УДК 004.8:004.056.5:004.7.056 DOI: 10.31673/2412-9070.2026.318103

В. М. ДАНИЛЬЧЕНКО[1], PhD, доцент;
ORCID: 0009-0004-6839-2132
Д. В. ГАРМАШ[2], магістр;
ORCID: 0009-0001-5551-5621
С. І. ОТРОХ[2], д-р техн. наук, професор;
ORCID: 0000-0001-9008-0902
О. В. САРАФАННІКОВ[2], аспірант,
ORCID: 0000-0002-8373-4629
[1]Державний університет інформаційно-комунікаційних технологій, Київ
[2]Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

ПІДВИЩЕННЯ БЕЗПЕКИ ЗАХИСТУ ДАНИХ СИСТЕМ ЗІ ШТУЧНИМ ІНТЕЛЕКТОМ ПРИ ВИКОРИСТАННІ КОНТЕКСТНО-ЗАЛЕЖНОЇ АВТЕНТИФІКАЦІЇ

Досліджено актуальність застосування контекстно-залежних методів автентифікації для підвищення безпеки інформаційних систем в умовах зростання кількості кіберзагроз. Проаналізовано обмеження традиційних статичних методів автентифікації, що не враховують контекстуальні параметри сеансів користувачів. Описано процес розроблення системи багатофакторного контекстного аналізу на основі каркасу Spring, PostgreSQL та аспектно-орієнтованого програмування. Система використовує технічні, географічні, часові та поведінкові параметри для динамічної оцінки ризиків несанкціонованого доступу. Реалізовано архітектуру на основі взірця Стратегія з модульною системою перевірювачів для перевірки правил доступу, включаючи мережеві адреси, типи пристроїв, переглядачі, операційні системи та часові обмеження. Використання аспектно-орієнтованого програмування Spring забезпечує декларативну інтеграцію перевірок доступу через анотації з мінімальним втручанням у код застосунку. Тип даних JSONB PostgreSQL дозволяє гнучко зберігати динамічні умови правил без зміни схеми бази даних. Розроблена система може бути використана в корпоративних інформаційних системах, хмарних платформах та банківських застосунках.

Ключові слова: контекстно-залежна автентифікація, Spring Framework, аспектно-орієнтоване програмування, PostgreSQL, багатофакторний аналіз, патерн Strategy, безпека даних, захист інформаційних систем.

Вступ

Постановка проблеми. Забезпечення безпеки даних є однією з найважливіших проблем сучасного цифрового суспільства. За даними звіту Verizon Data Breach Investigations Report 2024 року, понад 81% успішних кібератак пов'язані з компрометацією облікових записів користувачів через слабкі або викрадені паролі [1]. Традиційні методи автентифікації, що базуються виключно на статичних параметрах, таких як паролі або ПІН-коди, виявляються недостатньо ефективними для протидії сучасним загрозам, включаючи фішинг, атаки грубої сили та соціальну інженерію.

Двофакторна автентифікація (2ФА) частково вирішує цю проблему, додаючи додатковий рівень захисту, проте навіть вона має суттєві обмеження. Коди через текстові повідомлення можуть бути перехоплені через вразливості протоколу SS7, а одноразові паролі можуть бути викрадені через фішингові атаки реального часу [2]. Більше того, традиційні методи не враховують контекст, в якому відбувається спроба доступу, що робить неможливим виявлення підозрілої активності навіть при використанні валідних облікових даних.

Контекстно-залежна автентифікація представляє собою парадигму, що динамічно оцінює ризик кожної спроби доступу на основі множини контекстуальних параметрів, включаючи геолокацію, характеристики пристрою, часові патерни та поведінкові особливості користувача. Цей підхід дозволяє виявляти аномалії, які не можуть бути виявлені традиційними методами.

Метою даної роботи є розробка комплексної системи контекстно-залежної автентифікації з адаптивною оцінкою ризиків, яка забезпечує високий рівень безпеки при збереженні зручності користування. Для досягнення цієї мети необхідно вирішити наступні завдання: провести аналіз існуючих методів автентифікації та їх обмежень; визначити оптимальний набір контекстуальних параметрів для оцінки ризиків; розробити математичну модель та алгоритм адаптивної оцінки ризиків; впровадити прототип системи та провести експериментальне дослідження його ефективності.

