МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

Кваліфікаційна наукова
праця на правах рукопису

РЕВНЮК ОЛЕКСАНДР АНДРІЙОВИЧ

УДК 004.056.5:004.031.42

ДИСЕРТАЦІЯ

РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ОЦІНЮВАННЯ ЯКОСТІ ЗАХИСТУ ВЕБДОДАТКІВ

122 – Комп'ютерні науки
12 – Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

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

_____/О. А. Ревнюк/

Науковий керівник Загородна Наталія Володимирівна, кандидат технічних наук, доцент

Тернопіль - 2025

АНОТАЦІЯ

Ревнюк О.А. Розробка інформаційної системи оцінювання якості захисту вебдодатків. – Кваліфікаційна наукова праця на правах рукопису. Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 122 «Комп’ютерні науки». – Тернопільський національний технічний університет імені Івана Пулюя, Тернопіль, 2025.

Підготовка здійснювалась на кафедрі кібербезпеки Тернопільського національного технічного університету імені Івана Пулюя Міністерства освіти і науки України.

Зміст анотації. Дисертація присвячена вирішенню науково-практичної задачі – розроблення інформаційної технології кількісної оцінки рівня захищеності вебдодатків з урахуванням індивідуальних характеристик проєкту. У вступі обґрунтовано актуальність теми дисертації, зазначено зв'язок роботи з науковими напрямками та науково-дослідними концепціями, сформульовано мету і задачі дослідження, визначено об'єкт, предмет і методи дослідження, показано наукову новизну та практичне значення отриманих результатів, наведено інформацію про практичне використання, апробацію результатів та їх висвітлення у публікаціях.

У першому розділі проведено аналіз сучасних вебдодатків як багатокомпонентних систем у контексті життєвого циклу розробки програмного забезпечення (ПЗ). Здійснено порівняльний аналіз традиційних та ітеративних моделей життєвого циклу ПЗ, що дало змогу виявити принципові обмеження традиційних підходів та встановити переваги ітеративних моделей для безперервної інтеграції безпекових практик. Аналіз відкритих статистичних даних дозволив встановити пряму залежність між зростанням архітектурної складності вебдодатків та збільшенням кількості потенційних векторів атак і системних вразливостей, що дало змогу ідентифікувати основні категорії загроз вебсайтів. Досліджено концепцію Secure Development Lifecycle, в рамках якої проаналізовано існуючі методи й засоби тестування безпеки, що дало змогу ідентифікувати їхні обмеження у виявленні вразливостей проєктування та бізнес-логіки, а також

встановити відсутність можливості обчислення комплексної кількісної оцінки рівня захищеності.

У другому розділі досліджено проблему суб'єктивності експертних оцінок та відсутності уніфікованих кількісних метрик у задачах оцінювання безпеки вебдодатків, що дало змогу обґрунтувати необхідність розробки нових підходів до об'єктивізації процесів оцінювання захищеності. На основі аналізу існуючих міжнародних галузевих стандартів та традиційних методів тестування безпеки встановлено їхню неспроможність обчислити інтегральний показник захищеності вебдодатків. Вперше запропоновано метод кількісної оцінки захищеності вебдодатків на основі структурного аналізу та адаптивного відбору вимог стандарту OWASP ASVS. Для формалізації відібраних вимог розроблено критерії оцінювання, що дало змогу створити систематизований підхід до кількісного вимірювання рівня безпеки. Запропонований метод передбачає введення коефіцієнтів важливості для вимог та критеріїв з урахуванням архітектурних, функціональних та бізнес-особливостей вебдодатку. Застосування апарату нечіткої логіки для визначення коефіцієнтів важливості дало змогу мінімізувати суб'єктивність експертних суджень та підвищити точність оцінювання.

У третьому розділі проведено експериментальне дослідження рівня безпеки вебдодатку на прикладі спеціалізованого навчального середовища з попередньо вбудованими вразливостями, що дало змогу на практиці оцінити розроблений адаптивний метод кількісної оцінки захищеності та обґрунтувати його кореляцію з результатами виявлених критичних вразливостей у незалежних дослідженнях. Отримала подальший розвиток концепція використання апарату нечіткої логіки для підвищення точності та об'єктивності експертного оцінювання вагових коефіцієнтів через механізми фазифікації, узгодження думок експертів та дефазифікації, що дало змогу формалізувати суб'єктивні судження, мінімізувати вплив індивідуальних упереджень та забезпечити математично обґрунтовану основу для встановлення коефіцієнтів важливості.

У четвертому розділі на основі розробленого адаптивного методу спроєктовано архітектурну та логічну модель інформаційної системи кількісного

оцінювання рівня захищеності вебдодатків. Розроблено інформаційну систему кількісної оцінки безпеки, яка спрощує процес перевірки на відповідність стандарту OWASP ASVS та забезпечує прозорість і відтворюваність результатів з можливістю зміни, у разі потреби, не лише оцінок, а й коефіцієнтів важливості, встановлених за замовчуванням. Розроблена архітектура системи з модульним розділенням функціональних компонентів забезпечує високу ефективність експертного аналізу та супроводу вебдодатків, що дало змогу здійснити практичну апробацію розробленого методу кількісної оцінки захищеності вебдодатків.

Ключові слова: інформаційна система, інформаційні технології, безпека, програмне забезпечення, веб-додаток, життєвий цикл програмного продукту, якість ПЗ, вимоги, критерії, стандарти, метрики, тестування, оцінювання, вразливості, ризики, заходи безпеки, автентифікація, нечітка логіка.

ABSTRACT

Revniuk O.A. Development of an Information System for Evaluating Web Application Security Quality. – Qualifying scientific work as a manuscript. Dissertation for obtaining the degree of Doctor of Philosophy in specialty 122 "Computer Sciences". – Ternopil Ivan Puluj National Technical University, Ternopil, 2025.

The preparation was carried out at the Department of Cybersecurity of Ternopil Ivan Puluj National Technical University of the Ministry of Education and Science of Ukraine.

Content of the abstract. The dissertation is devoted to solving a scientific-practical problem – development of information technology for quantitative assessment of web application security level taking into account individual project characteristics. The introduction substantiates the importance of the dissertation topic, indicates the connection of the work with scientific directions and research concepts, formulates the purpose and objectives of the research, defines the object, subject and research methods, shows the scientific novelty and practical significance of the obtained results, provides information about practical use, approbation of results and their coverage in publications.

The first chapter analyzes modern web applications as multi-component systems in the context of the software development lifecycle (SDLC). A comparative analysis of traditional and iterative SDLC models was conducted, which allowed identifying fundamental limitations of traditional approaches and establishing advantages of iterative models for continuous integration of security practices. Analysis of open statistical data allows to establish a direct connection between the growth of web application architectural complexity and the increase in the number of potential attack vectors and system vulnerabilities, which allowed identifying main categories of information security threats. The Secure Development Lifecycle concept was studied and existing security testing methods and tools were analyzed, which allowed to identify their limitations in detecting design and business logic vulnerabilities, as well as to establish the absence of comprehensive quantitative security level assessment capabilities.

The second chapter investigated the problem of subjectivity in expert assessments and the lack of unified quantitative metrics on the problems of web application security

evaluation, which allowed substantiating the need to develop new approaches to objectifying security assessment processes. Based on analysis of existing international standards and traditional security testing methods the inability to provide an integral security indicator was established. For the first time, an methodology for quantitative web application security assessment was suggested based on structural analysis and adaptive selection of OWASP ASVS standard requirements. To formalize selected requirements evaluation criteria were developed. This allowed creating a systematized approach to quantitative security level measurement. The proposed methodology introduces the importance coefficients for requirements and criteria, taking into account architectural, functional and business features of the web application. Application of fuzzy logic apparatus to determine the imporatance coefficient allowed minimizing subjectivity of expert judgments and improving assessment accuracy.

The third chapter conducted experimental research on web application security level using a specialized educational environment with pre-implemented vulnerabilities, which allowed confirming practical applicability of the developed adaptive quantitative security assessment methodology and substantiating its correlation with results of identified critical vulnerabilities in different studies. The concept of using fuzzy logic apparatus to improve accuracy and objectivity of expert assessment of weight coefficients through fuzzification mechanisms, expert opinion coordination and defuzzification was further developed, which allowed formalizing subjective judgments, minimizing the influence of individual biases and providing a mathematically substantiated basis for quantitative measurement of security parameters.

The fourth chapter, based on the developed adaptive methodology, performed design of architectural and logical models of an information system for quantitative assessment of web application security level with a simplified security coefficient calculation process. An information system for quantitative security assessment was developed, which simplifies the process of checking compliance with the OWASP ASVS standard and ensures transparency and reproducibility of results, which allowed conducting practical modeling of corresponding information-technological complexes. The designed system architecture with modular separation of functional components

ensures high efficiency in the context of expert analysis and web resource support, which allowed practical approbation of the developed methodology for quantitative web application security assessment.

Key words: information system, information technologies, security, software, web-application, software lifecycle, software quality, requirements, criteria, standards, metrics, testing, assessment, vulnerabilities, risks, security controls, authentication, fuzzy logic.

ПЕРЕЛІК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІ

Праці, в яких опубліковано основні наукові результати дисертації:

1. Revniuk O. A., Zagorodna N. V., Kozak R. O., Karpinski M. P., Flud L. O. “The improvement of web-application SDL process to prevent Insecure Design vulnerabilities”, *Applied Aspects of Information Technology*, *7*(2), pp.162–174, 2024. I**SSN 2617-4316 (Print), https://doi.org/10.15276/aait.07.2024.12.**
2. Ревнюк О., Постолюк А. “Дослідження застосування адаптивних методик оцінювання ризиків безпеки для вебдодатків”, *Computer Systems and Information Technologies*, *3*, ст.34–43, 2024. **ISSN 2710-0766, https://doi.org/10.31891/csit-2024-3-5.**
3. Ревнюк О. А., Загородна Н. В. “Методологія кількісної оцінки захищеності вебдодатку електронної комерції на етапі експлуатації”, *Scientific Bulletin of Ivano-Frankivsk National Technical University of Oil and Gas*, *2(57)*, ст.107–119, 2024. **ISSN1993–9965 print, https://doi.org/10.31471/1993-9965-2024-2(57)-107-119.**
4. Revniuk O., Zagorodna N., Ulichev O. «Adaptive methodology for computing the quantitative Security status indicator of web applications», *Central Ukrainian Scientific Bulletin Technical Sciences*, *2*(10(41)), pp.3–10, 2024. **ISSN 2664-262Xб https://doi.org/10.32515/2664-262x.2024.10(41).2.3-10.**
5. Ulichev O., Papizh L., & Revniuk O. “Advancements in software Testing: a scientific perspective”, *Central Ukrainian Scientific Bulletin Technical Sciences*, *1*(10(41)), pp. 3–16, 2024. **ISSN 2664-262X, https://doi.org/10.32515/2664-262x.2024.10(41).1.3-16.**
6. Revniuk O., Zagorodna N., Kozak R., Yavorskyy B. I. «Development of an information system for the quantitative assessment of web application security based on the OWASP ASVS standard», Scientific Journal of the Ternopil National Technical University, 2 (118), pp. 56-65, 2025**. ISSN: 2522-4433 Print.**

Праці, які засвідчують апробацію матеріалів дисертації:

7. О. Ревнюк, Н. Загородна. “Моделі та методи оцінювання якості вебдодатків”, Матеріали IX науково-технічної конференції «Інформаційні моделі, системи та технології», ТНТУ, Тернопіль, Україна, 2021, ст. 73–74.
8. Ревнюк О. «Якість управління інформаційною безпекою організацій», Матеріали X науково-технічної конференції «Інформаційні моделі, системи та технології», ТНТУ, Тернопіль, Україна, 2022, ст. 42–43.
9. Н. Загородна, О. Ревнюк, Р. Козак. «Кількісна оцінка безпеки прототипу інтернет-магазину OWASP Juice Shop», Матеріали XIV міжнародної науково-технічної конференції ITSEC: Безпека інформаційних технологій, ЗУНУ-ДУІКТ, Тернопіль-Київ, 22-24 травня 2025, ст.75-76

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

ACID (англ. Atomicity, Consistency, Isolation, Durability) – набір властивостей, що гарантують надійність транзакцій у базах даних

API (англ. Application Programming Interface) – набір правил та протоколів для взаємодії між програмними компонентами

ASVS (англ. Application Security Verification Standard) – стандарт OWASP для оцінки рівня безпеки вебдодатків

CAPTCHA (англ. Completely Automated Public Turing test to tell Computers and Humans Apart) – технологія захисту від автоматизованих атак

CI/CD (англ. Continuous Integration/Continuous Deployment) – практики автоматизації процесів розробки та розгортання програмного забезпечення

CMS (англ. Content Management System) – програмне забезпечення для створення та управління цифровим контентом

COG (англ. Center of Gravity) – метод дефазифікації в нечіткій логіці для отримання чіткого значення

COMa (англ. Center of Maxima) – метод дефазифікації, що використовує середнє арифметичне максимальних значень

CWE (англ. Common Weakness Enumeration) – каталог типових вразливостей програмного забезпечення

DAST (англ. Dynamic Application Security Testing) – метод тестування безпеки працюючого застосунку

DC (англ. Data Criticality) – показник важливості та чутливості інформаційних активів

DevOps (англ. Development and Operations) – підхід до розробки ПЗ, що об'єднує команди розробки та операційного управління

DevSecOps (англ. Development, Security and Operations) – розширення DevOps з інтеграцією практик безпеки

HTML5 (англ. HyperText Markup Language version 5) – стандарт розмітки для створення вебсторінок

HTTP (англ. HyperText Transfer Protocol) – протокол прикладного рівня для передачі даних у всесвітній павутині

ISMS (англ. Information Security Management System) – комплекс заходів для управління інформаційною безпекою організації

MFA (англ. Multi-Factor Authentication) – метод автентифікації з використанням кількох незалежних факторів

MoM (англ. Mean of Maxima) – метод дефазифікації для визначення середнього значення з множини максимумів

MVP (англ. Minimum Viable Product) – версія продукту з мінімальним набором функцій для тестування ринку

NIST (англ. National Institute of Standards and Technology) – американська федеральна агенція, що розробляє стандарти кібербезпеки

OC (англ. Operational Criticality) – показник важливості системи для функціонування організації

OTP (англ. One-Time Password) – автентифікаційний код, що використовується лише один раз

OWASP (англ. Open Web Application Security Project) – некомерційна організація, що розробляє стандарти безпеки вебдодатків

ПЗ – програмне забезпечення, сукупність програм та документації для функціонування комп'ютерних систем

QA (англ. Quality Assurance) – процес контролю якості програмного продукту на всіх етапах розробки

RAD (англ. Rapid Application Development) – методологія розробки ПЗ з акцентом на швидкість створення прототипів

RWDLC (англ. Rapid Web Development Life Cycle) – адаптована модель SDLC для вебдодатків

SANS (англ. SysAdmin, Audit, Network and Security Institute) – організація з навчання та сертифікації в галузі кібербезпеки

SAST (англ. Static Application Security Testing) – аналіз вихідного коду для виявлення вразливостей

SDLC (англ. Software Development Life Cycle) – процес планування, створення, тестування та розгортання програмного забезпечення

SDL (англ. Security Development Lifecycle) – методологія Microsoft для інтеграції безпеки в процес розробки ПЗ

SEO (англ. Search Engine Optimization) – процес покращення видимості вебсайту в результатах пошуку

SOA (англ. Service-Oriented Architecture) – архітектурний підхід до розробки додатків як набору слабо пов'язаних сервісів

SQuaRE (англ. Software Quality Requirements and Evaluation) – серія стандартів ISO для оцінки якості ПЗ

STRIDE (англ. Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) – модель для ідентифікації загроз безпеці

СУІБ – система управління інформаційною безпекою, організаційно-технічний комплекс заходів для забезпечення інформаційної безпеки

TCP/IP (англ. Transmission Control Protocol/Internet Protocol) – базовий набір протоколів для передачі даних в мережі Інтернет

UX/UI (англ. User Experience/User Interface) – дизайн взаємодії користувача з програмним продуктом

ВСТУП

Актуальність теми.

Вебдодатки стали невід'ємною частиною життєдіяльності багатьох організацій та користувачів. Вони використовуються у різноманітних сферах – від електронної комерції до державних сервісів і соціальних мереж, забезпечуючи зручний доступ до послуг, автоматизацію бізнес-процесів і керування даними у реальному часі. Проте з розвитком вебтехнологій зростає і поверхня кіберзагроз, що призводить до необхідності впровадження нових, більш ефективних методів і засобів забезпечення захищеності вебдодатків. Особливо в зоні ризику опиняються інтернет-магазини та електронна комерція, які, сьогодні, є основою ведення бізнесу для багатьох компаній.

Сучасні дослідження у сфері безпеки вебсайтів характеризуються комплексним підходом, який охоплює як традиційні аспекти безпеки, так і нові виклики, пов'язані з розвитком хмарних технологій, мікросервісної архітектури та прогресивних вебдодатків. Особлива увага приділяється автоматизованим засобам виявлення вразливостей, включаючи розробку більш досконалих сканерів безпеки та методів статичного і динамічного аналізу коду.

Важливим етапом у розвитку досліджень стало формування галузевих стандартів та рекомендацій від провідних організацій з інформаційної безпеки, таких як OWASP, NIST та SANS. Ці стандарти, зокрема OWASP Top 10, не лише систематизували знання про відомі вразливості та методи захисту, але й сприяли розробці єдиної методологічної бази для проведення досліджень та аудиту безпеки вебдодатків на різних етапах їх життєвого циклу.

Незважаючи на інтенсивний розвиток методологічних баз та ініціатив стандартизації у галузі інформаційної безпеки, спостерігається суттєва прогалина в інструментарії оцінки захищеності вебсайтів. Важливий науковий внесок у сферу досліджень безпеки вебдодатків зробили А. П. Алдія, С. Сутікно, Й. Росмансіах (Бандунгський технологічний інститут, Індонезія), Ф. Буккафуррі (Університет Медітерранеа, Реджо-Калабрія, Італія), Д. Флейтер (дослідницькі установи США).

та інші дослідники, які вивчали проблематику виявлення та класифікації вразливостей. Аналіз опублікованих наукових досліджень свідчить про те, що існуючі автоматизовані системи сканування безпеки, хоча й забезпечують виявлення широкого спектру вразливостей, однак демонструють істотні обмеження у верифікації критичних параметрів захисту. Зокрема, залишається багато критичних моментів поза межами їх функціональності, які потрібно перевіряти вручну.

Чинні галузеві стандарти та нормативні документи у сфері вебзахисту здебільшого зосереджуються на класифікації та категоризації потенційних вразливостей вебдодатків, проте не пропонують уніфікованих метододів кількісної оцінки рівня їх захищеності. З огляду на різноманітність вебдодатків та диверсифікацію їх архітектурних рішень, актуалізується проблематика визначення необхідного та достатнього обсягу тестування безпеки. Відсутність стандартизованої системи обчислення ступеня захищеності вебдодатків суттєво ускладнює ідентифікацію потенційних векторів компрометації та розробки рішень для покращення рівня захисту.

Актуальність роботи обумовлена існуючим парадоксом: сучасні технології дозволяють ефективно виявляти широкий спектр вразливостей вебдодатків, проте відсутні уніфіковані метрики та методи комплексної оцінки рівня безпеки вебдодатку як цілісної системи. Отже, існує необхідність закриття прогалин у підходах, що грунтуються на автоматизованих засобах виявлення вразливостей та експертному оцінюванні захищеності вебзастосунків. Тому розробка науково обґрунтованого підходу до кількісного оцінювання захищеності вебзастосунків на основі галузевих стандартів та відомих методів виявлення вразливостей є актуальною науковою задачею.

Зв’язок роботи з науковими програмами, планами, темами. Дисертаційне дослідження пов’язане з виконанням науково-дослідної теми «Науково-методичні засади цифрової трансформації та сталого розвитку економіки», № держреєстрації ВК 75-24.

Мета і завдання дослідження. Метою дисертаційного дослідження є розробка інформаційної системи, що формалізує технології кількісного оцінювання рівня захищеності вебдодатку та враховує вимоги галузевих стандартів, експертних оцінок, життєвого циклу розробки, специфіку функціоналу та архітектуру вебдодатку.

Для досягнення поставленої мети необхідно виконати наступні завдання:

1. Провести аналітичний огляд сучасного стану та проблем безпеки вебдодатків, існуючих підходів, методів і технологій оцінки їх захищеності.
2. Побудувати систему кількісних критеріїв оцінювання рівня захисту вебдодатків шляхом формалізації якісних вимог стандарту OWASP ASVS, що узгоджуються з іншими галузевими стандартами.
3. Розробити метод кількісного оцінювання рівня захищеності вебдодатків з можливістю адаптації до індивідуальних характеристик проєкту та врахування експертних оцінок показників безпеки.
4. Провести експериментальні дослідження застосування розробленого методу оцінювання рівня захищеності вебдодатку і дослідити їх узгодженість з результатами тестування на вразливості.
5. Розробити інформаційну систему підтримки запропонованого методу оцінювання рівня захищеності вебдодатків.
6. Провести верифікацію розробленої інформаційної системи щодо коректності її функціонування та зручності використання.

Об’єктом дослідження є процес оцінювання рівня захищеності вебдодатків.

Предметом дослідження є методи, моделі, критерії та інструменти оцінювання рівня безпеки вебдодатків.

Методи дослідження: галузеві стандарти та рекомендації для формування базових критеріїв безпеки, методи оцінювання рівня безпеки відповідно до усталених практик, методи математичного моделювання – для формалізації процесів аналізу загроз, вразливостей і ризиків у вигляді кількісних моделей, теорія нечіткої логіки та нечітких множин – для моделювання невизначеності при оцінці рівня ризиків та безпеки, методи системного підходу – для проєктування структури

інформаційної системи оцінки безпеки як складного програмно-технічного комплексу, методи програмної інженерії – для розробки, реалізації та тестування прототипу інформаційної системи.

Наукова новизна одержаних результатів. В результаті проведення досліджень одержано такі нові результати:

Розроблено новий метод кількісного оцінювання безпеки вебдодатку, який покладено в основу спроєктованої інформаційної системи, в рамках якого:

- Уперше формалізовано якісні вимоги стандарту OWASP Application Security Verification Standard (ASVS) у вигляді системи кількісних критеріїв оцінки захищеності, узгоджених з іншими галузевими стандартами, що дає змогу підвищити ефективність контролю та управління комплексною безпекою вебдодатків.

- Уперше обґрунтовано адаптивну процедуру вибору множини вимог та критеріїв в залежності від архітектури, функціональності продукту та доступу до документації процесу розробки вебдодатку, що дало можливість забезпечити універсальність підходу для оцінювання рівня захищеності вебдодатків.

- Уперше запропоновано використання системи нечіткої логіки для опрацювання експертних оцінок коефіцієнтів важливості вимог та критеріїв, що дає змогу врахувати невизначеність суджень експертів та забезпечити об’єктивність процесу оцінювання.

Практичне значення отриманих результатів дисертаційного дослідження полягає у створенні інформаційної системи для автоматизації запропонованого адаптивного методу кількісного оцінювання рівня захищеності , що враховує індивідуальні характеристики проєктів, мінімізує суб'єктивність та невизначеність експертних оцінок через застосування апарату нечіткої логіки, уніфікує перевірку відповідності вебдодатку вимогам стандарту OWASP ASVS та надає зрозумілий числовий індикатор захищеності для прийняття обґрунтованих управлінських рішень щодо вдосконалення системи захисту вебдодатків.

Особистий внесок здобувача. Наукові результати, викладені в дисертаційній роботі, отримані здобувачем особисто та є оригінальними. Теоретичні положення, концепції та гіпотези, запозичені з наукових праць інших дослідників, належним чином атрибутовані через систему посилань і застосовані виключно для обґрунтування та розвитку власних наукових здобутків автора.

Апробація результатів дисертації.

Наукові та практичні результати дисертаційного дослідження доповідалися та обговорювалися на міжнародних та всеукраїнських конференціях, зокрема на:

1. IX науково-технічній конференції «Інформаційні моделі, системи та технології», Тернопіль, 2021.
2. X науково-технічної конференції «Інформаційні моделі, системи та технології», Тернопіль, 2022.
3. XIV міжнародній науково-технічній конференції ITSEC: Безпека інформаційних технологій, 2025.

Структура та обсяг дисертації. Дисертаційна робота викладена на 173 сторінках та складається з анотації, змісту, переліку скорочень, вступу, чотирьох основних розділів, в яких міститься 32 рисунки та 10 таблиць, списку використаних джерел з 86 найменувань, а також 5 додатків. За структурою, мовою та стилем викладення дисертація відповідає вимогам МОН України. Робота написана грамотною українською мовою з використанням сучасної наукової термінології, а стиль викладення матеріалу є послідовним та логічним.

РОЗДІЛ 1. ПЕРЕДУМОВИ ФОРМУВАННЯ ПІДХОДІВ ДО ОЦІНЮВАННЯ БЕЗПЕКИ ВЕБДОДАТКІВ

1.1. Структура вебдодатків та вплив моделей SDLC на безпеку програмного забезпечення

1.1.1. Структура клієнт-серверної взаємодії у сучасних вебдодатках

Сучасні вебдодатки є складними багатокомпонентними системами, які включають динамічний контент, інтерактивні елементи, бази даних та інтеграцію з різноманітними сервісами та додатками. Така складність забезпечує вебдодаткам необхідну гнучкість і функціональність, оскільки вони складаються з численних розділів, підсторінок, модулів та API, які розширюють їхні можливості та забезпечують користувачам широкий спектр сервісів (див. рис. 1.1).

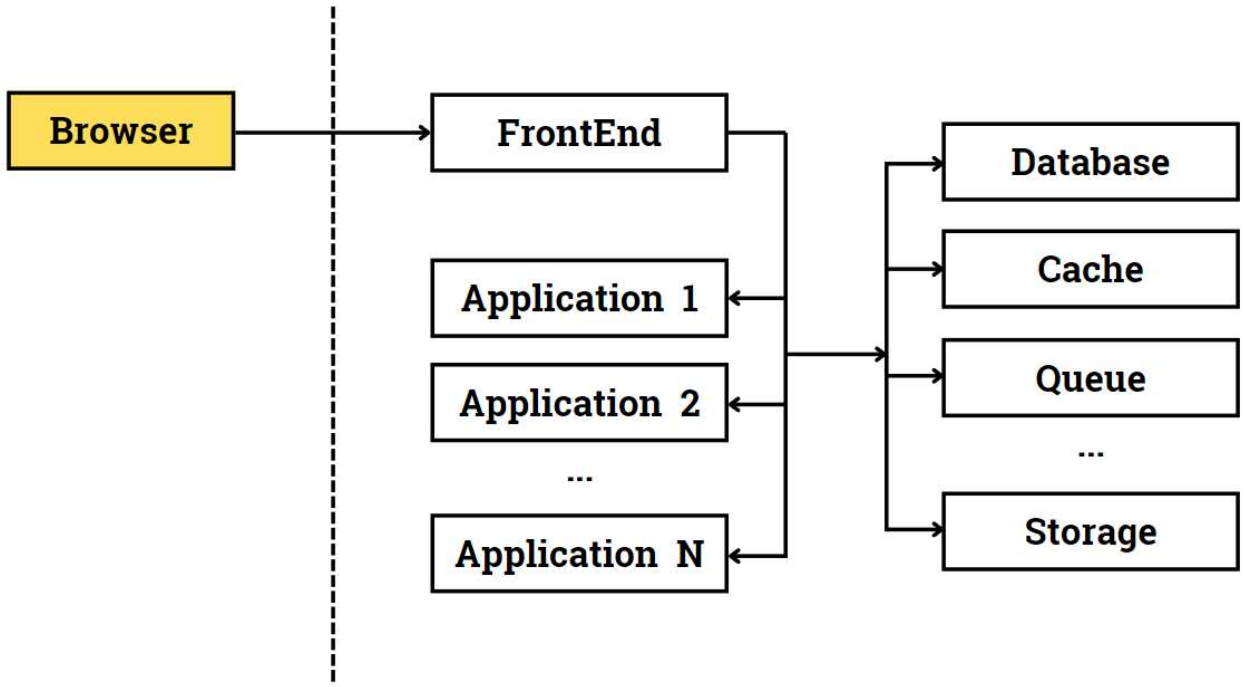

Рис. 1.1. Структура сучасного вебдодатку

Процес взаємодії між клієнтом і вебдодатком зазвичай починається через веббраузер, який взаємодіє з фронтенд частиною системи, що в свою чергу взаємодіє з серверними компонентами для обробки, збереження та передачі

необхідної інформації клієнту [1-3]. Комунікація між клієнтом і сервером часто здійснюється через HTTP-протокол, що дозволяє забезпечити передачу запитів і відповідей між різними компонентами додатку. Вебдодатки використовують різноманітні протоколи для забезпечення ефективної взаємодії між компонентами, такими як бази даних, де, наприклад, взаємодія з базою даних PostgreSQL може здійснюватися через "рідний" протокол, який підтримується на рівні TCP/IP [4]. Ці взаємодії дозволяють серверним компонентам забезпечувати надійний потік даних і підтримувати безперебійну роботу додатку.

Серверна частина сучасних вебдодатків виконує роботу, відповідальну за управління базою даних додатка та забезпечення безперервного потоку інформації між клієнтським інтерфейсом та сервером. Надійність та ефективність бекенду є важливими для забезпечення надійного користувацького досвіду. Крім того, архітектура серверної частини часто включає такі аспекти, як масштабованість, безпека та оптимізація продуктивності, щоб відповісти вимогам вебдодатків.

Отже, структура сучасного вебдодатку характеризується високим рівнем складності та багатокомпонентністю, що зумовлює необхідність ретельного підходу до процесу його розробки. Така складність вимагає не лише технічної узгодженості між клієнтською та серверною частинами, а й системного планування всіх етапів життєвого циклу програмного забезпечення. У зв’язку з цим доцільним є детальний розгляд основних етапів Software Development Life Cycle (SDLC), які забезпечують методичну базу для побудови надійних, масштабованих і безпечних вебдодатків. У подальшому розділі буде проаналізовано ключові фази SDLC та особливості їх реалізації в контексті створення веборієнтованих систем.

1.1.2. Основні етапи та особливості SDLC для вебдодатків

Життєвий цикл розробки програмного забезпечення (SDLC) являє собою концептуальну структуру, яка описує послідовність взаємопов'язаних етапів створення програмного продукту від початкової ідеї до завершення експлуатації [5]. За визначенням, наданим у фаховій літературі [6], SDLC охоплює основні етапи аналізу вимог, проєктування, розробки (реалізації), тестування, розгортання та

впровадження ПЗ. Кожен із етапів життєвого циклу має свої чітко визначені завдання: планування та аналіз визначає організацію проєкту (встановлення термінів, бюджету, ролей), з'ясовує функціональні та нефункціональні вимоги; проєктування визначає архітектуру системи; розробка включає написання коду; тестування гарантує відповідність реалізації специфікаціям. Завершення проходження всіх етапів призводить до формування кінцевого продукту з передбачуваними характеристиками. Зрозуміло, що розробка вебдодатків має певну специфіку (див. рис. 1.2).

Рис. 1.2. Життєвий цикл розробки (SDLC) вебдодатку

На початковому етапі здійснюється аналіз вимог, що передбачає визначення бізнес-цілей, технічних обмежень та функціональних очікувань від системи. До цього процесу залучаються Власник Продукту, Проєктний Менеджер, Бізнес-аналітик та Технічний директор. Особливу увагу приділяють сценаріям взаємодії користувачів, обсягам очікуваного навантаження, вимогам до безпеки й масштабованості, а також зовнішнім інтеграціям. Ретельний аналіз на цьому етапі закладає основу для ефективного проєктування та реалізації вебдодатку.

Наступним кроком є етап проєктування, у межах якого Системний архітектор та UX/UI дизайнер розробляють архітектурну модель системи та створюють концепцію інтерфейсу користувача. Архітектура включає визначення основних компонентів, їхню взаємодію, структуру баз даних, API-інтерфейси та інші технічні аспекти. UX/UI дизайн у вебдодатках відіграє критичну роль, адже він забезпечує не лише естетичну привабливість, а й високу зручність користування, адаптивність

до різних пристроїв та відповідність принципам доступності. Також на цьому етапі враховуються вимоги до інформаційної безпеки, продуктивності та майбутньої масштабованості системи.

Реалізація вебдодатку здійснюється на етапі розробки, в якому беруть участь Front-end та Back-end розробники. Front-end фахівці відповідають за створення інтерактивного користувацького інтерфейсу, використовуючи сучасні вебтехнології, такі як HTML5, CSS3, JavaScript або TypeScript разом із популярними фреймворками. Back-end розробники реалізують серверну логіку, забезпечують взаємодію з базами даних, реалізують API та механізми автентифікації й авторизації. Особливої уваги вимагає дотримання принципів безпечного програмування, що дозволяє запобігти поширеним вразливостям, зокрема SQL-ін’єкціям, XSS та CSRF-атакам.

Після завершення розробки виконується тестування, у якому задіяні Архітектор рішень, QA інженери, тестувальники та DevOps спеціалісти. У межах цього етапу проводиться функціональне, інтеграційне, навантажувальне та безпекове тестування. Значна увага приділяється автоматизації тестування для забезпечення швидкого зворотного зв’язку при зміні коду, а також відповідності вебдодатку сучасним стандартам якості. Особливо актуальним є тестування кросбраузерної та кросплатформної сумісності, а також перевірка системи на відповідність вимогам щодо захисту персональних даних.

У сфері веброзробки також активно застосовуються практики DevOps, зокрема безперервної інтеграції та доставки (CI/CD), що оптимізують життєвий цикл розробки, зокрема процес розгортання проєкту. Застосування цих практик дозволяє автоматизувати збірку та тестування при кожній зміні коду, що скорочує цикли розробки та спрощує процес розгортання нових версій системи [7]. Це забезпечує швидше оновлення вебдодатків і підвищує гнучкість команд розробників.

Остаточним етапом є впровадження розробленого продукту в робоче середовище з подальшим залученням кінцевих користувачів, тестувальників та менеджерів підтримки. На цьому етапі проводиться ознайомлення користувачів із

функціональністю системи, здійснюється моніторинг її поведінки в реальних умовах експлуатації, а також оперативно усуваються помилки, які могли залишитися непоміченими на попередніх етапах. Забезпечення технічної підтримки, збирання зворотного зв’язку та регулярне оновлення системи є критичними факторами для підтримання стабільності вебдодатку та його адаптації до змін у бізнес-вимогах або технічному середовищі.

Впровадження SDLC сприяє систематизації процесу розробки та забезпеченню якості готового продукту. Таким чином, SDLC є важливим інструментом в управлінні проєктами розробки інформаційних систем, оскільки дозволяє гарантувати завершення проєкту вчасно, у межах бюджету та з дотриманням встановлених стандартів якості. Крім того, цей підхід допомагає ефективно управляти ризиками та забезпечувати відповідність кінцевого продукту вимогам користувачів. Важливо також зазначити, що SDLC охоплює стадію експлуатації та підтримки, на якій програмний продукт вводиться в експлуатацію, а також вносяться необхідні зміни або доповнення в результаті зворотного зв'язку від користувачів. Різноманіття моделей SDLC деталізує порядок виконання етапів, які будуть розглянуті далі.

1.1.3. Традиційні моделі життєвого циклу розробки програмного забезпечення для вебдодатків

У контексті веброзробки основні принципи SDLC залишаються незмінними, однак процес розробки адаптується до специфіки та динаміки вебсередовища. Вебдодатки часто розробляються за ітеративними сценаріями, що передбачають часті релізи та швидке реагування на зміни вимог. Наприклад, модель Rapid Web Development Life Cycle (RWDLC) підкреслює гнучкий ітеративний підхід, пристосований до сучасних тенденцій веброзробки [8]. Існують різні моделі SDLC (каскадна, ітеративна, Agile тощо), кожна з яких має свої особливості, які можуть істотно впливати на ефективність та надійність розробки програмного забезпечення, особливо в сфері вебдодатків. Тому доцільно розглянути основні моделі SDLC, їхні переваги та недоліки зокрема.

Каскадна модель (Waterfall) є однією з найраніших моделей життєвого циклу розробки програмного забезпечення. Вона передбачає послідовне проходження чітко визначених етапів – від збору вимог до проєктування, реалізації, тестування та впровадження. Особливістю цієї моделі є відсутність зворотного зв'язку між фазами, що ускладнює внесення змін на пізніх етапах розробки. У контексті створення вебдодатків це може призводити до проблем із адаптацією до змін вимог користувачів або ринкових умов. Крім того, фіксований підхід до архітектури обмежує можливості гнучкої масштабованості та інтеграції сучасних технологій. Унаслідок цього каскадна модель підходить здебільшого для проєктів із добре визначеними, стабільними вимогами та обмеженою складністю [9].

V-модель (Verification and Validation Model) є вдосконаленою версією каскадної моделі, яка запроваджує чітку відповідність між етапами розробки та відповідними фазами тестування. Її V-подібна структура забезпечує покращену контрольованість процесу, дозволяючи планувати та виконувати тестування паралельно з проєктуванням. Це сприяє ранньому виявленню дефектів і забезпечує вищу якість кінцевого продукту, що є важливим при створенні складних вебдодатків [10].

Для кожного етапу розробки передбачений відповідний тип тестування: наприклад, системне тестування відповідає системному проєктуванню, а модульне – деталізованому дизайну. Такий підхід дозволяє краще відстежувати, як реалізація відповідає початковим вимогам, та своєчасно виявляти помилки на кожному рівні.

Водночас, як і каскадна модель, V-модель залишається лінійною, що ускладнює внесення змін на пізніх етапах. У випадку змін вимог або потреби у зміні архітектури вебдодатку, доводиться повертатися до попередніх фаз, що може бути затратним і малоефективним. Крім того, модель менш пристосована до гнучкої інтеграції нових технологій чи швидких ітерацій, що обмежує її застосування в умовах динамічного розвитку вебсервісів.

Спіральна модель (Spiral), запропонована Баррі Боемом у 1986 році, поєднує структурованість каскадної моделі з гнучкістю ітеративного підходу. Її основною особливістю є цикл, що повторюється, кожна ітерація якого включає етапи

планування, проєктування, розробки, тестування та оцінки результатів. Такий підхід дозволяє поступово розширювати функціональність продукту, адаптуючись до змін вимог і технологічного середовища, що особливо актуально для розробки вебдодатків [11].

Завдяки постійним ітераціям, модель дозволяє гнучко реагувати на нові побажання замовника, вносити зміни в архітектуру або інтерфейс на ранніх етапах, не порушуючи загальну логіку розробки. Це робить спіральну модель ефективною для складних, довготривалих або інноваційних вебпроєктів, де остаточні вимоги можуть уточнюватися у процесі.

Крім того, модель добре поєднується з сучасними підходами до управління розробкою, такими як DevOps чи Agile, дозволяючи створювати MVP-прототипи, регулярно збирати зворотний зв’язок і вдосконалювати продукт по мірі його розвитку.

Однак для ефективного застосування спіральної моделі необхідна висока зрілість процесів і чітке визначення цілей кожної ітерації. Без належної організації та документації можливі повтори, зростання складності проєкту або неузгодженість між окремими етапами. Крім того, ця модель менш формалізована порівняно з традиційними підходами SDLC, що може ускладнити її впровадження в компаніях із фіксованими бюджетами, ресурсами чи жорсткими термінами [12].

Інкрементна модель є гнучким підходом до розробки програмного забезпечення, що передбачає поетапне створення системи шляхом послідовного додавання функціональних компонентів (інкрементів). Кожен інкремент охоплює повний життєвий цикл – від аналізу вимог і проєктування до реалізації й тестування – і додає новий функціонал до вже існуючої версії продукту. Такий підхід дає змогу швидко створити початкову версію вебдодатку (наприклад, MVP) і поступово розширювати його функціональні можливості відповідно до зворотного зв’язку користувачів та змін бізнес-вимог.

Інкрементна модель особливо ефективна у веброзробці, де часто необхідна швидка адаптація до нових функцій, тенденцій або інтеграцій. Вона дозволяє

паралельну розробку окремих модулів, а також пришвидшує час виходу продукту на ринок за рахунок раннього розгортання ключового функціоналу.

Однак успішне застосування інкрементної моделі вимагає ретельного планування залежностей між інкрементами. Функціонально пов’язані модулі (наприклад, обробка профілю користувача та система повідомлень) мають бути розроблені узгоджено, щоб уникнути дублювання логіки або конфліктів між компонентами. Крім того, важливо дотримуватися єдиних архітектурних рішень, щоб зберігати цілісність і масштабованість вебдодатку [13].

Модель швидкої розробки (RAD, Rapid Application Development) орієнтована на максимально швидке створення програмного забезпечення завдяки ітеративному прототипуванню, активній участі кінцевих користувачів та широкому застосуванню засобів автоматизації. У контексті веброзробки ця модель є особливо ефективною для швидкого реагування на зміну вимог, запуску MVP або створення функціоналу з високим пріоритетом у стислі терміни [14].

Основою RAD є створення інтерактивних прототипів, які швидко проходять цикл розробки, тестування та вдосконалення на основі зворотного зв’язку. Такий підхід дозволяє гнучко адаптувати вебдодаток до змін, що надходять від користувачів або бізнес-замовників, і водночас підвищує ймовірність відповідності очікуванням цільової аудиторії.

Однак швидкість розробки в моделі RAD часто супроводжується певними викликами. Зокрема, може бракувати глибокого архітектурного проєктування, що в перспективі призводить до необхідності масштабного рефакторингу. Через фокус на прототипах нерідко створюється обмежена або фрагментарна технічна документація, що ускладнює подальшу підтримку вебдодатку або передачу проєкту іншим командам.

