Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

Інтелектуальна комп'ютерна система автоматичного виявлення вразливостей вебзастосунків та класифікації загроз
Назва теми

Рівень вищої освіти *другий (магістерський)*

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Шифр КвРКІ 240246.24.02.10 ПЗ

Виконав здобувач IV курсу, група КІ2м-24-2 _____ Дмитро МИКУЛЯК
Підпис Ініціали, прізвище

Керівник д.техн.наук, професор _____ Олег САВЕНКО
Науковий ступінь, учене звання Підпис Ініціали, прізвище

Нормоконтролер канд. фіз.-мат. наук, доцент _____ Тетяна КИСІЛЬ
Науковий ступінь, учене звання Підпис Ініціали, прізвище

До захисту допускаю:
завідувач кафедри КІІС _____ Ольга ПАВЛОВА
«01» травня 2026 р. підпис Ініціали, прізвище
дата

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Рівень вищої освіти ДРУГИЙ (МАГІСТЕРСЬКИЙ)

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ
Завідувачка кафедри КІІС
Ольга ПАВЛОВА

" 12 " 01 2026 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Микуляку Дмитру Анатолійовичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Інтелектуальна комп'ютерна система автоматичного виявлення вразливостей вебзастосунків та класифікації загроз
Керівник проекту (роботи) Савенко Олег Станіславович, д.т.н., професор.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 12.01.2026 р. № 6

2. Термін подання здобувачем роботи на кафедру 01.05.2026 р.

3. Вихідні дані до роботи Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Аналіз сучасних підходів до виявлення вразливостей вебзастосунків, класифікації кіберзагроз, стандартів безпеки та методів застосування штучного інтелекту в кібербезпеці.

Розроблення архітектури, моделі та методу інтелектуальної комп'ютерної системи автоматичного виявлення вразливостей вебзастосунків і класифікації загроз.

Побудова алгоритмічного забезпечення системи, зокрема алгоритму збору, нормалізації, класифікації, оцінювання ризику та ранжування загроз.

Програмна реалізацію прототипу системи, проведення експериментальної верифікації та оцінка ефективність запропонованих рішень.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтролер	Тетяна КИСІЛЬ, доцент каф. КІІС		
Антиплагіат	Андрій Нічепорук, доцент каф. КІІС		

7. Дата видачі завдання «_12_»_____01_____2026 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) кваліфікаційної роботи магістра	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики КвРМ з керівником	12.01.2026	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	12.01.2026	виконано
3	Робота над розділом 1 – аналіз відомих моделей, методів за темою; постановка задачі	20.01.2026	виконано
4	Робота над розділом 2 – розробка моделей для вирішення поставленої задачі	01.02.2026	виконано
5	Робота над науковою статтею	01.03.2026	виконано
6	Робота над розділом 3 – розробка методів для вирішення поставленої задачі	15.03.2026	виконано
7	Робота над розділом 4 – проектування та розробка ПЗ для вирішення поставленої задачі, експериментальна частина	01.04.2026	виконано
8	Оформлення пояснювальної записки згідно вимог	18.04.2026	виконано
9	Попередній захист ДРМ	29.04.2026	виконано
10	Захист ДРМ на засіданні ЕК	До 15.05.2026	виконано

Здобувач_____Дмитро МИКУЛЯК__
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи_____Олег САВЕНКО___
Підпис Ім'я, ПРІЗВИЩЕ

РЕФЕРАТ

Тема кваліфікаційної роботи магістра: Інтелектуальна комп'ютерна система автоматичного виявлення вразливостей вебзастосунків та класифікації загроз

Автор роботи: Микуляк Дмитро Анатолійович.

Керівник роботи: Савенко Олег Станіславович.

Пояснювальна записка: 118 с., 19 рис., 22 табл., 4 дод., 83 джерел.

ВЕБЗАСТОСУНОК, ВРАЗЛИВІСТЬ, ІНФОРМАЦІЙНА БЕЗПЕКА, КІБЕРЗАГРОЗА, КЛАСИФІКАЦІЯ ЗАГРОЗ, МАШИННЕ НАВЧАННЯ, ОЦІНКА РИЗИКУ, ШТУЧНИЙ ІНТЕЛЕКТ.

Об’єктом дослідження є процес автоматизованого забезпечення інформаційної безпеки вебзастосунків у сучасних інформаційно-телекомунікаційних системах.

Предметом дослідження є моделі, методи, алгоритми та програмно-архітектурні засоби автоматичного виявлення вразливостей вебзастосунків, інтелектуальної нормалізації результатів сканування, класифікації кіберзагроз і ризик-орієнтованого ранжування вразливостей.

Метою кваліфікаційної роботи магістра є розроблення інтелектуальної комп’ютерної системи автоматичного виявлення вразливостей вебзастосунків та класифікації загроз, яка забезпечує приймання результатів багатоканального аналізу безпеки, їх нормалізацію, класифікацію, оцінювання ризику, ранжування загроз і формування аналітичних звітів.

Для розв’язання поставлених задач використано методи системного аналізу, структурного та об’єктно-орієнтованого проєктування, тестування безпеки вебзастосунків, аналізу складу програмного забезпечення, машинного навчання, математичного моделювання, статистичного оцінювання, ризик-орієнтованого аналізу та програмної інженерії.

Наукова новизна одержаних результатів полягає в такому:

– удосконалено архітектурно-функціональну модель інтелектуальної комп’ютерної системи автоматичного виявлення вразливостей вебзастосунків, яка

поєднує багатоканальний збір даних, нормалізацію, семантичне збагачення, класифікацію, кореляцію та ризик-орієнтоване ранжування;

– удосконалено модель ризик-орієнтованого оцінювання вразливостей шляхом урахування CVSS, контекстних параметрів середовища, впевненості класифікатора, рівня міжджерельного підтвердження, клас-специфічних ваг загроз і епістемічної невизначеності;

– набули подальшого розвитку методи автоматичної класифікації безпекових знахідок на основі гібридного контуру текстових, контекстних, джерельних, графових і трансформерно-семантичних ознак.

На основі проведених досліджень розроблено архітектуру, алгоритмічне забезпечення та програмний прототип ІКС АВВЗ, який забезпечує приймання результатів із різних джерел аналізу безпеки, їх інтелектуальну нормалізацію, класифікацію, ризик-орієнтоване ранжування, збереження результатів, журналювання, формування звітів і доступ до основних функцій системи через REST API.

Практична значимість отриманих результатів полягає у можливості використання розробленого прототипу в процесах безпечної розробки програмного забезпечення, інтеграції безпекового аналізу в життєвий цикл розроблення програмних систем, автоматизованому аналізі результатів сканування та пріоритезації усунення вразливостей.

Результати роботи апробовано у науковій публікації автора [1] та використано під час виконання науково-дослідної практики.

У першому розділі проведено аналіз відомих моделей, методів і засобів виявлення вразливостей вебзастосунків та класифікації загроз.

У другому розділі розроблено архітектурну модель, модель нормалізації, модель класифікації та модель адаптивного оцінювання ризику.

У третьому розділі розроблено алгоритми функціонування системи.

У четвертому розділі наведено програмну реалізацію прототипу, результати експериментальної перевірки та перспективи практичного застосування системи.

У висновках узагальнено основні результати дослідження та ступінь досягнення поставленої мети.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ІКС АВВЗ – інтелектуальна комп’ютерна система автоматичного виявлення вразливостей вебзастосунків та класифікації загроз

AI – Artificial Intelligence (штучний інтелект)

LLM – Large Language Model (велика мовна модель)

ML – Machine Learning (машинне навчання)

SAST – Static Application Security Testing (статичне тестування безпеки застосунків)

DAST – Dynamic Application Security Testing (динамічне тестування безпеки застосунків)

IAST – Interactive Application Security Testing (інтерактивне тестування безпеки застосунків)

SCA – Software Composition Analysis (аналіз складу програмного забезпечення)

OWASP – Open Web Application Security Project (відкритий проєкт безпеки вебзастосунків)

CVE – Common Vulnerabilities and Exposures (каталог відомих вразливостей та експозицій)

CWE – Common Weakness Enumeration (перелік типових слабких місць програмного забезпечення)

CAPEC – Common Attack Pattern Enumeration and Classification (перелік і класифікація типових шаблонів атак)

CVSS – Common Vulnerability Scoring System (загальна система оцінювання критичності вразливостей)

NVD – National Vulnerability Database (національна база даних вразливостей)

CI/CD – Continuous Integration / Continuous Deployment (безперервна інтеграція та безперервне розгортання)

API – Application Programming Interface (прикладний програмний інтерфейс)

AST – Abstract Syntax Tree (абстрактне синтаксичне дерево)

CFG – Control Flow Graph (граф потоку керування)

GNN – Graph Neural Network (графова нейронна мережа)

TF-IDF – Term Frequency-Inverse Document Frequency (частота терміна – обернена частота документа)

DevSecOps – Development, Security and Operations (інтеграція розроблення, безпеки та експлуатації програмного забезпечення)

OOD – Out-of-Distribution (дані поза навчальним розподілом)

WAF – Web Application Firewall (міжмережевий екран вебзастосунків)

JSON – JavaScript Object Notation (текстовий формат обміну структурованими даними JavaScript Object Notation)

XML – Extensible Markup Language (розширювана мова розмітки)

CSV – Comma-Separated Values (значення, розділені комами)

REST – Representational State Transfer (архітектурний стиль передавання стану представлення)

HTTP – HyperText Transfer Protocol (протокол передавання гіпертексту)

SPA – Single Page Application (односторінковий вебзастосунок)

SQL – Structured Query Language (структурована мова запитів)

XSS – Cross-Site Scripting (міжсайтове виконання сценаріїв)

BOLA – Broken Object Level Authorization (порушення авторизації на рівні об’єктів)

IDOR – Insecure Direct Object Reference (небезпечне пряме посилання на об’єкт)

URL – Uniform Resource Locator (уніфікований покажчик ресурсу)

HTML – HyperText Markup Language (мова гіпертекстової розмітки)

SBOM – Software Bill of Materials (специфікація складу програмного забезпечення)

IaC – Infrastructure as Code (інфраструктура як код)

JWT – JSON Web Token (вебтокен JSON)

MFA – Multi-Factor Authentication (багатофакторна автентифікація)

TLS – Transport Layer Security (протокол захисту транспортного рівня)

BERT – Bidirectional Encoder Representations from Transformers (двонапрямні кодувальні представлення на основі трансформерів)

CNN – Convolutional Neural Network (згорткова нейронна мережа)

RNN – Recurrent Neural Network (рекурентна нейронна мережа)

LSTM – Long Short-Term Memory (довга короткочасна пам’ять)

API Gateway – Application Programming Interface Gateway (шлюз прикладного програмного інтерфейсу)

ВСТУП

Сучасний етап розвитку інформаційних технологій характеризується широким використанням вебзастосунків у фінансовому секторі, електронній комерції, державному управлінні, медицині, освіті, корпоративних інформаційних системах та хмарних сервісах. Вебзастосунки перестали бути лише допоміжними інтерфейсами взаємодії з користувачем і стали важливими компонентами інформаційно-телекомунікаційної інфраструктури, що забезпечують оброблення, передавання та зберігання критично важливих даних. Унаслідок цього рівень їх захищеності безпосередньо впливає на надійність програмних систем, безпеку даних користувачів і стійкість цифрової інфраструктури загалом [1, 2, 3, 4, 5, 6, 7, 8, 9].

Ускладнення архітектури сучасних вебзастосунків суттєво змінює характер задач інформаційної безпеки. Якщо раніше значна частина вебсистем будувалася як відносно монолітні програмні рішення, то нині активно використовуються мікросервісні архітектури, API-центричні моделі взаємодії, контейнеризація, хмарні платформи, CI/CD-конвеєри, сторонні бібліотеки та програмні залежності. Такий підхід забезпечує масштабованість, модульність і швидкість розроблення, однак одночасно збільшує площину атаки та кількість потенційних джерел вразливостей. Джерелом ризику може бути не лише помилка у власному коді, а й некоректна конфігурація середовища, вразлива залежність, контейнерний образ, зовнішній API, помилка автентифікації, недостатній контроль доступу або дефект у ланцюгу постачання програмного забезпечення [2; 3; 8; 23 – 27].

Для систематизації вразливостей і загроз у світовій практиці використовуються стандарти, бази знань і класифікації OWASP, CVE, CWE, CAPEC, CVSS, NVD, MITRE ATT&CK та SBOM. Вони дають змогу уніфікувати опис дефектів, оцінювати їх критичність, зіставляти знайдені проблеми з відомими вразливостями та визначати можливі сценарії експлуатації [9; 17 – 22; 28; 29]. Разом із тим наявність таких джерел знань сама по собі не вирішує проблему автоматичного аналізу, оскільки результати різних сканерів часто мають неоднорідний формат, дублюються, містять хибні спрацювання та потребують

контекстної інтерпретації.

Традиційні інструментальні підходи до виявлення вразливостей, зокрема SAST, DAST, IAST та SCA, є важливими складовими безпечної розробки програмного забезпечення. SAST дозволяє аналізувати вихідний код без запуску програми, DAST досліджує поведінку запущеного вебзастосунку, IAST поєднує runtime-контекст із внутрішнім аналізом, а SCA забезпечує аналіз складу програмного забезпечення, сторонніх бібліотек, залежностей і SBOM [12; 30 – 37]. Однак ізольоване використання цих підходів не забезпечує достатньої повноти, точності та практичної придатності результатів. У реальних умовах фахівець отримує велику кількість технічних повідомлень, серед яких значну частину становлять дублікати, неповні записи або хибнопозитивні спрацювання.

У зв’язку з цим актуальним напрямом є розроблення інтелектуальних комп’ютерних систем, здатних не лише збирати результати аналізу безпеки, а й виконувати їх нормалізацію, семантичне збагачення, класифікацію, кореляцію, ризик-орієнтоване ранжування та формування аналітичних звітів. Методи штучного інтелекту, машинного навчання, глибинного навчання, графового аналізу та великих мовних моделей створюють передумови для переходу від фрагментарного сигнатурного аналізу до контекстної й адаптивної інтерпретації загроз [4; 38 – 58]. Особливого значення набуває також оцінювання невизначеності результатів класифікації, оскільки в задачах кібербезпеки помилкова впевненість моделі може призвести до неправильного визначення пріоритету вразливості або пропуску критичної загрози [63 – 65].

Актуальність кваліфікаційної роботи полягає в необхідності розроблення інтелектуальної комп’ютерної системи автоматичного виявлення вразливостей вебзастосунків та класифікації загроз, яка забезпечує інтеграцію результатів багатоканального аналізу безпеки, їх інтелектуальну нормалізацію, класифікацію, оцінювання ризику з урахуванням невизначеності та формування ранжованого переліку загроз для практичного реагування.

Метою кваліфікаційної роботи магістра є розроблення інтелектуальної комп’ютерної системи автоматичного виявлення вразливостей вебзастосунків та

класифікації загроз, яка забезпечує приймання результатів багатоканального аналізу безпеки, їх нормалізацію, класифікацію, оцінювання ризику, ранжування загроз і формування аналітичних звітів.

Поставлена мета досягається розв’язанням таких основних завдань:

– проаналізувати сучасний стан проблеми автоматичного виявлення вразливостей вебзастосунків і класифікації загроз;

– дослідити стандарти, бази знань і шкали оцінювання вразливостей, зокрема OWASP, CVE, CWE, CAPEC, CVSS, NVD, MITRE ATT&CK та SBOM;

– виконати порівняльний аналіз інструментальних методів виявлення вразливостей вебзастосунків, зокрема SAST, DAST, IAST та SCA;

– обґрунтувати доцільність використання методів штучного інтелекту, машинного навчання, глибинного навчання, графового аналізу та великих мовних моделей у задачах класифікації кіберзагроз;

– розробити архітектурно-функціональну модель інтелектуальної комп’ютерної системи автоматичного виявлення вразливостей вебзастосунків та класифікації загроз;

– розробити модель інтелектуальної нормалізації результатів сканування, яка забезпечує уніфікацію, дедуплікацію та семантичне збагачення різнорідних безпекових знахідок;

– розробити модель інтелектуальної класифікації вразливостей і кіберзагроз на основі текстових, структурних, контекстних, джерельних, графових і трансформерно-семантичних ознак;

– удосконалити модель ризик-орієнтованого оцінювання вразливостей шляхом урахування CVSS, контекстних параметрів середовища, впевненості класифікатора, міжджерельного підтвердження та епістемічної невизначеності;

– розробити алгоритми багатоканального збору, нормалізації, класифікації та ризик-орієнтованого ранжування загроз;

– реалізувати програмний прототип ІКС АВВЗ з REST API, журналюванням, зберіганням результатів і формуванням звітів;

– провести експериментальну перевірку ефективності запропонованих

моделей, алгоритмів і програмного прототипу;

– оцінити практичну придатність розробленої системи для використання в процесах безпечної розробки програмного забезпечення та інтеграції безпекового аналізу в життєвий цикл розроблення програмних систем.

Об’єктом дослідження є процес автоматизованого забезпечення інформаційної безпеки вебзастосунків у сучасних інформаційно-телекомунікаційних системах.

Предметом дослідження є моделі, методи, алгоритми та програмно-архітектурні засоби автоматичного виявлення вразливостей вебзастосунків, інтелектуальної нормалізації результатів сканування, класифікації кіберзагроз і ризик-орієнтованого ранжування вразливостей.

Для розв’язання поставлених задач використано методи системного аналізу, структурного та об’єктно-орієнтованого проєктування, статичного, динамічного та інтерактивного тестування безпеки, аналізу складу програмного забезпечення, машинного навчання, глибинного навчання, семантичного аналізу, математичного моделювання, статистичного оцінювання, ризик-орієнтованого аналізу та програмної інженерії.

Наукова новизна одержаних результатів полягає в такому:

– удосконалено архітектурно-функціональну модель інтелектуальної комп’ютерної системи автоматичного виявлення вразливостей вебзастосунків та класифікації загроз, яка, на відміну від ізольованих засобів аналізу безпеки, поєднує багатоканальний збір даних, інтелектуальну нормалізацію, семантичне збагачення, класифікацію, кореляцію та ризик-орієнтоване ранжування в єдиному аналітичному контурі;

– удосконалено модель ризик-орієнтованого оцінювання вразливостей вебзастосунків шляхом побудови багатофакторної функції, яка враховує CVSS, контекстні параметри середовища, бізнес-критичність компонента, експлуатованість, рівень міжджерельного підтвердження, чутливість даних, компенсуючі механізми захисту, впевненість класифікатора та епістемічну невизначеність;

– удосконалено алгоритм інтелектуальної нормалізації безпекових знахідок, у якому різнорідні результати SAST, DAST, IAST, SCA та інших джерел приводяться до єдиного стандартизованого простору ознак із можливістю дедуплікації, семантичного зіставлення з базами знань і подальшої машинної обробки;

– набули подальшого розвитку методи автоматичної класифікації безпекових знахідок на основі гібридного класифікаційного контуру, що поєднує текстові, структурні, контекстні, джерельні, графові та трансформерно-семантичні ознаки;

– уперше в межах програмного прототипу реалізовано практичний підхід до епістемічно-алеаторної декомпозиції оцінки ризику, у якому нормована ентропія Шеннона використовується як явний епістемічний компонент композитного ризику та забезпечує інтервальне подання ризикової оцінки.

На основі проведених досліджень розроблено архітектуру, алгоритмічне забезпечення та програмний прототип ІКС АВВЗ. Прототип забезпечує приймання результатів із різних джерел аналізу безпеки, їх інтелектуальну нормалізацію, семантичне збагачення, класифікацію, ризик-орієнтоване ранжування, збереження результатів, журналювання, формування звітів і доступ до основних функцій системи через REST API.

Практична значимість отриманих результатів полягає у можливості використання розробленого прототипу для автоматизації аналізу безпеки вебзастосунків, інтеграції результатів різних інструментів сканування, зменшення кількості дубльованих і хибнопозитивних повідомлень, підвищення точності визначення критичних загроз, підтримки прийняття рішень щодо пріоритезації усунення вразливостей та інтеграції безпекового аналізу в життєвий цикл розроблення програмних систем.

Експериментальна перевірка програмного прототипу підтвердила працездатність запропонованих моделей і алгоритмів. За результатами тестування досягнуто точність класифікації 0,8718, прецизійність 0,8813, повноту 0,8718, F1-міру 0,8695; середнє значення F1-міри за результатами п’ятиблокової крос-

валідації становить 0,9475 ± 0,0369. Працездатність підходу до врахування невизначеності підтверджено зростанням значення нормованої ентропії Шеннона для нетипових випадків: $\widetilde{H}_{ID} = 0{,}760$, $\widetilde{H}_{OOD} = 0{,}877$, приріст становить $+0{,}118$.

За темою кваліфікаційної роботи опубліковано одну наукову публікацію [1]. Результати роботи також використано під час виконання науково-дослідної практики.

Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, переліку джерел посилань і додатків.

У вступі обґрунтовано актуальність теми, визначено об’єкт, предмет, мету, завдання, методи дослідження, наукову новизну та практичне значення роботи.

У першому розділі проведено аналіз сучасних підходів до виявлення вразливостей вебзастосунків, класифікації кіберзагроз, стандартів безпеки та методів застосування штучного інтелекту в кібербезпеці.

У другому розділі розроблено архітектуру, модель та метод інтелектуальної комп’ютерної системи автоматичного виявлення вразливостей вебзастосунків і класифікації загроз.

У третьому розділі розроблено алгоритмічне забезпечення системи, зокрема алгоритми збору, нормалізації, класифікації, оцінювання ризику та ранжування загроз.

У четвертому розділі здійснено програмну реалізацію прототипу системи, проведено експериментальну верифікацію та оцінено ефективність запропонованих рішень.

У висновках узагальнено основні результати роботи, підтверджено досягнення поставленої мети та визначено напрями подальшого розвитку системи.

1 АНАЛІЗ ВІДОМИХ МОДЕЛЕЙ, МЕТОДІВ ТА ЗАСОБІВ У ГАЛУЗІ АВТОМАТИЧНОГО ВИЯВЛЕННЯ ВРАЗЛИВОСТЕЙ ВЕБЗАСТОСУНКІВ ТА КЛАСИФІКАЦІЇ ЗАГРОЗ

У першому розділі проведено аналіз сучасних моделей, методів і засобів автоматичного виявлення вразливостей вебзастосунків та класифікації загроз. Розглянуто тенденції розвитку вебтехнологій, зміну профілю загроз унаслідок переходу до API-центричних, мікросервісних, контейнеризованих і хмарних архітектур, досліджено роль стандартів і баз знань OWASP, CVE, CWE, CAPEC, CVSS, а також виконано порівняльний аналіз інструментальних методів аналізу захищеності.

Окрему увагу приділено обґрунтуванню доцільності використання методів штучного інтелекту для підвищення ефективності виявлення, класифікації та ризик-орієнтованого оцінювання вразливостей. Показано, що сучасні засоби безпекового аналізу вже не можуть обмежуватися лише сигнатурним пошуком або ізольованою перевіркою коду, оскільки актуальні загрози дедалі частіше виникають на перетині програмної логіки, архітектури, конфігурації, ланцюга постачання програмного забезпечення та середовища експлуатації.

1.1 Обґрунтування актуальності дослідження та аналіз тенденцій розвитку вебтехнологій

Сучасний етап розвитку інформаційних технологій характеризується глибокою інтеграцією вебзастосунків у всі ключові сфери функціонування цифрового суспільства. Вебсистеми сьогодні є основою електронної комерції, цифрового банкінгу, державних сервісів, телемедицини, освітніх платформ, корпоративних інформаційних середовищ, хмарних сервісів та інфраструктурних платформ. Якщо на початкових етапах розвитку мережі Інтернет вебсередовище виконувало переважно інформаційно-довідкові функції, то нині вебзастосунки перетворилися на повноцінні платформи оброблення, зберігання та передавання

критично важливої інформації.

Паралельно зі зростанням ролі вебзастосунків відбувається істотне ускладнення їхньої архітектури. Від монолітних рішень, побудованих навколо централізованої серверної логіки, сучасні системи перейшли до розподілених моделей, у яких широко використовуються мікросервіси, API, контейнери, брокери повідомлень, зовнішні сервіси автентифікації, хмарні обчислення та безсерверні функції. Така еволюція забезпечує високу масштабованість, модульність, адаптивність та швидкість розробки, але одночасно формує нові поверхні атаки та збільшує кількість точок потенційного компрометування.

Відповідно до рекомендацій NIST щодо захисту мікросервісних застосунків, перехід до мікросервісної моделі супроводжується не лише підвищенням масштабованості та гнучкості розробки, а й збільшенням складності безпекового контролю, оскільки взаємодія між сервісами здійснюється через мережеві інтерфейси, API-виклики та проміжні інфраструктурні компоненти. У таких умовах особливого значення набувають API gateway, service mesh, централізоване керування ідентичністю, автентифікацією, авторизацією, моніторингом і політиками міжсервісної взаємодії [7,70]. Це підтверджує, що безпека сучасного вебзастосунку має розглядатися не як властивість окремого програмного модуля, а як системна характеристика всієї розподіленої архітектури.

У сучасному вебзастосунку джерелом загроз може виступати не лише власний програмний код, а й сторонні бібліотеки, фреймворки, контейнерні образи, конфігурації інфраструктури, механізми оркестрації, зовнішні API, секрети доступу, CI/CD-конвеєри та сервіси автоматичного розгортання. Унаслідок цього безпека перестає бути властивістю окремого шару застосунку. Вона набуває системного характеру і залежить від коректності взаємодії всіх компонентів, їх конфігурації, життєвого циклу залежностей та режимів експлуатації [3].

Окремої уваги потребує розвиток API-центричних архітектур. У сучасних вебсистемах API є не лише технічним інтерфейсом взаємодії між клієнтом і сервером, а й основою інтеграції мікросервісів, мобільних застосунків, зовнішніх партнерських сервісів, хмарних платформ і внутрішніх бізнес-процесів. Саме тому порушення

контролю доступу, помилки авторизації, неправильна перевірка токенів, надмірне розкриття даних, відсутність rate limiting або некоректна обробка виняткових ситуацій у API можуть призводити до критичних наслідків [10].

Зростання архітектурної складності безпосередньо впливає на профіль сучасних вразливостей. Якщо раніше домінували відносно прості та добре формалізовані технічні дефекти, зокрема SQL-ін’єкції, відбиті XSS-атаки або помилки роботи із сесіями, то сьогодні дедалі більшого значення набувають дефекти бізнес-логіки, порушення контролю доступу, помилки інтеграції між сервісами, проблеми ланцюга постачання програмного забезпечення, конфігураційні вразливості та ризики неправильної експлуатації хмарної інфраструктури. Це призводить до того, що класичні сигнатурні підходи, орієнтовані на пошук окремих шаблонів, дедалі частіше виявляються недостатніми [14].

Важливим чинником актуальності теми є також зміна організації самого процесу розробки. Поширення методологій Agile, DevOps, DevSecOps і CI/CD призвело до суттєвого прискорення постачання нових версій програмного забезпечення [8, 24]. У таких умовах часовий інтервал між внесенням змін до коду та розгортанням у продуктивному середовищі значно скоротився. Це означає, що повільні та ресурсоємні процедури аудиту безпеки, які виконуються лише наприкінці циклу розробки, уже не відповідають вимогам сучасної інженерної практики. Натомість виникає об’єктивна потреба у безперервному, автоматизованому та контекстно чутливому контролі безпеки протягом усього життєвого циклу вебзастосунку [8].

Ще однією суттєвою проблемою є зростання обсягу результатів, які генерують засоби аналізу безпеки. Навіть за використання кількох сканерів організація часто отримує велику кількість різнорідних звітів, у яких одна і та сама проблема може бути описана по-різному або дублюватися. Значна частина таких результатів має низьку практичну цінність або потребує трудомісткої ручної верифікації. Це створює ефект перевантаження аналітика технічними повідомленнями, знижує якість прийняття рішень і підвищує ризик пропуску дійсно критичних загроз [59].

За таких умов закономірним є перехід до інтелектуалізації процесів безпекового

аналізу. Інтелектуальні системи повинні не лише фіксувати окремі технічні ознаки дефекту, а й виконувати кореляцію результатів з різних джерел, зменшувати рівень шуму, враховувати контекст функціонування застосунку, підтримувати класифікацію загроз і формувати обґрунтовану пріоритезацію вразливостей. Саме такий підхід дозволяє перейти від реактивної моделі безпеки, орієнтованої на усунення вже виявлених наслідків, до проактивної моделі, у якій виявлення та оцінювання загроз вбудовуються безпосередньо в процес розробки та експлуатації.

Актуальність дослідження визначається одночасною дією кількох взаємопов'язаних чинників: ускладненням вебархітектур, збільшенням площини атаки, зміною характеру загроз, високою швидкістю сучасної розробки, поширенням ризиків ланцюга постачання, великою кількістю різнорідних результатів аналізу та необхідністю їх інтелектуальної інтерпретації. Це обґрунтовує доцільність розроблення інтелектуальної комп'ютерної системи автоматичного виявлення вразливостей вебзастосунків та класифікації загроз.

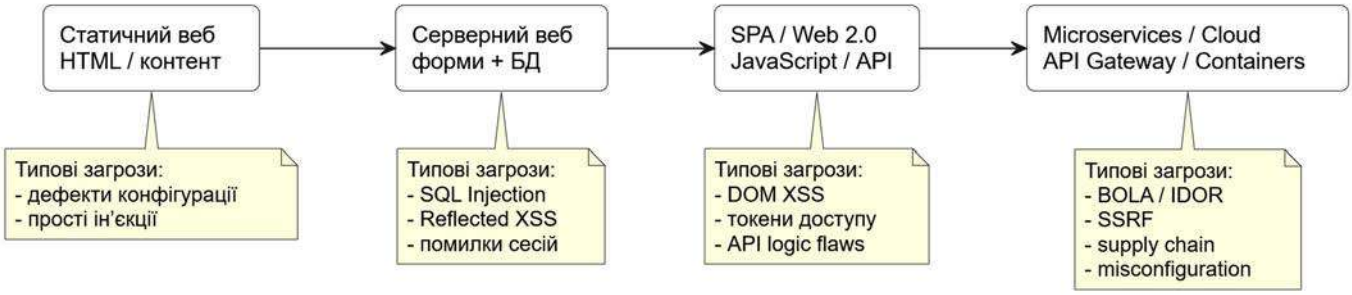


Рисунок 1.1 – Еволюція архітектури вебзастосунків: від моноліту до мікросервісної та API-центричної моделі

1.2 Класифікація та семантичний аналіз загроз за стандартами OWASP

Систематизація загроз у сфері безпеки вебзастосунків неможлива без використання міжнародних підходів, що формують єдиний понятійний апарат і забезпечують уніфікацію аналізу. Одним із найавторитетніших таких підходів є OWASP Top 10. Практична цінність OWASP полягає в тому, що він не лише

узагальнює найбільш критичні категорії ризиків, а й формує спільну методологічну основу для розробників, аудиторів безпеки, дослідників та інженерів захисту.

У межах цієї роботи OWASP Top 10 розглядається не як формальний перелік окремих вразливостей, а як структурований каркас предметної області, який дозволяє пов’язати технічні дефекти з ширшими класами загроз. Сучасна редакція OWASP Top 10:2025 відображає важливу тенденцію: фокус зміщується від суто локальних помилок програмування до більш загальних дефектів архітектури, контролю доступу, конфігурації, цілісності програмного забезпечення та ланцюга постачання. Це особливо важливо для побудови інтелектуальної системи, оскільки така система повинна працювати не лише з простими сигнатурними ознаками, а й зі складними контекстно залежними сценаріями [9].

Найбільш показовою зміною сучасного профілю загроз є провідна роль категорії Broken Access Control. Порушення контролю доступу охоплює не окрему небезпечну функцію, а дефекти логіки розмежування прав доступу до ресурсів системи. Типовими прикладами є IDOR, BOLA, горизонтальна й вертикальна ескалація привілеїв, обхід перевірок через альтернативні маршрути запитів, надмірний доступ до API-методів або некоректне застосування політик авторизації. Ця категорія особливо характерна для API-центричних і мікросервісних систем, де права доступу можуть перевірятися на кількох рівнях, а сама помилка проявляється лише в певному сценарії взаємодії [10, 25].

Другою важливою групою ризиків є Security Misconfiguration. До цієї категорії належать помилки налаштування серверів, контейнерів, cloud-середовищ, CORS-політик, HTTP-заголовків безпеки, механізмів журналювання, доступу до адміністративних інтерфейсів, секретів, змінних середовища та конфігурацій CI/CD. Небезпека конфігураційних помилок полягає в тому, що вони можуть бути відсутніми у вихідному коді, але проявлятися під час розгортання або експлуатації системи. Саме тому їх складно виявити лише засобами статичного аналізу коду.

Третьою актуальною категорією OWASP Top 10:2025 є Software Supply Chain Failures. Сучасні вебзастосунки значною мірою залежать від сторонніх бібліотек, контейнерних образів, SDK, фреймворків, пакетних менеджерів, build-скриптів і

зовнішніх сервісів. Уразливість може бути відсутньою у власному коді застосунку, але присутньою у транзитивній залежності або контейнерному образі, що істотно ускладнює традиційний підхід до аналізу безпеки. У таких умовах важливу роль відіграють Software Composition Analysis, SBOM та контроль ланцюга постачання програмного забезпечення [25]. Для ІКС АВВЗ результати SCA-аналізу мають розглядатися як повноцінне джерело для інтелектуального ранжування загроз, а не як допоміжна інформація.

Категорія Cryptographic Failures охоплює не лише використання слабких криптографічних алгоритмів, а й помилки практичної реалізації захисту даних [2]. До таких помилок належать неправильне зберігання ключів, передавання чутливої інформації без належного захисту, відсутність HSTS, неналежне керування TLS-сертифікатами, використання застарілих протоколів, неправильне хешування паролів або зберігання секретів у відкритому вигляді. У контексті інтелектуальної системи важливо враховувати не лише сам факт криптографічного дефекту, а й те, які дані піддаються ризику та в якому архітектурному контексті виникає проблема.

Ін’єкційні вразливості, попри зміну позицій у сучасній класифікації OWASP, залишаються одним із найнебезпечніших класів технічних загроз для вебзастосунків. Їх сутність полягає в тому, що неперевірені або неналежним чином оброблені вхідні дані передаються інтерпретатору як частина виконуваної команди, унаслідок чого атакувальник отримує змогу змінити логіку її виконання.

До типових різновидів належать SQL-, NoSQL-, LDAP-, шаблонні, командні ін’єкції та XSS. Небезпека ін’єкційних вразливостей зумовлена не лише тяжкими наслідками, а й тим, що сучасні форми таких атак можуть бути обфускованими, багатокроковими та нетривіальними для сигнатурного пошуку [13].

Категорія Insecure Design пов’язана з архітектурними й концептуальними помилками, які не можуть бути повністю усунуті лише виправленням окремих ділянок коду. Йдеться про відсутність threat modeling, неправильну модель довіри, відсутність обмежень на критичні операції, небезпечні бізнес-процеси або недостатній захист сценаріїв зловживання. Для інтелектуальної системи ця

категорія є складною, оскільки потребує не лише технічного аналізу, а й урахування логіки функціонування вебзастосунку.

Authentication Failures охоплює помилки, пов’язані з ідентифікацією користувачів, керуванням сесіями, токенами, паролями, MFA та механізмами відновлення доступу. У сучасних вебзастосунках ця категорія має особливе значення через широке використання OAuth, OpenID Connect, JWT, зовнішніх identity provider та federated authentication. Помилка в такому механізмі може мати системні наслідки, оскільки один дефект автентифікації здатний відкрити доступ до багатьох сервісів.

Software or Data Integrity Failures пов’язані з порушенням цілісності програмного коду, даних, CI/CD-процесів, оновлень, залежностей та артефактів розгортання. Ця категорія має прямий зв’язок із безпекою supply chain, оскільки компрометація build pipeline, контейнерного образу або пакета залежності може призвести до масового впровадження шкідливого коду в легітимне середовище.

Security Logging & Alerting Failures відображає проблему недостатнього журналювання, моніторингу, кореляції подій та реагування на інциденти [15]. Для задачі ІКС АВВЗ ця категорія має подвійне значення. З одного боку, слабке журналювання ускладнює виявлення реальної експлуатації вразливостей. З іншого боку, надмірна кількість неструктурованих подій створює проблему alert fatigue, через яку аналітик може пропустити критичний інцидент.

Mishandling of Exceptional Conditions охоплює неправильну обробку помилок, винятків, аварійних станів, граничних ситуацій і fail-open сценаріїв. Такі дефекти часто залишаються поза увагою класичних сканерів, оскільки проявляються лише за нетипових або навмисно спотворених умов виконання. Для розподілених систем ця категорія є особливо важливою, оскільки помилка одного сервісу може спричинити каскадний ефект у всій архітектурі.