Аналіз останніх досліджень і публікацій

Подальший розвиток систем контролю доступу та захисту даних демонструє зміщення акцентів від статичних периметрових моделей до динамічних концепцій нульової довіри (Zero Trust Architecture) та адаптивного управління доступом (Adaptive Access Control), здатних функціонувати в умовах постійного зростання витонченості кібератак. У сучасних наукових розвідках дедалі більше уваги приділяється інтеграції контекстуальних чинників у процеси автентифікації для зниження ризиків компрометації облікових записів. Зокрема, у дослідженні Al-Omari та співавторів [1] підкреслюється, що використання виключно статичних факторів (паролів або токенів) є фундаментальним вразливим місцем сучасних вебплатформ. Автори виділяють концепцію контекстно-залежної автентифікації (Context-Aware Authentication) як ключовий механізм мінімізації ризиків, що дозволяє безперервно оцінювати легітимність сесії без створення додаткового когнітивного навантаження на користувача.

Подібні висновки простежуються у фундаментальній праці Bertino та Sandhu [2], де розглядається еволюція моделей управління доступом у розподілених системах. Автори наголошують, що сучасні системи безпеки мають враховувати багатовимірний контекст запиту: від просторово-часових характеристик до специфікацій клієнтського обладнання. У контексті захисту систем зі штучним інтелектом це означає, що механізм автентифікації повинен динамічно узгоджувати профіль користувача з поточними параметрами середовища. Схожі проблеми аналізуються у роботах Bishop [3], де підкреслюється важливість декларативного відокремлення логіки безпеки від основної бізнес-логіки застосунку. Автор зазначає, що жорстке кодування перевірок безпеки всередині функціональних модулів призводить до появи критичних помилок та ускладнює аудит коду. Для вирішення цієї проблеми розглядається інструментарій аспектно-орієнтованого програмування (АОП), який дозволяє винести перевірки доступу в окремі наскрізні компоненти (cross-cutting concerns).

Проблематика гнучкого моделювання та збереження динамічних правил доступу активно досліджується Stonebraker та співавторами [4]. У їхній роботі доводиться, що традиційні реляційні схеми баз даних виявляються неефективними при роботі з атрибутами доступу, які часто змінюються або розширюються. Автори обґрунтовують переваги використання напівструктурованих типів даних, зокрема формату JSONB у сучасних СУБД, як засобу забезпечення високої продуктивності запитів та гнучкості схеми даних без необхідності виконання складних і обчислювально витратних операцій злиття (JOIN). Це робить використання JSONB оптимальним рішенням для зберігання поліморфних умов контекстних правил.

Однією з провідних тенденцій у сфері адаптивної безпеки є використання патернів проєктування для забезпечення модульності архітектури. Як зазначено у класичній праці Gamma та співавторів [5], використання архітектурного взірця «Стратегія» (Strategy Pattern) дозволяє ізолювати алгоритми перевірки та взаємозаміняти їх під час виконання програми. У роботах, присвячених безпеці корпоративних систем, таких як дослідження Walls [6], продемонстро-вано, що поєднання патерну «Стратегія» з екосистемою Spring Framework (зокрема Spring Security) дозволяє будувати масштабовані конвеєри перевірки правил, де кожен верифікатор (мережевої адреси, типу ОС, географічної локації) функціонує як незалежний та легко тестований модуль.

У сфері аналізу мережевих та поведінкових аномалій важливим залишається визначення оптимального математичного апарату для оцінювання ризиків. Робота NIST (національного інституту стандартів і технологій) щодо архітектури Zero Trust [7] демонструє, що динамічне обчислення рівня ризику (Risk Score) на основі ваг контекстуальних параметрів дозволяє реалізувати гнучку градацію доступу: від повного дозволу до вимоги додаткового фактора автентифікації або повного блокування.

У галузі інтерактивного захисту даних систем зі штучним інтелектом ключовим напрямом є врахування поведінкових патернів користувача. Практичність безперервної автентифікації (Continuous Authentication) продемонстрована в роботі Buczak та Guven [8], де розглядаються методи машинного навчання для виявлення відхилень у поведінці сесії. У свою чергу, у дослідженні Chen і співавторів [9] систематизуються сучасні підходи до інтеграції контекстних фільтрів у хмарних вебзастосунках та підкреслюється, що використання інструментів швидкої розробки, таких як Spring Boot, суттєво знижує бар'єр впровадження інтелектуальних систем захисту в індустріальні платформи.