Крім того, ефективне впровадження RAD-моделі потребує високої кваліфікації команди, яка здатна одночасно працювати в умовах швидких змін, забезпечувати стабільну інтеграцію нового функціоналу та підтримувати узгодженість між модулями вебдодатку. У великих проєктах із кількома командами

важливо координувати розробку, щоб уникнути дублювання рішень і забезпечити цілісність системи.

Зростання складності архітектури програмного забезпечення та ускладнення життєвого циклу його розробки прямо корелює зі збільшенням кількості потенційних векторів атак, зумовлених наявністю архітектурних і логічних вразливостей. Багаторівневі залежності між компонентами системи, а також недостатня інтеграція механізмів забезпечення безпеки на ранніх етапах проєктування, створюють передумови для формування критичних «вузьких місць» у структурі додатку. Ці вразливі точки часто залишаються поза увагою традиційних підходів до тестування, що знижує загальну ефективність захисту. Представлені у наступному підрозділі статистичні дані підтверджують безпосередній зв’язок між складністю архітектури вебдодатків та ймовірністю виникнення системних вразливостей.

1.2 Аналіз розповсюджених вразливостей вебдодатків за статистичними даними

За таких умов особливої актуальності набуває систематичне вивчення виявлених вразливостей шляхом їх класифікації та аналізу реальних сценаріїв експлуатації, що дозволяє точніше оцінити рівень ризику для складних архітектурних конфігурацій. Один із джерел таких даних – це звіт компанії Sucuri за 2023 рік [15], який надає детальну інформацію щодо основних факторів ризику для вебдодатків, зокрема неправильних налаштувань компонентів системи, використання застарілих програмних засобів і вразливих плагінів.

Згідно зі звітом, основними причинами компрометації вебдодатків є слабкі або неправильно налаштовані компоненти системи. Серед найпоширеніших факторів ризику виокремлюються: неналежна конфігурація сервера та вебсайту, порушення контролю доступу, а також використання застарілого програмного забезпечення та вразливих плагінів чи тем. Особливо це стосується популярних платформ для розробки сайтів, таких як WordPress. За даними Sucuri, абсолютну

більшість скомпрометованих вебсайтів становили саме ресурси на платформі WordPress, яка займає близько 62,8% ринку систем керування контентом, згідно з W3Techs [16]. Зокрема, Sucuri показав, що 95,5% усіх виявлених вразливостей припадають саме на сайти, побудовані на WordPress. Для порівняння, інші системи керування контентом, такі як Joomla (1,7%) і Magento (0,6%), мали значно менший відсоток інфікованих ресурсів (див. рис. 1.3).

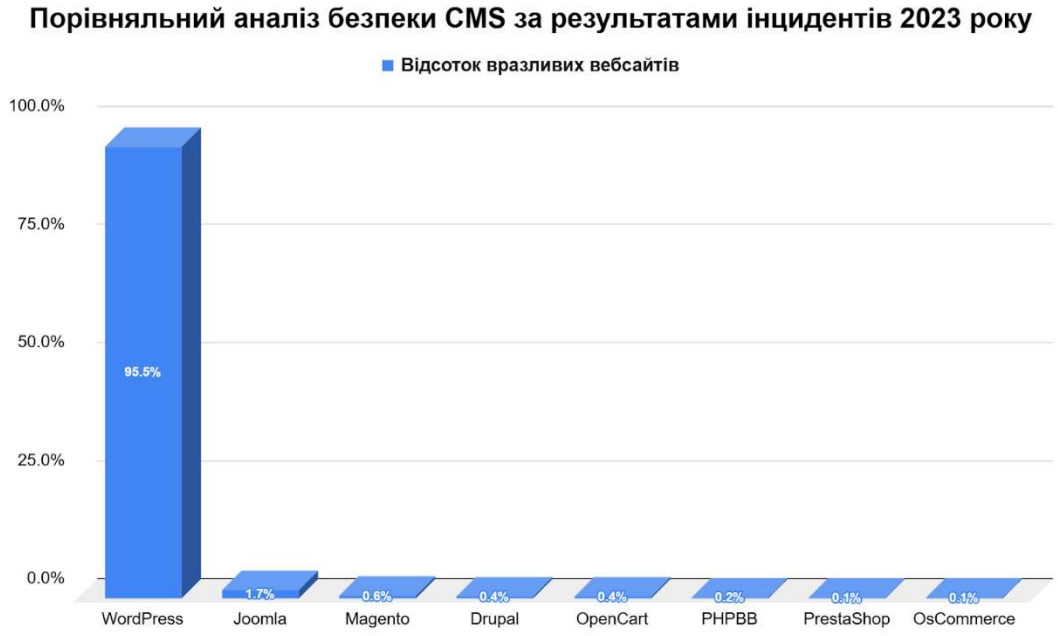

Рис. 1.3. Порівняльний аналіз безпеки CMS за результатами інцидентів 2023 року

Це свідчить про прямий зв'язок між популярністю системи керування контентом і спектром атак, оскільки велике поширення WordPress, разом із величезною кількістю плагінів, значно збільшує "атакувальну поверхню" цієї платформи. Саме тому платформи, які використовують WordPress, стають особливо вразливими до зловмисних атак. Ключові статистичні показники за 2023 рік, що підтверджують цю тенденцію, наведені у звіті компанії Sucuri (див. рис. 1.4).

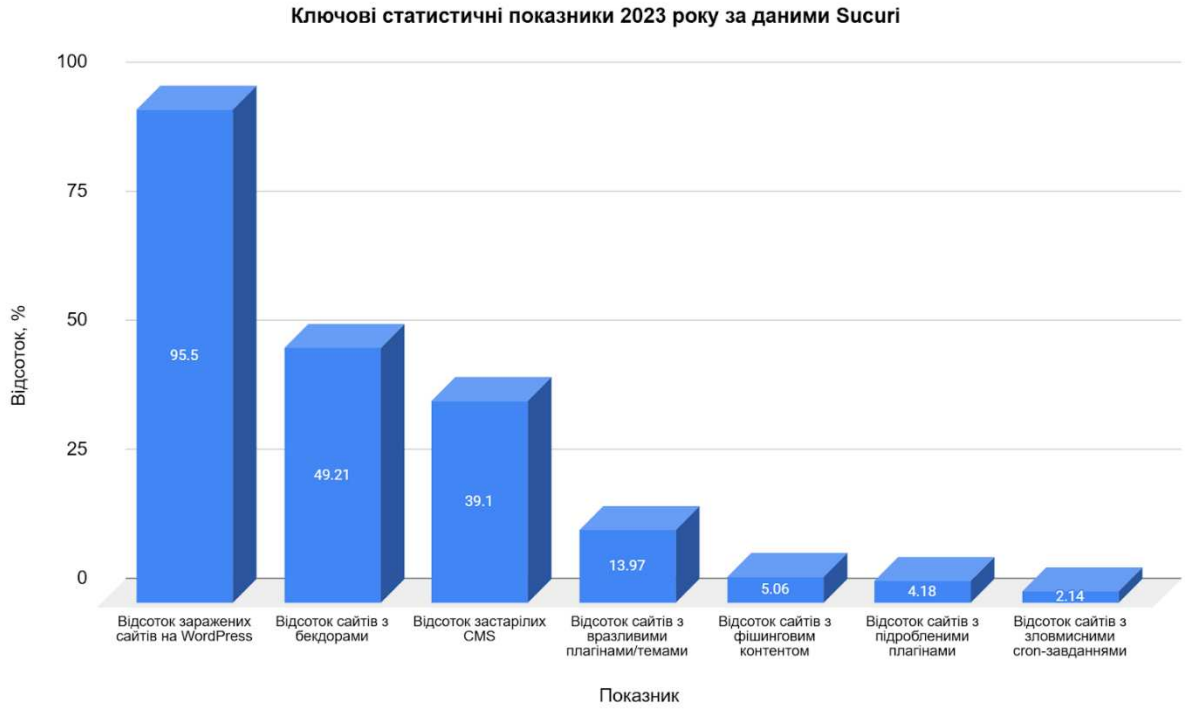

Рис. 1.4. Ключові статистичні показники 2023 року за даними Sucuri

Ці дані демонструють значну концентрацію вразливостей, що можуть виникнути в будь-якій системі керування контентом. Зокрема, близько 49,21% заражених вебдодатків містили бекдори, що дозволяють зловмисникам зберігати контроль над системою протягом тривалого часу. Це підкреслює важливість своєчасного виявлення та усунення таких загроз.

Значну частину заражених сайтів складають системи керування контентом, які на момент атаки були застарілими. Так, 39,1% таких систем не були оновлені до останніх версій, що створює потенційні вразливості для атак. Окрім того, 13,97% сайтів містили принаймні один відомий вразливий плагін або тему на момент очищення, що також підвищує ризик інфікування.

Додатково до вищезгаданих атак, існують й інші типи загроз. Наприклад, 5,06% заражених сайтів виявилися зараженими фішинговим контентом, зокрема через фальшиві форми авторизації, що ставить під загрозу конфіденційність користувачів. Ще 4,18% сайтів були уражені підробленими плагінами, що також є значним джерелом ризику для безпеки. Також було виявлено що, 2,14% сайтів мали

зловмисні cron-завдання, які автоматично запускали шкідливе програмне забезпечення, що підкреслює важливість належного контролю за автоматизованими процесами на сайтах.

Іншим суттєвим фактором є поширення SEO-спаму: 38,3% компрометованих баз даних містили SEO-спам, здебільшого у вигляді посилань на незаконні товари. Зокрема, гральний SEO-спам був виявлений на 87,201 сайтах, що втричі більше, ніж у 2022 році. Атака через SEO-спам спричиняє серйозні наслідки в репутаційній площині. Уражені ресурси засмічуються неконтрольованим вмістом, що призводить до зниження їх рейтингу в пошукових системах або повного виключення з індексації. Відповідно, органічний трафік значно падає, а частина потенційної аудиторії втрачається. До того ж, браузери та пошукові системи можуть видавати попередження про небезпеку, що негативно впливає на довіру користувачів.

Категорія Vulnerable and Outdated Components також підтверджується статистикою зі звіту, щодо застарілих систем та вразливих плагінів. Це свідчить про значну роль оновлень і використання безпечних компонентів для забезпечення належного рівня безпеки.

Також звіт Sucuri відзначає випадки порушення категорії Authentication and Identity, що стосується проблем управління аутентифікацією. Зокрема, 55% сайтів з вразливими базами даних мали заздалегідь створені зловмисні адміністративні облікові записи. Це підтверджує наявність серйозних проблем у механізмах автентифікації та управлінні користувачами.

Загалом, статистика Sucuri підтверджує, що більшість вразливостей вебдодатків стосуються контролю доступу, налаштування безпеки та оновлення компонентів. Експлуатація цих вразливостей тягне за собою різноманітні негативні наслідки, які охоплюють як технічні, так і організаційно-фінансові аспекти.

1.3 Огляд сучасних підходів до оцінювання безпеки вебдодатків

На сьогоднішній день існує низка наукових досліджень, які охоплюють різні аспекти забезпечення безпеки вебдодатків, а також загальних систем управління інформаційною безпекою (ISMS). Одним з важливих напрямів досліджень є аналіз ефективності контролю систем інформаційної безпеки на основі міжнародних стандартів. Наприклад, дослідження, присвячене впровадженню методів вимірювання ефективності контролів у системах управління інформаційною безпекою [17], базується на стандартах ISO/IEC 27004:2013 [18] та ISO/IEC 27002:2013 [19]. Автори цієї роботи розробили набір метрик та індикаторів для оцінки того, наскільки успішно організація впроваджує заходи безпеки та контролює їх ефективність. Це дослідження підкреслює важливість кількісного підходу до вимірювання результатів безпекових заходів і надає конкретні рекомендації щодо впровадження цих методів у практичну діяльність організації. Проте визначені метрики є загальними та універсальними для інформаційних систем та не адаптованими для оцінки безпеки. Ці стандарти визначають моделі якості, методи вимірювання та критерії оцінки програмних продуктів, зосереджуючись на функціональності, надійності, ефективності, зручності використання, підтримуваності тощо. Проте безпека в цих стандартах фактично не розглядається як окрема характеристика чи метрика, і відповідних методів кількісної оцінки безпекових аспектів у SQuaRE немає [20].

Існують дослідження [21], спрямовані на оцінку відповідності безпеки сервісів існуючим стандартам та регуляціям. Такі дослідження розглядають впровадження системного підходу до забезпечення відповідності стандартам інформаційної безпеки, зокрема у фінансовій та інших високоризикових сферах. Використовуючи методи динамічного та статичного аналізу, автори пропонують метод, який дозволяє автоматизувати процеси перевірки на відповідність вимогам безпеки. Це дослідження показало ефективність впровадження таких підходів на прикладі онлайн-платіжних сервісів, що допомагає підвищити захищеність даних і

відповідність сервісів до міжнародних стандартів, таких як PCI-DSS [22] та ISO 27002.

Незважаючи на наявність досліджень, що стосуються безпеки інформаційних систем, вони здебільшого не орієнтовані на оцінку безпеки саме програмного забезпечення, зокрема вебдодатків[23,24]. Водночас існують стандарти та методи, які зосереджуються на оцінці якості програмного забезпечення, однак безпека в них розглядається лише опосередковано або ж зовсім не враховується як окрема характеристика. Існують також різні рекомендації та методи забезпечення безпеки, але наразі відсутня єдина, узгоджена система, яка б дозволяла комплексно, з урахуванням архітектурних особливостей вебдодатка та його життєвого циклу розробки, якісно та кількісно оцінити рівень його захищеності. Таким чином, відсутність комплексного підходу до оцінки безпеки вебдодатків на всіх етапах їхнього життєвого циклу залишається актуальною проблемою в наукових дослідженнях і практиці.

Ще одним важливим напрямком є розробка концептуальних моделей для класифікації вебдодатків на основі їх операційної та поточної критичності. Наприклад, у дослідженні [25] було запропоновано використання двох ключових показників: операційної критичності (OC) та критичності даних (DC). Це дозволяє організаціям більш точно визначати вимоги до безпеки залежно від важливості вебдодатків для операційної діяльності компанії та конфіденційності обробки даних. Такий підхід дає змогу уникати надмірних витрат на непотрібні заходи безпеки і водночас забезпечувати адекватний рівень захищеності критичних для організації додатків. Та попри все, часто виникають ситуації - коли власник інтернет-магазину може змінити саму концепцію магазину, замінити методи оплати та взагалі повністю змінити власний продукт. І після імплементації нового функціоналу - знову ж таки виникає потреба оцінити та проаналізувати поточний рівень захищеності вебдодатку та наявності потенційних вразливостей, якими можуть скористатись кіберзлочинці.

З огляду на постійне зростання загроз для безпеки вебдодатків, з'являється дедалі більше стандартів, спрямованих на забезпечення інформацією належного

рівня захисту інформаційних систем. Однією із таких систем є Common Weakness Enumeration (CWE) [26]. Це загальнодоступний реєстр типових слабких місць у програмному забезпеченні, створений для полегшення ідентифікації, аналізу та управління вразливостями у програмних системах. CWE був розроблений та підтримується організацією MITRE у співпраці з експертами у галузі кібербезпеки.

Однією з основних переваг CWE є його універсальність. Він не обмежується виключно вебдодатками, а охоплює всі типи програмного забезпечення. Більше того, CWE застосовується на всіх етапах життєвого циклу розробки програмного забезпечення: від етапу проектування та кодування до тестування, впровадження та експлуатації. Такий підхід робить його цінним інструментом не лише для окремих проєктів, а й для організаційного управління ризиками.

Попри свою потужність та гнучкість, CWE має певні обмеження. Він не надає кількісної оцінки рівня захищеності програмного забезпечення, а лише ідентифікує потенційні вразливості. Іншими словами, CWE слугує базою для аналізу ризиків, але не дає прямої відповіді на питання, наскільки безпечним є вебдодаток.

Класифікацією сучасних вразливостей займається спільнота OWASP (Open Web Application Security Project). Це міжнародна некомерційна організація, що спеціалізується на аналізі та покращенні безпеки програмного забезпечення, члени якої виділяють найнебезпечніші вразливості на основі статистики та досліджень, що дозволяє їм зосередитися на найбільш критичних аспектах безпеки програмного забезпечення. Дослідження спільноти OWASP є важливим джерелом інформації для розробників та фахівців з безпеки програмного забезпечення, оскільки воно дозволяє виявляти та усувати потенційні загрози для безпеки вебдодатків на ранніх етапах їх розробки.

У 2003 році OWASP вперше запропонувала список найнебезпечніших вразливостей вебдодатків, який називається "OWASP Top Ten". Усі продукти цієї спільноти є безкоштовними та з відкритим кодом. З ними співпрацюють тисячі компаній та спеціалістів по всьому світу, які використовують його як стандартний інформаційний документ для тестувальників та розробників вебдодатків. Дослідження спільноти OWASP є важливим джерелом інформації для розробників

та фахівців з безпеки програмного забезпечення, оскільки дозволяє виявляти та усувати потенційні загрози для безпеки вебдодатків на ранніх етапах їх розробки.

Щороку перелік вразливостей переглядається, але на практиці оновлення відбувається раз в три-чотири роки. Якщо це відбувається частіше, це означає, що атаки на вебдодатки виходять на новий рівень, їх стає все більше і потрібна відповідна реакція спільнот по всьому світу. Вразливості у цьому списку впорядковані за частотою появи, тому їхні позиції в списку змінюються, також завдяки появі нових загроз.

Останнє оновлення переліку пріоритетних вразливостей вебдодатків відбулося у 2021 році (див. рис. 1.5).

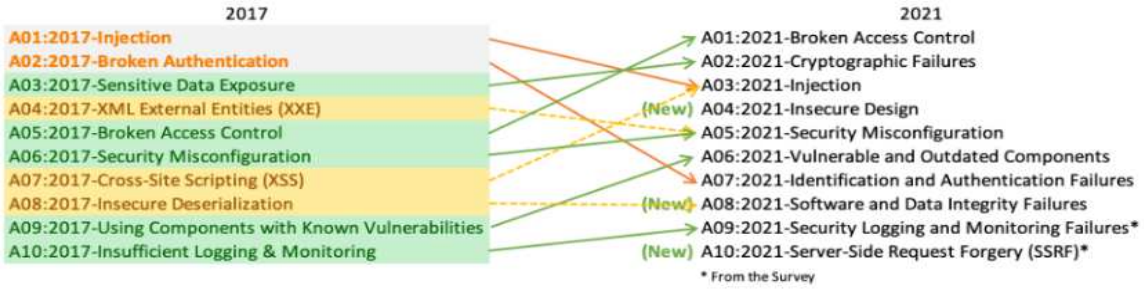

Рис. 1.5. Оновлений список загроз у 2021 році

З рисунку 1.5 видно, що частина вразливостей була трансформована і з'явилась під новими назвами, але з'явилася нові вразливості [27-29].

Як бачимо, статистичні дані, наведені у попередньому підрозділі, підтверджують і конкретизують пріоритетні напрями вразливостей, виділені в оновленому переліку OWASP Top 10 (2021). Зокрема, часті випадки неналежної конфігурації серверів і систем, описані у звіті Sucuri, співпадають із категоріями «Security Misconfiguration» та «Vulnerable and Outdated Components», що займають високі позиції у рейтингу. Також поширеність проблем із контролем доступу та автентифікацією, виявлена у статистиці (55% сайтів з шкідливими адміністративними обліковими записами), чітко корелює з категоріями «Broken Access Control» та «Identification and Authentication Failures» OWASP.

Виявлені в статистиці такі загрози, як бекдори, фішинговий контент, шкідливі cron-завдання і SEO-спам, ілюструють актуальність нових чи перетворених вразливостей, зокрема тих, що пов’язані із захистом даних та логуванням подій. Таким чином, аналіз статистики підтверджує актуальність і доцільність використання оновленого списку OWASP як базового орієнтиру для оцінки ризиків та розробки заходів з безпеки вебдодатків.

За останні роки спостерігається зростання випадків, пов'язаних з вразливістю "Insecure Design". Ця вразливість посідає четверте місце в переліку що свідчить про її значний вплив на безпеку вебдодатків. Короткий огляд витоків даних, наведений нижче, дає змогу зрозуміти ландшафт загроз та методів, використаних зловмисниками для здійснення атак.

Insecure Design - це вразливість, що є характерною для планування проєкту та його архітектури, тобто розглядається в рамках моделі безпечного проєктування вебдодатків. Від вразливості не вдасться уникнути декількома стрічками коду, як у випадку із ін'єкціями, що може мати непередбачувані наслідки для всього вебдодатку, а не окремого компонента. Головною причиною виникнення вразливості є неправильне моделювання загроз, яке виконується на етапі проєктування програмного забезпечення та поширюється аж до завершення проєкту. Вразливість може виникнути під час розробки вебдодатку, коли розробник вимикає важливі для безпеки функції під час тестування програмного забезпечення і забуває увімкнути їх знову. Ще однією причиною може бути надмірна інформація, отримана через валідування HTTP-запитів для клієнта або занадто просте підтвердження особи за допомогою легко прохідних секретних запитань.

Одним із завдань, що виникає перед командою розробки, є планування проєкту, яке має враховувати вимоги інформаційної безпеки з одного боку, а також те, що усі бізнес-процеси і функціонал не конфліктували між собою, запобігаючи виникненню "логічних" помилок. У зв’язку з цим Insecure Design вразливість може виникнути через недостатній бюджет проєкту, недостатньо продуману бізнес-логіку та невірну постановку завдань для розробників. Таким чином, вебдодаток

може бути безпечним на певний момент, а після додаткових змін у функціональних можливостях програмного забезпечення відкриваються нові можливості для атак на вебдодаток.

Огляд публікацій [30] показує, що вразливість Insecure Design досі є слабо формалізованою, а рекомендації щодо захисту можуть виглядати дещо абстрактно. Організація OWASP не лише вивчає та визначає списки найбільш популярних категорій вразливостей, але й надає певні рекомендації щодо їх усунення [31], зокрема:

- Використовувати безпечний цикл розробки та залучати професіоналів із безпеки вебдодатків для оцінки та проєктування засобів безпеки та приватності.
- Створювати та використовувати бібліотеку безпечних шаблонів дизайну.
- Використовувати моделювання загроз для важливих елементів бізнес-логіки.
- Введення перевірок на кожному рівні вебдодатку.
- Покриття модульними та інтеграційними тестами всього додатку.
- Розподіл на рівні системи та мережі, відповідно до потреб їх експозиції та захисту.
- Розподіл функціональності на всіх рівнях під час дизайну.
- Обмеження витрат ресурсів за користувачем чи службою.

Багато дослідників посилаються на цей список, який безперечно є корисним і розробленим на основі багаторічних досліджень та статистичних даних. Більшість авторів роблять акцент на безпеці розробки (SDL) та залученні спеціалістів з веббезпеки.

1.4 Інтеграція концепції Secure Development Lifecycle у процес забезпечення інформаційної безпеки вебдодатків

Для забезпечення належного рівня безпеки в процесі розробки вебдодатків необхідно впроваджувати практики, що гарантують надійний захист на кожному етапі життєвого циклу продукту. Однією з таких практик є концепція Secure

Development Lifecycle (SDL), яка передбачає комплексний підхід до розробки безпечних додатків. Цей підхід включає чітко визначені вимоги до програмного продукту на всіх етапах його життєвого циклу, починаючи від проєктування та розробки, через етапи тестування, сертифікації та впровадження, і до постійної підтримки і оновлення в процесі експлуатації. SDL спрямована на забезпечення високого рівня безпеки додатка шляхом інтеграції принципів безпеки на всіх етапах його створення, що дозволяє мінімізувати вразливості та захистити додаток від потенційних загроз [32-35]. Це допомагає забезпечити надійний захист не тільки під час розробки, а й протягом всього життєвого циклу продукту, що зображує схема процесу (див. рис. 1.6).

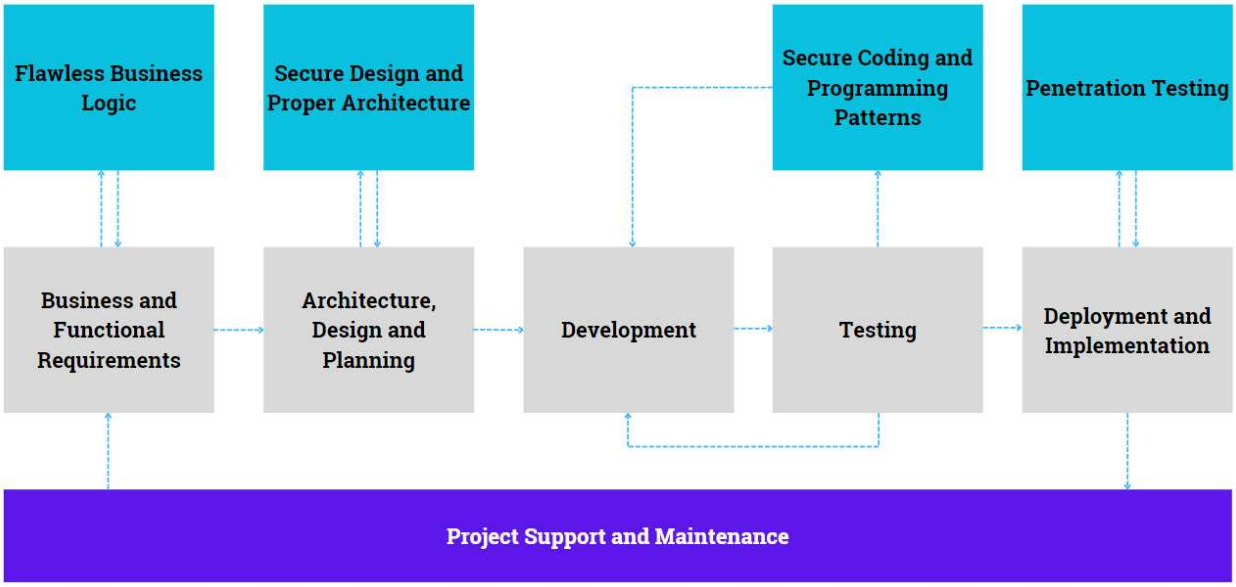

Рис. 1.6. Secure Development Lifecycle

Завдяки використанню SDL, розробники можуть системно підходити до безпеки на всіх етапах життєвого циклу додатку, починаючи від етапу проєктування до його виведення з експлуатації. Концепція SDL включає в себе залучення фахівців з безпеки, що дозволяє виявляти вразливості ще на етапі кодування і тестування. Це дає можливість оперативно виправляти потенційні проблеми і перевіряти відповідність програмного забезпечення встановленим стандартам безпеки. Важливим аспектом SDL є постійний моніторинг системи

після її впровадження, що дозволяє вчасно виявляти нові загрози і оновлювати захист від них. Цей підхід допомагає створити безпечне та стабільне середовище для користувачів, що, у свою чергу, гарантує надійність і безпеку вебдодатків на всіх етапах їхнього існування.

З моменту появи SDL поповнився та вдосконалювався новими рекомендаціями, стандартизацією та успішними практиками на основі результатів роботи команд з усього світу. Оскільки вразливість "Insecure Design" займає високу позицію у переліку пріоритетних вразливостей, це обумовило необхідність у цій роботі ретельно розглянути існуючі та доповнити новими підходами для захисту вебдодатків на кожному етапі SDL.

Першими важливими етапами безпечного циклу розробки є визначення бізнес- та функціональних вимог, а на їх основі - планування безпечної архітектури проєкту. Під час планування архітектури важливо враховувати майбутні ризики, можливість розширення проєкту та правильність підходу до обміну інформацією між компонентами, особливо при мікросервісній архітектурі. Дотримання балансу між функціоналом, який необхідний клієнту, та тим, що може пропонувати команда розробки, є важливим застереженням, оскільки перенасичення функціоналом може відкрити доступ до вразливостей "Insecure Design".

Етап розробки - це відповідальний етап, який настає після планування проєкту. Часто розробники, які будуть виконувати написання коду помилково думають про те, що до цього етапу все розплановано, всі схеми і функціональні вимоги враховані - а отже з інформаційною безпекою на проєкті все гаразд. Це може призвести до ситуації, коли засоби захисту вебдодатку не імплементовані або залишаються на неналежному рівні. З огляду на це було розроблено рекомендації, наведені нижче у цьому розділі, щодо аналізу ризиків і загроз, та підсилення захисту на етапі розробки.

Доступність дискового простору для звичайних користувачів. Якщо не брати до уваги хмарні сховища, а розглянути більш простіші варіанти реалізацій - то на будь якому вебдадатку присутні зображення контенту, файли для завантаження тощо. Важливо обмежити доступ до всієї файлової системи та виділити окремі

публічні логічні простори для взаємодії користувачів із файловою системою. Таким чином можна закрити вразливість неконтрольованого виконання завантажувальних файлів, а також - обмежити режим доступу (про який згадувалось в OWASP).

Захист запитів для взаємодії з конкретною сутністю. Перш за все - ніколи в запитах не використовувати зрозумілі і передбачувані ідентифікаторів записів сутностей, підміною яких можна отримати доступ до іншої інформації в проєкті. Попри всі застереження і приклади атак, використання такої практики - далеко не рідкість. В такому випадку варто користуватись додатковими захистом за допомогою “Middleware” різного типу. Простими словами - це захист під час запиту до серверу, який займає місце між початком запиту та виконанням дій конкретним класом. Варто перевіряти, чи має місце легітимність цього запиту, чи може клієнт взаємодіяти з записом, якого стосується запит та взагалі чи існує такий запис.

Валідації прав на виконання того чи іншого запиту. Часто розробники припускаються логічної помилки, стараючись чітко показати користувачу, що він робить не так в даний момент часу, аби йому було легше виправити свій ввід та отримати бажаний результат. Прикладом може бути детальний опис того, що введено невірно під час авторизації користувача. Також часто буває розмежування прав між функціоналом і замість того, щоб видати 404 помилку - розробники вказують 403. А для зловмисника це означає що він знайшов об'єкт, який буде атакувати. Іншим прикладом є CMS Wordpress - завдяки використання своїх CSS-класів з префіксом “wp-” та стандартизованої точки авторизації “wp-login” - зловмисники завжди знають з чим мають справу.

Безпека використання API та бібліотек. Розробники часто використовують для полегшення роботи уже написаний функціонал, однак не замислюються, чи проходить цей компонент SDL, чи оновлюється код та чи є підтримка розробників. Необачність у цьому питанні призводить, зокрема, до “атак на ланцюг постачань”.

Стандартизація та патерни програмування. Використання патернів програмування та принципів, таких як SOLID забезпечить написання якісного

коду. Під час правильних підходів до програмування не буде виникати проблем із випадковими змінами коду, залишенням “бекдорів”, консольних помилок та інших небажаних наслідків. Використання експериментаторами нових підходів, відхилень від загальноприйнятих практик та набагато складніші рішення простих задач можуть спричинити проблеми, які пов’язані із “Insecure Design”.

Етап тестування продукту. Якість покриття коду тестами - дуже залежить від вищесказаного про патерни програмування. Чим інтуїтивно простіше написаний код - тим легше його тестувати в майбутньому. Важливим є тестування бізнес-логіки, закладеної на початку проєкту. Часто розробники “запевняють” в тому, що “все працює, по іншому бути не може” і тестування виконується на довірі та за результатом покриття тестами.

Під час етапу розгортання дуже важливо розуміти весь “пройдений шлях” вебдодатком, його сутність і функціональні вимоги, сховища чи конфігураційні файли проєкту після його завершення можуть містити доступи, тестові чи чутливі дані. Перевірка конфігураційних файлів та сховищ на предмет конфіденційної інформації дасть змогу уникнути потенційної компрометації вебдодатку чи даних користувачів.

Підтримка проєкту включає в себе моніторинг та постійне оновлення програмного забезпечення, а також отримання відгуків від користувачів та аналіз поведінки вебдодатку в різних умовах експлуатації. На рисунку 1.7 подано оновлену схему SDL процесу, підсилену рекомендаціями та аналізом, наведеним вище у цьому розділі.

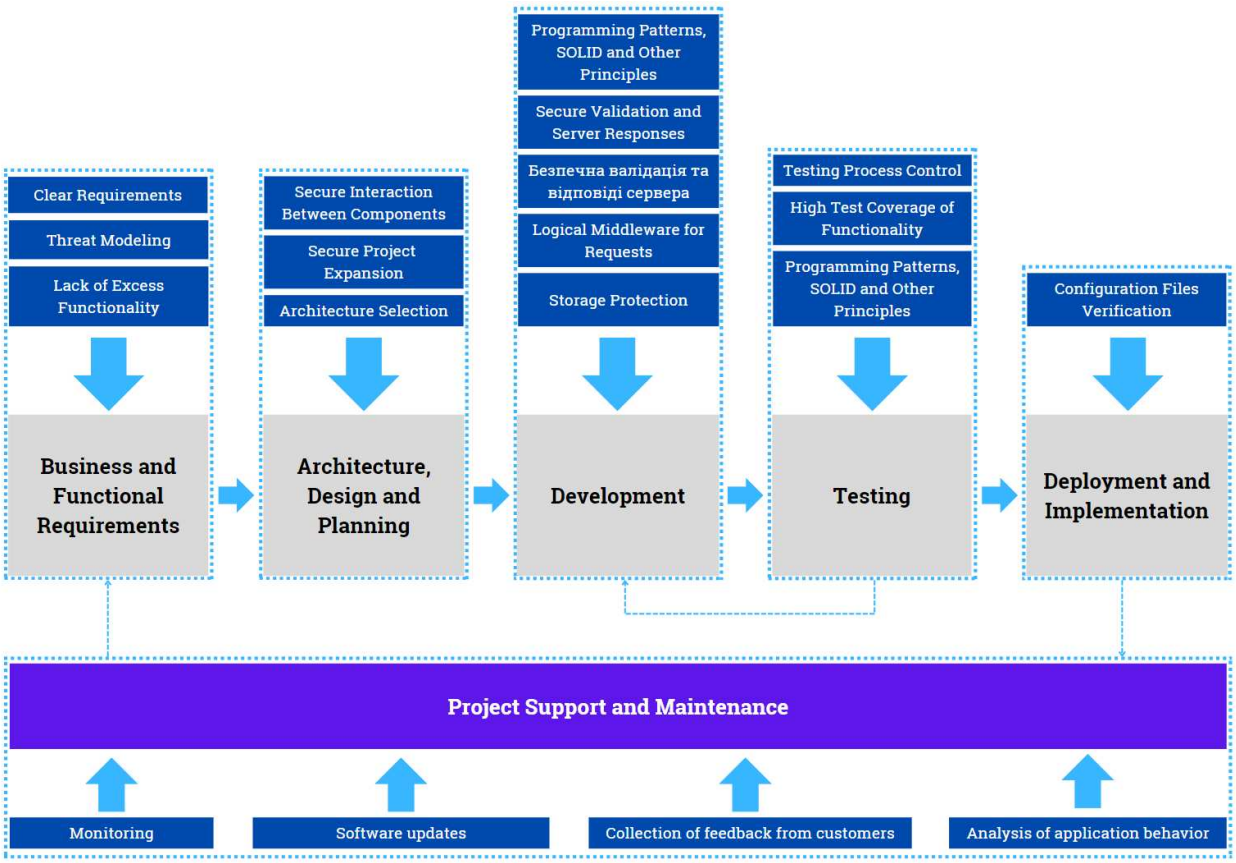

Рис. 1.7. Доповнення до Secure Development Lifecycle

Незалежно від обраної моделі SDLC, безпека вебдодатків має бути невід'ємною складовою кожного етапу розробки. Традиційні лінійні моделі, такі як каскадна чи V-модель, демонструють обмеження у цьому аспекті: безпекові вимоги часто враховуються лише на етапі тестування, що призводить до запізнілого виявлення вразливостей і підвищеного ризику їх експлуатації. Відсутність системного моделювання загроз, статичної та динамічної перевірки на ранніх фазах ускладнює впровадження підходу secure-by-design.

Натомість ітеративні моделі – зокрема спіральна, інкрементна та RAD – створюють більше можливостей для інтеграції безпеки як безперервного процесу. Завдяки повторюваним циклам проєктування, реалізації та перевірки, стає можливим поступове впровадження механізмів захисту, таких як автентифікація, контроль доступу, валідація введення, а також застосування практик threat modeling, OWASP ASVS, STRIDE, SAST, DAST тощо. Це забезпечує проактивне

виявлення та усунення ризиків ще до появи критичних вразливостей у бойовому середовищі.

Особливу цінність має безперервна верифікація безпеки в межах кожної ітерації, що дозволяє швидко адаптуватися до нових векторів атак, змін у середовищі або нормативних вимог. Такий підхід вимагає не лише технічних інструментів, а й належного рівня зрілості процесів, кваліфікованої команди та стратегічного планування інтеграції безпеки у архітектуру, код та тестування.

Таким чином, впровадження SDL-підходу забезпечує системну інтеграцію вимог інформаційної безпеки на кожному етапі життєвого циклу вебдодатку – від проєктування до експлуатації. Водночас, навіть за умови правильної реалізації SDL, надзвичайно важливим залишається етап перевірки та тестування безпеки системи, оскільки саме в процесі експлуатації можуть проявитися невраховані загрози та приховані вразливості.

1.5 Відомі методи та засоби тестування

Вразливості безпеки вебдодатків можна в деяких випадках легко виявити та ліквідувати, а в інших – важко виявити навіть сам факт вразливості системи безпеки. Можливі збитки від використання вразливостей безпеки у вебдодатку також можуть змінюватись від незначних фінансових втрат до повного банкрутства компанії. Щоб визначити ризики вразливості безпеки вебдодатку певної організації, слід оцінити ймовірності, пов’язані з джерелами загроз, векторами атак і вразливістю безпеки, а потім об’єднати їх з оцінкою технічної та репутаційної шкоди даній організації [36-42].

На даний час проводиться тестування методом білого ящика (з використанням внутрішніх даних про системи, включаючи аналіз вихідних кодів) з результатами тестування методами чорного і сірого ящика (коли аналіз проводиться з привілеями, ідентичними привілеїв потенційного зловмисника).

Тестування методом «білого ящика» має суттєву перевагу – це покриття коду. Оскільки вихідні коди доступні, то можуть бути проаналізовані на предмет

наявності потенційних вразливостей. До недоліків даного методу можна віднести складність, адже існуючі інструменти неідеальні та видають велику кількість помилкових спрацьовувань. Тому звіт, сформований в результаті роботи програми, повинен бути ретельно вивчений компетентним фахівцем. Враховуючи обсяг коду, який містять сучасні програми, такі звіти можуть досягати величезної довжини [43-49].

При ручному тестуванні за принципом «чорного ящика» дослідник за допомогою звичайного інтернет-браузера переміщається по сторінках додатків, підставляючи в поля введення і параметри запитів спеціальні символи, наприклад, символ одинарної лапки для виявлення потенційно вразливих до SQL-ін'єкцій сценаріїв.

В основі автоматизованого тестування (фаззінгу) лежить метод грубої сили, і основний недолік при такому підході компенсується його простотою і ефективністю. По суті фаззінг – це передача для обробки досліджуваним додатком великої кількості випадкових вхідних даних і аналіз результатів його роботи. В даний час існують фаззери, які генерують неповністю випадкові вхідні дані, а спираються на специфікації досліджуваних протоколів і форматів файлів. Такі інструменти можна також віднести до категорії методів «сірого ящика».

Тестування за принципом «сірого ящика» являє собою комбінацію методів, що використовуються при тестуванні за принципом «чорного ящика», а також технологій і прийомів реверс розробки. Цінність вихідного коду в процесі пошуку вразливостей полягає в тому, що він представляє логіку роботи програми в зрозумілому для дослідника поданні [50-57]. Основна мета аналізу – визначення внутрішньої логіки роботи захищеного додатку.

Автоматизоване виявлення вразливостей у вебдодатках здійснюється за допомогою спеціалізованих сканерів безпеки, серед яких найбільш поширеними є як відкриті, так і комерційні рішення. До відкритих засобів належать OWASP ZAP, Skipfish, Arachni, IronWASP, Vega та W3af. Ці інструменти орієнтовані на пошук типових вразливостей у динамічних вебдодатках, забезпечуючи базову автоматизацію процесу тестування безпеки [58, 59].

Серед комерційних рішень вирізняються Nessus та Acunetix. Nessus зосереджений переважно на мережевих перевірках, проте підтримує і тестування вебсервісів, маючи широку базу плагінів [60]. Acunetix спеціалізується на глибокому аналізі вебдодатків та формуванні детальних звітів із рекомендаціями щодо усунення виявлених вразливостей [61].

Окремо варто згадати Nikto – консольний інструмент для швидкої перевірки вебсерверів на наявність поширених конфігураційних помилок і застарілого ПЗ [62]. Усі зазначені сканери активно використовуються в практиці оцінювання безпеки вебдодатків і розглядаються в наукових дослідженнях, присвячених порівнянню ефективності засобів автоматизованого тестування.

Усі згадані сканери характеризуються високим рівнем автоматизації процесу тестування: здійснюється аналіз вебсторінок і параметрів запитів, генерується трафік і обробляються відповіді сервера з метою виявлення потенційних вразливостей. Застосування таких інструментів суттєво підвищує ефективність виявлення типових загроз безпеки у вебдодатках.

Аналіз наявних методів автоматизованого тестування свідчить[63] про їхню результативність переважно на етапі експлуатації програмного забезпечення. Сучасні сканери, зокрема OWASP ZAP, Nessus та Acunetix, забезпечують виявлення широкого спектра технічних вразливостей – ін’єкцій, помилок конфігурації, недоліків автентифікації тощо. Водночас більшість таких засобів орієнтована на використання вже після завершення етапу розробки та впровадження програмного продукту.

