсу https://demo.testfire.net/ за допомогою сервісу SecurityHeaders.com свідчать про повну відсутність ключових HTTP-заголовків безпеки, що є серйозною вразливістю з точки зору захисту веб-застосунку.

Сукупність наведених недоліків свідчить не лише про відсутність окремих механізмів захисту, а про повну неконфігурованість рівня клієнтської безпеки. Це означає, що навіть за наявності добре налаштованого TLS-з'єднання (що було проаналізовано в попередньому розділі), клієнт усе ще залишається вразливим до низки атак, спрямованих на маніпулювання контентом, інтерфейсом або логікою взаємодії.

У реальних умовах, коли веб-застосунки обробляють автентифікаційні сесії, платіжні операції або персональні дані, відсутність належних HTTP-заголовків створює критичні ризики для організації – як технічні, так і правові. Конфігурація HTTP-заголовків безпеки відіграє критичну роль у створенні додаткового шару захисту між веб-додатком і кінцевим користувачем. Їх ігнорування або неправильна реалізація не є формальністю – це суттєвий недолік у моделі безпеки застосунку, який створює можливості для реальних і потенційно катастрофічних атак.

3. РЕКОМЕНДАЦІЇ ЩОДО ПОКРАЩЕННЯ ЗАХИСТУ ВЕБ-ДОДАТКУ ОРГАНІЗАЦІЇ

3.1. Рекомендації щодо вдосконалення TLS-конфігурації протестованого веб-застосунку

Проведене дослідження конфігурації TLS-захисту вебресурсу demo.testfire.net продемонструвало наявність низки технічних недоліків, які мають прямий вплив на рівень криптографічного захисту каналу передачі даних між клієнтом і сервером. З огляду на вимоги актуальних стандартів безпеки, доцільним є впровадження цілого ряду покращень, що сприятимуть зменшенню ризиків, пов’язаних із перехопленням, дешифруванням або підміною трафіку.

Одним із найбільш критичних аспектів є підтримка застарілих версій протоколу TLS 1.0 та 1.1, які не лише мають історію виявлених криптографічних атак (BEAST, POODLE, renegotiation attack тощо), але й не відповідають вимогам галузевих нормативів. Їхнє використання є несумісним із сучасним рівнем довіри до вебзастосунків, особливо тих, що передають конфіденційні дані користувачів. Для досягнення належного рівня безпеки сервер повинен підтримувати виключно TLS 1.2 та TLS 1.3, останній із яких, окрім вищого рівня криптостійкості, також характеризується зниженим часом встановлення з’єднання і гарантує forward secrecy за замовчуванням.

Ще однією суттєвою проблемою є використання слабких параметрів при обміні ключами. Зокрема, застосування алгоритму Diffie-Hellman із довжиною ключа 1024 біти не забезпечує належного рівня стійкості до криптоаналізу з урахуванням сучасних обчислювальних можливостей. Відповідно до рекомендацій NIST (SP 800-131A Rev. 2, стор. 11), така довжина є неприйнятною для використання після 2013 року. Раціональним рішенням у цьому контексті є перехід на алгоритми на основі еліптичних кривих (ECDHE), які дозволяють досягти еквівалентного або вищого рівня захисту при меншому розмірі ключа та

забезпечують forward secrecy – механізм, який виключає можливість дешифрування історичного трафіку в разі компрометації приватного ключа сервера.

У частині налаштувань шифрувальних пакетів також спостерігається конфігураційна неузгодженість. Підтримка CBC-режиму (Cipher Block Chaining) і SHA-1 суперечить сучасним уявленням про криптографічну безпеку, з огляду на наявні атаки типу padding oracle та підтверджену нестійкість алгоритму SHA-1. Крім того, наявна ситуація, коли саме клієнт обирає шифрувальний пакет із запропонованого сервером списку, замість того, щоб сервер диктував порядок пріоритетів. Така поведінка знижує ефективність захисту, оскільки допускає варіанти з використанням слабших алгоритмів, якщо їх підтримує клієнт.

У табл. 3.1 подано порівняння поточних і рекомендованих налаштувань TLS для сервера.

Таблиця. 3.1

Порівняння поточних і рекомендованих налаштувань TLS для сервера.