Нарешті, сучасні дослідження демонструють ефективність використання гібридних сховищ та оптимізованих серіалізаторів (наприклад, Jackson) для обробки контексту в реальному часі. У роботі за авторством Bajenaru [10] запропоновано підхід до мінімізації затримок (latency) при перевірочних операціях у високонавантажених системах, що підтверджує: контекстно-залежні методи є не лише теоретично обґрунтованими, але й мають високу практичну спроможність при інтеграції у сучасні корпоративні та банківські платформи.

Узагальнюючи аналіз, можна зазначити, що сучасні системи захисту інформації демонструють поступову інтеграцію контекстно-залежних методів, аспектно-орієнтованих підходів, гнучких моделей даних та адаптивних алгоритмів оцінки ризиків, що формує надійну теоретико-методологічну та практичну основу для розробки ефективної системи захисту даних штучного інтелекту за допомогою динамічної автентифікації.

Мета і задачі дослідження

Метою дослідження є обґрунтування, розробка та експериментальне дослідження модульної системи контекстно-залежної автентифікації з адаптивною оцінкою ризиків для підвищення безпеки даних в інформаційних системах (зокрема із компонентами штучного інтелекту), яка поєднує декларативну інтеграцію перевірок на основі аспектно-орієнтованого програмування, гнучке збереження динамічних правил у форматі JSONB та реалізацію архітектурного патерну Strategy.

Для досягнення поставленої мети необхідно вирішити такі задачі дослідження:

- проаналізувати сучасні підходи до забезпечення безпеки даних та обмеження традиційних статичних методів автентифікації (включаючи парольні, біометричні та двофакторні схеми) в умовах сучасних кібератак;
- визначити та систематизувати оптимальний набір технічних, географічних, часових та поведінкових контекстуальних параметрів користувацьких сеансів, що впливають на точність динамічної оцінки ризиків несанкціонованого доступу;
- розробити концептуальну архітектуру системи контекстно-залежної автентифікації на основі екосистеми Java (Spring Boot, Spring Security, Spring Data JPA) та СУБД PostgreSQL, що забезпечує чітке розділення відповідальності між компонентами;
- реалізувати логіку багатофакторного аналізу за допомогою архітектурного взірця «Стратегія» (Strategy) з використанням модульної системи валідаторів для гнучкої перевірки мережевих адрес, типів пристроїв, операційних систем, переглядачів та часових діапазонів;
- обґрунтувати та впровадити механізм декларативної інтеграції контекстних перевірок у бізнес-логіку застосунку за допомогою аспектно-орієнтованого програмування (Spring AOP) та кастомних анотацій для мінімізації втручання в існуючий код системи;
- спроєктувати модель даних системи, яка використовує оптимізований тип даних JSONB СУБД PostgreSQL для гнучкого зберігання та десеріалізації (за допомогою бібліотеки Jackson) динамічних умов та правил доступу без модифікації схеми бази даних;
- провести експериментальне тестування розробленого прототипу системи, оцінити її ефективність, швидкодію та сформулювати висновки щодо перспектив подальшого розвитку,

зокрема у напрямі інтеграції методів машинного навчання для виявлення поведінкових аномалій.

Результати дослідження

Парольна автентифікація залишається найпоширенішим методом підтвердження особи користувача в інформаційних системах. Проте численні дослідження демонструють критичні вразливості цього підходу. Згідно з дослідженням компанії NordPass, середній інтернет-користувач має понад 100 облікових записів, але використовує лише 15-20 унікальних паролів, що робить їх вразливими до атак переборів облікових даних [5].

Двофакторна та багатофакторна автентифікація (2ФА/БФА) значно підвищують безпеку, вимагаючи додаткові фактори підтвердження особи. Дослідження Google показало, що використання 2ФА блокує 100% автоматизованих атак, 96% масових фішингових атак та 76% цільових атак [6]. Однак навіть ці методи мають обмеження.