Таблиця 1.1 – Категорії OWASP Top 10:2025 та їх значення для задачі ІКС АВВЗ

Категорія	Зміст	Значення для ІКС АВВЗ
A01:2025 Broken Access Control	Порушення правил доступу до ресурсів, IDOR, BOLA, ескалація привілеїв	Потребує контекстного аналізу ролей, маршрутів API та бізнес-логіки
A02:2025 Security Misconfiguration	Помилки конфігурації серверів, контейнерів, CORS, HTTP-заголовків, cloud-сервісів	Вимагає аналізу середовища розгортання, а не лише коду
A03:2025 Software Supply Chain Failures	Ризики залежностей, пакетів, build pipeline, контейнерних образів	Обґрунтовує інтеграцію SCA, SBOM і аналізу залежностей
A04:2025 Cryptographic Failures	Неправильне шифрування, зберігання ключів, відсутність HSTS, витік чутливих даних	Потребує перевірки не лише алгоритмів, а й практики реалізації
A05:2025 Injection	SQL/NoSQL/LDAP/command/template injection, XSS	Залишається важливим класом технічних дефектів
A06:2025 Insecure Design	Архітектурні помилки, відсутність threat modeling, небезпечні бізнес-сценарії	Пов’язано з аналізом системної моделі й сценаріїв атак
A07:2025 Authentication Failures	Помилки автентифікації, сесій, токенів, MFA	Потребує аналізу механізмів ідентифікації користувачів
A08:2025 Software or Data Integrity Failures	Порушення цілісності коду, даних, CI/CD, оновлень	Важливо для контролю довіри до артефактів розгортання
A09:2025 Security Logging & Alerting Failures	Недостатність журналювання, моніторингу й сповіщень	Пов’язано з проблемою alert fatigue та пріоритезації подій
A10:2025 Mishandling of Exceptional Conditions	Неправильна обробка винятків, помилки fail-safe/fail-open	Потребує аналізу поведінки системи в нестандартних умовах

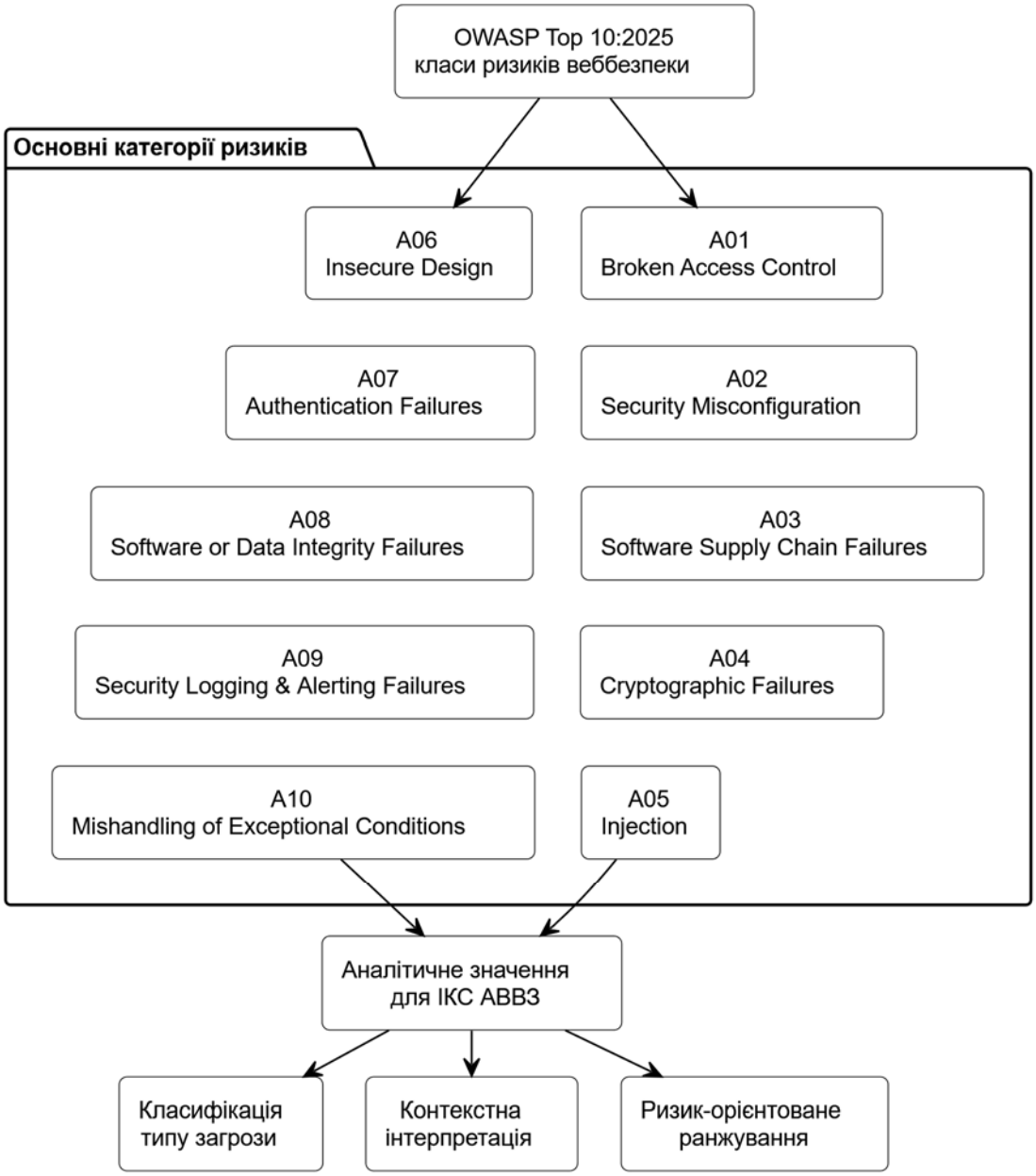


Рисунок 1.2 – Класифікація загроз вебзастосунків за стандартами OWASP та їх аналітичне значення

Таким чином, OWASP Top 10:2025 демонструє, що сучасний профіль загроз дедалі більше зміщується від окремих локальних помилок програмування до комплексних системних дефектів, пов’язаних із контролем доступу, конфігурацією, ланцюгом постачання, цілісністю програмного забезпечення, API-взаємодією та обробкою виняткових умов [9]. Для ІКС АВВЗ це означає, що система повинна підтримувати не лише виявлення окремих технічних дефектів, а й семантичну інтерпретацію контексту, у якому ці дефекти виникають.

1.3 Стандарти, бази знань і шкали оцінювання вразливостей

Для формального опису вразливостей і загроз поряд з OWASP використовуються також CVE, CWE, CAPEC та CVSS. Ці джерела виконують різні функції, але в сукупності формують методологічну основу для побудови системи класифікації загроз.

CVE забезпечує ідентифікацію конкретних відомих вразливостей. Кожен CVE-запис пов’язаний із певним дефектом у програмному продукті, бібліотеці, фреймворку, сервісі або інфраструктурному компоненті [21]. Практичне значення CVE полягає в тому, що він дозволяє однозначно зіставити знайдену проблему з уже відомою вразливістю, доступними патчами, exploit-інформацією та рекомендаціями щодо усунення [17].

CWE класифікує типові слабкі місця програмного забезпечення. Якщо CVE описує конкретну вразливість у конкретному продукті, то CWE узагальнює тип помилки, наприклад неправильну перевірку вхідних даних, некоректний контроль доступу, використання жорстко закодованих секретів або помилки керування пам’яттю. У задачі ІКС АВВЗ CWE є важливим для узагальнення результатів різних сканерів і приведення їх до єдиного семантичного простору [18].

CAPEC описує шаблони атак і дозволяє пов’язати вразливість із можливим сценарієм її експлуатації. Якщо CWE відповідає на питання, який тип слабкості присутній у системі, то CAPEC допомагає описати, як ця слабкість може бути використана атакувальником [22]. Це важливо для переходу від формального списку дефектів до аналізу реальних загроз.

CVSS надає кількісну шкалу оцінювання критичності вразливостей. Вона враховує характеристики експлуатації, вплив на конфіденційність, цілісність і доступність, а також додаткові часові й контекстні параметри. У практиці кібербезпеки CVSS широко використовується для первинної пріоритезації патчів, визначення рівня критичності інцидентів і планування реагування.

Разом з тим CVSS не завжди достатній для прийняття остаточного рішення. Одна й та сама CVSS-оцінка може мати різне практичне значення залежно від того,

чи використовується вразливий компонент у критичному сервісі, чи доступний він з Інтернету, чи обробляє чутливі дані, чи має компенсуючі засоби захисту [20]. Тому в межах ІКС АВВЗ CVSS доцільно розглядати як базову технічну складову ризику, яка має доповнюватися контекстною та ймовірнісною оцінкою.

Наприклад, під час аналізу вразливої бібліотеки SCA-інструмент може виявити конкретний CVE-ідентифікатор, CWE дозволяє віднести його до відповідного типу слабкості, CAPEC описує потенційний сценарій експлуатації, а CVSS надає базову кількісну оцінку критичності [19]. Однак остаточне рішення щодо пріоритету виправлення має враховувати не лише CVSS, а й контекст використання компонента, доступність експлуатації, роль сервісу в архітектурі та рівень невизначеності результату аналізу.

Таким чином, OWASP, CVE, CWE, CAPEC, CVSS і SBOM формують необхідний методологічний каркас для аналізу безпеки вебзастосунків. Проте жодне з цих джерел окремо не забезпечує повноцінного контекстного аналізу. Тому в межах даної роботи вони розглядаються як джерела знань, що мають бути інтегровані з інтелектуальними методами нормалізації, класифікації, кореляції та ризик-орієнтованої пріоритезації [28, 29].

1.4 Порівняльний аналіз інструментальних методів виявлення вразливостей

Забезпечення безпеки вебзастосунків у межах життєвого циклу розробки ґрунтується на використанні автоматизованих інструментальних підходів, серед яких основними є SAST, DAST, IAST та SCA. Кожен із цих методів має власну область застосування, сильні сторони та обмеження, а тому не може розглядатися як універсальний самодостатній засіб [11, 30].

SAST передбачає аналіз вихідного коду, байт-коду або бінарних представлень без фактичного запуску застосунку. Основною перевагою цього підходу є можливість раннього виявлення небезпечних конструкцій ще на етапах написання коду та збірки, що добре узгоджується з підходом раннього вбудовування безпеки в процес розробки. Крім того, SAST дозволяє формально

досягати високого покриття кодової бази, включно з гілками логіки, які не завжди активуються під час виконання. Разом із тим недоліками підходу є велика кількість хибних спрацювань, недостатній облік контексту виконання та складність інтерпретації результатів у масштабних проєктах [16, 30, 31, 41].

DAST орієнтований на аналіз уже запущеного застосунку і реалізує взаємодію з ним у режимі «чорної скриньки». Його цінність полягає в тому, що він досліджує реальну поведінку системи, працює незалежно від мови програмування і дозволяє виявляти експлуатаційні дефекти, які можуть бути непомітними на рівні статичного аналізу. DAST особливо корисний для виявлення помилок конфігурації, дефектів у runtime-середовищі, проблем маршрутизації, неправильних HTTP-заголовків, помилок автентифікації та доступних ззовні точок входу. Водночас DAST має обмеження, пов’язані з неповним покриттям функціоналу: сканер може не пройти складну автентифікацію, не активувати певний бізнес-сценарій або не знайти окрему точку входу. Крім того, результати DAST не завжди легко пов’язати з конкретною ділянкою коду, що ускладнює подальше виправлення [12, 32].

IAST поєднує ознаки статичного та динамічного аналізу, оскільки працює в середовищі виконання застосунку та водночас має доступ до його внутрішнього контексту. Це дозволяє зменшити кількість хибних спрацювань, краще локалізувати проблему в коді та враховувати реальні сценарії оброблення даних. Водночас такий підхід потребує інструментування застосунку, є залежним від сценаріїв виконання і зазвичай складніший в інтеграції.

SCA спрямований на аналіз залежностей, бібліотек, пакетів, контейнерних образів і сторонніх компонентів. Для сучасних вебзастосунків це особливо актуально, оскільки значна частина ризиків пов’язана саме з використанням вразливих, застарілих або скомпрометованих компонентів. Перевагою SCA є можливість оперативно виявляти відомі вразливості в екосистемі залежностей, формувати SBOM і відстежувати ризики ланцюга постачання. Однак цей підхід сам по собі не дає відповіді, чи використовується конкретна вразлива функціональність реально, який бізнес-вплив матиме експлуатація та чи існують компенсуючі засоби

захисту [36, 37].

Порівняльний аналіз показує, що жоден із зазначених підходів не забезпечує одночасно високої повноти, точності, контекстності та практичної придатності результатів. SAST добре працює на ранніх етапах, але генерує багато шуму. DAST показує реальну поведінку, але не дає повного покриття. IAST підвищує точність, але вимагає складнішої інтеграції. SCA виявляє ризики залежностей, але обмежений контекстно. Саме тому найбільш перспективним є гібридний підхід, у межах якого результати різних методів поєднуються в єдиному аналітичному контурі [34].

Якщо SAST вказує на потенційно небезпечну ділянку коду, DAST підтверджує її експлуатаційність, IAST надає внутрішній runtime-контекст, а SCA показує, що пов’язаний компонент має відому вразливість, то такий інцидент отримує вищий рівень достовірності. Додавання AI-модуля до такої архітектури дозволяє відсіювати дублікати, зменшувати кількість хибних спрацювань, ураховувати контекст та формувати більш обґрунтоване ранжування загроз [33].

Таблиця 1.2 – Порівняльна характеристика методів аналізу захищеності

Критерій	SAST	DAST	IAST	SCA
Об’єкт аналізу	Вихідний код, байт-код, бінарні представлення	Запущений застосунок	Код під час виконання	Залежності, бібліотеки, компоненти, образи
Тип тестування	З повним доступом до внутрішньої структури	Без доступу до внутрішньої структури	З частковим доступом до внутрішньої структури	Аналіз складу програмних компонентів
Етап використання	Розробка, збірка	Тестування, staging, pre-production	Тестування, експлуатація	Розробка, збірка, підтримка
Головна перевага	Раннє виявлення дефектів	Аналіз реальної поведінки системи	Контекст виконання	Контроль сторонніх компонентів і SBOM

Кінець таблиці 1.2

Головне обмеження	Високий рівень хибнопозитивних спрацювань	Неповне покриття функціоналу	Потреба в інструментуванні	Обмежений поведінковий контекст
Типові загрози	Синтаксичні та структурні дефекти	Експлуатаційні вразливості, misconfiguration	Контекстні дефекти	Вразливі залежності та supply chain ризики
Приклади інструментів	SonarQube, Semgrep, CodeQL	OWASP ZAP, Burp Suite, Nikto	Contrast Security, Seeker	Snyk, OWASP Dependency-Check, Trivy, Grype

1.5 Аналіз методів штучного інтелекту в задачах кібербезпеки

Застосування штучного інтелекту в задачах кібербезпеки є закономірним етапом розвитку систем автоматизованого аналізу [4, 38]. На відміну від жорстко детермінованих сигнатурних механізмів, інтелектуальні моделі здатні працювати з великими обсягами слабкоструктурованих даних, виявляти приховані закономірності, адаптуватися до нових патернів атак і виконувати глибшу інтерпретацію технічних артефактів.

Першим етапом автоматизації пошуку вразливостей були сигнатурні та правило-орієнтовані механізми. Їхня перевага полягає у простоті реалізації, швидкості роботи та високій ефективності проти вже відомих шаблонів. Проте такі підходи різко втрачають результативність у разі обфускації, зміни форми корисного навантаження, появи нових технік або виникнення логічних дефектів, які не мають очевидної сигнатури [50].

Наступним етапом розвитку стало класичне машинне навчання, що дозволило перейти від прямого пошуку сигнатур до використання простору ознак, на основі якого модель формує рішення про належність об’єкта до певного класу загроз. Водночас такі підходи істотно залежать від якості конструювання ознак і

від правильності навчальної вибірки [43, 53].

Глибинне навчання суттєво розширило можливості аналізу безпеки [38]. Сучасні нейронні архітектури дають змогу працювати з кодом, HTTP-трафіком і текстовими описами як із послідовними, структурованими або графовими даними. Для розподілених середовищ окремим прикладом поведінкового аналізу є виявлення бот-мереж на основі спостереження за мережевими об’єктами та їх взаємодіями, що підтверджує значення поведінкових і мережевих ознак у задачах класифікації кіберзагроз [72]. CNN-моделі добре виявляють локальні закономірності й шаблони, RNN та LSTM корисні для аналізу послідовностей, аномалій і часових залежностей. Трансформерні підходи, зокрема BERT, CodeBERT і подібні моделі, забезпечують глибший контекстний аналіз коду, текстових описів, звітів сканерів і технічної документації. GNN-моделі дозволяють працювати з графовими поданнями технічних залежностей і структурних зв’язків, що особливо важливо для аналізу програмного коду, AST, CFG, DFG, графів викликів і міжкомпонентних залежностей [40, 41, 46, 51, 54].

Для задачі виявлення вразливостей вебзастосунків особливе значення має здатність інтелектуальних підходів поєднувати різні типи даних: вихідний код, результати сканерів, параметри HTTP-запитів, контекст виконання, зовнішні бази знань, SBOM, CVE/CWE/CAPEC/CVSS-записи та описові текстові артефакти. Це дає можливість перейти від ізольованого аналізу окремого повідомлення сканера до змістовної класифікації вразливості, визначення її типу, узгодження з відомими класами ризику та формування ризик-орієнтованої оцінки [36, 45].

Окремим перспективним напрямом є використання великих мовних моделей для пояснення результатів сканування, генерації рекомендацій щодо виправлення, підготовки звітів і підтримки DevSecOps-команд. Проте LLM-рішення не можуть розглядатися як повністю автономний механізм прийняття рішень, оскільки для них залишаються актуальними проблеми хибних узагальнень, нестабільності відповідей, залежності від якості вхідного контексту та можливих адверсаріальних впливів [57, 62]. Практичне використання мовних моделей для пошуку вразливостей вебзастосунків також розглядається в українських дослідженнях,

зокрема через застосування API ChatGPT для аналізу вебзастосунків і підтримки формування результатів перевірки [67]. Тому в межах ІКС АВВЗ доцільно поєднувати LLM-компонент із формалізованими метриками ризику, результатами SAST/DAST/SCA/IAST, базами знань CVE/CWE/CAPEC/CVSS та методами оцінювання невизначеності [43, 47, 52, 56].

Важливим елементом інтелектуального підходу є нормалізація даних. У практиці кібербезпеки різні засоби аналізу генерують звіти у форматах XML, JSON, CSV, HTML або у власних структурах даних. Одна й та сама вразливість може бути описана різними термінами, мати різні ідентифікатори та різний рівень деталізації. Без узгодження таких даних неможливо ані навчити модель на якісній вибірці, ані забезпечити коректну інтерпретацію результатів [45, 56]. Саме тому інтелектуальна нормалізація є необхідною передумовою для побудови ІКС АВВЗ.

Для оцінювання якості інтелектуальних моделей у завданнях кібербезпеки недостатньо використовувати лише метрику точності класифікації, оскільки за умов незбалансованості даних вона може створювати хибне враження високої ефективності. Доцільніше застосовувати сукупність метрик прецизійності, повноти та F1-міри: прецизійність характеризує здатність зменшувати кількість хибнопозитивних спрацювань, повнота – здатність не пропускати реальні загрози, а F1-міра забезпечує збалансовану оцінку між цими показниками [55].

Окремим критично важливим аспектом є оцінювання невизначеності. У задачах кібербезпеки навіть формально правильна класифікація не завжди означає достатній рівень довіри до сформованого рішення. Це особливо актуально для нових, слабко представлених, прикордонних або контекстно складних загроз. Тому перспективним є використання ймовірнісних і Bayesian-підходів, які дозволяють враховувати не лише клас загрози, а й рівень упевненості моделі. У межах цієї роботи така логіка надалі розвивається через епістемічно-алеаторну декомпозицію ризику [50, 58].

Таблиця 1.3 – Узагальнена характеристика AI-підходів у завданнях кібербезпеки

Підхід	**Тип вхідних даних**	**Сильні сторони**	**Обмеження**
Правило-орієнтований підхід	Сигнатури, регулярні вирази, правила	Швидкість, простота, контрольованість	Слабка адаптивність до нових атак
Класичне машинне навчання	Вектори ознак	Відносна інтерпретованість, контрольованість	Залежність від ручного конструювання ознак
CNN / RNN / LSTM	Послідовності, шаблони, трафік	Аналіз локальних і часових патернів	Гірша робота зі складним контекстом
Transformer / BERT / CodeBERT	Текст, код, звіти сканерів, контекстні подання	Глибокий семантичний аналіз	Ресурсоємність, потреба у якісних даних
GNN	AST, CFG, DFG, графи викликів, залежності	Структурний аналіз коду й архітектури	Складність підготовки графових даних

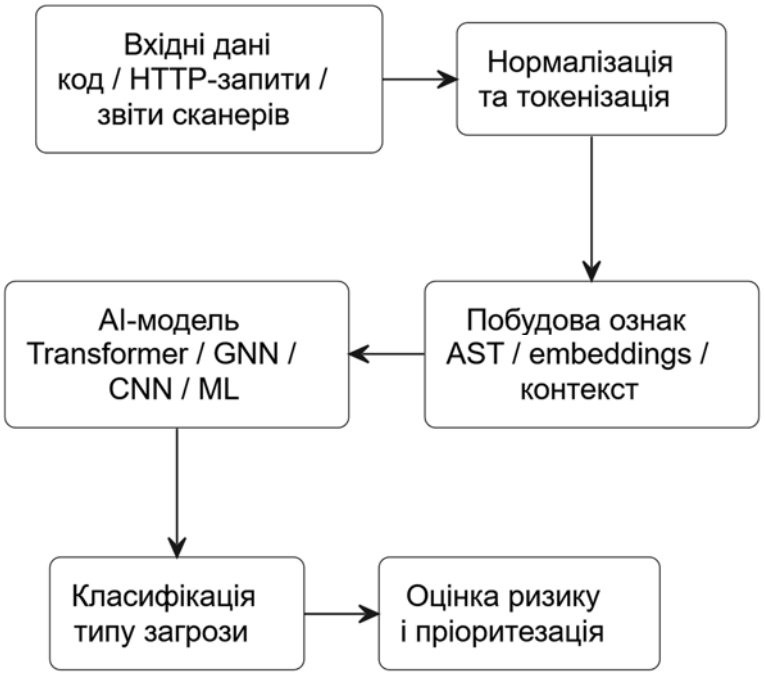


Рисунок 1.3 – Узагальнена блок-схема роботи AI-модуля інтелектуальної системи

Отже, методи штучного інтелекту створюють методологічну основу для переходу від фрагментарного сигнатурного аналізу до контекстної, семантичної та адаптивної інтерпретації загроз. Це робить їх ключовим елементом побудови інтелектуальної комп'ютерної системи автоматичного виявлення вразливостей вебзастосунків та класифікації загроз [67].

1.6 Актуальні проблеми та постановка задачі дослідження

Проведений аналіз сучасних підходів до забезпечення безпеки вебзастосунків дозволяє виділити низку ключових проблем, що стримують ефективність автоматизованого виявлення вразливостей і класифікації загроз.

Першою проблемою є фрагментованість результатів аналізу. На практиці організації часто використовують кілька безпекових інструментів одночасно: SAST, DAST, IAST, SCA, сканери конфігурацій, контейнерні сканери та засоби моніторингу. Однак результати їхньої роботи залишаються роз'єднаними, а одна й та сама проблема може бути подана як різні інциденти. Це створює дублювання, інформаційну надмірність та перевантаження аналітика.

Другою проблемою є високий рівень хибних спрацювань. Багато сучасних засобів безпеки генерують велику кількість технічних записів, значна частина яких не має достатньої практичної цінності або потребує ручної перевірки. У результаті формується ситуація, коли фахівець з кібербезпеки змушений витрачати значні ресурси на фільтрацію шуму, а не на аналіз дійсно критичних ризиків. Одним із напрямів підвищення достовірності виявлення атак є використання децепційних механізмів, зокрема пасток, приманок і контрольованих середовищ, які дозволяють фіксувати поведінку атакувальника або шкідливого програмного забезпечення в керованому контексті [66].

Третьою проблемою є відсутність контекстної оцінки ризику. Більшість існуючих систем обмежується технічними характеристиками вразливості або стандартним CVSS-оцінюванням. Водночас реальна критичність загрози залежить також від бізнес-ролі компонента, доступу до чутливих даних, важливості сервісу,

наявності компенсуючих механізмів захисту, доступності експлуатації та конкретного сценарію атаки [66].

Четверта проблема полягає в обмеженості сигнатурних і вузькотехнічних підходів щодо логічних, поведінкових і семантично складних вразливостей. Найбільш небезпечні сучасні загрози дедалі частіше пов’язані не з окремим шаблоном у коді, а з помилкою в логіці функціонування, маршрутизації запитів, взаємодії між компонентами або розмежуванні прав доступу. Такі дефекти потребують не лише синтаксичного, а й контекстного аналізу [60].

П’ята проблема стосується суперечності між швидкістю сучасної розробки та глибиною безпекової перевірки. У середовищах DevOps і CI/CD безпека повинна бути інтегрована в конвеєр розробки так, щоб не сповільнювати його критично, але водночас забезпечувати достатній рівень достовірності виявлення ризиків. Існуючі ізольовані інструменти часто не досягають цього балансу [27].

Шоста проблема пов’язана з ризиками ланцюга постачання програмного забезпечення. Сучасні застосунки містять велику кількість зовнішніх залежностей, транзитивних пакетів, контейнерних образів і build-артефактів. Вразливість або компрометація одного компонента може мати системний вплив, навіть якщо власний код застосунку не містить очевидного дефекту. Тому аналіз залежностей, SBOM і контроль supply chain мають бути частиною єдиного аналітичного процесу.

Сьомою суттєвою проблемою є недостатнє врахування невизначеності автоматично сформованих рішень. У практиці кібербезпеки навіть формально коректна класифікація не завжди означає достатній рівень довіри до результату, особливо у випадках нових, слабко представлених або прикордонних загроз. Тому перспективним напрямом розвитку інтелектуальних систем виявлення вразливостей є не лише підвищення точності класифікації, а й явне врахування епістемічної невизначеності при оцінюванні ризику та підтримці прийняття рішень.

Таким чином, об’єктивно виникає потреба в побудові інтегрованої інтелектуальної системи, яка б поєднувала результати багатоканального аналізу, виконувала їх нормалізацію, класифікацію, кореляцію та ризик-орієнтоване

ранжування. Саме це визначає постановку задачі даної кваліфікаційної роботи [60].

У межах дослідження необхідно розробити інтелектуальну комп’ютерну систему автоматичного виявлення вразливостей вебзастосунків та класифікації загроз, яка повинна:

– інтегрувати результати різних інструментів безпекового аналізу, зокрема SAST, DAST, IAST і SCA;

– забезпечувати інтелектуальну нормалізацію та уніфікацію різнорідних даних;

– зіставляти знайдені дефекти з OWASP, CVE, CWE, CAPEC, CVSS і SBOM;

– виконувати класифікацію вразливостей і загроз на основі інтелектуальних методів;

– оцінювати критичність загроз з урахуванням технічних, контекстних і ймовірнісних факторів;

– враховувати невизначеність результатів класифікації та ризикового оцінювання;

– формувати ранжований перелік ризиків для практичного реагування;

– підтримувати створення зрозумілих аналітичних звітів для DevSecOps-команд і фахівців із кібербезпеки.

Отже, задача дослідження полягає не лише у виявленні окремих вразливостей, а у побудові інтегрованого аналітичного контуру, який забезпечує агрегацію результатів SAST, DAST, IAST і SCA; нормалізацію різнорідних повідомлень сканерів; зіставлення знайдених дефектів із OWASP, CVE, CWE, CAPEC і CVSS; інтелектуальну класифікацію типу загрози; оцінювання критичності з урахуванням технічного, контекстного та ймовірнісного складників; формування ранжованого переліку ризиків для практичного реагування.

Ключовим недоліком існуючих систем є недостатнє врахування невизначеності результатів аналізу, що знижує достовірність ризикового оцінювання та ускладнює прийняття рішень. Отже, сформульована постановка задачі безпосередньо випливає з проведеного аналізу й визначає необхідність подальшого розроблення моделей та методів.

1.7 Висновки до першого розділу

У першому розділі проведено аналіз сучасного стану проблеми автоматичного виявлення вразливостей вебзастосунків і класифікації загроз. Показано, що розвиток вебтехнологій, поширення API-центричних і мікросервісних архітектур, активне використання зовнішніх компонентів, контейнерів, хмарної інфраструктури та CI/CD-конвеєрів істотно ускладнили завдання забезпечення безпеки вебзастосунків.

Встановлено, що сучасний профіль загроз зміщується від ізольованих технічних дефектів до комплексних проблем контролю доступу, конфігурації, ланцюга постачання програмного забезпечення, цілісності даних і архітектурної стійкості. Аналіз OWASP Top 10:2025 показав, що особливого значення для сучасних вебзастосунків набувають Broken Access Control, Security Misconfiguration, Software Supply Chain Failures, Cryptographic Failures, Injection, Insecure Design, Authentication Failures, Software or Data Integrity Failures, Security Logging & Alerting Failures та Mishandling of Exceptional Conditions.

Встановлено, що стандарти й бази знань OWASP, CVE, CWE, CAPEC, CVSS і SBOM формують необхідну методологічну основу для опису загроз, однак самі по собі не забезпечують повноцінної контекстної інтерпретації, кореляції та пріоритезації результатів аналізу. Тому їх доцільно використовувати як джерела знань у складі інтегрованої інтелектуальної системи.

Порівняльний аналіз інструментальних методів показав, що SAST, DAST, IAST та SCA мають істотну практичну цінність, проте в ізольованому використанні не забезпечують необхідної повноти, точності та придатності до застосування в сучасному високошвидкісному процесі розробки. SAST дозволяє виявляти дефекти на ранніх етапах, DAST досліджує реальну поведінку запущеного застосунку, IAST поєднує runtime-контекст із внутрішнім аналізом, а SCA виявляє ризики залежностей і ланцюга постачання. Найбільш перспективним є їх поєднання в єдиному аналітичному контурі.

Обґрунтовано доцільність використання методів штучного інтелекту, які забезпечують перехід від вузько сигнатурного підходу до контекстного,

семантичного й адаптивного аналізу. Показано, що інтелектуалізація аналізу безпеки дозволяє зменшувати кількість хибних спрацювань, підвищувати якість класифікації загроз, ураховувати поведінковий і бізнес-контекст, а також підтримувати ризик-орієнтоване прийняття рішень. У контексті аналізу сучасних підходів встановлено, що перспективним є гібридний класифікаційний контур, який поєднує текстові, контекстні, джерельні, структурні та семантичні ознаки.

На основі виконаного аналізу визначено головні проблеми предметної області: фрагментованість результатів різних інструментів, високий рівень хибних спрацювань, відсутність контекстної оцінки ризику, обмеженість традиційних підходів щодо логічних вразливостей, ризики ланцюга постачання, суперечність між швидкістю розробки та глибиною аналізу, а також недостатнє врахування невизначеності автоматично сформованих рішень.

Отже, сукупність проаналізованих тенденцій і обмежень існуючих підходів зумовлює потребу у створенні інтегрованої інтелектуальної комп’ютерної системи автоматичного виявлення вразливостей вебзастосунків та класифікації загроз. Така система має забезпечувати нормалізацію та кореляцію результатів різних сканерів, зіставлення знайдених дефектів із міжнародними базами знань, інтелектуальну класифікацію загроз, ризик-орієнтоване ранжування та врахування невизначеності результатів аналізу. Це формує підґрунтя для побудови формальної моделі ІКС АВВЗ у другому розділі роботи.

2 АРХІТЕКТУРА, МОДЕЛІ ТА МЕТОДИ ІНТЕЛЕКТУАЛЬНОЇ КОМП'ЮТЕРНОЇ СИСТЕМИ АВТОМАТИЧНОГО ВИЯВЛЕННЯ ВРАЗЛИВОСТЕЙ ВЕБЗАСТОСУНКІВ ТА КЛАСИФІКАЦІЇ ЗАГРОЗ

У цьому розділі розроблено моделі та методи інтелектуальної комп'ютерної системи автоматичного виявлення вразливостей вебзастосунків та класифікації загроз. Виконано формалізацію предметної області, побудовано архітектурну модель системи, визначено модель інтелектуальної нормалізації результатів сканування, модель класифікації вразливостей і кіберзагроз, а також модель адаптивного оцінювання ризику. Запропоновані положення утворюють теоретичну основу для подальшої алгоритмізації, проєктування та програмної реалізації розроблюваної системи.

2.1 Формалізація предметної області та постановка задачі моделювання

Побудова моделей інтелектуальної комп'ютерної системи автоматичного виявлення вразливостей вебзастосунків та класифікації загроз потребує формалізації предметної області, визначення основних сутностей, їх зв'язків, вхідних і вихідних даних, а також постановки задачі моделювання. Такий підхід дозволяє перейти від описового аналізу проблеми до формального подання системи як сукупності взаємопов'язаних перетворень, у межах яких результати роботи різних безпекових засобів інтегруються, нормалізуються, класифікуються та ранжуються за рівнем ризику.

У межах роботи розглядається множина вебзастосунків, що функціонують у сучасному IT-середовищі та є об'єктами автоматизованого аналізу безпеки:

$$W = \{w_1, w_2, \dots, w_N\} \tag{2.1}$$

де w_i – окремий вебзастосунок, для якого виконується аналіз безпеки.

Кожний вебзастосунок w_i описується множиною артефактів

$$A(w_i) = \{a_{i1}, a_{i2}, \dots, a_{iK}\}, \quad (2.2)$$

де $A(w_i)$ – множина артефактів, пов'язаних із вебзастосунком w_i; сирцевий код, конфігураційні файли, API-специфікації, залежності, SBOM-описи, контейнерні образи, IaC-конфігурації, журнали подій, HTTP-артефакти, метадані API gateway / service mesh та інші дані, що характеризують програмне й інфраструктурне середовище вебзастосунку. Множина джерел аналізу безпеки задається як:

$$S = \{s_1, s_2, \dots, s_M\}, \quad (2.3)$$

де s_j – окреме джерело аналізу безпеки, до яких належать SAST, DAST, IAST, SCA, аналіз конфігурацій, журнали подій та інші канали надходження результатів. Для кожного вебзастосунку w_iі кожного джерела s_j формується множина первинних результатів:

$$F_{\{ij\}} = \{f_{\{ij1\}}, f_{\{ij2\}}, \dots, f_{\{ijL\}}\}, \quad (2.4)$$

де $f_{\{ijL\}}$ – окрема знахідка, технічне повідомлення сканера або безпекова подія. Сукупність первинних результатів аналізу для вебзастосунку w_iвизначається як:

$$F_i^{raw} = \bigcup_{j=1}^{M} F_{ij}, \quad (2.5)$$

Множина F_i^{raw} є неоднорідною за структурою, змістом, повнотою опису та рівнем деталізації. Один і той самий дефект може бути поданий різними інструментами по-різному: із відмінними ідентифікаторами, рівнями критичності, назвами класів загроз і набором технічних метаданих. Саме тому первинні результати не можуть безпосередньо використовуватися для автоматизованого прийняття рішень і потребують попередньої інтелектуальної нормалізації. Окрему

первинну знахідку доцільно подати у вигляді кортежу:

$$f = \langle id_w, src, loc, desc, sev, ev, t \rangle, \quad (2.6)$$

де id_w – ідентифікатор вебзастосунку;

src – джерело формування запису;

loc – локалізація дефекту;

$desc$ – текстовий опис;

sev – початкова оцінка критичності;

ev – технічні докази або ознаки експлуатації;

t – часові метадані.

Після етапу інтелектуальної нормалізації кожна знахідка перетворюється у стандартизований запис

$$z = N(f), \quad (2.7)$$

де N – оператор нормалізації.

Нормалізований запис подається як

$$z =< id_w, a, x, y, cwe, cve, capec, owasp, cvss, sbom, p, r >, \quad (2.8)$$

де a – уражений артефакт;

x – вектор ознак;

y – клас вразливості;

cwe – ідентифікатор слабкості;

cve – ідентифікатор відомої вразливості, якщо знайдено відповідність;

$owasp$ – категорія OWASP;

$capec$ – шаблон атаки або сценарій експлуатації, якщо встановлено відповідність;

$cvss$ – базова або нормалізована технічна оцінка критичності;

$sbom$ – відомості про програмний компонент, залежність, пакет, контейнерний образ або транзитивну залежність, якщо запис пов'язаний із ризиком ланцюга постачання.

p – рівень довіри класифікаційної моделі;

r – інтегральна оцінка ризику.

Множина допустимих класів вразливостей визначається як

$$Y = \{y_1, y_2, \dots, y_K\}, \tag{2.9}$$

де елементи множини Y можуть відповідати категоріям OWASP Top 10:2025, зокрема A01 Broken Access Control, A02 Security Misconfiguration, A03 Software Supply Chain Failures, A04 Cryptographic Failures, A05 Injection, A06 Insecure Design, A07 Authentication Failures, A08 Software or Data Integrity Failures, A09 Security Logging & Alerting Failures, A10 Mishandling of Exceptional Conditions, а також класам CWE або внутрішній прикладній таксономії системи. Тоді задача класифікації для нормалізованого запису формулюється як побудова відображення

$$C: Z \rightarrow Y, \tag{2.10}$$

де Z – множина всіх нормалізованих записів. У випадку ймовірнісної моделі класифікації рішення системи визначається як

$$C(z) = \underset{y_k \in Y}{arg\ \max} P(y_k \mid x), \tag{2.11}$$

де $P(y_k \mid x)$ – імовірність того, що запис z, представлений вектором ознак x, належить до класу y_k. Важливою складовою предметної області є не лише визначення типу вразливості, а й оцінювання її реального ризику для конкретного вебзастосунку. Формально ризик подається функцією

$$R : Z \rightarrow [0;\ 10], \tag{2.12}$$

де *R(z)* – інтегральна оцінка критичності вразливості.

З огляду на те, що базове значення CVSS не завжди відображає реальну небезпеку вразливості в конкретному середовищі, у роботі використовується розширена базова модель ризику

$$R(zi) = \varphi(\widetilde{v_i^{cvss}}, p_i^{ai}, b_i^{ctx}, e_i^{exp}, q_i^{corr}), \qquad R(zi) : Z \rightarrow [0;\ 10], \tag{2.13}$$

де $\widetilde{v_i^{cvss}}$ – нормалізоване значення CVSS для і-го запису;

p_i^{ai} – імовірність, отримана AI-моделлю для і-го запису;

b_i^{ctx} – бізнес-критичність компонента;

e_i^{exp} – оцінка експлуатованості;

q_i^{corr} – рівень міжджерельного підтвердження;

Детальна адаптивна модель ризикового оцінювання, включаючи вагові коефіцієнти, нелінійне коригування та врахування епістемічної невизначеності, розглядається у підрозділі 2.5. На основі оціненого ризику формується ранжований перелік загроз:

$$O_i = sort_{desc}(\{\langle z, C(z), R(z)\rangle | z \in Z_i\}), \tag{2.14}$$

де Z_i – множина нормалізованих записів для w_i – вебзастосунку

O_i – впорядкований за спаданням ризику перелік вразливостей.

Таким чином, у формалізованому вигляді предметна область містить такі базові сутності: вебзастосунок, артефакт вебзастосунку, джерело аналізу, первинну знахідку, нормалізований запис, клас вразливості, інтегральну оцінку ризику та ранжований перелік загроз.

На основі введених позначень задача моделювання ІКС АВВЗ формулюється так: для кожного вебзастосунку w_iз множини W, за сукупністю неоднорідних

результатів аналізу F_i^{raw}, необхідно побудувати таку сукупність моделей і методів, яка забезпечує інтелектуальну нормалізацію та дедуплікацію первинних результатів, класифікацію знайдених вразливостей за стандартизованими класами загроз, адаптивне оцінювання ризику та формування впорядкованого переліку пріоритетних загроз для подальшого усунення.

У термінах функціонального моделювання це відповідає композиції відображень:

$$F_i^{raw} \rightarrow N \rightarrow Z_i \rightarrow C \rightarrow Y_i \rightarrow R \rightarrow O_i, \quad (2.15)$$

З точки зору ефективності функціонування системи задача має багатокритеріальний характер. Узагальнений цільовий функціонал подається як:

$$J = w_1 F1 + w_2 Prec + w_3 Rec + w_4 Corr - w_5 Dup - w_6 T, \quad (2.16)$$

де $F1$ – F1 міра класифікації;

$Prec$ – прецизійність;

Rec – повнота;

$Corr$ – рівень кореляції між результатами різних джерел;

Dup – частка дубльованих повідомлень;

T – середній час обробки;

$w_1 \ldots w_6$ – вагові коефіцієнти критерію оптимізації.