Категорія	**Поточна конфігурація**	**Рекомендована конфігурація**
Підтримувані версії TLS	1.0, 1.1, 1.2	Лише TLS 1.2 та 1.3
Ключовий обмін	DHE 1024 біт	ECDHE або DHE ≥ 2048 біт
Cipher Suites	Дозволені CBC, SHA-1, DH 1024	Тільки GCM, ChaCha20, SHA256 або SHA384
Пріоритетність шифрів	Клієнт обирає шифр	Сервер встановлює порядок (server-preferred cipher order)
Сертифікат	RSA 2048, DV, без OCSP Must-Staple	RSA або ECDSA, EV або OV, OCSP Must-Staple увімкнений
TLS 1.3	Відсутній	Повна підтримка

Не менш важливим аспектом є питання довіри до сертифіката. Хоча поточний сертифікат є чинним і довіреним, він не використовує розширену перевірку (EV), що в критично важливих системах може розглядатися як недолік.

Окрім того, відсутність OCSP Must-Staple знижує надійність перевірки статусу відкликання сертифіката. У цьому контексті доцільно розглядати не лише технічне оновлення сертифіката, а й впровадження політики автоматичної ротації ключів та контролю за чинністю.

Щодо сумісності, підтримка старих клієнтів (наприклад, Internet Explorer 8 чи Java 6u45) реалізується ціною безпеки, що є небажаним компромісом. Поточні стандарти безпеки рекомендують відмовитися від такої зворотної сумісності на користь підтримки лише клієнтів, що реалізують TLS 1.2 або новіші версії.

На завершення варто зазначити, що забезпечення надійного TLS-захисту не є одноразовим завданням. Воно потребує постійного моніторингу, оновлення компонентів (OpenSSL, бібліотек шифрування, конфігурацій сервера) та періодичного аудиту з використанням таких інструментів, як SSL Labs, Mozilla Observatory, testssl.sh тощо. Систематичне впровадження рекомендованих змін дозволить зменшити поверхню атаки та підвищити загальну стійкість веб-ресурсу до загроз, пов’язаних із передачею даних у відкритих мережах.

3.2. Рекомендації для налаштуванні HTTP-заголовків в протестованому веб-додатку

Аналіз результатів тестування за допомогою сервісу SecurityHeaders.com засвідчив наявність серйозних недоліків у конфігурації HTTP-заголовків веб-застосунку. Відсутність ключових заголовків, які є стандартом безпеки в індустрії, свідчить про вразливість системи до низки атак і порушень принципів захищеного доступу. Щоб виправити цю ситуацію, необхідно вжити комплексних заходів, які охоплюють як технічну реалізацію захисних політик, так і стратегічне планування безпеки на рівні організації. В таблиці зображені нижче порівняння поточних і рекомендованих налаштувань HTTP.

Таблиця 3.2

Порівняння поточних і рекомендованих налаштуваннь HTTP

Категорія	**Поточна конфігурація**	**Рекомендована конфігурація**
Підтримувані версії TLS	TLS 1.0, TLS 1.1, TLS 1.2	Тільки TLS 1.2 і TLS 1.3
Алгоритм обміну ключами	DHE з довжиною ключа 1024 біт	ECDHE або DHE з довжиною ключа не менше 2048 біт
Cipher Suites	Дозволені CBC, SHA-1, DH 1024	Тільки шифри з AES-GCM, ChaCha20, з використанням SHA-256 або SHA-384
Пріоритетність шифрів	Визначає клієнт	Сервер визначає порядок шифрів (server-preferred cipher order)
Сертифікат	RSA 2048, DV, без OCSP Must-Staple	RSA або ECDSA, EV або OV, OCSP Must-Staple активований
Підтримка TLS 1.3	Відсутня	Повна підтримка TLS 1.3

У першу чергу слід зосередитися на забезпеченні гарантованого переходу користувачів на захищений протокол HTTPS. Це реалізується через механізм HTTP Strict-Transport-Security (HSTS), який наказує браузеру ніколи не використовувати небезпечне HTTP-з'єднання навіть у разі помилки в конфігурації або атаки типу «людина посередині». Ігнорування цього механізму не лише знижує рівень безпеки, а й створює можливості для зловмисників перехопити чи модифікувати дані на етапі ініціалізації з'єднання. Запровадження HSTS дозволяє централізовано і стабільно підтримувати захищене середовище для комунікації між користувачем і сервером, формуючи довгострокову довіру браузера до ресурсу.

