https://doi.org/10.31891/2307-5732-2026-361-69
УДК 004.413

ЯШИНА ОКСАНА
Хмельницький національний університет
https://orcid.org/0000-0001-7816-1662
e-mail: yashynao@khmnu.edu.ua

МЕТОДИ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ ДЛЯ РОЗРОБКИ ВЕБЗАСТОСУНКІВ В ASP.NET CORE

В статті представлено методи та ключові аспекти безпекових питань щодо розробки вебзастосунків із використанням технології ASP.NET Core. Загалом даного роду питання розглядаються з різних точок зору, а саме управління доступом та довірою, захисту від ін'єкцій та міжсайтових атак, архітектурних рішень та ізоляції тощо.

У сучасних розподілених системах безпека ґрунтується на принципі найменших привілеїв, що забезпечується методом авторизації Claims-Based Authorization, який є науково більш гнучким та точним, ніж традиційна авторизація на основі ролей. Доступ надається не на підставі визначення ролі, а на підставі набору вимог чи тверджень, що дозволяє реалізувати так званий гранульований (деталізований) контроль доступу (Fine-Grained Access Control). У контексті прикладного програмного інтерфейсу, ключ аутентифікації користувача виступає як механізм інкапсуляції довіри. Токен містить криптографічно підписані твердження, що дозволяє застосункам бути без стану. Однак, науковою проблемою тут є відкликання токенів та забезпечення короткого часу життя токенів доступу для мінімізації ризику їх компрометації.

Для боротьби із вразливостями документ OWASP Top 10 вбудовано у фреймворк через методи безпечного кодування. ASP.NET Core використовує механізми контекстного кодування, наприклад, у Razor Pages для автоматичного контекстного кодування вихідних даних (HTML-кодування). Це є первинною лінією захисту від XSS-атак шляхом зміни інтерпретації даних браузером. Механізм захисту від CSRF-атак базується на використанні синхронізованих токенів, що є криптографічним способом підтвердження, що запит походить із довіреного джерела. Також використовується параметризація запитів через використання об'єктно-реляційного відображення, таких як Entity Framework Core і є реалізацією патерну безпечного доступу до даних, який мінімізує простір атаки для SQL-ін'єкцій, відділяючи команди від даних.

З погляду системної архітектури, ASP.NET Core вирішує питання безпеки на рівні конфігурації та розгортання. Система проміжного програмного забезпечення дозволяє централізовано впроваджувати політики безпеки, такі як перенаправлення на HTTPS та управління CORS (Cross-Origin Resource Sharing), що забезпечує агрегацію контролю безпеки на мережевому рівні.

Свою роль у забезпеченні безпекових питань відіграє також і управління секретами, оскільки сучасна парадигма вимагає повного відділення секретів ключів, паролів від коду та конфігураційних файлів. ASP.NET Core підтримує інтеграцію зі сховищами секретів, що є реалізацією принципу «не зберігати секрети у коді», а це підвищує безпеку шляхом ізоляції чутливої інформації.

Ключові слова: *ASP.NET Core, безпека, проектування, вебзастосунок, інформаційні технології, інженерія програмного забезпечення.*

YASHYNA OKSANA
Khmelnytskyi National University

SECURITY METHODS FOR WEB APPLICATION DEVELOPMENT IN ASP.NET CORE

The article presents methods and key aspects of security issues related to the development of web applications using ASP.NET Core technology. In general, these issues are considered from different points of view, namely access and trust management, protection against injections and cross-site attacks, architectural solutions and isolation.

In modern distributed systems, security is based on the principle of least privilege, which is ensured by the Claims-Based Authorisation method, which is scientifically more flexible and accurate than traditional role-based authorisation. Access is granted not on the basis of role definition, but on the basis of a set of requirements or assertions, which allows for the implementation of so-called fine-grained access control. In the context of an application programming interface, the user authentication key acts as a trust encapsulation mechanism. The token contains cryptographically signed assertions, allowing applications to be stateless. However, the scientific problem here is revoking tokens and ensuring a short lifetime for access tokens to minimise the risk of their compromise.