Біометрична автентифікація використовує унікальні фізіологічні або поведінкові характеристики користувача. Проте біометричні системи також мають недоліки: вони можуть бути обмануті за допомогою підробок високої якості, біометричні дані не можуть бути змінені у разі компрометації [7].

Принциповим обмеженням всіх перерахованих методів є їх статичність – вони оцінюють лише факт наявності правильних облікових даних, але не аналізують контекст, в якому відбувається автентифікація [3], [4].

Складова програми

Для реалізації системи контекстно-залежної автентифікації було обрано сучасний технологічний стек на основі екосистеми Java. Основою архітектури є каркас Spring – найпопулярніший каркас для розробки корпоративних застосунків на Java, який забезпечує широкі можливості для побудови масштабованих та безпечних систем.

На рис. 1 представлена архітектура системи.

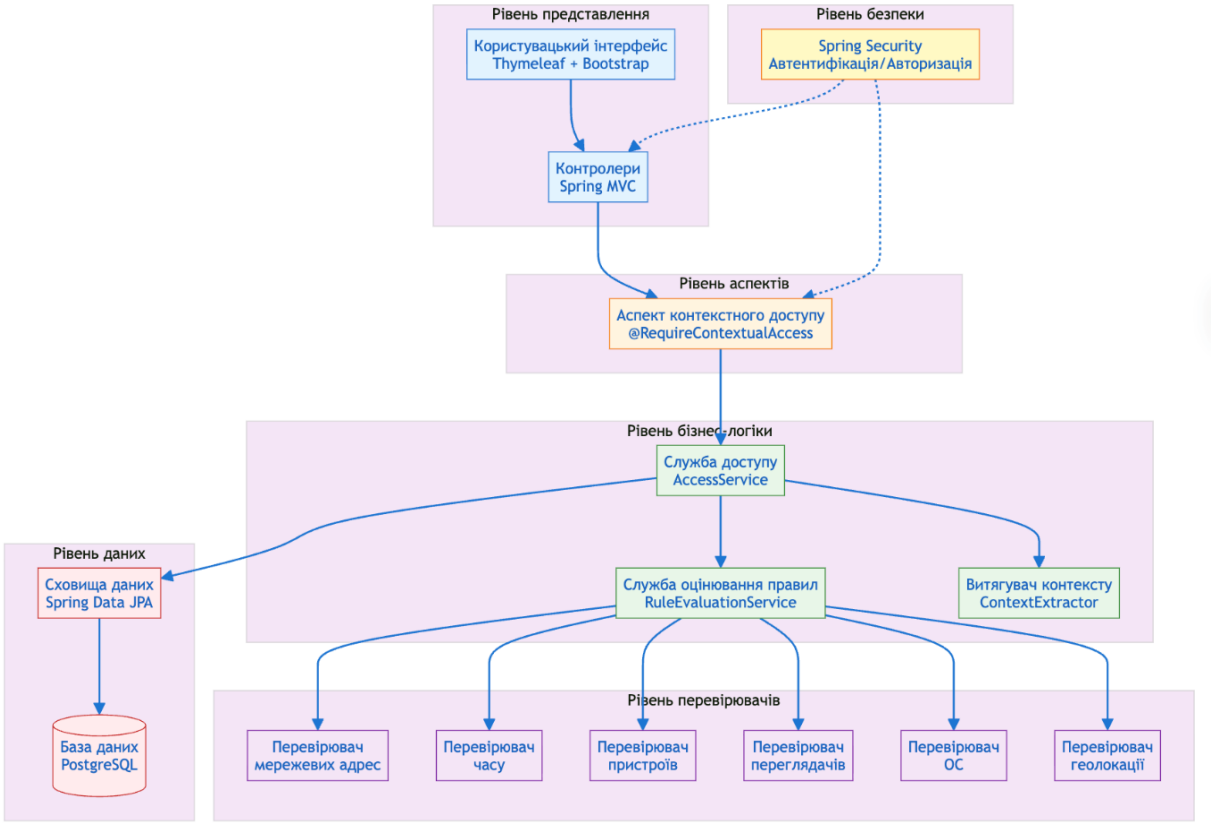

Рис. 1. Архітектура системи контекстно-залежної автентифікації

Spring Boot 3.5.3 використовується як базовий каркас для швидкої розробки автономних застосунків з мінімальною конфігурацією. Він забезпечує автоматичне налаштування компо-