Подібна концентрація на завершальному етапі створює ризик виявлення вразливостей уже після того, як вони стали частиною робочого середовища, що, у свою чергу, може призвести до значних витрат на усунення недоліків та потенційних порушень безпеки. Крім того, багато типів вразливостей, особливо пов’язаних із помилками проєктування або порушенням принципів безпечного програмування, залишаються поза межами виявлення засобами автоматизованого сканування.

Таким чином, ефективне забезпечення безпеки вебдодатків вимагає не лише застосування сучасних засобів тестування, а й впровадження підходу до оцінювання критеріїв безпеки на всіх етапах життєвого циклу вебдодатків – від етапу вимог і проєктування до етапів реалізації, тестування, розгортання та супроводу.

1.6 Висновок до розділу 1

1. Проведений аналіз сучасних вебдодатків як складних багатокомпонентних систем в контексті життєвого циклу розробки програмного забезпечення (SDLC) дав змогу встановити їхню специфіку та обґрунтувати необхідність застосування систематизованого підходу до забезпечення безпеки на всіх етапах розробки.

2. Здійснено компаративний аналіз традиційних (каскадна, V-модель) та ітеративних (спіральна, інкрементальна, швидка розробка) моделей SDLC, що дало змогу виявити принципові обмеження традиційних підходів щодо інтеграції заходів безпеки на ранніх етапах та встановити переваги ітеративних моделей для безперервної інтеграції безпекових практик.

3. Встановлено пряму кореляційну залежність між зростанням архітектурної складності вебдодатків та збільшенням кількості потенційних векторів атак і системних вразливостей на основі аналізу статистичних даних Sucuri (2023) та класифікації OWASP Top 10 (2021), що дало змогу ідентифікувати основні категорії загроз інформаційній безпеці.

4. Досліджено концепцію Secure Development Lifecycle (SDL) та проаналізовано існуючі методи й засоби тестування безпеки (тестування "білого", "чорного", "сірого" ящиків, автоматизовані сканери), що дало змогу виявити їхні обмеження у виявленні вразливостей проєктування та бізнес-логіки, а також відсутність можливості комплексної кількісної оцінки рівня захищеності системи.

Отже, актуальною задачею є розроблення єдиної узгодженої системи комплексної якісної та кількісної оцінки рівня захищеності вебдодатків з

урахуванням їхніх архітектурних особливостей та специфіки життєвого циклу розробки для подолання проблематики недостатнього рівня формалізації вимог безпеки та зменшення суб'єктивності в оціночних процедурах.

РОЗДІЛ 2. ТЕОРЕТИКО-МЕТОДИЧНЕ ОБҐРУНТУВАННЯ ТА АПАРАТ КІЛЬКІСНОГО ОЦІНЮВАННЯ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ ВЕБДОДАТКІВ

2.1 Проблематика формалізації вимог безпеки та ризики суб’єктивності в оціночних процедурах

Сучасні підходи до оцінювання безпеки вебдодатків характеризуються значними обмеженнями у забезпеченні комплексного та об'єктивного аналізу. Навіть комплексні автоматизовані інструменти (SAST, DAST, IAST, SCA) охоплюють лише певні аспекти безпеки – кожен із них оптимізований під конкретні класи вразливостей. Наприклад, динамічні і статичні сканери виявляють вразливості на основі відомих шаблонів, але мають обмеження у тому, що можуть виявити. Вони часто генерують помилкові спрацьовування (false positives), пропускають нові або складні вразливості (false negatives) і не аналізують бізнес-логіку додатку чи міжмодульні інформаційні потоки. Також, чорні, сірі та білі ящики тестування забезпечують різний доступ до інформації: у білому ящику тестер має повну інформацію про систему, у сірому – частково, а в чорному – жодної. У жодному з цих підходів не формується інтегральний показник безпеки системи; результати є фрагментарними й залежать від завдань, які ставлять аудитори. Попри наявність сучасних стандартів і засобів, об’єктивна та кількісна оцінка загального рівня захищеності залишається недосяжною. Як зазначають дослідники, «вимірювати безпеку точно важко, і багато показників безпеки мають проблеми» [25]. З огляду на це, в сучасних підходах найчастіше використовується кількісне ранжування (наприклад, рівні 1=низький, 2=середній, 3=високий), яке по суті є якісною інтерпретацією експертних оцінок [64]. Але через високу долю інтуїтивних суджень і досвіду аудиторів такі оцінки не стабільні та часто не узагальнюються в єдине числове значення.

Формалізувати вимоги безпеки – означає описати їх так точно, щоб їх можна було автоматично перевіряти і порівнювати. На практиці це складніше через ряд

причин. По-перше, багато вимог мають розпливчастий або якісний характер. Наприклад, положення про «адекватний захист даних» чи «ефективний контроль доступу» часто не мають чітких числових порогів. По-друге, складність перевірки таких вимог велика: належність системи до безпечного стану може виявлятися лише через складний аналіз інформаційних потоків, механізмів шифрування, інтеграцій із зовнішніми сервісами тощо. Проста наявність SSL-сертифіката чи модуля автентифікації не гарантує фактичну стійкість до атак, але формально інструменти це часто трактують як виконання вимоги. По-третє, відсутність уніфікованих критеріїв ускладнює порівняння результатів. Наприклад, стандартні вимоги не надають єдиної математичної формули для обчислення «балу захищеності»; замість цього вони задають набір правил і процедур. У підсумку експерти часто інтерпретують одні й ті ж вимоги по-різному в залежності від контексту.

З огляду на це, роль формалізації зводиться переважно до створення структурованих чек-листів і правил перевірки, але математична модель самих вимог поки не реалізована у жодному широко відомому стандарті. Аналіз Security NIST показує, що зусилля зі створення якісно визначених метрик часто не досягають повної коректності: формальні специфікації важко верифікувати і підтвердити їхню відповідність реальним сценаріям [65].

Часто, результати аудиту безпеки суттєво залежать від людського чинника. Досвід і кваліфікація аудитора визначають, наскільки глибоко і всебічно буде проведено оцінку. Кваліфіковані фахівці здатні виявити нетривіальні проблеми та сформулювати ризики детальніше, а початківці можуть пропустити деякі аспекти. Крім того, доступність документації та інформації про систему суттєво впливає на об’єктивність результатів. Якщо архітектурні специфікації, документація з безпеки або журнали аудиту неповні чи застарілі, аудиторові доводиться спиратися на припущення. Неповна інформація про бізнес-логіку чи внутрішню взаємодію компонентів призводить до великих прогалин у оцінці. Як відзначають фахівці, відсутність або неточність документації про процеси безпеки «може призвести до проблем під час наступного аудиту» [66]. У таких умовах до висновків

закрадаються додаткові невизначеності і припущення, що збільшує вплив людського фактору.

Ще один чинник – функціональність і складність системи. Чим більше модулів і взаємодій містить додаток, тим складніше охопити усі можливі канали атак. Складні вебсервіси з великою кількістю API, мікросервісів або динамічних компонентів важче оцінити у повному обсязі. Аудитори, як правило, обмежені в часі й ресурсах, тому можуть фокусуватися на очевидних елементах, а більш складні/специфічні функціональні блоки залишати з меншим рівнем уваги.

Враховуючи комплексність процесу, зручно сказати, що на кожному етапі оцінювання неможливо усунути повністю людський чинник і упередженість. Як пишуть фахівці з ризик-менеджменту, «на всіх стадіях процесу оцінки ризику – від вибору методу до визначення ймовірностей та наслідків – присутня суб’єктивність» [67]. Тобто навіть формальні методи захисту потребують інтерпретації людиною, що робить їх результат невідтворюваним між різними оцінювачами. NIST підкреслює, що цінним є такий метод вимірювання, який давав би однаковий результат незалежно від того, хто його проводить, – але на практиці цього досягти важко.

Зазвичай, сучасні підходи до оцінювання безпеки в основному дають структуровані або якісні результати. Це можуть бути звіти з переліком знайдених вразливостей, ступенем відповідності контролям або словесні оцінки рівня ризику. Однак власникам бізнесу та менеджерам продукту було б добре мати універсальний числовий індикатор, який можна було б використовувати для порівняння різних систем або відстеження динаміки захищеності з часом. Нестача конкретних числових метрик відчутна на практиці. Як показує дослідження NIST, метрики безпеки зазвичай засновані на експертних оціночних судженнях, а не на об’єктивних вимірах [64].

На практиці менеджерам доводиться зводити ці якісні дані до цифрових індикаторів на свій розсуд, наприклад, обчислювати сумарні оцінки, середні бали, виконувати зважування. Але такі власноручні формули рідко універсальні і зрозумілі. Немає загальновизнаного підходу до стандартизованого «бального»

виміру захищеності. Водночас в науковій спільноті підкреслюється, що саме кількісні метрики дають змогу приймати оптимальні управлінські рішення. Як зауважує В. Сендерс із Illinois University, кількісні метрики безпеки дозволяють «порівняти альтернативи та вибрати найкращий варіант» при проєктуванні чи експлуатації систем [68]. Без таких метричних оцінок у керівників відсутній інструмент для зрозумілого відстеження тенденцій або вимірювання прогресу.

Узагальнюючи викладене, можна стверджувати, що ключовою проблемою у сфері оцінювання безпеки вебдодатків є недостатній рівень формалізації вимог та значна частка суб'єктивності в інтерпретації результатів. Це ускладнює уніфіковане застосування існуючих методів на практиці, особливо в контексті забезпечення відтворюваності, порівнюваності та обґрунтованості висновків безпекових аудитів. У цьому контексті постає потреба у зверненні до вже існуючих авторитетних підходів, здатних задати структуровану рамку для оцінювання.

Серед наявних стандартів у сфері безпеки вебдодатків OWASP Application Security Verification Standard (ASVS) вирізняється своєю глибиною, системністю та практичною орієнтованістю. Його структура охоплює широкий спектр вимог безпеки – від контролю доступу та автентифікації до криптографії, управління конфігураціями та журналювання подій – що дозволяє здійснювати всебічну перевірку захищеності програмного забезпечення. ASVS також підтримує градацію рівнів верифікації, що робить його придатним як для типових, так і для критично важливих вебдодатків. Завдяки формалізованому підходу до опису вимог та верифікаційних критеріїв, стандарт сприяє зниженню суб'єктивності оцінювання та підвищенню повторюваності перевірок. Крім того, ASVS активно інтегрується у сучасні практики розробки, зокрема DevSecOps, і базується на перевірених підходах OWASP, що забезпечує його актуальність в умовах постійної еволюції загроз. Сукупність цих характеристик дають змогу глибше осмислити потенціал формалізованого підходу до оцінювання, що особливо важливо в умовах зростаючої складності сучасних вебдодатків і вимог до їхнього захисту.

2.2.Структурний аналіз OWASP ASVS: рівні верифікації, критерії відповідності, практичні сценарії застосування

У сфері забезпечення інформаційної безпеки вебдодатків особливу роль відіграють стандарти, які формалізують вимоги до архітектури, реалізації та тестування захисних механізмів. Одним із найвідоміших та широко використовуваним веброзробниками стандартом у сфері вебтехнологій є OWASP Application Security Verification Standard (ASVS) [69]. Стандарт OWASP ASVS є відкритим галузевим стандартом перевірки безпеки вебдодатків. Він покликаний уніфікувати охоплення та глибину аналізу безпеки, надаючи основу для проєктування, розробки та тестування технічних засобів захисту додатків. ASVS описує конкретні вимоги до безпеки, згруповані за функціональними розділами (архітектура, автентифікація, управління сесіями, тощо). Кожна вимога має унікальний ідентифікатор у форматі “розділ.секція.номер”, та прив’язана до одного або кількох рівнів верифікації. На практиці ASVS широко використовується як базис для оцінки безпеки вебдодатків: версія 4.0.3 містить 286 контрольних вимог, затверджених спільнотою фахівців з безпеки. За рахунок відкритого підходу та довготривалої підтримки стандарт полегшує формулювання вимог у контрактах і сприяє підвищенню довіри до додатків.

Однією з ключових переваг ASVS є його комплексний підхід, що охоплює широкий спектр функціональних і нефункціональних аспектів безпеки, від проєктування до впровадження та подальшого супроводу вебдодатків. Стандарт забезпечує чітку структуру для впровадження контролів на різних етапах розробки, надаючи можливість адаптації вимог безпеки до конкретних умов використання.

Важливим аспектом практичного застосування OWASP ASVS є його взаємозв’язок із базою знань CWE (Common Weakness Enumeration) – загальновизнаною системою класифікації програмних вразливостей, що розробляється та підтримується організацією MITRE. Хоча ASVS не є прямою реалізацією CWE, між цими двома ресурсами простежується методична сумісність: багато вимог ASVS мають відповідності з типовими слабкостями, визначеними в

CWE. Це дає змогу розглядати ASVS не лише як перелік контрольних точок перевірки, а й як інструмент, що забезпечує системну основу для виявлення, класифікації та документування вразливостей у контексті загальноприйнятої таксономії.

У специфікаціях ASVS нерідко наводяться посилання на відповідні ідентифікатори CWE, що значно спрощує процес зіставлення вимог безпеки з потенційними дефектами реалізації. Такий підхід сприяє більш ефективному використанню ASVS у середовищах з підвищеними вимогами до автоматизації аналізу безпеки, таких як DevSecOps-практики або аудит із застосуванням статичних аналізаторів коду. Отже, інтеграція з CWE підсилює аналітичний потенціал ASVS і робить його особливо корисним у задачах системного моніторингу та управління ризиками безпеки.

ASVS виділяє три рівні перевірки безпеки, кожен з яких призначений для вебдодатків із різним ступенем критичності та ризиків. Перший рівень орієнтований на вебдодатки з низьким рівнем ризику і може бути повністю протестований за допомогою тестування на проникнення без необхідності доступу до вихідного коду. Другий рівень призначений для вебдодатків, що обробляють конфіденційну інформацію, і вимагає більш глибокої перевірки, зокрема, доступу до документації та вихідного коду. Цей рівень рекомендовано для більшості сучасних вебдодатків, оскільки він надає збалансований підхід до забезпечення безпеки. Третій рівень призначений для критично важливих вебдодатків, таких як фінансові або медичні системи, які виконують транзакції з високим ступенем ризику і вимагають найвищого рівня захищеності.

Кожен рівень ASVS включає перелік вимог щодо захищеності, які співвідносяться з конкретними функціями безпеки, що повинні бути інтегровані в архітектуру та процеси розробки вебдодатків. Ці вимоги охоплюють всі аспекти безпеки, включно з автентифікацією, управлінням сесіями, контролем доступу, обробкою помилок, захистом даних та іншими ключовими елементами, що впливають на надійність вебдодатків. Завдяки своїй універсальності та широкій сфері застосування, ASVS є важливим інструментом для забезпечення

відповідності вебдодатків сучасним вимогам безпеки. На рисунку 2.1 відображені всі рівні стандарту перевірки безпеки застосунків OWASP.

Рис. 2.1. Рівні стандарту перевірки безпеки додатків OWASP 4.0

Рівні верифікації ASVS служать критерієм масштабування перевірок: наприклад, усі вимоги базового рівня (L1) мають бути автоматизовані та не потребують доступу до коду, тоді як вимоги L2 і L3 вимагають залучення додаткових засобів. Таким чином, вибір рівня визначається контекстом застосування: стандарт рекомендує починати з L1 для загальної перевірки всього портфелю додатків, використовувати L2 для більшості сучасних вебсервісів, а L3 – для найкритичніших проєктів з високими вимогами до безпеки.

Стандарт ASVS систематизує заходи безпеки у функціональні та нефункціональні категорії, охоплені різними розділами. Серед функціональних аспектів виділяють: автентифікацію користувачів (перевірка облікових даних, політики паролів, MFA тощо), управління сесіями (генерація, зберігання та відновлення сесійних токенів), контроль доступу (рольова автентифікація, перевірка прав доступу), перевірку і санітизацію вводу (валідація параметрів, захист від ін’єкцій) та захист бізнес-логіки (перевірка критичних бізнес-процесів). З іншого боку, нефункціональні аспекти включають криптографічний захист даних (шифрування конфіденційних даних у спокої та під час передачі), безпечну обробку помилок та журналювання, щоб не розкривати внутрішню інформацію, захист від шкідливого коду (наприклад, контроль вмісту й перевірка контейнерів), а також безпечну конфігурацію й налаштування середовища (налаштування TLS, безпечні значення за замовчуванням). Кожен з цих аспектів деталізовано у відповідних

розділах ASVS, таких як V2 – Автентифікація, V7 – Обробка помилок, V6 – Криптографія тощо, причому кожна окрема вимога прикріплена до одного чи кількох рівнів верифікації. Наприклад, розділ про автентифікацію пропонує вимоги до довжини пароля, процедури скидання пароля та рекуперації облікового запису, а розділ про криптографію встановлює стандарти використання сучасних алгоритмів і управління ключами.

OWASP ASVS інтегрується в різні фази розробки програмного забезпечення як орієнтир безпеки. У фазі проєктування ASVS використовується для планування заходів безпеки: з самого початку проводять аналіз загроз і визначають, які секції ASVS релевантні конкретному застосунку. Наприклад, уже на рівні архітектури закладають механізми автентифікації і контролю доступу (планування політик паролів, управління ролями тощо). Зазвичай, це дозволяє врахувати вимоги безпеки «з самого початку» (Security by Design) і не додавати їх вторинно. Під час розробки розробники звертаються до ASVS як до чек-листа. У код вводяться ключові заходи безпеки: сильна валідація вводу, правильна обробка аутентифікації, безпечне зберігання облікових даних, валідація усіх зовнішніх параметрів. Наприклад, на етапі кодування слід реалізувати політики складних паролів, захист від SQL-ін’єкцій та XSS через санітизацію даних. Дотримання рекомендацій ASVS на рівні розробки зменшує необхідність виправляти критичні вразливості в пізніших стадіях життєвого циклу. На етапі тестування та верифікації ASVS служить базовим чек-листом для оцінки виконання вимог. Перевіряються як загальні вимоги рівня L1 (застосовні до всіх додатків), так і ті, що відповідають обраному рівню L2 чи L3. Зазвичай це включає як автоматизовані тести, так і ручне тестування: сканування вразливостей, код-рев’ю, тестування бізнес-логіки та перевірку конфігурації системи. Для додатків високого рівня безпеки (L3) можуть виконуватись додаткові пентести і ретельне тестування «open-box» з доступом до вихідних кодів. Успішне проходження ASVS перевірок означає, що більшість відомих вразливостей були враховані, що дозволяє виявити й усунути дефекти на цьому етапі.

ASVS здобув популярність у галузях з підвищеними вимогами до безпеки. Наприклад, у фінансовому та медичному секторах стандарт рекомендує застосовувати рівень 3, оскільки ці системи оперують найконфіденційнішими даними. Додатки, що обробляють банківські операції або медичні записи, зазвичай проходять всі перевірки ASVS рівня 3: ретельне тестування архітектури і коду, моделювання загроз та інтеграція вимог регуляторів (наприклад, стандарти MAS чи PCI DSS часто узгоджуються з ASVS). Відомо, що компанії, які впроваджують ASVS у свої процеси, реєструють менше інцидентів безпеки та підвищують довіру клієнтів. Зокрема, численні кейси показують, що стандартизація перевірок за ASVS дозволяє знизити кількість вразливостей при релізах і покращити якість аудиторських звітів.

Використання ASVS має низку важливих переваг. Відкритість і підтримка спільнотою: ASVS розроблено OWASP – некомерційною організацією, підтримуваною міжнародною спільнотою експертів. Це гарантує постійне оновлення стандарту, а також доступність стандарту без ліцензійних обмежень. Широке прийняття: багато організацій різного масштабу використовують ASVS як основу для перевірки безпеки вебдодатків, що полегшує інтеграцію з існуючими політиками безпеки та робочими процесами. Актуальність: ASVS регулярно оновлюється з урахуванням нових загроз і архітектур. Наприклад, у версії 4.0 був доданий окремий розділ перевірки бізнес-логіки та посилено розділ про захист вебсервісів, що робить стандарт більш повним порівняно з ASVS 3.0. Сумісність з іншими стандартами: ASVS узгоджується з існуючими фреймворками (наприклад, PCI DSS, NIST, OWASP Top 10), забезпечуючи чіткі рекомендації та уніфікуючи вимоги. Зручність і автоматизація: формалізовані вимоги ASVS (з чіткими рівнями пріоритету) полегшують побудову автоматизованих тестів і чек-листів, оскільки зрозуміло, які саме заходи слід перевіряти на різних стадіях розробки. За рахунок цих переваг ASVS став цінним інструментом у побудові культури безпеки програмного забезпечення та забезпечує стандартизований підхід до організації захищеності вебдодатків. Вище описані підходи не вирішують проблеми оцінювання загального рівня захищеності вебдодатків. Аналіз вказує на чітку

потребу нового методу, що поєднав би експертні знання та формалізм для отримання кількісних метрик захищеності. Такий метод мав би будуватися на прозорих метриках і алгоритмічних моделях, які забезпечують відтворюваність результатів незалежно від оцінювача. Водночас ці метрики повинні бути інформативні та зрозумілі для власників і менеджерів IT-продуктів. Таким чином, сучасні стандарти й інструменти потрібно доповнювати або переосмислювати через призму кількісних оцінок. Це відкриє можливість перетворити результати аудиту безпеки з експертних звітів на зручні числові індикатори, за якими зможуть чітко орієнтуватися керівники і власники IT-продуктів. Враховуючи обмеження існуючих підходів (частка інтуїції, неповнота даних, невідтворюваність), є виправданим розробляти нові методи аналізу і обчислення метрик захищеності. Тому розробка методу, який би дозволив кількісно оцінювати рівень захищеності вебдодатків є досить актуальним, оскільки це дозволить забезпечити прозорість і об'єктивність процесу оцінки рівня безпеки вебдодатка. Таке рішення дозволить не лише отримати рекомендаційну інформацію про те, що мало б бути імплементоване розробниками в продукт для захисту від вразливостей - а й оцінювати рівень захищеності в числовому вигляді, що зробить процеси аналізу безпеки більш об’єктивними та доступними для розуміння звичайних користувачів та власників вебдодатків. Це в свою чергу дозволить організаціям оцінити поточний стан захищеності своїх вебдодатків та прийняти подальші міри для вдосконалення власного продукту.

Таким чином, з урахуванням виявлених проблем суб’єктивності в оцінювальних процедурах і складності інтерпретації вимог стандарту OWASP ASVS, постає необхідність формалізації критеріїв, що забезпечать більш об’єктивну та уніфіковану оцінку безпеки вебдодатків. Наступним кроком є розроблення системи таких критеріїв на основі структурного аналізу змісту та ієрархії стандарту ASVS.

2.3 Розробка системи критеріїв для оцінки вимог безпеки вебдодатків на основі стандарту безпеки ASVS

У межах даного дослідження було сформульовано та реалізовано власний метод кількісної оцінки безпеки вебдодатків на основі стандарту OWASP ASVS. На відміну від спроб охоплення всього спектра вимог стандарту, що часто є надмірним для застосування у типовому розробницькому процесі, було обґрунтовано доцільність селективного підходу, за якого до остаточної моделі включено лише 133 вимоги з понад 260. Такий відбір здійснювався на основі системного аналізу кожного з розділів ASVS, в результаті якого було визначено, які саме вимоги є достатніми для забезпечення базової та розширеної безпеки в умовах типової архітектури вебдодатків.

Результатом цієї процедури стало те, що вимоги, які мають надвисоку складність верифікації (наприклад, потребують глибинного аналізу логіки сторонніх компонентів, middleware-рівня або хмарної інфраструктури), були виключені зі складу базової оцінки. Натомість перевага надавалась вимогам, які, по-перше, є технічно об'єктивованими (тобто такими, що піддаються перевірці без додаткового суб'єктивного трактування), і, по-друге, забезпечують покриття основних векторів атаки, які на сьогодні є найпоширенішими згідно зі статистикою OWASP Top 10.

Окреме місце у формуванні вибірки для масштабованості та повторюваності: обрані вимоги повинні бути придатними для багаторазового застосування в процесах розробки, тестування та впровадження без необхідності індивідуального доопрацювання під кожен окремий проєкт. Завдяки цьому сформована вибірка не лише дозволяє здійснити об’єктивну оцінку рівня захищеності вебдодатку, а й сприяє впровадженню процедурного мислення щодо безпеки, де реалізація контролів не є випадковою, а вибудовується за структурованим набором критеріїв. Зрештою, для кожного тематичного розділу (автентифікація, керування сесіями, логування тощо) збережено всі базові вимоги, а також ті розширені положення, які

є критичними з огляду на принципи розділення повноважень, конфіденційності, цілісності, наглядовості та стійкості до типових атак.

Таким чином, скорочення загального переліку вимог є формалізованим шляхом до підвищення ефективності впровадження стандарту в умовах обмежених ресурсів. Це також відповідає сучасним підходам до гнучкої безпеки (agile security), де замість надмірного бюрократичного навантаження перевага надається точковим, але стратегічно виправданим перевіркам, які дають найбільший ефект для підвищення реального рівня захищеності вебдодатку.

У межах розробки методу оцінювання безпеки вебдодатків на основі OWASP ASVS було виконано повномасштабну процедуру формалізації кожної з відібраних вимог. Ця робота передбачала не лише їх опис, а й трансформацію формулювань з тексту стандарту у структуровану систему перевірних характеристик, що дозволяє здійснювати кількісне оцінювання рівня їх виконання у конкретному вебдодатку. Такий підхід забезпечив максимальну чіткість, відтворюваність та об’єктивність процесу верифікації.

Для кожної вимоги було розроблено набір чітко визначених критеріїв оцінювання, що дозволяють розкласти абстрактне формулювання вимоги на операційно доступні, логічно завершені та технічно здійсненні одиниці аналізу. Такий підхід дозволяє уникнути суб’єктивізму в процесі оцінювання та забезпечує високий ступінь деталізації безпеки на рівні практичної реалізації функціоналу. Усі критерії були приведені до уніфікованої шкали оцінювання, побудованої за принципом триточкової градації з кроком 0.5. Це дозволяє гнучко враховувати не лише бінарну наявність або відсутність функціоналу, але й часткову реалізацію або обмежене покриття вимоги. Таким чином, кожна вимога набуває не просто статусу "виконано/не виконано", а отримує кількісне значення, що відображає реальний рівень її дотримання у межах конкретного проєкту. Щоб зрозуміти процедуру оцінювання вимог розглянемо, для прикладу одну із вимог.

Вимога: Verify that password change functionality requires the user's current and new password.

Зазначена вимога стосується забезпечення безпечного та контрольованого процесу зміни пароля, що, згідно з практиками безпечного управління автентифікацією, має мінімізувати ризики несанкціонованого доступу до облікових записів, зокрема в разі компрометації сесій або вразливостей в інтерфейсах користувача. Саме тому вона була визнана однією з пріоритетних у структурі розділу, присвяченого автентифікації.

На першому етапі опрацювання було сформульовано 9 окремих критеріїв перевірки, кожен із яких дозволяє оцінити специфічний аспект функціонування механізму зміни пароля. Як вже згадувалось кожен критерій було забезпечено трирівневою шкалою оцінювання:

- 0 - вимога не виконується;
- 0.5 - виконується частково або з обмеженнями;
- 1- повна відповідність вимогі.

Наприклад, якщо система дозволяє змінити пароль без будь-якої повторної перевірки особи – оцінка дорівнює 0. Якщо перевірка здійснюється, але може бути обійдена (наприклад, при вже активній сесії), – це оцінюється у 0.5. І, навпаки, якщо поточний пароль обов’язковий і без нього зміна неможлива – система отримує 1 бал. Для кращого розуміння, слід розглянути список критеріїв для однієї вимоги.

Критерій №1: «Чи вимагається введення поточного пароля при зміні пароля?»

Одним із ключових аспектів безпечного управління даними автентифікації користувача є наявність механізму повторної перевірки особи під час виконання чутливих дій, таких як зміна пароля. У межах запропонованої моделі оцінювання цей механізм формалізовано у вигляді даного критерію, що виступає індикатором належної реалізації політики повторної автентифікації.

Зміна пароля – це одна з найкритичніших транзакцій в життєвому циклі автентифікатора, оскільки вона відкриває можливість повного контролю над обліковим записом. Тому вимога повторного підтвердження особи є визнаним мінімумом захисної практики у сучасних підходах до безпеки. Хоча стандарт NIST SP 800-63B [70] у розділі 5.1.2 прямо не зобов’язує до використання поточного

пароля як конкретного засобу верифікації, він встановлює принципову вимогу щодо необхідності проходження автентифікації користувачем перед виконанням будь-яких «чутливих операцій». Це положення тлумачиться в рамках дослідження як методичне обґрунтування для застосування повторного введення пароля при його зміні, що, на практиці, є найбільш поширеною і технічно досяжною формою підтвердження ідентичності.

З функціонального погляду, наявність вимоги введення поточного пароля слугує бар'єром для зловмисників, які отримали тимчасовий або обмежений доступ до сесії користувача, оскільки без володіння дійсним автентифікатором вони не можуть змінити ключ до облікового запису. Відсутність цього механізму призводить до суттєвого зниження захищеності, оскільки навіть нетривалий або частковий компроміс може завершитися повною втратою контролю над обліковим записом.

Запропонований критерій оцінюється за трирівневою шкалою, яка дозволяє врахувати наступні значення:

- *0 балів* - повна відсутність механізму повторної автентифікації;
- *0.5 бала* - часткова реалізація, наприклад, вимога до введення поточного пароля залежить від контексту (авторизована сесія, trusted device тощо);
- *1 бал* - присвоюється лише тоді, коли введення поточного пароля або альтернативного автентифікатора (наприклад, біометрії чи OTP) є обов'язковою передумовою для зміни пароля в усіх випадках, незалежно від стану сесії користувача.

Таким чином, даний критерій виконує подвійну функцію: з одного боку, він верифікує відповідність системи принципам автентифікаційного контролю, встановленим стандартом NIST SP 800-63B; з іншого – є практичним індикатором захисту користувацьких облікових записів від типових сценаріїв внутрішньосесійного захоплення чи подальшого ескалування прав. Його врахування в межах загальної оцінки дозволяє більш точно і кількісно виміряти ступінь відповідності системи сучасним вимогам інформаційної безпеки.

Критерій №2: «Чи вимагається введення нового пароля при зміні пароля?»

Одним з основних елементів забезпечення цілісності процесу зміни пароля є обов’язковість введення нового автентифікаційного значення, яке відповідає встановленим політикам безпеки. Саме цю функцію відображає другий критерій оцінювання. Він розглядається як логічне завершення операції оновлення автентифікатора та є передумовою для підвищення стійкості системи до компрометації користувацьких облікових записів.

Нормативною основою для формування цього критерію виступає стандарт ISO/IEC 27002:2022 [71], зокрема його положення, викладені в розділі 9.4.3, що регламентує управління паролями користувачів. Хоча у згаданому стандарті прямо не зазначено, що новий пароль обов’язково має бути введений під час процедури зміни, однак документ містить чіткі рекомендації щодо впровадження контролю політики паролів, включаючи вимоги до мінімальної довжини, складності, унікальності та періодичності зміни. У даному контексті логічним наслідком дотримання таких вимог є необхідність заміни старого пароля новим, що відповідає актуальній політиці.

Формулювання цього критерію базується на розумінні того, що ефективна безпекова політика повинна не лише вимагати факт зміни пароля, а й забезпечувати його якісне оновлення. В іншому випадку допускається можливість повторного використання старого або слабкого пароля, що суперечить принципам захисту від атак на основі попередніх знань або словників, а також знижує ефективність таких механізмів, як аудит змін, історія паролів і автоматичне виявлення шаблонів зловживань.

Запропонована трирівнева шкала оцінювання дозволяє диференціювати ступінь відповідності системи цьому критерію наступним чином:

- *0 балів* - система дозволяє користувачу завершити зміну пароля без фактичного оновлення значення – наприклад, через залишення поля нового пароля порожнім або автоматичне повторне використання попереднього;
- *0.5 бала* - часткове виконання, яке фіксується у випадках, коли введення нового пароля є обов’язковим, але допускається використання того самого значення, що й раніше. Така практика хоча й забезпечує формальне оновлення

пароля, однак не сприяє підвищенню безпеки, адже залишає систему вразливою до атак із повторним використанням скомпрометованих або простих паролів;

- *1 бал* - лише в тих випадках, коли система блокує використання попереднього пароля та забезпечує відповідність нового значення критеріям безпеки (довжина, складність, відсутність в історії, перевірка на унікальність).

Таким чином, критерій «введення нового пароля» оцінює не лише наявність самої дії, а й якість її реалізації в контексті вимог до автентифікаторів. Він забезпечує вимірюваність такого аспекту захисту, який часто залишається поза увагою при поверхневому аналізі. У межах загального методу цей критерій відіграє важливу роль у формуванні індикаторів операційної придатності та гігієни облікових записів, що є критично важливим для запобігання компрометаціям через повторне використання слабких або зламаних автентифікаторів.

Критерій №3: «Чи є механізм перевірки правильності поточного пароля?»

Наявність механізму перевірки правильності поточного пароля під час його зміни є фундаментальною умовою забезпечення цілісності автентифікаційного процесу та захисту від несанкціонованих змін облікових даних. Саме цей аспект було формалізовано у вигляді окремого критерію в рамках запропонованого методу. Його завдання полягає у визначенні, наскільки система здатна підтвердити справжність дій користувача перед внесенням змін до критичних ідентифікаційних параметрів.

Визначальним нормативним джерелом для цього критерію виступає NIST SP 800-63B, зокрема розділ 5.1.1.2, у якому викладено вимоги до використання автентифікаторів, заснованих на запам’ятованих секретах (memorized secret verifiers). У вказаному стандарті прямо зазначається, що система перевірки (verifier) повинна вимагати від користувача проходження автентифікації шляхом подання відповідного секрету (наприклад, пароля) перед виконанням будь-якої чутливої операції.

У контексті розробленої моделі така перевірка трактується як обов’язковий елемент логіки безпечної взаємодії: сам факт, що користувач ввів пароль у попередній сесії, не повинен автоматично давати право на зміну цього пароля без

додаткового підтвердження. Це особливо актуально з огляду на загрози, пов’язані з викраденням сесій (session hijacking), фішингом або атаками з використанням залишених або активних сесій на публічних пристроях. У всіх подібних випадках відсутність механізму перевірки поточного пароля відкриває шлях до ескалації доступу без фактичного знання автентифікатора.

З технічної точки зору, перевірка правильності поточного пароля означає, що система повинна не просто приймати довільне значення у відповідному полі, а саме звіряти його з автентичними обліковими даними користувача, виконуючи повноцінну процедуру валідації. Якщо такої перевірки не здійснюється або вона є лише формальною – це істотно знижує довіру до безпечності механізму зміни автентифікаційних даних.

Для даного критерію передбачено трирівневу шкалу оцінювання, яка дозволяє кількісно відобразити ступінь реалізації контролю:

- *0 балів* присвоюється у випадках, коли система взагалі не перевіряє правильність поточного пароля – наприклад, якщо допускається зміна пароля без введення поточного, або якщо введене значення не верифікується;
- *0.5 бала* відповідає ситуаціям, у яких перевірка реалізована, але її можна обійти, наприклад, коли система довіряє активній сесії і дозволяє зміну без повторної валідації;
- *1 бал* отримують лише ті системи, які незмінно вимагають перевірки коректності введеного поточного пароля, незалежно від обставин, і в разі помилки – блокують виконання дії зі зміни автентифікатора.

Таким чином, цей критерій виконує важливу функцію виявлення наявності та коректності реалізації контрольного механізму, що забезпечує достовірність змін у чутливих даних користувача. Він не тільки забезпечує відповідність кращим практикам безпеки, задекларованим у NIST SP 800-63B, а й відіграє критичну роль у контексті захисту вебдодатків від внутрішньосесійних атак, маніпуляцій із боку зловмисників або компрометації користувацького профілю внаслідок стороннього втручання.

Критерій №4: «Чи є механізм перевірки нового пароля на відповідність вимогам безпеки?»

Оцінювання якості нового пароля під час його створення або зміни є одним із фундаментальних аспектів забезпечення надійності автентифікаційної системи. В межах розробленої моделі цей критерій покликаний визначити, наскільки ефективно система реалізує політики контролю за довжиною, складністю, унікальністю та іншими параметрами, що знижують ймовірність успішного зламу пароля методами перебору або соціальної інженерії.

Нормативним підґрунтям для обґрунтування цього критерію виступає стандарт ISO/IEC 27002:2022, зокрема його розділ 9.4.3, у якому прямо рекомендовано впроваджувати контроль за параметрами пароля. Це положення, хоча й не формулює обов'язковість перевірки нового пароля під час його зміни, але слугує достатнім підґрунтям для методичного включення відповідного механізму перевірки у систему оцінювання безпеки. У сучасних практиках безпеки перевірка нових автентифікаторів є критично важливою не лише з погляду відповідності внутрішній політиці, а й у межах попередження повторного використання вразливих, слабких або раніше скомпрометованих значень.

З технічної точки зору, ефективна реалізація цього критерію має включати:

- перевірку на мінімальну довжину пароля;
- аналіз складності (використання великих/малих літер, цифр, спеціальних символів);
- відсутність пароля в списках відомих або заборонених значень;
- унікальність щодо останніх паролів (історія змін);
- додатково – аналіз шаблонності або надто простих комбінацій.

Відсутність перевірки на відповідність політиці відкриває шлях до встановлення слабких паролів, що значно підвищує ризик компрометації облікового запису навіть при наявності інших захисних механізмів. Натомість системи, які інтегрують повноцінну перевірку відповідно до заданої політики, демонструють вищу стійкість до атак на автентифікатор і знижують ймовірність успішного зловживання з боку як зовнішніх, так і внутрішніх загроз.

Критерій оцінюється за трирівневою шкалою:

- *0 балів* – відсутність будь-якої перевірки: система приймає будь-яке значення, включаючи надто короткі, прості або повторно використані паролі;
- *0.5 бала* – часткова реалізація: перевіряється лише один або кілька параметрів, наприклад, довжина, але без перевірки складності, історії чи чорних списків;
- 1 бал – реалізовано повний механізм перевірки відповідності нового пароля політиці безпеки, що охоплює всі ключові аспекти, рекомендовані ISO/IEC 27002:2022, з блокуванням прийому небезпечних значень.

Таким чином, даний критерій дозволяє визначити, наскільки система дотримується принципу «security by design» у сфері управління автентифікаційними засобами. Його наявність у моделі є необхідною умовою для формування цілісної картини ефективності засобів захисту та підтримки гігієни паролів у середовищі з підвищеними вимогами до інформаційної безпеки.

Критерій №5: «Чи є механізм підтвердження зміни пароля через електронну пошту або інший канал зв'язку?»

Ефективне забезпечення безпеки користувацьких облікових записів передбачає не лише контроль за коректністю виконання чутливих операцій, а й наявність засобів зворотного інформування користувача про ці події. Однією з найважливіших операцій у цьому контексті є зміна пароля, яка безпосередньо впливає на автентифікаційний ланцюг доступу. Критерій покликаний оцінити, чи система реалізує належний рівень інформування про події безпеки, що дає змогу користувачеві оперативно реагувати на потенційно несанкціоновані дії.

Хоча стандарт ISO/IEC 27002:2022 у розділі 16.1.4 не містить прямої вимоги щодо обов'язкового надсилання повідомлень саме про зміну пароля, він містить загальну норму, згідно з якою безпекові події повинні бути виявлені та задокументовані з метою своєчасного реагування. Це положення дає нормативне підґрунтя для реалізації механізмів сповіщення, які є невід'ємною частиною ефективної політики реагування на інциденти та користувацької обізнаності.

У межах даного критерію аналізується, чи після зміни пароля система ініціює процедуру інформування користувача через канал зв'язку, прив'язаний до облікового запису (наприклад, електронну пошту, SMS, push-нотифікацію тощо). Відсутність такого механізму позбавляє користувача можливості оперативно виявити зловживання, пов'язані з несанкціонованим доступом до облікового запису, і тому розцінюється як істотний дефіцит безпеки. Навпаки, багатоканальна система сповіщень значно підвищує стійкість до атак, зокрема в ситуаціях, коли компрометація пароля відбулася без зламу поштової чи мобільної інфраструктури.

У моделі оцінювання критерій формалізовано за трирівневою шкалою:

- *0 балів* – система не надсилає жодних повідомлень про зміну пароля, що позбавляє користувача критично важливої інформації про стан облікового запису;
- *0.5 бала* – система надсилає повідомлення, але обмежується лише одним каналом зв'язку, зазвичай електронною поштою. Така реалізація частково компенсує ризики, але не гарантує своєчасного реагування, особливо у випадку компрометації цього каналу;
- *1 бал* – реалізована багатоканальна система оповіщення, яка забезпечує відправлення повідомлень щонайменше двома незалежними шляхами (наприклад, email та SMS). Це дозволяє значно знизити ймовірність непомічених інцидентів.

З практичної точки зору, цей критерій має високу прикладну цінність, оскільки легко перевіряється при тестуванні та не потребує доступу до коду чи внутрішньої інфраструктури. Він також безпосередньо впливає на індикатори time-to-detect і time-to-respond, які є критичними у моделі реагування на інциденти безпеки.

Таким чином, цей критерій дозволяє оцінити не лише наявність технічної реалізації сповіщення, а й її надійність, охоплення та пристосованість до сучасних сценаріїв загроз. Його інтеграція в метод дозволяє підвищити рівень контролю над ланцюгом автентифікації та реалізувати принципи проактивного захисту на етапі постфактум-комунікації.