Отже, моделювання ІКС АВВЗ спрямоване на формальне перетворення різнорідних безпекових результатів у змістовний аналітичний висновок. Оскільки задача є багатоджерельною, багатокритеріальною та контекстно залежною, її розв’язання потребує взаємопов’язаних моделей нормалізації, класифікації та оцінювання ризику. Формалізована модель предметної області наведена на рисунку 2.1 і є основою для подальшого моделювання системи.

Рисунок 2.1 – Формалізована модель предметної області ІКС АВВЗ

2.2 Узагальнена архітектурно-функціональна модель ІКС АВВЗ

Побудова інтелектуальної комп'ютерної системи автоматичного виявлення вразливостей вебзастосунків та класифікації загроз потребує визначення її концептуальної архітектурно-функціональної моделі. У межах цього підрозділу система розглядається на рівні функціональних рівнів, модулів і потоків даних без деталізації програмної реалізації, яка розглядається у четвертому розділі.

Узагальнена модель ІКС АВВЗ має забезпечувати повний цикл перетворення первинних результатів аналізу безпеки у впорядкований перелік пріоритетних загроз. Такий цикл охоплює приймання даних із різнорідних джерел, попередню обробку, інтелектуальну нормалізацію, класифікацію, кореляцію, оцінювання ризику та формування результатів для подальшого реагування [7].

У загальному вигляді архітектурно-функціональну модель системи подано як:

$$A =< L, M, D, K, I, O^{out} >, \tag{2.17}$$

де L – множина архітектурних рівнів;

M – множина функціональних модулів;

D – сукупність потоків даних;

K – база знань і зовнішні довідкові джерела;

I – множина вхідних впливів;

O^{out} – множина вихідних результатів.

Множина архітектурних рівнів визначається як

$$L = \{L_1, L_2, L_3, L_4, L_5\}, \tag{2.18}$$

де L_1 – рівень збору та інтеграції даних;

L_2 – рівень попередньої обробки та нормалізації;

L_3 – рівень інтелектуального аналізу і класифікації;

L_4 – рівень оцінювання ризику та кореляції;

L_5 – рівень візуалізації, звітності та інтеграції із зовнішніми процесами.

Рівень L_1 забезпечує приймання та інтеграцію результатів аналізу з різнорідних джерел, зокрема SAST-, DAST-, IAST-, SCA-засобів, журналів подій, HTTP-артефактів і результатів перевірки конфігурацій. Його основне призначення полягає у формуванні єдиного вхідного потоку даних для подальшої обробки.

Рівень L_2 виконує попередню обробку та нормалізацію отриманих результатів. На цьому етапі усувається структурна й семантична неоднорідність записів, виконується очищення даних, виділення ключових атрибутів, первинна дедуплікація та зіставлення з базами знань.

Рівень L_3 відповідає за інтелектуальну класифікацію нормалізованих записів. Його функція полягає у визначенні класу вразливості, формуванні ймовірнісного профілю загрози та передаванні результатів до модуля кореляції й ризикового оцінювання.

Рівень L_4 забезпечує кореляцію, підтвердження та оцінювання ризику. На цьому рівні враховуються результати класифікації, бізнес-контекст, експлуатованість, міжджерельне підтвердження та фактори критичності.

Рівень L_5 відповідає за формування підсумкових результатів, зокрема аналітичних звітів, структурованих відповідей API, службових сповіщень та даних для подальшого реагування [68]. Множина функціональних модулів системи задається як

$$M = \{m_{scan}, m_{ing}, m_{norm}, m_{kb}, m_{cls}, m_{corr}, m_{risk}, m_{rep}, m_{api}, m_{mon}\}. \quad (2.19)$$

Формально вхідний потік даних подано як:

$$D_{in} = \bigcup_{j=1}^{M_s} D_{j,} \quad (2.20)$$

де D_j – потік даних від j – го джерела аналізу;

M_s – кількість активних джерел.

Основні перетворення даних у межах архітектурно-функціональної моделі подано як:

$$Z = N(D_{in}, K), \quad (2.21)$$

$$Y = C(Z, T), \quad (2.22)$$

$$\Omega = R(C(Z, T), B, E, Q), \quad (2.23)$$

де N – оператор нормалізації;

K – сукупність зовнішніх знань і довідкових структур.

C – оператор інтелектуальної класифікації;

T – набір параметрів навченої моделі.

B – бізнес-контекст застосунку;

E – параметри експлуатованості;

Q – оцінка рівня підтвердження різними джерелами;

Ω – впорядкована множина загроз.

$$O = \{o_{dash}, o_{report}, o_{api}, o_{alert}\}, \quad (2.24)$$

де o_{dash} – інтерактивні візуалізації;

o_{report} – аналітичні звіти;

o_{api} – структуровані відповіді API;

o_{alert} – попередження та сигнали для зовнішніх систем.

Архітектурну модель ІКС АВВЗ доцільно подати у вигляді орієнтованого графа

$$G_A = < V_A, E_A >, \tag{2.25}$$

де $V_A = M$ – множина функціональних модулів;

E_A – множина інформаційних і керувальних зв’язків між ними.

Запропонована архітектурно-функціональна модель ґрунтується на принципах модульності, масштабованості, інтероперабельності, адаптивності та пояснюваності результатів. Кожний компонент має окрему функціональну відповідальність, що забезпечує можливість незалежного розвитку підсистем, підключення нових джерел аналізу, зміни моделей класифікації та уточнення ризикового оцінювання без повної перебудови системи. Такий підхід узгоджується з дослідженнями архітектурних варіантів багатокомп’ютерних систем із пастками та приманками, у яких критерії централізації визначають придатність системи до масштабування, координації компонентів і підвищення стійкості процесу виявлення атак [68]. Окремий напрям розвитку таких архітектур становлять багатокомп’ютерні системи виявлення шкідливого програмного забезпечення з метаморфною функціональністю, що підтверджує доцільність модульної побудови підсистем аналізу та можливість розширення набору джерел без повної перебудови всієї системи [69]. Узагальнену архітектурно-функціональну модель ІКС АВВЗ наведено на рисунку 2.2.

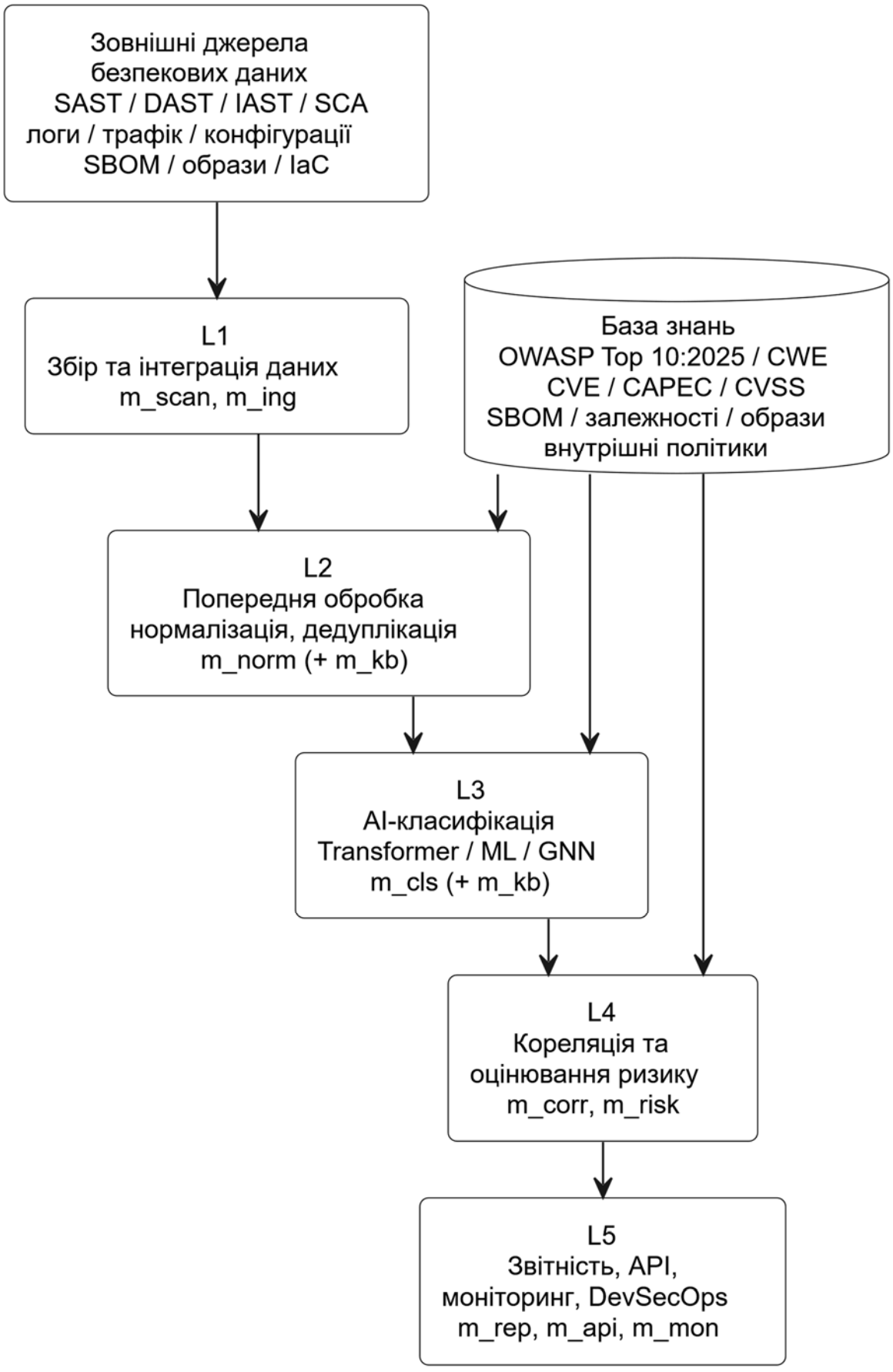


Рисунок 2.2 – Узагальнена архітектурно-функціональна модель ІКС АВВЗ

Таблиця 2.1 – Основні функціональні модулі ІКС АВВЗ

Модуль	Рівень	Позначення	Основне призначення
Модуль взаємодії зі сканерами	(L_1)	(m_{scan})	Підключення до SAST, DAST, IAST, SCA та інших джерел
Модуль збору даних	(L_1)	(m_{ing})	Приймання, маршрутизація та первинна перевірка результатів
Модуль нормалізації	(L_2)	(m_{norm})	Уніфікація, очищення, семантичне збагачення та дедуплікація
Модуль баз знань	(L_2, L_3)	(m_{kb})	Доступ до OWASP Top 10:2025, CWE, CVE, CAPEC, CVSS, SBOM, метаданих залежностей, контейнерних образів та внутрішніх політик безпеки.
Модуль класифікації	(L_3)	(m_{cls})	Класифікація вразливостей і загроз на основі інтелектуальних методів
Модуль кореляції	(L_4)	(m_{corr})	Підтвердження та узгодження результатів з різних джерел
Модуль ризикового оцінювання	(L_4)	(m_{risk})	Розрахунок інтегрального ризику та ранжування загроз
Модуль звітності	(L_5)	(m_{rep})	Формування аналітичних звітів і представлення результатів
API-модуль	(L_5)	(m_{api})	Інтеграція із зовнішніми системами та конвеєрами безпечної розробки
Модуль моніторингу	крос-рівневий	(m_{mon})	Журналювання, контроль стану та технічний моніторинг

В таблиці 2.1 показано детальне функціональне призначення модулів m та їх

розподіл за архітектурними рівнями L . Видно, що запропонована модель описує ІКС АВВЗ як багаторівневу модульну систему, у якій результати різнорідних засобів аналізу проходять послідовний шлях від приймання та нормалізації до інтелектуальної класифікації, кореляції, оцінювання ризику й формування підсумкових результатів. Вона створює основу для подальшої побудови моделей нормалізації, класифікації та ризикового оцінювання, що розглядаються у наступних підрозділах.

2.3 Модель інтелектуальної нормалізації результатів сканування

Однією з ключових проблем побудови ІКС АВВЗ є неоднорідність вхідних результатів аналізу. Різні інструменти безпеки формують повідомлення у відмінних форматах, з різним рівнем деталізації, термінологією, правилами оцінювання критичності та структурою технічних атрибутів. Унаслідок цього навіть одна й та сама вразливість може бути представлена кількома формально різними записами, що ускладнює кореляцію, класифікацію, оцінювання ризику та подальше прийняття рішень.

У межах цієї роботи під інтелектуальною нормалізацією результатів сканування розуміється багатокроковий процес перетворення сирих повідомлень від різних джерел аналізу у стандартизовані, дедуплікові, семантично збагачені та придатні до машинної обробки записи.

Множину сирих результатів для вебзастосунку w_i задамо як

$$F_i^{raw} = \{f_1, f_2, \dots, f_n\}, \tag{2.26}$$

де f_k – окреме повідомлення сканера, логічний запис або безпекова подія.

Кожне таке повідомлення подається у вигляді первинного кортежу

$$f_k = \langle src_k, fmt_k, loc_k, desc_k, sev_k, meta_k \rangle, \tag{2.27}$$

де src_k – джерело повідомлення;

fmt_k – формат подання;

loc_k – локалізація проблеми;

$desc_k$ – текстовий опис;

sev_k – первинна оцінка критичності;

$meta_k$ – додаткові метадані.

Метою нормалізації є побудова відображення

$$N: F_i^{raw} \rightarrow Z_i, \tag{2.28}$$

де Z_i – множина стандартизованих записів, придатних до подальшого інтелектуального аналізу. У загальному вигляді оператор нормалізації доцільно подати як композицію кількох послідовних перетворень

$$N = N_6 \circ N_5 \circ N_4 \circ N_3 \circ N_2 \circ N_1, \tag{2.29}$$

де N_1 – синтаксичний розбір і виділення атрибутів;

N_2 – уніфікація структури запису;

N_3 – семантичне зіставлення і класифікаційне збагачення;

N_4 – дедуплікація і кореляція подібних записів;

N_5 – побудова векторного представлення ознак;

N_6 – формування фінального нормалізованого запису.

На першому етапі виконується розбір результатів сканування залежно від їхнього джерела та формату. Це дозволяє перевести XML-, JSON-, CSV-, HTML- або текстові повідомлення в уніфіковану проміжну форму. Для результатів SCA та перевірки ланцюга постачання додатково нормалізуються відомості про назву пакета, версію компонента, тип екосистеми, ідентифікатор пакета, транзитивність залежності, наявність відповідного CVE, CVSS-оцінку, SBOM-запис, контейнерний образ або build-артефакт. Це дозволяє розглядати ризики

залежностей не як допоміжні повідомлення, а як повноцінні об’єкти подальшої класифікації та ризик-орієнтованого ранжування.

$$p_k = N_1(f_k) = \langle s_k, l_k, d_k, v_k, m_k \rangle, \quad (2.30)$$

де P_k – проміжне представлення запису;

s_k – ідентифікатор джерела;

l_k – локалізація;

d_k – очищений текстовий опис;

v_k – початковий рівень критичності;

m_k – службові атрибути.

На цьому етапі здійснюється очищення службових символів, виділення URL, параметрів, шляхів, ідентифікаторів компонентів, нормалізація позначень рівнів критичності та вилучення технічного шуму, що не має змістового значення для подальшої класифікації.

Другий етап спрямований на приведення всіх повідомлень до єдиної внутрішньої схеми.

$$u_k = N_2(p_k) = \langle id_w, a_k, t_k, q_k, h_k, e_k \rangle, \quad (2.31)$$

де id_w – ідентифікатор вебзастосунку;

a_k – уражений артефакт;

t_k – нормалізований текстовий опис;

q_k – нормалізований рівень критичності;

h_k – структурні атрибути;

e_k – технічні докази.

Уніфікація забезпечує спільний набір базових полів для записів, отриманих із різних сканерів, що створює основу для подальшого машинного зіставлення, кореляції та векторизації.

Третій етап є семантичним збагаченням запису. Його призначення полягає у зіставленні текстового опису та технічних атрибутів із відомими категоріями OWASP, CWE, CVE, CAPEC та іншими таксономічними сутностями:

$$c_k = N_3\,(u_k, K), \tag{2.32}$$

де K – база знань;

c_k – збагачений запис.

Після цього запис подається так:

$$c_k = \langle id_w, a_k, t_k, q_k, cwe_k, cve_k, owasp_k, capec_k, \mu_k \rangle, \tag{2.33}$$

де $cwe_k,\ cve_k,\ owasp_k,\ capec_k$ – ідентифікатори зіставлених знань;

$\mu_k \in [0; 1]$ – рівень достовірності семантичного зіставлення, який доцільно обчислювати на основі комбінованої функції подібності

$$\mu_k = \lambda_1 sim_{txt}(t_k, K) + \lambda_2 sim_{loc}(a_k, K) + \lambda_3 sim_{attr}(h_k, K), \tag{2.34}$$

де sim_{txt} – текстова подібність;

sim_{loc} – подібність за локалізацією;

sim_{attr} – подібність за структурними атрибутами;

$\lambda_1 + \lambda_2 + \lambda_3 = 1$.

Четвертий етап спрямований на усунення дубльованих результатів і об’єднання повідомлень, які описують одну й ту саму вразливість. Для двох записів c_i та c_j визначено функцію інтегральної подібності, тоді

$$S(c_i, c_j) = \alpha_1 sim_{txt}(c_i, c_j) + \alpha_2 sim_{loc}(c_i, c_j) + \alpha_3 sim_{type}(c_i, c_j) + \alpha_4 sim_{src}(c_i, c_j), \tag{2.35}$$

де $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = 1$.

У цьому випадку критерій дедуплікації можна подати як

$$c_i \sim c_j \Leftrightarrow S(c_i, c_j) \geq \tau, \tag{2.36}$$

де τ – порогове значення схожості.

Усі записи, що задовольняють умову еквівалентності, утворюють кластер повідомлень про одну потенційну вразливість

$$G_r = \{c_k \in C_i \mid c_k \sim c_r\}, \tag{2.37}$$

де $C_i = c_1, c_2, \dots, c_n$ – множина збагачених записів для вебзастосунку w_i.

Такий кластер надалі використовується для побудови агрегованого запису

$$d_r = Agg(G_r), \tag{2.38}$$

де Agg – оператор агрегації, що об’єднує джерела підтвердження, найбільш повний опис, технічні докази, скориговану оцінку критичності та узгоджену категорію загрози. Після дедуплікації множина записів має вигляд:

$$D_i = \{d_1, d_2, \dots, d_m\}, \qquad m \leq n. \tag{2.39}$$

На п’ятому етапі кожен агрегований запис d_r перетворюється у вектор ознак, необхідний для роботи моделей машинного та глибинного навчання. Оператор векторизації має вигляд

$$x_r = N_{5(d_r)}, \tag{2.40}$$

де $x_r \in R^m$ – вектор ознак фіксованої розмірності, який у загальному випадку такий вектор можна подати як

$$x_r = \langle x_r^{txt}, x_r^{struct}, x_r^{src}, x_r^{ctx}, x_r^{sev} \rangle, \quad (2.41)$$

де x_r^{txt} – текстові ембедінги опису;

x_r^{struct}- структурні ознаки локалізації;

x_r^{src}- ознаки джерел підтвердження;

x_r^{ctx}- контекстні ознаки середовища;

x_r^{sev} – ознаки первинної критичності й технічних доказів.

На завершальному етапі оператор N_6 формує фінальний нормалізований запис:

$$z_r = N_{6(d_r, x_r)}, \quad (2.42)$$

де

$$z_r = \langle id_w, a_r, x_r, cwe_r, cve_r, capec_r, owasp_r, cvss_r, sbom_r, src_r, conf_r \rangle, \quad (2.43)$$

Повна модель інтелектуальної нормалізації подається як ланцюг перетворень:

$$F_i^{raw} \to N_1 \to P_i \to N_2 \to U_i \to N_3 \to C_i \to N_4 \to D_i \to N_5 \to X_i \to N_6 \to Z_i, \quad (2.44)$$

де P_i, U_i, C_i, D_i, X_i, Z_i – множини проміжних результатів відповідних етапів нормалізаційного контуру.

Якість інтелектуальної нормалізації доцільно оцінювати за такими критеріями: мінімізація дублювання результатів, точність зіставлення з таксономіями OWASP, CWE і CVE, збереження релевантної змістової інформації та інформативність ознакового представлення для класифікатора. Узагальнену цільову функцію якості нормалізації подано як:

$$J_N = b_1 \cdot Q_{map} + b_2 \cdot Q_{dedup} + b_3 \cdot Q_{sem} + b_4 \cdot Q_{vec} - b_5 \cdot L_{noise}, \quad (2.45)$$

де Q_{map} – якість зіставлення з базами знань;

Q_{dedup} – якість дедуплікації;

Q_{sem} – повнота семантичного збагачення;

Q_{vec} – інформативність ознакового представлення;

L_{noise} – залишковий рівень шуму.

Таким чином, інтелектуальна нормалізація реалізується як послідовність взаємопов’язаних етапів, що забезпечують перехід від неоднорідних сирих повідомлень до стандартизованого простору ознак для подальшої класифікації та ризикового оцінювання.

Модель інтелектуальної нормалізації результатів сканування, що охоплює етапи розбору, уніфікації, семантичного збагачення, дедуплікації, векторизації та формування фінального нормалізованого запису, наведено на рисунку 2.3.

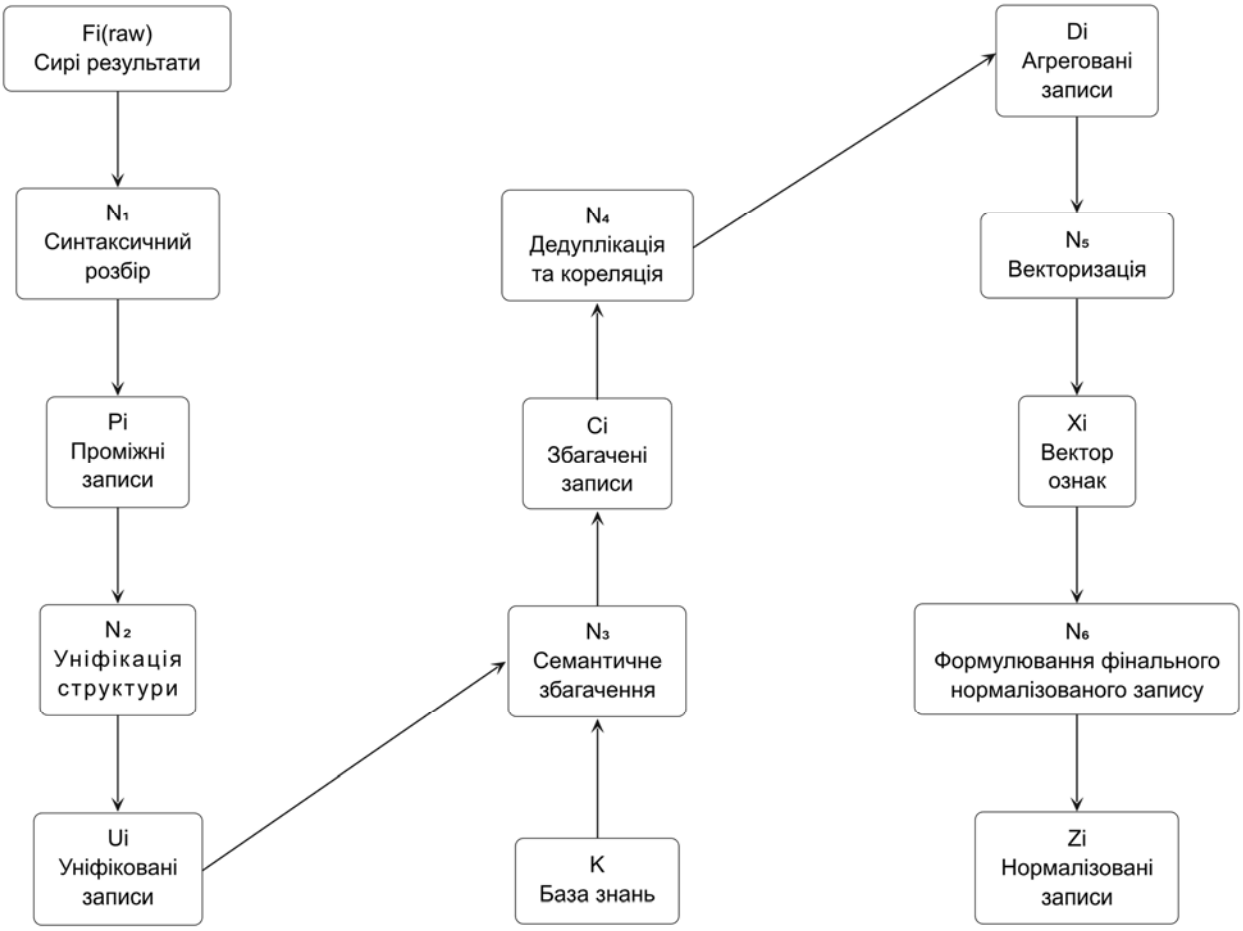


Рисунок 2.3 – Модель інтелектуальної нормалізації результатів сканування

Наведена модель показує, що інтелектуальна нормалізація є проміжною ланкою між етапом збору безпекових результатів і подальшим етапом інтелектуальної класифікації, оскільки саме вона забезпечує перетворення неоднорідних повідомлень у придатний до аналітичної обробки простір ознак.

2.4 Модель інтелектуальної класифікації вразливостей і загроз

Після виконання інтелектуальної нормалізації результатів сканування наступним ключовим етапом функціонування ІКС АВВЗ є класифікація вразливостей і кіберзагроз. На цьому етапі здійснюється перехід від стандартизованого, але ще неінтерпретованого запису до формалізованого висновку щодо типу вразливості, її належності до певного класу загроз та потенційного впливу на вебзастосунок.

У межах даної роботи класифікація розглядається як задача відображення множини нормалізованих записів у множину класів загроз:

$$Z = \{z_1, z_2, \dots, z_n\}, \tag{2.46}$$

де Z – множина нормалізованих записів, а

$$Y = \{y_1, y_2, \dots, y_K\}, \tag{2.47}$$

де Y – множина допустимих класів вразливостей і кіберзагроз.
Тоді задача класифікації полягає у побудові відображення

$$C : Z \to Y, \tag{2.48}$$

Кожен нормалізований запис z_i формується як структурований об’єкт, що містить текстові, структурні, контекстні та джерельні ознаки. Формально такий запис можна подати як

$$z_i = \langle x_i^{txt}, x_i^{str}, x_i^{ctx}, x_i^{src}, x_i^{aux} \rangle, \quad (2.49)$$

де x_i^{txt} – текстові ознаки опису вразливості;

x_i^{str} – структурні ознаки локалізації та типу артефакту;

x_i^{ctx} – контекстні ознаки середовища виконання;

x_i^{src} – ознаки джерел підтвердження;

x_i^{aux} – додаткові технічні атрибути.

Для структурної компоненти класифікації важливими є не лише загальні характеристики артефакту, а й низькорівневі програмні дефекти, зокрема вразливості переповнення буфера, для яких застосування методів штучного інтелекту може підвищувати точність виявлення потенційно небезпечних шаблонів коду [71]. Після векторизації всі ознаки об'єднуються в єдиний вектор представлення

$$z_i = [x_i^{txt} | x_i^{str} | x_i^{ctx} | x_i^{src} | x_i^{aux}], \quad (2.50)$$

де | означає операцію конкатенації ознак. У найпростішому випадку одноетикеткової класифікації модель визначається як функція. Таке подання забезпечує можливість сумісного використання ознак різної природи в межах єдиного класифікаційного контуру.

$$\hat{y}_i = \underset{y_k \in Y}{arg\,max}\, P(y_k | x_i, T), \quad (2.51)$$

де T – набір параметрів навченої моделі;

$P(y_k | x_i, T)$ – умовна імовірність того, що запис x_i належить до класу y_k.

У багатомітковому випадку, коли один запис може одночасно належати до кількох класів загроз, класифікація подається як

$$\hat{y}_i = \sigma\big(f(x_i, T)\big), \quad (2.52)$$

де $\hat{y}_i$ – вектор імовірностей за всіма класами;

$f(.)$ – вихід моделі до порогування;

σ – сигмоїдна функція активації.

Рішення про належність до класу y_k приймається за правилом

$$\hat{y}_{ik} = \begin{cases} 1, & p_{ik} \geq \tau_k \\ 0, & p_{ik} < \tau_k \end{cases}, \tag{2.53}$$

де τ_k – порогове значення для k-го класу.

З огляду на специфіку предметної області класифікація вразливостей не повинна спиратися лише на один тип ознак або одну модель. Саме тому в роботі запропоновано гібридну модель класифікації, що поєднує кілька спеціалізованих підмодулів аналізу. Узагальнено гібридну модель можна подати як

$$C(x_i) = F\left(C_{txt}(x_i^{txt}), C_{str}(x_i^{str}), C_{ctx}(x_i^{ctx}), C_{src}(x_i^{src})\right), \tag{2.54}$$

де C_{txt} – текстовий класифікатор;

C_{str} – структурний класифікатор;

C_{cxt} – контекстний класифікатор;

C_{src}- класифікатор, що враховує джерела підтвердження;

F – агрегуючий механізм прийняття остаточного рішення.

Текстовий компонент C_{txt} працює з описом вразливості, поясненнями сканерів, ідентифікаторами, частинами HTTP-повідомлень, назвами загроз та іншими символьними представленнями. Нехай $e_{txt,i}$ – векторне подання текстового опису. Тоді текстовий класифікатор можна подати як

$$p_{txt,i} = softmax\left(W_{txt} \cdot e_{txt,i} + b_{txt}\right), \tag{2.55}$$

де W_{txt} і b_{txt} – параметри моделі, а $p_{txt,i} \in [0;1]^K$ – вектор імовірностей за класами.

Структурний компонент C_{str} орієнтований на ознаки, пов'язані з локалізацією та технічним контекстом дефекту. Такий підхід особливо корисний тоді, коли тип загрози визначається не стільки самим текстом повідомлення, скільки місцем виникнення проблеми: у контролері, API-ендпоінті, механізмі авторизації, файлі конфігурації, шаблоні HTML, контейнері або залежності.

Нехай x_i^{str} – вектор структурних ознак. Тоді

$$p_{str,i} = softmax(W_{str} \cdot x_i^{str} + b_{str}), \tag{2.56}$$

Контекстний компонент C_{ctx} враховує, у якому середовищі й у якій бізнес-функції виникає вразливість. Це важливо, оскільки однакові технічні сигнали в різних контекстах можуть означати різні класи ризику.

Нехай x_i^{ctx} – вектор контекстних ознак. Тоді

$$p_{ctx,i} = softmax(W_{ctx} \cdot x_i^{ctx} + b_{ctx}), \tag{2.57}$$

Окремий інтерес становить аналіз того, якими саме джерелами підтверджується вразливість. Якщо одна й та сама проблема зафіксована лише одним інструментом, це одна ситуація; якщо той самий дефект незалежно підтвердили SAST, DAST і SCA, достовірність класифікації суттєво зростає.

Нехай x_i^{src} – вектор ознак джерельного підтвердження. Тоді

$$p_{src,i} = softmax(W_{src} \cdot x_i^{src} + b_{src}), \tag{2.58}$$

Фінальне рішення моделі формується шляхом об'єднання результатів кількох підмодулів [47]. Один із доцільних способів агрегування – зважене усереднення

$$p_i = w_1 \cdot p_{txt,i} + w_2 \cdot p_{str,i} + w_3 \cdot p_{ctx,i} + w_4 \cdot p_{src,i}, \tag{2.59}$$

де $w_1 + w_2 + w_3 + w_4 = 1$, а p_i – підсумковий вектор імовірностей за класами загроз. Зважене агрегування дозволяє враховувати внесок кожного підмодуля залежно від його інформативності та ролі в конкретному сценарії класифікації.

Остаточний клас визначається як

$$\hat{y}_i = \underset{k}{arg\,max}\, p_{ik}. \tag{2.60}$$

Для навчання моделі класифікації доцільно використовувати крос-ентропійну функцію втрат. У випадку одноетикеткової класифікації вона має вигляд

$$L_{cls} = -\sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \cdot log(p_{ik}), \tag{2.61}$$

де $y_{ik} \in 0; 1$ – істинна належність об'єкта і до класу k, а p_{ik}- оцінена моделлю імовірність. Для багатоміткової класифікації доцільно використовувати бінарну крос-ентропію

$$L_{multi} = -\sum_{i=1}^{N} \sum_{k=1}^{K} [y_{ik} \cdot log(\hat{y}_{ik}) + (1 - y_{ik}) \cdot log(1 - \hat{y}_{ik})], \tag{2.62}$$

У випадку незбалансованості класів, що є характерною для задач кібербезпеки, доцільно використовувати вагові коефіцієнти класів

$$\mathrm{L_w} = -\sum_{\mathrm{i=1}}^{\mathrm{N}} \sum_{\mathrm{k=1}}^{\mathrm{K}} \alpha_{\mathrm{k}} \cdot \mathrm{y_{ik}} \cdot \log(\mathrm{p_{ik}}), \tag{2.63}$$

де α_k – ваговий коефіцієнт для класу k;

$y_{(ik)}$ – істинна належність об'єкта і до класу k;

$p_{(ik)}$ – оцінена моделлю ймовірність належності до класу k.

Оскільки задача виявлення вразливостей є задачею з асиметричною ціною помилки, звичайна точність не є достатньою. Тому якість моделі доцільно

оцінювати за такими показниками: прецизійності, повноти, F1-міра, макро-F1 та матриця помилок між класами загроз. Для кожного класу y_k маємо

$$Precision_k = \frac{TP_k}{(TP_k + FP_k)}, \quad (2.64)$$

$$Recall_k = \frac{TP_k}{(TP_k + FN_k)}, \quad (2.65)$$

$$F1_k = 2 \cdot Precision_k \cdot \frac{Recall_k}{(Precision_k + Recall_k)}, \quad (2.66)$$

Інтегральний критерій якості можна подати як

$$J_C = n_1 \cdot F1_{macro} + n_2 \cdot Precision_{macro} + n_3 \cdot Recall_{macro} - n_4 \cdot Err_{crit}, \quad (2.67)$$

де Err_{crit} – штраф за помилки у найбільш критичних класах загроз.

$n_1 - n_4$ – вагові коефіцієнти.

У структурі ІКС АВВЗ модель класифікації перетворює нормалізовані технічні записи на змістові висновки про клас загрози, формує ймовірнісний профіль вразливості та передає до ризикового модуля як результат класифікації, так і рівень упевненості. Гібридну модель класифікації, що поєднує текстовий, структурний, контекстний і джерельний підмодулі аналізу, наведено на рисунку 2.4.

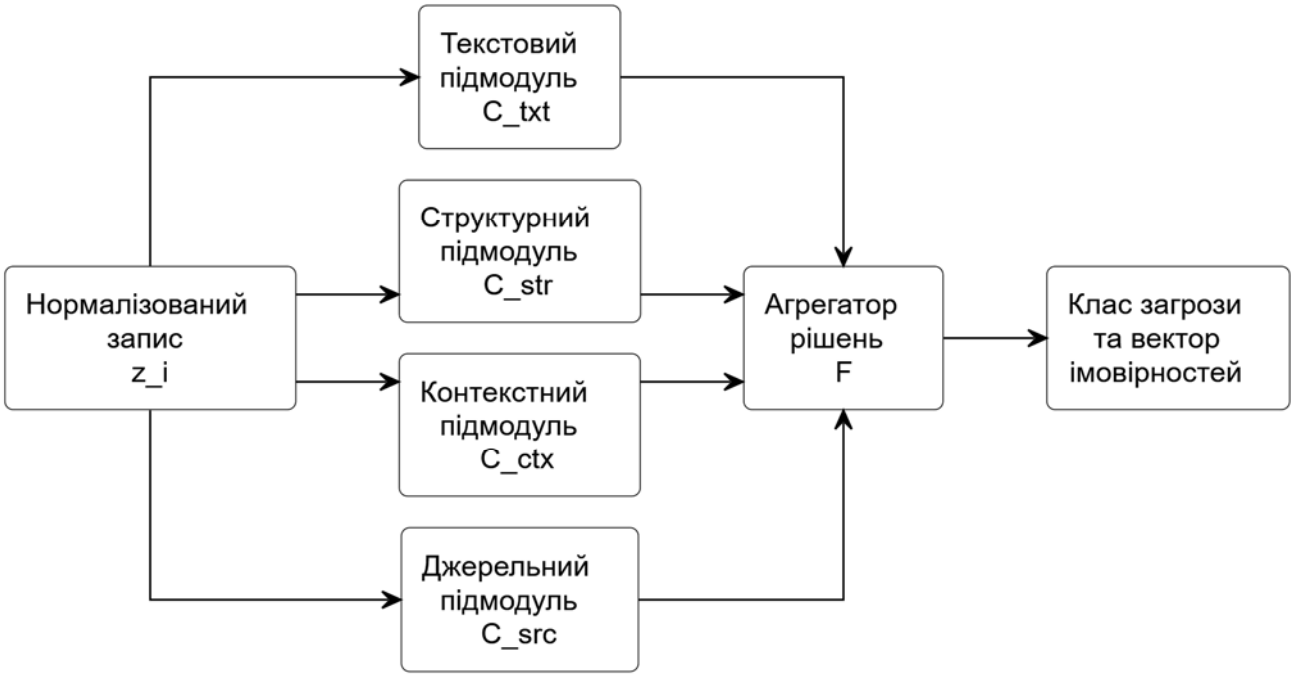


Рисунок 2.4 – Гібридна модель класифікації вразливостей і кіберзагроз

Подана структура моделі відображає доцільність поєднання кількох типів ознак і аналітичних підмодулів у межах єдиного класифікаційного контуру, що підвищує точність розпізнавання загроз і зменшує чутливість до шуму вхідних даних.

Таблиця 2.2 – Компоненти гібридної моделі класифікації

Компонент	Позначення	Функція
Текстовий підмодуль	C_{txt}	Аналіз описів, технічних повідомлень і параметрів запитів
Структурний підмодуль	C_{str}	Аналіз типу артефакту, локалізації та технічної структури
Контекстний підмодуль	C_{ctx}	Урахування середовища, ролі компонента та бізнес-контексту
Джерельний підмодуль	C_{src}	Урахування кількості та типу джерел підтвердження
Агрегатор рішень	F	Формування фінального ймовірнісного рішення

З таблиці 2.2 видно, що запропонована модель класифікації має комбінований характер і поєднує кілька незалежних аналітичних контурів, що підвищує стійкість системи до шуму та неповноти вхідних даних.