Combating OWASP Top 10 vulnerabilities is built into the framework through secure coding practices. ASP.NET Core uses context-sensitive encoding mechanisms, such as in Razor Pages, to automatically context-encode output (HTML encoding). This is the primary line of defence against XSS attacks by changing how the browser interprets data. The mechanism for protecting against CSRF attacks is based on the use of synchronised tokens, which are a cryptographic way of confirming that a request comes from a trusted source. It also uses request parameterisation through object-relational mapping, such as Entity Framework Core, and implements a secure data access pattern that minimises the attack surface for SQL injections by separating commands from data.

From a system architecture perspective, ASP.NET Core addresses security issues at the configuration and deployment level. The middleware system allows for the centralised implementation of security policies such as HTTPS redirection and CORS (Cross-Origin Resource Sharing) management, providing aggregated security control at the network level.

Secret management also plays a role in ensuring security, as the modern paradigm requires complete separation of key secrets and passwords from code and configuration files. ASP.NET Core supports integration with Key Vaults (secret storage), which is an implementation of the 'do not store secrets in code' principle, and this increases security by isolating sensitive information.

Keywords: *ASP.NET Core, security, information technology, software engineering, software design, web application.*

Стаття надійшла до редакції / Received 17.12.2025
Прийнята до друку / Accepted 11.01.2026
Опубліковано / Published 29.01.2026

Вступ та постановка проблеми

Безпека вебзастосунків входить у сферу інформаційної безпеки, яка зосереджується на недоліках безпеки на рівні вебзастосунків та їх рішеннях. Розвиток мережі Інтернет в сучасних умовах набуває великого прискорення, а обсяги інформації, що поширюється через Інтернет, зростає з кожним днем. Різні бізнес-галузі продовжують інтегрувати свою операційну діяльність в цифровий світ, що несе свої ризики та робить інформаційну безпеку вебзастосунків важливішою, ніж будь-коли.

Сфера вебтехнологій є однією з найбільш динамічних в інженерії програмного забезпечення і сучасні дослідження та практичні інновації тут зосереджені на підвищенні продуктивності, безпеки, доступності та на використанні нових парадигм розробки для забезпечення кращого досвіду користувача. Великої уваги приділяється таким аспектам, як нова архітектура та продуктивність фронтенду, куди входить серверний рендеринг (SSR) та гідратація, WebAssembly, Core Web Vitals та оптимізація UX; розвиток API та комунікаційні патерни, де розглядається GraphQL та Bounded Contexts, Real-time та двостороння комунікація, API Gateway та Observability.

Науковий аналіз безпекових питань в ASP.NET Core вимагає конкретизації механізмів та пов'язаних з ними загроз, що стосуються криптографії, управління довірою та архітектурної ізоляції.

Аналіз останніх досліджень

У своїх дослідженнях щодо безпеки вебзастосунків Пірог О. охоплює основи веббезпеки, зокрема типи вебсистем, клієнтсерверну архітектуру, модель OSI, протоколи та методи шифрування. Розглянуто контроль доступу та процеси ідентифікації, аутентифікації, авторизації та управління сесіями, методи протидії атакам brute-force і захисту вебсерверів. Також є опис ключових вразливостей вебзастосунків, такі як XSS, CSRF, SQL-ін'єкції, XXE, SSRF та ін., а також методів їх запобігання. Розглянуто DoS-атаки та способи боротьби з ними, а також захист від витоків даних. Висвітлено процес тестування безпеки, тестування на проникнення з використанням інструментів: Nmap, Burp Suite, Metasploit та ін. Окрема увага приділяється безпечному програмуванню впродовж усього SDLC, моделюванню загроз (STRIDE, PASTA) та запобіганню помилок програмування [1].

Дослідниками Ersahin B. та Ersahin M. було проаналізовано найпоширеніші та серйозні вебвразливості разом з їхніми рішеннями. Їх дослідження зосереджені на запобіганні розробниками проблем безпеки протягом життєвого циклу розробки, а також виокремлення найкращих практик та інструментів у контрольний список безпеки для вебзастосунків. [2].

Кіщук Н. та Сердюк П. у своїх працях дають детальний опис вразливостей, так званих слабких місць у ASP.NET Core-застосунках [3].

