**Ткаченко Віталій Андрійович**, аспірант 1 курсу
Київський національний університет імені Тараса Шевченка, місто Київ
ORCID: 0009-0004-5812-5313

**АНАЛІЗ ТА ОЦІНКА СИСТЕМ БЕЗПЕКИ У ФРЕЙМВОРКУ BLAZOR**

У статті проведено комплексний аналіз систем безпеки фреймворку Blazor, який є сучасною технологією для розробки веб-додатків від Microsoft. Досліджено архітектурні особливості Blazor Server та Blazor WebAssembly з точки зору безпеки, проаналізовано вбудовані механізми захисту та потенційні вразливості. Особливу увагу приділено методам забезпечення безпеки, включаючи налаштування автентифікації, захист від CSRF-атак та безпечне зберігання даних. Розроблено практичні рекомендації щодо підвищення рівня безпеки Blazor-додатків та запропоновано перспективні напрямки подальших досліджень у цій галузі.

У сучасному світі веб-додатки стали невід'ємною частиною бізнес-процесів та повсякденного життя користувачів. З розвитком технологій зростає і складність кіберзагроз, що робить питання безпеки веб-додатків надзвичайно актуальним. Особливої уваги заслуговує аналіз безпеки нових технологій, серед яких важливе місце посідає фреймворк Blazor від Microsoft.

Blazor представляє собою інноваційний підхід до розробки веб-додатків, що дозволяє створювати інтерактивні користувацькі інтерфейси з використанням C# замість JavaScript. Ця особливість не тільки спрощує розробку для .NET-розробників, але й створює нові виклики з точки зору безпеки, оскільки традиційні механізми захисту веб-додатків потребують адаптації до особливостей архітектури Blazor.

Метою даного дослідження є комплексний аналіз систем безпеки у фреймворку Blazor, виявлення потенційних вразливостей та розробка рекомендацій щодо їх усунення. Основними завданнями є дослідження архітектурних особливостей Blazor з точки зору безпеки, аналіз вбудованих механізмів захисту, оцінка потенційних загроз та розробка практичних рекомендацій щодо забезпечення безпеки Blazor-додатків.

Фреймворк Blazor пропонує два основних підходи до розробки веб-додатків: Blazor Server та Blazor WebAssembly, кожен з яких має свої особливості з точки зору безпеки. В Blazor Server весь код додатку виконується на сервері, а взаємодія з клієнтом відбувається через SignalR. Це забезпечує високий рівень безпеки, оскільки чутливий код та дані ніколи не потрапляють на клієнтську сторону. Натомість, Blazor WebAssembly виконує код безпосередньо у браузері користувача, що створює додаткові виклики для забезпечення безпеки.

Вбудовані механізми автентифікації та авторизації в Blazor базуються на стандартних можливостях платформи .NET Core. Система автентифікації підтримує різноманітні провайдери ідентичності, включаючи Windows Authentication, Identity Server, Azure Active Directory та інші. Авторизація реалізується через декларативні атрибути та програмні перевірки на рівні компонентів.

Особливу увагу варто приділити безпеці комунікації між сервером та клієнтом через SignalR у Blazor Server. Цей механізм використовує WebSocket з'єднання з автоматичним fallback на Long Polling у разі необхідності. Для захисту передачі даних застосовується HTTPS протокол та вбудовані механізми антипідробки (anti-forgery).

Blazor Server має специфічні вразливості, пов'язані з особливостями його архітектури [1, с. 1]. Основним ризиком є можливість DOS-атак через відкриті SignalR з'єднання, оскільки кожне клієнтське підключення споживає серверні ресурси. Також існує ризик витоку пам'яті при неправильному управлінні життєвим циклом компонентів та потенційна вразливість до CSRF-атак при недостатньому захисті SignalR endpoints.

У випадку Blazor WebAssembly основні загрози пов'язані з виконанням коду на стороні клієнта. Зловмисники можуть спробувати модифікувати або декомпілювати .NET збірки, що завантажуються в браузер. Крім того, існує ризик витоку чутливих даних через локальне сховище браузера та можливість несанкціонованого доступу до API endpoints.

Порівняльний аналіз з іншими веб-фреймворками виявляє як спільні аспекти безпеки, так і унікальні особливості Blazor , зокрема вбудовані механізми захисту, про які також згадують в роботах з аналізу безпеки веб-фреймворків [3, с. 1]. Подібно до Angular та React, Blazor стикається з типовими вразливостями клієнтського рендерингу, такими як XSS-атаки при неправильній санітизації користувацького вводу, вразливості керування станом додатку та ризики, пов'язані з управлінням залежностями через пакетні менеджери. Всі три фреймворки також поділяють проблеми безпеки, пов'язані з рендерингом на стороні сервера (SSR): потенційні витоки пам'яті, вразливості серіалізації даних та необхідність захисту від CSRF-атак.