2.5 Модель адаптивного оцінювання ризику з урахуванням невизначеності

Після класифікації вразливостей і кіберзагроз наступним критично важливим етапом функціонування ІКС АВВЗ є оцінювання ризику, оскільки саме на цьому етапі система переходить від визначення типу загрози до оцінювання її фактичної небезпеки для конкретного вебзастосунку. У практиці кібербезпеки одна й та сама вразливість може мати різну значущість залежно від контексту експлуатації, критичності відповідного компонента, наявності публічного експлойту, кількості джерел підтвердження, архітектурної ролі вразливого модуля та потенційного

бізнес-впливу. Саме тому у межах даної роботи пропонується адаптивна модель оцінювання ризику, яка поєднує формальні метрики CVSS з результатами інтелектуальної класифікації, контекстними характеристиками середовища та показниками підтвердження вразливості [6, 20, 60].

У загальному вигляді ризик вразливості доцільно розглядати як функцію від множини технічних, імовірнісних і контекстних параметрів. Для кожного нормалізованого та класифікованого запису z_i задано вектор оцінювальних параметрів

$$r_i = \langle v_i^{cvss}, p_i^{ai}, b_i^{ctx}, e_i^{exp}, q_i^{corr}, s_i^{sens}, k_i^{cls}, m_i^{mit} \rangle, \quad (2.68)$$

де v_i^{cvss} – базове значення критичності за CVSS;

p_i^{ai} – імовірність належності до класу загроз, отримана класифікатором;

b_i^{ctx} – показник бізнес-критичності компонента;

e_i^{exp} – оцінка експлуатованості;

q_i^{corr} – рівень міжджерельного підтвердження;

s_i^{sens} – рівень чутливості даних або сервісу;

k_i^{cls} – клас-специфічна вага загрози, що враховує належність вразливості до категорії OWASP Top 10:2025, класу CWE, типового операційного впливу та актуальності відповідної категорії для сучасних API-центричних, мікросервісних, контейнеризованих і supply chain-залежних вебзастосунків.

m_i^{mit} – оцінка сили компенсуючих механізмів захисту.

Тоді адаптивна оцінка ризику визначається як відображення

$$R: Z \rightarrow [0;\ 10], \quad (2.69)$$

де $R(z_i)$ – інтегральна оцінка ризику конкретної вразливості [60].

У формулах (2.68) – (2.77) подано базову аналітичну модель адаптивного оцінювання ризику. У фінальній реалізації програмного прототипу цю модель

розширено до восьмифакторної схеми за рахунок явного врахування додаткових контекстних параметрів, механізмів міжджерельного підтвердження та політики інтерпретації прикордонних випадків підвищеної епістемічної невизначеності [65].

Методологія CVSS є базовим міжнародним стандартом кількісного оцінювання вразливостей. Вона дозволяє формалізувати технічну складність атаки, тип впливу на конфіденційність, цілісність і доступність та низку інших параметрів. Проте CVSS має фундаментальне обмеження: він оцінює переважно технічну критичність самої вразливості, але не завжди відображає реальний ризик для конкретного застосунку. Саме тому в ІКС АВВЗ базове значення CVSS використовується лише як один із компонентів підсумкової оцінки. Для подальших обчислень доцільно перейти до нормалізованого значення CVSS

$$\tilde{v}_{cvss,i} = \frac{v_{cvss,i}}{10} \qquad \tilde{v}_{cvss,i} \in [0;1] \tag{2.70}$$

Другим важливим компонентом є впевненість моделі класифікації у правильності розпізнавання загрози. Нехай

$$p_i^{ai} = \max_k P(\, y_k \mid x_i, T\,), \qquad p_i^{ai} \in [0;1] \tag{2.71}$$

де $p_i^{ai} \in [0;1]$ – максимальна імовірність, отримана класифікатором для запису z_i.

Бізнес-критичність відображає роль уразливого компонента в загальній логіці функціонування вебзастосунку. Вона може задаватися експертно, політикою організації або обчислюватися на основі архітектурної ролі сервісу. Позначимо цей показник як

$$b_i^{ctx} \in [0;1], \tag{2.72}$$

де значення, близькі до 1, відповідають компонентам високої бізнес-важливості: модулям автентифікації, підсистемам обробки платежів, компонентам з доступом

до конфіденційних даних, API керування привілеями, адміністративним інтерфейсам. Оцінка експлуатованості задається як

$$e_i^{exp} \in [0; 1], \tag{2.73}$$

Для її формування доцільно враховувати наявність публічного експлойту, простоту експлуатації, необхідність автентифікації, потребу у взаємодії користувача, доступність вразливого інтерфейсу ззовні та можливість автоматизованої атаки. Рівень підтвердження вразливості кількома джерелами подається як

$$q_i^{corr} = \frac{|S_{i,conf}|}{|S_{max}|}, \qquad q_i^{corr} \in [0; 1], \tag{2.74}$$

де $S_{i,conf}$ множина джерел, які підтвердили конкретну вразливість, а S_{max} – множина всіх доступних типів джерел у системі. Тоді $q_{(i,corr)} \in [0; 1]$.

Ще одним фактором є значущість ресурсу, до якого належить уразливий компонент. Рівень чутливості даних і сервісу задається величиною

$$s_i^{sens} \in [0; 1], \tag{2.75}$$

де високі значення відповідають персональним даним користувачів, платіжній інформації, обліковим даним і токенам, адміністративним конфігураціям та службовим API з критичними привілеями.

На практиці реальний ризик може бути знижений, якщо у системі вже діють компенсуючі механізми: WAF, обмеження мережевого доступу, сегментація сервісів, багатофакторна автентифікація, додатковий аудит привілеїв, ізоляція середовищ виконання. Нехай

$$m_i^{mit} \in [0; 1], \tag{2.76}$$

де великі значення відповідають сильним компенсуючим заходам.

Інтегральну оцінку ризику доцільно визначати як зважену функцію, що поєднує технічну критичність, імовірність класифікації, контекстні фактори та компенсаційні механізми захисту.

$$R_i^{base} = 10 \cdot \left(\alpha 1 \cdot \widetilde{v_i^{cvss}} + \alpha 2 \cdot p_i^{ai} + \alpha 3 \cdot b_i^{ctx} + \alpha 4 \cdot e_i^{exp} + \alpha 5 \cdot q_i^{corr} + \alpha 6 \cdot s_i^{sens} + \alpha 7 \cdot k_i^{cls} - \alpha 8 \cdot m_i^{mit}\right), \quad (2.77)$$

де

$$\alpha 1 + \alpha 2 + \alpha 3 + \alpha 4 + \alpha 5 + \alpha 6 + \alpha 7 + \alpha 8 = 1, \quad (2.78)$$

$$\alpha j \geq 0, \qquad j = 1, \dots, 8. \quad (2.79)$$

Для забезпечення коректності оцінки можна використовувати функцію обмеження

$$R_i^* = min\left(10, max\left(0, R_i^{base}\right)\right), \quad (2.80)$$

У ряді випадків проста лінійна модель може бути недостатньою, оскільки деякі фактори мають неадитивний ефект. Наприклад, поєднання високого CVSS, високої бізнес-критичності та наявності публічного експлойту повинно підвищувати ризик сильніше, ніж проста сума цих факторів. Тому доцільно ввести нелінійну коригувальну функцію

$$R_i^{nl} = R_i^* \cdot \left(1 + \rho \cdot \widetilde{v_i^{cvss}} \cdot b_i^{ctx} \cdot e_i^{exp}\right), \qquad \rho \in [0; 1], \quad (2.81)$$

де ρ – коефіцієнт нелінійного посилення. Такий множник посилює вплив комбінації високої технічної критичності, високої бізнес-критичності та високої експлуатованості, що краще відповідає реальній логіці пріоритезації загроз.

Після цього фінальна оцінка ризику визначається як

$$R_i^{final} = min(10, R_i^{nl}). \quad (2.82)$$

Для подальшого використання у системі DevSecOps, звітності та на дашбордах кількісна оцінка ризику має бути перетворена у категоріальне представлення

$$L(z_i) = \begin{cases} \text{Низький}, 0 \le R_i^{final} < 4 \\ \text{Середній}, 4 \le R_i^{final} < 7 \\ \text{Високий}, 7 \le R_i^{final} < 9 \\ \text{Критичний}, 9 \le R_i^{final} \le 10 \end{cases}, \quad (2.83)$$

Після оцінювання ризику для кожного запису система формує впорядкований список загроз

$$O = sort_{desc}(< z_i, y_i, R_i^{final}, L(z_i) >), \quad (2.84)$$

Отриманий впорядкований список загроз може використовуватися як основа для автоматизованого або напівавтоматизованого прийняття рішень у процесах безпечної розробки та безперервного розгортання, а також для підтримки роботи фахівця з безпеки. За наявності зворотного зв'язку від фахівця або експлуатаційного середовища адаптацію ваг моделі можна подати як:

$$w(t + 1) = w(t) + \lambda D(t), \quad (2.85)$$

де $w(t)$ вектор ваг на кроці t;

λ – коефіцієнт навчання,

$D(t)$ – коригувальний вектор.

Ефективність моделі адаптивного оцінювання ризику доцільно оцінювати за такими показниками: здатність розрізняти реально критичні та другорядні загрози,

узгодженість із експертною оцінкою, стійкість до шуму у вхідних даних, здатність знижувати пріоритет хибних спрацювань та інформативність ранжування. Узагальнену функцію якості ризикового модуля можна подати як

$$J_R = \mu_1 Q_{(rank)} + \mu_2 Q_{(expert)} + \mu_3 Q_{(stable)} - \mu_4 FP_{(risk)}, \quad (2.86)$$

де $Q_{(rank)}$ – якість ранжування;

$Q_{(expert)}$ – узгодженість з експертною оцінкою;

$Q_{(stable)}$ – стійкість оцінки;

$FP_{(risk)}$ – частка завищених ризикових оцінок для хибних спрацювань.

Модель адаптивного оцінювання ризику, описана у підрозділі 2.5, визначає детерміновану інтегральну оцінку $R(z_i)$на основі технічних і контекстних параметрів. Проте на практиці ML-класифікатор може бути різною мірою впевнений у своєму рішенні залежно від характеристик вхідного запису. Якщо класифікатор рівномірно розподіляє ймовірності між кількома класами загроз, це вказує на суттєву невизначеність щодо природи загрози. В умовах кібербезпеки така невизначеність має практичний наслідок: при похибці класифікації справжня вразливість може бути недооцінена.

У зв’язку з цим пропонується розширення моделі оцінювання ризику шляхом введення епістемічно-алеаторної декомпозиції. Нехай

$$p = (p_1, p_2, \dots, p_K), \quad (2.87)$$

де p – вектор ймовірностей класів загроз, обчислений класифікатором для запису z_i, а K – кількість класів.

Нормована ентропія Шеннона визначається як

$$H_i = -\sum_{k=1}^{K} p_{ik} \log_2 p_{ik}, \quad (2.88a)$$

$$\widetilde{H_i} = \frac{H_i}{\log_2 K}, \qquad \widetilde{H_i} \in [0;1]. \tag{2.88b}$$

де $\widetilde{H_i}$ – нормована ентропія Шеннона для i – го запису, що характеризує епістемічну невизначеність класифікаційного рішення [64]. При $\widetilde{H_i} \to 0$ класифікатор є впевненим у своєму рішенні, а при $\widetilde{H_i} \to 1$ розподіл імовірностей наближається до рівномірного, що відповідає високій епістемічній невизначеності. У завданнях кібербезпеки вартість хибнонегативного рішення, як правило, істотно перевищує вартість хибнопозитивного, тобто

$$C(FN) \gg C(FP), \tag{2.89}$$

Нехай $R_i^a(z_i)$ – базова алеаторна складова ризику, обчислена на основі технічних і контекстних параметрів вразливості.

Тоді композитну оцінку ризику пропонується визначати як

$$R_i^{comp} = R_i^a \cdot \left(1 + \lambda \cdot \widetilde{H_i}\right), \qquad \lambda \in [0;1]. \tag{2.90}$$

де λ – коефіцієнт консервативного підсилення ризику за наявності невизначеності класифікатора.

$$I_R(z_i) = [R_i^a, min\left(10, R_i^{comp}\right)], \tag{2.91}$$

де $I_R(z_i)$ – інтервальна оцінка ризику, нижня межа якої відповідає базовій алеаторній складовій, а верхня межа враховує епістемічну невизначеність класифікаційного рішення [64].

Таким чином, при зростанні епістемічної невизначеності значення композитного ризику не зменшується, а коригується у бік більш обережної оцінки, що є доцільним для задач кібербезпеки, де вартість хибнонегативного рішення істотно перевищує вартість хибнопозитивного.

У загальній архітектурі системи модель адаптивного оцінювання ризику

виконує функцію фінального інтелектуального узагальнення результатів попередніх модулів. Вона використовує класифікаційний висновок моделі, інтегрує технічну оцінку за CVSS, ураховує бізнес-контекст застосунку, силу компенсуючих механізмів та формує числову й категоріальну оцінку ризику [63].

Модель адаптивного оцінювання ризику вразливостей, яка інтегрує технічну критичність, результати інтелектуальної класифікації, контекстні параметри середовища та фактори невизначеності, наведено на рисунку 2.5.

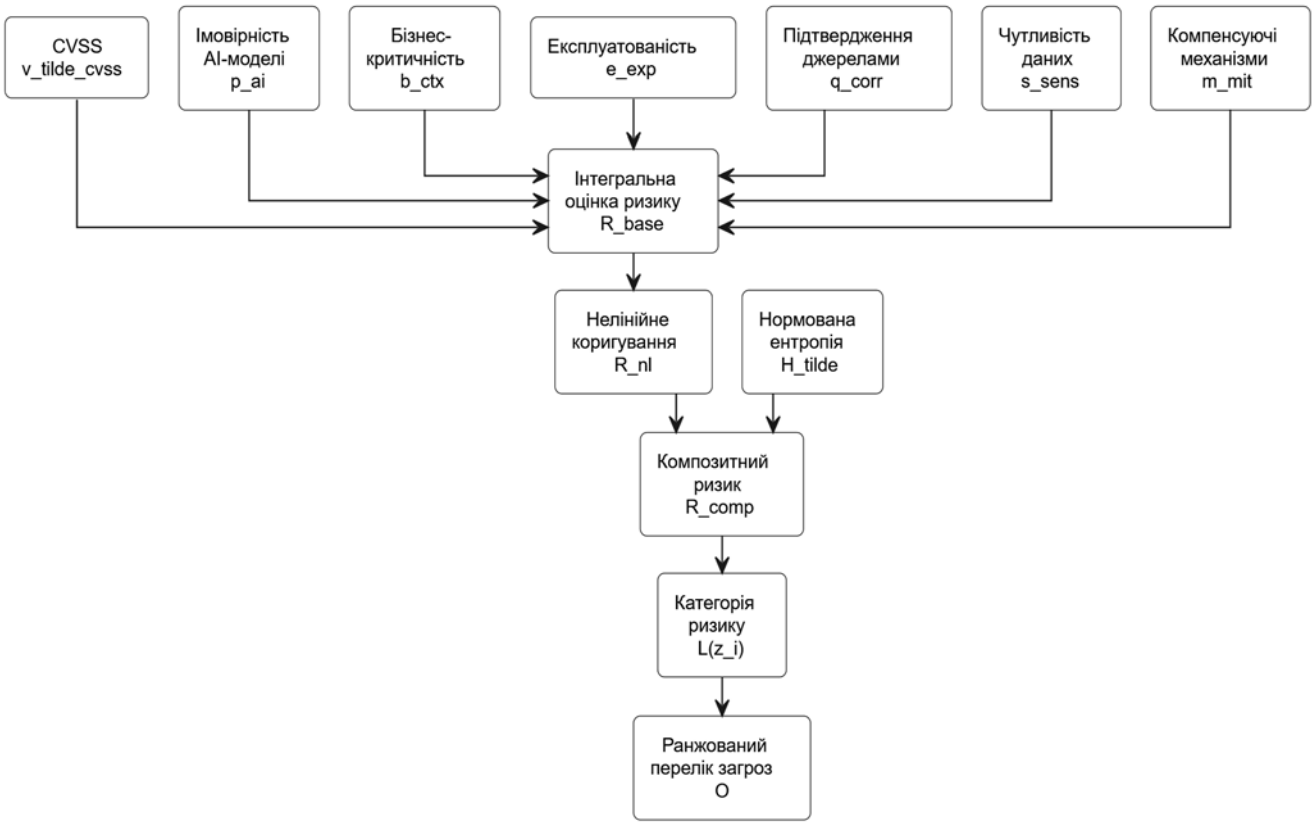


Рисунок 2.5 – Модель адаптивного оцінювання ризику вразливостей

Наведена модель відображає логіку формування інтегральної оцінки ризику на основі поєднання технічних, імовірнісних, контекстних і компенсаційних факторів, а також урахування епістемічної невизначеності класифікаційного рішення. Основні компоненти моделі адаптивного оцінювання ризику вразливостей наведено в таблиці 2.4.

Таблиця 2.4 – Компоненти моделі адаптивного оцінювання ризику

Компонент	Позначення	Діапазон	Зміст
Нормалізована технічна критичність	$(\widetilde{v_{cvss}})$	[0;1]	Нормоване значення базової технічної оцінки вразливості
Імовірність класифікації	(p^{ai})	[0;1]	Рівень упевненості класифікаційної моделі
Бізнес-критичність	(b^{ctx})	[0;1]	Значущість компонента для функціонування сервісу
Експлуатованість	(e^{exp})	[0;1]	Імовірність практичної експлуатації вразливості
Рівень підтвердження	(q^{corr})	[0;1]	Узгодженість результатів між різними джерелами
Чутливість даних або сервісу	(s^{sens})	[0;1]	Важливість даних чи сервісу, на які впливає загроза
Компенсуючі механізми	(m^{mit})	[0;1]	Сила наявних засобів зниження ризику
Епістемічна невизначеність	$(\widetilde{H})$	[0;1]	Нормована ентропія класифікаційного рішення
Композитний ризик	(R^{comp})	[0;10]	Підсумкова оцінка ризику з урахуванням невизначеності
Інтервальна оцінка ризику	$I_R(z_i)$	[0;10]	Діапазон між базовою алеаторною оцінкою та композитним ризиком з урахуванням епістемічної невизначеності

Отже, модель адаптивного оцінювання ризику спирається на поєднання технічних, імовірнісних, контекстних та компенсаційних факторів, а її розширення епістемічно-алеаторною декомпозицією дозволяє явно враховувати рівень

невизначеності класифікаційного рішення. Використання позначень $\widetilde{H}$, R_{comp} та $I_R(z_i)$ забезпечує узгодженість формальної моделі ризику з алгоритмічною й експериментальною частинами роботи.

2.6 Висновки до другого розділу

У другому розділі кваліфікаційної роботи розроблено теоретико-модельну основу інтелектуальної комп'ютерної системи автоматичного виявлення вразливостей вебзастосунків та класифікації загроз. Якщо у першому розділі основна увага була зосереджена на аналізі предметної області, існуючих підходів і виявленні обмежень традиційних засобів безпеки, то в другому розділі здійснено перехід до формального опису системи, її функціональних компонентів і моделей, що забезпечують інтелектуальну обробку результатів аналізу безпеки.

У підрозділі 2.1 виконано формалізацію предметної області та постановку задачі моделювання. Визначено множину вебзастосунків, артефактів, джерел аналізу, первинних і нормалізованих записів, класів загроз та параметрів ризику. Показано, що задача автоматичного виявлення вразливостей є багатоджерельною, багатокритеріальною та контекстно залежною.

У підрозділі 2.2 побудовано архітектурну модель ІКС АВВЗ. Обґрунтовано доцільність реалізації системи як багаторівневої модульної структури, що охоплює рівні збору даних, нормалізації, інтелектуального аналізу, кореляції, ризикового оцінювання, звітності та інтеграції із зовнішніми середовищами.

У підрозділі 2.3 розроблено модель інтелектуальної нормалізації результатів сканування. Формально описано процес перетворення неоднорідних повідомлень від різних засобів аналізу безпеки у стандартизовані, дедуплікові та семантично збагачені записи, придатні до подальшого машинного аналізу.

У підрозділі 2.4 побудовано модель класифікації вразливостей і кіберзагроз. Запропоновано гібридний підхід, що поєднує текстовий, структурний, контекстний і джерельний підмодулі аналізу та агрегує їх результати в єдине ймовірнісне рішення. Окремо визначено межу між реалізованим інтерпретованим

класифікаційним контуром і перспективними графово-орієнтованими та трансформерними рішеннями, які належать до наступного циклу розвитку системи.

У підрозділі 2.5 розроблено модель адаптивного оцінювання ризику вразливостей, яка поєднує нормалізовану технічну оцінку вразливості, результати інтелектуальної класифікації, бізнес-критичність компонента, експлуатованість, міждерельне підтвердження, чутливість даних і компенсуючі механізми захисту. Модель розширено епістемічно-алеаторною декомпозицією, у межах якої невизначеність класифікаційного рішення формалізується через нормовану ентропію Шеннона $\tilde{H}$та враховується в композитній оцінці ризику R^{comp}.

Отже, у другому розділі сформовано цілісну модельну основу ІКС АВВЗ, що охоплює формалізацію предметної області, архітектурну модель системи, модель інтелектуальної нормалізації результатів сканування, модель класифікації вразливостей і кіберзагроз та модель адаптивного оцінювання ризику. Результати другого розділу створюють теоретичне підґрунтя для розроблення алгоритмів функціонування системи та її програмної реалізації, що розглядаються у наступному розділі.

3 АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ ІНТЕЛЕКТУАЛЬНОЇ КОМП'ЮТЕРНОЇ СИСТЕМИ АВТОМАТИЧНОГО ВИЯВЛЕННЯ ВРАЗЛИВОСТЕЙ ВЕБЗАСТОСУНКІВ

У третьому розділі кваліфікаційної роботи розроблено алгоритмічне забезпечення інтелектуальної комп'ютерної системи автоматичного виявлення вразливостей вебзастосунків та класифікації загроз. Якщо у попередньому розділі було сформовано формальну, структурну та математичну основу функціонування системи, то в межах цього розділу зазначені положення конкретизовано у вигляді алгоритмів багатоканального збору результатів сканування, інтелектуальної класифікації вразливостей і ризик-орієнтованого ранжування загроз.

Основна увага приділяється логіці перетворення вхідних безпекових даних у впорядкований аналітичний результат. Для цього послідовно розглядаються алгоритми приймання та уніфікації результатів від різнорідних джерел, формування ознакового представлення вразливостей, визначення класу загрози, обчислення рівня впевненості моделі та побудови ранжованого переліку загроз відповідно до їхнього практичного ризику.

Таким чином, розділ формує алгоритмічну основу подальшої програмної реалізації ІКС АВВЗ, яка розглядається у четвертому розділі.

3.1 Алгоритм багатоканального збору та обробки результатів сканування

Ефективність функціонування ІКС АВВЗ значною мірою визначається тим, наскільки повно, стабільно та узгоджено система здатна збирати результати від різнорідних засобів аналізу безпеки. У сучасному середовищі захисту вебзастосунків потенційно корисна інформація про вразливості може надходити з різних джерел: SAST-сканерів, DAST-інструментів, IAST-засобів, SCA-модулів, журналів подій, аналізаторів HTTP-трафіку, систем контролю конфігурацій, засобів моніторингу контейнерного середовища та зовнішніх баз знань про вразливості. Якщо така інформація обробляється ізольовано, це призводить до

фрагментованості, дублювання, суперечностей і втрати контексту. Саме тому в ІКС АВВЗ пропонується алгоритм багатоканального збору та обробки результатів сканування, орієнтований на інтеграцію всіх доступних каналів у єдиний аналітичний контур.

У загальному вигляді множину каналів отримання даних можна подати як

$$S = \{s_1, s_2, \dots, s_M\}, \tag{3.1}$$

де s_j- окреме джерело або інструмент аналізу безпеки. Для конкретного вебзастосунку w_iкожен канал формує локальну множину результатів

$$F_{ij} = \left\{f_{ij1}, f_{ij2}, \dots, f_{ijn_j}\right\}, \tag{3.2}$$

де f_{ijk}- окреме повідомлення, запис або знахідка, сформована j-м джерелом для i-го застосунку.

Сукупний вхідний потік даних для вебзастосунку w_iвизначається як

$$F_i^{in} = \bigcup_{j=1}^{M} F_{ij}. \tag{3.3}$$

Основна задача алгоритму багатоканального збору полягає в тому, щоб перетворити множину F_i^{in}у впорядкований, синхронізований і придатний до подальшої інтелектуальної обробки потік записів, який надходить до модуля нормалізації. Для цього в системі реалізується поетапний алгоритм, що включає ініціалізацію каналів, приймання даних, перевірку цілісності, уніфікацію транспортного формату, тимчасове буферування, маршрутизацію, пріоритезацію та передавання в підсистему обробки.

Алгоритм багатоканального збору та обробки результатів сканування можна подати як відображення

$$A_{scan}: \langle W, S, T \rangle \rightarrow Q, \tag{3.4}$$

де W – множина вебзастосунків;

S – множина каналів збору;

T – множина часових параметрів і політик запуску;

Q – внутрішня черга уніфікованих повідомлень для подальшої обробки.

У спрощеному вигляді алгоритм можна представити послідовністю перетворень

$$F_i^{in} \xrightarrow{A_1} F_i^{chk} \xrightarrow{A_2} F_i^{uni} \xrightarrow{A_3} F_i^{buf} \xrightarrow{A_4} Q_i, \tag{3.5}$$

де A_1 – перевірка цілісності й валідності повідомлень;

A_2 – приведення до єдиного транспортного формату;

A_3 – буферизація та впорядкування за часом і пріоритетом;

A_4 – розміщення повідомлень у внутрішній черзі системи.

На початковому етапі система формує перелік активних каналів збору відповідно до конфігурації, середовища виконання та режиму запуску. Для кожного джерела s_j задається набір параметрів

$$cfg_j = \langle type_j, addr_j, fmt_j, auth_j, mode_j, prio_j \rangle, \tag{3.6}$$

де $type_j$- тип джерела;

$addr_j$- адреса або маршрут доступу;

fmt_j- формат даних;

$auth_j$- параметри автентифікації;

$mode_j$- режим роботи (push/pull, on-demand, scheduled);

$prio_j$- базовий пріоритет джерела.

На підставі цих параметрів формується множина активних джерел

$$S^{act} = \{ s_j \in S \mid status(s_j) = ready \}. \quad (3.7)$$

Цей етап має критичне значення для стабільності алгоритму, оскільки дозволяє виключити недоступні або некоректно налаштовані джерела ще до початку фактичного збору даних.

На другому етапі система виконує приймання результатів із кожного активного джерела. Для цього використовується функція отримання

$$R_j = fetch(s_j, t), \quad (3.8)$$

де R_j – масив повідомлень, отриманих від джерела s_j у момент або інтервал часу t.

У результаті для всіх активних каналів формується проміжна множина

$$F_i^{raw}(t) = \bigcup_{s_j \in S^{act}} fetch(s_j, t), \quad (3.9)$$

На практичному рівні цей етап може реалізовуватися через API-виклики до сканерів, обробку файлів звітів, приймання повідомлень із черг, вебхуки, читання журналів та планувальники завдань. Важливо, що алгоритм не прив’язується жорстко до конкретного способу доставки результатів, а працює з абстрактною моделлю каналу.

Усі отримані записи мають пройти початкову перевірку коректності. Для кожного повідомлення $f_k \in F_i^{raw}$ виконується предикат валідності

$$V(f_k) = \begin{cases} 1, & \text{якщо запис коректний,} \\ 0, & \text{інакше.} \end{cases} \quad (3.10)$$

До коректних записів належать ті, що мають розпізнаваний формат, містять мінімально необхідний набір полів, не пошкоджені під час передавання та відповідають очікуваному типу джерела. Після фільтрації утворюється множина

$$F_i^{chk} = \{ f_k \in F_i^{raw} \mid V(f_k) = 1 \}. \quad (3.11)$$

Некоректні або частково зруйновані записи не передаються далі, а фіксуються у журналі моніторингу для подальшого аналізу.

Оскільки результати від різних джерел можуть надходити в різних форматах, алгоритм багатоканального збору має виконувати перетворення в єдину транспортну структуру ще до передавання в модуль нормалізації. Для цього використовується функція

$$u_k = transform(f_k), \quad (3.12)$$

де u_k – уніфіковане транспортне представлення запису, який можна подати як

$$u_k = \langle id_w, src, timestamp, payload, hash, prio \rangle, \quad (3.13)$$

де id_w – ідентифікатор вебзастосунку;
src- джерело повідомлення;
$timestamp$ – час отримання або створення;
$payload$ – корисне навантаження;
$hash$ – контрольна ознака для перевірки дублювання;
$prio$ – пріоритет повідомлення.

Після уніфікації формується множина

$$F_i^{uni} = \{ u_k \mid f_k \in F_i^{chk} \}. \quad (3.14)$$

Оскільки потоки від різних джерел можуть надходити нерівномірно, з різною затримкою або пакетами різного розміру, система використовує буферизацію. Це дозволяє уникнути втрати послідовності подій і забезпечити стабільне

навантаження на наступні модулі. Для буферизації використовується множина тимчасового накопичення

$$B_i(t) = \{ u_k \in F_i^{uni} \mid t_k \in \Delta t \}, \tag{3.15}$$

де Δt – часовий інтервал накопичення.

Після цього виконується впорядкування записів за комбінованим ключем часу та пріоритету:

$$ord(u_k) = \langle prio_k, timestamp_k \rangle. \tag{3.16}$$

Тоді впорядкована множина набуває вигляду

$$F_i^{buf} = \text{sort}(B_i(t), ord). \tag{3.17}$$

Такий механізм дозволяє першими обробляти записи, що належать до критичніших джерел або мають вищий системний пріоритет.

На наступному етапі впорядковані записи передаються у внутрішню чергу системи. Нехай Q_i- черга для застосунку w_i. Тоді операція додавання має вигляд

$$Q_i \leftarrow enqueue\left(Q_i, F_i^{buf}\right). \tag{3.18}$$

Черга Q_iвиконує кілька функцій: розв’язує проблему асинхронності джерел; розділяє етапи збору та інтелектуальної обробки; дозволяє масштабувати систему шляхом паралельного споживання повідомлень; забезпечує повторну обробку у випадку збою наступного модуля.

Залежно від типу джерела, типу даних та конфігурації системи окремі записи можуть спрямовуватися до різних гілок первинної обробки. Для цього вводиться функція маршрутизації

$$route(u_k) = m_r, \tag{3.19}$$

де m_r – відповідний модуль або підпроцес первинної обробки.

У разі масштабування системи така маршрутизація може бути пов’язана з вибором наступного варіанта централізації або перерозподілу функцій між модулями, що узгоджується з підходами до визначення правил зміни архітектурної централізації в багатокомп’ютерних системах [70].

Оскільки система працює в реальному середовищі з багатьма зовнішніми джерелами, алгоритм має враховувати ситуації недоступності джерела, пошкодження повідомлення, непідтримуваного формату або перевантаження внутрішньої черги. Для цього вводиться функція стану операції

$$status(u_k) \in \{ok, retry, reject\}. \tag{3.20}$$

Якщо запис не може бути оброблений негайно через тимчасову проблему, він переводиться в режим повторної спроби:

$$u_k \in Q^{retry} \iff status(u_k) = retry. \tag{3.21}$$

Якщо ж запис є непридатним до обробки, він фіксується в журналі помилок:

$$u_k \in Q^{err} \iff status(u_k) = reject. \tag{3.22}$$

Такий механізм підвищує надійність роботи системи й запобігає втраті даних через короткочасні помилки. Ефективність алгоритму багатоканального збору доцільно оцінювати за кількома критеріями: повнотою приймання даних, стійкістю до помилок каналів, часом доставки повідомлення до модуля нормалізації, рівнем втрат або відкинутих повідомлень, середньою довжиною внутрішньої черги та масштабованістю при зростанні кількості джерел. Узагальнений функціонал якості можна подати як

$$J_{scan} = \lambda_1 Q_{full} + \lambda_2 Q_{stable} + \lambda_3 Q_{throughput} - \lambda_4 L_{drop} - \lambda_5 T_{delay},$$

(3.23)

де Q_{full} – повнота збору;

Q_{stable} – стійкість до збоїв;

$Q_{throughput}$ – пропускна здатність;

L_{drop} – частка втрат;

T_{delay} – середня затримка доставлення.

Алгоритм багатоканального збору та обробки результатів сканування є початковою динамічною ланкою функціонування ІКС АВВЗ. Саме він визначає, які результати потрапляють до системи, у якому вигляді вони передаються на наступні етапи, наскільки повно враховується інформація з різних джерел та чи вдається уникнути втрат, хаотичності та розривів у ланцюгу аналітичної обробки.

Таблиця 3.1 – Основні етапи алгоритму багатоканального збору

Етап	Призначення	Результат
Ініціалізація джерел	виявлення активних каналів	перелік доступних джерел
Отримання даних	приймання результатів сканування	сирі записи
Перевірка валідності	фільтрація пошкоджених або неповних повідомлень	коректні записи
Уніфікація формату	приведення до єдиної транспортної схеми	уніфіковані записи
Буферизація	часове накопичення	буфер повідомлень
Сортування	впорядкування за пріоритетом і часом	впорядкований потік
Черга обробки	асинхронне передавання даних	внутрішня черга
Маршрутизація	спрямування в потрібний модуль	керований потік обробки

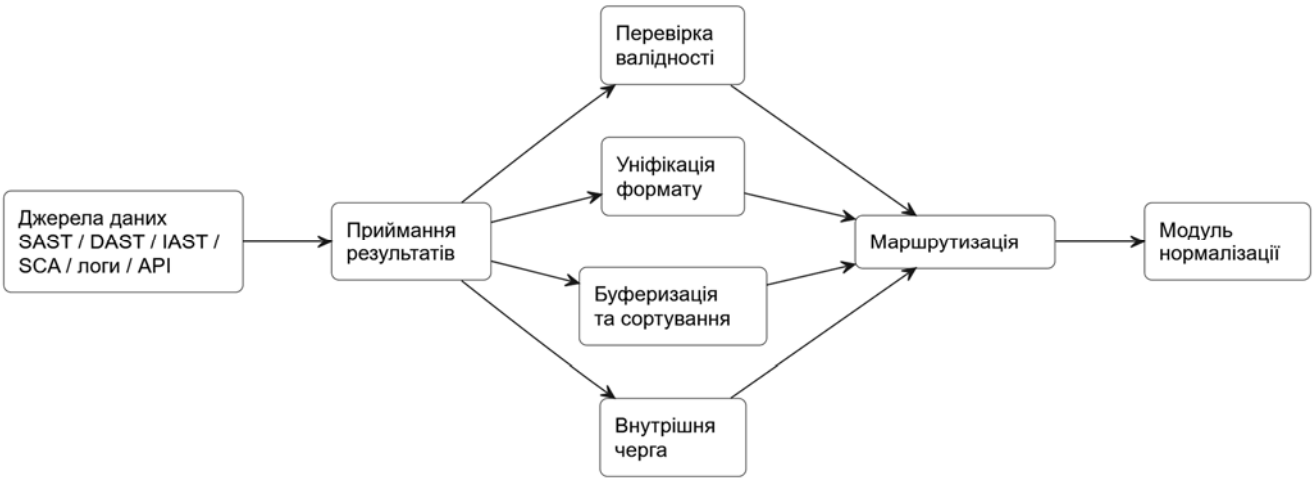


Рисунок 3.1 – Алгоритм багатоканального збору та обробки результатів сканування.

3.2 Алгоритм інтелектуальної класифікації вразливостей

Після виконання багатоканального збору, перевірки, уніфікації та нормалізації результатів сканування наступним ключовим етапом функціонування ІКС АВВЗ є інтелектуальна класифікація вразливостей. Призначення цього етапу полягає у визначенні типу вразливості або кіберзагрози на основі аналізу сукупності текстових, структурних, контекстних і технічних ознак. Якщо попередній алгоритм забезпечує коректне надходження й підготовку даних, то алгоритм інтелектуальної класифікації виконує їх змістову інтерпретацію. У загальному вигляді алгоритм інтелектуальної класифікації можна розглядати як процедуру перетворення множини нормалізованих записів

$$Z = \{z_1, z_2, \dots, z_n\} \tag{3.24}$$

у множину класифікованих об’єктів

$$Y^* = \{\langle z_i, \hat{y}_i, p_i \rangle\}_{i=1}^{n}, \tag{3.25}$$

де $\hat{y}_i$ – прогнозований клас вразливості;

p_i – вектор імовірностей або довіри моделі до прийнятого рішення.

Для кожного нормалізованого запису z_i алгоритм повинен виділити інформативні ознаки, сформувати внутрішнє представлення запису, виконати класифікацію за допомогою гібридної моделі, обчислити ймовірності належності до класів, визначити фінальне рішення та зберегти пояснювальні атрибути для подальшої інтерпретації та ризикового оцінювання.Алгоритм інтелектуальної класифікації задамо як відображення

$$A_{cls}: Z \times \Theta \to Y^*, \tag{3.26}$$

де Z – множина нормалізованих записів;

Θ – параметри навченої моделі класифікації,

Y^* – множина класифікованих результатів.

У процедурному вигляді це перетворення можна подати ланцюгом

$$Z \overset{B_1}{\to} X \overset{B_2}{\to} H \overset{B_3}{\to} P \overset{B_4}{\to} Y^*, \tag{3.27}$$

де B_1 – побудова ознакового представлення;

B_2 – обчислення внутрішніх латентних представлень;

B_3 – формування векторів імовірностей за класами;

B_4 – прийняття фінального рішення та формування пояснювальних атрибутів.

На вхід алгоритму надходить нормалізований запис, сформований модулем інтелектуальної нормалізації. Кожен запис доцільно подати у вигляді

$$z_i = \langle id_w, a_i, x_i^{txt}, x_i^{str}, x_i^{ctx}, x_i^{src}, meta_i \rangle, \tag{3.28}$$

де id_w – ідентифікатор вебзастосунку;

a_i – локалізація або артефакт;

x_i^{txt} – текстові ознаки;

x_i^{str} – структурні ознаки;

x_i^{ctx} – контекстні ознаки;

x_i^{src} – ознаки джерел підтвердження;

$meta_i$ – додаткові атрибути.

На цьому етапі алгоритм перевіряє наявність мінімально необхідного набору ознак та визначає, чи придатний запис до класифікації. Якщо ключові ознаки відсутні або мають недостатню інформативність, запис може бути позначений як такий, що потребує додаткового аналізу. Після прийняття запису до обробки виконується формування єдиного вектора ознак, що об'єднує всі доступні представлення. Формально ця операція має вигляд

$$x_i = \Psi(z_i) = [x_i^{txt} \parallel x_i^{str} \parallel x_i^{ctx} \parallel x_i^{src}], \tag{3.29}$$

де Ψ – оператор побудови інтегрованого вектора ознак.

Такий підхід дозволяє поєднати в одному просторі ознак семантичний зміст повідомлення, структуру вразливого компонента, контекст функціонування та інформацію про джерела, які підтвердили знахідку. Саме це забезпечує перевагу гібридного алгоритму над підходами, що працюють лише з текстом або лише зі структурними ознаками.