У [4] демонструється, як організації трансформували свої практики безпеки, впроваджуючи заходи безпеки на початкових етапах розробки, а не розглядаючи їх як міркування на завершальному етапі. Завдяки аналізу численних тематичних досліджень та результатів досліджень, у цій статті досліджуються переваги ранньої інтеграції безпеки, включаючи скорочення часу виявлення вразливостей, підвищення операційної ефективності та покращення співпраці команди. У статті також досліджуються рамки впровадження, методології та організаційні проблеми, пов'язані з цією трансформацією, надаючи уявлення про успішні стратегії пом'якшення наслідків та найкращі практики інтеграції безпеки в сучасних життєвих циклах розробки програмного забезпечення.

У [5] наведено повний огляд оцінки вразливостей вебрзастосунків та тестування на проникнення, підкреслюючи необхідність проактивних заходів безпеки для захисту конфіденційних даних та збереження цілісності додатків. Це дослідження виявляє та класифікує вразливості вебзастосунків (OWASP), які можуть поставити під загрозу безпеку застосунку та призвести до витоків даних користувачів. Основною метою проекту є безпека та конфіденційність даних користувачів. Інтеграція практик безпеки в життєвий цикл розробки програмного забезпечення шляхом дослідження реальних тематичних досліджень та галузевих стандартів. Ці дослідження доповнюють зростаючий обсяг знань у сфері безпеки вебзастосунків, допомагаючи організаціям створювати стійкий захист від кіберзагроз.

Загалом цей аналіз демонструє, що наявна досить потужна методологічна основа безпекових питань щодо розробки вебзастсунків. Разом з тим виявлено, що досить мало досліджень, які демонструють аналітичну складову безпеки під час розробки вебзастосунків із використанням технології ASP.NET Core.

Метою роботи є визначення головних методів та аспектів безпекових питань під час проектування та розробки вебзастосунків із використанням ASP.NET Core. Це дозволить враховувати проблемні моменти та розв’язувати їх під час створення програмного продукту.

Основна частина

Актуальні дослідження концентруються на подоланні обмежень односторінкових застосунків (SPA) та забезпеченні високої швидкості завантаження, а саме увага приділяється вивченню та оптимізації механізмів SSR та статичного генерування сайтів (SSG) для зменшення часу до інтерактивності (TTI). Актуальним є дослідження проблем гідратації та розробка нових фреймворків, які мінімізують навантаження на клієнта, наприклад, Island Architecture.

Використання WASM для винесення ресурсомістких обчислень, наприклад, криптографія, відеообробка, 3D-графіка, у веббраузер із продуктивністю, близькою до нативної. Дослідження фокусуються на інтеграції WASM із сучасними JavaScript-фреймворками та покращенні взаємодії між WASM-модулями та DOM. Core Web Vitals та оптимізація UX, що включає розробку методологій для системного моніторингу та

покращення показників Core Web Vitals (LCP, FID, CLS) на реальних користувацьких даних, а не лише на синтетичних тестах. Сучасні вебзастосунки вимагають більш гнучких та ефективних методів обміну даними, ніж традиційний REST. Дослідження ефективності GraphQL у великих розподілених системах (мікросервісах) та його інтеграції з концепцією обмежених контекстів (Bounded Contexts) з Domain-Driven Design (DDD). Актуальним є управління кешуванням та запобігання надмірному вибору даних у складних запитах.

Real-time та двостороння комунікація включає використання WebSockets та нових протоколів, таких як WebTransport, зокрема HTTP/3, для забезпечення низькозатримуваних, стійких з'єднань у реальному часі. Дослідження стосуються масштабування цих рішень у великомасштабних розподілених чатах або ігрових сервісах. API Gateway та Observability полягає у розробці розширених функцій API Gateway для забезпечення єдиної точки входу, управління версіями, а також інтеграції з розподіленим трасуванням та логуванням для покращення спостережуваності.