Однак Blazor має суттєві відмінності в механізмах безпеки. На відміну від Angular, який використовує TypeScript, та React, що базується на JavaScript, Blazor WebAssembly компілює C# код у WebAssembly, що забезпечує додатковий рівень захисту від реверс-інжинірингу та маніпуляцій з кодом. SignalR у Blazor Server пропонує вбудовану підтримку WebSocket з автоматичним шифруванням та стисненням даних, тоді як у Angular та React подібну функціональність потрібно реалізовувати окремо через Socket.io або інші бібліотеки.

Інтеграція з екосистемою .NET надає Blazor доступ до потужних вбудованих механізмів захисту, серед яких: система Data Protection API (DPAPI) для безпечного зберігання критичних даних, вбудована підтримка криптографічних операцій через System.Security.Cryptography, автоматична валідація та санітизація форм через DataAnnotations, механізм попередження CSRF-атак через антипідробні токени (AntiforgeryToken), та розширена система керування ідентичністю ASP.NET Core Identity. Додатково, Blazor успадковує від .NET надійну систему типізації та компіляції, що зменшує ризик помилок під час виконання та підвищує загальну безпеку додатку.

Налаштування автентифікації в Blazor вимагає комплексного підходу. Для Blazor Server рекомендується використовувати Cookie-автентифікацію з належним налаштуванням параметрів безпеки cookies. У випадку Blazor WebAssembly оптимальним є використання JWT-токенів з коротким терміном дії та механізмом оновлення. Інтеграція з OAuth 2.0 та OpenID Connect забезпечує надійну автентифікацію через зовнішніх провайдерів. Також існують ризик неправильного налаштування автентифікації у Blazor WebAssembly, що є однією з основних проблем, з якими стикаються розробники [2, с. 389].

Захист від CSRF-атак реалізується через вбудований механізм антипідробки токенів у Blazor Server. Для Blazor WebAssembly важливо забезпечити належну валідацію JWT-токенів на сервері та використовувати механізм SameSite cookies. Додатково рекомендується імплементувати CORS політики та валідацію заголовків запитів.

Безпечне зберігання даних на клієнті вимагає особливої уваги у Blazor WebAssembly. Чутливі дані не повинні зберігатися у localStorage, натомість рекомендується використовувати sessionStorage або спеціалізовані рішення для шифрування даних на клієнті. Важливо також забезпечити очищення даних при виході користувача з системи.

Захист API endpoints повинен включати валідацію моделей, фільтрацію вхідних даних та механізми rate limiting. Для Blazor WebAssembly критично важливо реалізувати належну авторизацію на рівні API, оскільки клієнтський код може бути модифікований.

При розробці безпечних Blazor додатків важливо дотримуватися принципу найменших привілеїв та забезпечувати багаторівневий захист. Рекомендується використовувати вбудовані механізми валідації форм та захист від XSS-атак через вбудований HTML-санітайзер. Важливо також забезпечити належне логування безпекових подій та впровадити механізми виявлення аномалій.

Конфігурація безпеки на рівні проекту повинна включати налаштування HTTPS, коректні заголовки безпеки (включаючи CSP), та належну конфігурацію автентифікації та авторизації. Для Blazor WebAssembly важливо налаштувати integrity checks для завантажуваних ресурсів та реалізувати механізми оновлення клієнтських бібліотек.

Моніторинг та аудит безпеки повинні здійснюватися на регулярній основі. Рекомендується використовувати інструменти статичного аналізу коду, проводити penetration testing та впровадити систему моніторингу безпекових інцидентів. Важливо також забезпечити регулярне оновлення залежностей та компонентів фреймворку.

Проведене дослідження демонструє, що фреймворк Blazor надає потужні вбудовані механізми безпеки, але вимагає ретельного підходу до їх налаштування та використання. Основними викликами є забезпечення безпеки SignalR комунікації в Blazor Server та захист клієнтського коду в Blazor WebAssembly.

Для покращення безпеки Blazor додатків рекомендується: впровадження комплексного підходу до автентифікації та авторизації, використання сучасних криптографічних методів, регулярний аудит безпеки та навчання розробників щодо специфічних аспектів безпеки Blazor.

**Література**

1. Iacobuț F.-C., Calomfirescu R.-A., Dan C., Drăghici B. G., Stan O. P. Security Analysis and Architecture of a Blazor-Based Web Application // 2024 16th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Iasi, Romania. – 2024. – P. 1–6. – DOI: 10.1109/ECAI61503.2024.10607494.
2. André P. M., Stiévenart Q., Ghafari M. Developers Struggle with Authentication in Blazor WebAssembly // 2022 IEEE International Conference on Software Maintenance and Evolution (ICSME), Limassol, Cyprus. – 2022. – P. 389–393. – DOI: 10.1109/ICSME55016.2022.00045.
3. Aborujilah A., Adamu J., Shariff S. M., Awang Long Z. Descriptive Analysis of Built-in Security Features in Web Development Frameworks // 2022 16th International Conference on Ubiquitous Information Management and Communication (IMCOM), Seoul, Korea, Republic of. – 2022. – P. 1–8. – DOI: 10.1109/IMCOM53663.2022.9721750.