На третьому етапі інтегрований вектор ознак перетворюється у внутрішнє латентне представлення, зручне для подальшого класифікаційного аналізу. Це перетворення можна описати як

$$h_i = \phi(x_i, \Theta_h), \tag{3.30}$$

де h_i – латентний вектор, а ϕ – функція проєкції у прихований простір ознак.

У практичному сенсі саме на цьому етапі модель формує узагальнений образ вразливості, у якому згорнуто текстову семантику, тип локалізації дефекту, ознаки контексту та характер підтвердження від джерел. Це дозволяє відокремити інформативні патерни від технічного шуму і сформувати внутрішню структуру ознак, на якій будується остаточне рішення.

Відповідно до моделі, побудованої у підроздлі 2.4, алгоритм інтелектуальної класифікації використовує кілька спеціалізованих підмодулів. Для кожного запису паралельно або послідовно обчислюються часткові результати:

$$p_i^{txt} = C_{txt}(x_i^{txt}), \tag{3.31}$$

$$p_i^{ctx} = C_{ctx}(x_i^{ctx}) \tag{3.32}$$

$$p_i^{str} = C_{str}(x_i^{str}), \tag{3.33}$$

$$p_i^{src} = C_{src}(x_i^{src}), \tag{3.34}$$

де кожен вектор $p_i^{(\cdot)} \in [0;1]K$ є оцінкою імовірностей належності до K класів загроз. Текстовий підмодуль аналізує семантику повідомлення. Структурний – ураховує технічну локалізацію та тип артефакту. Контекстний – враховує роль компонента у вебзастосунку та важливість середовища. Джерельний – оцінює достовірність на підставі кількості та типу джерел підтвердження. Після отримання результатів від усіх підмодулів виконується їх агрегація. У базовому варіанті агрегування реалізується через зважене поєднання:

$$p_i = \omega_1 p_i^{txt} + \omega_2 p_i^{str} + \omega_3 p_i^{ctx} + \omega_4 p_i^{src}, \tag{3.35}$$

де

$$\omega_1 + \omega_2 + \omega_3 + \omega_4 = 1, \qquad \omega_j \geq 0. \tag{3.36}$$

У результаті отримується підсумковий вектор імовірностей

$$p_i = \langle p_{i1}, p_{i2}, \dots, p_{iK} \rangle. \tag{3.37}$$

Такий механізм дозволяє керовано враховувати внесок кожного типу ознак і, за необхідності, адаптувати ваги до специфіки середовища або профілю

вебзастосунку. На основі підсумкового вектора імовірностей алгоритм формує фінальне рішення. Для одноетикеткової класифікації маємо правило

$$\hat{y}_i = arg \max_{y_k \in Y} p_{ik}\,. \tag{3.38}$$

Максимальна ймовірність

$$conf_i = \max_{y_k \in Y} p_{ik} \tag{3.39}$$

розглядається як рівень упевненості моделі. У випадку, коли потрібно підтримувати багатоміткову класифікацію, рішення приймається за пороговим правилом

$$\hat{Y}_i = \{\, y_k \in Y \mid p_{ik} \geq \tau_k \,\}, \tag{3.40}$$

де τ_k- поріг для класу y_k.

Оскільки система орієнтована на практичне використання в розробці та аналітичних процесах безпеки, алгоритм класифікації має формувати не лише клас загрози, а й пояснювальну інформацію. Для цього визначається множина пояснювальних атрибутів

$$E_i = \langle feat_i^{top}, src_i^{top}, ctx_i^{key}, alt_i \rangle, \tag{3.41}$$

де $feat_i^{top}$ – найбільш вагомі ознаки, що вплинули на рішення;

src_i^{top} – джерела, що найбільше підтвердили класифікацію;

ctx_i^{key} – ключові контекстні фактори;

alt_i – альтернативні класи з високими ймовірностями.

Тоді класифікований результат подається як

$$y_i^* = \langle z_i, \hat{y}_i, p_i, conf_i, E_i \rangle. \quad (3.42)$$

У реальному середовищі частина записів може не мати достатньо однозначної класифікації. Для таких випадків доцільно ввести поріг упевненості γ. Якщо

$$conf_i < \gamma, \quad (3.43)$$

то запис переводиться в категорію невизначених або умовно класифікованих випадків:

$$status_i = \begin{cases} classified, & conf_i \geq \gamma, \\ review, & conf_i < \gamma. \end{cases} \quad (3.44)$$

Це дозволяє не переоцінювати сумнівні результати, направляти складні випадки на додатковий аналіз та підвищувати загальну надійність функціонування системи. У реалізованому прототипі як базовий класифікаційний контур використано TF-IDF-векторизацію текстових описів знахідок і логістичну регресію. Водночас у логіці системи вже враховуються контекстні, джерельні та структурні ознаки, а подальший розвиток до повноцінного graph-based GNN-контру та fine-tuned Transformer/LLM-рішень визначено як наступний цикл розвитку. Така межа узгоджується з метою ти не завищує поточний рівень реалізації.

У межах практичної реалізації ІКС АВВЗ алгоритм інтелектуальної класифікації має бути достатньо швидким для використання у циклах безпечної розробки й водночас достатньо гнучким для обробки складних багатокомпонентних записів. Тому доцільно передбачити два режими роботи: пакетний – для великої множини записів після повного сканування, та потоковий – для обробки окремих подій або результатів у близькому до реального часу режимі. Для пакета з nзаписів сумарний час обробки можна подати як

$$T_{cls}(n) = \sum_{i=1}^{n}(t_i^{prep} + t_i^{model} + t_i^{agg}), \quad (3.45)$$

де t_i^{prep} – час підготовки ознак;

t_i^{model} – час роботи підмодулів;

t_i^{agg} – час агрегації та формування рішення.

Ефективність алгоритму інтелектуальної класифікації доцільно оцінювати за такими показниками: точністю класифікації, повнотою виявлення критичних загроз, стійкістю до шумних або неповних даних, часткою невизначених випадків та середнім часом прийняття рішення. Узагальнений критерій якості алгоритму можна подати як

$$Jcls = \mu 1F1 + \mu 2Precision + \mu 3Recall - \mu 4U - \mu 5Tavg, \quad (3.46)$$

де U – частка записів, переданих на додатковий перегляд;

T_{avg} – середній час класифікації одного запису;

$\mu 1, \dots, \mu 5$ – вагові коефіцієнти, причому $\mu 1 + \dots + \mu 5 = 1, \mu i \geq 0$.

Слід зазначити, що вагові коефіцієнти μi у формулі (3.46) використовуються для оцінювання якості саме підсистеми класифікації і не тотожні ваговим коефіцієнтам $w1, \dots, w6$ у формулі (2.17), яка характеризує узагальнений критерій ефективності функціонування системи в цілому.

Алгоритм інтелектуальної класифікації є центральним аналітичним механізмом ІКС АВВЗ. Саме він забезпечує перехід від структурованого технічного запису до семантичного висновку про тип загрози, імовірнісну оцінку достовірності рішення, формування основи для адаптивного ризикового оцінювання та зменшення частки хибних спрацювань за рахунок гібридної обробки ознак.

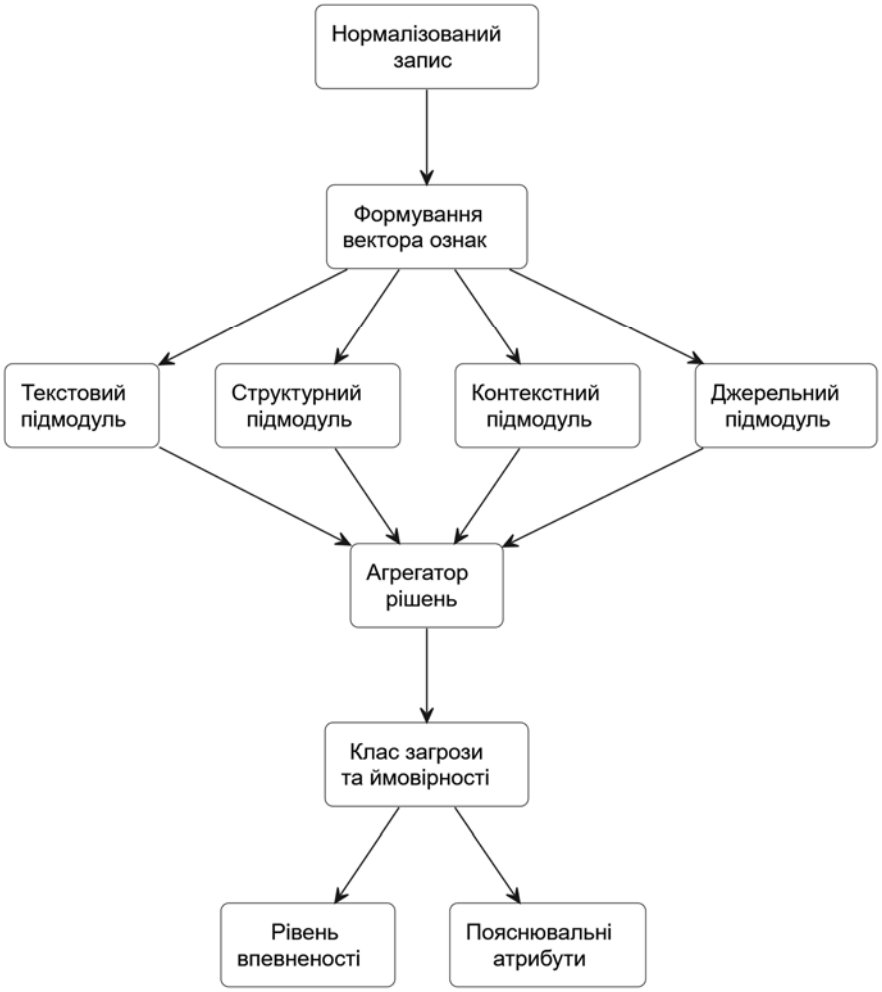


Рисунок 3.2 – Алгоритм інтелектуальної класифікації вразливостей.

Таблиця 3.2 – Етапи алгоритму інтелектуальної класифікації

Етап	Призначення	Результат
Приймання запису	отримання нормалізованого об’єкта	запис для аналізу
Формування ознак	інтеграція текстових, структурних, контекстних і джерельних ознак	вектор ознак
Робота підмодулів	часткова класифікація за окремими типами ознак	часткові вектори імовірностей
Агрегація	поєднання результатів підмодулів	підсумковий вектор імовірностей
Прийняття рішення	визначення класу загрози	прогнозований клас
Оцінка впевненості	визначення достовірності рішення	рівень довіри
Пояснювальний вихід	формування інтерпретації рішення	пояснювальні атрибути

3.3 Алгоритм ризик-орієнтованого ранжування загроз

Після завершення етапу інтелектуальної класифікації вразливостей система отримує не лише формалізовані записи про потенційні дефекти безпеки, а й інформацію про тип загрози, рівень упевненості класифікатора та допоміжні ознаки, необхідні для подальшої аналітичної обробки. Проте навіть коректно класифікований перелік вразливостей ще не дає відповіді на головне практичне запитання: які саме загрози необхідно усувати в першу чергу. У реальних умовах експлуатації вебзастосунків кількість виявлених дефектів може бути значною, а ресурси команд розробки та безпеки – обмеженими. Для усунення цієї проблеми в ІКС АВВЗ використано алгоритм ризик-орієнтованого ранжування загроз, який дозволяє перетворити множину класифікованих вразливостей на впорядкований перелік пріоритетів реагування.

У межах даної роботи під ризик-орієнтованим ранжуванням розуміється алгоритмічний процес обчислення інтегральної оцінки небезпеки для кожної вразливості з урахуванням її технічної критичності, типу загрози, бізнес-контексту, рівня експлуатованості, міжджерельного підтвердження, чутливості вразливого компонента та сили компенсуючих механізмів захисту. На відміну від звичайного сортування за CVSS або початковою оцінкою сканера, такий підхід дозволяє враховувати реальний контекст використання вебзастосунку, зменшувати вплив хибних спрацювань і формувати практично цінний порядок усунення вразливостей [53, 54]. Нехай після етапу класифікації маємо множину об’єктів

$$Y^* = \{y_1^*, y_2^*, \dots, y_n^*\}, \tag{3.47}$$

де кожен елемент

$$y_i^* = \langle z_i, \hat{y}_i, p_i, conf_i, E_i \rangle \tag{3.48}$$

де z_i – нормалізований запис;

$\hat{y}_i$ – прогнозований клас загрози;

p_i – вектор імовірностей по класах;

$conf_i$ – рівень упевненості моделі;

E_i – пояснювальні атрибути.

Задача ризик-орієнтованого ранжування полягає у побудові відображення

$$A_{rank}: Y^* \to \Omega, \tag{3.49}$$

де Ω – впорядкована множина загроз, відсортованих за спаданням інтегрального ризику. У процедурному вигляді алгоритм можна подати як послідовність перетворень

$$Y^* \overset{R_1}{\to} G \overset{R_2}{\to} V \overset{R_3}{\to} L \overset{R_4}{\to} \Omega, \tag{3.50}$$

де R_1 – виділення ризикових параметрів;

R_2 – обчислення інтегральної числової оцінки ризику;

R_3 – визначення категоріального рівня ризику;

R_4 – сортування та формування ранжованого переліку загроз.

На першому етапі алгоритм формує для кожної вразливості набір параметрів, необхідних для оцінювання ризику. Для кожного запису y_i^* визначається вектор

$$g_i = \langle \widetilde{v_i^{cvss}}, p_i^{ai}, b_i^{ctx}, e_i^{exp}, q_i^{corr}, s_i^{sens}, k_i^{cls}, m_i^{mit} \rangle. \tag{3.51}$$

де $\tilde{v}_i^{cvss}$ – нормалізоване значення CVSS;

p_i^{ai} – імовірність або рівень впевненості класифікації;

b_i^{ctx} – бізнес-критичність ураженого компонента;

e_i^{exp} – оцінка експлуатованості;

q_i^{corr} – рівень міжджерельного підтвердження;

s_i^{sens} – чутливість даних або сервісу;

k_i^{cls} – клас-специфічна вага загрози, що враховує належність вразливості до категорії OWASP/CWE та її типовий операційний вплив.

m_i^{mit} – сила компенсуючих механізмів захисту.

Оскільки параметри ризику мають різну природу, їх необхідно привести до спільної шкали. Для цього всі компоненти переводяться до інтервалу [0;1]. Нормалізація CVSS задається як

$$\tilde{v}_i^{cvss} = \frac{v_i^{cvss}}{10}, \tag{3.52}$$

де $v_i^{cvss} \in [0,10]$.

Інші параметри або вже мають значення в інтервалі $[0, 1]$, або визначаються через спеціальні шкали та словники відповідностей. Наприклад, бізнес-критичність може задаватися так:

$$b_i^{ctx} = \begin{cases} 0.2, & \text{низька важливість}, \\ 0.5, & \text{середня важливість}, \\ 0.8, & \text{висока важливість}, \\ 1.0, & \text{критичний компонент}. \end{cases} \tag{3.53}$$

Аналогічно можуть нормалізуватися експлуатованість, чутливість даних та сила компенсуючих механізмів. Після формування вектора параметрів алгоритм обчислює базову інтегральну оцінку ризику. У межах даної роботи доцільно використати зважену адитивну модель

$$R_i^{base} = 10\big(\alpha_1 \widetilde{v_i^{cvss}} + \alpha_2 p_i^{ai} + \alpha_3 b_i^{ctx} + \alpha_4 e_i^{exp} + \alpha_5 q_i^{corr} + \alpha_6 s_i^{sens} + \alpha_7 k_i^{cls} - \alpha_8 m_i^{mit}\big), \tag{3.54}$$

де $\alpha 1, \dots, \alpha 8$ – вагові коефіцієнти моделі ризику.

$$\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 + \alpha_6 + \alpha_7 + \alpha_8 = 1, \quad \alpha_j \geq 0 \tag{3.55}$$

У цій формулі технічна критичність, бізнес-контекст, експлуатованість, підтвердження та чутливість збільшують ризик, компенсуючі механізми зменшують його, а впевненість моделі відіграє роль коефіцієнта достовірності інтерпретації. Для запобігання виходу за межі шкали використовується обмеження

$$R_i^{clip} = \min\{10, \max\{0, R_i^{base}\}\}, \tag{3.56}$$

У деяких випадках поєднання кількох факторів повинно різко підвищувати пріоритет загрози. Наприклад, якщо вразливість має високий CVSS, належить до критичного бізнес-компонента, легко експлуатується та підтверджена кількома джерелами, то її небезпека є вищою за просту лінійну суму складових. Для цього вводиться нелінійний коефіцієнт посилення

$$k_i^{nl} = 1 + \rho \cdot \tilde{v}_i^{cvss} \cdot b_i^{ctx} \cdot e_i^{exp}, \tag{3.57}$$

де $\rho \in [0,1]$- коефіцієнт посилення. Тоді скоригована оцінка ризику має вигляд

$$R_i^{nl} = R_i^{clip} \cdot k_i^{nl}. \tag{3.58}$$

Фінальна оцінка знову обмежується шкалою $[0, 10]$:

$$R_i^{final} = \min\{10, R_i^{nl}\}. \tag{3.59}$$

Для практичного використання недостатньо мати лише числову оцінку. Потрібно також сформувати категоріальний рівень ризику, зручний для візуалізації, звітності та прийняття рішень. Нехай функція категоризації має вигляд

$$L_i = \begin{cases} \text{Низький}, & 0 \le R_i^{final} < 4, \\ \text{Середній}, & 4 \le R_i^{final} < 7, \\ \text{Високий}, & 7 \le R_i^{final} < 9, \\ \text{Критичний}, & 9 \le R_i^{final} \le 10. \end{cases} \tag{3.60}$$

Після обчислення числової та категоріальної оцінки для кожної загрози формується об’єкт ранжування

$$r_i^* = \langle z_i, \hat{y}_i, R_i^{final}, L_i, conf_i, E_i \rangle. \quad (3.61)$$

де z_i – вихідний нормалізований запис;

$\hat{y}_i$ – клас загрози;

R_i^{final} – інтегральний ризик;

L_i – категорія ризику;

$conf_i$ – рівень довіри до класифікації;

E_i – пояснювальні атрибути.

Саме ця структура передається далі до модуля звітності та функціонального рівня візуалізації. На завершальному етапі алгоритм виконує впорядкування всіх записів за спаданням фінального значення ризику:

$$\Omega = \mathrm{sort}_{\mathrm{desc}}\{r_i^*\}_{i=1}^{n} \quad (3.62)$$

У разі рівності оцінок ризику можуть використовуватися додаткові правила сортування: за критичністю класу загрози, за наявністю кількох підтверджень, за чутливістю компонента та за часом виявлення. Формально комбінований ключ сортування можна подати як

$$ord(r_i^*) = \langle R_i^{final}, q_i^{corr}, b_i^{ctx}, timestamp_i \rangle. \quad (3.63)$$

Таким чином, система формує не просто перелік знайдених вразливостей, а впорядкований список пріоритетів усунення. Для інтеграції з процесами безпечної розробки алгоритм може додатково формувати групи реагування. Нехай множина результатів Ω розбивається на підмножини

$$\Omega = \Omega_{crit} \cup \Omega_{high} \cup \Omega_{med} \cup \Omega_{low}, \quad (3.64)$$

де Ω_{crit} – загрози, що вимагають негайного втручання;

Ω_{high} – загрози високого пріоритету;

Ω_{med} – загрози, що можуть бути заплановані в найближчий цикл виправлення;

Ω_{low} – загрози низького пріоритету або кандидати на додаткову перевірку.

Ефективність алгоритму ризик-орієнтованого ранжування визначається не лише математичною коректністю розрахунку, а й його практичною користю. Основними критеріями оцінки є адекватність ранжування з точки зору експертної оцінки, стійкість до незначних шумових змін у параметрах, зменшення пріоритету хибних спрацювань, здатність виносити дійсно критичні загрози у верхню частину списку та узгодженість із реальними сценаріями експлуатації. Узагальнений функціонал якості алгоритму можна подати як

$$J_{rank} = \nu_1 Q_{top} + \nu_2 Q_{expert} + \nu_3 Q_{stable} - \nu_4 FP_{top}, \quad (3.65)$$

де Q_{top}- якість верхньої частини ранжованого списку;

Q_{expert} – узгодженість із експертною оцінкою;

Q_{stable} – стійкість до шуму;

FP_{top} – частка хибних спрацювань у верхніх пріоритетах;

ν_i – вагові коефіцієнти.

Алгоритм ризик-орієнтованого ранжування загроз є завершальною аналітичною ланкою внутрішнього інтелектуального циклу ІКС АВВЗ. Якщо модуль багатоканального збору формує сукупність даних, модуль нормалізації уніфікує їх, а модуль класифікації визначає тип загрози, то саме алгоритм ранжування відповідає на головне прикладне питання: що потрібно виправляти першочергово.

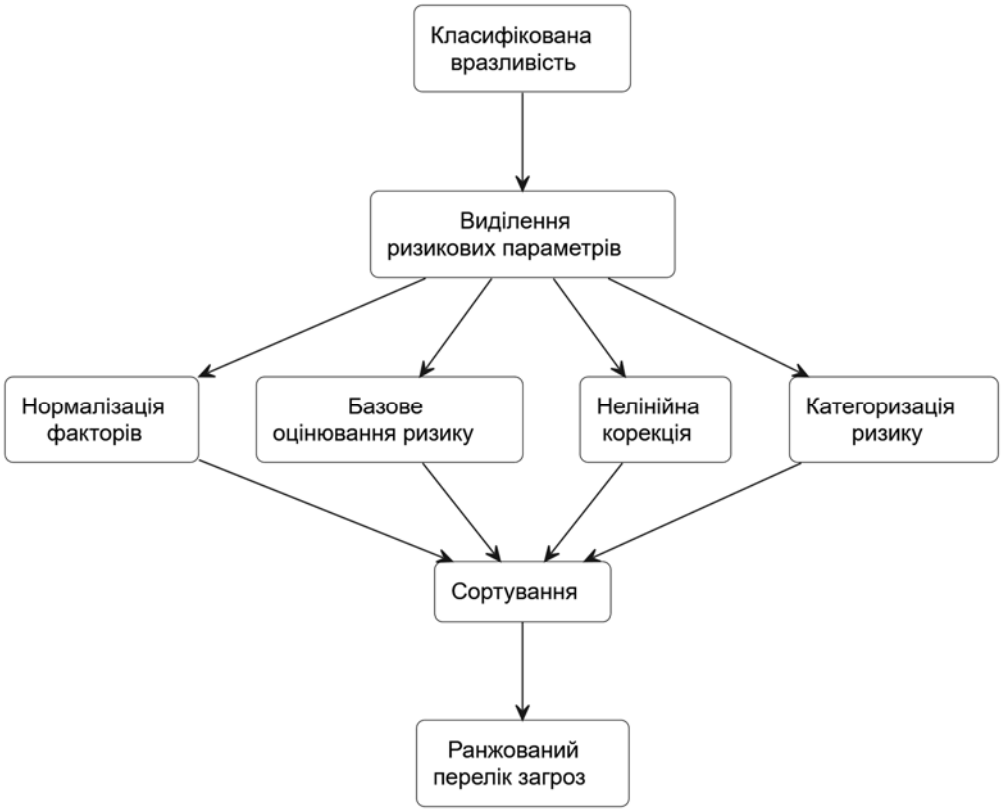


Рисунок 3.3 – Алгоритм ризик-орієнтованого ранжування загроз.

Таблиця 3.3 – Основні етапи алгоритму ризик-орієнтованого ранжування

Етап	Призначення	Результат
Виділення параметрів	формування набору факторів ризику	вектор ризикових ознак
Нормалізація	приведення параметрів до спільної шкали	нормалізовані значення
Базове оцінювання	обчислення інтегрального ризику	числове значення ризику
Нелінійна корекція	посилення критичних комбінацій факторів	скоригований ризик
Категоризація	визначення рівня небезпеки	категорія ризику
Сортування	впорядкування загроз за пріоритетом	ранжований перелік
Зони реагування	групування для практичного використання	Списки для конвеєрів безпечної розробки

3.4 Висновки до третього розділу

У третьому розділі кваліфікаційної роботи розроблено алгоритмічне забезпечення інтелектуальної комп’ютерної системи автоматичного виявлення вразливостей вебзастосунків та класифікації загроз.

У підрозділі 3.1 сформовано алгоритм багатоканального збору та обробки результатів сканування. Визначено послідовність дій, що охоплює ініціалізацію джерел, отримання даних із різних каналів, перевірку валідності, уніфікацію транспортного формату, буферизацію, впорядкування, постановку у внутрішню чергу та маршрутизацію до наступних етапів обробки. Обґрунтовано, що цей алгоритм забезпечує цілісне надходження неоднорідних результатів від різних засобів аналізу безпеки.

У підрозділі 3.2 розроблено алгоритм інтелектуальної класифікації вразливостей. Показано, що процес класифікації доцільно реалізовувати як багатоступеневу процедуру, яка включає формування інтегрованого вектора ознак, роботу спеціалізованих підмодулів, агрегацію часткових результатів, визначення фінального класу загрози, оцінювання впевненості моделі та формування пояснювальних атрибутів.

У підрозділі 3.3 розроблено алгоритм ризик-орієнтованого ранжування загроз. Запропонований алгоритм базується на інтеграції технічної критичності вразливості, результатів інтелектуальної класифікації, бізнес-критичності компонента, експлуатованості, міжджерельного підтвердження, чутливості даних та сили компенсуючих механізмів захисту. У результаті формується числова й категоріальна оцінка ризику, а також впорядкований перелік пріоритетів реагування.

Основними результатами третього розділу є:

– розроблення алгоритму багатоканального збору та обробки результатів сканування;

– розроблення алгоритму інтелектуальної класифікації вразливостей;

– розроблення алгоритму ризик-орієнтованого ранжування загроз;

– формалізація вхідних і вихідних даних для кожного алгоритмічного етапу; визначення критеріїв ефективності для збору, класифікації та ранжування загроз.

Отже, у третьому розділі сформовано алгоритмічну основу функціонування ІКС АВВЗ. Вона забезпечує логічний перехід від формальних моделей, розроблених у другому розділі, до програмної реалізації, функціональної архітектури та експериментальної перевірки системи, що розглядаються у четвертому розділі.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНА ВЕРИФІКАЦІЯ ІНТЕЛЕКТУАЛЬНОЇ КОМП'ЮТЕРНОЇ СИСТЕМИ

У четвертому розділі кваліфікаційної роботи здійснюється перехід від модельного та алгоритмічного опису ІКС АВВЗ до її практичної реалізації, експериментальної перевірки та оцінювання ефективності. Якщо в попередніх розділах було обґрунтовано актуальність задачі, сформовано модельну основу системи та побудовано алгоритми її функціонування, то в межах цього розділу розглядається програмний прототип, реалізований для підтвердження працездатності запропонованих рішень у прикладному середовищі аналізу безпеки вебзастосунків.

Основною метою розділу є демонстрація того, яким чином розроблені в роботі моделі та алгоритми були трансформовані у функціонуючий програмний прототип, які вхідні дані використовувалися під час експериментальної перевірки, які результати одержано в процесі тестування та яку практичну цінність має запропонована система для автоматизації процесів безпечної розробки програмного забезпечення. У межах розділу послідовно розглядаються програмна реалізація основних підсистем, організація експериментального дослідження, результати перевірки ефективності, регресійна верифікація програмного прототипу, а також практичне значення й перспективи подальшого розвитку системи.

4.1 Архітектура, програмна реалізація та інформаційне забезпечення прототипу ІКС АВВЗ

Після розроблення алгоритмів багатоканального збору даних, інтелектуальної класифікації вразливостей та ризик-орієнтованого ранжування загроз необхідно перейти до опису функціональної архітектури програмної системи, яка забезпечує практичну реалізацію цих алгоритмів. Якщо в попередніх підрозділах було визначено логіку роботи окремих аналітичних процедур, то в

межах даного підрозділу розглядається цілісна програмна організація ІКС АВВЗ як сукупності функціонально взаємопов’язаних модулів, сервісів, внутрішніх інтерфейсів і потоків даних.

У межах даної роботи під функціональною архітектурою програмної системи розуміється впорядкована структура модулів і сервісів, що реалізують основні функції ІКС АВВЗ: отримання результатів аналізу безпеки, нормалізацію та підготовку даних, інтелектуальну класифікацію вразливостей, адаптивне оцінювання ризику, формування звітів, інтеграцію із зовнішніми системами, моніторинг і керування виконанням.

Функціональна архітектура ІКС АВВЗ ґрунтується на кількох принципах. По-перше, це модульна декомпозиція, за якої кожна підсистема виконує обмежене коло функцій і може бути реалізована, протестована та модифікована незалежно від інших модулів. По-друге, це слабке зв’язування модулів, що означає взаємодію між ними переважно через стандартизовані внутрішні інтерфейси та формалізовані структури даних, а не через жорстку залежність від внутрішньої реалізації. По-третє, це орієнтація на потоки даних, оскільки ІКС АВВЗ по суті є аналітичною системою, у якій кожен наступний модуль працює з результатом попереднього. По-четверте, це масштабованість, що передбачає можливість підключення нових джерел сканування, нових моделей класифікації, нових зовнішніх баз знань і нових механізмів звітності. По-п’яте, це придатність до інтеграції з конвеєрами безпечної розробки та безперервного розгортання, що вимагає наявності зовнішніх програмних інтерфейсів, автоматизованого формування результатів і підтримки асинхронних сценаріїв обробки [24; 25; 81; 83].

У поточному прототипі така архітектура узгоджується з фактично реалізованим набором модулів, журналюванням, зберіганням, API-шаром і контурами інтеграції. Водночас спеціалізовані SIEM/SOAR-адаптери, промислова інтеграція з великими корпоративними середовищами та повномасштабна експлуатаційна валідація належать до напрямів наступного циклу розвитку, а не до функціоналу, який слід подавати як завершений промисловий продукт [36, 81].

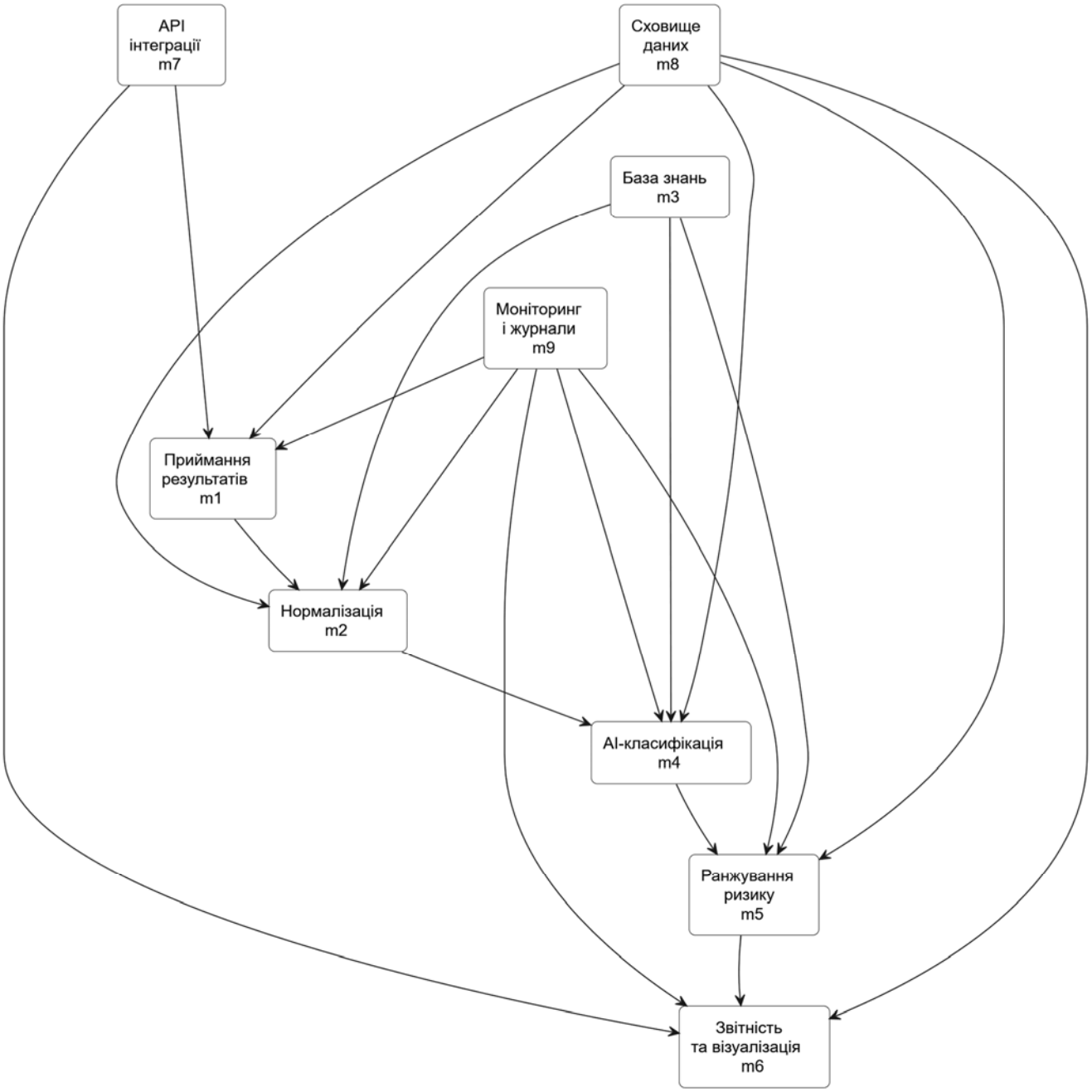


Рисунок 4.1 – Функціональна архітектура програмної системи ІКС АВВЗ

Після визначення функціональної архітектури наступним рівнем опису є структура програмного та інформаційного забезпечення ІКС АВВЗ. Вона конкретизує, як логічні функціональні модулі реалізуються у вигляді програмних компонентів, внутрішніх пакетів, сервісів, моделей даних і механізмів зберігання інформації.

Таблиця 4.1 – Функціональні модулі програмної системи ІКС АВВЗ

Модуль	Основне призначення	Вхід	Вихід
Підсистема приймання	збір результатів від сканерів	зовнішні джерела	сирі записи
Підсистема нормалізації	очищення, уніфікація, дедуплікація	сирі записи	нормалізовані записи
Підсистема баз знань	доступ до OWASP, CWE, CVE, NVD	запити модулів	довідкові дані
Підсистема класифікації	визначення типу загрози	нормалізовані записи	класифіковані загрози
Підсистема оцінювання ризику	оцінювання ризику та пріоритезація	класифіковані загрози	ранжований перелік
Підсистема звітності	формування звітів і дашбордів	ранжований перелік	звіти, візуалізації
API-підсистема	інтеграція із зовнішніми системами	API-запити	структуровані відповіді
Підсистема зберігання	довготривале та проміжне зберігання	усі типи даних	записи в сховищі
Підсистема журналювання	контроль функціонування	системні події	журнали, метрики

У межах роботи програмне забезпечення розглядається як сукупність компонентів приймання даних, нормалізації, доступу до баз знань, класифікації, оцінювання ризику, звітності, API-взаємодії, зберігання та журналювання. Інформаційне забезпечення охоплює сирі результати сканування, нормалізовані записи, класифіковані загрози, ризикові оцінки, бази знань, журнали подій і конфігураційні параметри [81, 82].

Така побудова забезпечує розділення прикладної логіки, AI-компонентів, сховища даних, зовнішніх інтеграцій і засобів представлення результатів. Структура програмного та інформаційного забезпечення ІКС АВВЗ наведена на рисунку 4.2, а її основні компоненти – у таблиці 4.2.

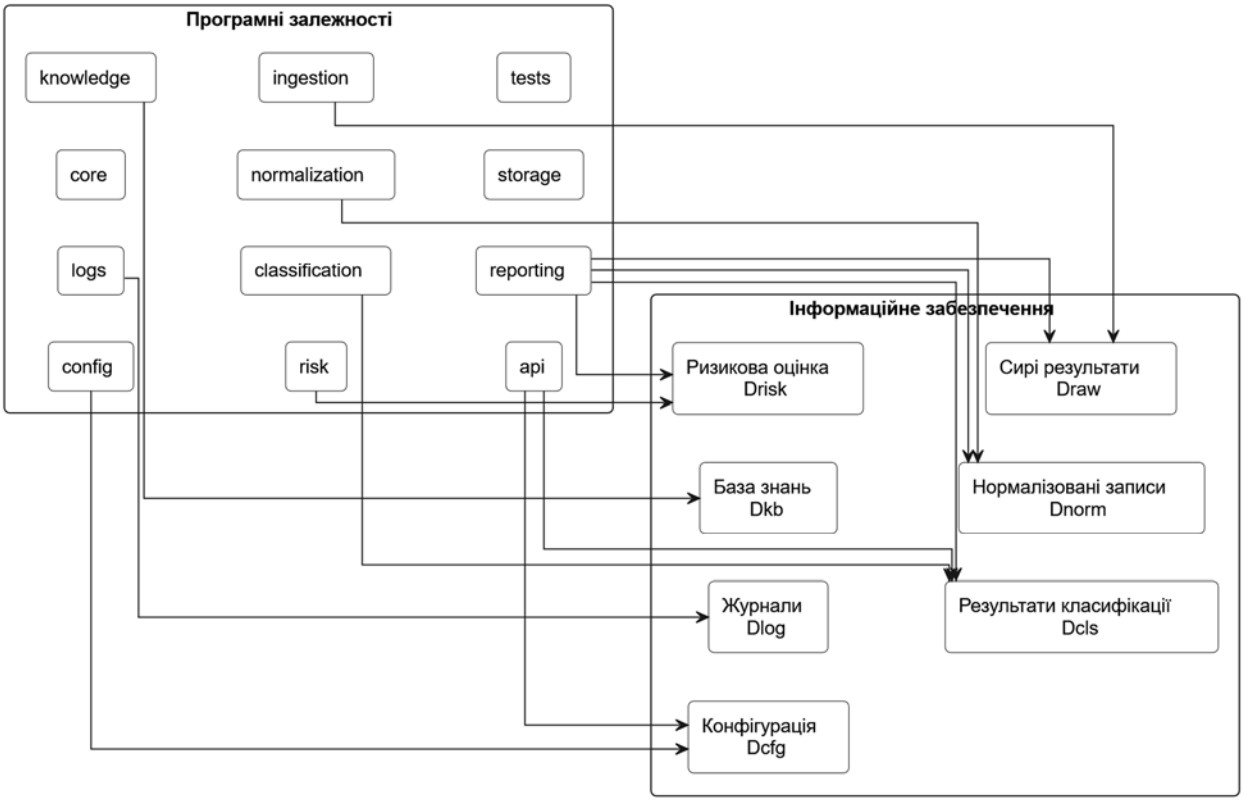


Рисунок 4.2 – Структура програмного та інформаційного забезпечення ІКС АВВЗ

Таблиця 4.2 – Основні компоненти програмного та інформаційного забезпечення ІКС АВВЗ

Компонент	Призначення	Основні дані
Приймання результатів	Збір даних від сканерів	Сирі результати
Нормалізація	Очищення, уніфікація, дедуплікація	Нормалізовані записи
База знань	Довідкове та семантичне збагачення	CVE, CWE, CAPEC, OWASP
Класифікація	Аналіз і визначення типу загрози	Класифіковані записи
Оцінювання ризику	Обчислення ризику й ранжування	Ризикові оцінки
Звітність	Представлення результатів	Звіти, дашборди
API	Інтеграція із зовнішніми системами	Запити та відповіді
Сховище даних	Довготривале та проміжне зберігання	Усі типи записів
Журнали і конфігурація	Контроль і керування	Службові дані

На основі наведеної архітектури реалізовано програмний прототип ІКС АВВЗ. Прототип відтворює основні функціональні контури системи та забезпечує

повний цикл обробки безпекових результатів: приймання даних із зовнішніх інструментів аналізу, нормалізацію, семантичне збагачення, класифікацію, оцінювання ризику, зберігання результатів і формування аналітичного звіту.