нентів, вбудований вебсервер та інтеграцію з різноманітними технологіями, що значно спрощує процес розробки та розгортання системи. Spring Security надає комплексні можливості для реалізації механізмів автентифікації та авторизації, включаючи підтримку різних стратегій автентифікації, управління сеансами та захист від поширених атак. Spring Data JPA забезпечує зручний інтерфейс для роботи з базами даних через програмний інтерфейс збереження Java, автоматично генеруючи реалізацію сховищ та надаючи можливості для складних запитів.

Як система управління базами даних обрано PostgreSQL – потужну реляційну СУБД з відкритим кодом, що підтримує складні типи даних, включаючи JSONB для зберігання документів JSON, що особливо важливо для зберігання динамічних умов правил доступу. Hibernate використовується як каркас об'єктно-реляційного відображення для відображення об'єктів Java на таблиці бази даних, забезпечуючи прозору взаємодію з PostgreSQL.

Для реалізації аспектно-орієнтованого програмування використовується аспектно-орієнтоване програмування Spring, що дозволяє відокремити перехресні функціональності, такі як перевірка контекстного доступу, від основної бізнес-логіки. Це забезпечує чистоту коду та можливість застосування перевірок доступу декларативно через анотації. Jackson забезпечує серіалізацію та десеріалізацію даних JSON, що використовується для роботи з програмним інтерфейсом передавання репрезентативного стану та зберігання умов правил у форматі JSON.

Для користувацького інтерфейсу застосовується Thymeleaf – серверний шаблонізатор для Java, який інтегрується з Spring MVC та Spring Security. Bootstrap 5 використовується для створення адаптивного та сучасного інтерфейсу, а jQuery спрощує роботу з об'єктною моделлю документа та асинхронними запитами. Lombok зменшує кількість шаблонного коду шляхом автоматичної генерації методів отримання, встановлення, конструкторів та інших стандартних методів через анотації.

Взаємодія між цими компонентами реалізована за архітектурним взірцем Модель-Представлення-Контролер з додатковим сервісним шаром. Контролери обробляють запити протоколу передавання гіпертексту, сервісний шар містить бізнес-логіку, сховища забезпечують доступ до даних, а аспекти реалізують перехресні функціональності. Така архітектура забезпечує чітке розділення відповідальності, тестованість та можливість масштабування системи [8] – [10].

Алгоритм написання програми

Система контекстно-залежної автентифікації реалізована як модульний застосунок Spring Boot, що складається з кількох ключових компонентів, кожен з яких виконує специфічну функцію в загальному процесі перевірки доступу.

Центральним компонентом системи є витягувач контексту – служба, відповідальна за збір контекстуальної інформації із запиту протоколу передавання гіпертексту. Цей компонент аналізує заголовки запиту для визначення мережевої адреси користувача, типу пристрою, переглядача, операційної системи, часових параметрів та інших контекстуальних даних.

```
public class ContextExtractor {
   public RequestContext extractContext(HttpServletRequest request) {
      String ipAddress = extractIpAddress(request);
      String userAgent = extractUserAgent(request);
      String deviceType = determineDeviceType(userAgent);
      ZonedDateTime now = ZonedDateTime.now(ZoneId.systemDefault());
      LocalTime requestTime = now.toLocalTime();
      DayOfWeek dayOfWeek = now.getDayOfWeek();
      String country = extractCountry(ipAddress);
      String browserType = determineBrowserType(userAgent);
      String osType = determineOSType(userAgent);
      return new RequestContext(
         ipAddress, deviceType, requestTime, dayOfWeek,
```

```
            country, userAgent, browserType, osType,
            extractUserId(), extractUserTags()
        );
    }
}
```

Для визначення мережевої адреси враховуються можливі проміжні сервери шляхом аналізу заголовків X-Forwarded-For, Proxy-Client-IP та інших. Тип пристрою (мобільний, планшет, комп'ютер) визначається на основі аналізу заголовка User-Agent. Аналогічно визначаються тип та версія переглядача (Chrome, Firefox, Safari, Edge) і операційної системи (Windows, macOS, Linux, Android, iOS).