Ще одним критично важливим елементом захисту є впровадження Content Security Policy (CSP), яка дозволяє контролювати джерела контенту, що завантажуються на веб-сторінку. Ключова ідея полягає в тому, щоб з схвалених доменів і протоколів створити білий список, з якого дозволено завантаження

вмісту, попереджаючи шкідливі атаки, зокрема впровадження несанкціонованого коду[13]. Без такої політики браузер не має обмежень щодо того, з яких доменів можна завантажувати скрипти, стилі, зображення та інші елементи, що відкриває двері для атак типу XSS. Наявність CSP є не просто бажаною практикою, а визнаним стандартом захисту веб-додатків, що суттєво знижує ризик несанкціонованого виконання шкідливого коду на боці клієнта.

Захист від атак clickjacking також є слабким місцем при відсутності заголовка X-Frame-Options. Цей механізм регламентує можливість вбудовування ресурсу в iframe, що використовується зловмисниками для обману користувача – наприклад, приховування інтерфейсу під фоновим шаром із власними кнопками або формами. Такий підхід дозволяє змусити користувача взаємодіяти з легітимним сайтом, не усвідомлюючи цього. Встановлення жорсткої політики щодо фреймування дозволяє повністю усунути цей клас атак.

Заголовок X-Content-Type-Options виконує важливу функцію контролю над типами контенту, який обробляється браузером. У випадку його відсутності браузер може самостійно "вгадати" MIME-тип переданих даних, що у певних випадках може призвести до виконання потенційно шкідливого вмісту. Це, своєю чергою, відкриває шлях до атаки через маніпуляції з розширенням файлів або типами відповідей сервера. Жорстка інтерпретація MIME-типів, зазначених сервером, дозволяє зберігати інтеграцію контенту у визначених межах і запобігати неочікуваному рендерингу або виконанню файлів.

Важливу роль в межах сучасної практики відіграє також політика Referrer-Policy. Відсутність цього заголовка дозволяє браузеру автоматично передавати повні URL-адреси сторінок, з яких здійснюється перехід, включно з параметрами запитів, токенами аутентифікації або іншими конфіденційними даними. Враховуючи сучасну практику використання параметризованих запитів для авторизації або ідентифікації користувача, це створює серйозні ризики витоку персональних або службових даних. Запровадження продуманої політики, яка обмежує або виключає передачу такого контексту при зовнішніх переходах, є необхідним кроком для дотримання принципів конфіденційності.

Ще один важливий аспект, який не було реалізовано у протестованому веб-додатку – це Permissions-Policy. Цей заголовок дозволяє серверу вказати, які можливості браузера можуть бути активовані для певного ресурсу, зокрема доступ до камери, мікрофона, геолокації, Bluetooth та інших чутливих інтерфейсів. Відсутність чіткої політики доступу до таких можливостей значно підвищує ризики для кінцевого користувача, особливо у випадку використання вразливих браузерів або недобросовісних розширень. Обмеження таких можливостей повинно бути реалізовано за замовчуванням, із подальшим поступовим дозволом лише для перевірених сценаріїв використання.

Серйозну проблему становить і заголовок Server, який явно вказує на тип серверного програмного забезпечення (у даному випадку Apache-Coyote/1.1). Така інформація є надзвичайно цінною для зловмисника, оскільки дозволяє ідентифікувати відому платформу, встановити її версію, а відтак – знайти відповідні публічно задокументовані вразливості. Розкриття подібної інформації класифікується як інформаційне розкриття (information disclosure) і прямо суперечить принципам приховування середовища виконання. З метою зменшення ризиків такого типу слід або повністю вилучити відповідний заголовок, або замінити його на узагальнене значення, яке не розкриває технічних деталей.