Критерій №6: «Чи використовується багатофакторна автентифікація (MFA) під час зміни пароля?»

Враховуючи потенційну чутливість процесу зміни пароля, особливо у випадках, коли автентифікатор вже скомпрометовано або є ознаки ризикованої активності, використання багатофакторної автентифікації (MFA) розглядається як один з найефективніших засобів запобігання несанкціонованим змінам. Саме цей аспект втілюється в даному критерії, який входить до структури оцінювання безпечності процесів автентифікації та управління обліковими записами.

Нормативне підґрунтя для обґрунтування важливості цього механізму міститься у стандарті NIST SP 800-63B, а саме в розділі 5.1.2. Хоча стандарт прямо не зобов'язує застосовувати багатофакторну автентифікацію під час зміни пароля, він рекомендує забезпечувати додатковий захист для sensitive operations – тобто операцій, які можуть мати серйозні наслідки у випадку компрометації.

З погляду інформаційної безпеки, використання MFA при зміні пароля дозволяє нейтралізувати ризики, що виникають у ситуаціях, коли зловмисник уже має доступ до первинного автентифікатора (наприклад, пароля), але ще не отримав повного контролю над обліковим записом. У такому випадку другий фактор (одноразовий код, push-підтвердження, апаратний токен тощо) слугує надійним бар'єром для ескалації доступу. Таким чином, реалізація MFA у цьому контексті виступає як засіб зміцнення довіри до автентифікації перед критичними змінами, і є продовженням загального принципу defense-in-depth.

У межах методу оцінювання, критерій оцінюється за трирівневою шкалою:

- *0 балів* – MFA не використовується зовсім. Користувач може змінити пароль, надавши лише поточний пароль, що створює значну вразливість у разі компрометації автентифікатора;
- *0.5 бала* – MFA застосовується, але тільки в певних умовах: наприклад, при зміні пароля з нового пристрою, географічно підозрілої IP-адреси або за наявності додаткових тригерів ризику;
- *1 бал* – багатофакторна автентифікація завжди використовується при зміні пароля, незалежно від контексту, що забезпечує постійний рівень захисту.

З погляду реалізації, така вимога не є надмірною або технічно складною: більшість сучасних фреймворків автентифікації підтримують інтеграцію MFA

через стандартні API або сервіси (наприклад, Google Authenticator, SMS-шлюзи, апаратні токени), що робить впровадження доступним навіть для малих і середніх проєктів.

Таким чином, критерій використання MFA при зміні пароля є показником зрілості та проактивності безпекової моделі вебдодатку. Його включення до системи оцінювання дозволяє виявити не лише базову відповідність стандартам, а й ступінь готовності системи протистояти сучасним загрозам, які пов'язані з обходом класичних однофакторних систем автентифікації. З методичної точки зору, цей критерій сприяє підвищенню точності кількісної оцінки безпеки, оскільки враховує не тільки формальну присутність контролю, а й його адаптивність до ризику.

Критерій №7: «Чи є захист від brute-force атак під час зміни пароля?»

Захист від атак перебору (brute-force) є одним із базових елементів сучасної системи автентифікації, зокрема в контексті обробки чутливих операцій, таких як зміна пароля користувача. Критерій дозволяє оцінити, наскільки система здатна виявляти та блокувати багаторазові неуспішні спроби взаємодії, спрямовані на підбір поточного пароля або обхід перевірочних механізмів.

Хоча в ISO/IEC 27002:2022, а саме в розділі 9.4.3, немає прямої вказівки на обов'язковість впровадження захисту саме на етапі зміни пароля, проте документ містить рекомендацію щодо лімітування кількості невдалих спроб автентифікації як загального засобу протидії brute-force атакам. У практичній реалізації, механізм захисту від перебору може включати:

- блокування користувача після певної кількості невдалих спроб;
- запровадження часової затримки між спробами;
- виявлення аномальної активності з подальшим логуванням і сповіщенням адміністратора.

У межах процесу зміни пароля ці механізми мають бути активними при будь-якому невдалому введенні поточного пароля або повторної автентифікації, оскільки саме ці точки найчастіше атакуються зловмисниками з метою перехоплення контролю над обліковим записом. Відсутність такого захисту

фактично відкриває систему до безперешкодного підбору пароля, особливо якщо не реалізовано додаткові механізми захисту .

Запропонована трирівнева шкала оцінювання дозволяє гнучко враховувати рівень реалізації цього контролю:

- *0 балів* – відсутність будь-яких обмежень. Користувач (чи зловмисник) може необмежено вводити неправильні паролі, без наслідків;
- *0.5 бала* – часткова реалізація: наприклад, блокування або додаткові перевірки починають діяти лише після великої кількості спроб, або їх можна обійти (через повторне відкриття сесії, зміну IP тощо);
- *1 бал* – повна реалізація захисту: після встановленого ліміту спроб система блокує доступ або активує додаткові етапи перевірки.

З технічної точки зору, реалізація такого контролю є не лише доцільною, але й відносно простою в інтеграції, оскільки переважна більшість фреймворків автентифікації мають вбудовану підтримку таких механізмів. Проте їхня наявність не завжди гарантує належну конфігурацію, тому оцінка за цим критерієм повинна враховувати не лише присутність механізму, а і його ефективність та надійність проти сучасних атак.

Таким чином, критерій захисту від brute-force атак у процесі зміни пароля виконує подвійну функцію: він забезпечує стійкість до технічних атак на рівні інтерфейсу автентифікації, а також дозволяє виявити рівень уваги системи до інфраструктурних аспектів безпеки, які часто ігноруються у фокусі на функціональні перевірки. Його інтеграція до загального методу сприяє формуванню реалістичного уявлення про здатність системи протидіяти масованим атакам у критичних точках доступу.

Критерій №8: «Чи проводиться регулярний аудит для перевірки процесу зміни пароля?»

Регулярне проведення внутрішнього аудиту є однією з ключових умов підтримки ефективного функціонування системи управління інформаційною безпекою (СУІБ), зокрема в частині контролю над критичними процесами автентифікації, як-от зміна пароля. Саме цій складовій присвячений критерій, що

дозволяє оцінити не лише технічні, а й організаційно-процедурні аспекти забезпечення безпеки.

Згідно з положеннями міжнародного стандарту ISO/IEC 27001:2022 [72], а саме розділом 9.2, організація повинна здійснювати регулярні внутрішні аудити з метою перевірки того, наскільки її СУІБ відповідає встановленим вимогам та ефективно виконує функцію контролю.

У контексті управління паролями це означає, що організація повинна мати запланований, документований і повторюваний процес перевірки того, як реалізовано зміну пароля, які технічні й адміністративні засоби при цьому застосовуються, та чи дотримуються відповідні політики. Такий аудит має охоплювати не лише наявність функціональності, а й практичну відповідність її поведінки вимогам стандартів, внутрішніх регламентів та індикаторів безпеки.

Фактичне значення цього критерію полягає в тому, що навіть формально реалізовані механізми безпеки можуть бути неефективними без належного контролю за їх дотриманням, а виявити це можливо лише через системний моніторинг та аудит. Аудит дає змогу своєчасно:

- виявляти порушення у процедурі зміни пароля (наприклад, обходи перевірок);
- перевіряти правильність логів;
- підтверджувати, що процеси дійсно відповідають політиці та стандартам;
- фіксувати інциденти і прогалини, які не відображаються в інтерфейсі.

У межах запропонованого методу критерій оцінюється за трирівневою шкалою:

- *0 балів* – аудит не проводиться взагалі. Компанія не перевіряє відповідність процесу зміни пароля політикам безпеки або робить це лише формально;
- *0.5 бала* – аудит має місце, але нерегулярно, за відсутності чіткого графіка або лише в окремих випадках (наприклад, під час інцидентів або внутрішніх розслідувань);
- *1 бал* – повноцінний, регулярний аудит, який проводиться відповідно до плану, документується, охоплює всі ключові етапи зміни пароля та генерує звітність для відповідальних осіб.

З організаційного погляду, наявність такого аудиту не лише підвищує рівень контролю, а й виконує функцію зворотного зв’язку для оптимізації політик і процедур у сфері безпеки. Аудит дозволяє виявити неефективні або неактуальні налаштування, верифікувати відповідність функціоналу вимогам стандартів (OWASP ASVS, ISO/IEC, NIST) та бути джерелом об’єктивних даних для прийняття рішень у сфері ризик-менеджменту.

Таким чином, цей критерій розширює оцінювання безпеки за межі суто технічних аспектів, інтегруючи системний організаційний контроль у загальний метод. Його наявність в моделі підвищує довіру до результатів оцінки безпеки та робить метод придатним для впровадження у середовищі з формалізованими підходами до управління інформаційною безпекою, включаючи підприємства, що проходять сертифікацію ISO/IEC 27001.

Критерій №9: «Чи існує механізм для реагування на виявлені порушення у процесі зміни пароля?»

У контексті забезпечення безпеки процесу зміни пароля критично важливо не лише виявляти відхилення від встановлених політик, а й оперативно реагувати на них. Реакція на інциденти є одним із ключових елементів циклу безпеки, який завершує ланцюг «виявлення – аналіз – реагування» та забезпечує зворотний зв’язок до системи управління інформаційною безпекою. Саме це охоплює даний критерій.

Нормативною основою для оцінювання цього аспекту є розділ A.16.1 стандарту ISO/IEC 27001:2022, який регламентує управління інцидентами інформаційної безпеки. У ньому вказано, що організація повинна забезпечити своєчасне та координоване реагування на виявлені порушення політик або аномальні події.

У прикладній площині це означає, що в разі виявлення підозрілої активності під час або після спроби зміни пароля, система має ініціювати автоматичні або напівавтоматичні заходи реагування. Це може бути:

- блокування облікового запису до підтвердження особи;
- інформування адміністратора або користувача;

- реєстрація інциденту в системі SIEM;
- запуск додаткового логування або моніторингу.

Тому, цей критерій фокусується не лише на наявності технічної реалізації, а передусім на наявності процедурного ланцюга, який дає змогу перейти від виявлення проблеми до її обробки та усунення. Для забезпечення об’єктивності оцінювання запропоновано трирівневу шкалу.

- *0 балів* – відсутність будь-якого механізму реагування. Порушення не фіксуються, не аналізуються та не призводять до жодних дій;
- *0.5 бала* – механізм формально існує, але працює неефективно: наприклад, події реєструються, але не перевіряються вручну, або ж повідомлення не викликають жодної реакції;
- *1 бал* – реалізовано повноцінний механізм реагування, що функціонує автоматично або за підтримки персоналу, включаючи блокування облікового запису, інформування, запис до журналів аудиту та подальший аналіз інциденту.

Цей критерій є особливо важливим у контексті динамічного реагування на атаки, які вже перебувають у фазі виконання, тобто коли потенційна шкода може бути запобігнута лише завдяки швидкому втручанню. Його інтеграція до моделі дозволяє оцінити готовність системи до роботи в реальному часі з інцидентами, які супроводжують або безпосередньо випливають із дій користувача у рамках зміни автентифікаційних даних.

Таким чином, критерій завершує логіку перевірки вимоги до безпеки зміни пароля, розширюючи її з технічного рівня на операційно-процедурний вимір. Його наявність у загальній методиці дозволяє підвищити рівень відповідності системи сучасним вимогам управління інформаційною безпекою та зробити кількісну оцінку більш повною, верифікованою і релевантною до індустріальних практик.

Ретельний аналіз вимоги 2.1.6 OWASP ASVS засвідчив, що безпечна реалізація механізму зміни пароля потребує не лише технічної перевірки введених даних, а й забезпечення комплексного контролю за автентичністю, відповідністю політикам безпеки, захистом від атак перебору, багатофакторною перевіркою особи, наявністю системи сповіщень та внутрішнього аудиту. Запропонована

модель охоплює як функціональні, так і організаційні аспекти реалізації цієї вимоги, опираючись на положення міжнародних стандартів ISO/IEC 27001:2022, ISO/IEC 27002:2022 та NIST SP 800-63B. У разі відсутності прямого формулювання в стандартах використовувалися обґрунтовані інтерпретації рекомендацій, що дозволяло встановити нормативну відповідність навіть для практично орієнтованих критеріїв. Її застосування дозволяє здійснити кількісну оцінку рівня відповідності безпеки зміни пароля та забезпечує об’єктивну основу для аудиту, вдосконалення політик і практичного впровадження захисних заходів.

Схожим способом було визначено критерії та методи їх оцінювання для кожної вимоги. Тому опис вимоги в межах даного дослідження не є простим переказом формулювання з ASVS, а становить формалізовану модель перевірки, яка включає: аналітичне тлумачення, систему критеріїв з прикладами, прив’язку до міжнародних нормативів, та логіку оцінювання. Це дозволяє вивести оцінку безпеки з площини декларативного підтвердження у площину об’єктивного кількісного аналізу, що робить запропонований метод придатним для практичного застосування як у внутрішньому аудиті, так і в межах зовнішніх оцінювальних процедур.

2.4. Адаптивний метод кількісної оцінки захищеності вебдодатків

Запропонований метод кількісної оцінки рівня захищеності вебдодатків на основі стандарту OWASP ASVS демонструє об’єктивність та практичну корисність у реальних умовах. Цей підхід дозволяє отримати кількісну оцінку рівня захищеності вебдодатків на етапі експлуатації. Метод ґрунтується на чітко структурованому наборі вимог, що охоплюють широкий спектр аспектів безпеки, починаючи від автентифікації користувачів і управління сесіями, до захисту від атак та належного зберігання конфіденційної інформації. Побудована система критеріїв для перевірки кожної вимоги охоплює повною мірою тестування вимоги і дозволяє створити уніфіковану систему для тестування безпеки вебдодатку за умови відсутності технічної документації. Кількісні показники оцінки критеріїв не

лише відображають поточний стан захищеності вебдодатку, а є інструментом для систематичного аналізу впроваджених заходів безпеки.

Однак, слід підкреслити, що даний метод може отримати подальший розвиток з врахуванням специфіки різних вебдодатків, особливо в контексті варіативності наборів вимог до безпеки. Різні вебдодатки мають різну архітектуру та функціональність, що визначає їх потреби у захисті. Наприклад, вебдодатки, які не використовують API-інтерфейси, не повинні отримувати знижені оцінки за відсутність впровадження вимог до API, адже ці вимоги для них є нерелевантними. Перед тим як розпочати процес оцінювання, необхідно здійснити попередній аналіз вебдодатка користувача з метою визначення його архітектурних особливостей і функціональних характеристик. Цей етап дозволить коректно сформувати набір вимог до безпеки, що відповідає специфіці конкретного вебдодатка. Очевидно, що в залежності від складності додатку для його тестування може бути застосована різна кількість вимог. Тому актуальним є питання розширення запропонованого підходу шляхом створення адаптивної системи оцінювання, яка передбачає індивідуалізацію вимог для кожного вебдодатка.

Проте, важливо враховувати не лише технічні характеристики вебдодатків, але й їхній бізнес-контекст, вимоги до конфіденційності та критичність захисту даних. Адаптивний підхід до оцінювання дозволить також краще враховувати специфіку різних галузей застосування вебдодатків. Наприклад, у фінансових або медичних системах окремі вимоги до безпеки можуть мати підвищену важливість, тоді як для стандартних комерційних вебдодатків такі вимоги можуть бути менш критичними. Індивідуалізація процесу оцінювання на ранньому етапі також дозволить отримати більш точні результати для подальшого вдосконалення безпеки, зокрема визначити конкретні зони ризику та сконцентрувати увагу на найбільш вразливих аспектах.

Тому, впровадження адаптивної моделі оцінювання, яка враховує специфіку вебдодатків, є важливим кроком у підвищенні точності та об’єктивності оцінювання захищеності вебдодатків. Такий підхід дозволить уникнути помилкових знижень загального коефіцієнта безпеки для додатків, які не

використовують певні технології або функції, й, навпаки, забезпечить фокус на найбільш критичних аспектах для кожного конкретного вебдодатка.

З вищесказаного можна зробити висновки про те, що оцінки за окремими критеріями не дають загального уявлення про якість виконання окремої вимоги чи оцінку безпеки вебдодатку загалом. Крім того, для врахування особливостей архітектури вебдодатку, діючого функціоналу та наявної документації було запропоновано ввести коефіцієнти важливості вимог та критеріїв. Цей етап передбачає суб'єктивну оцінку експертом, який, спираючись на власний досвід та розуміння критичності функціоналу, присвоює коефіцієнти важливості кожній вимогі в межах розділу та кожному критерію в межах вимоги. Для кращого розуміння розглянемо структуру методу (див. рис. 2.2).

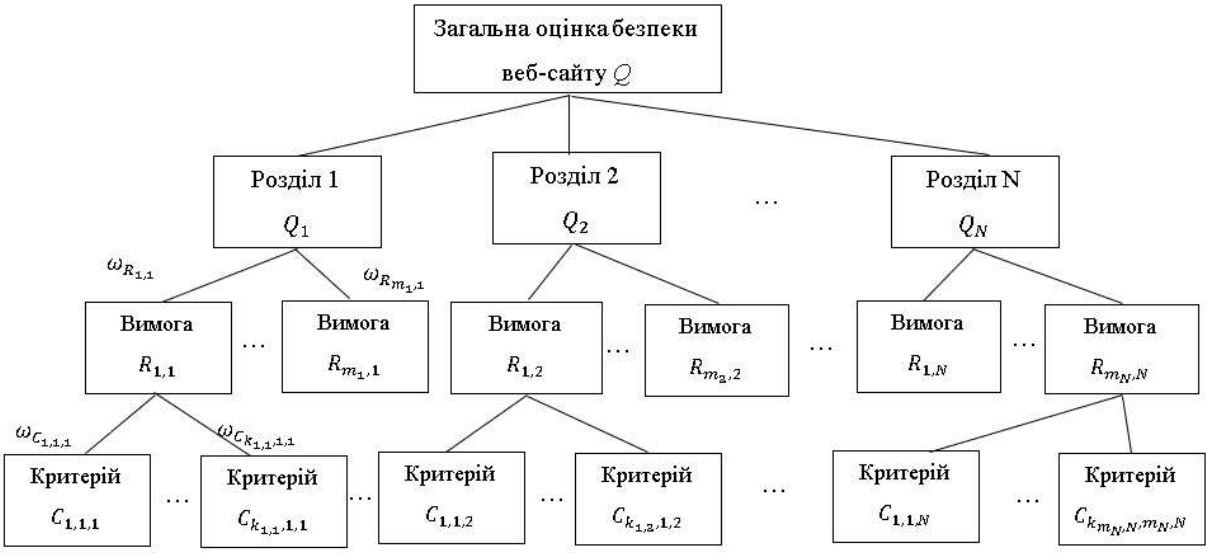

Рис. 2.2. Загальна структура адаптивного методу для оцінювання вебдодатку

Для кращого розуміння суті рисунку розпишемо детальніше використані позначення:

$R_{m,n}$ - оцінка *m-ої* вимоги в межах *n-го* розділу, $m=1..m_n$, $n=1..N$, m_n - кількість вимог в межах *n-го* розділу, причому $m_1+m_2+...+,_N = M$, M - загальна кількість вибраних вимог, N - загальна кількість вибраних розділів зі стандарту ASVS.

$C_{k,m,n}$ - значення *k-го* критерію *m-ої* вимоги в межах *n-го* розділу, *k=1..*$k_{m,n}$, $k_{m,n}$ - кількість розроблених авторами критеріїв для оцінювання *m-ої* вимоги *n-го* розділу. Нагадаємо що оцінка кожного критерію відбувається за одним з трьох значень “0”, “0,5” та “1”.

$\omega_{C_{k,m,n}}$ - коефіцієнт важливості *k-го* критерію *m-ої* вимоги *n-го* розділу, що визначається технічним експертом з врахуванням того, як даний критерій впливає на виконання вимоги, до якої він застосовується. Ці коефіцієнти визначені авторами методу апріорі. Однак, експерт, який проводить оцінку, має можливість коригувати ці значення, враховуючи специфіку досліджуваного вебдодатку та вибравши зручну для себе шкалу оцінювання.

$\omega_{R_{m,n}}$- коефіцієнт важливості *k-го* критерію *m-ої* вимоги *n-го* розділу, що визначається для кожної вимоги аудитором, що проводить оцінку безпеки вебдодатку в залежності від наявної інформації, архітектури та функціоналу системи, а також з огляду на бізнес-контекст, вимоги до захисту інформації. Аудитор може вибрати зручну для себе шкалу для оцінювання важливості вимог. В даній роботі запропоновано використовувати шкалу від 0 (вказує на відсутність функціоналу, що відповідає даній вимогі, в вебдодатку) до 10 (критично важлива вимога для безпеки функціоналу).

На першому кроці обчислень необхідно провести оцінку виконання кожної вимоги. Для цього спершу здійснимо нормалізацію коефіцієнтів важливості критеріїв в межах даної вимоги за формулою:

$$\varpi_{C_{k,m,n}} = \frac{\omega_{C_{k,m,n}}}{\sum_{k=1}^{k_{m,n}} \omega_{C_{k,m,n}}} \quad (1)$$

де - $\varpi_{C_{k,m,n}}$ нормалізований ваговий коефіцієнт *k-го* критерію *m-ої* вимоги *n-го* розділу, $\sum_{k=1}^{k_{m,n}} \omega_{C_{k,m,n}}$ - загальна сума вагових коефіцієнтів критеріїв *m-ої* вимоги *n-го* розділу.

Тоді інтегральну оцінку виконання *m-ої* вимоги *n-го* розділу можна отримати як збалансовану ваговими коефіцієнтами суму оцінок експерта безпеки функціоналу вебдодатку за критеріями в межах даної вимоги:

$$R_{m,n} = \sum_{k=1}^{k_{m,n}} \varpi_{C_{k,m,n}} \cdot C_{k,m,n} \tag{2}$$

Для отримання оцінок для кожного розділу необхідно спершу нормалізувати коефіцієнти важливості вимог. Це можна зробити за формулою:

$$\varpi_{R_{m,n}} = \frac{\omega_{R_{m,n}}}{\sum_{m=1}^{m_n} \omega_{R_{m,n}}} \tag{3}$$

де $\varpi_{R_{m,n}}$ - нормалізований ваговий коефіцієнт *m-ої* вимоги *n-го* розділу, $\sum_{m=1}^{m_n} \omega_{R_{m,n}}$ - сума вагових коефіцієнтів вимог *n-го* розділу.

Наступним етапом розрахунків є визначення кількісної оцінки захищеності вебдодатку для певного розділу. Оцінку безпеки *n-го* розділу з врахуванням особливостей структури сайту та індивідуального підходу аудитора можна обчислити як:

$$Q_n = \sum_{k=1}^{m_{,n}} \varpi_{R_{m,n}} \cdot R_{m,n} \tag{4}$$

Завершальним етапом є інтегрована оцінка безпеки сайту, яку можна обчислити, як середню оцінку безпеки всіх розділів

$$Q = \frac{\sum_{i=1}^{N} Q_n}{N} \tag{5}$$

Отже, в основі розробленого методу лежить диференційований підхід до класифікації вимог безпеки, що враховує рівень критичності захисту даних та

специфіку функціональних компонентів вебдодатку і дозволяє отримати кількісну метрику оцінки безпеки вебдодатку.

2.5 Оцінка та узгодження коефіцієнтів важливості

Оцінка коефіцієнтів важливості критеріїв – це критичний і водночас складний етап у будь-якому багатокритеріальному аналізі (особливо в задачах прийняття рішень, експертного оцінювання чи побудови моделей). Правильна постановка акцентів дозволяє виділити стратегічні критерії, яким слід приділити більше уваги та виділити більше фінансування. В той самий час неправильна оцінка ваг може повністю змінити пріоритети альтернатив або призвести до хибного вибору. В будь-якому випадку ваги критеріїв безпосередньо впливають на вибір кращої альтернативи в задачах прийняття рішень, а також на кінцеву оцінку в задачах оцінювання за багатьма критеріями.

В той самий час, автори [73, 74] стверджують, що існує низка викликів та проблем в експертному оцінюванні. Зокрема, в [75] стверджується, що оцінювання коефіцієнтів важливості має суб’єктивний характер, оскільки кожен експерт спирається на свій досвід, знання, інтуїцію, тому оцінки різних експертів можуть суттєво відрізнятись. Іншою проблемою є те, що експерту іноді важко однозначно виставити певний детермінований бал оцінці, експерти часто сумніваються між суміжними оцінками [76]. Крім того, складно порівнювати критерії різної природи. Наприклад, критерії «вартість проєкту» і «рівень екологічної безпеки» вимірюються різними шкалами і мають різний зміст. Ще одним викликом при оцінюванні вважають взаємозалежність критеріїв. Для прикладу, підвищення надійності може вплинути на вартість. У такому випадку складно оцінити важливість цих критеріїв ізольовано.

Для усунення невизначеності експерта при оцінюванні багато науковців пропонують використовувати теорію нечітких множин [77, 78]. Для уникнення суб’єктивності оцінки, важливість критеріїв в нашій задачі було запропоновано оцінити трьом різних експертам галузі.

В дисертаційній роботі побудовано систему нечіткої логіки для оцінювання вагових коефіцієнтів критеріїв та вимог на основі [79]. На рисунку 2.3 наведено структурну схему запропонованої системи для знаходження вагового коефіцієнта на основі експертних оцінок трьох експертів з врахуванням невизначеності.

Рис. 2.3. Структурна схема системи нечіткої логіки оцінювання коефіцієнту важливості критерію з врахуванням невизначеності та неузгодженості думок експертів.

Запропонований метод оцінки безпеки сайту передбачає оцінювання коефіцієнтів важливості для критеріїв та вимог безпеки. Розглянемо більш детально процес оцінювання одного такого коефіцієнта трьома експертами. Кожному з експертів було запропоновано виставити коефіцієнт важливості вимоги за шкалою від 0 (вимога не застосовна або абсолютно не важлива) до 10

(максимальний рівень важливості). Тобто на вхід в систему поступає три оцінки 1, 2, 3. Після цього відбувається фазифікація коефіцієнтів - процес перетворення чітких точних, вхідних даних на нечіткі множини. У контексті нечіткої логіки, фазифікація дозволяє системі працювати з невизначеністю та суб'єктивністю. Після цього отримуємо нечіткі множини W1, W2, W3, що відповідають думці кожного з трьох експертів відповідно. На наступному етапі застосовуємо правило узгодження думок експертів шляхом переобчислення функції належності для кожного можливого значення оцінки. Останнім етапом виступає дефазифікація – обернена операція до фазифікації, яка відповідає за перетворення нечітких множин у чітке числове значення.

Розглянемо детальніше кожен з описаних етапів. Для початку розглянемо коротко теоретичні відомості теорії нечітких множин, які лежать в основі запропонованої системи оцінювання.

2.5.1 Елементи теорії нечітких множин

Розглянемо елементи теорії нечітких множин. Класичною множиною X називається сукупність об’єктів, які виступають елементами цієї множини та об’єднані певною ознакою. Традиційно таку множину записують у вигляді:

$$X = \{x_1, x_2, \dots x_n\}, \qquad n = \overline{1, N} \tag{6}$$

Нечіткою множиною називають сукупність пар виду $(x, \mu(x))$, де $x \epsilon X$, а$\mu(x)$ називають функцією належності, що визначається відображенням: $\mu(x): X \to [0,1]$. Тоді нечітку множину $\tilde{X}$ можна подати у вигляді:

$$\tilde{X} = \{(x_1, \mu(x_1)), (x_2, \mu(x_2)), \dots (x_n, \mu(x_n))\}, \qquad n = \overline{1, N}, \;\; \mu(x): X \to [0,1] \tag{7}$$

В [80] наведено кілька варіантів функції належності, зокрема трикутну, параболічну, екстрапараболічну. Крім того, існують трапецеподібна функція, сигмоїдна, гаусова та інші.

Опишемо детальніше кілька функцій належності. Зокрема трикутна функція має вигляд:

$$(x) = \begin{cases} 1, & x = b \\ \dfrac{x-a}{b-a}, & a < x < b \\ \dfrac{c-x}{c-b}, & b < x < c \\ 0, & \text{в інших випадках} \end{cases} \tag{8}$$

На рисунку 2.4 приведено приклад такої функції.

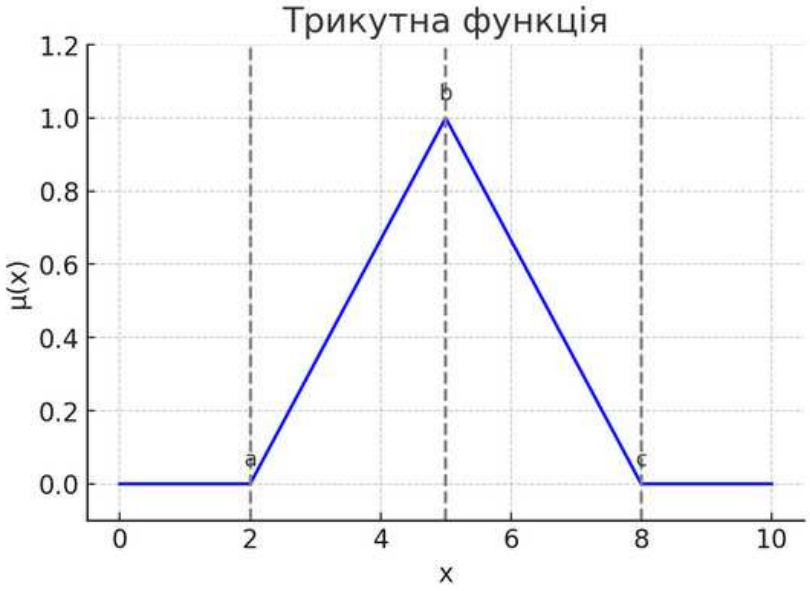

Рис. 2.4. Приклад трикутної функції належності

Трапецеподібна функція описується формулою:

$$\mu(x) = \begin{cases} 1, & b \le x \le c \\ \dfrac{x-a}{b-a}, & a < x < b \\ \dfrac{d-x}{d-c}, & c < x < d \\ 0, & \text{в інших випадках} \end{cases} \tag{9}$$

На рисунку 2.5 приведено приклад такої функції.

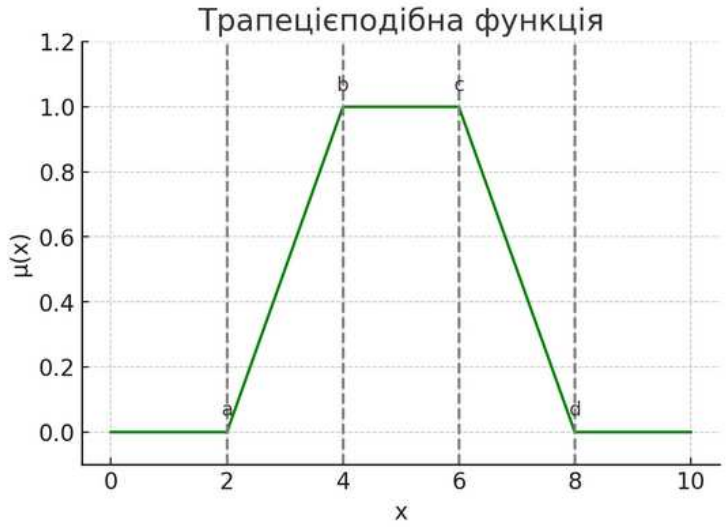

Рис. 2.5. Приклад трапецеподібної функції належності

Гаусова (нормальна крива) – гладка симетрична функція, що добре моделює нечіткі поняття з "розмитим центром", що визначається формулою:

$$(x) = exp\left(-\frac{(x-c)^2}{2\sigma^2}\right) \quad (10)$$

Де c- центр, математичне сподівання, а σ – середньо-квадратичне відхилення (параметри нормального розподілу)

Ця функція використовується, коли потрібно забезпечити м’які границі нечітких множин (див. рис. 2.6).

Рис. 2.6. Приклад гаусової функції належності

Для моделювання понять типу "високий рівень", "велике значення" тощо використовують сигмоїду функцію:

$$\mu(x) = \frac{1}{1 + e^{-a(x-c)}} \tag{11}$$

де a і c – параметри функції, що визначають крутість та центр переходу відповідно.

Приклад графіку такої функції наведено на рисунку 2.7.

Рис.2.7. Приклад сигмоїдної функції належності

2.5.2 Фазифікація оцінок експертів

Фазифікація є першим і ключовим кроком у більшості систем нечіткої логіки. Заміна точних числових оцінок нечіткими множинами дозволяє:

- Моделювати невизначеність: людське мислення часто оперує неточними поняттями. Фазифікація допомагає комп'ютерним системам "розуміти" та обробляти таку невизначеність.
- Спрощувати представлення знань: замість складних математичних моделей, фазифікація дозволяє використовувати інтуїтивно зрозумілі лінгвістичні терміни.
- Створювати більш гнучкі системи: системи, що використовують фазифікацію, можуть краще адаптуватися до мінливих умов і неповних даних.

Отже, на даному етапі фіксована оцінка експерта ω перетворювалась нечітку множину вигляду:

$$W = \{0, \mu(0)), (1, \mu(1)), \dots (10, \mu(10))\} \quad (12)$$

Якщо експерт впевнений в своїй оцінці w і присвоює йому значення функції належності 1, а всім іншим можливим значенням 0, то нечітка множина зрештою трансформується у звичайну множину, яка містить один елемент – ваговий коефіцієнт, який вибрав експерт:

$$W = \{0,0), (1,0), \dots, (w, 1), \dots, (10,0)\} = \{w\} \quad (13)$$

Проте, як показує досвід, експерти часто сумніваються між кількома суміжними оцінками, та зрештою вибирають одну з них. Щоб врахувати невизначеність експерта для проміжних оцінок від 1 до 9 було обрано трикутну функцію належності, що обчислювалася за формулою:

$$\mu(x) = \begin{cases} 1, & x = w \\ \dfrac{x - w + 2}{2}, & w - 2 < x < w \\ \dfrac{w + 2 - x}{2}, & w < x < w + 2 \\ 0, & \text{в інших випадках} \end{cases}, \quad x = \overline{1,9} \tag{14}$$

В такому випадку, якщо експерт після роздумів, зрештою обирав оцінку критерію *w=5*, то нечітка множина, що відповідала цій оцінці мала вигляд:

$$W = \left\{0,0), (1,0), \dots, \left(4, \frac{1}{2}\right), (5,1), \left(6, \frac{1}{2}\right) \dots, (10,0)\right\} \tag{15}$$

Ця функція належності не враховує крайніх значень оцінок, тому у випадку, коли експерт обирав значення вагового коефіцієнта 0, що передбачало відсутність певного функціоналу та незастосовність вимоги, нечітка множина трансформувалась в чітку з єдиним коефіцієнтом 0.

$$W = \{0,1), (1,0), \dots, (w, 0), \dots, (10,0)\} = \{0\} \tag{16}$$

У випадку, якщо експерт присвоював максимальне значення вагового коефіцієнта 10, то функцію належності запропоновано визначати у вигляді

$$W = \left\{0,0), (1,0), \dots, (w, 0), \dots, \left(9, \frac{1}{2}\right), (10,1)\right\} \tag{17}$$

2.5.3 Узгодження оцінок експертів

Коли оцінки експертів виражаються нечіткими множинами, виникає потреба в спеціальних методах для агрегування та узгодження цих суб'єктивних суджень. Ці методи дозволяють отримати єдину, консолідовану оцінку, яка відображає колективну думку групи експертів.

Існує кілька підходів до агрегування нечітких оцінок, які можна класифікувати за різними ознаками [81]:

- методи, засновані на операціях нечіткої алгебри;
- методи, засновані на нечітких інтегралах;
- методи, засновані на відстані та подібності;
- методи, засновані на ієрархічному аналізі.

Нехай після першого кроку маємо три нечіткі множини оцінок експертів:

$\tilde{A} = \{x, \mu_{\tilde{A}}(x)\}, \quad x \in X, \quad X = \{x, 0 \le x \le 10, x \epsilon Z\}, \mu_{\tilde{A}}(x): X \to [0,1]$

$\tilde{B} = \{x, \mu_{\tilde{B}}(x)\}, \quad x \in X, \quad X = \{x, 0 \le x \le 10, x \epsilon Z\}, \mu_{\tilde{B}}(x): X \to [0,1]$

$\tilde{C} = \{x, \mu_{\tilde{C}}(x)\}, \quad x \in X, \quad X = \{x, 0 \le x \le 10, x \epsilon Z\}, \mu_{\tilde{C}}(x): X \to [0,1]$

Перша група методів використовує стандартні операції над нечіткими множинами для об'єднання оцінок. До них відносять операцію перетину, що застосовує оператор мінімуму (min) для об'єднання нечітких множин та вважається консервативним підходом. Результатом є нечітка множина, яка представляє "спільну" частину всіх оцінок:

$$\mu_{\tilde{A} \cap \tilde{B} \cap \tilde{C}}(x) = min\left(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x), \mu_{\tilde{C}}(x)\right) \tag{18}$$

Іншою операцією є об’єднання множин, що застосовує оператор максимуму (max) та, по факту, є більш оптимістичних підходом. Формула для обчислення функції належності для двох об’єднаних множин має вигляд:

$$\mu_{\tilde{A} \cup \tilde{B} \cup \tilde{C}}(x) = max\left(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x), \mu_{\tilde{C}}(x)\right) \tag{19}$$

Ще одним компромісним підходом є обчислення середнього значення функцій належності для кожної можливої оцінки експерта. Це простий метод, який забезпечує компроміс між консервативним і оптимістичним підходами.

$$\mu_{\tilde{A}+\tilde{B}+\tilde{C}}(x) = \frac{(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x), \mu_{\tilde{C}}(x))}{3} \tag{20}$$

Ще одним підходом, який враховує довіру до експертів є зважене середнє. Якщо експерти мають різний рівень знань або досвіду, їхнім оцінкам можуть бути присвоєні різні вагові коефіцієнти.

$$\mu_{\tilde{A}+\tilde{B}+\tilde{C}}(x) = w_A \cdot \mu_{\tilde{A}}(x) + w_B \cdot \mu_{\tilde{B}}(x) + w_C \cdot \mu_{\tilde{C}}(x) \quad (21)$$

де w_A, w_B, w_C – вагові коєфіцієнти довіри до експертів. Вагові коефіцієнти можуть обиратись суб'єктивно (за рішенням керівника проєкту) або об'єктивно (на основі попередньої роботи експертів або їхньої узгодженості).

До популярних методів, заснованих на нечітких інтегралах відносять інтеграл Шоке (Choquet Integral) та інтеграл Сугено (Sugeno Integral) [82]. Якщо маємо групу експертів, то звичайна "вага" експерта просто показує, наскільки він важливий. Методи узгодження оцінок на основі нечітких інтегралів дозволяють враховувати взаємозв’язки між експертами та вагу кожної оцінки, забезпечуючи гнучке об’єднання думок. До переваг цих методів відносять узгодженість, оскільки вони дозволяють зменшити вплив крайніх або суперечливих оцінок.

Методи узгодження експертних оцінок, засновані на відстані та подібності, орієнтовані на те, щоб знайти компромісну оцінку, яка найменше відрізняється від індивідуальних оцінок експертів або є найбільш схожою до них у певному сенсі. Подібність оцінок визначається як ступінь схожості між експертними думками, особливо коли використовуються нечіткі множини. Такі методи дозволяють визначити, наскільки думки експертів близькі, і сформувати узгоджену оцінку як опорну точку, що найбільш подібна до всіх інших.

В даній роботі для узгодження думок експертів вибрано метод середнього, як найпростіший метод, що забезпечує компроміс консервативним і оптимістичним підходами. Обчисливши середні значення функціє належності за формулою (19), отримуємо узгоджену нечітку множину оцінок експертів:

$$\widetilde{W} = \{0, \mu(0)), (1, \mu(1)), \dots (10, \mu(10))\} \quad (22)$$

При узгодженні думок експертів важливо пам’ятати про виявлення розбіжностей. Важливо не тільки агрегувати оцінки, а й аналізувати ступінь розбіжностей. Якщо розбіжності занадто великі, це може свідчити про необхідність додаткових дискусій між експертами або залучення нових фахівців. У деяких випадках процес узгодження може бути ітераційним. Після першого агрегування результати можуть бути надані експертам для обговорення та можливої корекції їхніх початкових оцінок.

2.5.4 Етап дефазифікації

Дефазифікація є невід'ємною частиною нечітких систем, оскільки вона дозволяє перетворити нечіткі рекомендації в конкретні дії. Без цього етапу висновки нечіткої системи залишалися б "розмитими" і непридатними для використання в реальному світі. Дефазифікація є завершальним етапом у роботі розробленої системи нечіткої логіки, який полягає у перетворенні нечіткої множини узгоджених оцінок експертів назад у точне, числове значення вагового коефіцієнта.

Існує кілька різних методів дефазифікації, але до найбільш поширених належать: метод максимального значення (Mode), метод центру тяжіння (Center of Gravity - COG), метод центру максимумів (Center of Maxima - COMa), метод середнього максимуму (Mean of Maxima - MoM) [83].

Найбільш популярним методом є метод центру тяжіння. Він обчислює "центр" площі під кривою функції належності вихідної нечіткої множини. Для дискретних множин як (21), по суті, це зважене середнє всіх можливих вихідних значень, де вага визначається ступенем належності:

$$\omega = defuzzification(\widetilde{W}) = \frac{\sum_{i=0}^{N} x_i \cdot \mu(x_i)}{\sum_{i=0}^{N} \mu(x_i)}, \quad x_i = i, N = 10 \tag{23}$$

Цей спосіб є інтуїтивно зрозумілий, проте чутливий до крайніх значень.