Особливу роль відіграє веббезпека та конфіденційність, оскільки із розвитком браузерних технологій постійно виникають нові загрози та вимоги до конфіденційності. Тому важливими є такі складові безпеки як конфіденційність та SameSite Cookies, куди входять дослідження нових механізмів браузерної безпеки та їхнього впливу на міжсайтову функціональність, зокрема, управління SameSite-атрибутами та використання Storage Access API для контролю відстеження. Важливою складовою також є безпека клієнтського коду (Client-Side Security), що включають методи запобігання атакам Client-Side Supply Chain Attack (атаки на сторонні залежності JavaScript), а також вдосконалення політик Content Security Policy (CSP) для мінімізації ризиків XSS-атак. Також важливим моментом є аутентифікація без пароля, а саме впровадження та дослідження протоколів WebAuthn (FIDO2) як стандарту для стійкої до фішингу, багатофакторної аутентифікації.

З наукової точки зору, безпекові питання для ASP.NET Core розглядаються через призму моделей загроз (Threat Modeling), архітектурних рішень та застосування криптографічних примітивів для забезпечення конфіденційності, цілісності та доступності даних. Основні проблеми пов'язані з управлінням довірою та ізоляцією компонентів у розподілених вебсередовищах.

Критично важливою складовою будь-якого вебзастосунку є безпека даних і ASP.NET (як класичний ASP.NET, так і сучасний ASP.NET Core) пропонує широкий набір вбудованих методів, функцій та механізмів для захисту інформації на різних рівнях. Дані рівні включають:

- методи захисту від поширених вебзагроз;
- методи автентифікації та авторизації;
- криптографію та захист даних у стані спокою;
- конфігурацію та роботу з секретами.

ASP.NET Core має вбудовані методи та засоби для боротьби з найбільш відомими вразливостями. Сюди входить Cross-Site Scripting, Cross-Site Request Forgery, SQL Injection та небезпечна десеріалізація. ASP.NET Core автоматично кодує виведені дані (HTML Encoding) у Razor Pages та MVC View, унеможливлюючи виконання шкідливого скрипту. Рекомендується завжди використовувати перевірені хелпери для рендерингу даних. ASP.NET Core також надає вбудований фільтр [ValidateAntiForgeryToken], який автоматично генерує та перевіряє криптографічний токен у формі, гарантуючи, що запит був ініційований із власного домену користувача. Використання сучасних ORM, таких як Entity Framework Core є стандартною практикою та забезпечує параметризовані запити по замовчуванню, ефективно запобігаючи виконанню довільного коду бази даних. Платформа також надає інструменти для безпечної роботи з даними, наприклад, фільтри валідації моделі, які запобігають зв'язуванню (model binding) небажаних властивостей (overposting).

Фундаментальними аспектами безпеки, які ASP.NET реалізує через гнучкий конвеєр проміжного програмного забезпечення є автентифікація та авторизація, куди входить ASP.NET Core Identity, автентифікація на основі токенів (JWT), політики авторизації.

ASP.NET Core Identity - це вбудована система управління користувачами, яка забезпечує хешування паролів з використанням сильних алгоритмів, управління ролями, двофакторну автентифікацію та зовнішні входи, наприклад, Google, Facebook. Для сучасних API та односторінкових застосунків (SPA) ASP.NET Core має вбудовану підтримку JSON Web Tokens (JWT). Це дозволяє здійснювати автентифікацію без збереження стану (stateless), що є критично важливим для масштабування. Замість простої перевірки ролей, ASP.NET Core підтримує авторизацію на основі політик та вимог, що дозволяє надавати доступ на основі конкретних атрибутів користувача, а не лише його ролі. ASP.NET має механізми для захисту конфіденційних даних, що зберігаються або передаються за допомогою криптографії та захисту даних у стані спокою. Сюди входить Data Protection API, шифрування конфігураційних файлів, HTTPS/TLS.

Data Protection API - це вбудований криптографічний API, який використовується для захисту важливих даних усередині самого застосунку. Наприклад, він захищає куки автентифікації, токени CSRF та ключі сесії. ASP.NET Core Data Protection API - це не просто шифрувальний інструмент, а криптографічна підсистема, що використовує автентифіковане шифрування для забезпечення як конфіденційності, так і цілісності даних. У науковому контексті це розглядається як реалізація механізмів для захисту «даних у транзиті» в межах одного застосунку, наприклад, куки автентифікації, токени CSRF. Ключові дослідницькі питання тут включають управління ключами, а саме зберігання, ротацію та відкликання, яке має бути стійким до компрометації.