У фактичній реалізації використано Python 3.12, FastAPI, Pydantic v2, scikit-learn, SQLite, Docker і Docker Compose. Такий стек забезпечує відтворюваність розгортання, придатність до експериментальної перевірки, підтримку REST API та можливість інтеграції з DevSecOps-середовищем [74,81 - 83]. Структуру реалізованого програмного прототипу наведено на рисунку 4.3, а основні програмні підсистеми – у таблиці 4.3.

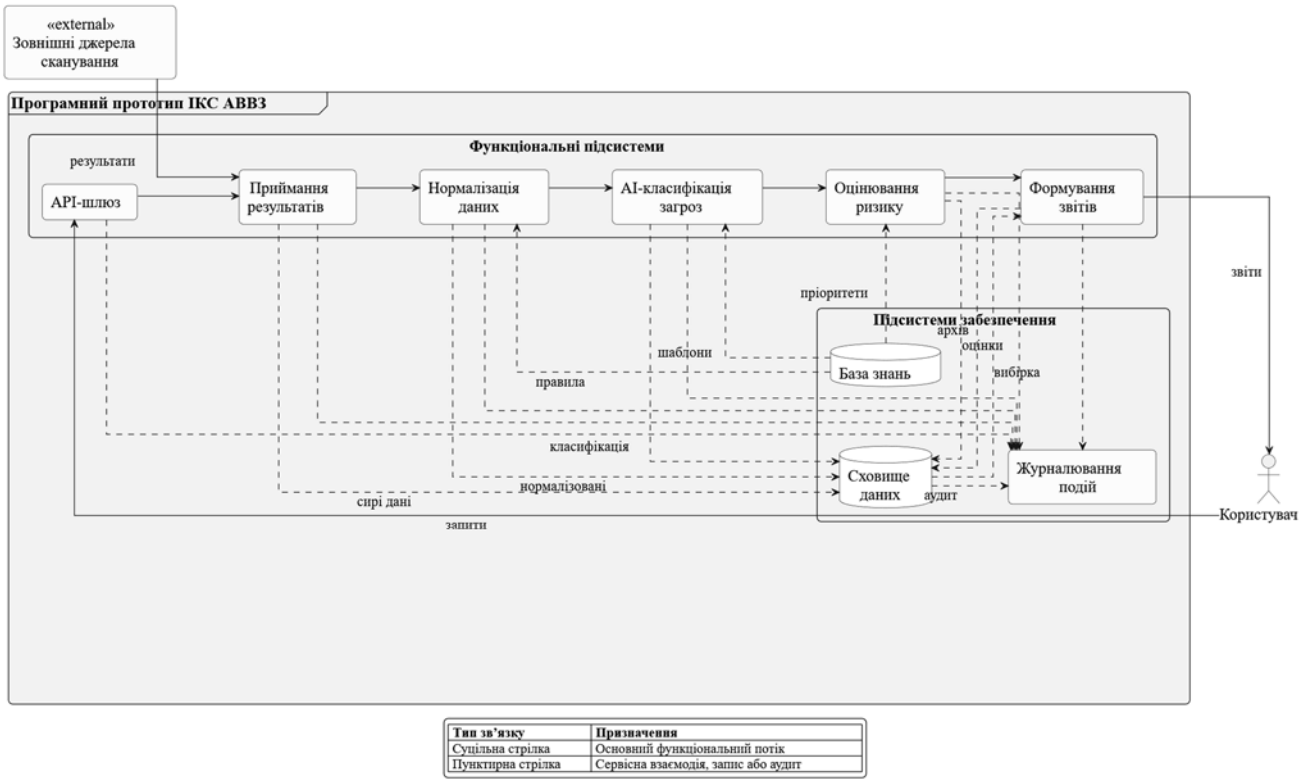


Рисунок 4.3 – Структура програмного прототипу інтелектуальної комп'ютерної системи автоматичного виявлення вразливостей вебзастосунків та класифікації загроз

Таблиця 4.3 – Основні програмні підсистеми реалізованого прототипу

Підсистема	Основне призначення	Основний результат

Приймання результатів	імпорт і валідація звітів сканерів	сирі записи
Нормалізація	очищення, уніфікація, дедуплікація	нормалізовані записи
База знань	збагачення та семантичне зіставлення	довідкові відповідності
Класифікація	визначення типу загрози	клас загрози та впевненість
Оцінювання ризику	інтегральна оцінка небезпеки	ризиковий бал і категорія
Сховище	зберігання проміжних і фінальних даних	структуровані записи
API	інтеграція із зовнішніми системами	програмний доступ
Звітність	формування результатів аналізу	звіти та аналітичні подання
Журналювання	контроль і трасування виконання	журнали й технічні події

Наведені на рисунку 4.3 і в таблиці 4.3 програмні підсистеми відображають узагальнений рівень побудови прототипу ІКС АВВЗ. Фактична реалізація має детальнішу внутрішню структуру, що охоплює базові функціональні модулі, допоміжні сервіси обробки, інтеграційні компоненти, засоби журналювання, зберігання, керування доступом і транспортної взаємодії. Тому подання у вигляді дев’яти підсистем слід розглядати як узагальнену схему прототипу, тоді як експериментальна верифікація враховує його повний фактичний склад.

Фактична реалізація програмного прототипу охоплює приймання результатів аналізу з каналів SAST, DAST, IAST, SCA, LOG, HTTP, CONFIG та CONTAINER, їх нормалізацію, кореляцію, класифікацію, розрахунок ризику, ранжування, збереження результатів, журналювання, формування JSON-, HTML- та PDF-звітів, а також надання доступу до основних функцій системи через REST API. Така реалізація підтверджує, що запропоновані у другому та третьому розділах моделі й алгоритми не лише описані формально, а й реалізовані у вигляді працездатного дослідного програмного прототипу.

Класифікаційний контур прототипу реалізовано як гібридну схему, що поєднує текстові, контекстні, джерельні та структурні ознаки, а також включає компоненти code-graph аналізу й локального трансформерно-семантичного опрацювання для поглиблення представлення вразливостей. Така побудова дозволяє поєднати формальні ознаки безпекових записів із семантичним змістом технічних описів і контекстом джерела, з якого отримано знахідку.

4.2 Організація експериментального дослідження

Експериментальне дослідження в межах даної кваліфікаційної роботи було організовано з метою перевірки працездатності програмного прототипу ІКС АВВЗ, а також оцінювання ефективності запропонованих моделей і алгоритмів у задачах автоматизованого виявлення вразливостей вебзастосунків, інтелектуальної класифікації загроз та ризик-орієнтованого ранжування результатів аналізу. У контексті цієї роботи експериментальне дослідження мало не лише демонстраційний, а й верифікаційний характер. Його завданням було підтвердити, що програмний прототип коректно приймає результати від різних джерел аналізу безпеки, модуль нормалізації зводить неоднорідні записи до єдиного подання, модуль класифікації визначає тип вразливості або кіберзагрози, модуль ризикового оцінювання забезпечує змістовне ранжування загроз, а система в цілому придатна для подальшого використання в середовищах безпечної розробки та безперервного розгортання.

Для цього в процесі експерименту було поставлено такі завдання: перевірити коректність багатоканального збирання та інтеграції результатів аналізу безпеки; оцінити якість нормалізації неоднорідних вхідних записів; перевірити працездатність алгоритму класифікації вразливостей; оцінити якість ризик-орієнтованого ранжування загроз; дослідити узгодженість вихідних результатів системи з експертною логікою пріоритезації вразливостей; перевірити загальну стабільність функціонування програмного прототипу.

Для забезпечення репрезентативності використовувалася не одна однорідна вибірка, а комбінована множина вхідних даних, яка включала результати зовнішніх

інструментів аналізу безпеки, еталонний корпус маркованих текстових описів, прикладний наскрізний набір записів для двох вебзастосунків із дубльованими та неоднозначними кейсами, а також довідкові знання й контекстні атрибути, що застосовувалися для семантичного збагачення та адаптивного оцінювання ризику.

Перед запуском експериментів усі дані проходили процедуру попередньої підготовки, яка включала усунення дублікатів технічних записів, видалення пошкоджених або неповних записів, приведення форматів до єдиної схеми, формування контрольних відповідностей для частини набору та зіставлення з контекстною інформацією про компоненти застосунку. Експериментальне дослідження проводилося у програмному середовищі, достатньому для запуску прототипу ІКС АВВЗ, модулів обробки даних та компонентів штучного інтелекту. До складу експериментального середовища входили локальна обчислювальна платформа для запуску Python-модулів, ізольоване віртуальне середовище .venv, засоби контейнеризації, локальне сховище даних, вбудовані довідкові бази знань, модулі журналювання запусків і механізми збирання технічної статистики [74,79,80, 83].

Для забезпечення повноти перевірки експериментальна перевірка була організована у вигляді п’яти сценаріїв, кожен із яких оцінював окремий аспект функціонування ІКС АВВЗ: багатоканальне приймання результатів, нормалізацію та дедуплікацію, інтелектуальну класифікацію, ризик-орієнтоване ранжування та формування звітів. Для кожного запуску фіксувалися кількість прийнятих і валідних записів, кількість нормалізованих і класифікованих результатів, кількість проранжованих загроз, часові характеристики окремих етапів обробки, а також службові події та помилки. Така організація експерименту дає змогу оцінити систему не як набір ізольованих модулів, а як цілісний програмний прототип, придатний до використання в умовах, наближених до реального середовища безпечної розробки [76].

Додатково виконано регресійну перевірку працездатності програмного прототипу в ізольованому віртуальному середовищі .venv. Перевірка охоплювала основні функціональні групи прототипу: приймання та нормалізацію даних,

класифікацію, ризик-орієнтоване оцінювання, врахування невизначеності, формування звітів, REST API, журналювання та збереження результатів. За результатами тестового запуску всі перевірки завершилися успішно: 121 тест виконано успішно, помилок не зафіксовано.

Таблиця 4.4 – Основні групи вхідних даних експериментального дослідження

Група даних	Реалізація у прототипі	Призначення
Результати сканерів	sample_semgrep_report.json, sample_zap_report.json, sample_iast_report.json, sample_container_report.json, sample_sca_report.json, sample_log_report.json, sample_http_report.json, sample_config_report.json	перевірка багатоканального приймання, пакетної маршрутизації та уніфікації
Еталонний корпус	195 маркованих текстових описів за 7 класами загроз	навчання та оцінювання ML-класифікатора
Прикладний наскрізний набір	17 базових записів для 2 вебзастосунків + 2 дублікати (N_in = 19, N_norm = 12)	перевірка повного контуру нормалізації, кореляції, ранжування та звітності
Довідкові дані і контекст	THREAT_KNOWLEDGE, довідкові знання threat_intel та контекстні атрибути application_id, location, extra	семантичне збагачення, збагачення довідковими відповідностями та адаптивне ризикове оцінювання

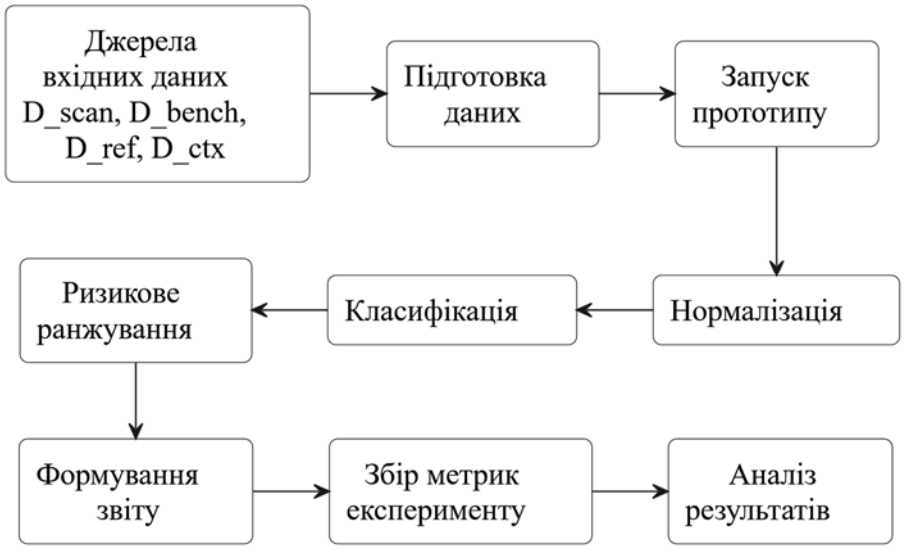


Рисунок 4.4 – Організаційна схема експериментального дослідження ІКС АВВЗ

Таблиця 4.5 – Основні показники, що фіксувалися під час експерименту

Показник	Зміст
K_{arch}	покриття архітектурних рівнів, базових модулів і розширювальних інтелектуальних підсистем
Q_{parser}	успішність обробки гетерогенних звітів і багатоканальних пакетів
Q_{kb}	повнота збагачення знаннями OWASP/CWE/CAPEC
Q_{sem}	покриття семантичного шару для фінальних знахідок
Q_{ref}	покриття збагачення довідковими посиланнями CVE/CWE/CAPEC
N_{in}	кількість отриманих вхідних записів
N_{valid}	кількість валідних записів після фільтрації
N_{norm}	кількість нормалізованих записів
N_{cls}	кількість класифікованих загроз
N_{risk}	кількість проранжованих загроз
R_{fmt}	кількість підтверджених форматів звітності
A_{api}	кількість доступних кінцевих точок REST API
$S_{persist}$	успішність запису та повторного читання результатів зі сховища
T_{proc}	час проходження основних етапів
Err_{sys}	кількість технічних помилок або збоїв

Таблиця 4.6 – Результати регресійної перевірки працездатності програмного прототипу ІКС АВВЗ

Група перевірок	Зміст перевірки	Результат
Приймання даних	перевірка обробки вхідних результатів сканування	виконано успішно
Нормалізація	перевірка уніфікації, очищення та підготовки записів	виконано успішно
Класифікація	перевірка роботи класифікаційного контуру	виконано успішно
Оцінювання ризику	перевірка CVSS, confidence, context, corroboration, mitigation, uncertainty	виконано успішно
Звітність	формування JSON-, HTML- та PDF-звітів	виконано успішно
REST API	перевірка доступу до основних функцій системи	виконано успішно
Журналювання і збереження	перевірка запису подій, результатів і службових даних	виконано успішно
Регресійні тести	повний тестовий запуск у .venv	121 пройдено успышно

Таким чином, у підрозділі 4.2 було визначено організацію експериментального дослідження, структуру вхідних даних, основні сценарії перевірки та систему показників, за якими оцінювалася ефективність функціонування ІКС АВВЗ. Сформована експериментальна база дала змогу перейти від опису умов дослідження до безпосереднього аналізу кількісних і якісних результатів роботи програмного прототипу.

4.3 Результати експериментального дослідження, перевірка ефективності системи

За результатами експериментальної перевірки встановлено, що запропонований програмний прототип інтелектуальної комп'ютерної системи автоматичного виявлення вразливостей вебзастосунків та класифікації загроз забезпечує повний цикл оброблення вхідних безпекових записів: приймання результатів сканування, нормалізацію, інтелектуальну класифікацію, ризик-орієнтоване ранжування та формування підсумкового звіту [74, 74].

У межах експерименту було оброблено 19 вхідних записів, з яких усі 19 пройшли первинну валідацію. Після нормалізації та дедуплікації 12 записів були доведені до стану фінальних нормалізованих артефактів, 10 отримали первинне класифікаційне рішення, а для 12 записів було сформовано фінальну ризикову оцінку. Узагальнені результати проходження записів через контур ІКС АВВЗ наведено в таблиці 4.7.

Таблиця 4.7 – Узагальнені результати проходження записів через контур ІКС АВВЗ

Показник	Позначення	Значення
Кількість вхідних записів	N_{in}	19
Кількість валідних записів	N_{valid}	19
Кількість нормалізованих записів	N_{norm}	12
Кількість класифікованих записів	N_{cls}	10
Кількість проранжованих загроз	N_{risk}	12
Кількість відхилених записів	N_{rej}	0
Кількість записів, переданих на додатковий перегляд	N_{review}	2

Показник $N_{review} = 2$ характеризує лише записи, що були передані на додатковий перегляд через недостатню визначеність первинного класифікаційного рішення. Його не слід ототожнювати з ширшим механізмом ручної верифікації у

фінальній політиці прийняття рішень, де враховується також рівень ризику та епістемічна невизначеність.

Експериментальна перевірка була спрямована не лише на оцінювання точності класифікації, а й на підтвердження працездатності запропонованого контуру “нормалізація – класифікація – ризик-орієнтоване ранжування”. Саме тому оцінювалися не тільки класичні метрики Accuracy, Precision, Recall і F1-score, а й стабільність моделі за результатами крос-валідації та поведінка показника невизначеності для типових і нетипових вхідних випадків.

Для оцінювання якості інтелектуальної класифікації використано стандартні метрики багатокласової класифікації: Accuracy, Precision, Recall та F1-score. Отримані значення наведено в таблиці 4.8 [75, 77, 77].

Таблиця 4.8 – Показники якості класифікації вразливостей

Метрика	Позначення	Значення
Точність класифікації	Accuracy	0,8718
Прецизійність	Precision	0,8813
Повнота	Recall	0,8718
F1-міра	F1-score	0,8695
5-fold крос-валідація	CV F1	0,9475 ± 0,0369

Отримані результати свідчать про достатньо високу якість фінального гібридного контуру. Значення F1-міри 0,8695 вказує на збалансованість між прецизійністю та повнотою, що є критично важливим для задач кібербезпеки, де небажаними є як пропущені загрози, так і надмірна кількість хибнопозитивних спрацювань [77, 77].

Різниця між значенням F1-міри на фінальній тестовій вибірці та середнім значенням F1-міри за результатами п’ятиблокової крос-валідації пояснюється тим, що фінальна тестова вибірка містила складніші та менш типові приклади, зокрема прикордонні випадки між класами загроз і OOD-записи. Тому результат 0,8695 відображає більш консервативну оцінку якості моделі в умовах, наближених до

практичного використання, тоді як крос-валідація характеризує середню стабільність моделі на збалансованіших підвибірках [75, 76, 80].

Для об’єктивнішого оцінювання ефективності запропонованого підходу виконано порівняння з базовими підходами до класифікації. Результати наведено в таблиці 4.9.

Таблиця 4.9 – Порівняння якості класифікації з базовими підходами

Підхід	Прецизійність	Повнота	F1-міра
Базовий сигнатурно-правиловий підхід	0,7934	0,7436	0,7360
Поточний ансамблевий ML-підхід	0,8960	0,8462	0,8615
Базовий графовий підхід	0,7761	0,6923	0,6663
Базовий підхід на основі великої мовної моделі	0,7979	0,7692	0,7596
Фінальний гібридний контур	0,8813	0,8718	0,8695

Порівняння показало, що фінальний гібридний контур забезпечує найкращий баланс між прецизійністю, повнотою та F1-мірою. Порівняно з базовим сигнатурно-правиловим підходом значення F1-міри зросло з 0,7360 до 0,8695, що відповідає приблизно 18,1 % відносного приросту. Порівняно з поточним ансамблевим ML-підходом приріст є меншим, однак фінальний контур демонструє кращу збалансованість за рахунок вищої повноти. Це підтверджує доцільність поєднання текстових, контекстних, джерельних, структурних, графових і трансформерно-семантичних ознак у межах єдиного інтелектуального контуру.

На рисунку 4.5 узагальнено ключові результати експериментальної перевірки: часові характеристики етапів оброблення, показники інтервальної оцінки ризику, порівняння якості класифікації з базовими підходами та відмінність епістемічної невизначеності для ID- та OOD-випадків. Таке подання дозволяє оцінити систему не лише за якістю класифікації, а й за обчислювальною ефективністю, стійкістю до нетипових вхідних даних та здатністю підтримувати ризик-орієнтоване прийняття рішень.

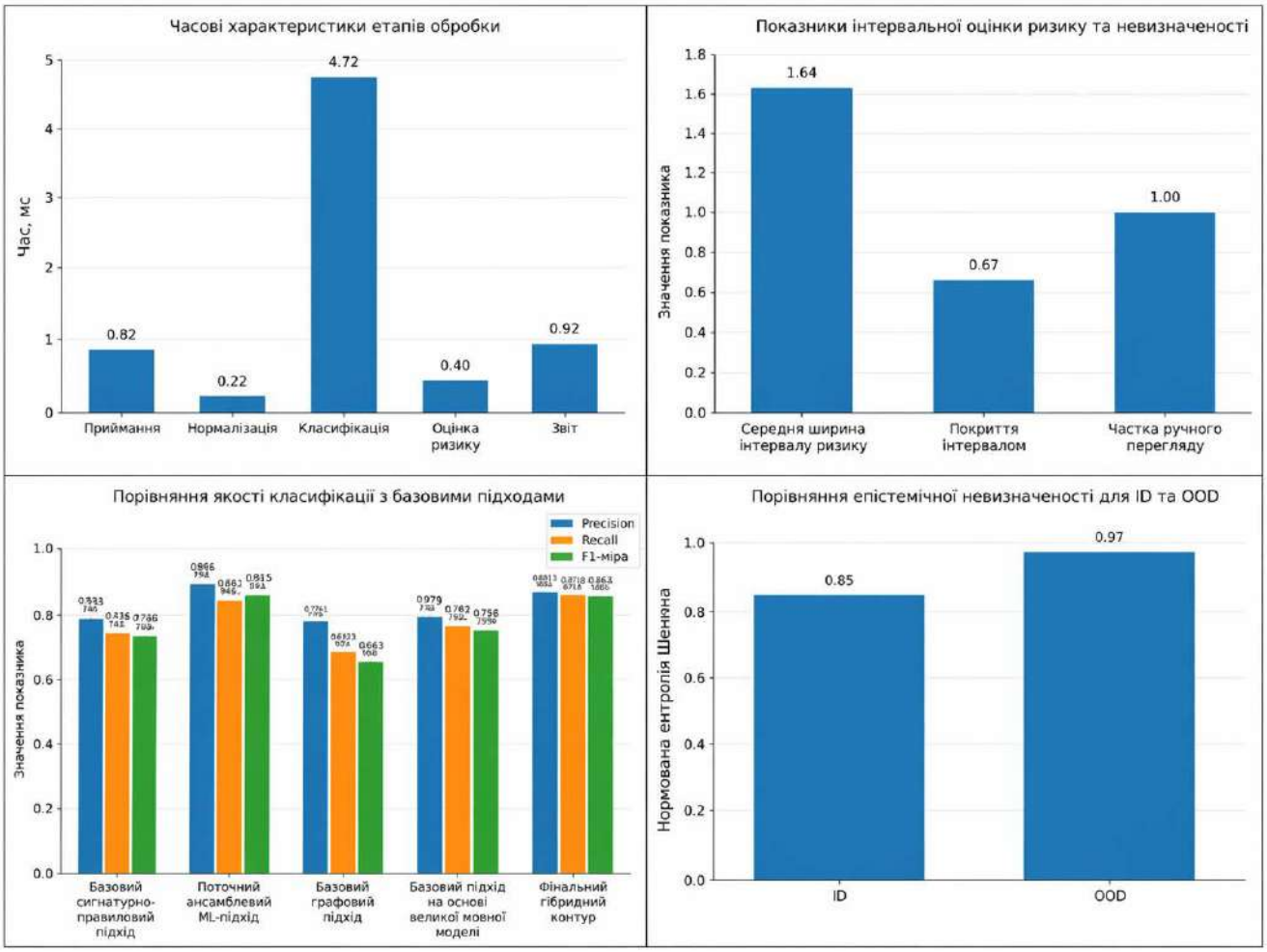


Рисунок 4.5 – Порівняльні результати ефективності, часових характеристик та невизначеності в ІКС АВВЗ

Одним із важливих результатів функціонування системи є зменшення інформаційної надмірності за рахунок нормалізації та дедуплікації записів. У межах експерименту коефіцієнт дедуплікації становив 0,3684, а коефіцієнт збереження інформативності – 0,8333. Це означає, що система усунула 36,84 % надлишкових або дубльованих повідомлень і водночас зберегла 83,33 % інформативних результатів у стані, придатному для подальшої класифікації та ризик-орієнтованого ранжування. Результати наведено в таблиці 4.10.

Таблиця 4.10 – Показники ефективності нормалізації

Показник	Позначення	Значення
Коефіцієнт дедуплікації	(K_{dedup})	0,3684
Коефіцієнт збереження інформативності	(K_{info})	0,8333

Окремо перевірено якість ризик-орієнтованого ранжування. На відміну від простого сортування за CVSS, запропонований підхід враховує контекст вебзастосунку, бізнес-критичність компонента, підтвердження з різних джерел та компенсуючі механізми захисту. Результати наведено в таблиці 4.11.

Таблиця 4.11 – Показники якості ризик-орієнтованого ранжування

Показник	Позначення	Значення
Якість верхніх k позицій	(Q_{top-k})	1,0000 $(k = 5)$
Узгодженість з експертною оцінкою	(Q_{expert})	0,9167

Значення $Q_{top-k} = 1{,}0000$ свідчить, що всі реально критичні або експертно високопріоритетні загрози були розміщені у верхній частині ранжованого списку. Значення $Q_{expert} = 0{,}9167$ підтверджує високу узгодженість результатів системи з експертною логікою пріоритезації. Технічну ефективність прототипу оцінено за часовими характеристиками основних етапів оброблення. Деталізовані значення, узагальнені на рисунку 4.5, наведено в таблиці 4.12.

Таблиця 4.12 – Часові характеристики функціонування прототипу

Етап	Час
Приймання результатів	0,04 мс
Нормалізація	0,75 мс
Класифікація	41,65 мс
Оцінювання ризику	132,58 мс

Кінець таблиці 4.12

Формування звіту	2,00 мс
Загальний час	177,01 мс
Середній час на 1 запис	10,41 мс

Найбільш ресурсомістким етапом є оцінювання ризику, що зумовлено врахуванням контекстних, бізнесових та безпекових параметрів. Водночас загальний час оброблення експериментального набору становить 177,01 мс, а середній час на один запис – 10,41 мс, що підтверджує придатність прототипу для напівавтоматизованих сценаріїв аналізу безпеки.

Додатково проведено перевірку блоку епістемічно-алеаторної декомпозиції ризику. У цій частині ризик подавався не лише як скалярна оцінка, а як поєднання базової ризикової складової та епістемічної невизначеності, вираженої через нормовану ентропію Шеннона. Кількісні результати наведено в таблиці 4.13 [63-65].

Таблиця 4.13 – Показники епістемічно-алеаторної декомпозиції ризику

Показник	Значення
Коефіцієнт впливу невизначеності	0,3
Середня ширина інтервалу	1,283
Покриття інтервалом	0,417
Частка випадків на ручний перегляд	1,0000

Для перевірки реакції системи на нетипові вхідні приклади виконано OOD-дослідження. Середнє значення нормованої ентропії Шеннона для типових ID-випадків становило 0,760, тоді як для нетипових OOD-випадків – 0,877. Зростання епістемічної невизначеності для OOD-прикладів підтверджує здатність системи виявляти нестандартні або недостатньо представлені у навчальному розподілі випадки та переводити їх у режим ручної верифікації [63-65].

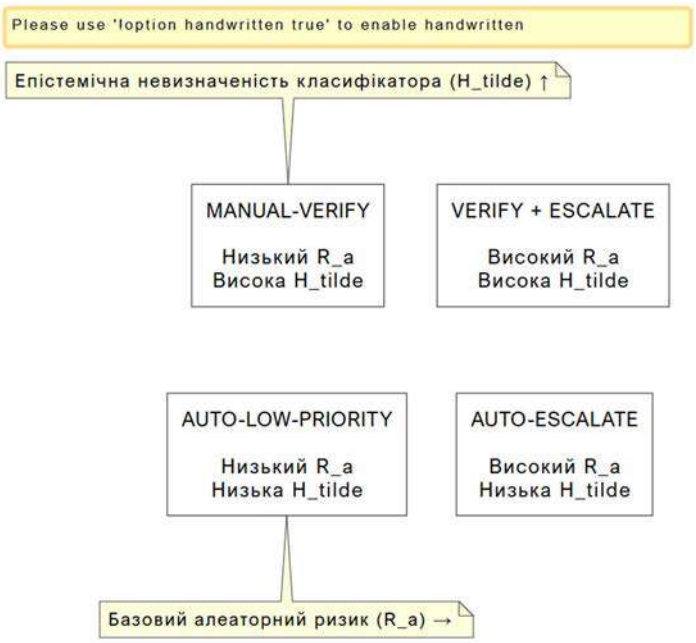


Рисунок 4.6 – Матриця прийняття рішень за базовим алеаторним ризиком R_a та епістемічною невизначеністю класифікатора $\widetilde{H}$

Окремий аналіз модуля політики показав, що декомпозиція на базовий ризик та невизначеність дозволяє формалізувати чотири операційні режими реагування: автоматичну ескалацію для впевнено високих загроз, перевірку з подальшою ескалацією для високих, але невизначених випадків, автоматичне віднесення до нижчого пріоритету для впевнено низьких ризиків та ручну перевірку для невизначених випадків із нижчим базовим ризиком. Такий механізм підвищує інтерпретованість результатів і дозволяє приймати рішення не лише за рівнем ризику, а й за рівнем невизначеності класифікатора.

Проведене експериментальне дослідження показало, що запропонований програмний прототип ІКС АВВЗ забезпечує цілісний цикл інтелектуального аналізу вразливостей вебзастосунків. Система коректно інтегрує результати з різних джерел, зменшує інформаційну надмірність, виконує уніфікацію записів, класифікує основні типи вебзагроз, формує ризик-орієнтований порядок пріоритетів і генерує підсумковий звіт. Приклад частини автоматично сформованого звіту наведено на рисунку 4.7.

Табличний огляд знахідок

Головна таблиця поєднує технічну, аналітичну та операційну інформацію: джерела підтвердження, ризиковий інтервал, невизначеність і policy-рішення.

#ID / ЛОКАЦІЯ	ЗНАХІДКА / КОМПОНЕНТ	КЛАС / OWASP / CWE/CVE	ДЖЕРЕЛА	РИЗИК	POLICY	ML / UNCERTAINTY
435d876bafb43f4e /app/auth/login.py:47	Runtime SQL Injection in login endpoint webapp-01	SQL_INJECTION A03:2021 · CWE CWE-89 · CVE -	SAST IAST primary=SAST · count=2	CRITICAL · 10.00 CVSS 9.80 · interval 9.17-10.00	VERIFY_AND_ESCALATE Q=B · SLA 8h · review Так	52% Висока · H~ 0.751
6d9068c1104e9320 /api/users/*/profile	IDOR in user profile endpoint webapp-01	BROKEN_ACCESS_CONTROL A01:2021 · CWE CWE-284 · CVE -	DAST primary=DAST · count=1	CRITICAL · 10.00 CVSS 8.10 · interval 8.70-10.00	VERIFY_AND_ESCALATE Q=B · SLA 8h · review Так	70% Висока · H~ 0.579
43d2f0d7c87989f6 /app/comments/views.py:88	Stored XSS in comment field webapp-02	XSS A03:2021 · CWE CWE-79 · CVE -	SAST LOG primary=SAST · count=2	CRITICAL · 10.00 CVSS 8.00 · interval 8.21-10.00	VERIFY_AND_ESCALATE Q=B · SLA 8h · review Так	51% Висока · H~ 0.751
6406c34662803512 /api/products/search	SQL Injection via search API webapp-02	SQL_INJECTION A03:2021 · CWE CWE-89 · CVE -	DAST primary=DAST · count=1	CRITICAL · 10.00 CVSS 9.10 · interval 9.28-10.00	VERIFY_AND_ESCALATE Q=B · SLA 8h · review Так	52% Висока · H~ 0.753
846917fa66078564 /api/orders/*	Horizontal privilege escalation in orders API webapp-02	BROKEN_ACCESS_CONTROL A01:2021 · CWE CWE-284 · CVE -	DAST HTTP primary=DAST · count=2	CRITICAL · 10.00 CVSS 8.30 · interval 8.84-10.00	VERIFY_AND_ESCALATE Q=B · SLA 8h · review Так	72% Висока · H~ 0.565
e858f560a7025a8d /api/webhooks/register	SSRF in webhook URL parameter webapp-01	SSRF A10:2021 · CWE CWE-918 · CVE -	DAST primary=DAST · count=1	CRITICAL · 9.28 CVSS 8.60 · interval 7.79-9.28	VERIFY_AND_ESCALATE Q=B · SLA 8h · review Так	62% Висока · H~ 0.636
e36b799b00c2f1e3 pom.xml:line 45	Vulnerable dependency: log4j 2.14.1 webapp-01	VULNERABLE_COMPONENT A06:2021 · CWE CWE-937 · CVE CVE-2021-44228	SCA primary=SCA · count=1	CRITICAL · 9.05 CVSS 10.00 · interval 7.47-9.05	VERIFY_AND_ESCALATE Q=B · SLA 8h · review Так	58% Висока · H~ 0.706
7ac1342c3e5a36c0 /search?q=	Reflected XSS in search parameter webapp-01	XSS A03:2021 · CWE CWE-79 · CVE -	DAST HTTP primary=DAST · count=2	HIGH · 8.63 CVSS 7.40 · interval 7.21-8.63	VERIFY_AND_ESCALATE Q=B · SLA 8h · review Так	62% Висока · H~ 0.656
0a6e7ba698dc3f44 requirements.txt:3	Outdated library: Django 3.0.7 webapp-01	VULNERABLE_COMPONENT A06:2021 · CWE CWE-937 · CVE CVE-2021-35042	SCA primary=SCA · count=1	HIGH · 7.97 CVSS 7.50 · interval 6.41-7.97	MANUAL-VERIFY Q=D · SLA 24h · review Так	49% Висока · H~ 0.811
57cd724c9e8c5333 /config/settings.py:12	Debug mode enabled in production config webapp-01	SECURITY_MISCONFIGURATION A05:2021 · CWE CWE-16 · CVE -	SAST CONFIG primary=SAST · count=2	HIGH · 7.38 CVSS 5.30 · interval 6.03-7.38	MANUAL-VERIFY Q=D · SLA 24h · review Так	54% Висока · H~ 0.744
3762da33148057fb /app/middleware/session.py:23	Missing HttpOnly flag on session cookie webapp-01	UNKNOWN A05:2021 · CWE CWE-16 · CVE -	SAST primary=SAST · count=1	MEDIUM · 6.77 CVSS 5.40 · interval 5.38-6.77	MANUAL-VERIFY Q=D · SLA 24h · review Так	36% Висока · H~ 0.859
a1068d5ac34d0a50 /app/middleware/headers.py	Missing X-Frame-Options header webapp-02	UNKNOWN A05:2021 · CWE CWE-16 · CVE -	SAST primary=SAST · count=1	MEDIUM · 5.53 CVSS 4.30 · interval 4.36-5.53	MANUAL-VERIFY Q=D · SLA 24h · review Так	31% Висока · H~ 0.898

Рисунок 4.7 – Фрагмент автоматично сформованого звіту ІКС АВВЗ за результатами аналізу вебзастосунку

4.4 Практичне значення та перспективи застосування розробленої системи

Практичне значення розробленої інтелектуальної комп’ютерної системи автоматичного виявлення вразливостей вебзастосунків та класифікації загроз полягає в тому, що вона забезпечує не лише технічне агрегування результатів безпекового аналізу, а й їх цілісну інтелектуальну обробку, адаптивну інтерпретацію та підтримку прийняття рішень щодо усунення загроз. На відміну від традиційних підходів, за яких фахівець з кібербезпеки змушений окремо

аналізувати результати різних сканерів, вручну зіставляти дублікати, оцінювати достовірність знахідок і самостійно визначати пріоритети реагування, запропонована система автоматизує значну частину цієї роботи та зменшує обсяг рутинних аналітичних операцій.

У межах цієї роботи практичне значення системи доцільно розглядати на кількох рівнях: технічному, організаційному, експлуатаційному, освітньо-науковому та стратегічному. На технічному рівні ІКС АВВЗ дозволяє об’єднати в єдиному програмному контурі результати аналізу, отримані від різних засобів перевірки безпеки вебзастосунків. Це усуває проблему фрагментованості безпекових звітів, зменшує дублювання інформації, забезпечує єдине середовище її інтерпретації та підвищує узгодженість аналізу загроз.

На організаційному рівні система сприяє зменшенню навантаження на фахівців із кібербезпеки, оскільки скорочує обсяг ручної аналітичної роботи, пов’язаної з первинним розбором, сортуванням, зіставленням і пріоритезацією вразливостей. У практичному сенсі це означає, що ІКС АВВЗ може виконувати роль аналітичного посередника між сканерами безпеки та командою, яка відповідає за виправлення вразливостей.

Одним із найбільш перспективних напрямів практичного застосування ІКС АВВЗ є її інтеграція в процеси безпечної розробки програмного забезпечення та автоматизованого контролю якості. У сучасному життєвому циклі розробки особливо важливим є не просто факт виявлення вразливості, а швидкість, з якою ця інформація вбудовується в конвеєр розробки, тестування, релізу та супроводу [24, 25]. У такому контексті розроблена система може використовуватися для автоматичного приймання результатів перевірки після кожного запуску системи, класифікації знайдених вразливостей без ручного розбору, формування пріоритетів виправлення ще до стадії релізу та генерації структурованих звітів для команди розробки.

У такому контексті розроблена система може використовуватися для автоматичного приймання результатів перевірки після кожного запуску системи, класифікації знайдених вразливостей без ручного розбору, формування пріоритетів виправлення ще до стадії релізу та генерації структурованих звітів для команди

розробки [55, 63]. Це дає змогу розглядати ІКС АВВЗ не лише як окреме рішення для безпеки вебзастосунків, а як основу для побудови ширшого класу інтелектуальних систем аналізу вразливостей.