Метод максимального значення працює за дуже простим принципом – вибирає x з максимальним значенням функції належності:

$$\omega = defuzzification(\widetilde{W}) = arg\ (\max_{x}(\mu(x)) \quad (24)$$

Якщо є кілька x з максимальним значенням функції належності$\mu(x)$, то точний ваговий коєфіцієнт можна знайти за метод центру максимумів:

$$\omega = defuzzification(\widetilde{W}) = \frac{\sum x_{max}}{k}, \quad (25)$$

Де k – кількість значень x з максимальними значеннями$\mu(x)$, $x_{max} = \arg\ (\max_{x}(\mu(x))$.

Для отримання збалансованої оцінки в системі було використано метод центру тяжіння та обчислювали точні значення коефіцієнтів важливості за формулою (22). В розділі 3 буде приведено приклад розрахунку точного значення вагового коефіцієнта на основі узгоджених оцінок трьох експертів з врахуванням їх невизначеності.

2.6 Висновок до розділу 2

1. Проведено аналіз проблеми у сфері оцінювання безпеки сучасних вебдодатків, а саме значної частки суб'єктивності експертних оцінок та відсутності уніфікованих кількісних метрик, що дало змогу обґрунтувати необхідність розробки нових підходів до об'єктивізації процесів оцінювання захищеності.
2. Досліджено обмеження існуючих міжнародних стандартів (ISO/IEC 27004, ISO/IEC 27002) та традиційних методів тестування безпеки (SAST, DAST, методи "чорного", "білого", "сірого" ящиків), що дало змогу встановити їхню

неспроможність надавати інтегральний показник захищеності та виявляти вразливості бізнес-логіки.

3. Розроблено адаптивний метод кількісної оцінки захищеності вебдодатків на основі структурного аналізу та селективного відбору вимог стандарту OWASP Application Security Verification Standard (ASVS) з формалізацією відібраних вимог через чітко визначені критерії оцінювання за шкалою, що дало змогу створити систематизований підхід до кількісного вимірювання рівня безпеки.

4. Запропоновано введення адаптивних коефіцієнтів важливості для вимог та критеріїв з урахуванням архітектурних, функціональних та бізнес-особливостей вебдодатку та застосування апарату нечіткої логіки для об'єктивізації процесу їх визначення, що дало змогу мінімізувати суб'єктивність експертних суджень та підвищити точність оцінювання.

Отже, у даному розділі закладено науково-методичні засади для об'єктивного, прозорого та відтворюваного кількісного вимірювання рівня захищеності вебдодатків, що створює підґрунтя для експериментальної перевірки та практичного застосування розробленого методу.

РОЗДІЛ 3. ВЕРИФІКАЦІЯ ЕФЕКТИВНОСТІ АДАПТИВНОГО МЕТОДУ КІЛЬКІСНОЇ ОЦІНКИ ЗАХИЩЕНОСТІ ВЕБДОДАТКІВ ТА ЙОГО РЕАЛІЗАЦІЯ НА ПРИКЛАДІ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

3.1 Експериментальне дослідження рівня безпеки вебдодатку eCommerce

Розроблений метод кількісного оцінювання якості безпеки вебсайтів характеризується значною комплексністю та багаторівневою структурою. Згідно з результатами дослідження, представленими у розділі 2, метод включає 133 специфікованих вимоги до безпеки, кожна з яких деталізується через 5-10 критеріїв оцінювання, що забезпечує всебічний аналіз потенційних векторів загроз та вразливостей. Водночас, практичне застосування методу передбачає адаптивний підхід до вибору релевантних критеріїв оцінювання залежно від функціональних особливостей та архітектурних рішень конкретного вебдадатку. Така гнучкість підходу дозволяє оптимізувати процес тестування шляхом відбору, унаслідок чого набір застосовуваних критеріїв може суттєво варіюватися між різними об'єктами дослідження, забезпечуючи при цьому максимальну ефективність та релевантність проведеного аналізу.

3.1.1 Загальний аналіз "OWASP Juice Shop" як об’єкту дослідження

Предметом дослідження є навчальний вебдодаток "OWASP Juice Shop" [84], який являє собою спеціалізоване середовище для тестування безпеки електронної комерції. Цей додаток є репрезентативним зразком сучасного інтернет-магазину, що містить широкий спектр завчасно впроваджених вразливостей та офіційно дозволений для використання в освітніх цілях. У рамках даного дослідження проводиться комплексна оцінка безпеки цього вебдодатку з метою отримання кількісного показника якості захищеності.

Для забезпечення відповідності нормативно-правовим вимогам і дотримання авторських прав у даному дослідженні застосовано загальновідомий навчальний ресурс, який є спеціалізованим середовищем для тестування безпеки.

OWASP Juice Shop включає всі ключові функціональні можливості, властиві сучасним інтернет-магазинам. Додаток забезпечує управління каталогом продукції з підтримкою пошуку, сортування та детального перегляду товарів, функціонал кошика для маніпуляцій із товарами, а також інтегровані механізми оформлення замовлень. Він підтримує багаторівневу систему доступу, включаючи реєстрацію й авторизацію користувачів із розподілом на ролі (наприклад, звичайний користувач і адміністратор), що реалізовано через впровадження JWT токенів. Серверна частина додатку побудована на фреймворці Node.js, яка забезпечує високу продуктивність і масштабованість, тоді як SQLite використовується для зберігання даних, таких як інформація про товари, користувачів та замовлення. Фронтенд реалізовано за допомогою фреймворку Angular, що дозволяє створювати динамічний та зручний інтерфейс користувача.

Більше того, OWASP Juice Shop спеціально розроблений із впровадженими вразливостями, включаючи SQL-ін'єкції, міжсайтовий скриптинг (XSS) і атаки типу CSRF. Це дозволяє імітувати реальні кіберзагрози, характерні для систем електронної комерції, і розробляти стратегії їх виявлення та нейтралізації. Завдяки підтримці контейнеризації через Docker, додаток забезпечує спрощене налаштування середовища тестування як для локальних, так і для хмарних інфраструктур. Таким чином, OWASP Juice Shop є важливим інструментом для систематичного аналізу ризиків і вивчення принципів забезпечення безпеки у сфері електронної комерції.

Крім того, OWASP Juice Shop дозволяє проводити тестування широкого спектра сценаріїв, включаючи перевірку вразливостей, пов'язаних із недостатньою валідацією даних, використанням застарілого або небезпечного програмного забезпечення, а також некоректною конфігурацією серверної частини. Це забезпечує інтеграцію реальних кейсів у навчальний процес.

Додаток також підтримує реалізацію принципу DevSecOps завдяки можливості інтеграції із сучасними інструментами CI/CD. Це дозволяє проводити автоматизоване тестування безпеки на кожному етапі розробки.

У контексті досліджень у сфері кібербезпеки, OWASP Juice Shop надає платформу для емпіричного аналізу ефективності різних методів захисту, включаючи динамічне і статичне тестування, а також застосування гібридних підходів. Такий підхід дозволяє отримати глибше розуміння потенційних ризиків і механізмів їх мінімізації, що є критично важливим для забезпечення стійкості сучасних систем електронної комерції.

Відповідно до методу, описаної у другому розділі, першим етапом оцінювання безпеки є вибірка релевантних вимог оцінювання. Важливо зазначити, що загальний набір вимог ASVS стосується різних етапів розробки програмного забезпечення, включаючи проєктування, розробку, тестування та експлуатацію. В даному випадку, маючи лише доступ до етапу експлуатації готового вебдодатку, було відібрано лише ту множину вимог, яку є можливість перевірити для готового продукту без доступу до документації та процесів розробки. Тому з кожного розділу ASVS було відібрано саме ті вимоги захищеності, які користувач може перевірити вручну на даному сайті.

Наступним етапом оцінювання безпеки стало застосування системи розроблених критеріїв для кожної відібраної вимоги. Як зазначено у другому розділі, для кожної вимоги було розроблено набір чітко визначених критеріїв оцінювання, що дозволяють розкласти абстрактне формулювання вимоги на операційно доступні, логічно завершені та технічно здійсненні одиниці аналізу. Такий підхід дозволяє уникнути суб'єктивізму в процесі оцінювання та забезпечує високий ступінь деталізації безпеки на рівні практичної реалізації функціоналу. Усі критерії були приведені до уніфікованої шкали оцінювання з кроком 0,5. Це дозволяє гнучко враховувати не лише бінарну наявність або відсутність функціоналу, але й часткову реалізацію або обмежене покриття вимоги, надаючи кожній вимогі кількісне значення, що відображає реальний рівень її дотримання у межах конкретного проєкту.

Таблиця 3.1 містить перелік відібраних розділів із кількістю вибраних вимог та кількістю побудованих критеріїв для перевірки цих вимог, за умови відсутності доступу до документації щодо SDLC проєкту. Оскільки максимальна оцінка

кожного критерію - “1”, то кількість критеріїв фактично і визначає максимальну кількість балів оцінювання за розділ. Для кожного розділу було визначено максимальну кількість балів, яку можна одержати, оцінюючи всі відібрані вимоги захищеності за допомогою розроблених критеріїв оцінювання до кожної вимоги. Ці бали варіюються залежно від кількості вимог та переліку критеріїв оцінювання для них у кожному розділі.

Таблиця 3.1 - Статистика підрозділів, вимог та критеріїв для оцінки вебдодатку інтернет-магазину

Назва розділу	**Кількість вимог**	**Кількість критеріїв**
Authentication / Автентифікація	12	85
Session Management / Керування сеансами	10	54
Access Control / Контроль доступу	5	43
Validation, Sanitization and Encoding / Валідація, санітизація та кодування	6	57
Data Protection / Захист даних	6	52
Files and Resources / Файли та ресурси	3	30
Configuration / Конфігурація	7	44
Загальна кількість	49	365

Згідно з визначеними критеріями та детальним описом, можливо провести практичний аналіз вебдодатку на наявність вразливостей, використовуючи його як звичайний користувач. Отримані результати дозволяють виконати кількісну оцінку виявлених вразливостей.

Оскільки в проєкті реалізовано процес автентифікації, усі вимоги з цього розділу слід вважати обов'язковими. Розглянемо детальніше розділ автентифікації, що загалом містить 12 вимог.

Таблиця 3.2 містить кількісні оцінки вимог, оцінених за розробленими критеріями та загальний показник захисту для розділу “Автентифікація”, який містить дванадцять безпекових вимог.

Таблиця 3.2 - Список вимог із кількістю критеріїв перевірки вимоги

Вимога	**Кількість критеріїв для перевірки вимоги**
Перевірте, чи встановлені користувачем паролі мають довжину щонайменше 12 символів (після об'єднання кількох пробілів)	6
Переконайтеся, що дозволені паролі довжиною щонайменше 64 символи, а паролі довжиною понад 128 символів заборонені.	6
Переконайтеся, що не виконується скорочення пароля. Однак, кілька послідовних пробілів можна замінити одним пробілом.	5
Перевірте, чи дозволено використовувати в паролях будь-які друковані символи Unicode, включаючи нейтральні для мови символи, такі як пробіли та емодзі.	7
Перевірте, чи можуть користувачі змінювати свій пароль.	8
Переконайтеся, що для функції зміни пароля потрібні поточний та новий пароль користувача.	9

Продовження таблиці 3.2

Перевірте, чи немає правил складання паролів, які обмежують тип дозволених символів. *	11
Переконайтеся, що користувач може тимчасово переглянути весь замаскований пароль або тимчасово переглянути останній введений символ пароля на платформах, які не мають цієї вбудованої функції.	8
Перевірте, чи ефективні засоби контролю за автоматизацією для зменшення ризиків порушення перевірки облікових даних, атак методом повного перебору та блокування облікових записів.**	9
Перевірте, чи безпечні сповіщення надсилаються користувачам після оновлення даних автентифікації, таких як скидання облікових даних, зміна електронної пошти або адреси, вхід з невідомих або ризикованих місць.***	6
Перевірте, чи немає підказок для пароля або автентифікації на основі знань (так званих «секретних питань»).	5
Відновлення облікових даних перевірки пароля жодним чином не розкриває поточний пароль.	5

*Не повинно бути вимог щодо використання великих або малих літер, цифр або спеціальних символів.

**Такі засоби контролю включають блокування найпоширеніших зламаних паролів, програмне блокування, обмеження швидкості, CAPTCHA, постійно зростаючі затримки між спробами, обмеження IP-адреси або обмеження на основі ризику, такі як місцезнаходження, перший вхід на пристрої, нещодавні спроби розблокування облікового запису тощо. Переконайтеся, що для одного облікового запису можливе не більше 100 невдалих спроб на годину.

***Бажано використовувати push-сповіщення замість SMS або електронної пошти, але за відсутності push-сповіщень SMS або електронна пошта є прийнятними, якщо в сповіщенні не розкривається конфіденційна інформація.

3.1.2 Оцінка безпеки вебдодатку без врахування коефіцієнтів важливості

У статті [85] наведено оцінку рівня захищеності вебдодатку без врахування коефіцієнтів важливості для окремих вимог, що дозволило наочно продемонструвати базову реалізацію авторського методу кількісного оцінювання безпеки вебдодатків за допомогою єдиної уніфікованої шкали оцінювання кожного критерію. Для демонстрації процедури оцінювання розглянемо конкретний приклад вимоги "Verify that password change functionality requires the user's current and new password". Згідно з наведеною таблицею 3.2, для цієї вимоги було визначено 9 критеріїв оцінювання, кожен з яких має однакову вагу та оцінюється за шкалою від 0 до 1, де 0 означає відсутність реалізації, 0,5 – часткову реалізацію, а 1 – повну реалізацію відповідної функції або механізму безпеки: Оцінювання даної вимоги здійснювалося за критеріями, які наведені в підрозділ 2.3 розділу 2. Нижчу наведено результати кількісних оцінок для кожного з визначених для даної вимоги критеріїв.

Оцінка за критерієм №1 “Чи вимагається введення поточного пароля при зміні пароля?”: 1 бал

Згідно з алгоритмом оцінювання критерію 1 бал присвоюється лише тоді, коли введення поточного пароля або альтернативного автентифікатора (наприклад, біометрії чи OTP) є обов’язковою передумовою для зміни пароля в усіх випадках, незалежно від стану сесії користувача. В інтернет-магазині OWASP Juice Shop введення поточного паролю вимагається (див. рис. 3.1).

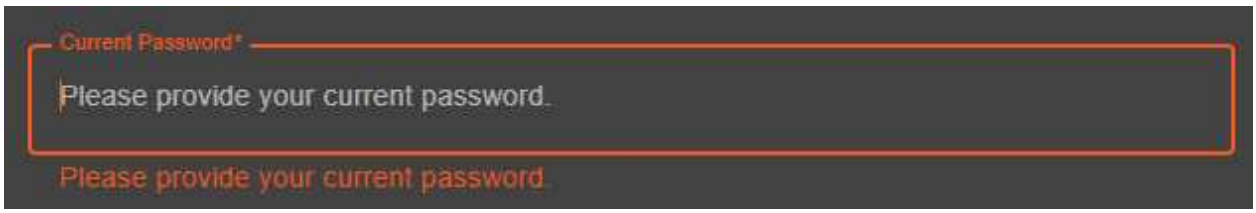

Рис. 3.1. Перевірка вимоги введення поточного пароля

Оцінка за критерієм №2 "Чи вимагається введення нового пароля при зміні пароля?": 0.5 бала

Згідно з алгоритмом оцінювання, 0.5 бала присвоюється, коли присутнє часткове виконання, яке фіксується у випадках, коли введення нового пароля є обов'язковим, але допускається використання того самого значення, що й раніше. Така практика хоча й забезпечує формальне оновлення пароля, однак не сприяє підвищенню безпеки, адже залишає систему вразливою до атак із повторним використанням скомпрометованих або простих паролів. В інтернет-магазині OWASP Juice Shop введення нового паролю вимагається, але можна ввести такий пароль, що й був до того. Попри все, система зберегла новий пароль (див. рис. 3.2).

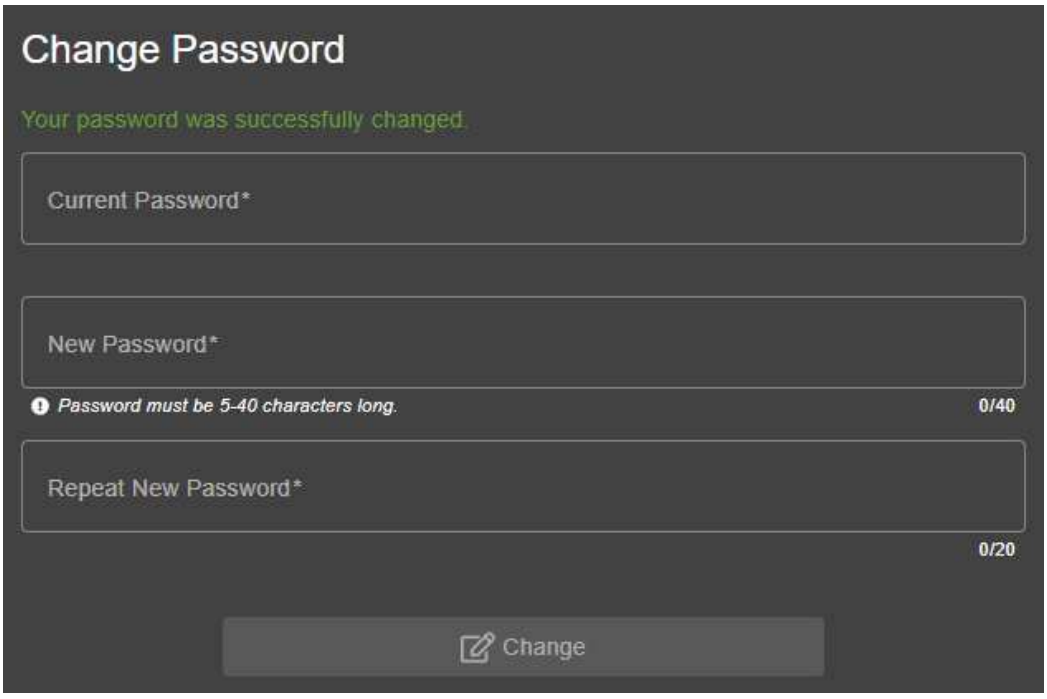

Рис. 3.2. Збереження простого пароля

Оцінка за критерієм №3 "Чи є механізм перевірки правильності поточного пароля?": 1 бал

Згідно з алгоритмом оцінювання, 1 бал отримують лише ті системи, які незмінно вимагають перевірки коректності введеного поточного пароля, незалежно від обставин, і в разі помилки – блокують виконання дії зі зміни автентифікатора. Інтернет магазин повністю відповідає цій вимогі (див. рис. 3.3).

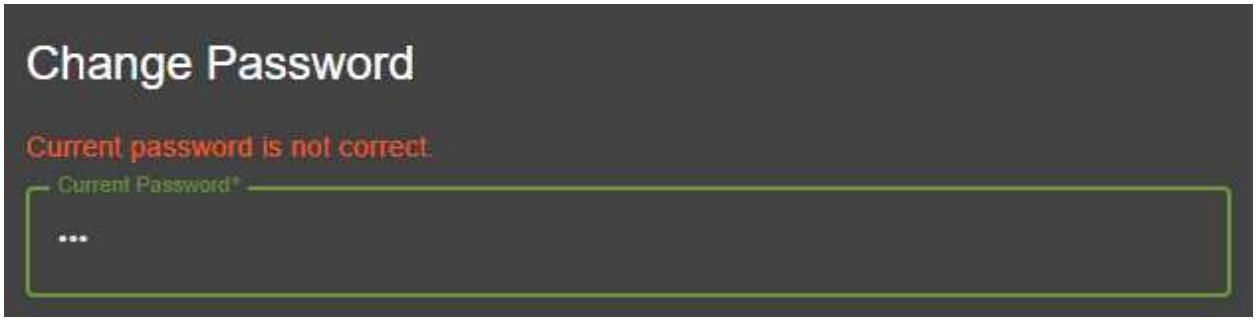

Рис 3.3. Перевірка коректності поточного паролю

Оцінка за критерієм №4 "Чи є механізм перевірки нового пароля на відповідність вимогам безпеки?": 0.5 бала

Згідно з алгоритмом оцінювання, 0.5 бала відповідає частковій реалізації: перевіряється лише один або кілька параметрів, наприклад, довжина, але без перевірки складності, історії чи чорних списків. В досліджуваному проєкті можна ввести пароль з п'яти символів і система буде вважати його прийнятним (див. рис. 3.4).

Рис. 3.4. Механізм перевірки нового пароля

Оцінка за критерієм №5 “Чи є механізм підтвердження зміни пароля через електронну пошту або інший канал зв'язку?”: 0 балів

Згідно з алгоритмом оцінювання, 0 балів визначає випадок, коли система не надсилає жодних повідомлень про зміну пароля, що позбавляє користувача критично важливої інформації про стан облікового запису. Що відповідає поведінці досліджуваного об’єкта.

Оцінка за критерієм №6 “Чи використовується багатофакторна автентифікація (MFA) під час зміни пароля?”: 0 балів

Згідно з алгоритмом оцінювання, 0 балів визначається, коли MFA не використовується зовсім; користувач може змінити пароль, надавши лише поточний пароль, що створює значну вразливість у разі компрометації автентифікатора. Згідно рисунку 2, ніякого функціоналу багатофакторної автентифікації не передбачено.

Оцінка за критерієм №7 “Чи є захист від brute-force атак під час зміни пароля?”: 0 балів

Згідно з алгоритмом оцінювання, 0 балів визначає відсутність будь-яких обмежень: користувач (чи зловмисник) може необмежено вводити неправильні паролі, без наслідків. Ця поведінка відповідає для досліджуваного інтернет-магазину.

Оцінка за критерієм №8 “Чи проводиться регулярний аудит для перевірки процесу зміни пароля?”: 0 балів

Згідно з алгоритмом оцінювання, 0 балів отримує вебдодаток де аудит не проводиться взагалі: компанія не перевіряє відповідність процесу зміни пароля політикам безпеки або робить це лише формально. Так, як OWASP Juice Shop створений для тестування вразливостей і є навчальним додатком - політика - реалізація даної вимоги відсутня.

Оцінка за критерієм №9 “Чи існує механізм для реагування на виявлені порушення у процесі зміни пароля?”: 0 балів

Згідно з алгоритмом оцінювання, 0 балів відповідає відсутності будь-якого механізму реагування: порушення не фіксуються, не аналізуються та не призводять

до жодних дій. Як і в попередньому критерії OWASP Juice Shop створений для тестування вразливостей і є навчальним додатком, тому дані механізми відсутні.

Таким чином сумарно, ця вимога отримує 3,0 бали з 9 можливих. Цей приклад демонструє, як користувач, який має необхідні навички, може оцінити критерії для кожної вимоги. Отримані оцінки можна підсумувати для визначення загального рівня захисту вебдодатка в межах розділу в цілому.

Схожим чином, проводиться оцінка критеріїв для тестування інших вимог розділу автентифікація. Загальна статистика перевірки вимог розділу автентифікація за визначеними критеріями наведена в таблиці 3.3.

Таблиця 3.3–Перелік оцінених вимог за критеріями для розділу “Автентифікація”

Вимога	**Результати оцінювання критеріїв для кожної вимоги**			**Сума набраних балів/ максимальна кількість балів**
	0 балів	**0,5 бала**	**1 бал**	
Перевірте, чи встановлені користувачем паролі мають довжину щонайменше 12 символів (після об'єднання кількох пробілів)	5	1	0	0,5/6
Переконайтеся, що дозволені паролі довжиною щонайменше 64 символи, а паролі довжиною понад 128 символів заборонені.	6	0	0	0/6

Продовження таблиці 3.3

Переконайтеся, що не виконується скорочення пароля. Однак, кілька послідовних пробілів можна замінити одним пробілом.	5	0	0	0/5
Перевірте, чи дозволено використовувати в паролях будь-які друковані символи Unicode, включаючи нейтральні для мови символи, такі як пробіли та емодзі.	4	2	1	2/7
Перевірте, чи можуть користувачі змінювати свій пароль.	5	2	1	2/8
Переконайтеся, що для функції зміни пароля потрібні поточний та новий пароль користувача.	5	2	2	3/9
Перевірте, чи немає правил складання паролів, які обмежують тип дозволених символів. *	4	1	6	6,5/11
Переконайтеся, що користувач може тимчасово переглянути весь замаскований пароль або тимчасово переглянути останній введений символ пароля на платформах, які не мають цієї вбудованої функції.	8	0	0	0/8
Перевірте, чи ефективні засоби контролю за автоматизацією для зменшення ризиків порушення перевірки облікових даних, атак методом повного перебору та блокування облікових записів.**	9	0	0	0/9
Перевірте, чи безпечні сповіщення надсилаються користувачам після оновлення даних автентифікації, таких як скидання облікових даних, зміна електронної пошти або адреси, вхід з невідомих або ризикованих місць.***	2	2	2	3/6
Перевірте, чи немає підказок для пароля або автентифікації на основі знань (так званих «секретних питань»).	5	0	0	0/5

Продовження таблиці 3.3

Перевірте, чи немає правил складання паролів, які обмежують тип дозволених символів. *	5	0	0	0/5
Відновлення облікових даних перевірки пароля жодним чином не розкриває поточний пароль.	2	1	2	2,5/5
Разом	62	11	14	19,5/85

З таблиці 3.3 очевидно, що левова частка критеріїв для оцінка розділу автентифікації не виконується, що свідчить про критичні недоліки у забезпеченні належного захисту облікових записів користувачів. Загальний показник відповідності стандартам безпеки становить лише 22,9% (19,5 балів з можливих 85), що свідчить про серйозні прогалини в реалізації основних вимог до автентифікації.

Найбільш проблемними виявилися базові вимоги до захисту паролів. Система не забезпечує мінімальну довжину паролів у 12 символів та не підтримує паролі довжиною до 64 символів. Відсутня функціональність тимчасового перегляду введеного пароля та механізми запобігання усіченню паролів. Критично низьким є рівень захисту від автоматизованих атак - система не має ефективних засобів протидії brute force атакам та тестуванню викрадених облікових даних.

Водночас, деякі аспекти демонструють помірний рівень відповідності стандартам. Система частково підтримує використання Unicode-символів у паролях, включаючи пробіли та емодзі, та має відносно прийнятний рівень відсутності композиційних обмежень для паролів. Функціональність зміни паролів реалізована на середньому рівні, проте вимога введення поточного та нового пароля виконується краще.

У сфері відновлення облікових записів система показує змішані результати: відсутні небезпечні підказки паролів та "секретні питання", але процедура відновлення частково захищена від розкриття поточного пароля. Позитивним

аспектом є реалізація безпечних сповіщень користувачів про зміни в автентифікаційних даних.

Результати оцінювання вказують на нагальну потребу комплексного перегляду та модернізації системи автентифікації, особливо в частині базових вимог до паролів та захисту від автоматизованих атак, що є фундаментальними елементами кібербезпеки.

Наведений вище алгоритм розрахунку оцінки захищеності, продемонстрований на розділі “Автентифікація” є частиною розробленого методу визначення кількісного рівня безпеки вебдодатку. В таблиці 3.4 наведено узагальнені кількісні показники виконання критеріїв для кожного розділу, оцінка яких здійснювалась за аналогічним алгоритмом.

Таблиця 3.4 – Результати розрахунків оцінки захищеності для кожного з розділів

Розділ	**Розподіл балів за критеріями для кожної розділу**			**Сума набраних балів за розділом**	**Макси-мальна кіль-кість балів за розділом**	**Відносний показник захисту за розділом**
	0 балів	**0,5 бала**	**1 бал**			
Authentication / Автентифікація	60	11	14	19,5	85	0,23 (19,5 / 85)
Session Management / Керування сеансами	46	4	4	6	54	0.11 (6 / 54)
Access Control / Контроль доступу	34	16	3	18,5	43	0.43 (18.5 / 43)

Продовження таблиці 3.4

Validation, Sanitization and Encoding / Валідація, санітизація та кодування	39	17	1	9,5	57	0.17 (9.5 / 57)
Data Protection / Захист даних	34	12	5	11	52	0.21 (11 / 52)
Files and Resources / Файли та ресурси	21	9	2	4,5	30	0.15 (4.5 / 30)
Configuration / Конфігурація	27	11	6	11,5	44	0.26 (11.5 / 44)
Разом	261	80	35	80,5	365	0,22 (80,5 / 365)

Таблиця 3.4 містить результати розрахунків оцінки захищеності для кожного з розділів, де відображено загальну кількість набраних балів та максимально допустиму кількість набраних балів. Оцінку захисту для всього розділу, можна отримати як відносний показник набраних за критеріями балів до їх максимально можливої кількості за формулою для нормування та узагальнення результатів тестування.

На основі цієї таблиці можна зробити оцінку вебдодатку за відібраними розділами та вимогами відповідно. Найнижчі показники виявлено в категоріях " Керування сеансами" (0,11), "Файли та ресурси" (0,15) та " Валідація, санітарна обробка та кодування " (0,17). Це свідчить про серйозні ризики, пов'язані з можливим викраденням сесій, некоректною обробкою даних і завантажених файлів, що може стати джерелом атак типу SQL-ін'єкцій, XSS або зловмисного виконання коду. Категорія "Контроль доступу" показала відносно вищий результат

(0,43), однак навіть тут існує значний простір для вдосконалення. Загалом, отримані результати свідчать про низький рівень впровадження безпекових практик, а загальна оцінка захищеності вебдодатку становить 0.22.

Запропонований підхід до кількісної оцінки відповідності вимогам стандарту OWASP ASVS дозволяє отримати загальне уявлення про рівень реалізації механізмів безпеки у вебдодатку. Водночас зазначене рішення не можна вважати повністю завершеним, оскільки воно не враховує диференційовану важливість окремих розділів стандарту та вимог у їх межах. У процесі побудови узагальненого показника відсутність вагових коефіцієнтів призводить до того, що всі складові оцінюваної системи вважаються рівнозначними, що суперечить практиці забезпечення безпеки, де значення окремих компонентів може суттєво відрізнятися залежно від контексту застосування. Таким чином, для підвищення точності та релевантності оцінки доцільним є впровадження системи вагових коефіцієнтів, що відображатимуть відносну значущість відповідних розділів і вимог стандарту в загальній структурі захисту.

3.1.3 Оцінка безпеки вебдодатку з врахування коефіцієнтів важливості

Розроблений метод кількісної оцінки безпеки вебдодатків передбачає використання коефіцієнтів важливості для вимог та їхніх критеріїв оцінювання.

Розглянемо знову вимогу “*Перевірте, чи для функції зміни пароля потрібен поточний та новий пароль користувача*” та критерії для її оцінювання.

На початковому етапі оцінювання встановлюються коефіцієнти важливості для кожного критерію в межах окремо взятої вимоги. Для забезпечення об'єктивності та стандартизації процесу оцінювання застосовується десятибальна шкала, де значення коефіцієнта важливості варіюється від 0 до 10 балів. При цьому значення 0 відповідає мінімальному рівню пріоритетності критерію, тоді як значення 10 позначає максимальну пріоритетність у контексті досліджуваної вимоги. Результати щодо встановлення коефіцієнтів важливості для критеріїв оцінювання вимоги, представлено у вигляді таблиці 3.5.

Таблиця 3.5 - Коефіцієнти важливості критеріїв в межах вимоги

Критерії	**Коефіцієнт важливості**
Чи вимагається введення поточного пароля при зміні пароля?	10
Чи вимагається введення нового пароля при зміні пароля?	10
Чи є механізм перевірки правильності поточного пароля?	9
Чи є механізм перевірки нового пароля на відповідність вимогам безпеки?	9
Чи є механізм підтвердження зміни пароля через електронну пошту або інший канал зв'язку?	7
Чи використовується багатофакторна автентифікація (MFA) під час зміни пароля?	6
Чи є захист від brute-force атак під час зміни пароля?	8
Чи проводиться регулярний аудит для перевірки процесу зміни пароля?	5
Чи існує механізм для реагування на виявлені порушення у процесі зміни пароля?	6

Після цього потрібно здійснити процедуру нормалізації встановлених коефіцієнтів важливості. Нормалізація виконується з метою приведення коефіцієнтів до стандартизованого вигляду, що забезпечує коректність подальших математичних обчислень та порівняльного аналізу. Процес нормалізації коефіцієнтів важливості здійснюється за допомогою математичної формули (1). Ця формула дозволяє привести коефіцієнти до стандартизованого вигляду, зберігаючи при цьому відносні пропорції між критеріями оцінювання. Результати нормалізації

коефіцієнтів важливості, отримані шляхом застосування зазначеної формули до первинних даних експертного оцінювання важливості, представлено у табл. 3.6.

Таблиця 3.6 - Нормалізовані коефіцієнти важливості критеріїв в межах вимоги

Критерії	**Нормалізований коефіцієнт важливості**
Чи вимагається введення поточного пароля при зміні пароля?	0.14
Чи вимагається введення нового пароля при зміні пароля?	0.14
Чи є механізм перевірки правильності поточного пароля?	0.13
Чи є механізм перевірки нового пароля на відповідність вимогам безпеки?	0.13
Чи є механізм підтвердження зміни пароля через електронну пошту або інший канал зв'язку?	0.1
Чи використовується багатофакторна автентифікація (MFA) під час зміни пароля?	0.09
Чи є захист від brute-force атак під час зміни пароля?	0.11
Чи проводиться регулярний аудит для перевірки процесу зміни пароля?	0.07
Чи існує механізм для реагування на виявлені порушення у процесі зміни пароля?	0.09

Відповідно до методу, викладеного у другому розділі дослідження, наступний етап передбачає кількісну оцінку безпекових параметрів для кожного критерію в рамках аналізованої вимоги. В підрозділі 3.1.2 було продемонстровано приклад формалізованого експертного опитування, в ході якого фахівець з безпеки надає обґрунтовані відповіді на попередньо розроблені критерії оцінювання, специфічні для кожної досліджуваної вимоги.

Інтегральна оцінка виконання вимоги розраховується за формулою (2), яка забезпечує математично обґрунтоване поєднання кількісних значень усіх критеріїв з урахуванням їх важливості. Результати обчислення кількісного значення оцінки вимоги "*Перевірте, чи для функції зміни пароля потрібен поточний та новий пароль користувача*", як зваженої суми оцінок критеріїв наведено в таблиці 3.7.

Таблиця 3.7 - Обчислення кількісних безпекових значень для критеріїв в межах вимоги

Критерії	**Нормалізований коефіцієнт важливості**	**Відповідь експерта**	**Зважене значення критерію**
Чи вимагається введення поточного пароля при зміні пароля?	0.14	1 бал	0.14
Чи вимагається введення нового пароля при зміні пароля?	0.14	0,5 бал	0.07
Чи є механізм перевірки правильності поточного пароля?	0.13	1 бал	0.13
Чи є механізм перевірки нового пароля на відповідність вимогам безпеки?	0.13	0,5 бала	0.07

Продовження таблиці 3.7

Чи є механізм підтвердження зміни пароля через електронну пошту або інший канал зв'язку?	0.1	0 балів	0
Чи використовується багатофакторна автентифікація (MFA) під час зміни пароля?	0.09	0 балів	0
Чи є захист від brute-force атак під час зміни пароля?	0.11	0 балів	0
Чи проводиться регулярний аудит для перевірки процесу зміни пароля?	0.07	0 балів	0
Чи існує механізм для реагування на виявлені порушення у процесі зміни пароля?	0.09	0 балів	0
Оцінка вимоги за критеріями			0,41

Згідно з розробленим методом, максимальна оцінка вимоги дорівнює одиниці. З таблиці 3.7 видно, що кількісна оцінка вимоги “*Перевірте, чи для функції зміни пароля потрібен поточний та новий пароль користувача*” становить 0,41. Цей показник характеризує загальний рівень відповідності досліджуваної вимоги встановленим стандартам безпеки та якості в межах конкретного розділу системи. Отримане значення інтегральної оцінки свідчить про недостатній або нижче середнього рівень виконання вимоги, що потребує подальшого аналізу та розробки заходів щодо покращення показників безпеки системи.

В підрозділі 3.1.1 для розділу "Автентифікація" було здійснено вибір 11 вимог, для кожної з яких тепер теж необхідно встановити коефіцієнти важливості

в межах розділу, враховуючи специфіку та критичність аспектів автентифікації в контексті загальної безпеки системи.

Для кожної вимоги розділу "Автентифікація" проводиться повний цикл обчислень, що включає нормалізацію коефіцієнтів важливості за формулою (3). Комплексні результати експертного оцінювання, включаючи перелік вимог розділу "Автентифікація", встановлені експертом коефіцієнти важливості, їх нормалізовані значення та результати проведених розрахунків оцінювання, представлено у табл. 3.8.

Таблиця 3.8 - Результати розрахунку вимог для розділу “Автентифікація”

Вимога	**Коефіцієнт важливості**	**Нормалізований коефіцієнт важливості**	**Оцінка вимоги як зваженої суми критеріїв**	**Зважене значення вимоги**
Перевірте, чи встановлені користувачем паролі мають довжину щонайменше 12 символів (після об'єднання кількох пробілів)	9	0,09	0,08	0,0072
Переконайтеся, що дозволені паролі довжиною щонайменше 64 символи, а паролі довжиною понад 128 символів заборонені.	7	0,07	0	0

Продовження таблиці 3.8

Переконайтеся, що не виконується скорочення пароля. Однак, кілька послідовних пробілів можна замінити одним пробілом.	8	0,08	0	0
Перевірте, чи дозволено використовувати в паролях будь-які друковані символи Unicode, включаючи нейтральні для мови символи, такі як пробіли та емодзі.	6	0,06	0,43	0,0258
Перевірте, чи можуть користувачі змінювати свій пароль.	9	0,09	0,38	0,0342
Переконайтеся, що для функції зміни пароля потрібні поточний та новий пароль користувача.	10	0,10	0,41	0,041
Перевірте, чи немає правил складання паролів, які обмежують тип дозволених символів. *	7	0,07	0,59	0,0413

Продовження таблиці 3.8

Переконайтеся, що користувач може тимчасово переглянути весь замаскований пароль або тимчасово переглянути останній введений символ пароля на платформах, які не мають цієї вбудованої функції.	5	0,05	0	0
Перевірте, чи ефективні засоби контролю за автоматизацією для зменшення ризиків порушення перевірки облікових даних, атак методом повного перебору та блокування облікових записів.**	10	0,10	0	0
Перевірте, чи безпечні сповіщення надсилаються користувачам після оновлення даних автентифікації, таких як скидання облікових даних, зміна електронної пошти або адреси, вхід з невідомих або ризикованих місць.***	8	0,08	0,5	0,004
Перевірте, чи немає підказок для пароля або автентифікації на основі знань (так званих «секретних питань»).	9	0,09	0	0
Відновлення облікових даних перевірки пароля жодним чином не розкриває поточний пароль.	10	0,10	0,5	0,05
Оцінка розділу “Автентифікація”	0.20			

Застосування формули (4) до експериментальних даних, отриманих для розділу "Автентифікація", дозволило визначити кількісну оцінку захищеності

цього розділу. Розрахункове значення захищеності розділу "Автентифікація" становить 0,20, що характеризує поточний рівень безпеки механізмів автентифікації в досліджуваній системі. Отримане значення свідчить про відносно низький рівень захищеності розділу "Автентифікація", що вказує на необхідність впровадження додаткових заходів безпеки та вдосконалення існуючих механізмів автентифікації для підвищення загального рівня захисту системи.

Розроблений метод оцінювання застосовується послідовно до всіх розділів досліджуваної системи з метою отримання комплексної картини рівня захищеності вебдодатку. Кожен розділ підлягає аналогічному процесу експертного оцінювання, що включає встановлення коефіцієнтів важливості, їх нормалізацію, розрахунок кількісних безпекових значень критеріїв та визначення інтегральної оцінки захищеності розділу.

Систематичне застосування описаної логіки розрахунків забезпечує однорідність та порівнянність результатів оцінювання між різними функціональними розділами системи. Це дозволяє ідентифікувати найбільш проблемні аспекти безпеки та визначити пріоритетні напрями для вдосконалення захисту системи.

Результати розрахунків безпекових значень для всіх розділів системи представлено у таблиці 3.9. Ці дані становлять основу для проведення порівняльного аналізу рівня захищеності різних функціональних компонентів системи та формування комплексної оцінки безпеки вебдодатку в цілому.

Таблиця 3.9 - Безпекові значення розділів

Розділ	**Значення захищеності**
Автентифікація / Автентифікація	0.20
Session Management / Керування сеансами	0.43
Access Control / Контроль доступу	0.35

Продовження Таблиця 3.9

Validation, Sanitization and Encoding / Валідація, санітизація та кодування	0.52
Data Protection / Захист даних	0.28
Files and Resources / Файли та ресурси	0.39
Configuration / Конфігурація	0.47

Завершальним етапом комплексного оцінювання безпеки вебдодатку є визначення інтегрованої оцінки захищеності, яка характеризує загальний рівень безпеки досліджуваного проєкту тестувальником. Цей узагальнений показник отримується шляхом агрегації результатів оцінювання всіх функціональних розділів системи.

Інтегрована оцінка безпеки вебдодатку розраховується як середнє арифметичне значення оцінок захищеності всіх досліджуваних розділів. Математичне обчислення цього показника здійснюється за формулою (5), яка забезпечує рівнозначний внесок кожного розділу до загальної оцінки безпеки системи.