Хоча зазвичай конфігураційні файли не містять безпосередньо секретів у сучасних системах, ASP.NET Core заохочує використання Secrets Manager під час розробки та Azure Key Vault або іншого сховища секретів у продукті, щоб повністю виключити збереження чутливої інформації в коді чи конфігурації.

ASP.NET Identity використовує алгоритми хешування з ключем (Key Derivation Functions, KDF), такі як удосконалений PBKDF2. Це науково обґрунтований підхід, який вимагає високих обчислювальних витрат на атаку «грубою силою» за рахунок солі та ітерацій, що підвищує ентропію хешу. Платформа активно сприяє використанню HTTPS для шифрування даних під час передачі. У ASP.NET Core є вбудований middleware HTTPS Redirection, який автоматично перенаправляє HTTP-запити на захищений HTTPS-порт. Правильне управління секретами є першочерговим завданням для безпеки даних. До управління секретами входить ієрархія конфігурації, управління секретами на рівні розробки, Cross-Origin Resource Sharing. ASP.NET Core підтримує гнучку ієрархію конфігурації, яка дозволяє легко перезаписувати налаштування із зовнішніх, безпечних джерел, таких як змінні середовища (Environment Variables) або зовнішні сховища секретів.

Для уникнення збереження секретів у системі контролю версій (Git) під час розробки використовується інструмент Secret Manager. Для контролю доступу до API з різних доменів використовується Cross-Origin Resource Sharing (CORS) middleware, що дозволяє чітко визначити, які домени можуть взаємодіяти з API.

Важливо в контексті безпеки вебзастосунків також виділити методи тестування безпеки у вебзастосунках на базі ASP.NET Core, що включає кілька критично важливих підходів, які забезпечують перевірку стійкості до реальних атак, зокрема тестування на проникнення. Зокрема, до таких методів відносяться:

- статичний аналіз безпеки застосунків;
- динамічний аналіз безпеки застосунків;
- інтерактивний аналіз безпеки застосунків;
- ручне тестування на проникнення.

Статичний аналіз безпеки застосунків оцінює код без його фактичного виконання. SAST-інструменти, такі як Roslyn Analyzers або спеціалізовані комерційні рішення, сканують вихідний код ASP.NET Core для виявлення антипатернів та потенційних вразливостей. Цей метод шукає небезпечні API-виклики, помилки конфігурації безпеки, наприклад неправильне налаштування CORS, та вразливі патерни, наприклад, пряме використання незахищених SQL-запитів замість Entity Framework Core. Його висока ефективність проявляється на ранніх етапах розробки, дозволяючи виправити вразливості ще до розгортання.

Динамічний аналіз безпеки застосунків симулює реальні атаки на застосунок під час його виконання у тестовому або стейджинг-середовищі. DAST-інструменти, наприклад, OWASP ZAP або Burp Suite, діють як зловмисники, надсилаючи спеціально сформовані HTTP-запити для виявлення вразливостей, які проявляються лише під час інтерактивної взаємодії. Тестування фокусується на ін'єкціях, наприклад XSS та SQL Injection, спробах обходу авторизації (IDOR) шляхом маніпуляції ідентифікаторами, та перевірці правильності налаштування HTTP-заголовків безпеки та обмеження методів HTTP.

IAST поєднує переваги SAST та DAST, відстежуючи потік виконання коду в реальному часі. Інструмент IAST інтегрується всередину середовища виконання ASP.NET Core і моніторить виконання коду, отримуючи дані від DAST-сканера. Коли DAST надсилає шкідливий ввід, IAST відстежує, як цей ввід поширюється по коду застосунку (Data Flow Analysis) до «чутливих точок» (Sinks), таких як база даних. Це дозволяє точно локалізувати рядок коду, який є вразливим, і значно зменшує кількість помилкових спрацьовувань (false positives) порівняно з іншими автоматизованими методами.