Окрему увагу необхідно звернути на реалізацію захисту cookie. Попри наявність атрибутів Secure та HttpOnly, що є позитивною практикою, у конфігурації cookie відсутній параметр SameSite. Його ігнорування створює потенційний канал для атак типу CSRF (Cross-Site Request Forgery), які дозволяють зловмисникам виконувати несанкціоновані дії від імені користувача, зокрема фінансові операції або зміну налаштувань облікового запису[15]. Ураховуючи стрімке зростання популярності CSRF-атак у реальному середовищі, впровадження політики SameSite є не просто рекомендованим, а обов’язковим елементом сучасної безпеки.

Підсумовуючи вищезазначене, можна зробити висновок, що ефективне управління HTTP-заголовками є фундаментальним компонентом комплексної безпеки веб-застосунку. Впровадження належної політики заголовків дозволяє створити багаторівневий захист, який покриває як технічні, так і поведінкові

вектори атак. Організації, що прагнуть до високих стандартів кібербезпеки, мають не лише впровадити ці механізми, а й забезпечити їх постійний аудит, оновлення відповідно до нових рекомендацій та інтеграцію у внутрішні процеси розробки й тестування. Такий підхід дозволяє не лише підвищити рівень безпеки, а й підготувати інфраструктуру до відповідності міжнародним стандартам та вимогам нормативного регулювання у сфері захисту інформації.

3.3. Рекомендації для підвищення рівня безпеки веб-додатків організації

На основі проведеного аналізу найбільш поширених вразливостей веб-додатків згідно з OWASP Top 10, а також результатів практичного сканування з використанням інструментів SSL Labs та Security Headers, можна сформувати низку рекомендацій, спрямованих на підвищення загального рівня кіберзахисту організаційних веб-ресурсів.

1) Розробка стратегій безпечного проєктування

Організаціям слід закладати вимоги до безпеки ще на етапі архітектурного планування веб-додатків. Варто впровадити техніки threat modeling, а також використовувати шаблони безпечного програмування. Це дозволить запобігти загрозам типу Insecure Design і мінімізувати ризики, пов’язані з неочевидними логічними помилками або небезпечними архітектурними рішеннями.

2) Покращення автентифікації та управління сесіями

Використання багатофакторної автентифікації (MFA) є ключовим елементом захисту облікових записів користувачів. Усі паролі повинні зберігатися в зашифрованому вигляді з використанням сучасних алгоритмів хешування (наприклад, bcrypt або Argon2). Сесійні токени мають бути короткочасними, шифрованими та захищеними через атрибути HttpOnly, Secure, SameSite=Strict.

3) Застосування принципів мінімального доступу

Слід впровадити строгу модель контролю доступу, що дозволяє кожному користувачу тільки ті дії, які передбачені його роллю. Усі перевірки доступу

повинні здійснюватися на стороні сервера. Це дозволить уникнути порушень типу Broken Access Control.

4) Регулярне оновлення всіх компонентів

Одна з найпоширеніших причин вразливостей – використання застарілих або вразливих бібліотек. Рекомендується створити внутрішню політику управління залежностями: використовувати системи автоматичного сповіщення про вразливості (наприклад, OWASP Dependency-Check, Snyk) та тестувати оновлення в ізольованому середовищі перед впровадженням.

5) Усунення криптографічних помилок

Необхідно забезпечити використання сучасних протоколів шифрування, зокрема TLS 1.3. Заборонити застосування слабких алгоритмів, таких як RC4, MD5 чи SHA-1. Усі передані дані мають шифруватися за допомогою HTTPS. Крім того, сертифікати повинні бути чинними, виданими авторитетним CA, а також підтримувати перевірку OCSP та HSTS.

6) Безпечна обробка введення

Для захисту від ін’єкцій (SQL, XSS та ін.) потрібно впровадити параметризовані запити, використовувати ORM-фреймворки та сувору валідацію введення. Заборонено динамічно формувати SQL-запити на основі введення користувача без належної фільтрації.

7) Захист від SSRF та небажаних внутрішніх запитів

Рекомендується обмежити зовнішні HTTP-запити серверної частини додатка, впровадити фільтрацію URL, а також сегментувати мережу, ізолюючи внутрішні служби. У випадку необхідності взаємодії з зовнішніми API – використовувати проксі або WAF для додаткового контролю.