Застосування формули передбачає обчислення загального показника захищеності вебдодатку як частки від ділення суми оцінок безпеки всіх розділів на загальну кількість відібраних розділів. Результат розрахунку за описаним методом в другому розділі становить 0,38, що характеризує інтегровану оцінку безпеки досліджуваного вебдодатку. Отримане значення інтегрованої оцінки свідчить про середньо-низький рівень захищеності вебдодатку, що вказує на необхідність системного підходу до вдосконалення механізмів безпеки та впровадження комплексних заходів захисту для підвищення загального рівня захищеності інтернет-магазину.

3.1.4 Порівняльний аналіз двох підходів

Для комплексного аналізу ефективності розробленого методу доцільно провести порівняння результатів оцінювання безпеки, отриманих за двома підходами: без урахування коефіцієнтів важливості та з їх застосуванням. Такий порівняльний аналіз дозволить визначити вплив системи вагових коефіцієнтів на кінцеві оцінки безпеки функціональних розділів та оцінити практичну значущість запропонованих методичних удосконалень. В таблиці 3.10 наведено результати оцінювання розділів з таблиці 3.1 з врахуванням та без врахування коефіцієнтів важливості.

Таблиця 3.10 - Оцінки безпеки розділів

Розділ	**Значення захищеності без врахуванням коефіцієнтів важливості**	**Значення захищеності з врахуванням коефіцієнтів важливості**
Authentication / Автентифікація	0.23	0.20
Session Management / Керування сеансами	0.11	0.43
Access Control / Контроль доступу	0.43	0.35
Validation, Sanitization and Encoding / Валідація, санітизація та кодування	0.17	0.52
Data Protection / Захист даних	0.21	0.28
Files and Resources / Файли та ресурси	0.15	0.39
Configuration / Конфігурація	0.26	0.47
Загальна оцінка сайту	0.22	0.38

Результати порівняльного аналізу, представлені в таблиці 3.10, знаходять своє підтвердження в дослідженні [86], які проводили аналіз ефективності сканерів вразливостей для вебдодатків електронної комерції. Зокрема, автори класифікували виявлені вразливості за чотирма рівнями критичності: високий (High), середній (Medium), низький (Low) та інформаційний (Informational), що корелює з необхідністю диференційованого підходу до оцінювання різних вимог безпеки системи.

Особливо показовими є результати сканування комерційним інструментом Acunetix, який виявив 22 вразливості різних рівнів критичності, включаючи 2 високого рівня, 1 середнього, 6 низьких та 13 інформаційних. Така різноманітність рівнів критичності підтверджує обґрунтованість застосування вагових коефіцієнтів у розробленій методиці, оскільки демонструє нерівнозначність впливу різних типів вразливостей на загальний рівень безпеки системи.

Результати даного дослідження показують суттєві зміни в оцінках після впровадження системи вагових коефіцієнтів, що демонструє пряму кореляцію з практичними результатами виявлення критичних вразливостей у дослідженні Tryhubets B. Зокрема, розділи "Керування сеансами" та "Валідація, санітизація та кодування", які мали найнижчі початкові оцінки (0,11 та 0,17 відповідно), точно корелюють з категоріями A02:2021-Cryptographic Failures та A03:2021-Injection, де в дослідженні було виявлено найбільшу кількість критичних вразливостей. Саме в категорії A03:2021-Injection автори зафіксували 2 вразливості високого рівня критичності (SQL injection), а в категорії A02:2021-Cryptographic Failures - множинні проблеми з налаштуваннями безпеки cookies, що підтверджує обґрунтованість низьких початкових оцінок у відповідних розділах нашого методу.

Навпаки, розділи "Контроль доступу" та "Автентифікація", які отримали вищі початкові оцінки (0,43 та 0,24 відповідно), відповідають категоріям A01:2021-Broken Access Control та A07:2021-Identification and Authentication Failures, де в практичному дослідженні було виявлено значно менше критичних вразливостей або вони взагалі не були зафіксовані більшістю сканерів.

Автори статті наголошують, що "використання відносних показників є недоцільним через складність об'єкта дослідження та велику кількість вбудованих вразливостей", що прямо підтверджує необхідність диференційованого підходу з урахуванням вагових коефіцієнтів, як це реалізовано в даному дослідженні. Значна зміна загальної оцінки безпеки з 0,22 до 0,38 після впровадження вагових коефіцієнтів демонструє критичну важливість врахування реальної критичності різних категорій вразливостей. Такий підхід дозволяє уникнути рівнозначного трактування всіх аспектів безпеки та забезпечує більш адекватне відображення складної структури загроз, характерних для сучасних вебсайтів електронної комерції.

Таким чином, результати незалежного дослідження Tryhubets B. та співавторів підтверджують ефективність запропонованого підходу. Впровадження системи вагових коефіцієнтів дозволяє ідентифікувати критичні слабкі місця, які потребують першочергової уваги, та оптимізувати розподіл ресурсів для покращення безпеки.

3.2 Використання нечіткої логіки для підвищення точності експертного оцінювання безпекових критеріїв

Традиційні методи агрегації експертних оцінок часто не враховують природну невизначеність людського сприйняття та суб’єктивність оцінок фахівців, що може призводити до некоректних висновків при формуванні коефіцієнтів важливості безпекових параметрів. Впровадження системи нечіткої логіки в процес експертного оцінювання критеріїв безпеки вебдодатків представляє собою підхід до вирішення проблеми суб'єктивності та невизначеності людських суджень у технічній сфері. Застосування математичного апарату системи нечіткої логіки дозволяє формалізувати процес перетворення кількісних оцінок експертних суджень у нечіткі множини через механізми фазифікації, узгодження оцінок різних експертів із застосуванням алгоритмів агрегації та зведення нечіткої узгодженої оцінки в точне значення коефіцієнту шляхом дефазифікації. Такий підхід не лише

підвищує об'єктивність результуючих коефіцієнтів важливості, але й створює математично обґрунтовану основу для узгодження думок множини незалежних експертів, що є критично важливим для забезпечення валідності та надійності системи оцінювання безпеки вебдодатків.

Для оцінювання всіх значень всіх вагових коефіцієнтів, що будуть використовуватись в інформаційній системі за замовчуванням було залучено трьох експертів. Відбір експертів здійснювався за наступними критеріями:

- професійна компетентність у сфері веббезпеки;
- практичний досвід у галузі не менше 5 років;
- відсутність конфлікту інтересів щодо оцінюваних критеріїв та вимог.

Для демонстрації практичного застосування системи нечіткої логіки, яка описана в розділі 2.5, розглянемо детальний приклад розрахунку коефіцієнта важливості одного із критерію в межах однієї вимоги на основі експертних оцінок трьох незалежних експертів галузі інформаційної безпеки. Кожен з експертів проводив незалежне оцінювання важливості критеріїв та вимог без можливості консультацій з іншими членами експертної групи, що забезпечило об'єктивність та достовірність отриманих результатів. За результатами експертного оцінювання було отримано наступні оцінки важливості критерію за 10-бальною шкалою (від 0 до 10):

- експерт 1: $\omega_1 = 7$;
- експерт 2: $\omega_2 = 8$;
- експерт 3: $\omega_3 = 10$.

Згідно теорії нечітких множин, що описана в підрозділі 2.5.1 цієї роботи, першим етапом роботи системи нечіткої логіки є фазифікація – перетворення чітких числових оцінок експертів у нечіткі множини для врахування невизначеності експертних суджень.

Для перетворення кількісної оцінки вагового коефіцієнта критерію $w = 7$ ($0 \leq w \leq 10$) експерта 1 застосовуємо трикутну функцію належності згідно з формулою (14). Отримаємо:

$$\mu(x) = \begin{cases} 1, & x = 7 \\ \dfrac{x - 7 + 2}{2}, & 5 < x < 7 \\ \dfrac{7 + 2 - x}{2}, & 7 < x < 9 \\ 0, & \text{в інших випадках} \end{cases}, \quad x = \overline{1,9}$$

Отже, нечітка множина для експерта 1, що враховує невизначеність експерта при оцінюванні, присвоюючи максимальне значення (μ = 1) обраній оцінці та часткові значення (μ = 0.5) суміжним варіантам:

$\tilde{W}_1$ = {(0,0), (1,0), (2,0), (3,0), (4,0), (5,0), (6,0.5), (7,1), (8,0.5), (9,0), (10,0)}

Аналогічно, для другого експерта з оцінкою $\omega_2 = 8$ застосовується та сама трикутна функція належності за формулою (14). Підставляючи w = 8:

$$\mu(x) = \begin{cases} 1, & x = 8 \\ \dfrac{x - 8 + 2}{2}, & 6 < x < 8 \\ \dfrac{8 + 2 - x}{2}, & 8 < x < 10 \\ 0, & \text{в інших випадках} \end{cases}, \quad x = \overline{1,9}$$

Нечітка множина для другого експерта відповідає теоретичним засадам розділу 2.5.2, де трикутна функція належності моделює невпевненість експерта щодо точного значення коефіцієнта важливості:

$\tilde{W}_2$ = {(0,0), (1,0), (2,0), (3,0), (4,0), (5,0), (6,0), (7,0.5), (8,1), (9,0.5), (10,0)}

Оскільки експерт 3 присвоїв максимальне значення вагового коефіцієнта (w = 10), що відповідає найвищому рівню важливості критерію, застосовуємо спеціальне правило фазифікації для крайніх значень оцінок. Згідно з розробленим методом, описаним у формулі (17), для максимального значення вагового коефіцієнта функція належності визначається наступним чином:

$\tilde{W}_3$ = {(0,0), (1,0), (2,0), (3,0), (4,0), (5,0), (6,0), (7,0), (8,0), (9,0.5), (10,1)}

Така структура нечіткої множини відображає високу впевненість експерта у максимальній важливості критерію (μ(10) = 1), водночас допускаючи деяку

невизначеність щодо попереднього значення ($\mu(9) = 0.5$). Це узгоджується з теоретичними засадами нечіткої логіки, згідно з якими фазифікація дозволяє моделювати невизначеність людського мислення, яке часто оперує неточними поняттями.

На даному етапі всі оцінки експертів виражено нечіткими множинами $\tilde{W}_1$, $\tilde{W}_2$, $\tilde{W}_3$, що дозволило врахувати суб'єктивну невизначеність кожного з експертів при оцінюванні важливості критерію. Як зазначається у в розділі 2.5.2, заміна точних числових оцінок нечіткими множинами забезпечує кілька ключових переваг:

- моделювання невизначеності - дозволяє системі обробляти ту невизначеність, з якою стикаються експерти при виставленні конкретних балів;
- спрощення представлення знань - замість складних математичних моделей використовуються інтуїтивно зрозумілі підходи;
- створення гнучких систем - забезпечує кращу адаптацію до неповних даних та мінливих умов.

Таким чином, процес фазифікації дозволив трансформувати детерміновані оцінки експертів у нечіткі множини, які більш адекватно відображають реальний процес експертного оцінювання з притаманною йому невизначеністю та суб'єктивністю.

Наступним етапом алгоритму, за яким функціонує запропонована система нечіткої логіки, є узгодження думок експертів, що зображено на структурній схемі (див. рис. 2.3). Як зазначено в розділі 2.5.3, для агрегування нечітких оцінок існує кілька підходів, які можна класифікувати за різними ознаками: методи, засновані на операціях нечіткої алгебри; методи, засновані на нечітких інтегралах; методи, засновані на відстані та подібності; методи, засновані на ієрархічному аналізі.

Для узгодження думок експертів було обрано метод середнього арифметичного. Згідно з методом, цей підхід є компромісним рішенням, оскільки забезпечує баланс між консервативним (операція перетину з оператором мінімуму) і оптимістичним (операція об'єднання з оператором максимуму) підходами. Метод

полягає в обчисленні середнього значення функцій належності для кожної можливої оцінки експерта.

Отже, після першого етапу фазифікації, отримали три нечіткі множини оцінок експертів $\tilde{W}_1$, $\tilde{W}_2$, $\tilde{W}_3$. Після цього розраховуємо узгоджену функцію належності для кожного значення x згідно з формулою (20). Результати обчислення узгодженої функції належності представлені у вигляді:

- x = 0…5: μ(0...5) = (0 + 0 + 0) / 3 = 0
- x = 6: μ(6) = (0.5 + 0 + 0) / 3 = 0.167
- x = 7: μ(7) = (1 + 0.5 + 0) / 3 = 0.5
- x = 8: μ(8) = (0.5 + 1 + 0) / 3 = 0.5
- x = 9: μ(9) = (0 + 0.5 + 0.5) / 3 = 0.333
- x = 10: μ(10) = (0 + 0 + 1) / 3 = 0.333

З отриманих значень випливає, що максимальна ступінь належності (μ = 0.5) досягається для значень 7 та 8, що свідчить про концентрацію експертних оцінок у цьому діапазоні та відображає колективну думку щодо пріоритетності відповідного критерію. Узгоджена нечітка множина має вигляд:

$\tilde{W}$ = {(0,0), (1,0), (2,0), (3,0), (4,0), (5,0), (6,0.167), (7,0.5), (8,0.5), (9,0.333), (10,0.333)}

Отримана узгоджена нечітка множина відображає колективну думку групи експертів та враховує різноманітність їхніх індивідуальних оцінок, забезпечуючи компромісне рішення для подальшого етапу дефазифікації.

Згідно з розділом 2.5.4, завершальним етапом роботи системи нечіткої логіки є дефазифікація, що полягає у перетворенні нечіткої множини узгоджених оцінок експертів у точне числове значення вагового коефіцієнта. Як зазначено в методиці, дефазифікація є невід'ємною частиною нечітких систем, оскільки дозволяє перетворити нечіткі рекомендації в конкретні дії, без чого висновки нечіткої системи залишалися б "розмитими" і непридатними для використання в реальному світі.

Для отримання збалансованої оцінки коефіцієнта важливості застосовано метод центру тяжіння. Згідно з формулою (23), для дискретних множин це зважене

середнє всіх можливих вихідних значень. Підставляючи значення узгодженої нечіткої множини:

$$\omega = \frac{0 \cdot 0 + \cdots + 5 \cdot 0 + 6 \cdot 0.167 + 7 \cdot 0.5 + 8 \cdot 0.5 + 9 \cdot 0.333}{0 + 0 + 0 + 0 + 0 + 0 + 0.167 + 0.5 + 0.5 + 0.333 + 0.333} =$$

$$= \frac{0 + 0 + 0 + 0 + 0 + 0 + 1.002 + 3.5 + 4 + 2.997 + 3.33}{1.833} = \frac{14.829}{1.833} = 8.09$$

В результаті застосування системи нечіткої логіки для узгодження експертних оцінок важливості критерію отримано коефіцієнт важливості ω = 8.09. Для використання в системі оцінювання значення коефіцієнта важливості заокруглюється до цілого числа (ω = 8). Цей результат відображає узгоджену думку трьох експертів з врахуванням невизначеності їхніх суджень та демонструє високу важливість даного критерію для забезпечення безпеки вебдодатку.

Запропонований метод забезпечує комплексний підхід до експертного оцінювання через врахування суб'єктивності та невизначеності експертних оцінок шляхом фазифікації, об'єктивне узгодження різних думок експертів методом середнього арифметичного та отримання точного числового значення коефіцієнта важливості для подальшого використання в системі багатокритеріального оцінювання безпеки. Такий метод дозволяє мінімізувати вплив індивідуальних упереджень експертів та забезпечити більш об'єктивну оцінку важливості критеріїв безпеки.

3.3 Висновок до розділу 3

1. Проведено експериментальне дослідження рівня безпеки вебдодатку на прикладі спеціалізованого навчального середовища "OWASP Juice Shop" з попередньо впровадженими вразливостями, що дало змогу підтвердити практичну застосовність розробленого адаптивного методу кількісної оцінки захищеності вебдодатків, яка ґрунтується на стандарті OWASP ASVS.

2. Здійснено первинну оцінку безпеки вебдодатку без врахування коефіцієнтів важливості та виявлено низький загальний рівень захищеності із значними прогалинами у ключових аспектах.

3. Обґрунтовано та емпірично підтверджено важливість впровадження системи вагових коефіцієнтів для критеріїв та вимог безпеки через порівняльний аналіз результатів оцінювання, що дало змогу встановити суттєве підвищення точності та релевантності оцінки захищеності з кореляцією до виявлених критичних вразливостей у незалежних дослідженнях.

4. Запропоновано та детально розглянуто використання апарату нечіткої логіки для підвищення точності та об'єктивності експертного оцінювання вагових коефіцієнтів через механізми фазифікації, узгодження думок експертів та дефазифікації, що дало змогу формалізувати суб'єктивні судження, мінімізувати вплив індивідуальних упереджень та забезпечити математично обґрунтовану основу для кількісного вимірювання безпекових параметрів.

Отже, результати розділу 3 доводять, що розроблений метод забезпечує комплексний, об'єктивний та адаптивний підхід до оцінки безпеки вебдодатків з урахуванням як наявності, так і значущості захисних механізмів, проте широке практичне впровадження потребує автоматизації процесів оцінювання для ефективного застосування в промислових масштабах.

РОЗДІЛ 4. ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ КІЛЬКІСНОГО ОЦІНЮВАННЯ ЯКОСТІ ЗАХИЩЕНОСТІ ВЕБДОДАТКІВ

4.1. Проєктування та реалізація архітектурної і логічної моделі інформаційної системи

Представлений метод у попередніх розділах може бути адаптований до функціоналу, архітектури та рівня критичності даних конкретного проєкту вебдодатку. ASVS широко використовується в галузі інформаційної безпеки для оцінки надійності архітектури, реалізації та конфігурації вебдодатків, проте його практичне застосування стикається з рядом об'єктивних обмежень.

Сучасний стан розвитку вебтехнологій характеризується експоненціальним зростанням складності програмних систем та відповідним підвищенням вимог до їх інформаційної безпеки. Розроблений метод містить 133 категоризовані вимоги, деталізовані через понад 800 специфічних критеріїв перевірки, що охоплюють всі ключові аспекти захищеності вебдодатків. Така детальність та всеохопність, хоча й забезпечує комплексність оцінювання, одночасно створює значні практичні виклики для ефективного застосування в реальних умовах розробки та експлуатації вебдодатків.

Традиційний підхід до застосування методу базується на мануальному аналізі кожного критерію експертом з інформаційної безпеки, що передбачає індивідуальне оцінювання релевантності, встановлення вагових коефіцієнтів та формування інтегральної оцінки захищеності системи. Емпіричні дослідження показують, що повний цикл оцінювання одного вебдодатку вимагає від 30 до 50 годин роботи кваліфікованого спеціаліста, що робить такий підхід економічно неефективним для більшості організацій та критично обмежує можливості широкого впровадження в індустрії програмного забезпечення.

У контексті сучасних вимог до швидкості розробки програмного забезпечення, зокрема методологій DevOps та Continuous Integration/Continuous Deployment (CI/CD), мануальні процеси оцінювання безпеки стають критичним

вузьким місцем, що суперечить принципам гнучкої розробки. Це обумовлює необхідність створення автоматизованих рішень, здатних забезпечити ефективне застосування методу без втрати якості та повноти аналізу.

Алгоритмічні аспекти автоматизації включають розробку математичних моделей для нормалізації вагових коефіцієнтів, агрегації оцінок на різних ієрархічних рівнях та адаптивної селекції релевантних критеріїв на основі характеристик досліджуваного вебдодатку. Застосування методів нечіткої логіки та теорії множин дозволяє формалізувати процес експертного оцінювання та підвищити об'єктивність результатів аналізу.

Технологічна реалізація автоматизованої системи потребує інтеграції сучасних компонентів управління даними, веброзробки та алгоритмічного моделювання. Системи управління базами даних забезпечують структуроване зберігання ієрархічної структури вимог та результатів оцінювання, вебфреймворки надають архітектурну основу для масштабованих рішень, а сучасні фронтенд-технології дозволяють створювати інтуїтивні користувацькі інтерфейси з підтримкою візуалізації складних аналітичних даних.

Основною метою розробленої інформаційної системи є автоматизація процесу кількісної оцінки безпеки вебдодатку з метою підвищення ефективності роботи тестувальника та забезпечення можливості широкого впровадження методу в промислових масштабах. Архітектура інформаційної системи (див. рис. 4.1), складається з двох взаємопов'язаних частин: методу адаптивного оцінювання та алгоритму оцінювання захищеності вебдодатків, які забезпечують комплексний підхід до автоматизації процесу безпекового аудиту.

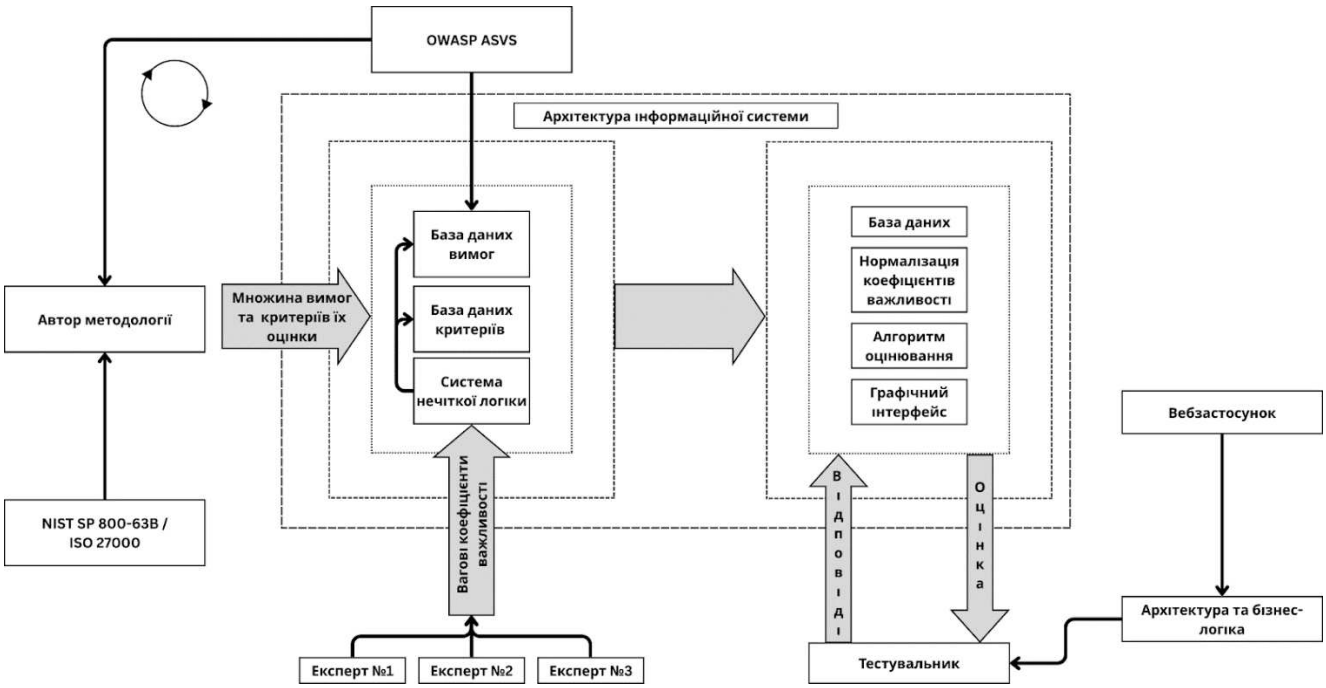

Рис. 4.1. Архітектура інформаційної системи

З рисунку 1 очевидно, що одним з основних етапів розробки інформаційної системи є організація та створення структури даних, здатної забезпечити ефективне зберігання, обробку та використання результатів оцінювання. У цьому контексті ключову роль відіграє база даних, яка забезпечує взаємодію між аналітичною моделлю, користувацьким інтерфейсом та функціональними модулями вебдодатку. Авторами було розроблено реляційну модель бази даних, що відображає структуру даних та операційні процеси користувацької взаємодії. В основу моделі покладено концепцію структурованого, ієрархічного зберігання вимог безпеки, а також механізми фіксації результатів оцінювання та адаптації вибірки вимог відповідно до функціональних характеристик об'єкта дослідження.

Логічна структура бази даних формує три головні функціональні підсистеми: управління користувачами та їхніми проєктами, зберігання нормативної бази вимог безпеки, а також фіксація результатів оцінювання на різних рівнях деталізації. На рівні користувача передбачено можливість створення одного або кількох проєктів, кожен з яких є окремим випадком оцінювання певного вебдодатку. Для збереження відповідних зв'язків та ідентифікації використовуються таблиці користувачів і

проєктів, що уможливлюють персоналізовану роботу з системою та ізольоване зберігання даних.

Система базується на структурі стандарту OWASP ASVS. У базі даних це представлено у вигляді трирівневої ієрархії, де розділи (sections) об’єднують семантично споріднені вимоги, кожна з яких деталізується через низку критеріїв перевірки. Всі елементи – як вимоги, так і критерії – мають атрибути базової та нормалізованої ваги, що дозволяє застосовувати формальний апарат зваженого підрахунку при оцінюванні відповідності. Це особливо важливо у контексті адаптивної оцінки, де не всі елементи методу однаково релевантні до кожного з оцінюваних проєктів.

Для забезпечення адаптивності системи впроваджено механізм попереднього опитування, результати якого зберігаються у таблицях запитань та відповідей. Відповіді користувача автоматично визначають множину релевантних до проєкту розділів, вимог і критеріїв. Це дозволяє динамічно підлаштовувати методичну вибірку до архітектурних, функціональних і бізнесових характеристик конкретного вебдодатку. Таке рішення забезпечує індивідуалізований підхід до кожного кейсу, що, у свою чергу, підвищує точність та об’єктивність оцінки.

У межах виконання оцінювання всі результати фіксуються на чотирьох рівнях: критерії, вимоги, розділи та проєкт в цілому. На кожному рівні виконується агрегація результатів нижчого рівня з урахуванням вагових коефіцієнтів. Окрему увагу в моделі приділено забезпеченню логічної цілісності даних. У схемі, яка зображена на рисунку 4.2, усі таблиці взаємопов’язані через зовнішні ключі, що дозволяє підтримувати узгодженість записів при виконанні операцій модифікації. Унікальні індекси у таблицях результатів запобігають дублюванню оцінок для одного і того ж критерію чи вимоги в межах конкретного проєкту, а каскадні обмеження забезпечують правильне видалення пов’язаних сутностей.

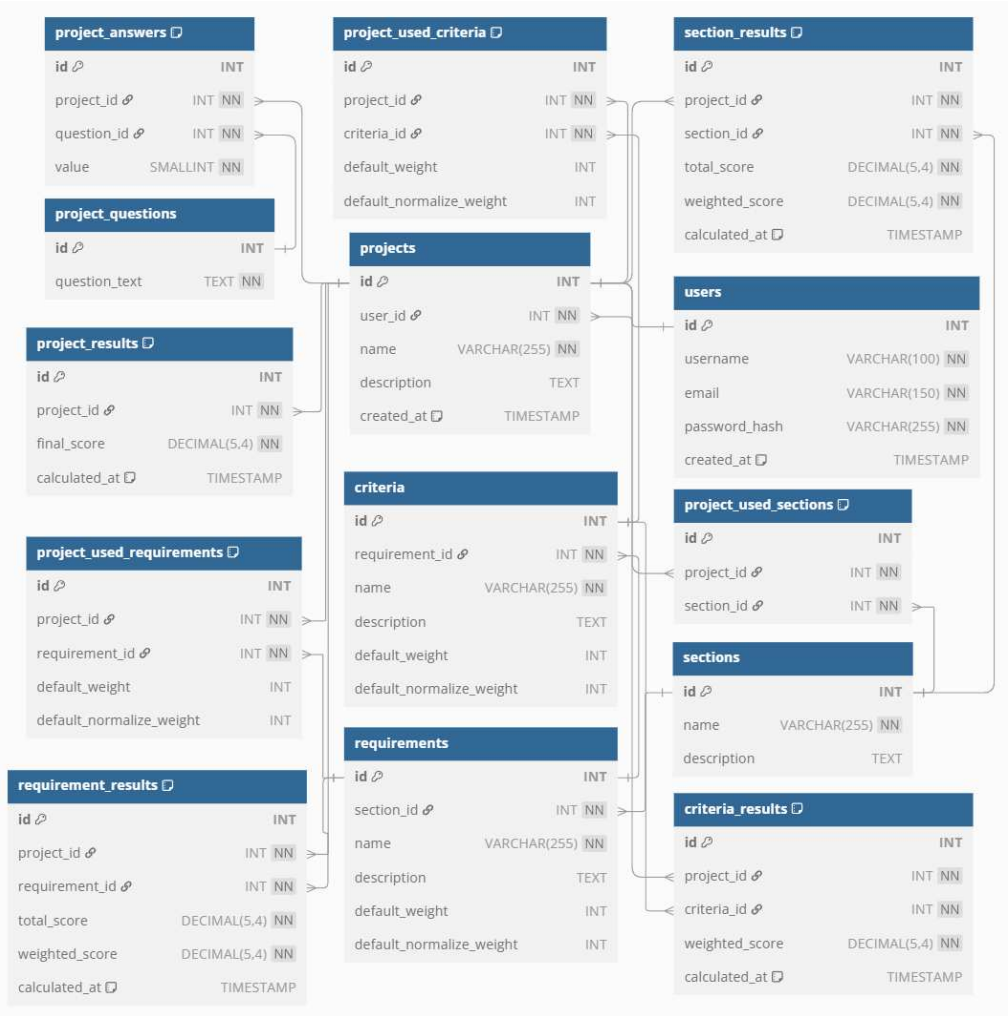

Рис. 4.2. Схема бази даних

Розроблена база даних не лише реалізує методичну модель оцінювання, але й створює технічну основу для реалізації складних сценаріїв користувацької взаємодії. Модульна структура та можливість формування контекстно залежної вибірки вимог забезпечують ефективну підтримку повного життєвого циклу оцінювання захищеності вебдодатку.

Ефективне використання розробленої бази даних забезпечується через комплекс сценаріїв користувацької взаємодії з системою. Однією з ключових складових розробленої інформаційної системи є реалізація сценаріїв взаємодії користувача з програмним забезпеченням у рамках повного життєвого циклу оцінювання захищеності вебдодатку. Взаємодія між користувачем, вебінтерфейсом, службами бізнес-логіки та обчислювальними модулями

реалізована відповідно до принципів сервісно-орієнтованої архітектури (SOA), з використанням шаблонів, характерних для Laravel-фреймворку. Відповідну послідовність дій представлено у вигляді діаграми типу sequence diagram (див. рис. 4.3), яка формалізує обмін повідомленнями між актором (користувачем) і внутрішніми компонентами системи.

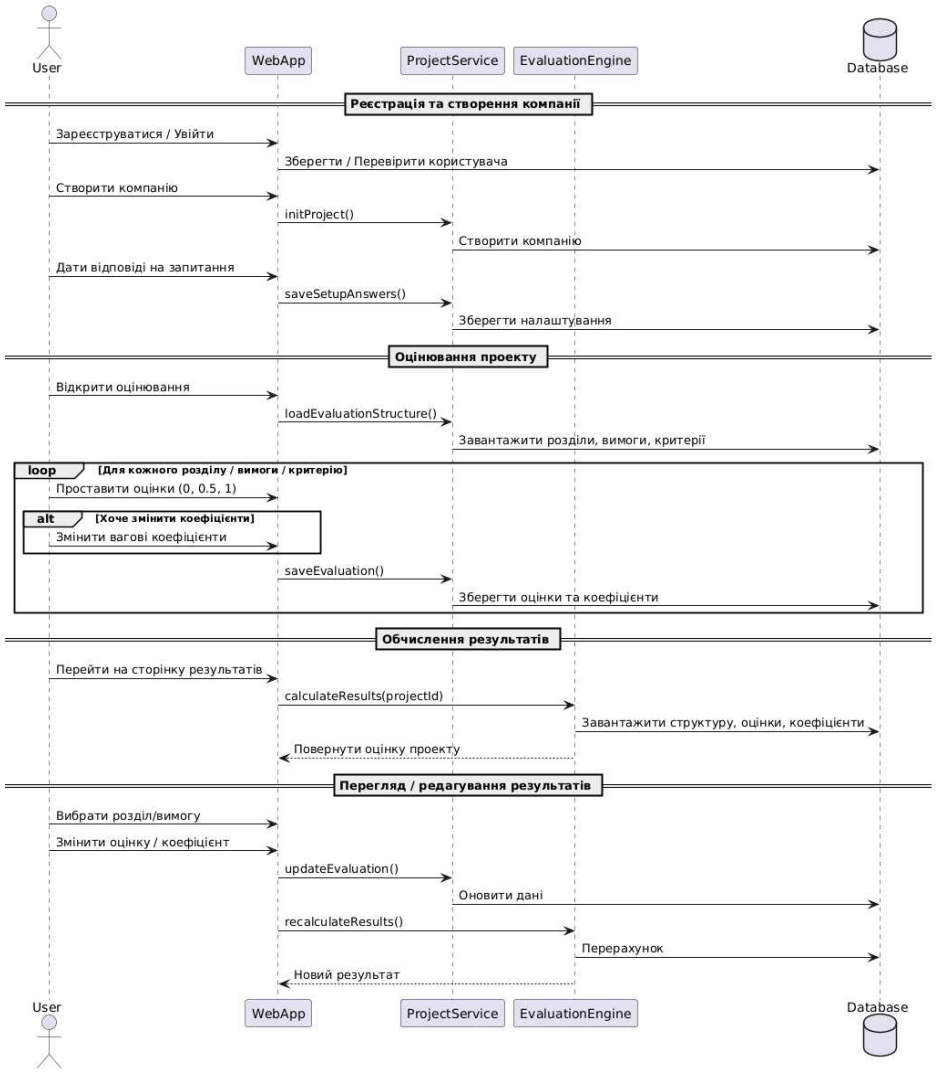

Рис. 4.3 - Послідовність дій представлено у вигляді діаграми

На початковому етапі користувач взаємодіє з інтерфейсом системи з метою автентифікації, що реалізується у вигляді операцій реєстрації або входу до облікового запису. Для того необхідно скористатись відповідними формами реєстрації та авторизації (див. рис. 4.4).

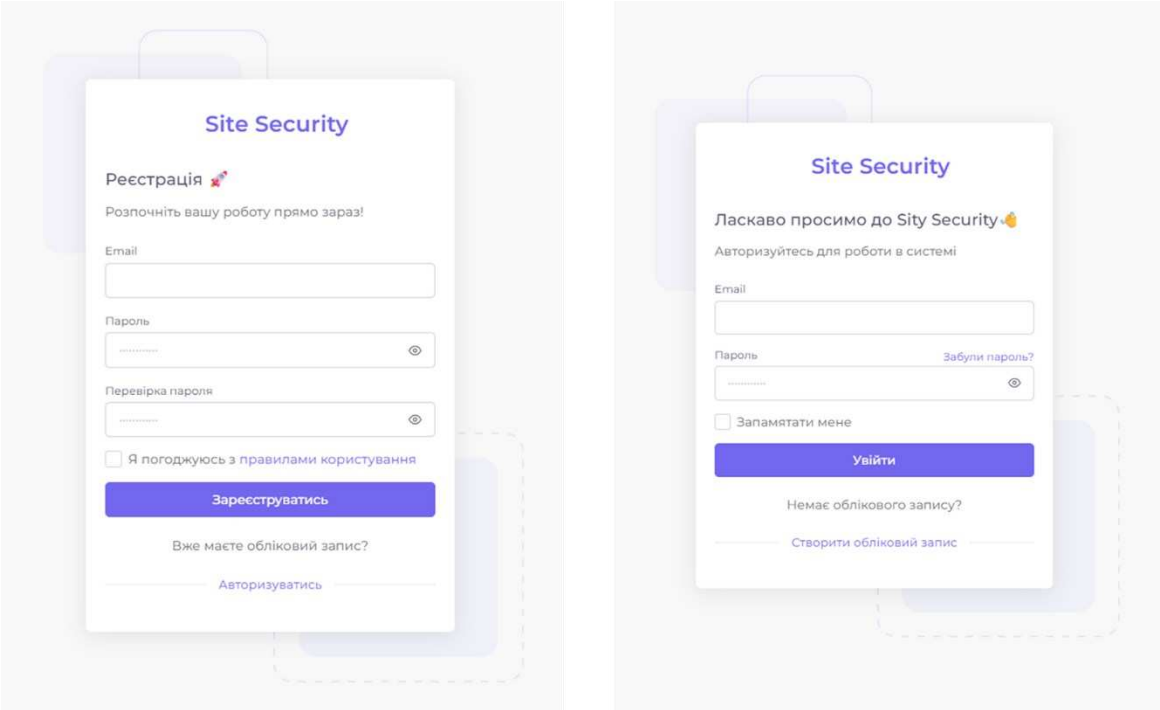

Рис. 4.4. Форми реєстрації та авторизації

Якщо ж користувачеві потрібно відновити пароль - потрібно скористатись відповідними формами (див. рис. 4.5).

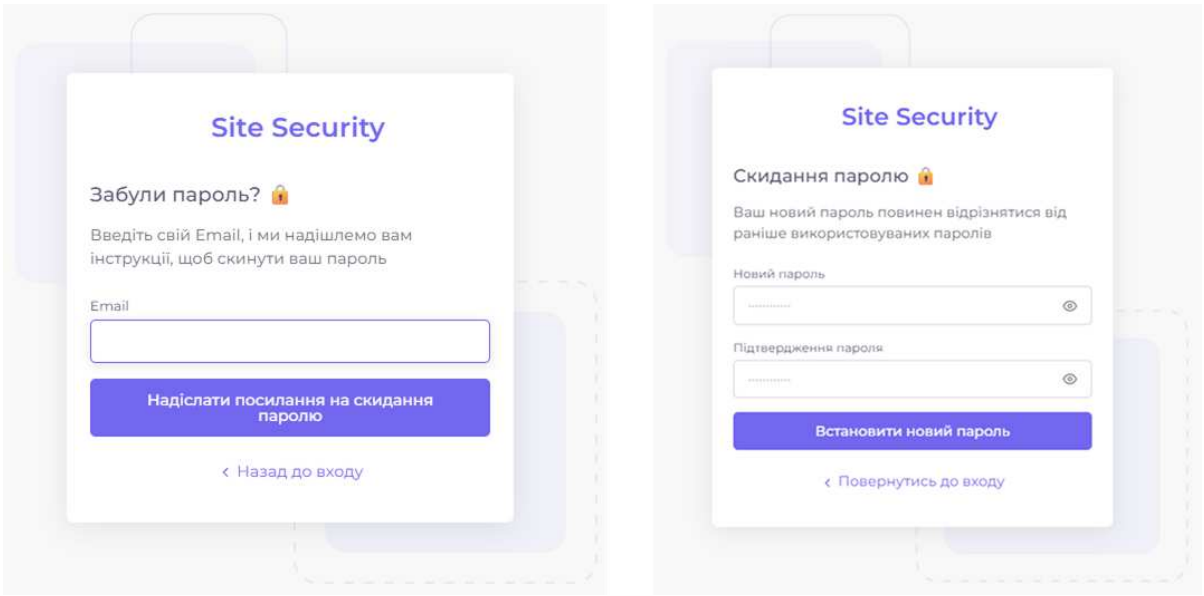

Рис 4.5. Форми для відновлення та скидання паролю

Вебінтерфейс системи оцінювання безпеки здійснює автоматизовані запити до бази даних з метою збереження та верифікації облікових даних користувача.

Після успішного завершення процедури ідентифікації користувач отримує доступ до функціоналу створення нового проєкту оцінювання. У контексті даної системи термін "проєкт" визначається як формалізований об'єкт дослідження, що представляє вебдодаток, рівень інформаційної безпеки якого підлягає комплексному аналізу. Процес ініціалізації проєкту реалізується через виклик сервісної операції initProject() в архітектурному модулі ProjectService, що забезпечує створення відповідного запису в реляційній базі даних системи.

На етапі створення проєкту користувачеві надається спеціалізована форма (див. рис. 4.6), яка містить структурований перелік параметрів для характеристики досліджуваного вебдодатку.

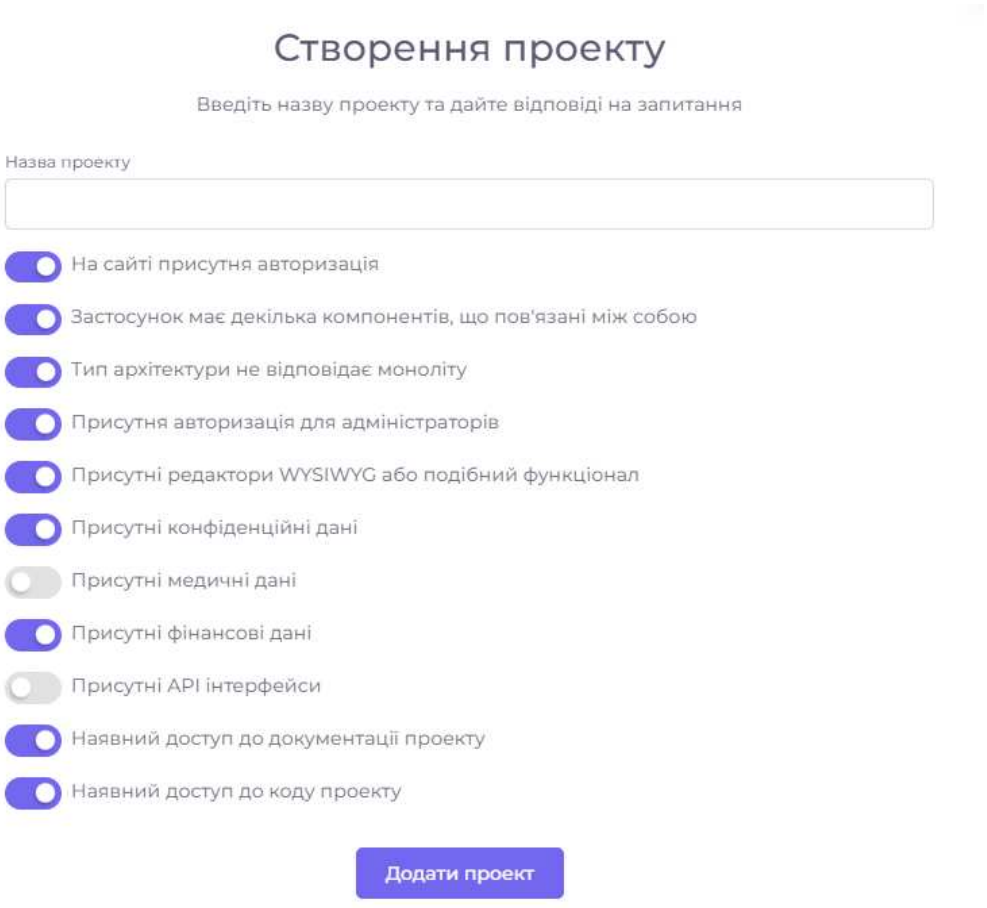

Рис. 4.6. Форма створення проєкту

Ця форма включає питання, що охоплюють архітектурні характеристики як структурні особливості побудови системи, функціональні характеристики як

спектр реалізованих можливостей, та контекстні характеристики як умови експлуатації та середовище функціонування. Призначення даного опитувальника полягає в ідентифікації релевантних вимог безпеки, специфічних для конкретного типу вебдодатку, що дозволяє адаптувати процес оцінювання відповідно до індивідуальних особливостей об'єкта дослідження.