Ручне тестування на проникнення є найбільш комплексним і глибоким методом, де кваліфікований фахівець вручну досліджує бізнес-логіку застосунку, яку автоматизовані інструменти не можуть виявити. Воно фокусується на складних логічних вразливостях, наприклад зловживання бізнес-логікою або спроби несанкціонованого підвищення привілеїв. Це тестування є незамінним для перевірки коректності реалізації Claims-based Authorization та унікальних функціональних вимог ASP.NET Core-застосунку.

Висновки

Отже, безпека даних у ASP.NET - це не окремий компонент, а комплексна система, вбудована в архітектуру фреймворку, яка вимагає від розробника свідомого використання вбудованих механізмів та дотримання сучасних практик для захисту від відомих вразливостей. Критичним елементом є захист стану застосунку та конфіденційних даних у стані спокою.

У сфері захисту стану застосунку та даних у стані спокою, ASP.NET Core Data Protection API є центральним криптографічним примітивом, наукове значення якого полягає у використанні автентифікованого шифрування, що не лише забезпечує конфіденційність через шифрування, але й цілісність через код автентифікації повідомлення. Актуальні дослідницькі питання тут стосуються забезпечення стійкості системи до атак, зокрема, через регулярну та безпечну ротацію майстер-ключів, а також розробку методів моніторингу, які б виявляли спроби підробки захищеного стану, наприклад, куки. Що стосується хешування паролів, застосування Key Derivation Functions, таких як PBKDF2, є реалізацією принципу повільного хешування. Це підвищує обчислювальну складність для атакуючого при спробі перебору (brute-force attack), що є емпірично доведеною необхідністю для захисту облікових даних.

У розподілених архітектурах критичне значення має деталізований контроль доступу та інкапсуляція довіри. Claims-Based Authorization є впровадженням політики найменших привілеїв на основі динамічного набору тверджень про користувача. Це дозволяє здійснювати динамічну оцінку дозволу доступу в реальному часі. Використання JSON Web Tokens (JWT) для Stateless-автентифікації в API вирішує проблему масштабування, але створює значну проблему відкликання токенів (Revocation). Оскільки JWT підписаний, але не перевіряється на сервісах при кожному запиті, необхідно використовувати короткий час життя (TTL) токена

або впроваджувати додаткові механізми перевірки, такі як чорні списки (Blacklists) або короткочасні сесії. Це є активною темою досліджень у сфері безпеки мікросервісів.

Боротьба із вразливостями, наприклад за допомогою XSS (Cross-Site Scripting) та SQL-ін'єкції, реалізується через архітектурні патерни безпечного кодування. Контекстне кодування вихідних даних у Razor є основним механізмом запобігання XSS, забезпечуючи, що дані, введені користувачем, завжди інтерпретуються браузером як текст, а не як виконуваний код. Це підхід, заснований на зміні контексту інтерпретації. Запобігання SQL-ін'єкціям досягається завдяки використанню патерну параметризованих запитів через EF Core, що ізолює команди SQL від даних користувача, перетворюючи потенційно шкідливий ввід на літеральне значення, а не на частину виконуваного запиту. Захист від CSRF (Cross-Site Request Forgery) забезпечується використанням криптографічного токена, який виступає як доказ походження запиту, що є реалізацією концепції захисту від підробки запиту.

На рівні системної конфігурації безпека забезпечується через ізоляцію чутливої інформації та централізоване управління політиками. Інтеграція ASP.NET Core із зовнішніми сховищами секретів гарантує, що конфіденційні дані, наприклад, ключі API, рядки підключення, ніколи не зберігаються у системі контролю версій або у файлах конфігурації. Це є реалізацією принципу ізоляції секретів. Використання Middleware для HTTPS Redirection та CORS дозволяє забезпечити агрегований контроль за мережевою безпекою та політикою доступу між джерелами відповідно до заданих правил, що дозволяє аналітично відстежувати та керувати доступом на периметрі застосунку.