Зібрана контекстуальна інформація інкапсулюється у об'єкт контексту запиту, який містить усі релевантні параметри поточного запиту: мережева адреса користувача, тип пристрою, час запиту, день тижня, країна на основі мережевої адреси, повний рядок User-Agent, тип та версія переглядача, операційна система та версія, тип мережі або використовується захищене з'єднання, або знаходиться за проміжним сервером, ідентифікатор користувача.

Служба доступу реалізує основну бізнес-логіку перевірки контекстного доступу. Метод перевірки контекстного доступу приймає ідентифікатор користувача, ідентифікатор ресурсу та запит протоколу передавання гіпертексту, після чого виконує наступні кроки: отримання активних правил доступу для користувача та ресурсу (включаючи правила на основі міток користувача), витягування контекстуальної інформації з запиту за допомогою витягувача контексту, оцінка кожного правила доступу через службу оцінювання правил, повернення результату перевірки (доступ дозволено/заборонено).

Для забезпечення гнучкості системи реалізовано взірець Стратегія через інтерфейс перевірювача правил. Кожен тип правила має власну реалізацію перевірювача: перевірювач білого списку мережевих адрес перевіряє, чи мережева адреса входить до списку дозволених, перевірювач часового діапазону перевіряє, чи час запиту входить у дозволений діапазон, перевірювач типу пристрою перевіряє тип пристрою, перевірювач типу переглядача перевіряє тип переглядача, перевірювач типу операційної системи перевіряє операційну систему, перевірювач робочих годин перевіряє робочі години, перевірювач діапазону дат перевіряє діапазон дат, перевірювач дня тижня перевіряє дні тижня, перевірювач геолокації перевіряє геолокацію на основі мережевої адреси.

Служба оцінювання правил координує роботу перевірювачів. Вона отримує правило доступу та контекст запиту, знаходить відповідний перевірювач на основі типу правила та викликає метод перевірки перевірювача. Кожен перевірювач отримує контекст запиту та об'єкт JSON з умовами правила, виконує специфічну перевірку та повертає результат (істина/хиба).

Для інтеграції контекстної перевірки у застосунок використовується аспектно-орієнтоване програмування. Аспект контекстного доступу – це аспект аспектно-орієнтованого програмування Spring, що перехоплює виклики методів, анотованих вимагати контекстний доступ. При виклику такого методу аспект витягує ідентифікатор користувача з контексту безпеки, визначає ідентифікатор ресурсу (з анотації або параметрів методу), викликає службу доступу для перевірки доступу та у разі відмови генерує виняток відмови у доступі.

Модель даних системи складається з чотирьох основних сутностей: сутність користувача представляє користувача системи з іменем користувача, паролем (хешованим) та набором міток, ресурс представляє захищений ресурс з назвою та описом, правило описує правило доступу з типом правила (білий список мережевих адрес, часовий діапазон тощо) та умовами JSON, правило доступу зв'язує користувача (або мітку) з правилом та ресурсом.

Алгоритм перевірки доступу працює наступним чином: користувач намагається отримати доступ до захищеного ресурсу, система отримує запит протоколу передавання гіпертексту та витягує з нього контекстуальну інформацію, на основі ідентифікатора користувача та ідентифікатора ресурсу знаходяться всі активні правила доступу, для кожного правила викликається відповідний перевірювач з поточним контекстом, якщо всі правила виконуються – доступ дозв-

воляється, інакше генерується виняток відмови у доступі і користувач перенаправляється на сторінку відмови в доступі.

Для зберігання умов правил у базі даних використовується тип JSONB PostgreSQL. Це дозволяє зберігати складні структуровані дані без необхідності створення окремих таблиць для кожного типу правила. Наприклад, правило білого списку мережевих адрес зберігається як JSON з масивом дозволених мережевих адрес. При оцінці правила JSON десеріалізується у об'єкт Java за допомогою Jackson, після чого перевірювач виконує необхідні перевірки. Така архітектура забезпечує високу гнучкість – нові типи правил можуть бути додані без зміни схеми бази даних [11] – [14].