8) Конфігурація HTTP заголовків

Згідно з результатами тестування Security Headers, організаціям рекомендується реалізувати такі заголовки:

Strict-Transport-Security – для примусової роботи через HTTPS;

X-Content-Type-Options: nosniff – для захисту від MIME-атаки;

Content-Security-Policy – для захисту від XSS;

X-Frame-Options – для запобігання Clickjacking.

9) Налаштування сертифікатів і SSL/TLS

На основі аналізу результатів сканування сервера за допомогою SSL Labs було виявлено низку важливих аспектів, які потребують вдосконалення для забезпечення належного рівня криптографічного захисту. Перш за все, рекомендується використовувати лише надійні криптографічні алгоритми, які підтримують forward secrecy (наприклад, ECDHE), що гарантує конфіденційність сесійних ключів навіть у разі їх компрометації. Поточне використання 1024-бітних ключів є недостатнім – необхідно впровадити ключі довжиною не менше 2048 біт, які відповідають сучасним стандартам безпеки. Крім того, слід уникати включення кореневого сертифіката в сертифікаційний ланцюг, оскільки така практика суперечить рекомендаціям галузі та призводить до зайвого мережевого навантаження.

Також важливо впровадити протокол TLS 1.3, який забезпечує покращений рівень захисту та ефективніше встановлення з’єднань. Варто поступово відмовитися від застарілих версій TLS 1.0 і TLS 1.1, оскільки вони містять відомі вразливості. Підтримка сучасних протоколів підвищує не лише безпеку, а й довіру з боку клієнтів і партнерів. Зрештою, регулярний моніторинг конфігурацій та повторні тести є необхідною умовою для забезпечення безперервного вдосконалення захисту веб-додатків.

10) Моніторинг та журналювання

Ретельне налаштування збору та аналізу логів є критично важливою складовою захисту веб-додатків організації. Усі події, що стосуються автентифікації (спроби входу, вдалого чи невдалого), зміни прав доступу, дій з привілеями адміністратора, а також виникнення критичних системних або прикладних помилок повинні фіксуватись із зазначенням часу, IP-адреси та ідентифікатора користувача. Це дозволяє не лише виявляти підозрілу активність у режимі реального часу, але й проводити детальний ретроспективний аналіз у разі інцидентів безпеки.

Інтеграція спеціалізованих систем управління інформаційною безпекою (SIEM – Security Information and Event Management), таких як Splunk, IBM QRadar чи ELK Stack, значно підвищує ефективність цього процесу. Вони забезпечують централізований збір, кореляцію, збагачення та аналіз подій із багатьох джерел, дозволяючи виявляти складні загрози, які важко виявити простим переглядом логів. Крім того, ці системи автоматично формують звітність відповідно до вимог безпекових стандартів (наприклад, ISO/IEC 27001, NIST), що є цінним для внутрішнього аудиту або зовнішніх перевірок.

Регулярне тестування налаштувань логування, перевірка повноти логів, їх збереження в захищеному середовищі (наприклад, із захистом від несанкціонованого видалення або зміни), а також обмеження доступу до логів лише уповноваженим особам – усе це є обов’язковими елементами комплексної політики моніторингу безпеки.

ВИСНОВОК

У кваліфікаційній роботі проведено комплексне дослідження проблеми забезпечення безпеки веб-додатків організацій, визначено її мету, основні завдання, а також обґрунтовано актуальність теми в умовах постійного зростання кіберзагроз, що спрямовані на веб-середовище.

У процесі теоретичного аналізу встановлено, що веб-додатки залишаються однією з найбільш уразливих ланок інформаційної інфраструктури зокрема у розрізі стандарту OWASP Top 10, який залишається фундаментальним орієнтиром для виявлення й усунення критичних вразливостей. Також, досліджено міжнародні та національні стандарти, нормативні документи й практичні підходи до забезпечення захисту веб-застосунків, включаючи вимоги ISO/IEC 27001, OWASP ASVS, NIST SP 800-53, PCI DSS, а також законодавчі акти України.

Проведений огляд сучасних підходів до захисту веб-додатків засвідчив важливість застосування цілісного підходу, що включає як запобіжні, так і виявлювальні та реагувальні заходи. Особливу увагу приділено використанню криптографічних протоколів SSL/TLS для забезпечення конфіденційності переданих даних та впровадженню політик безпеки через HTTP-заголовки.