Водночас розроблений прототип не позиціонується як завершений промисловий продукт, а є дослідним програмним прототипом, призначеним для підтвердження практичної здійсненності запропонованих моделей і алгоритмів. Його поточна реалізація підтверджує працездатність основного аналітичного контуру ІКС АВВЗ, однак подальший розвиток системи передбачає розширення набору підтримуваних форматів промислових сканерів, збільшення обсягу навчальних і тестових даних, поглиблену перевірку графових і трансформерно-семантичних компонентів, а також інтеграцію з реальними CI/CD-, SIEM- та DevSecOps-середовищами.

До обмежень поточної версії програмного прототипу належать залежність якості класифікації від репрезентативності навчальної вибірки, обмежений обсяг експериментального набору даних, необхідність подальшого розширення підтримки реальних форматів звітів промислових SAST-, DAST- і SCA-інструментів, а також потреба в додатковій валідації моделі на великих наборах даних із реальних середовищ безпечної розробки. Запропонована архітектура системи передбачає можливість подальшого підключення нових джерел даних, моделей класифікації та механізмів коригування ризикової оцінки.

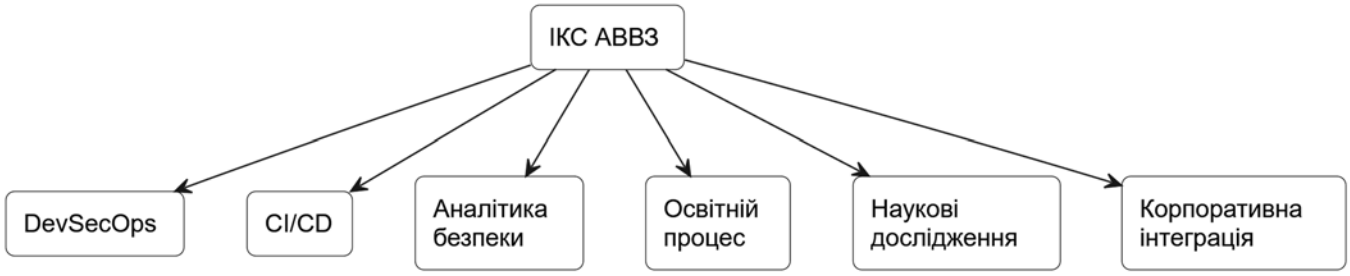


Рисунок 4.8 – Основні напрями практичного застосування ІКС АВВЗ

У таблиці 4.14 узагальнено практичне значення та перспективи застосування ІКС АВВЗ за основними напрямами використання.

Таблиця 4.14 – Практичне значення та перспективи застосування ІКС АВВЗ

Напрям	Практичне значення
Технічне використання	інтеграція результатів сканування, класифікація, ранжування
DevSecOps / CI/CD	автоматизація контролю безпеки в конвеєрі розробки
Робота фахівця з безпеки	зменшення шуму, підтримка пріоритезації
Освітнє середовище	навчальна та лабораторна платформа
Наукові дослідження	база для розвитку AI-моделей у кібербезпеці
Корпоративне середовище	інтеграція з інфраструктурою безпечної розробки
Подальший розвиток	розширення джерел, моделей, контекстного аналізу

Для чіткішого відмежування поточної реалізації від перспектив подальшого розвитку в таблиці 4.15 наведено основні обмеження прототипу та відповідні напрями його вдосконалення.

Таблиця 4.15 – Обмеження поточної версії прототипу та напрями подальшого розвитку

Обмеження поточної версії	Напрям подальшого розвитку
Обмежений експериментальний набір даних	розширення датасету реальними звітами SAST, DAST, IAST, SCA, LOG, HTTP, CONFIG та CONTAINER
Часткова підтримка промислових форматів сканерів	додавання конекторів до OWASP ZAP, Semgrep, CodeQL, Trivy, Burp Suite та інших інструментів
Потреба в поглибленій перевірці графових і трансформерно-семантичних компонентів	окреме експериментальне тестування code-graph, GNN і локальних transformer-модулів
Необхідність людської валідації прикордонних випадків	додавання механізмів active learning і feedback-loop від аналітика
Обмежена перевірка в промисловому середовищі	інтеграція з CI/CD, GitHub Actions, GitLab CI, Jira, SIEM/SOAR-системами

Водночас результати виконаного дослідження засвідчили, що розроблений програмний прототип має потенціал подальшого розвитку. Перспективними напрямами є розширення аналізу структурних характеристик програмного коду, поглиблення механізмів нормалізації та семантичного збагачення даних, удосконалення інтеграції із зовнішніми джерелами знань і сервісами безпечної розробки, а також розширення експериментальної бази для перевірки роботи системи на більших і більш різнорідних наборах даних. Це дає підстави розглядати ІКС АВВЗ не лише як працездатний дослідний прототип, а й як основу для подальшого наукового та прикладного розвитку.

4.5 Висновки до четвертого розділу

У четвертому розділі виконано перехід від теоретичних моделей і алгоритмів ІКС АВВЗ до практичної реалізації, експериментальної перевірки та оцінювання прикладної цінності системи. Розроблений прототип підтвердив працездатність запропонованої архітектури та забезпечив повний цикл обробки результатів аналізу безпеки: приймання вхідних даних, нормалізацію, семантичне збагачення, гібридну класифікацію, ризик-орієнтоване ранжування, урахування невизначеності та формування аналітичних звітів.

Програмна реалізація ІКС АВВЗ виконана як модульний багатошаровий комплекс із підтримкою базових функціональних рівнів, REST API, локального сховища, журналювання, інтеграційних механізмів і звітних форматів. Фактична реалізація охоплює приймання результатів із каналів SAST, DAST, IAST, SCA, LOG, HTTP, CONFIG та CONTAINER, нормалізацію, кореляцію, класифікацію, розрахунок ризику, ранжування, збереження результатів, формування JSON-, HTML- і PDF-звітів та доступ до функцій системи через REST API.

Регресійна перевірка підтвердила працездатність програмного прототипу: 121 тест завершено успішно, помилок не зафіксовано. Це підтверджує, що прототип є не лише описаною в роботі концепцією, а працездатною дослідною програмною реалізацією запропонованих моделей і алгоритмів.

Експериментальна перевірка засвідчила ефективність фінального гібридного контуру класифікації: точність становила 0,8718, прецизійність – 0,8813, повнота – 0,8718, F1-міра – 0,8695, а середнє значення F1-міри за результатами п’ятиблокової крос-валідації – $0{,}9475 \pm 0{,}0369$. Порівняно з базовим сигнатурно-правиловим підходом досягнуто приблизно 18,1 % відносного приросту F1-міри. Також підтверджено ефективність нормалізації та дедуплікації: коефіцієнт дедуплікації становив 0,3684, коефіцієнт збереження інформативності – 0,8333.

Результати ризик-орієнтованого ранжування показали високу узгодженість із експертною логікою пріоритезації: якість верхніх позицій ранжованого списку становила 1,0000, а узгодженість з експертною оцінкою – 0,9167. Часові характеристики підтвердили придатність прототипу для напівавтоматизованих сценаріїв аналізу безпеки: загальний час проходження експериментального набору становив 177,01 мс, а середній час на один запис – 10,41 мс.

Окремо підтверджено працездатність епістемічно-алеаторної декомпозиції ризику. При коефіцієнті впливу невизначеності 0,3 середня ширина інтервалу ризику становила 1,283, а для нетипових випадків зафіксовано зростання нормованої ентропії з 0,760 до 0,877. Це підтверджує здатність системи враховувати невизначеність класифікаційного рішення та підтримувати обережнішу маршрутизацію випадків, що потребують ручної перевірки.

Отже, у четвертому розділі підтверджено працездатність, інженерну цілісність і прикладну доцільність ІКС АВВЗ. Отримані результати засвідчують, що розроблена система може використовуватися як основа для автоматизації аналізу безпеки вебзастосунків, інтеграції з процесами безпечної розробки програмного забезпечення та подальшого розвитку інтелектуальних засобів виявлення, класифікації й пріоритезації вразливостей.

ВИСНОВКИ

У кваліфікаційній роботі розв’язано актуальне науково-практичне завдання розроблення інтелектуальної комп’ютерної системи автоматичного виявлення вразливостей вебзастосунків та класифікації загроз, орієнтованої на інтеграцію результатів багатоканального аналізу безпеки, їх інтелектуальну нормалізацію, автоматичну класифікацію, адаптивне оцінювання ризику, урахування невизначеності та формування аналітичних звітів для використання у процесах безпечної розробки програмного забезпечення.

У процесі виконання роботи здійснено аналіз сучасного стану проблеми автоматизованого виявлення вразливостей вебзастосунків. Встановлено, що традиційні інструментальні підходи, зокрема SAST, DAST, IAST і SCA, мають істотну практичну цінність, однак характеризуються низкою обмежень: неоднорідністю форматів результатів, високою часткою хибних спрацювань, фрагментованістю висновків, складністю кореляції знахідок із різних джерел та недостатнім урахуванням контексту функціонування застосунку. Показано, що в умовах сучасної швидкої розробки програмного забезпечення ці обмеження знижують ефективність аналізу безпеки та ускладнюють процес прийняття рішень щодо усунення вразливостей.

На основі проведеного аналізу обґрунтовано доцільність використання методів штучного інтелекту, машинного навчання, структурного аналізу та семантичного збагачення у задачах виявлення вразливостей і класифікації кіберзагроз. Показано, що інтелектуальні підходи дозволяють не лише автоматизувати аналіз результатів сканування, а й підвищити якість їх інтерпретації, зменшити інформаційний шум, забезпечити семантичне зіставлення неоднорідних записів та підтримати ризик-орієнтовану пріоритезацію знайдених загроз.

У роботі розроблено модельну основу інтелектуальної комп’ютерної системи автоматичного виявлення вразливостей вебзастосунків та класифікації загроз. Сформовано архітектурну модель ІКС АВВЗ, модель інтелектуальної нормалізації результатів сканування, модель класифікації вразливостей і кіберзагроз та модель адаптивного оцінювання ризику. Побудовані моделі забезпечили формальний опис

основних етапів функціонування системи та створили основу для її алгоритмічної й програмної реалізації.

Удосконалено модель ризик-орієнтованого оцінювання вразливостей за рахунок використання восьмифакторної зваженої функції ризику з нелінійним мультиплікативним підсиленням. На відміну від стандартної технічної оцінки CVSS, запропонована модель враховує не лише технічну критичність, а й рівень упевненості класифікатора, контекстні параметри середовища, клас-специфічні ваги загроз, силу міжджерельного підтвердження, чутливість даних і компенсуючі механізми захисту.

Удосконалено алгоритм інтелектуальної нормалізації результатів сканування, у якому контекстні параметри ризику виводяться автоматично з текстових описів, метаданих і службових атрибутів без потреби ручної анотації. У результаті забезпечено уніфікацію різнорідних записів, зменшення дублювання, семантичне збагачення безпекових знахідок і підготовку даних до подальшої інтелектуальної класифікації.

Набули подальшого розвитку методи автоматичної класифікації безпекових знахідок за категоріями загроз за рахунок використання фінального гібридного контуру, що поєднує текстові, контекстні, джерельні, структурні, графові та трансформерно-семантичні ознаки. Такий підхід забезпечив кращу збалансованість між прецизійністю та повнотою порівняно з базовими підходами до класифікації.

У межах роботи розроблено алгоритми багатоканального збирання та обробки результатів сканування, інтелектуальної класифікації вразливостей і ризик-орієнтованого ранжування загроз. Побудовано функціональну архітектуру програмної системи та спроєктовано структуру її програмного й інформаційного забезпечення. Це дало змогу перейти від формального опису моделі до інженерного рівня реалізації ІКС АВВЗ як цілісного програмного комплексу.

Реалізовано працездатний дослідний програмний прототип системи, який включає підсистеми приймання результатів сканування, нормалізації, доступу до баз знань, гібридної класифікації, оцінювання ризику, зберігання даних, програмної взаємодії, звітності та журналювання. Фактична реалізація прототипу охоплює приймання результатів з каналів SAST, DAST, IAST, SCA, LOG, HTTP, CONFIG та CONTAINER, нормалізацію, кореляцію, класифікацію, ризик-орієнтоване оцінювання,

ранжування, збереження результатів, журналювання, формування JSON-, HTML- і PDF-звітів та доступ до функцій системи через REST API.

Регресійна перевірка підтвердила працездатність програмного прототипу: 121 тест завершено успішно, помилок не зафіксовано. Це дає підстави стверджувати, що розроблений прототип є не лише концептуальним описом запропонованої системи, а фактично реалізованим дослідним програмним засобом, який підтверджує практичну здійсненність розроблених моделей і алгоритмів.

У ході експериментальної перевірки підтверджено працездатність розробленого прототипу та ефективність запропонованих рішень. Порівняльний аналіз базових підходів показав, що фінальний гібридний контур забезпечує системне покращення відносно базового сигнатурно-правилового підходу та стабільне покращення балансу між прецизійністю й повнотою відносно поточного ансамблевого підходу машинного навчання. Експериментально підтверджено ефективність підсистеми нормалізації, зокрема зменшення інформаційної надмірності та збереження інформативності записів.

Окремим результатом роботи є практична апробація епістемічно-алеаторної декомпозиції ризику. У роботі використано нормовану ентропію Шеннона як явний показник епістемічної невизначеності, інтегрований у модель композитного ризику з інтервальним поданням результату. Це дозволило не лише оцінювати рівень небезпеки, а й формалізувати рівень довіри до цієї оцінки та визначати випадки, що потребують ручної верифікації.

Практичне значення одержаних результатів полягає у створенні програмного рішення, здатного автоматизувати ключові етапи аналізу безпеки вебзастосунків: інтеграцію результатів сканування, нормалізацію, класифікацію, оцінювання ризику, урахування невизначеності та пріоритезацію реагування. Система може бути використана у процесах безпечної розробки програмного забезпечення, у роботі фахівців з кібербезпеки, а також як навчальна й дослідницька платформа для задач аналізу вразливостей.

При цьому в роботі чітко зафіксовано межу між уже реалізованим функціоналом поточного прототипу та перспективами його подальшого розвитку.

Поточна версія системи підтверджує працездатність основного аналітичного контуру, модулів нормалізації, кореляції, класифікації, ризик-орієнтованого оцінювання, звітності, журналювання, збереження та REST API. Водночас розширення підтримки промислових форматів сканерів, поглиблення графового збагачення, ширше багатомовне узагальнення, повномасштабна інтеграція з SIEM/SOAR-середовищами та окремий розширений блок абляційного аналізу залишаються напрямами подальшого розвитку.

Отже, у кваліфікаційній роботі розв’язано актуальне науково-прикладне завдання розроблення інтелектуальної комп’ютерної системи автоматичного виявлення вразливостей вебзастосунків та класифікації загроз. У роботі виконано аналіз сучасного стану проблеми, розроблено моделі інтелектуальної нормалізації, гібридної класифікації та адаптивного оцінювання ризику, побудовано алгоритмічну й програмну основу системи, реалізовано працездатний дослідний програмний прототип та експериментально підтверджено його ефективність.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Drozd A., Mykuliak D. Intelligent computer system for automatic detection of web application vulnerabilities and threat classification. Measuring and Computing Devices in Technological Processes. 2026. No. 1. P. 368 – 376. DOI: https://doi.org/10.31891/2219-9365-2026-85-45.

2. Stallings W. Cryptography and Network Security: Principles and Practice. 8th ed. Hoboken (NJ, USA) : Pearson Education, 2022. 833 p.

3. Anderson R. Security Engineering: A Guide to Building Dependable Distributed Systems. 3rd ed. Wiley, 2020. 1223 p.

4. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. 4th ed. Hoboken : Pearson, 2021. 1168 p.

5. ISO/IEC 27001:2022. Information security management systems. Geneva : ISO, 2022.

6. ISO/IEC 27005:2022. Information security risk management. Geneva : ISO, 2022. (дата звернення: 15.01.2026).

7. NIST SP 800-53 Rev. 5. Security and Privacy Controls for Information Systems and Organizations. Gaithersburg : National Institute of Standards and Technology, 2020. URL: https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final

8. National Institute of Standards and Technology. NIST SP 800-218: Secure Software Development Framework (SSDF) Version 1.1. Gaithersburg, 2022. URL: https://csrc.nist.gov/publications/detail/sp/800-218/final (дата звернення: 01.02.2026).

9. OWASP Top 10:2025. The Ten Most Critical Web Application Security Risks. OWASP Foundation. URL: https://owasp.org/Top10/2025/en/

10. OWASP Foundation. OWASP API Security Top 10 – 2023. 2023. URL: https://owasp.org/API-Security/editions/2023/en/0x11-t10/

11. OWASP Foundation. OWASP Application Security Verification Standard 5.0.0. 2025. URL: https://owasp.org/www-project-application-security-verification-standard/

12. OWASP Foundation. OWASP Web Security Testing Guide. Version 4.2. 2020. URL: https://owasp.org/www-project-web-security-testing-guide/v42/

13. McDonald M. Grokking Web Application Security. Foreword by S. McClure. Shelter Island : Manning Publications, 2024. 336 p. ISBN 978-1-63343-826-2.

14. Rains T. Cybersecurity Threats, Malware Trends, and Strategies: Learn to Mitigate Exploits, Malware, Phishing, and Other Social Engineering Attacks. Birmingham : Packt Publishing, 2020. 428 p. eBook ISBN 978-1-80020-589-5.

15. Johansen G. Digital forensics and incident response. 3rd ed. Birmingham : Packt Publishing, 2023. 820 p.

16. Seacord R. C. Effective C: An Introduction to Professional C Programming. 2nd ed. San Francisco : No Starch Press, 2024. 312 p. ISBN 978-1-7185-0412-7.

17. The MITRE Corporation. Common Vulnerabilities and Exposures (CVE). URL: https://www.cve.org

18. Common Weakness Enumeration (CWE) / The MITRE Corporation. URL: https://cwe.mitre.org (дата звернення: 02.02.2026).

19. Common Attack Pattern Enumeration and Classification (CAPEC). URL: https://capec.mitre.org (дата звернення: 05.02.2026).

20. FIRST (Forum of Incident Response and Security Teams). Common Vulnerability Scoring System v4.0 Specification. URL: https://www.first.org/cvss/specification-document

21. National Vulnerability Database (NVD). URL: https://nvd.nist.gov

22. MITRE ATT&CK Framework. URL: https://attack.mitre.org

23. Chandramouli R., Butcher Z., Chetal A. Attribute-Based Access Control for Microservices-Based Applications Using a Service Mesh. NIST Special Publication 800-204B. Gaithersburg : National Institute of Standards and Technology, 2021. DOI: https://doi.org/10.6028/NIST.SP.800-204B. URL: https://csrc.nist.gov/pubs/sp/800/204/b/final

24. Chandramouli R. Implementation of DevSecOps for a Microservices-Based Application with Service Mesh. NIST Special Publication 800-204C. Gaithersburg : National Institute of Standards and Technology, 2022. DOI:

https://doi.org/10.6028/NIST.SP.800-204C. URL: https://csrc.nist.gov/pubs/sp/800/204/c/final

25. Chandramouli R., Kautz F., Torres-Arias S. Strategies for the Integration of Software Supply Chain Security in DevSecOps CI/CD Pipelines. NIST Special Publication 800-204D. Gaithersburg : National Institute of Standards and Technology, 2024. DOI: https://doi.org/10.6028/NIST.SP.800-204D. URL: https://csrc.nist.gov/pubs/sp/800/204/d/final

26. Chandramouli R., Butcher Z. Guidelines for API Protection for Cloud-Native Systems : March 2026 Update. NIST Special Publication 800-228-upd1. Gaithersburg : National Institute of Standards and Technology, 2026. DOI: https://doi.org/10.6028/NIST.SP.800-228-upd1. URL: https://csrc.nist.gov/pubs/sp/800/228/upd1/final

27. Boyens J., Smith A., Bartol N., Winkler K., Holbrook A., Fallon M. Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations. NIST Special Publication 800-161 Rev. 1, Update 1. Gaithersburg : National Institute of Standards and Technology, 2024. DOI: https://doi.org/10.6028/NIST.SP.800-161r1-upd1. URL: https://csrc.nist.gov/pubs/sp/800/161/r1/upd1/final

28. OWASP Foundation. CycloneDX Bill of Materials Specification (ECMA-424). 2024. URL: https://owasp.org/www-project-cyclonedx/

29. Ecma International. ECMA-424: CycloneDX Bill of Materials Specification. 1st ed. Geneva : Ecma International, 2024. URL: https://ecma-international.org/publications-and-standards/standards/ecma-424/

30. GitHub. CodeQL Documentation. URL: https://codeql.github.com/docs/

31. Semgrep. Semgrep Documentation. URL: https://semgrep.dev/docs/

32. Zed Attack Proxy Project. ZAP Getting Started Guide. URL: https://www.zaproxy.org/getting-started/

33. PortSwigger. Burp Suite Documentation. URL: https://portswigger.net/burp/documentation

34. OWASP Foundation. OWASP Dependency-Check. URL: https://owasp.org/www-project-dependency-check/

35. Aqua Security. Trivy Documentation: SBOM. URL: https://trivy.dev/docs/latest/supply-chain/sbom/

36. Anchore. Grype: Vulnerability Scanning Documentation. URL: https://oss.anchore.com/docs/guides/vulnerability/getting-started/

37. Anchore. Syft: SBOM Generation Documentation. URL: https://oss.anchore.com/docs/guides/sbom/getting-started/

38. Uddin M. N., Zhang Y., Hei X. Deep Learning Aided Software Vulnerability Detection: A Survey. 2025. URL: https://arxiv.org/abs/2503.04002

39. Wang S., Zhang H., Wang T. A comprehensive survey on deep learning for code vulnerability detection. *Journal of Systems and Software*. 2022. Vol. 182. Art. 111070. DOI: https://doi.org/10.1016/j.jss.2021.111070

40. Shaon M. S. H., Akter M. S. Modern Approaches to Software Vulnerability Detection: A Survey of Machine Learning, Deep Learning, and Large Language Models. *Electronics*. 2025. Vol. 14, No. 22. Art. 4449. DOI: https://doi.org/10.3390/electronics14224449

41. Chen Liang, Wei Q., Du J., Wang Y., Jiang Z. Survey of source code vulnerability analysis based on deep learning. *Computers & Security*. 2025. Vol. 146. Art. 104098. DOI: https://doi.org/10.1016/j.cose.2024.104098

42. Lin J., Mohaisen D. From Large to Mammoth: A Comparative Evaluation of Large Language Models in Vulnerability Detection. *Proceedings of the Network and Distributed System Security Symposium* (NDSS). 2025. URL: https://www.ndss-symposium.org/ndss-paper/from-large-to-mammoth-a-comparative-evaluation-of-large-language-models-in-vulnerability-detection/

43. Ben Yaala S., Bouallegue R. Vulnerability detection in large language models: addressing security concerns. *Journal of Cybersecurity and Privacy*. 2025. Vol. 5, No. 3. Art. 71. DOI: https://doi.org/10.3390/jcp5030071

44. Geluvaraj B., Satwik P. M., Kumar T. A. The future of cybersecurity: major role of artificial intelligence, machine learning, and deep learning in cyberspace. *Electronics*. 2025. Vol. 14, No. 11. Art. 2252. DOI: https://doi.org/10.3390/electronics14112252

45. Alani M. M. Big data in cybersecurity: a survey of applications and future trends. *Journal of Reliable Intelligent Environments*. 2021. Vol. 7, No. 2. P. 85 – 114. DOI: https://doi.org/10.1007/s40860-020-00120-3

46. Cao S., Sun X., Bo L., Wei Y., Li B. BGNN4VD: Constructing Bidirectional Graph Neural-Network for Vulnerability Detection. *Information and Software Technology*. 2021. Vol. 136. Art. 106576. DOI: https://doi.org/10.1016/j.infsof.2021.106576

47. Yao L. Hierarchical graph attention network for vulnerability detection. *Information and Software Technology*. 2023. Vol. 155. Art. 107091. DOI: https://doi.org/10.1016/j.infsof.2022.107091

48. Jiang Y., Zhang Y., Su X., Treude C., Wang T. StagedVulBERT: Multi-Granular Vulnerability Detection With a Novel Pre-Trained Code Model. *IEEE Transactions on Software Engineering*. 2024. Vol. 50, No. 12. P. 3454 – 3471. DOI: https://doi.org/10.1109/TSE.2024.3493245

49. Chakraborty S., Krishna R., Ding Y., Ray B. Deep Learning based Vulnerability Detection: Are We There Yet? *IEEE Transactions on Software Engineering*. 2022. Vol. 48, No. 9. P. 3280 – 3296. DOI: https://doi.org/10.1109/TSE.2021.3087402

50. Ding Y., Fu Y., Ibrahim O., Sitawarin C., Chen X., Alomair B., Wagner D., Ray B., Chen Y. Vulnerability Detection with Code Language Models: How Far Are We? 2025 *IEEE/ACM 47th International Conference on Software Engineering* (ICSE). 2025. DOI: https://doi.org/10.1109/ICSE55347.2025.00038

51. Alshomrani M., Albeshri A., Alturki B., Alallah F. S., Alsulami A. A. Survey of Transformer-Based Malicious Software Detection Systems. *Electronics*. 2024. Vol. 13, No. 23. Art. 4677. DOI: https://doi.org/10.3390/electronics13234677

52. Varga A. A Machine Learning-enhanced web-crawler for vulnerability detection: A binary classification approach : Master's thesis : AI and Machine Learning. Karlskrona : Blekinge Institute of Technology, 2025. 62 p. URL: https://www.diva-portal.org/smash/get/diva2%3A1962633/FULLTEXT01.pdf (дата звернення: 01.03.2026).

53. Oduleye B. E., Asuquo P., Bliss U. S. Web Vulnerabilities using Machine Learning for Prevention and Detection: A Critical Review. *International Journal on Emerging Technologies*. 2025. Vol. 16, No. 2. P. 120 – 139. DOI: https://doi.org/10.65041/IJET.2025.16.2.18

54. Busko N. A., Fedorchenko E. V., Kotenko I. V. Automatic Exploit Assessment Based on Deep Learning Methods. *Ontology of Designing*. 2024. Vol. 14, No. 3. P. 408 – 420. DOI: https://doi.org/10.18287/2223-9537-2024-14-3-408-420.

55. Vadodaria P., Dubey A. AI-Driven Automated Vulnerability Scanning for Real-Time Threat Detection and Mitigation. *International Journal of Innovative Research in Science, Engineering and Technology*. 2025. Vol. 14, No. 3. DOI: https://doi.org/10.15680/IJIRSET.2025.1403324

56. Boucena A. Leveraging Large Language Models For Automated Software Vulnerability Detection and Analysis : Master's thesis. Guelma : University of Guelma, 2025. 95 p. URL: https://dspace.univ-guelma.dz/jspui/handle/123456789/18272 (дата звернення: 01.03.2026).

57. Gyamfi N. K., Goranin N., Ceponis D., Čenys H. A. Automated System-Level Malware Detection Using Machine Learning: A Comprehensive Review. *Applied Sciences*. 2023. Vol. 13, No. 21. Art. 11908. DOI: https://doi.org/10.3390/app132111908. URL: https://www.mdpi.com/2076-3417/13/21/11908

58. Alzboon M. S., Al-Batah M. S., Alqaraleh M., Alzboon F., Alzboon L. Phishing Website Detection Using Machine Learning. Gamification and Augmented Reality. 2025. Vol. 3. Art. 81. DOI: https://doi.org/10.56294/gr202581

59. González-Granadillo G., González-Zarzosa S., Diaz R. Security information and event management (SIEM): analysis, trends, and usage in critical infrastructures. *Sensors*. 2021. Vol. 21, No. 14. Art. 4759. DOI: https://doi.org/10.3390/s21144759

60. Rosado D. G., Moreno J., Sánchez L. E., Santos-Olmo A., Serrano M. A., Fernández-Medina E. MARISMA-BiDa pattern: integrated risk analysis for big data. *Computers & Security*. 2021. Vol. 102. Art. 102155. DOI: https://doi.org/10.1016/j.cose.2020.102155

61. Khan S., Khan N., Wei T. Data analytic for cyber security: framework solutions and trends. *Eurasia Proceedings of Science, Technology, Engineering and Mathematics*. 2022. Vol. 18. P. 1 – 6. (дата звернення: 01.03.2026).

62. Zou Q., Zhang L., Singhal A., Sun X., Liu P. Attacks on machine learning systems: from security analysis to attack mitigation. Cham : Springer, 2022. 312 p. (дата звернення: 01.03.2026).

63. Tabassi E. Artificial Intelligence Risk Management Framework (AI RMF 1.0). NIST AI 100-1. Gaithersburg : National Institute of Standards and Technology, 2023. DOI: https://doi.org/10.6028/NIST.AI.100-1. URL: https://www.nist.gov/publications/artificial-intelligence-risk-management-framework-ai-rmf-10

64. Abdar M., Pourpanah F., Hussain S., Rezazadegan D., Liu L., Ghavamzadeh M., Fieguth P., Cao X., Khosravi A., Acharya U. R., Makarenkov V., Nahavandi S. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. Information Fusion. 2021. Vol. 76. P. 243 – 297. DOI: https://doi.org/10.1016/j.inffus.2021.05.008

65. Gawlikowski J., Tassi C. R. N., Ali M., Lee J., Humt M., Feng J., Kruspe A., Triebel R., Jung P., Roscher R., Shahzad M., Yang W., Bamler R., Zhu X. X. A survey of uncertainty in deep neural networks. Artificial Intelligence Review. 2023. Vol. 56. P. 1513 – 1589. DOI: https://doi.org/10.1007/s10462-023-10562-9

66. Kashtalian A., Lysenko S., Savenko B., Sochor T., Kysil T. Principle and Method of Deception Systems Synthesizing for Malware and Computer Attacks Detection. *Radioelectronic and Computer Systems*. 2023. No. 4(108). P. 112 – 151. DOI: https://doi.org/10.32620/reks.2023.4.10

67. Муляр І., Лєнков С., Гловюк В., Анікін В., Сотніков Є. Метод пошуку вразливостей вебзастосунків з використанням API ChatGPT. *Смарт технології: промислова та цивільна інженерія*. 2024. Т. 2, № 15. С. 46 – 55. DOI: https://doi.org/10.32347/st.2024.2.1203.

68. Kashtalian A., Lysenko S., Sachenko A., Savenko B., Savenko O., Nicheporuk A. Evaluation Criteria of Centralization Options in the Architecture of Multicomputer

Systems with Traps and Baits. *Radioelectronic and Computer Systems*. 2025. No. 1. P. 264 – 297. DOI: https://doi.org/10.32620/reks.2025.1.18.

69. Kashtalian A., Lysenko S., Nicheporuk A., Savenko O., Sochor T., Avsiyevych V. Multi-computer Malware Detection Systems with Metamorphic Functionality. *Radioelectronic and Computer Systems*. 2024. No. 1. P. 152 – 175. DOI: https://doi.org/10.32620/reks.2024.1.13.

70. Kashtalian A., Lysenko S., Kysil T., Sachenko A., Savenko O., Savenko B. Method and Rules for Determining the Next Centralization Option in Multicomputer System Architecture. *International Journal of Computing*. 2025. Vol. 24, No. 1. P. 35 – 51. DOI: https://doi.org/10.47839/ijc.24.1.3875.

71. Savenko O., Sierhieiev Y., Gaj P., Balej J. Using Artificial Intelligence in the Context of Buffer Overflow Vulnerabilities. *CEUR Workshop Proceedings*. 2025. Vol. 4013. P. 211 – 220. URL: https://ceur-ws.org/Vol-4013/

72. Savenko O., Sachenko A., Lysenko S., Markowsky G., Vasylkiv N. Botnet Detection Approach Based on the Distributed Systems. *International Journal of Computing*. 2020. Vol. 19, No. 2. P. 190 – 198. DOI: https://doi.org/10.47839/ijc.19.2.1761.

73. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Київ : ДП «УкрНДНЦ», 2016. 17 с. URL: https://nv-oneu.com.ua/downloads/dstu-8302-2015.pdf

74. scikit-learn developers. scikit-learn: Machine Learning in Python. URL: https://scikit-learn.org/

75. scikit-learn developers. Model Evaluation: Quantifying the Quality of Predictions. scikit-learn documentation. URL: https://scikit-learn.org/stable/modules/model_evaluation.html

76. scikit-learn developers. Cross-validation: Evaluating Estimator Performance. scikit-learn documentation. URL: https://scikit-learn.org/stable/modules/cross_validation.html (дата звернення: 01.03.2026).

77. scikit-learn developers. f1_score. scikit-learn documentation. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html (дата звернення: 01.03.2026).

78. scikit-learn developers. precision_recall_fscore_support. scikit-learn documentation. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html (дата звернення: 01.03.2026).

79. scikit-learn developers. train_test_split. scikit-learn documentation. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html (дата звернення: 01.03.2026).

80. scikit-learn developers. cross_validate. scikit-learn documentation. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_validate.html (дата звернення: 01.03.2026).

81. FastAPI. FastAPI Documentation. URL: https://fastapi.tiangolo.com/ (дата звернення: 01.03.2026).

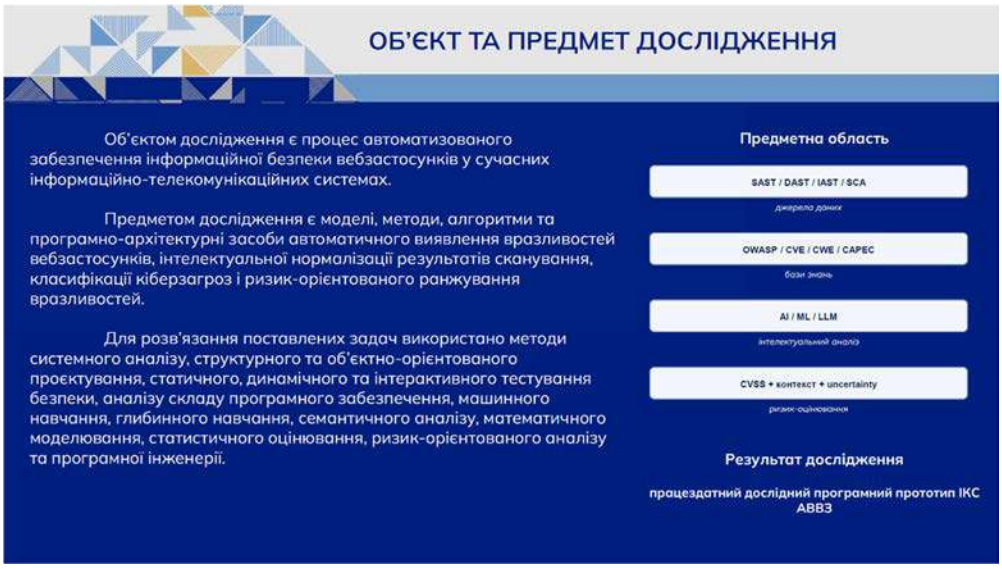
82. Pydantic. Pydantic Documentation. URL: https://pydantic.dev/docs/ (дата звернення: 01.03.2026).

83. Docker Inc. Docker Documentation. URL: https://docs.docker.com/ (дата звернення: 01.03.2026).

ДОДАТОК А

(обов’язковий)

Презентація до роботи



НАУКОВА НОВИЗНА ТА ПРАКТИЧНА ЦІННІСТЬ ОТРИМАНИХ РЕЗУЛЬТАТІВ

Наукова новизна отриманих результатів:

-удосконалено архітектурно-функціональну модель ІКС АВВЗ з єдиним аналітичним контуром;
-удосконалено модель ризик-орієнтованого оцінювання з урахуванням CVSS, контексту, впевненості класифікатора, міжджерельного підтвердження та епістемічної невизначеності;
-удосконалено алгоритм інтелектуальної нормалізації безпекових знахідок;
-набули подальшого розвитку методи автоматичної класифікації за рахунок гібридного класифікаційного контуру.

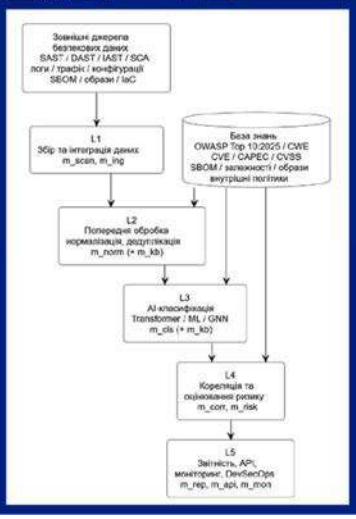
Практична значимість отриманих результатів:

Полягає у можливості використання розробленого прототипу для автоматизації аналізу безпеки вебзастосунків, інтеграції результатів різних інструментів сканування, зменшення кількості дубльованих і хибнопозитивних повідомлень, підвищення точності визначення критичних загроз, підтримки прийняття рішень щодо пріоритезації усунення вразливостей та інтеграції безпекового аналізу в життєвий цикл розроблення програмних систем.

Ключова ідея

нормалізація → класифікація → оцінювання ризику → врахування невизначеності → ранжований результат

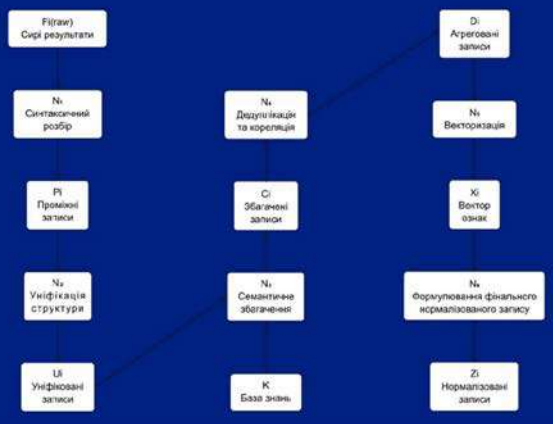
У загальному вигляді архітектурно-функціональна модель системи



L1 – збір та інтеграція результатів аналізу безпеки;
L2 – попередня обробка, нормалізація та дедуплікація;
L3 – AI-класифікація вразливостей і загроз;
L4 – кореляція та ризик-орієнтоване оцінювання;
L5 – звітність, API, моніторинг та інтеграція.

Модель ІКС АВВЗ: від джерел даних до звітності та DevSecOps

МОДЕЛЬ ІНТЕЛЕКТУАЛЬНОЇ НОРМАЛІЗАЦІЇ



Нормалізація результатів сканування

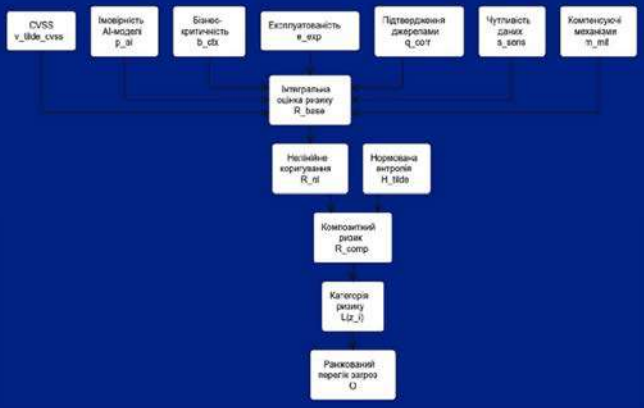
-Синтаксичний розбір сирих повідомлень;
-Уніфікація структури записів;
-Семантичне збагачення через базу знань;
-Дедуплікація та кореляція подібних знахідок;
-Векторизація та формування нормалізованого запису.