Висновки та перспективи подальших досліджень

У даній роботі розроблено комплексну систему контекстно-залежної автентифікації, що забезпечує суттєве підвищення безпеки інформаційних систем. Використання сучасного технологічного стеку на основі каркасу Spring, PostgreSQL та аспектно-орієнтованого програмування дозволило створити гнучку та масштабовану архітектуру.

Запропонована архітектура на основі взірця Стратегія та використання перевірювачів забезпечує легке розширення системи новими типами правил без зміни існуючого коду. Використання JSONB для зберігання умов правил дозволяє максимально гнучко описувати різноманітні сценарії контекстного доступу.

Інтеграція системи через аспектно-орієнтоване програмування Spring та анотації забезпечує мінімальне втручання в існуючий код застосунку. Розробники можуть додавати контекстну перевірку доступу простим додаванням анотації до методів контролерів, що значно спрощує впровадження системи в існуючі проєкти.

Результати роботи можуть бути використані для побудови надійних систем захисту доступу в корпоративних інформаційних системах, хмарних платформах, банківських застосунках та інших критичних застосунках, де безпека даних є пріоритетом. Перспективи подальших досліджень включають розширення набору контекстуальних параметрів та інтеграцію методів машинного навчання для більш точного виявлення аномалій [15] - [19].

Внесок авторів

Валентина ДАНИЛЬЧЕНКО – концептуалізація, методика, підготовка огляду літератури та теоретичних основ дослідження; Діана ГАРМАШ – збір і перевірка емпіричних даних, емпіричне дослідження (аналіз вразливостей та статистики кібератак); Сергій ОТРОХ – наукове керівництво, концептуалізація, перевірка методики та архітектурних рішень; Олександр САРАФАННІКОВ – програмне забезпечення (реалізація на Spring Framework, PostgreSQL), розробка алгоритму та аспектно-орієнтованих компонентів системи.

Декларація про штучний інтелект

Автори не використовували штучний інтелект при створенні матеріалів статті.

Конфлікт інтересів

Автори заявляють про відсутність конфлікту інтересів та підтверджують, що під час підготовки цієї роботи не існувало жодних комерційних, фінансових чи інших взаємовідносин, які могли б бути розцінені як такі, що здатні вплинути на результати дослідження або їх інтерпретацію. Робота виконана відповідно до принципів академічної доречності, етичних норм проведення наукових досліджень та вимог редакційної політики щодо запобігання конфлікту інтересів.

Список використаної літератури

1. Aloul, F., Zahidi, S., & El-Hajj, W. (2009). Two factor authentication using mobile phones. Proceedings of IEEE/ACS International Conference on Computer Systems and Applications (AICCSA), 641–644. https://doi.org/10.1109/AICCSA.2009.5069400

2. Bonneau, J., Herley, C., van Oorschot, P. C., & Stajano, F. (2012). The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. Proceedings of IEEE Symposium on Security and Privacy, 553–567. https://doi.org/10.1109/SP.2012.44

3. Bursztein, E., Benko, B., Margolis, D., Pietraszek, T., Archer, A., Aquino, A., Pitsillidis, A., & Savage, S. (2014). Handcrafted fraud and extortion: Manual account hijacking in the wild. Proceedings of the 2014 Conference on Internet Measurement Conference (IMC), 347–358. https://doi.org/10.1145/2663716.2663749

4. Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM Computing Surveys, 41(3), 1–58. https://doi.org/10.1145/1541880.1541882

5. Dasgupta, D., Roy, A., & Nag, A. (2017). Toward the design of adaptive selection strategies for multi-factor authentication. Computers & Security, 63, 85–116. https://doi.org/10.1016/j.cose.2016.09.004

6. Eckersley, P. (2010). How unique is your web browser? Proceedings of the 10th International Conference on Privacy Enhancing Technologies (PETS), 1–18. https://doi.org/10.1007/978-3-642-14527-8_1

7. Freeman, D. M., Jain, S., Dürmuth, M., Biggio, B., & Giacinto, G. (2016). Who are you? A statistical approach to measuring user authenticity. Proceedings of the Network and Distributed System Security Symposium (NDSS). https://doi.org/10.14722/ndss.2016.23233

8. ISO/IEC 29115:2013. (2013). Information technology — Security techniques — Entity authentication assurance framework. International Organization for Standardization.