Література

1. Безпека вебдодатків : навч. посібн. / О. В. Пірог. – Електронні дані. – Житомир : Житомирська політехніка, 2025. 290 с.
2. Ersahin B., Ersahin M. Web application security. *South Florida Journal of Development*. Miami, 2022. T. 3. P. 4194-4203. ISSN SN 2675-5459.
3. Kishchuk N., Serdyuk P. Common vulnerabilities in asp.net applications. *VI International Scientific and Practical Conference «GRUNDLAGEN DER MODERNEN WISSENSCHAFTLICHEN FORSCHUNG»*, Zurich, Switzerland, 24 May 2024.DOI https://doi.org/10.36074/logos-24.05.2024.042.
4. Kaithe B. K. Shift Left Security: A Paradigm Shift in Software Development Security Integration. *European Journal of Computer Science and Information Technology*. 2025. Vol. 1324.
5. Gandikota P. S. S. K., Valluri D., Mundru S. B. Web Application Security through Comprehensive Vulnerability Assessment. *Procedia Computer Science*. 2023. Vol. 230. P. 168–182.
6. Босько В. В., Константинова Л.В., Марченко К.М. Web-програмування. Частина 1 (frontend). Кропивницький: ЦНТУ, 2022. 210 с.
7. Griffiths C. The Latest 2024 Cyber Crime Statistics. URL: https://aag-it.com/the-latestcyber-crime-statistics/ (date of access: 11.10.2025).
8. Мосіюк О. О. Web-технології. Житомир: ЖДУ ім. І. Франка, 2020. 54 с.
9. Терейковський І. А., Гнатюк С. О. Захист інформації в комп'ютерних системах. К.: КПІ, 2022. 135 с.
10. Стандарт якості ПЗ ISO 25010:2011. URL: https://qalight.ua/baza-znaniy/yakist-pz-za-iso-250102011/ (дата звернення 29.10.2025).
11. Jacobs M. 9 Ways Hackers Exploit ASP.NET - and How to Prevent Them. URL: https://medium.com/letstalk-tech/9-ways-hackers-exploit-asp-net-and-how-toprevent-them-cf87a8b70991 (date of access: 21.10.2025).

References

1. Bezpeka vebdodatkiv : navch. posibn. / O. V. Piroh. – Elektronni dani. – Zhytomyr : Zhytomyrska politekhnika, 2025. 290 s.
2. Ersahin B., Ersahin M. Web application security. *South Florida Journal of Development*. Miami, 2022. Vol. 4. P. 4194–4203. ISSN SN 2675-5459.
3. Kishchuk N., Serdyuk P. Common vulnerabilities in asp.net applications. *VI International Scientific and Practical Conference «GRUNDLAGEN DER MODERNEN WISSENSCHAFTLICHEN FORSCHUNG»*, Zurich, Switzerland, 24 May 2024.DOI https://doi.org/10.36074/logos-24.05.2024.042.
4. Kaithe B. K. Shift Left Security: A Paradigm Shift in Software Development Security Integration. *European Journal of Computer Science and Information Technology*. 2025. Vol. 1324.
5. Gandikota P. S. S. K., Valluri D., Mundru S. B. Web Application Security through Comprehensive Vulnerability Assessment. *Procedia Computer Science*. 2023. Vol. 230. P. 168–182.
6. Bosko V. V., Konstantynova L.V., Marchenko K.M. Web-prohramuvannia. Chastyna 1 (frontend). Kropyvnytskyi: TsNTU, 2022. 210 s.Griffiths C. The Latest 2024 Cyber Crime Statistics. URL: https://aag-it.com/the-latestcyber-crime-statistics/ (date of access: 11.10.2025).
7. Mosiiuk O. O. Web-tekhnolohii. Zhytomyr: ZhDU im. I. Franka, 2020. 54 s.
8. Tereikovskyi I. A., Hnatiuk S. O. Zakhyst informatsii v kompiuternykh systemakh. K.: KPI, 2022. 135 s.
9. Standart jakosti PZ ISO 25010:2011. URL: https://qalight.ua/baza-znaniy/yakist-pz-za-iso-250102011/ (date of access: 29.10.2025)..
10. Jacobs M. 9 Ways Hackers Exploit ASP.NET - and How to Prevent Them. URL: https://medium.com/letstalk-tech/9-ways-hackers-exploit-asp-net-and-how-toprevent-them-cf87a8b70991 (date of access: 21.10.2025).