Питання наявності механізмів авторизації є фундаментальним у процесі оцінювання, оскільки значна частина вимог стандарту OWASP Application Security Verification Standard стосується процедур автентифікації та управління сесіями. У випадку вебдодатків, призначених виключно для статичного відображення інформації без необхідності ідентифікації користувачів, відповідні вимоги виключаються з переліку критеріїв оцінювання. Водночас значна кількість сучасних вебдодатків реалізує диференційовані механізми авторизації для різних категорій користувачів. Типовим прикладом є системи, де звичайні користувачі мають доступ до публічного контенту без автентифікації, проте адміністративний персонал повинен проходити процедуру ідентифікації для доступу до функцій управління контентом або модерації.

Важливим фактором для визначення критеріїв оцінювання є архітектурна складність системи. Вебдодатки, що включають декілька функціональних модулів у межах єдиного програмного середовища, навіть за умови відсутності мікросервісної архітектури, потребують дотримання специфічних вимог безпеки, спрямованих на забезпечення цілісності та безпечної міжкомпонентної взаємодії.

Системи, побудовані на принципах сервісно-орієнтованої архітектури або мікросервісного підходу, вимагають адаптації критеріїв оцінювання відповідно до особливостей обраної архітектурної парадигми. Така адаптація включає врахування специфічних векторів атак, притаманних розподіленим системам, та відповідних механізмів захисту.

Сучасні системи керування контентом часто інтегрують WYSIWYG-редактори або аналогічний функціонал, що забезпечує візуальне редагування текстових матеріалів та вебсторінок без необхідності технічних знань від користувача. Наявність такого функціоналу обумовлює необхідність застосування

посилених вимог безпеки щодо обробки та виведення користувацьких даних, зокрема з урахуванням ризиків Cross-Site Scripting атак.

Використання Application Programming Interface значно розширює потенційну поверхню атаки вебдодатку, відповідно вимоги до безпеки API-інтерфейсів набувають критичного значення в процесі комплексного оцінювання безпеки системи.

Обробка конфіденційних даних потребує окремого аналізу та застосування спеціалізованих вимог безпеки. Згідно з класифікацією OWASP ASVS, медичні та фінансові дані належать до особливих категорій інформації, що вимагають дотримання суворих стандартів щодо процедур зберігання, передачі та контролю доступу.

Наявність доступу експерта з оцінювання безпеки до технічної та архітектурної документації проєкту є критичною передумовою для проведення якісного аналізу. Відсутність документації унеможливлює верифікацію низки вимог безпеки, зокрема тих, що стосуються процесів розробки та життєвого циклу програмного забезпечення, результатів моделювання загроз, механізмів журналювання подій безпеки, системи контролю доступу та розподілу привілеїв.

Верифікація значної частини вимог безпеки можлива виключно за умови наявності доступу до вихідного коду проєкту. Це стосується механізмів валідації вхідних даних, реалізації процедур авторизації та контролю доступу, алгоритмів обробки користувацького введення, криптографічних механізмів та їх імплементації, систем логування та аудиту подій безпеки.

Отримані відповіді передаються до модуля ProjectService, де виконується їх обробка та збереження. Цей етап є критично важливим, оскільки саме на основі отриманих даних система здійснює адаптивну побудову вибірки безпекових вимог, релевантних до конкретного кейсу оцінювання. Далі проєкт буде відображатись на власній панелі проєктів. На рисунку 4.7 можна побачити в якому статусі знаходиться проєкт, скільки він має вимог та критеріїв для оцінки.

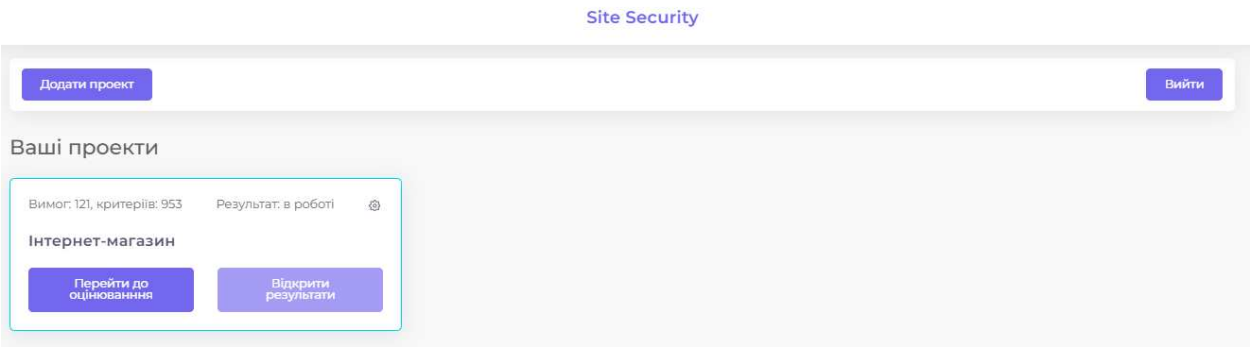

Рис. 4.7. Створений проєкт для оцінювання

Ініціація процесу оцінювання здійснюється через активацію відповідного програмного інтерфейсу, що призводить до автоматизованого завантаження методичної структури оцінювання (див. рис. 4.8). Дана структура включає релевантні тематичні розділи, конкретні вимоги безпеки та розроблені критерії оцінювання, що базуються на стандарті OWASP Application Security Verification Standard. Формування відповідної структури реалізується архітектурним сервісом ProjectService через звернення до реляційної бази даних системи, де зберігаються вимоги, розподілені за тематичними розділами. Варто зазначити, що певні розділи, зокрема "API and Web Service", можуть не містити активних вимог для конкретного проєкту, що обумовлено результатами попереднього етапу ідентифікації релевантних критеріїв оцінювання. У таких випадках відповідні розділи виключаються з розрахунку загальної оцінки проєкту, забезпечуючи адаптивність методу оцінювання до специфічних характеристик досліджуваного вебдодатку.

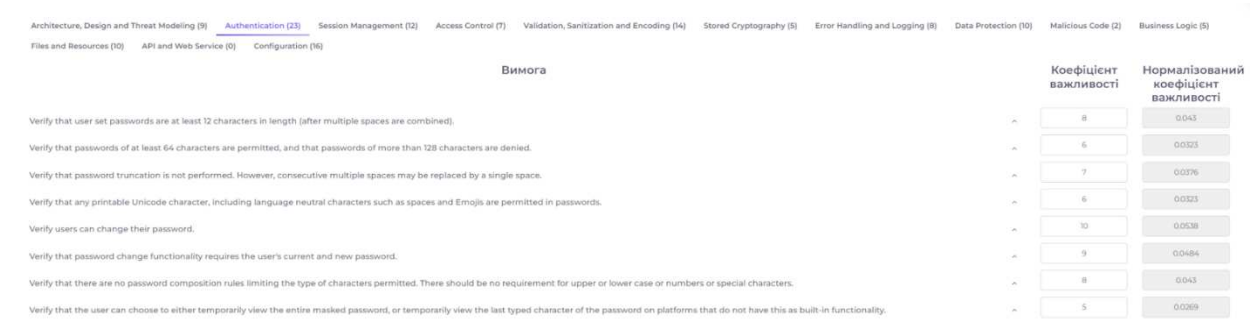

Рис. 4.8. Загальне вікно для оцінювання проєкту

Процес оцінювання реалізується через поетапне присвоєння балів за кожним критерієм у межах відповідних вимог і тематичних розділів. Кожна вимога супроводжується полем для введення коефіцієнта важливості, що дозволяє експерту здійснювати кількісне ранжування критеріїв відповідно до їх значущості для конкретного об'єкта дослідження.

Реалізація методу передбачає необхідність визначення значної кількості коефіцієнтів важливості для всіх критеріїв та вимог, що в загальному випадку, становить понад 800 індивідуальних параметрів оцінювання.

Традиційний підхід до експертного оцінювання, при якому кожен експерт повинен проставити коефіцієнти важливості для всіх критеріїв та вимог індивідуально для кожного оцінюваного проєкту, характеризується надмірною ресурсоємністю та часовими витратами. Розрахунки показують, що за умови витрати 2-3 хвилин на оцінку одного критерію, загальний час роботи одного експерта над повним набором критеріїв складатиме від 26 до 40 годин, що робить такий підхід практично неприйнятним для широкого впровадження методу.

З метою оптимізації процесу експертного оцінювання та забезпечення практичної застосовності розробленого методу було прийнято рішення про створення уніфікованої бази коефіцієнтів важливості на основі консолідованих експертних суджень. Водночас, система збереження коефіцієнтів важливості передбачає можливість їх гнучкого налаштування відповідно до специфіки конкретного проєкту або вимог замовника. Тестувальник, який проводитиме оцінювання безпеки вебдодатку, має можливість переглянути базові коефіцієнти важливості та внести корективи лише для тих вимог та критеріїв, які потребують адаптації під особливості оцінюваної системи. Такий підхід забезпечує оптимальний баланс між стандартизацією процесу оцінювання та необхідною гнучкістю для врахування індивідуальних характеристик різних вебпроєктів.

Згідно розділу 3, система забезпечує попередньо визначені коефіцієнти важливості, трьома незалежними експертами, із врахуванням логіки нечітких множин, проте експерт, який проводить оцінку, має можливість модифікувати ці значення відповідно до специфічних вимог проєкту або власної професійної

оцінки. Механізм коефіцієнтів включає можливість повного виключення окремих вимог з розрахунку шляхом присвоєння нульового значення коефіцієнта важливості, що забезпечує гнучкість методу оцінювання.

Система автоматично обчислює нормалізований коефіцієнт важливості, що відображається в недоступному для редагування полі та оновлюється в реальному часі при зміні експертом базового коефіцієнта. Деталізація структури оцінювання реалізується через інтерактивний доступ до критеріїв кожної вимоги (див. рис. 4.9), які також оснащені індивідуальними коефіцієнтами важливості. Логіка обчислення коефіцієнтів на рівні критеріїв відповідає алгоритмам, застосованим для вимог, забезпечуючи узгодженість методичного підходу на всіх рівнях ієрархії оцінювання.

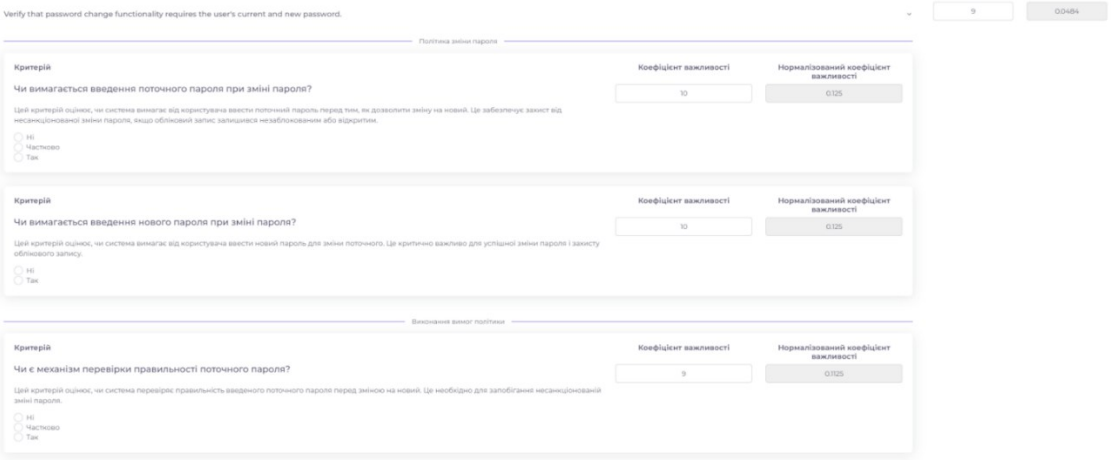

Рис. 4.9. Критерії оцінювання для вимоги

Процес оцінювання базується на триступеневій шкалі з дискретними значеннями 0, 0.5 та 1, що для користувача відповідають "Ні", "Частково" та "Так" відповідно. Дана градація забезпечує достатню деталізацію для точного відображення ступеня відповідності досліджуваного вебдодатку встановленим критеріям безпеки, водночас зберігаючи простоту використання для експерта-оцінювача. Система візуалізації передбачає колірне кодування обраних критеріїв (див. рис. 4.10) сприяє інтуїтивному сприйняттю результатів оцінювання та полегшує навігацію в межах великих масивів критеріїв.

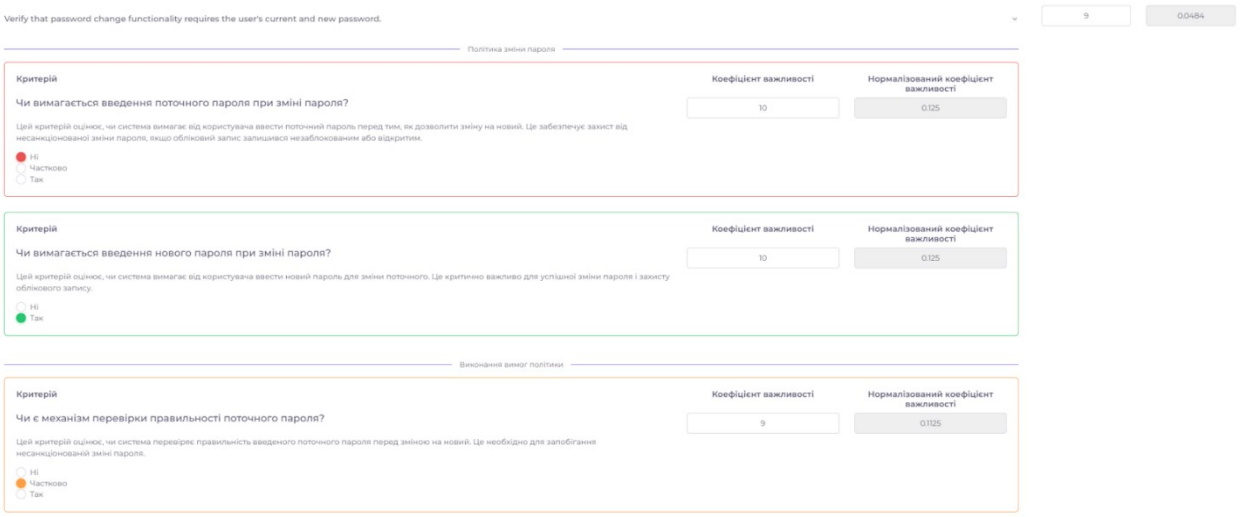

Рис. 4.10. Інтерактивність під час вибору відповідей для критеріїв

Кожен варіант відповіді супроводжується контекстним поясненням, що активується при наведенні курсора, забезпечуючи експерту додаткову інформацію для прийняття обґрунтованих рішень щодо оцінювання. Ця функціональність особливо важлива для забезпечення консистентності оцінювання між різними експертами та мінімізації суб'єктивності інтерпретації критеріїв. Усі введені оцінки та значення коефіцієнтів важливості передаються через програмний інтерфейс до архітектурного сервісу ProjectService, де здійснюється їх структуроване збереження у відповідних таблицях реляційної бази даних, забезпечуючи персистентність даних та можливість подальшого аналізу результатів оцінювання. Завершення процедури введення оцінок ініціює автоматизований процес обчислення результатів оцінювання через передачу унікального ідентифікатора проєкту до спеціалізованого обчислювального модуля EvaluationEngine. Зазначений модуль здійснює повне завантаження структури оцінювання з реляційної бази даних, що включає введені експертні оцінки, вагові коефіцієнти важливості, а також ієрархічні структурні зв'язки між критеріями, вимогами та тематичними розділами.

Обчислення інтегральної оцінки проєкту реалізується через застосування алгоритмів зваженого агрегування, що передбачають попередню нормалізацію

вагових коефіцієнтів для забезпечення коректного математичного представлення відносної важливості окремих компонентів оцінювання. Результат обчислення передається до вебінтерфейсу та візуалізується для користувача, забезпечуючи негайний доступ до отриманих показників (див. рис. 4.11). Система управління проєктами автоматично оновлює відображення у списку проєктів, присвоюючи кожному проєкту відповідне кольорове кодування згідно з досягнутим рівнем оцінки безпеки. Додатково активується функціонал детального перегляду результатів, що надає можливість поглибленого аналізу отриманих показників за окремими категоріями та критеріями оцінювання.

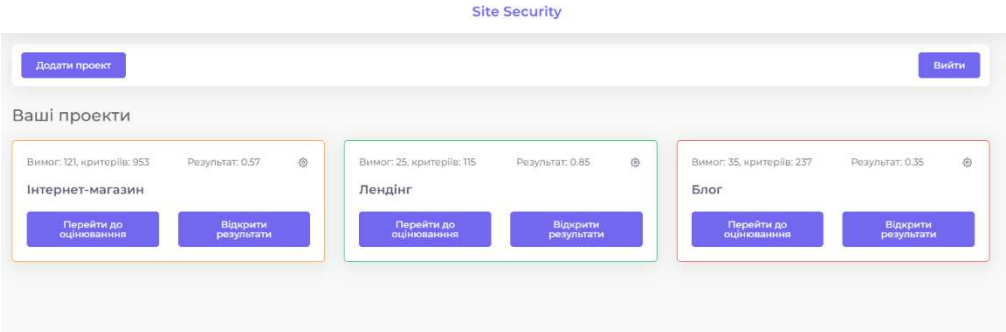

Рис. 4.11. Список оцінених проєктів

Окрім первинного обчислення, система підтримує можливість перегляду та редагування результатів оцінювання. На рисунку 4.12 зображено графічне представлення інформації про результати оцінювання для кожного розділу.

Рис. 4.12. Графічні результати оцінювання проекту для кожного розділу

Нижче, користувач має змогу обрати будь-який розділ або вимогу, змінити відповідну оцінку чи ваговий коефіцієнт. Відповідно до кольорової схеми - розділи та вимоги підсвічені кольорами для легшої навігації проєкту. Критерії підсвічені залежно від відповіді: червоний - “Ні”, помаранчевий - “Частково” та зелений - “Так” відповідно (див. рис. 4.13).

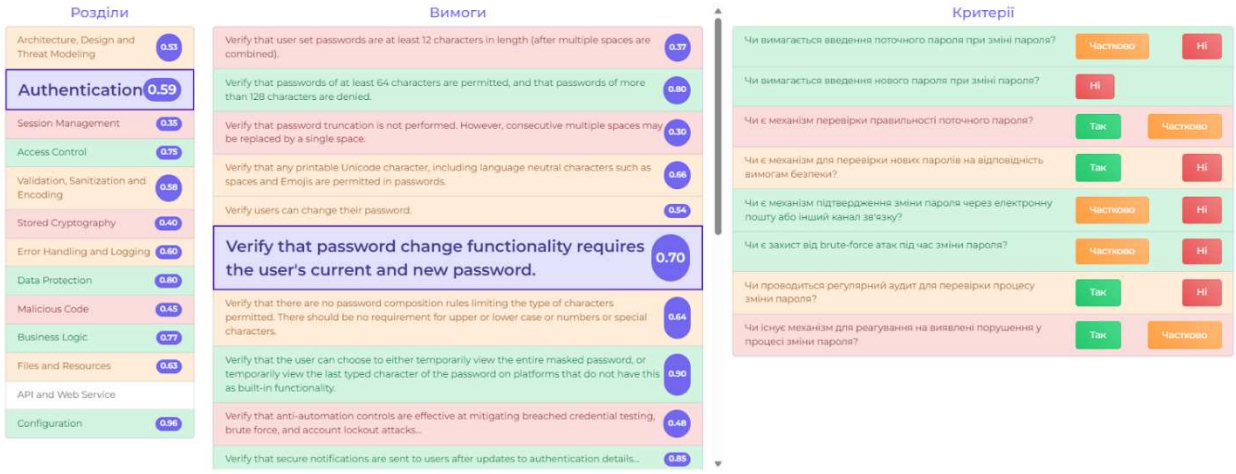

Рис. 4.13. Детальний звіт проєкту

Також для експерта доступна можливість змінити відповіді для критеріїв, що динамічно перебудує графік та змінить забарвлення для розділів, вимог і критеріїв. У відповідь на такі дії вебінтерфейс ініціює операцію оновлення через сервіс ProjectService, яка призводить до внесення змін у відповідні записи бази даних. Одразу після цього повторно викликається модуль EvaluationEngine, що здійснює перерахунок оцінок на основі оновлених даних. Новий результат передається до вебінтерфейсу та оперативно відображається користувачеві. Така послідовність дій забезпечує гнучкість і динамічність процесу оцінювання, дозволяючи вносити зміни без необхідності повторного проходження всіх попередніх етапів.

У рамках розробки інформаційної системи було створено формальну об'єктно-орієнтовану модель, що відображає її логічну та програмну архітектуру. Відповідна структура програмного забезпечення що представлена у вигляді UML-діаграми класів (див. рис. 4.14), деталізує базові сутності предметної області та компоненти функціональної взаємодії користувача із системою.

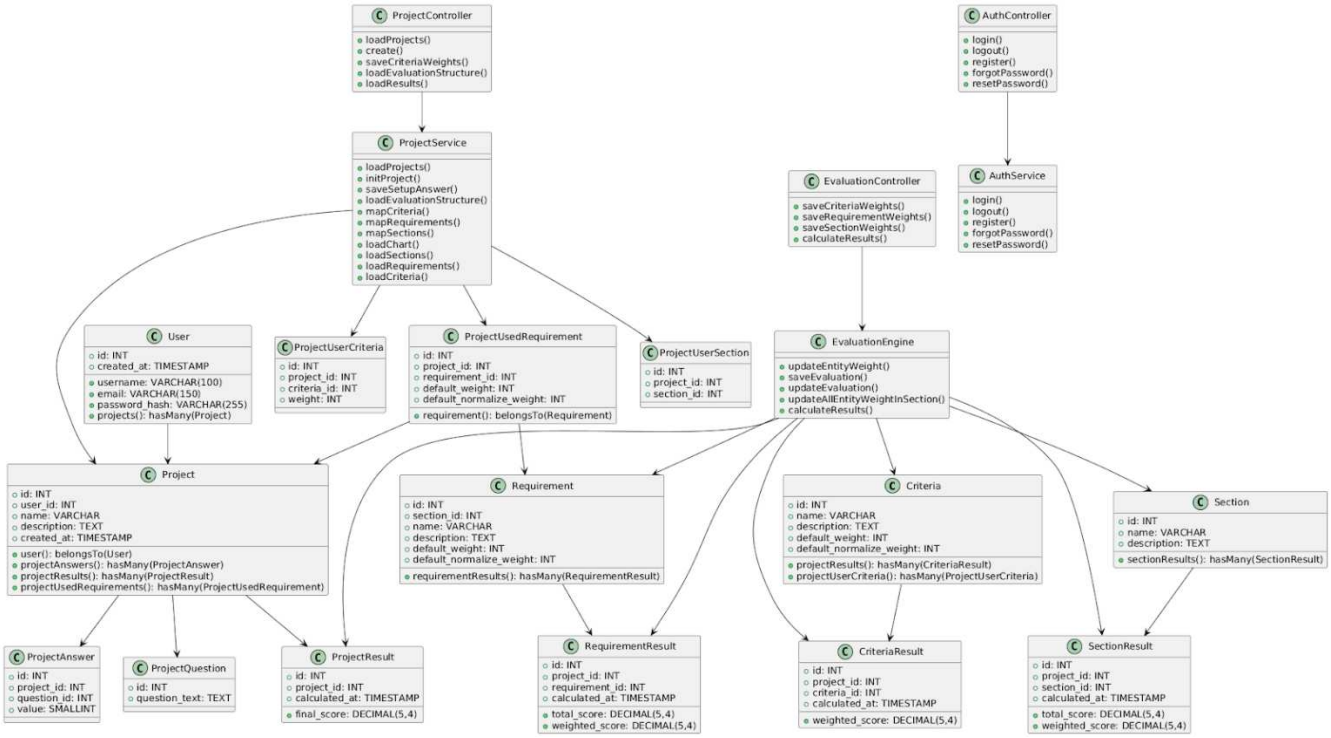

Рис. 4.14. UML-діаграма класів

Розроблена модель базується на архітектурній парадигмі фреймворку Laravel з дотриманням фундаментальних принципів об'єктно-орієнтованого проєктування, включаючи модульність, інкапсуляцію та розділення відповідальностей. Концептуальну основу системи формують доменні сутності User, Project, Section, Requirement, Criteria, що репрезентують процес оцінювання безпеки. Клас User представляє зареєстрованого користувача, здатного ініціювати створення множини проєктів оцінювання. Кожен проєкт асоціюється з результатами попереднього опитування через клас ProjectAnswer та містить повну методичну структуру, що включає ієрархічно організовані розділи, вимоги, критерії та відповідні оцінки на кожному структурному рівні.

Результати оцінювання персистентно зберігаються у спеціалізованих класах CriteriaResult, RequirementResult, SectionResult та ProjectResult, забезпечуючи повну трасованість процесу оцінювання. Сутності Section, Requirement і Criteria реалізують ієрархічну структуру вимог безпеки відповідно до стандарту OWASP Application Security Verification Standard. Розділи об'єднують тематично пов'язані

вимоги, які деталізуються через конкретні критерії оцінювання. Кожна сутність містить атрибути вагових коефіцієнтів default_weight та default_normalize_weight, що використовуються в моделі оцінювання.

Адаптація базової структури до специфічних вимог конкретного проєкту реалізується через допоміжні класи ProjectUsedRequirement, ProjectUserCriteria та ProjectUserSection, які зберігають проєктно-специфічні параметри вагових коефіцієнтів та забезпечують механізм гнучкого налаштування методу оцінювання. Компоненти взаємодії користувача із системою реалізовані через контролери AuthController, ProjectController та EvaluationController, що відповідають за маршрутизацію HTTP-запитів, координацію користувацьких дій та делегування бізнес-логіки відповідним сервісним компонентам.

AuthController забезпечує процеси автентифікації, включаючи реєстрацію, автентифікацію та відновлення доступу. ProjectController реалізує створення проєктів, завантаження структури оцінювання, збереження вагових коефіцієнтів та візуалізацію результатів. EvaluationController відповідає за персистентне збереження оцінок, управління ваговими коефіцієнтами на всіх ієрархічних рівнях та ініціацію обчислювального процесу.

Бізнес-логіка системи концентрується в сервісних класах AuthService, ProjectService та EvaluationEngine. Клас ProjectService виконує функції формування структури оцінювання, збереження відповідей опитування, побудови аналітичних візуалізацій та координації взаємодії з відповідними моделями даних. Аналітичним ядром системи є EvaluationEngine, що забезпечує збереження експертних оцінок, оновлення вагових коефіцієнтів, нормалізацію вагових значень та обчислення інтегральних показників безпеки. Цей модуль оперує з результуючими класами всіх ієрархічних рівнів, реалізуючи повний цикл обчислень на основі введених даних та конфігураційної структури проєкту.

Міжоб'єктні зв'язки реалізовані через механізми ORM-фреймворку Laravel Eloquent, використовуючи стандартні відношення hasMany(), belongsTo() та hasOne() для ефективної навігації між пов'язаними сутностями. Такий підхід спрощує формування запитів до бази даних, мінімізує дублювання коду та

підвищує узгодженість між логікою застосунку та структурою даних, одночасно знижуючи складність супроводу та розширення системи. Побудована модель класів не лише формально репрезентує структуру методичної моделі оцінювання, а й забезпечує її безпосередню програмну реалізацію, дозволяючи масштабування функціональних можливостей, повторне використання компонентів та відповідність сучасним архітектурним вимогам до програмних засобів у галузі інформаційної безпеки.

4.2. Вибрані технології для реалізації безпечного вебдодатку

Реалізація системи кількісного оцінювання якості захищеності вебдодатків базується на технологічному стеку, який забезпечує відповідність сучасним вимогам інформаційної безпеки та стандартам OWASP Application Security Verification Standard. Архітектура системи розроблена з урахуванням принципів безпечної розробки програмного забезпечення, масштабованості та надійності функціонування в умовах високих навантажень.

Серверна частина системи реалізована на основі фреймворку Laravel мови програмування PHP, що забезпечує дотримання архітектурного патерну Model-View-Controller для логічного розділення компонентів системи та полегшення процедур аудиту безпеки. Laravel включає комплекс інтегрованих механізмів захисту від основних типів вебатак, зокрема автоматичний захист від Cross-Site Request Forgery атак через генерацію та валідацію токенів, запобігання SQL-ін'єкціям завдяки використанню підготовлених запитів в Object-Relational Mapping Eloquent, та попередження Cross-Site Scripting атак через автоматичне екранування вихідних даних. Система автентифікації фреймворку реалізує криптографічно безпечне хешування паролів за алгоритмом bcrypt з автоматичною генерацією криптографічної солі, що відповідає рекомендаціям OWASP щодо захисту облікових даних користувачів.

Клієнтська частина системи побудована на основі стандартів HTML5 та CSS3 з використанням фреймворку Bootstrap 5 для забезпечення адаптивності

користувацького інтерфейсу до різних типів пристроїв та роздільних здатностей екранів. Bootstrap 5 включає механізми валідації форм на клієнтському рівні, що формує первинний рубіж захисту від некоректних або шкідливих вхідних даних. Динамічна функціональність інтерфейсу реалізована засобами JavaScript у поєднанні з бібліотекою jQuery для оптимізації асинхронної взаємодії з серверною частиною через AJAX-запити. Всі асинхронні операції включають обов'язкову валідацію CSRF-токенів та додаткову верифікацію цілісності відповідей сервера, забезпечуючи безперервний захист даних під час їх передачі між клієнтом та сервером.

Організація зберігання та обробки даних здійснюється через реляційну систему керування базами даних MySQL версії 8.0, що забезпечує підтримку ACID-транзакційності операцій, критично важливої для збереження цілісності даних при проведенні оцінювання безпеки. Система керування базами даних налаштована з активованим шифруванням даних у стані спокою та обов'язковим використанням протоколів SSL/TLS для всіх з'єднань. Реалізована ролева модель контролю доступу на рівні бази даних з дотриманням принципу мінімальних привілеїв для кожного типу операцій.

Обслуговування HTTP-запитів на інфраструктурному рівні здійснюється вебсервером NGINX, конфігурованим з обов'язковими заголовками безпеки відповідно до рекомендацій OWASP. Конфігурація включає HTTP Strict Transport Security для примусового використання HTTPS-протоколу, Content Security Policy для запобігання XSS-атакам через контроль джерел завантажуваного контенту, та X-Frame-Options для захисту від clickjacking атак. NGINX функціонує як зворотний проксі-сервер з налаштованими механізмами обмеження частоти запитів для мітигації розподілених атак типу "відмова в обслуговуванні", створюючи додатковий рівень захисту системи.

Керування версіями програмного коду організовано через розподілену систему контролю версій Git з використанням платформи GitHub як центрального репозиторію. Процес розробки включає обов'язковий код-рев'ю для всіх модифікацій, автоматизоване сканування коду на предмет потенційних

вразливостей за допомогою статичного аналізу, та регулярну перевірку програмних залежностей на наявність відомих вразливостей згідно з базою даних Common Vulnerabilities and Exposures. Захист основної гілки коду реалізований через механізми branch protection rules з вимогою успішного проходження автоматизованих тестів безпеки, що гарантує високу якість програмного коду на всіх етапах життєвого циклу розробки.

Автоматизація розгортання та керування інфраструктурою реалізована через платформу Laravel Forge, що забезпечує автоматичне налаштування SSL-сертифікатів з подальшим моніторингом термінів дії та своєчасним оновленням. Процес розгортання включає автоматичну інсталяцію критичних оновлень безпеки операційної системи та програмних залежностей, безперервний моніторинг безпеки інфраструктури та створення зашифрованих резервних копій даних, формуючи комплексне середовище для безпечної експлуатації системи.

Архітектура вебдодатку забезпечує кросплатформенну сумісність та можливість роботи з даними без необхідності попередньої інсталяції спеціалізованого програмного забезпечення на клієнтських пристроях. Використання JavaScript-бібліотеки Chart.js дозволяє реалізувати інтерактивну візуалізацію аналітичної інформації через різноманітні типи графічних представлень, сприяючи інтуїтивному сприйняттю результатів оцінювання безпеки вебдодатків.

4.3. Висновки до розділу 4

Практична реалізація системи кількісного оцінювання захищеності вебдодатків підтвердила ефективність розробленого методу через створення повнофункціонального інструменту, що забезпечує комплексну підтримку всіх етапів оціночного процесу. Система реалізована з використанням сучасного технологічного стеку, що гарантує відповідність вимогам інформаційної безпеки, забезпечує архітектурну масштабованість та конфігураційну гнучкість.

Архітектурна організація системи базується на принципі модульного розділення функціональних компонентів, що включають підсистеми управління користувачами, адаптивного формування вибірки безпекових вимог, фіксації результатів оцінювання, зваженого агрегування показників та візуалізації аналітичних даних. Особливістю реалізації є підтримка багаторівневої деталізації результатів з можливістю їх подальшого редагування, що забезпечує динамічність процесу оцінювання та підвищує точність отриманих результатів.

Система інкорпорує механізми інтерактивної взаємодії з користувачем через спеціалізований інтерфейс, що сприяє ефективній інтерпретації даних та ідентифікації критичних областей з недостатнім рівнем захищеності. Програмна реалізація повною мірою відображає методичну логіку дослідження, включаючи ієрархічну структуризацію розділів, вимог та критеріїв, адаптацію вибірки на основі результатів попереднього опитування, нормалізацію вагових коефіцієнтів та обчислення інтегральних значень через систему взаємопов'язаних сервісів.

Інформаційна безпека системи забезпечується через імплементацію комплексних заходів на рівні обробки даних, з'єднань та автентифікації, що відповідають актуальним рекомендаціям OWASP та гарантують стійкість до типових векторів атак. Розроблена інформаційна система демонструє високу ефективність у контексті експертного аналізу, внутрішнього контролю безпеки та супроводу вебдодатків.

Модульна архітектурна побудова створює сприятливі умови для перспективного розширення функціональності, інтеграції з зовнішніми системами та автоматизації процедур повторного аналізу. Отримані результати практичної реалізації відкривають нові можливості для застосування розробленого методу у сфері практичного забезпечення кібербезпеки та створюють основу для подальших досліджень у галузі кількісного оцінювання захищеності вебдодатків.

ВИСНОВКИ

У дисертаційній роботі розв'язано актуальне наукове завдання – розроблення уніфікованого інструменту для реалізації запропонованого методу кількісного оцінювання рівня захищеності вебдодатку з урахуванням вимог міжнародних галузевих стандартів, індивідуальних характеристик проєкту. Основні наукові й практичні результати роботи полягають в наступному:

1. В результаті проведеного аналізу вебдодатків як складних багатокомпонентних систем та їх життєвого циклу розробки, дослідження існуючих міжнародних стандартів та традиційних методів тестування безпеки, ідентифікації їх обмежень, а також встановлення залежності між архітектурною складністю та вразливостями, обґрунтовано необхідність розробки нових підходів до оцінювання рівня захищеності вебдодатків. Це дозволило виокремити ключові фактори, що впливають на рівень безпеки вебдодатків, та визначити напрями для створення адаптивних методів оцінювання з урахуванням специфічних характеристик кожного проєкту.

2. Уперше розроблено метод оцінювання захищеності вебдодатків, який, на відміну від існуючих, дозволяє формувати кількісну оцінку рівня безпеки вебдодатку з використанням запропонованої системи критеріїв оцінювання та коефіцієнтів важливості. Зазначений метод адаптує вибір вимог та критеріїв до типу, архітектури та функціональності вебдодатку, забезпечуючи систематизований підхід до кількісного вимірювання рівня безпеки.

3. Уперше запропоновано використання апарату нечіткої логіки для підвищення точності та об'єктивності експертного оцінювання вагових коефіцієнтів важливості вимог та критеріїв. Це дало змогу формалізувати суб'єктивні судження експертів через механізми фазифікації, узгодження думок та дефазифікації, мінімізувати вплив індивідуальних упереджень та забезпечити математично обґрунтовану основу для кількісного вимірювання безпекових параметрів. Розроблений підхід дозволяє досягти врахувати невизначеність

експерта в оцінці, консенсусу серед експертів різного рівня кваліфікації та забезпечити стабільність результатів оцінювання.

4. Розроблено інформаційну систему підтримки методу визначення кількісної оцінки безпеки, яка автоматизує процес перевірки на відповідність стандарту OWASP ASVS та забезпечує прозорість і відтворюваність результатів. Спроектована архітектура системи з модульним розділенням функціональних компонентів (управління користувачами, адаптивне формування вимог, фіксація та агрегування результатів, візуалізація) забезпечує високу ефективність її функціонування. Система може інтегруватися в існуючі процеси DevSecOps та забезпечувати неперервний моніторинг рівня безпеки протягом усього життєвого циклу розробки програмного забезпечення.

5. Проведено експериментальне дослідження рівня безпеки вебдодатку на прикладі "OWASP Juice Shop", що підтвердило практичну застосовність та узгодженість розробленого адаптивного методу кількісної оцінки захищеності з результатами незалежних досліджень.

6. Проведено верифікацію розробленої інформаційної системи, що підтвердило коректність її функціонування та зручність користування. Сформовано систему початкових запитань користувачеві, що дає можливість автоматизувати відбір множини релевантних критеріїв та вимог для перевірки визначеного вебдодатку. Спроєктовано базу даних запитань, на основі яких обчислюється кількісна оцінка критеріїв та вимог, що дає можливість недосвідченому користувачеві здійснити оцінку якості захищеності вебдодатку. Зручний інтерфейс візуалізації результатів дає можливість користувачеві проаналізувати слабкі місця вебдодатку та зручно змінювати попередні відповіді під час оцінювання.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Wakil K., N.A. D. Extracting the Features of Modern Web Applications based on Web Engineering Methods. International Journal of Advanced Computer Science and Applications. 2019. Vol. 10, no. 2. URL: https://doi.org/10.14569/ijacsa.2019.0100209.

2. Wakil K., Jawawi D. N. A. Extensibility Interaction Flow Modeling Language Metamodels to Develop New Web Application Concerns. Kurdistan Journal of Applied Research. 2017. Vol. 2, no 3. C. 172–177. URL: https://doi.org/10.24017/science.2017.3.23.

3. Wakil K., Abang J. D. N. Model Driven Web Engineering: A Systematic Mapping Protocol. *ISCI 2014 -- IEEE Symposium on Computers & Informatics*, Jan. 5, 2014.

4. *PostgreSQL documentation*. *Chapter 53. Frontend/Backend Protocol*. URL: https://www.postgresql.org/docs/current/protocol.html (access date: 08.05.2023).

5. Sommerville I. Software Engineering. Tenth Edition. 10th. Courier Westford in the United States of America. 2016. 812 c.

6. M. I. H. Software Development Life Cycle (SDLC) Methodologies for Information Systems Project Management. International Journal For Multidisciplinary Research. 2023. Vol. 5, no 5. URL: https://doi.org/10.36948/ijfmr.2023.v05i05.6223.

7. Humble J., Farley D. Continuous Delivery: Reliable Software Releases Build through, Test, and Deployment Automation. Addison-Wesley Professional, 2011. 464 p. ISBN 978-0-321-60191-9.

8. Rapid Web Development Life Cycle: A Structured Methodology for Web Application Development / R. Kumar Das et al. *International Journal of Applied Engineering Research*. 2015. Vol. 10, iss. 16.

9. Mitigating Software Vulnerabilities through Secure Software Development with a Policy-Driven Waterfall Model / S. Hussain et al. *Journal of Engineering*. 2024. P. 1–15. URL: https://doi.org/10.1155/2024/9962691.

10. Abdulrazeg A. A., Norwawi N. M., Basir N. Extending V-model practices to support SRE to build secure web application. *2014 International Conference on Advanced Computer Science and Information System*, Jakarta, Indonesia, 18–19 Oct.

2014. IEEE, 2015. ISBN 978-1-4799-8075-8. URL: https://doi.org/10.1109/ICACSIS.2014.7065838.

11. Sathio S. A., Farah Siddiqui I., Arain Q. A. A Secure Software Specification Development Strategy for Enterprises : A Case Study Approach. International Journal of Scientific Research in Computer Science, Engineering and Information Technology. 2021. pp. 260–266. ISSN: 2456-3307. URL: https://doi.org/10.32628/cseit217155 .