Було виконано практичну частину, яка включала тестування конфігурації веб-додатка за допомогою інструментів SSL Labs і SecurityHeaders.com, що дозволило виявити типові недоліки в реалізації захисту та сформулювати пропозиції щодо їх усунення.

У процесі дослідження було розроблено комплекс рекомендацій, спрямованих на підвищення рівня захищеності веб-додатків організації. Зокрема, запропоновано оптимізувати налаштування протоколів шифрування TLS, впровадити сучасні механізми аутентифікації, реалізувати безпечні заголовки HTTP (Strict-Transport-Security, Content-Security-Policy, X-Content-Type-Options тощо).

Таким чином, у результаті проведеного дослідження підтверджено, що належний рівень безпеки веб-додатків неможливий без комплексного підходу до

управління ризиками, який передбачає інтеграцію захисту на етапі проєктування, використання сучасних протоколів і стандартів, регулярне тестування, а також забезпечення відповідності законодавчим вимогам. Практичні результати роботи мають прикладне значення й можуть бути використані фахівцями з кібербезпеки для вдосконалення систем захисту веб-додатків в організаціях різного профілю.

ПЕРЕЛІК ПОСИЛАНЬ

1. CYCOGNITO. 2024 State of Web Application Security Testing. URL: https://www.cycognito.com/documents/reports/CyCognito-State-of-Web-Application-Seucirty-Testing-2024.pdf (дата звернення: 20.05.2025).

2. OWASP Top Ten | OWASP Foundation. *OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation*. URL: https://owasp.org/www-project-top-ten (дата звернення: 20.05.2025).

3. IBM. What is Log4Shell? | IBM. *IBM - United States*. URL: https://www.ibm.com/think/topics/log4shell (дата звернення: 20.05.2025).

4. OWASP top 10 (2021) threat levels & scenarios - euriun technologies. *Euriun Technologies*. URL: https://www.euriun.com/owasp (дата звернення: 20.05.2025).

5. SSL Server Test (Powered by Qualys SSL Labs). *Qualys SSL Labs*. URL: https://www.ssllabs.com/ssltest (дата звернення: 20.05.2025).

6. Altoro Mutual. Altoro Mutual. URL: https://demo.testfire.net (дата звернення: 20.05.2025).

7. Про захист інформації в інформаційно-телекомунікаційних системах : Закон України від 05.07.1994 № 80/94-ВР : станом на 20 квіт. 2025 р. URL: https://zakon.rada.gov.ua/laws/show/80/94-вр#Text (дата звернення: 20.05.2025).

8. Про затвердження Положення про організацію заходів із забезпечення інформаційної безпеки в банківській системі України : Постанова Нац. банку України від 28.09.2017 № 95. URL: https://zakon.rada.gov.ua/laws/show/v0095500-17#Text (дата звернення: 20.05.2025).

9. SO/IEC 27001:2022. Information technology – Security techniques – Management systems for information security: Requirements. Geneva: ISO, 2022. (дата звернення: 20.05.2025).

10. NIST. NIST Special Publication 800-53: Recommended Security Controls for Federal Information Systems and Organizations. NIST, 2017 (дата звернення: 20.05.2025).

11. Website Security Guide: Secure & Protect Your Website. Sucuri. URL: https://sucuri.net/guides/website-security (date of access: 20.05.2025).

12. Cloudflare. (n.d.). What happens in a TLS handshake? URL: https://www.cloudflare.com/learning/ssl/how-does-ssl-work/ (дата звернення: 20.05.2025).

13. Захист веб-додатків: як використовувати Content Security Policy (CSP) для підвищення безпеки. qanapa.com. URL: https://qanapa.com/2025/01/03/zakhyst-veb-dodatkiv-iak-vykorystovuvaty-content-security-policy-csp-dlia-pidvyshchennia-bezpeky (дата звернення: 21.05. 2025).

14. SQL Injection | OWASP Foundation. OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation. URL: https://owasp.org/www-community/attacks/SQL_Injection (дата звернення: 21.05.2025).

Cross Site Request Forgery (CSRF) | OWASP Foundation. OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation. URL: https://owasp.org/www-community/attacks/csrf (дата звернення: 21.05.2025).