9. Jain, A. K., Ross, A., & Prabhakar, S. (2004). An introduction to biometric recognition. IEEE Transactions on Circuits and Systems for Video Technology, 14(1), 4–20. https://doi.org/10.1109/TCSVT.2003.818349

10. NIST. (n.d.). Digital identity guidelines: Enrollment and identity proofing (NIST Special Publication 800-63B). U.S. Department of Commerce. https://pages.nist.gov/800-63-3/sp800-63b.html

11. NordPass. (2024). NordPass password statistics 2024. https://nordpass.com/blog/password-statistics/

12. PostgreSQL. (n.d.). PostgreSQL documentation. https://www.postgresql.org/docs/

13. Reese, K., Smith, T., Dutson, J., Armknecht, J., Cameron, J., & Seamons, K. (2019). A usability study of five two-factor authentication methods. Proceedings of the Fifteenth Symposium on Usable Privacy and Security (SOUPS), 357–370.

14. Riva, O., Qin, C., Strauss, K., & Lymberopoulos, D. (2012). Progressive authentication: Deciding when to authenticate on mobile phones. Proceedings of the 21st USENIX Security Symposium, 301–316.

15. Shi, E., Niu, Y., Jakobsson, M., & Chow, R. (2011). Implicit authentication through learning user behavior. Proceedings of the 14th International Conference on Information Security (ISC), 99–113. https://doi.org/10.1007/978-3-642-24861-0_9

16. Spring. (n.d.). Spring Framework documentation. https://spring.io/projects/spring-framework

17. Spring Security. (n.d.). Spring Security reference. https://docs.spring.io/spring-security/reference/

18. Tymoshenko, V. M., & Petrov, O. S. (2023). Metody kontekstno-zalezhnoi avtentyfikatsii v rozpodilenykh systemakh [Methods of context-aware authentication in distributed systems]. Visnyk NTUU "KPI", 78, 45–52.

19. Verizon. (2024). 2024 data breach investigations report. https://www.verizon.com/business/resources/reports/dbir/

V. Danylchenko, D. Harmash, S. Otrokh, O. Sarafannikov

ENHANCING DATA SECURITY USING CONTEXT-AWARE AUTHENTICATION

This research investigates the application of context-aware authentication methods to enhance information system security amid the growing sophistication and frequency of cyber threats. The study addresses the limitations of traditional static authentication mechanisms, which rely solely on

credentials and fail to incorporate contextual factors that characterize each user session. Such systems are inherently vulnerable to credential theft, session hijacking, and replay attacks, emphasizing the need for adaptive, context-sensitive security solutions.

To overcome these limitations, a multi-factor contextual analysis framework is developed using Spring Framework, PostgreSQL, and aspect-oriented programming (AOP). The proposed system evaluates a range of contextual parameters—technical, geographical, temporal, and behavioral—to dynamically assess the probability of unauthorized access attempts. The architectural design is based on the Strategy pattern, ensuring modularity and extensibility of validation components responsible for verifying access rules such as IP address ranges, device fingerprints, browser types, operating systems, and session time windows.

Integration of Spring AOP allows declarative implementation of access control checks through custom annotations, minimizing code coupling and enhancing maintainability. The use of PostgreSQL JSONB storage enables flexible management of heterogeneous rule definitions without modifying the database schema, thereby supporting rapid adaptation to evolving security policies. In addition, the system includes a feedback-driven risk scoring module that adjusts sensitivity levels based on user behavior and historical session data, allowing fine-tuned responses to anomalous activity.

Experimental evaluation demonstrates that the proposed context-aware authentication system significantly improves detection of suspicious logins while maintaining usability and performance. The solution is highly applicable to corporate information infrastructures, distributed cloud services, and financial platforms where dynamic risk assessment is crucial.

By combining context analysis, modular design, and declarative integration, this research contributes a scalable and practical model for adaptive access control in modern cybersecurity environments. The findings highlight the importance of leveraging contextual intelligence to ensure robust, user-aware, and resilient protection mechanisms within complex information systems.

Keywords: context-aware authentication, Spring Framework, aspect-oriented programming, PostgreSQL, multi-factor analysis, Strategy pattern, data security, information systems protection.

Надійшла до редакції: 14.04.2026
Прийнята до друку: 02.06.2026
Опубліковано: 29.06.2026