12. Kasım Ö. Agile Software Development with Secure and Scrum-Centric Approach. *AJIT-e:* Academic Journal of Information Technology. 2024. Vol. 15, no 4. pp. 292–308. URL: https://doi.org/10.5824/ajite.2024.04.002.x.

13. Thakur A., Singh D., Maurya H. C. A Survey on Incremental Software Development Life Cycle Model. International Journal of Engineering Technology and Computer Research (IJETCR). 2013. Vol. 3, iss. 1. P. 102–16. ISSN 2348-2117.

14. Risks of rapid application development / R. Agarwal et al. *Communications of the ACM*. 2000. Vol. 43. no 11es. p. 1. URL: https://doi.org/10.1145/352515.352516.

15. 2023 Hacked Website & Malware Threat Report / B. Martin et al. ; ed. R. MacLeod. Sucuri Inc, 2024. 36 p. URL: https://sucuri.net/wp-content/uploads/2024/06/2023-Hacked-Website-Malware-Threat-Report.pdf (access date: 17.08.2023).

16. Usage Statistics and Market Share of Content Management Systems, June 2025. *W3Techs - extensive and reliable web technology surveys*. URL: https://w3techs.com/technologies/overview/content_management (access date: 17.08.2023).

17. Aldya A. P., Sutikno S., Rosmansyah Y. Measuring effectiveness of control of information security management system based on SNI ISO/IEC 27004: 2013 standard. IOP Conference Series: Materials Science and Engineering. 2019. Vol. 550, p. 012020. URL: https://doi.org/10.1088/1757-899x/550/1/012020.

18. Lundholm K., Hallberg J., Granlund H. Design and Use of Information Security Metrics Application of the ISO/IEC 27004 Standard. Stockholm : FOI – Swedish Defence Research Agency. 2011. p 52.

19. ISO/IEC 27002:2022. Information security, cybersecurity and privacy protection – Information security controls. Replaces ISO/IEC 27002:2013/Cor 2:2015. 2022.

20. Kazuhiro E., Motoei A., Komiyama T. Introduction of Quality Requirement and Evaluation Based on ISO/IEC SQuaRE Series of Standard. Communications in Computer and Information Science, Springer-Verlag Berlin Heidelberg, Germany. Springer-Verlag Berlin Heidelberg, 2013. P. 94–101. URL: https://doi.org/10.1007/978-3-642-35795-4_12.

21. Buccafurri F., Fotia L., Furfaro A. An analytical processing approach to supporting cyber security compliance assessment. *SIN '15: Proceedings of the 8th International Conference on Security of Information and Networks*. 2015. pp. 46–53. URL: https://doi.org/10.1145/2799979.2800007.

22. Payment Card Industry (PCI) Data Security Standard. 2017.

23. Ю. Войчур, Д. Медзатий. Метод аналізу вимог до програмного забезпечення на предмет пошуку значень атрибутів якості. Вимірювальна та обчислювальна техніка в технологічних процесах. Міжнародний науково-технічний журнал «Вимірювальна та обчислювальна техніка в технологічних процесах». 2024. №1. pp.146-151. ISSN2219-9365. URL: https://doi.org/10.31891/2219-9365-2024-77-18

24. Hovorushchenko T., Medzatyi D., Voichur Yu., Lebiga M. Method for forecasting the level of software quality based on quality attributes. Journal of Intelligent & Fuzzy Systems. 2023. vol. 44, no. 3, pp. 3891-3905. ISSN 1814-4225 (print). URL: https://doi.org/10.32620/reks.2023.3.19

25. Disanayake C., Dilhara S., Dharmaratna N. S. Rationalized Security Frameworks for Web Applications. Annual International Conference On Business Innovation (ICOBI) 2023, Homagama, Sri Lanka. Homagama : NSBM Green University, 2023.

26. Mitre. *Common Weakness Enumeration*. URL: https://cwe.mitre.org/ (access date: 07.12.2023).

27. OWASP Foundation. OWASP Top Ten. URL: https://owasp.org/www-project-top-ten.

28. Mahesh B. Evolving Trends in Web Application Vulnerabilities: A Comparative Study of OWASP Top 10 2017 and OWASP Top 10 2021. *International Journal of Engineering Technology and Management Sciences*. 2023. Vol. 7, no 6, pp. 262–269.

29. An OWASP Top Ten Driven Survey on Web Application Protection Methods / Fredj et al. *Risks and Security of Internet and Systems. CRiSIS 2020. Lecture Notes in Computer Science()*, 12 Feb. 2021 / eds.: J. Garcia-Alfaro et al. Springer, Cham, 2021. ISBN 978-3-030-68886-8. URL: https://doi.org/10.1007/978-3-030-68887-5_14.

30. Yudin O., Kharchenko V., Pevnev V. Scanning of Web-Applications: Algorithms and Software for Search of Vulnerabilities "Code Injection" and "Insecure Design". *2023 IEEE 12th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, Dortmund, Germany, 7 Sep. 2023. URL: https://doi.org/10.1109/idaacs58523.2023.10348918.

31. A04 Insecure Design - OWASP Top 10:2021. *OWASP Foundation, the Open Source Foundation for Application Security. OWASP Foundation*. URL: https://owasp.org/Top10/A04_2021-Insecure_Design (access date: 13.06.2025).

32. Keary E., Manico J. Secure Development Lifecycle. *Owasp.org*. URL: https://owasp.org/www-pdf-archive/Jim_Manico_(Hamburg)__Securiing_the_SDLC.pdf (access date: 27.12.2023).

33. Gurung G., Shah R., Jaiswal D. P. Software Development Life Cycle Models-A Comparative Study. International Journal of Scientific Research in Computer Science, Engineering and Information Technology. 2020. pp. 30–37. URL: https://doi.org/10.32628/cseit206410.

34. Acharya B., Sahu P. Software development lifecycle models: a review paper. International Journal of Advanced Research in Engineering and Technology (IJARET). 2020. Vol.11, no 12. URL: https://doi.org/10.34218/IJARET.11.12.2020.019.

35. Geogy M., Dharani A. Prominence of each phase in Software development life cycle contributes to the overall quality of a product. *2015 International Conference on Soft-Computing and Networks Security (ICSNS)*. Coimbatore, India, Feb. 25–27 2015. URL: https://doi.org/10.1109/icsns.2015.7292390.

36. Maltseva I., Chernish Y., Shtonda R. Analysis of some cyber threats in war. Cybersecurity: Education, Science, Technique. 2022. Vol. 16, no. 4, pp. 37–44. URL: https://doi.org/10.28925/2663-4023.2022.16.3744.

37. Kudinov O. The influence of information technologies on the socio-economic development of territorial communities. *Economics and Region*. 2022. URL: https://doi.org/10.32782/eir.2022.1(84).2549.

38. Tkach Y. Conceptual model of cyber space security. Technical Sciences and Technologies. 2020. Vol. 22, no. 4, pp. 96–108. URL: https://doi.org/10.25140/2411-5363-2020-4(22)-96-108.

39. Functions of the information security and cybersecurity system of critical information infrastructure / Y. Khlaponin et al. Cybersecurity: Education, Science, Technique. 2022. Vol. 3, no 15. pp. 124–134. URL: https://doi.org/10.28925/2663-4023.2022.15.1241341.

40. Abozeid A., AlHabshy A. A., ElDahshan K. A. Software Security Optimization Architecture (SoSOA) and its Adaptation for Mobile Applications. International Journal of Interactive Mobile Technologies (iJIM). 2021. Vol. 15, no 11. pp. 148. URL: https://doi.org/10.3991/ijim.v15i11.20133.

41. Measuring the Sustainable-Security of Web Applications Through a Fuzzy-Based Integrated Approach of AHP and TOPSIS / A. Agrawal et al. IEEE Access. 2019. Vol. 7. pp. 153936–153951. URL: https://doi.org/10.1109/access.2019.2946776.

42. Improving risk assessment model of cyber security using fuzzy logic inference system / M. Alali et al. Computers & Security. 2018. Vol. 74. pp. 323–339. URL: https://doi.org/10.1016/j.cose.2017.09.011.

43. Holik F., Neradova S. Vulnerabilities of modern web applications. *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. Opatija, Croatia, May 22–26. 2017. URL: https://doi.org/10.23919/mipro.2017.7973616.

44. Cyberattack Classificator Verification / I. Korobiichuk et al. International Conference on Diagnostics of Processes and Systems DPS 2017: Advanced Solutions in

Diagnostics and Fault Tolerant Control. 2018. pp. 402–411. URL: https://doi.org/10.1007/978-3-319-64474-5_34.

45. A study on web application security and detecting security vulnerabilities / S. Kumar et al. *2017 6th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, Noida, India, Sep.20–22, 2017. URL: https://doi.org/10.1109/icrito.2017.8342469.

46. Information Security Risk Assessment / I. Kuzminykh et al. *Encyclopedia*. 2021. Vol. 1, no 3. pp. 602–617. URL: https://doi.org/10.3390/encyclopedia1030050.

47. Software Package for Information Leakage Threats Relevance Assessment / V. Lakhno et al. Cybernetics Perspectives in Systems. 2022. Vol. 503. pp. 290. URL: https://doi.org/10.1109/USBEREIT51232.2021.9454994.

48. Improving Security of Web-Based Application Using ModSecurity and Reverse Proxy in Web Application Firewall / R. A. Muzaki et al. *2020 International Workshop on Big Data and Information Security (IWBIS)*, Depok, Indonesia, Oct. 17–18, 2020. URL: https://doi.org/10.1109/iwbis50925.2020.9255601.

49. Nagpure S., Kurkure S. Vulnerability Assessment and Penetration Testing of Web Application. *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)*, PUNE, India, Sep. 17–18, 2017. URL: https://doi.org/10.1109/iccubea.2017.8463920.

50. Kumar J. P., Nagendra K. V., Ravi T. Analysis of Security Vulnerabilities for Web based Application. *Fourth International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom2012)*, Bangalore, India. 2012. URL: https://doi.org/10.1049/cp.2012.2535.

51. Analyzing Risk Evaluation Frameworks and Risk Assessment Methods / R. Radu et al. *2020 19th RoEduNet Conference: Networking in Education and Research (RoEduNet)*. Bucharest, Romania, Dec. 11–12, 2020. URL: https://doi.org/10.1109/roedunet51892.2020.9324879 .

52. Web application security vulnerabilities detection approaches: A systematic mapping study / S. Rafique et al. *2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed*

Computing (SNPD), Takamatsu, Jun. 1–3 , 2015. URL: https://doi.org/10.1109/snpd.2015.7176244.

53. Applying Adaptive Security Techniques for Risk Analysis of Internet of Things (IoT)-Based Smart Agriculture / A. R. Riaz et al. *Sustainability*. 2022. Vol. 14, no 17. p. 10964. URL: https://doi.org/10.3390/su141710964.

54. 1. Sánchez-García I. D., Mejía J., Gilabert T. S. F. Cybersecurity Risk Assessment: A Systematic Mapping Review, Proposal, and Validation. *Applied Sciences*. 2022. URL: https://doi.org/10.3390/app13010395. 2022. Vol. 13, no. 1. p. 395. URL: https://doi.org/10.3390/app13010395.

55. Shrivastava A., Choudhary S., Kumar A. XSS vulnerability assessment and prevention in web application. *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*, Dehradun, India, Oct. 14–16, 2016. URL: https://doi.org/10.1109/ngct.2016.7877529.

56. Shevchenko S., Zhdanova Y. Information protection model based on information security risk assessment for small and medium-sized business. Cybersecurity Education Science Technique. 2022. no. 13. pp. 158–175. URL: https://doi.org/10.28925/2663-4023.2021.13.158157.

57. Teodoro N., Serrao C. Web application security: Improving critical web-based applications quality through in-depth security analysis. *2011 International Conference on Information Society (i-Society)*. London, Jun. 27–29, 2011. URL: https://doi.org/10.1109/i-society18435.2011.5978496 .

58. Potti U.-S., Huang H.-S., Chen H.-T. Security Testing Framework for Web Applications: Benchmarking ZAP V2.12.0 and V2.13.0 by OWASP as an example. *Proceedings of the Universal Academic Cluster International April Conference in Tokyo and Kyoto*. 2024. 1–7.

59. Performance Evaluation of Open Source Web Application Vulnerability Scanners based on OWASP Benchmark / P. A. Sarpong et.al. *International Journal of Computer Applications*. 2021. Vol. 174, no. 18. pp. 15–22. URL: https://doi.org/10.5120/ijca2021921070.

60. A Comparative Analysis of Vulnerability Management Tools: Evaluating Nessus, Acunetix, and Nikto for Risk-Based Security Solutions / B. Swetha et al. SSRN. Nov. 27, 2024. URL: http://dx.doi.org/10.2139/ssrn.5036282.

61. Erturk E., Rajan A. Web Vulnerability Scanners: A Case Study. arXiv preprint. 2017. RL: https://doi.org/10.48550/arXiv.1706.08017.

62. An analytical processing approach to supporting cyber security compliance assessment / F. Buccafurri et al. *SIN '15: The 8th International Conference on Security of Information and Networks*. New York. USA. 2015. URL: https://doi.org/10.1145/2799979.2800007.

63. Flater D. Bad Security Metrics Part 1: Problems. IT Professional. 2018. Vol. 20, no 1. pp. 64–68. URL: https://doi.org/10.1109/mitp.2018.011301733.

64. NISTIR 7564. Directions in Security Metrics Research. U.S. Department of Commerce, 2009. 26 c.

65. NIST Technical Series Publications. URL: https://nvlpubs.nist.gov/ (access date: 17.06.2024).

66. Audit and Compliance – is Documentation Your Weakest Link?. *Q Software*. URL: https://www.qsoftware.com/audit-reporting/poor-documentation-could-jeopardize-your-security-audit/ (access date: 19.06.2024).

67. Balancing Objectivity and Subjectivity in Risk Assessment. Pivot Point Security. URL: https://www.pivotpointsecurity.com/balancing-objectivity-subjectivity-when-assessing-risk (access date: 21.10.2024).

68. Sanders W. H. Quantitative Evaluation of Security Metrics. *2010 Seventh International Conference on the Quantitative Evaluation of Systems (QEST)*, Williamsburg, VA. USA. Sep. 15–18, 2010. URL: https://doi.org/10.1109/qest.2010.50 .

69. OWASP Application Security Verification Standard (ASVS). *OWASP Foundation*. URL: https://owasp.org/www-project-application-security-verification-standard (access date: 15.12.2024).

70. 800-63B. Digital Identity Guidelines: Authentication and Lifecycle Management. National Institute of Standards and Technology, 2017.

71. ISO/IEC 27002:2022. Information security, cybersecurity and privacy protection – Information security controls. Replaces ISO/IEC 27002:2013/Cor 2:2015. 2022.

72. ISO/IEC 27001:2022. Information security, cybersecurity and privacy protection – Information security management systems – Requirements. Replaces ISO/IEC 27001:2013/Cor 2:2015. 2022.

73. Walton D. Evaluating Expert Opinion Evidence. Law, Governance and Technology Series. Cham, 2016. p. 117–144. URL: https://doi.org/10.1007/978-3-319-19626-8_4.

74. Walton D. When Expert Opinion Evidence Goes Wrong. SSRN Electronic Journal. 2019. URL: https://doi.org/10.2139/ssrn.3465148.

75. An Integrated Spherical Fuzzy Multi-criterion Group Decision-Making Approach and Its Application in Digital Marketing Technology Assessment / K. Gao et al. International Journal of Computational Intelligence Systems. 2023. Vol. 16, no 1. URL: https://doi.org/10.1007/s44196-023-00298-3.

76. A new approach for handling uncertainty of expert judgments in complex decision problems / J. Więckowski et al. *IEEE Access*. 2024. p. 1. URL: https://doi.org/10.1109/access.2024.3445265 (access date: 13.06.2025).

77. Fuzzy Multi-Criteria Decision Making / ed. C. Kahraman. Boston, MA : Springer US, 2008. URL: https://doi.org/10.1007/978-0-387-76813-7 .

78. Deptuła A. M., Rudnik K. Fuzzy Approach Using Experts' Psychological Conditions To Estimate The Criteria Importance For The Assessment Of Innovative Projects Risk. *Opole University of Technology, Institute of Processes and Products Innovation*. 2018. Vol. 9, no. 1. pp. 13–23.

79. Firdaus T., Yadav S., Devare M. Modified Clustering Algorithm for Energy Efficiency Utilizing Fuzzy Logic In WSN. *Proceedings of National Conference on Machine Learning*. 2019.

80. Ic Y. T. A fuzzy computing approach to aggregate expert opinions using parabolic and exparabolic approximation procedures for solving multi-criteria group decision-making problems. *Neural Computing and Applications*. 2024. URL: https://doi.org/10.1007/s00521-024-09448-w.

81. Lu C., Lan J., Wang Z. Aggregation of Fuzzy Opinions Under Group Decision-Making Based on Similarity and Distance. Journal of Systems Science and Complexity. 2006. Vol. 19, no.1, pp. 63–71. URL: https://doi.org/10.1007/s11424-006-0063-y .

82. Sims j. R., Zhenyuan w. Fuzzy measures and fuzzy integrals: an overview. *International Journal of General Systems*. 1990. Vol. 17, no. 2-3, pp. 157–189. URL: https://doi.org/10.1080/03081079008935106.

83. Gilda K. S., Satarkar D. S. L. Analytical overview of defuzzification methods. International Journal of Advance Research, Ideas and Innovations in Technology. 2020. Vol. 6, no 2.

84. OWASP Juice Shop. OWASP Foundation. URL: https://owasp.org/www-project-juice-shop/ (access date: 23.01.2025).

85. Revniuk O.A., Zagorodna N.V. Методологія кількісної оцінки захищеності вебдодатку електронної комерції на етапі експлуатації. Scientific Bulletin of Ivano-Frankivsk National Technical University of Oil and Gas. 2024. Vol. 57, no.2, pp. 107–119. URL: https://doi.org/10.31471/1993-9965-2024-2(57)-107-119.

86. Tryhubets B., Tryhubets M., Zagorodna N. Analysis of the efficiency of open source and commercial vulnerability scanners for e-commerce web applications. Scientific journal of the Ternopil national technical university. 2024. Vol. 116, no. 4. pp. 23–30. URL: https://doi.org/10.33108/visnyk_tntu2024.04.023.

ДОДАТКИ

ДОДАТОК А
СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ ТА ВІДОМОСТІ ПРО АПРОБАЦІЮ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНОЇ РОБОТИ

Праці, в яких опубліковано основні наукові результати дисертації:

1. Revniuk O. A., Zagorodna N. V., Kozak R. O., Karpinski M. P., Flud L. O. “The improvement of web-application SDL process to prevent Insecure Design vulnerabilities”, *Applied Aspects of Information Technology*, *7*(2), pp.162–174, 2024. I**SSN 2617-4316 (Print), https://doi.org/10.15276/aait.07.2024.12.**
2. Ревнюк О., Постолюк А. “Дослідження застосування адаптивних методик оцінювання ризиків безпеки для вебдодатків”, *Computer Systems and Information Technologies*, *3*, ст.34–43, 2024. **ISSN 2710-0766, https://doi.org/10.31891/csit-2024-3-5.**
3. Ревнюк О. А., Загородна Н. В. “Методологія кількісної оцінки захищеності вебдодатку електронної комерції на етапі експлуатації”, *Scientific Bulletin of Ivano-Frankivsk National Technical University of Oil and Gas*, *2(57)*, ст.107–119, 2024. **ISSN1993–9965 print, https://doi.org/10.31471/1993-9965-2024-2(57)-107-119.**
4. Revniuk O., Zagorodna N., Ulichev O. «Adaptive methodology for computing the quantitative Security status indicator of web applications», *Central Ukrainian Scientific Bulletin Technical Sciences*, *2*(10(41)), pp.3–10, 2024. **ISSN 2664-262Xб https://doi.org/10.32515/2664-262x.2024.10(41).2.3-10.**
5. Ulichev O., Papizh L., & Revniuk O. “Advancements in software Testing: a scientific perspective”, *Central Ukrainian Scientific Bulletin Technical Sciences*, *1*(10(41)), pp. 3–16, 2024. **ISSN 2664-262X, https://doi.org/10.32515/2664-262x.2024.10(41).1.3-16.**

6. Revniuk O., Zagorodna N., Kozak R., Yavorskyy B. I. «Development of an information system for the quantitative assessment of web application security based on the OWASP ASVS standard», Scientific Journal of the Ternopil National Technical University, 2 (118), pp. 56-65, 2025. **ISSN: 2522-4433 Print**.

Праці, які засвідчують апробацію матеріалів дисертації:

7. О. Ревнюк, Н. Загородна. “Моделі та методи оцінювання якості вебдодатків”, Матеріали IX науково-технічної конференції «Інформаційні моделі, системи та технології», ТНТУ, Тернопіль, Україна, 2021, ст. 73–74.
8. Ревнюк О. «Якість управління інформаційною безпекою організацій», Матеріали X науково-технічної конференції «Інформаційні моделі, системи та технології», ТНТУ, Тернопіль, Україна, 2022, ст. 42–43.
9. Н. Загородна, О. Ревнюк, Р. Козак. «Кількісна оцінка безпеки прототипу інтернет-магазину OWASP Juice Shop», Матеріали XIV міжнародної науково-технічної конференції ITSEC: Безпека інформаційних технологій, ЗУНУ-ДУІКТ, Тернопіль-Київ, 22-24 травня 2025, ст.75-76

ДОДАТОК Б
АКТИ ВПРОВАДЖЕННЯ

Проректор з наукової роботи
Тернопільського національного
технічного університету ім. І. Пулюя
_____ Павло МАРУЩАК

АКТ

Про впровадження результатів дисертаційної роботи «Розробка інформаційної системи оцінювання якості захисту веб-додатків» здобувача ступеня доктора філософії PhD за спеціальністю 125 «Кібербезпека та захист інформації» **Ревнюка Олександра Андрійовича** у навчальний процес Тернопільського національного технічного університету ім. І. Пулюя

Даним документом підтверджується застосування результатів дисертаційної роботи здобувача ступеня доктора філософії Ревнюка О. А. в освітній діяльності кафедри кібербезпеки. У процесі викладання дисципліни «Безпека програмного забезпечення» для студентів спеціальності 125 «Кібербезпека та захист інформації» були використані:

- концептуальна модель для оцінки рівня захисту веб-додатків відповідно до OWASP ASVS;
- методика формалізованої кількісної оцінки безпеки;
- розроблений вебзастосунок для проведення автоматизованого аналізу рівня безпеки.

Застосування цих результатів дозволило підвищити рівень обізнаності студентів з сучасними підходами до забезпечення захисту вебдодатків та підтримати прийняття рішень щодо їх удосконалення.

Завідувач кафедри кібербезпеки _____ Наталія ЗАГОРОДНА

ДОДАТОК В
АКТИ ВПРОВАДЖЕННЯ

Проректор з наукової роботи
Тернопільського національного
технічного університету ім. І. Пулюя
_____ Павло МАРУЩАК

АКТ

Про впровадження результатів дисертаційної роботи «Розробка інформаційної системи оцінювання якості захисту веб-додатків» здобувача ступеня доктора філософії PhD за спеціальністю 122 «Комп'ютерні науки» Ревнюка Олександра Андрійовича в навчальному процесі Тернопільського національного технічного університету ім. І. Пулюя.

Цим актом підтверджується, що результати дисертаційної роботи здобувача ступеня доктора філософії Ревнюка О. А. використано в навчальному процесі кафедри комп'ютерних наук та кафедри кібербезпеки. Зокрема, при проведенні лекційних занять з дисципліни "Веб-технології" для студентів освітнього рівня бакалавр спеціальностей 122 «Комп'ютерні науки» та 125 «Кібербезпека та захист інформації» використано концептуальну модель оцінювання рівня безпеки веб-додатків на основі стандарту OWASP ASVS та методику кількісної оцінки рівня захищеності на основі системи формалізованих критеріїв.

Використання вказаних результатів у навчальному процесі дало змогу підвищити ступінь проінформованості студентів щодо галузевих стандартів контролю безпеки веб-додатків, методики їх застосування та забезпечення аналітичної підтримки прийняття рішень щодо вдосконалення архітектури та впровадження засобів захисту.

Завідувач кафедри кібербезпеки _____ Наталія ЗАГОРОДНА

Завідувач кафедри комп'ютерних наук _____ Ігор БОДНАРЧУК

ДОДАТОК Г
АКТИ ВПРОВАДЖЕННЯ

ТОВ «САСДЖЕТ»

Код ЄДРПОУ 45345707
м. Тернопіль, вул. За Рудкою, 33

АКТ ВПРОВАДЖЕННЯ

результатів дисертаційного дослідження Ревнюка Олександра Андрійовича "Розробка інформаційної системи оцінювання якості захисту веб-додатків" в ТОВ «SaaSJet»

Результати, отримані Ревнюком О. А. при виконанні ним дисертаційного дослідження на тему «Розробка інформаційної системи оцінювання якості захисту веб-додатків», а саме:

- концептуальна модель оцінювання рівня безпеки веб-додатків на основі стандарту OWASP ASVS;
- методика кількісної оцінки рівня захищеності на основі формалізованих критеріїв;
- програмна реалізація веб-додатку для проведення автоматизованої перевірки рівня безпеки веб-застосунків;

були використані у ТОВ «SaaSJet» для супроводу процесів внутрішнього аудиту безпеки, підвищення якості захисту веб-продуктів компанії та оптимізації процесів інтеграції вимог безпеки в життєвий цикл розробки (SDLC).

Впровадження результатів дисертаційної роботи дозволило підвищити ефективність внутрішнього контролю безпеки веб-продуктів та забезпечити аналітичну підтримку прийняття рішень щодо вдосконалення архітектури та впровадження засобів захисту.

Даний акт не є підставою для проведення фінансових розрахунків.

Директор ТОВ «САСДЖЕТ» _____ Васьківська О.А.

ТОВАРИСТВО З ОБМЕЖЕНОЮ ВІДПОВІДАЛЬНІСТЮ «САСДЖЕТ» 45345707 УКРАЇНА

ДОДАТОК Д

СПИСОК ВІДІБРАНИХ ВИМОГ

Розділ «Architecture, Design and Threat Modeling»

1. Verify the use of a secure software development lifecycle that addresses security in all stages of development.
2. Verify the use of threat modeling for every design change or sprint planning to identify threats, plan for countermeasures, facilitate appropriate risk responses, and guide security testing.
3. Verify that all user stories and features contain functional security constraints, such as "As a user, I should be able to view and edit my profile. I should not be able to view or edit anyone else's profile"
4. Verify the use of unique or special low-privilege operating system accounts for all application components, services, and servers.
5. Verify that communications between application components, including APIs, middleware and data layers, are authenticated. Components should have the least necessary privileges needed.
6. Verify that all authentication pathways and identity management APIs implement consistent authentication security control strength, such that there are no weaker alternatives per the risk of the application.
7. Verify that there is an explicit policy for management of cryptographic keys and that a cryptographic key lifecycle follows a key management standard such as NIST SP 800-57.
8. Verify that all keys and passwords are replaceable and are part of a well-defined process to re-encrypt sensitive data.
9. Verify that a common logging format and approach is used across the system.
10. Verify the application encrypts communications between components, particularly when these components are in different containers, systems, sites, or cloud providers.
11. Verify that application components verify the authenticity of each side in a communication link to prevent person-in-the-middle attacks. For example, application components should validate TLS certificates and chains.
12. Verify that a source code control system is in use, with procedures to ensure that check-ins are accompanied by issues or change tickets. The source code control system should have access control and identifiable users to allow traceability of any changes.
13. Verify that application deployments adequately sandbox, containerize and/or isolate at the network level to delay and deter attackers from attacking other applications, especially when they are performing sensitive or dangerous actions such as deserialization.
14. Verify the application does not use unsupported, insecure, or deprecated client-side technologies such as NSAPI plugins, Flash, Shockwave, ActiveX, Silverlight, NACL, or client-side Java applets.

Authentication

1. Verify that user set passwords are at least 12 characters in length (after multiple spaces are combined).
2. Verify that passwords of at least 64 characters are permitted, and that passwords of more than 128 characters are denied.
3. Verify that password truncation is not performed. However, consecutive multiple spaces may be replaced by a single space.
4. Verify that any printable Unicode character, including language neutral characters such as spaces and Emojis are permitted in passwords.
5. Verify users can change their password.
6. Verify that password change functionality requires the user's current and new password.
7. Verify that there are no password composition rules limiting the type of characters permitted. There should be no requirement for upper or lower case or numbers or special characters.
8. Verify that the user can choose to either temporarily view the entire masked password, or temporarily view the last typed character of the password on platforms that do not have this as built-in functionality.
9. Verify that anti-automation controls are effective at mitigating breached credential testing, brute force, and account lockout attacks. Such controls include blocking the most common breached passwords, soft lockouts, rate limiting, CAPTCHA, ever increasing delays between attempts, IP address restrictions, or risk-based restrictions such as location, first login on a device, recent attempts to unlock the account, or similar. Verify that no more than 100 failed attempts per hour is possible on a single account.
10. Verify that secure notifications are sent to users after updates to authentication details, such as credential resets, email or address changes, logging in from unknown or risky locations. The use of push notifications - rather than SMS or email - is preferred, but in the absence of push notifications, SMS or email is acceptable as long as no sensitive information is disclosed in the notification.
11. Verify system generated initial passwords or activation codes SHOULD be securely randomly generated, SHOULD be at least 6 characters long, and MAY contain letters and numbers, and expire after a short period of time. These initial secrets must not be permitted to become the long term password.
12. Verify that enrollment and use of user-provided authentication devices are supported, such as a U2F or FIDO tokens.
13. Verify that renewal instructions are sent with sufficient time to renew time bound authenticators.
14. Verify that passwords are stored in a form that is resistant to offline attacks. Passwords SHALL be salted and hashed using an approved one-way key derivation or password hashing function. Key derivation and password hashing functions take a password, a salt, and a cost factor as inputs when generating a password hash.

15. Verify that the salt is at least 32 bits in length and be chosen arbitrarily to minimize salt value collisions among stored hashes. For each credential, a unique salt value and the resulting hash SHALL be stored.

16. Verify that if bcrypt is used, the work factor SHOULD be as large as verification server performance will allow, with a minimum of 10.

17. Verify that a system generated initial activation or recovery secret is not sent in clear text to the user.

18. Verify password hints or knowledge-based authentication (so-called "secret questions") are not present.

19. Verify password credential recovery does not reveal the current password in any way.

20. Verify shared or default accounts are not present (e.g. "root", "admin", or "sa").

21. Verify that if an authentication factor is changed or replaced, that the user is notified of this event.

22. Verify that time-based OTPs have a defined lifetime before expiring.

23. Verify that time-based OTP can be used only once within the validity period.

Session Management

1. Verify the application never reveals session tokens in URL parameters.

2. Verify the application generates a new session token on user authentication.

3. Verify that session tokens possess at least 64 bits of entropy.

4. Verify that session tokens are generated using approved cryptographic algorithms.

5. Verify that logout and expiration invalidate the session token, such that the back button or a downstream relying party does not resume an authenticated session, including across relying parties.

6. If authenticators permit users to remain logged in, verify that re-authentication occurs periodically both when actively used or after an idle period.

7. Verify that the application gives the option to terminate all other active sessions after a successful password change (including change via password reset/recovery), and that this is effective across the application, federated login (if present), and any relying parties.

8. Verify that users are able to view and (having re-entered login credentials) log out of any or all currently active sessions and devices.

9. Verify that cookie-based session tokens have the 'Secure' attribute set.

10. Verify that cookie-based session tokens have the 'HttpOnly' attribute set.

11. Verify that cookie-based session tokens utilize the 'SameSite' attribute to limit exposure to cross-site request forgery attacks.

12. Verify that cookie-based session tokens use the "Host-" prefix so cookies are only sent to the host that initially set the cookie.

Access Control

1. Verify that the application enforces access control rules on a trusted service layer, especially if client-side access control is present and could be bypassed.
2. Verify that all user and data attributes and policy information used by access controls cannot be manipulated by end users unless specifically authorized.
3. Verify that access controls fail securely including when an exception occurs. (C10)
4. Verify that sensitive data and APIs are protected against Insecure Direct Object Reference (IDOR) attacks targeting creation, reading, updating and deletion of records, such as creating or updating someone else's record, viewing everyone's records, or deleting all records.
5. Verify that the application or framework enforces a strong anti-CSRF mechanism to protect authenticated functionality, and effective anti-automation or anti-CSRF protects unauthenticated functionality.
6. Verify administrative interfaces use appropriate multi-factor authentication to prevent unauthorized use.
7. Verify that directory browsing is disabled unless deliberately desired. Additionally, applications should not allow discovery or disclosure of file or directory metadata, such as Thumbs.db, .DS_Store, .git or .svn folders.

Validation, Sanitization and Encoding

1. Verify that all input (HTML form fields, REST requests, URL parameters, HTTP headers, cookies, batch files, RSS feeds, etc) is validated using positive validation (allow lists).
2. Verify that structured data is strongly typed and validated against a defined schema including allowed characters, length and pattern (e.g. credit card numbers, e-mail addresses, telephone numbers, or validating that two related fields are reasonable, such as checking that suburb and zip/postcode match).
3. Verify that URL redirects and forwards only allow destinations which appear on an allow list, or show a warning when redirecting to potentially untrusted content.
4. Verify that all untrusted HTML input from WYSIWYG editors or similar is properly sanitized with an HTML sanitizer library or framework feature.
5. Verify that the application avoids the use of eval() or other dynamic code execution features. Where there is no alternative, any user input being included must be sanitized or sandboxed before being executed.
6. Verify that output encoding is relevant for the interpreter and context required. For example, use encoders specifically for HTML values, HTML attributes, JavaScript, URL parameters, HTTP headers, SMTP, and others as the context requires, especially from untrusted inputs (e.g. names with Unicode or apostrophes, such as ねこ or O'Hara).
7. Verify that data selection or database queries (e.g. SQL, HQL, ORM, NoSQL) use parameterized queries, ORMs, entity frameworks, or are otherwise protected from database injection attacks.

8. Verify that the application protects against JSON injection attacks, JSON eval attacks, and JavaScript expression evaluation.

9. Verify that the application protects against LDAP injection vulnerabilities, or that specific security controls to prevent LDAP injection have been implemented.

10. Verify that the application protects against OS command injection and that operating system calls use parameterized OS queries or use contextual command line output encoding.

11. Verify that the application protects against Local File Inclusion (LFI) or Remote File Inclusion (RFI) attacks.

12. Verify that the application protects against XPath injection or XML injection attacks. (C4)

13. Verify that the application uses memory-safe string, safer memory copy and pointer arithmetic to detect or prevent stack, buffer, or heap overflows.

14. Verify that sign, range, and input validation techniques are used to prevent integer overflows.

Stored Cryptography

1. Verify that regulated private data is stored encrypted while at rest, such as Personally Identifiable Information (PII), sensitive personal information, or data assessed likely to be subject to EU's GDPR.

2. Verify that regulated health data is stored encrypted while at rest, such as medical records, medical device details, or de-anonymized research records.

3. Verify that regulated financial data is stored encrypted while at rest, such as financial accounts, defaults or credit history, tax records, pay history, beneficiaries, or de-anonymized market or research records.

4. Verify that all cryptographic modules fail securely, and errors are handled in a way that does not enable Padding Oracle attacks.

5. Verify that industry proven or government approved cryptographic algorithms, modes, and libraries are used, instead of custom coded cryptography.

6. Verify that random number, encryption or hashing algorithms, key lengths, rounds, ciphers or modes, can be reconfigured, upgraded, or swapped at any time, to protect against cryptographic breaks.

Error Handling and Logging

1. Verify that the application does not log credentials or payment details. Session tokens should only be stored in logs in an irreversible, hashed form.

2. Verify that the application does not log other sensitive data as defined under local privacy laws or relevant security policy.

3. Verify that each log event includes necessary information that would allow for a detailed investigation of the timeline when an event happens.

4. Verify that all logging components appropriately encode data to prevent log injection.

5. Verify that security logs are protected from unauthorized access and modification.

6. Verify that time sources are synchronized to the correct time and time zone. Strongly consider logging only in UTC if systems are global to assist with post-incident forensic analysis.

7. Verify that a generic message is shown when an unexpected or security sensitive error occurs, potentially with a unique ID which support personnel can use to investigate.

8. Verify that exception handling (or a functional equivalent) is used across the codebase to account for expected and unexpected error conditions.

Data Protection

1. Verify the application protects sensitive data from being cached in server components such as load balancers and application caches.

2. Verify the application minimizes the number of parameters in a request, such as hidden fields, Ajax variables, cookies and header values.

3. Verify the application can detect and alert on abnormal numbers of requests, such as by IP, user, total per hour or day, or whatever makes sense for the application.

4. Verify that regular backups of important data are performed and that test restoration of data is performed.

5. Verify that backups are stored securely to prevent data from being stolen or corrupted.

6. Verify the application sets sufficient anti-caching headers so that sensitive data is not cached in modern browsers.

7. Verify that data stored in browser storage (such as localStorage, sessionStorage, IndexedDB, or cookies) does not contain sensitive data.

8. Verify that users have a method to remove or export their data on demand.

9. Verify that users are provided clear language regarding collection and use of supplied personal information and that users have provided opt-in consent for the use of that data before it is used in any way.

10. Verify that all sensitive data created and processed by the application has been identified, and ensure that a policy is in place on how to deal with sensitive data.

Malicious Code

1. Verify that if the application has a client or server auto-update feature, updates should be obtained over secure channels and digitally signed. The update code must validate the digital signature of the update before installing or executing the update.

2. Verify that the application employs integrity protections, such as code signing or subresource integrity. The application must not load or execute code from untrusted sources, such as loading includes, modules, plugins, code, or libraries from untrusted sources or the Internet.

Business Logic

1. Verify the application has appropriate limits for specific business actions or transactions which are correctly enforced on a per user basis.
2. Verify that the application has anti-automation controls to protect against excessive calls such as mass data exfiltration, business logic requests, file uploads or denial of service attacks.
3. Verify the application has business logic limits or validation to protect against likely business risks or threats, identified using threat modeling or similar methodologies.
4. Verify that the application monitors for unusual events or activity from a business logic perspective. For example, attempts to perform actions out of order or actions which a normal user would never attempt.
5. Verify that the application has configurable alerting when automated attacks or unusual activity is detected.

Files and Resources

1. Verify that files obtained from untrusted sources are scanned by antivirus scanners to prevent upload and serving of known malicious content.
2. Verify that the web tier is configured to serve only files with specific file extensions to prevent unintentional information and source code leakage. For example, backup files (e.g. .bak), temporary working files (e.g. .swp), compressed files (.zip, .tar.gz, etc) and other extensions commonly used by editors should be blocked unless required.
3. Verify that direct requests to uploaded files will never be executed as HTML/JavaScript content.
4. Verify that the web or application server is configured with an allow list of resources or systems to which the server can send requests or load data/files from.

API and Web Service

1. Verify API URLs do not expose sensitive information, such as the API key, session tokens etc.
2. Verify that requests containing unexpected or missing content types are rejected with appropriate headers (HTTP response status 406 Unacceptable or 415 Unsupported Media Type).
3. Verify that enabled RESTful HTTP methods are a valid choice for the user or action, such as preventing normal users using DELETE or PUT on protected API or resources.
4. Verify that JSON schema validation is in place and verified before accepting input.
5. Verify that REST services explicitly check the incoming Content-Type to be the expected one, such as application/xml or application/json.

Configuration

1. Verify that server configuration is hardened as per the recommendations of the application server and frameworks in use.
2. Verify that the application, configuration, and all dependencies can be re-deployed using automated deployment scripts, built from a documented and tested runbook in a reasonable time, or restored from backups in a timely fashion.
3. Verify that all components are up to date, preferably using a dependency checker during build or compile time. (C2)
4. Verify that all unneeded features, documentation, sample applications and configurations are removed.
5. Verify that web or application server and application framework debug modes are disabled in production to eliminate debug features, developer consoles, and unintended security disclosures.
6. Verify that the HTTP headers or any part of the HTTP response do not expose detailed version information of system components.
7. Verify that every HTTP response contains a Content-Type header. Also specify a safe character set (e.g., UTF-8, ISO-8859-1) if the content types are text/*, /+xml and application/xml. Content must match with the provided Content-Type header.
8. Verify that all API responses contain a Content-Disposition: attachment; filename="api.json" header (or other appropriate filename for the content type).
9. Verify that a Content Security Policy (CSP) response header is in place that helps mitigate impact for XSS attacks like HTML, DOM, JSON, and JavaScript injection vulnerabilities.
10. Verify that all responses contain a X-Content-Type-Options: nosniff header.
11. Verify that a suitable Referrer-Policy header is included to avoid exposing sensitive information in the URL through the Referer header to untrusted parties.
12. Verify that the content of a web application cannot be embedded in a third-party site by default and that embedding of the exact resources is only allowed where necessary by using suitable Content-Security-Policy: frame-ancestors and X-Frame-Options response headers.
13. Verify that the application server only accepts the HTTP methods in use by the application/API, including pre-flight OPTIONS, and logs/alerts on any requests that are not valid for the application context.
14. Verify that the supplied Origin header is not used for authentication or access control decisions, as the Origin header can easily be changed by an attacker.
15. Verify that the Cross-Origin Resource Sharing (CORS) Access-Control-Allow-Origin header uses a strict allow list of trusted domains and subdomains to match against and does not support the "null" origin.