ГІБРИДНА МОДЕЛЬ КЛАСИФІКАЦІЇ

Гібридний контур класифікації вразливостей і кіберзагроз

-Текстовий підмодуль аналізує описи та повідомлення сканерів;
-Структурний підмодуль враховує локалізацію та тип артефакту;
-Контекстний підмодуль враховує середовище та роль компонента;
-Джерельний підмодуль враховує кількість і тип підтверджень;
-Агрегатор формує клас загрози та вектор імовірностей.

МОДЕЛЬ АДАПТИВНОГО ОЦІНЮВАННЯ РИЗИКУ



Ризик-орієнтоване оцінювання з урахуванням невизначеності

-CVSS використовується як базова технічна складова;
-Контекст, експлуатованість і чутливість даних уточнюють реальний ризик;
-Підтвердження різними джерелами підвищує достовірність оцінки;
-Нормована ентропія Шеннона відображає епістемічну невизначеність;
-Результат подається як категорія ризику та ранжований перелік загроз.

АЛГОРИТМИ ФУНКЦІОНУВАННЯ СИСТЕМИ

1. **Багатоканальний збір**
   SAST / DAST / IAST / SCA / LOG / HTTP / CONFIG / CONTAINER
2. **Нормалізація**
   очищення, уніфікація, семантичне збагачення, дедуплікація
3. **Класифікація**
   гібридний контур: текстові, структурні, контекстні та джерельні ознаки
4. **Ризик-оцінювання**
   CVSS, confidence, контекст, corroboration, mitigation, uncertainty
5. **Ранжування і звітність**
   пріоритети реагування, JSON / HTML / PDF, REST API

Алгоритмічне забезпечення переводить різнорідні технічні повідомлення у структурований, класифікований і ранжований перелік загроз.

ПРОГРАМНИЙ ПРОТОТИП ІКС АВВЗ

Фактична реалізація прототипу

-Приймання результатів з каналів SAST, DAST, IAST, SCA, LOG, HTTP, CONFIG та CONTAINER;
-Нормалізація, кореляція, класифікація, оцінювання ризику та ранжування;
-Зберігання результатів, журналювання та формування JSON-, HTML- і PDF-звітів;
-REST API для доступу до основних функцій системи;
-Регресійна перевірка: 121 тест виконано успішно.

Стек реалізації

Python 3.12	FastAPI
Pydantic v2	scikit-learn
SQLite	Docker / Compose
JSON	HTML / PDF
pytest	REST API

Статус: працездатний дослідний програмний прототип не позиціонується як завершений промисловий продукт

РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТУ: ЯКІСТЬ КЛАСИФІКАЦІЇ

Показники якості фінального гібридного контуру

Метрика	Позначення	Значення
Точність класифікації	Accuracy	0,8718
Прецизійність	Precision	0,8813
Повнота	Recall	0,8718
F1-міра	F1-score	0,8695
5-fold крос-валідація	CV F1	0,9475 ± 0,0369

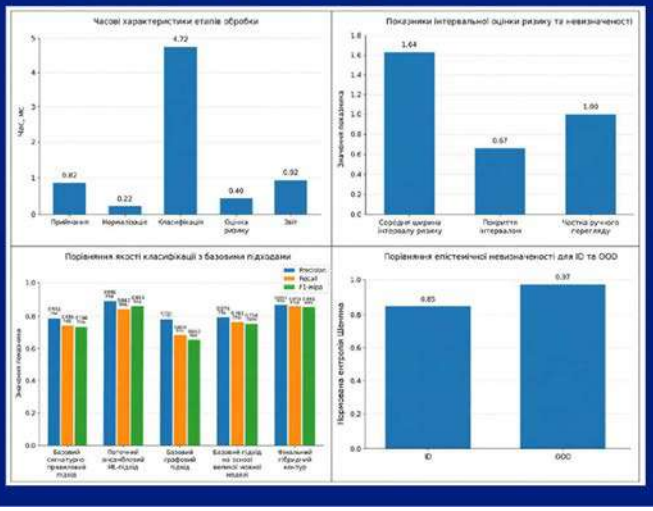
Порівняння з базовими підходами

Підхід	F1
Сигнатурно-правиловий	0,7360
Ансамблевий ML	0,8615
Графовий підхід	0,6663
LLM-підхід	0,7596
Фінальний гібридний контур	**0,8695**

Ключовий результат: F1-міра фінального гібридного контуру зросла з 0,7360 до 0,8695 порівняно з базовим сигнатурно-правиловим підходом, що відповідає приблизно 18,1 % відносного приросту.

Висновок: гібридизація ознак дає кращий баланс між прецизійністю та повнотою для задач кібербезпеки.

РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТУ: НОРМАЛІЗАЦІЯ, РИЗИК ТА НЕВИЗНАЧЕНІСТЬ



Додаткові кількісні результати

N_in = 19 записів	N_norm = 12 записів
N_review = 2 записи	K_dedup = 0,3684
K_info = 0,8333	Q_top-k = 1,0000
Q_expert = 0,9167	T_total = 177,01 мс
T_record = 10,41 мс	

Епістемічна невизначеність:

$\bar{H}$_ID = 0,760
$\bar{H}$_OOD = 0,877
приріст = +0,118

Система чутлива до нетипових випадків і переводить невизначені результати до ручної верифікації.

Фрагмент автоматично сформованого звіту ІКС АВВЗ за результатами аналізу вебзастосунку


ІКС АВВЗ - фінальний операційний звіт аналізу вразливостей

Операційний підсумок

12 | 8.73 | 12

5 | 2 | 2

Метод і контур

Пріоритети черги

Operational readiness


Табличний огляд знахідок

ВИСНОВКИ

У кваліфікаційній роботі розв'язано актуальне науково-практичне завдання розроблення інтелектуальної комп'ютерної системи автоматичного виявлення вразливостей вебзастосунків та класифікації загроз, орієнтованої на інтеграцію результатів багатоканального аналізу безпеки, їх інтелектуальну нормалізацію, автоматичну класифікацію, адаптивне оцінювання ризику, урахування невизначеності та формування аналітичних звітів для використання у процесах безпечної розробки програмного забезпечення:

- обґрунтовано актуальність теми, визначено об'єкт, предмет, мету, завдання, методи дослідження, наукову новизну та практичне значення роботи;
- проведено аналіз сучасних підходів до виявлення вразливостей вебзастосунків, класифікації кіберзагроз, стандартів безпеки та методів застосування штучного інтелекту в кібербезпеці;
- розроблено архітектуру, модель та метод інтелектуальної комп'ютерної системи автоматичного виявлення вразливостей вебзастосунків і класифікації загроз.;
- розроблено алгоритмічне забезпечення системи, зокрема алгоритми збору, нормалізації, класифікації, оцінювання ризику та ранжування загроз.;
- здійснено програмну реалізацію прототипу системи, проведено експериментальну верифікацію та оцінено ефективність запропонованих рішень.
- узагальнено основні результати роботи, підтверджено досягнення поставленої мети та визначено напрями подальшого розвитку системи

ПУБЛІКАЦІЇ

Наукова публікація автора

✓ Drozd A., Mykuliak D. Intelligent computer system for automatic detection of web application vulnerabilities and threat classification. Measuring and Computing Devices in Technological Processes. 2026. No. 1. P. 368–376.

Результати роботи також використано під час виконання науково-дослідної практики.

ДОДАТОК Б

(обов'язковий)

Наукова праця здобувача

Міжнародний науково-технічний журнал
«Вимірювальна та обчислювальна техніка в технологічних процесах»
ISSN 2219-9365

https://doi.org/10.31891/2219-9365-2026-85-45
УДК 004.52:004.75

ДРОЗД Андрій
Хмельницький національний університет
https://orcid.org/0009-0008-1049-1911
e-mail: andriydrozdit@gmail.com
МИКУЛЯК Дмитро
Хмельницький національний університет
e-mail: mykuliak.dmytro.ua@gmail.com
https://orcid.org/0009-0000-9437-8685

ІНТЕЛЕКТУАЛЬНА КОМП'ЮТЕРНА СИСТЕМА АВТОМАТИЧНОГО ВИЯВЛЕННЯ ВРАЗЛИВОСТЕЙ ВЕБ-ЗАСТОСУНКІВ ТА КЛАСИФІКАЦІЇ ЗАГРОЗ

У цій статті досліджується проблема автоматизованого виявлення та класифікації вразливостей веб-застосунків із використанням інтелектуальних комп'ютерних систем в умовах безперервного циклу розробки. Здійснено системний аналіз сучасних підходів до виявлення шкідливого програмного забезпечення та кібератак (SAST, DAST, SCA), зокрема архітектури систем децепції та методів машинного навчання, на основі фундаментальних досліджень вітчизняних та зарубіжних науковців.

Розглянуто актуальні загрози згідно з міжнародними стандартами OWASP Top 10:2021 та таксономією CWE. Особливу увагу приділено застосуванню великих мовних моделей (LLM) та архітектур на основі трансформерів (Transformers) для підвищення точності виявлення логічних вразливостей у вихідному коді, що є перспективним напрямком порівняно з традиційними статичними сканерами.

Авторами запропоновано концептуальну архітектуру інтелектуальної системи, яка базується на синергії графових нейронних мереж (GNN) та великих мовних моделей (LLM) для забезпечення семантичного аналізу та контекстної пріоритизації загроз із використанням розширених метрик CVSS. Обґрунтовано доцільність запровадження модуля нормалізації даних із гетерогенних сканерів у єдиний ознаковий простір. Експериментальні результати демонструють суттєве зниження рівня хибних спрацювань (False Positives) та підвищення метрики F1-score при використанні гібридної моделі. Робота становить практичний інтерес для фахівців з кібербезпеки, DevSecOps-інженерів та розробників засобів автоматизованого аудиту.

Ключові слова: веб-безпека, виявлення вразливостей, інтелектуальна система, машинне навчання, великі мовні моделі, LLM, трансформери, GNN, OWASP Top 10, CWE, DevSecOps.

DROZD Andriy, MYKULIAK Dmytro
Khmelnytskyi National University

INTELLIGENT COMPUTER SYSTEM FOR AUTOMATIC DETECTION OF WEB APPLICATION VULNERABILITIES AND THREAT CLASSIFICATION

The article investigates the problem of automated detection and classification of web application vulnerabilities using intelligent computer systems in a continuous development cycle. A systematic analysis of modern approaches to malware and cyberattack detection (SAST, DAST, SCA), including deception system architectures and machine learning methods, is carried out based on fundamental research by domestic and foreign scientists.

Current threats are reviewed according to the international standards OWASP Top 10:2021 and the CWE taxonomy. Special attention is paid to the application of Large Language Models (LLM) and Transformer-based architectures to improve the accuracy of detecting logical vulnerabilities in source code, representing a promising advancement over traditional static scanners.

The authors propose a conceptual architecture of an intelligent system based on the synergy of Graph Neural Networks (GNN) and LLMs to provide semantic analysis and context-aware threat prioritization using extended CVSS metrics. The feasibility of introducing a data normalization module from heterogeneous scanners into a unified feature space is substantiated. Experimental results demonstrate a significant reduction in the False Positive rate and an increase in the F1-score when using the hybrid model. The study is of practical interest to cybersecurity professionals, DevSecOps engineers, and developers of automated audit tools.

Keywords: web security, vulnerability detection, intelligent system, machine learning, Large Language Models, LLM, Transformers, GNN, OWASP Top 10, CWE, DevSecOps.

Стаття надійшла до редакції / Received 14.01.2026
Прийнята до друку / Accepted 13.02.2026
Опубліковано / Published 05.03.2026

ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

Сучасний етап розвитку інформаційних технологій невідривно пов'язаний із глобальною цифровізацією бізнес-процесів та перенесенням критичної інфраструктури в онлайн-середовище. Основою такої цифровізації є веб-застосунки та хмарні архітектури. Водночас цей етап характеризується стрімким ускладненням архітектури програмного забезпечення (ПЗ). Перехід від монолітних додатків до мікросервісних архітектур, інтеграція великої кількості сторонніх компонентів, використання контейнеризації та API-орієнтованих підходів суттєво розширили поверхню для можливих кібератак.

Міжнародний науково-технічний журнал
«Вимірювальна та обчислювальна техніка в технологічних процесах»
ISSN 2219-9365

Як наслідок, безпека веб-застосунків стала одним із найкритичніших викликів для індустрії. Попри наявність розвиненої міжнародної нормативної бази (стандарти серії ISO/IEC 27001, ISO/IEC 27002 щодо впровадження систем управління інформаційною безпекою, та ISO/IEC 27005 у контексті управління ризиками) і глобальних таксономій вразливостей (CVE – Common Vulnerabilities and Exposures, CWE – Common Weakness Enumeration, CAPEC – Common Attack Pattern Enumeration and Classification), практична реалізація процесів виявлення, класифікації та пріоритизації загроз у реальних конвеєрах безперервної розробки та розгортання (CI/CD і DevSecOps) залишається надзвичайно фрагментованою.

Згідно зі звітами міжнародних аналітичних агенцій та бази даних OWASP (Open Worldwide Application Security Project), понад 80% сучасних веб-застосунків містять щонайменше одну критичну вразливість на момент релізу. Традиційні підходи до забезпечення безпеки, які базуються на періодичних ручних аудитах або точковому застосуванні ізольованих інструментів перевірки, виявляються неефективними в умовах концепції Agile, де нові версії продукту можуть випускатися кілька разів на день.

Методи автоматизованого аналізу, такі як SAST (Static Application Security Testing), DAST (Dynamic Application Security Testing) та SCA (Software Composition Analysis), стали стандартом де-факто для індустрії. Однак їх розрізнене використання не здатне забезпечити цілісного бачення стану безпеки продукту. Аналіз поточної ситуації у сфері автоматизованого тестування безпеки дозволяє виділити наступні критичні проблеми, що потребують невідкладного наукового та інженерного вирішення:

Гетерогенність результатів аналізу та відсутність уніфікації. Однією з головних проблем є відсутність уніфікованої схеми інтеграції даних, отриманих від різних інструментів. Системи SAST аналізують вихідний код, виявляючи структурні недоліки (наприклад, жорстко закодовані паролі чи відсутність перевірки вводу). Системи DAST взаємодіють із працюючим застосунком через HTTP-запити, імітуючи поведінку зловмисника, тоді як SCA-сканери перевіряють сторонні бібліотеки (Open Source) на наявність відомих CVE. Кожен із цих інструментів генерує звіти у власному форматі (JSON, XML, SARIF), використовує власні алгоритми визначення критичності та власну термінологію. Це ускладнює агрегацію, кореляцію та дедуплікацію даних, створюючи хаос у процесі усунення знайдених проблем.

Проблема високого рівня хибних спрацювань (False Positives). Статичні аналізатори (SAST), маючи доступ до всього коду, часто використовують жорсткі регулярні вирази або прості лексичні аналізатори. Як наслідок, вони генерують величезну кількість сповіщень про потенційні вразливості, які в реальному контексті виконання не становлять загрози (наприклад, змінна, що не пройшла очищення (sanitization) у одному модулі, може бути очищена на рівні Middleware до потрапляння в базу даних). DAST-аналізатори, навпаки, можуть давати хибнонегативні результати (False Negatives) через неможливість доступу до прихованих ендпоінтів або через складні механізми автентифікації. Постійний потік хибних сповіщень призводить до ефекту «втоми від попереджень» (alert fatigue), через що фахівці з безпеки починають ігнорувати повідомлення сканерів, пропускаючи справді критичні загрози.

Існуюча парадигма визначення критичності здебільшого спирається на базовий скоринг CVSS (Common Vulnerability Scoring System). Проте базовий CVSS-скоринг оцінює лише технічну складність експлуатації вразливості та її теоретичний вплив на конфіденційність, цілісність і доступність системи (тріада CIA). Ця метрика є абсолютно статичною і не враховує бізнес-контекст застосунку, архітектурні особливості середовища розгортання, наявність компенсуючих механізмів (наприклад, Web Application Firewall) та цінність скомпрометованих активів. У результаті, вразливість із високим базовим балом у внутрішньому тестовому середовищі може розглядатися інструментами як пріоритетніша для виправлення, ніж вразливість із середнім балом на публічному платіжному шлюзі.

Незважаючи на існування детальних таксономій, на практиці спостерігається семантичний розрив. Відсутній автоматизований, формалізований зв'язок у ланцюгу формування загрози: від абстрактного типу помилки розробника (класифікація CWE) через конкретний екземпляр уразливості (CVE) до потенційного методу або патерну атаки (CAPEC). Аналізатори вказують на рядок коду, але не дають розуміння сценарію експлуатації, що критично важливо для побудови систем захисту (наприклад, систем децепції чи пасток).

Хоча профільні стандарти ISO/IEC 29147 (Розкриття інформації про вразливості) та ISO/IEC 30111 (Процеси обробки вразливостей) чітко регламентують організаційні та управлінські аспекти, вони не визначають конкретних математичних чи алгоритмічних механізмів для інтелектуальної агрегації результатів аналізу. У традиційних системах цей процес досі виконується мануально, що є неприйнятним для масштабних корпоративних середовищ.

Таким чином, постає гостра науково-практична потреба у створенні формалізованої інтегрованої моделі, що забезпечить: синхронізацію таксономічних знань (CWE, CAPEC) для глибокого семантичного розуміння природи загроз; адаптивну інтеграцію результатів мультиінструментального сканування із приведенням гетерогенних даних до єдиного ознакового простору: динамічний ризик-скоринг, що поєднує базові метрики CVSS із контекстними коефіцієнтами експлуатації згідно з рекомендаціями стандарту управління ризиками ISO/IEC 27005.

Розв'язання цієї багатовимірної проблеми класичними детермінованими алгоритмами є неможливим через високу варіативність вихідного коду та мінливість ландшафту кіберзагроз. Єдиним ефективним шляхом

Міжнародний науково-технічний журнал
«Вимірювальна та обчислювальна техніка в технологічних процесах»
ISSN 2219-9365

є розроблення інтелектуальної комп'ютерної системи, здатної до автоматичного вилучення ознак, абстрагування та машинного навчання на великих масивах даних про вразливості. Залучення методів глибинного навчання (Deep Learning), зокрема графових нейронних мереж (GNN) для структурного аналізу коду та великих мовних моделей (LLM) для контекстної обробки природної мови та синтаксису програмування, відкриває нові перспективи у створенні систем, здатних не лише виявляти, але й інтелектуально класифікувати та пріоритизувати кіберзагрози з точністю, що наближається до людської експертизи.

АНАЛІЗ ДОСЛІДЖЕНЬ ТА ПУБЛІКАЦІЙ

Питання автоматизації виявлення вразливостей, зниження відсотка хибних спрацювань та інтелектуалізації систем кіберзахисту перебуває в центрі уваги багатьох міжнародних та вітчизняних дослідницьких інституцій. Детальний аналіз сучасної фахової літератури (2020–2026 рр.) дозволяє класифікувати існуючі підходи за декількома ключовими векторами.

Методологічна база, стандартизація та онтологія загроз. Фундаментом для будь-якої системи автоматичного виявлення є стандарти та бази знань, що підтримуються міжнародними спільнотами, такими як MITRE Corporation та OWASP Foundation. У роботі [18] детально описується застосування таксономій CWE (Common Weakness Enumeration) та CAPEC, які дозволяють формалізувати типи помилок у коді та методи їх експлуатації зловмисниками. Проте, як справедливо зазначають автори методології OWASP [3] та дослідник [4], статичні списки (наприклад, щорічні звіти OWASP Top 10) хоча й дають загальне розуміння трендів, але потребують постійної адаптації до динамічного мікросервісного середовища сучасних веб-застосунків. Питання ризик-менеджменту та оцінювання критичності інцидентів традиційно базуються на серії стандартів ISO/IEC 2700x. Вони задають чудові концептуальні рамки для побудови СУІБ (Систем управління інформаційною безпекою), але, як зазначено у загальних положеннях [1], залишають суттєві прогалини в інженерній та алгоритмічній реалізації процесів пріоритизації на рівні програмного коду.

Інтелектуальні методи аналізу на основі глибинного навчання (Deep Learning). Сучасні дослідження (2023–2026 рр.) демонструють стрімкий зсув від класичних методів машинного навчання (таких як Random Forest чи SVM) у бік використання архітектур глибинного навчання. Зокрема, у роботі [7] та колектив авторів роботи [19] проводять масштабний аналіз застосування архітектур на основі трансформерів (Transformers, BERT) для виявлення шкідливого програмного забезпечення та автоматичного аудиту вихідного коду. Вони доводять, що здатність трансформерів утримувати контекст на великих ділянках коду дозволяє значно знизити рівень хибних спрацювань (FPR). Більше того, в роботі номер [8] та номер [15] наголошують на надзвичайній ефективності використання великих мовних моделей (Large Language Models, LLM) для семантичного аналізу вихідного коду. Автори стверджують, що LLM здатні виявляти складні логічні вразливості (Business Logic Vulnerabilities) та помилки порушення контролю доступу (Broken Access Control), які концептуально недоступні для класичних детермінованих SAST-інструментів, що базуються на абстрактних синтаксичних деревах (AST) чи регулярних виразах. Практичне застосування API LLM-моделей (наприклад, ChatGPT) для пошуку вразливостей також досліджено вітчизняними науковцями в роботі номер [5].

Гібридні архітектури, системи децепції та нормалізація даних. Окремий потужний напрям становлять роботи української наукової школи у сфері кібербезпеки, зокрема дослідження в роботах [2, 6, 17]. Ці фундаментальні праці зосереджені на створенні багаторівневих систем захисту з елементами кібердецепції (Cyber Deception) та пасток (Honeypots), що дозволяє виявляти цілеспрямовані атаки та ботнети в режимі реального часу. Особливої уваги в контексті нашого дослідження заслуговують розроблені ними критерії оцінювання варіантів централізації в архітектурі мультикомп'ютерних систем [6] та методи синтезу систем виявлення [2]. Математичний апарат нормалізації та агрегації результатів у таких гетерогенних розподілених системах закладає міцну теоретичну основу для розв'язання задачі уніфікації даних від різних засобів сканування (SAST/DAST/SCA), з якою стикаються сучасні DevSecOps-інженери.

Автоматизація та практична інтеграція у CI/CD. Низка прикладних досліджень присвячена архітектурним питанням впровадження систем безпеки у життєвий цикл розробки ПЗ (SDLC). Роботи [14], [16] та [9] фокусуються на розробленні автоматизованих сканерів та веб-краулерів, здатних працювати в режимі реального часу (Real-Time Threat Detection). Однак, попри досягнення в швидкості сканування, автори відзначають, що залишається невирішеною ключова проблема ефективної агрегації даних. Відсутність механізмів дедуплікації часто призводить до конфліктів у звітності та багаторазового дублювання однієї і тієї ж інформації, що нівелює переваги автоматизації.

Виділення невирішеної частини загальної проблеми Незважаючи на значні успіхи наукової спільноти в окремих галузях (вдосконалення трансформерів, розвиток систем децепції, розробка нових метрик), аналіз публікацій дозволяє стверджувати, що більшість існуючих систем фокусуються фрагментарно: або на покращенні процесу виявлення (Detection), або виключно на процесі категоризації (Classification), не забезпечуючи замкненого циклу інтелектуальної обробки інцидентів безпеки.

На сьогоднішній день відсутні комплексні формалізовані моделі, які б одночасно:

Міжнародний науково-технічний журнал
«Вимірювальна та обчислювальна техніка в технологічних процесах»
ISSN 2219-9365

Нормалізували гетерогенні дані від SAST, DAST та SCA до єдиного ознакового простору.
Використовували синергію графових нейронних мереж (GNN) для математично точного аналізу структури потоків даних застосунку та великих мовних моделей (LLM) для глибокого контекстуального та семантичного аналізу описів уразливостей.

Забезпечували на основі цього динамічний контекстно-орієнтований ризик-скоринг (Risk-Scoring), що доповнює стандартні оцінки CVSS.

Розроблення такої інтелектуальної комп'ютерної системи становить актуальну науково-прикладну задачу, якій і присвячено дане дослідження.

ФОРМУЛЮВАННЯ ЦІЛЕЙ СТАТТІ

Інтелектуальна система автоматичного виявлення вразливостей веб-застосунків (ІКС-АВВЗ) ґрунтується на концепції багаторівневої обробки даних. Вона охоплює збір гетерогенних результатів сканування, їхню нормалізацію, гібридну нейромережеву класифікацію та контекстуальний ризик-скоринг. Для формального обґрунтування запропоноване рішення описано математичною моделлю.

Формалізація предметної області та цільова функція

Розглянемо множину веб-застосунків $W = \{w_1, w_2, \dots, w_N\}$. Кожен застосунок $w_i \in W$ складається з множини структурних або логічних артефактів:

$$A(w_i) = \{a_{i1}, a_{i2}, \dots, a_{iK}\}, \tag{1}$$

де a_{ik} може бути файлом вихідного коду, функцією, модулем, графом залежностей або HTTP-запитом.

Множина відомих класів загроз позначається $C = \{c_1, c_2, \dots, c_M\}$, де кожен c_j відповідає категорії з таксономій CWE (наприклад, CWE-79 для XSS або CWE-89 для SQL-ін'єкцій).

Кожному артефакту a_{ik} ставиться у відповідність вектор ознак $x \in \mathbb{R}^d$ через оператор Φ:

$$x = \Phi(a_{ik}), x \in \mathrm{R}^d, \tag{2}$$

де $\Phi(\cdot)$ формує ембеддинги на основі AST, графів залежностей або поведінкових характеристик.

Класифікація виконується відображенням

$$f_\theta: \mathbb{R}^d \to \Delta^{M-1}, \tag{3}$$

де θ- параметри моделі, а Δ^{M-1}- симплекс, що задає розподіл імовірностей належності артефакту до кожного класу.

Навчання моделі зводиться до мінімізації очікуваної втрати:

$$L(f_\theta) = \mathbb{E}_{(x,y)\sim P}[l(f_\theta(x), y)] \to \min_\theta, \tag{4}$$

де P- невідомий розподіл даних і міток, а $l(\cdot,\cdot)$- функція втрат. На практиці мінімізується емпіричний ризик:

$$\hat{L}_n(f_\theta) = \frac{1}{n}\sum_{i=1}^{n} l(f_\theta(x_i), y_i), \tag{5}$$

як функцію втрат використовують крос-ентропію:

$$l(f_\theta(x), y) = -\sum_{j=1}^{M} y^{(j)} \log(f_\theta(x)^{(j)}), \tag{6}$$

де $y^{(j)}$- індикатор належності до класу j.

Інтеграція SAST/DAST/SCA та нормалізація ознак

Основним недоліком традиційних конвеєрів безпеки є ізольованість різних аналізаторів. Позначимо множини сирих сповіщень від статичного, динамічного та компоновочного аналізу як $S_{\text{SAST}}, S_{\text{DAST}}, S_{\text{SCA}}$. Об'єднаний простір інцидентів:

$$S_{\text{total}} = S_{\text{SAST}} \cup S_{\text{DAST}} \cup S_{\text{SCA}}. \tag{6}$$

Для перетворення різнорідних форматів (JSON, XML, SARIF) у спільний векторний простір вводиться оператор нормалізації:

$$N: S_{\text{total}} \to \mathrm{R}^d, \tag{7}$$

Міжнародний науково-технічний журнал
«Вимірювальна та обчислювальна техніка в технологічних процесах»
ISSN 2219-9365

який проводить дедуплікацію та об'єднує дані з різних джерел. Наприклад, якщо статичний аналізатор знаходить підозрілий рядок, а динамічний підтверджує його експлуатацію, оператор Nзливає ці сповіщення в один вектор для подальшої обробки.

Гібридна модель (GNN + LLM)

Для зменшення хибних спрацьовувань запропоновано ансамблеву модель, що поєднує:

Графовий компонент (GNN). Код моделюється як орієнтований граф $G = (V, E)$. Вузли Vвідповідають операторам, змінним та API-ендпоінтам, а ребра E- залежностям потоків даних і керування. Оновлення embedding-вузлів на k-му шарі:

$$h_v^{(k)} = \sigma(W^{(k)} \cdot \text{AGGREGATE}\{h_u^{(k-1)} : u \in \mathcal{N}(v)\}), \tag{8}$$

де $W^{(k)}$- матриця ваг, σ- нелінійність, $\mathcal{N}(v)$- сусіди.

Мовний компонент (LLM). У той же час вектор xподається на донавчену модель Transformer, яка аналізує текстову семантику коду, коментарів та HTTP-відповідей, даючи розподіл $P_{\text{LLM}}(y \mid x)$.

Фінальний розподіл комбінується:

$$P_{\text{final}}(y \mid x) = \alpha \cdot P_{\text{GNN}}(y \mid x) + (1 - \alpha)\, P_{\text{LLM}}(y \mid x), \tag{9}$$

де $\alpha \in [0,1]$підбирається на валідаційній вибірці.

Межі узагальнюючої похибки оцінюють через нерівність Хефдінга: для випадкової вибірки розміром nз імовірністю не менше $1 - \delta$:

$$| L(f_\theta) - \hat{L}_n(f_\theta) | \leq \sqrt{\frac{\ln(2/\delta)}{2n}}. \tag{10}$$

Отже, за достатньо великого nгібридна модель узгоджена й не перенавчається. Для гладких функцій втрат із константою Lградієнтний спуск збігається до локального мінімуму за умов кроку $\eta \leq 1/L$.

Контекстуальний ризик-скоринг

Запропоновано динамічний індекс ризику:

$$\text{Risk}_{\text{total}} = \text{BaseScore}_{\text{CVSS}} \cdot K_{\text{asset}} \cdot K_{\text{exposure}}, \tag{11}$$

де:

$\text{BaseScore}_{\text{CVSS}}$- базова оцінка з CVSS (0–10)

K_{asset}- коефіцієнт бізнес-важливості активу (максимальний для критичних модулів)

K_{exposure}- коефіцієнт експозиції (близький до 1 для публічних API, менший для внутрішніх)

Цей підхід дозволяє не лише знаходити вразливості, а й генерувати пріоритетний план усунення, орієнтований на бізнес-критичні загрози. Ми сформували формальну математичну модель інтелектуальної системи автоматичного виявлення вразливостей вебзастосунків (ІКС АВВЗ), що базується на концепції багаторівневої обробки даних та інтеграції різнорідних джерел безпекової інформації. Запропонований підхід забезпечує перехід від фрагментарного аналізу до цілісної моделі прийняття рішень, яка поєднує структурний, поведінковий і семантичний контексти.

Формалізація предметної області через множину вебзастосунків, їхніх артефактів та простір класів загроз дозволила звести задачу виявлення вразливостей до задачі багатокласової класифікації у векторному просторі ознак. Введення операторів відображення Ф та N забезпечує узгоджене представлення гетерогенних даних (SAST, DAST, SCA) у спільному ознаковому просторі, що усуває проблему ізольованості традиційних аналізаторів та зменшує кількість дублікатів і суперечливих сповіщень.

Гібридна архітектура (GNN + LLM) дозволяє одночасно враховувати топологічну структуру програмного коду (граф залежностей потоків даних і керування) та його семантичний зміст. Ансамблеве поєднання ймовірнісних розподілів підвищує точність класифікації та знижує рівень хибнопозитивних і хибнонегативних спрацьовувань. Теоретичне обґрунтування через мінімізацію емпіричного ризику та оцінку меж узагальнюючої похибки (нерівність Хефдінга) підтверджує статистичну узгодженість моделі за умови достатнього обсягу навчальної вибірки.

Запропонований механізм контекстуального ризик-скорингу розширює класичний підхід CVSS за рахунок врахування бізнес-критичності активів і рівня їх експозиції. Це забезпечує перехід від суто технічної оцінки вразливості до управлінсько-орієнтованої моделі пріоритезації, що підвищує ефективність планування заходів з усунення загроз.

Таким чином, розроблена математична модель ІКС АВВЗ забезпечує: формальну узгодженість задачі виявлення вразливостей, інтеграцію різнорідних безпекових джерел у єдиний аналітичний простір; підвищення точності класифікації за рахунок гібридної нейромережевої архітектури; адаптивну бізнес-орієнтовану пріоритезацію ризиків.

Міжнародний науково-технічний журнал
«Вимірювальна та обчислювальна техніка в технологічних процесах»
ISSN 2219-9365

Отримані результати створюють теоретичну основу для подальшої реалізації системи в межах архітектури та експериментальної валідації її ефективності на реальних наборах даних.

ЕКСПЕРИМЕНТ

Мета експериментальної частини - емпірично перевірити ефективність інтелектуальної комп'ютерної системи автоматичного виявлення вразливостей веб-застосунків (ІКС-АВВЗ) та довести статистично значущу перевагу гібридної моделі (GNN + LLM) над базовими підходами. Порівняльне тестування проводилось одночасно для чотирьох архітектур:

1.Базова модель - Random Forest (RF), класичний ансамблевий метод, типовий для традиційних сканерів.

2.Структурна модель - графова нейронна мережа (GNN), навчена лише на графах потоків даних і керування (DFG/CFG).

3.Семантична модель - велика мовна модель (LLM) на базі Transformer, донавчена на текстових фрагментах коду та звітах про вразливості.

4.Гібридна архітектура - поєднання GNN та LLM із зваженим об'єднанням передбачень.

Ефективність оцінювали за точністю, повнотою, F1-оцінкою, площею під ROC-кривою (ROC-AUC), часткою хибних спрацювань (FPR) та обчислювальною складністю інференсу.

Формування датасету

Для валідації було сформовано комбінований датасет обсягом $N = 14{,}042$ події безпеки з двох груп:

1.Реальні дані (62,3 %) - 8 742 підтверджені вразливості з 12 open-source веб-проєктів (Python, Java, Node.js), розмічені за таксономією CWE та базами NVD, OWASP Benchmark і ExploitDB.

2.Синтетичні дані (37,7 %) - 5 300 згенерованих фрагментів коду з мутаціями (SQL Injection, XSS, RCE), створених за допомогою фаззингу та синтаксичних мутацій.

3.Вибірку випадково поділено на тренувальну (70 %), валідаційну (15 %) та тестову (15 %) частини, щоб уникнути перенавчання і забезпечити коректну оцінку.

Метрики та статистичні гіпотези

Якість оцінювали за матрицею помилок, обчислюючи:

Точність: $P = \frac{TP}{TP+FP}$

Повнота: $R = \frac{TP}{TP+FN}$

F1-оцінка: $F_1 = 2\frac{P \cdot R}{P+R}$

Частка хибних спрацювань (FPR): $\frac{FP}{FP+TN}$

Інтегральним показником слугувала площа під ROC-кривою:

$$\text{AUC} = \int_0^1 TPR(FPR)\, d(FPR). \tag{12}$$

Формульовано гіпотези:

H_0: середня точність гібридної та найкращої ізольованої моделі однакові ($\mu_{\text{Hybrid}} = \mu_{\text{LLM}}$).

H_1: гібридна модель має вищу точність ($\mu_{\text{Hybrid}} > \mu_{\text{LLM}}$).

Перевірка виконувалась однобічним t-критерієм Стьюдента на рівні $\alpha = 0{,}05$.

4. Результати тестування

Таблиця 1

Порівняльний аналіз ефективності

Архітектура	Precision	Recall	F1-score	ROC-AUC
Random Forest	0.81	0.74	0.77	0.84
GNN	0.86	0.82	0.84	0.89
LLM	0.88	0.85	0.86	0.91
GNN + LLM	0.92	0.89	0.90	0.95

Таблиця 2

Аналіз хибних спрацювань

Архітектура	FPR	Відносне зниження
Random Forest	0,18 (18 %)	–
GNN	0,13 (13 %)	−27,7 %
LLM	0,11 (11 %)	−38,8 %
GNN + LLM	0,08 (8 %)	−55,5 %

Гібридна архітектура показала найкращі показники: F1-оцінка підвищилась на 13 % порівняно з Random Forest, ROC-AUC - на 11 %. Статистика $t = 4{,}12$, $p < 0{,}001$ дозволяє відхилити H_0 на користь H_1.

Міжнародний науково-технічний журнал
«Вимірювальна та обчислювальна техніка в технологічних процесах»
ISSN 2219-9365

Зниження FPR до 8 % на 55,5 % менше базового рівня, що істотно зменшує "втому від попереджень".

Оцінка узагальнюючої похибки та складність
Для $n = 14\,042$та довіри 95 %:

$$\epsilon \leq \sqrt{\frac{\ln(2/0{,}05)}{2 \cdot 14\,042}} \approx 0{,}011, \tag{13}$$

що відповідає максимальному відхиленню не більше 1,1 %. Складність інференсу GNN - $\mathcal{O}(|V|+|E|)$, LLM - $\mathcal{O}(L^2)$. Завдяки кешуванню і батчингу середній час аналізу одного комміту становив ~1,2 с.

Оцінювання економічної ефективності впровадження ІКС-АВВЗ

Наша мета - зменшити прямі й непрямі витрати підприємства (ліквідація інцидентів, ручний аудит, простій сервісів, штрафи). Оцінка проєкту базується на метриках TCO, ROI, NPV та строку окупності.

1. Модель витрат (TCO)

Сукупна вартість першого року:

$$\text{TCO} = C_{\text{dev}} + C_{\text{infra}} + C_{\text{support}} + C_{\text{staff}}, \tag{14}$$

де компоненти представляють розробку, інфраструктуру, підтримку та персонал. Для типової IT-компанії (2026 р.):

Таблиця 3

Оцінювання економічної ефективності

Стаття	Опис	Вартість (USD/рік)
C_{dev}	Адаптація GNN + LLM, інтеграція	45 000
C_{infra}	Хмарні потужності	18 000
$C_{support}$	Оновлення баз, моніторинг	12 000
C_{staff}	Робота фахівців	40 000
Всього (TCO)	-	115 000

2. Модель економічного ефекту й ROI

Економія:

$$\text{Savings} = (N_{\text{before}} - N_{\text{after}}) \times C_{\text{inc}}, \tag{15}$$

де $C_{\text{inc}} = 120{,}000$ USD, зниження інцидентів із 3 до 1 на рік дає 240 000 USD економії. Рентабельність:

$$\text{ROI} = \frac{\text{Savings} - \text{TCO}}{\text{TCO}} \times 100\% \approx 108{,}7\%. \tag{16}$$

3. Чиста приведена вартість (NPV) й строк окупності

Для періоду $T = 3$роки, ставки дисконту $r = 0{,}15$:

$$\text{NPV} = \sum_{t=1}^{T} \frac{CF_t}{(1+r)^t} - I_0 \approx 432{,}968 \text{ USD}, \tag{17}$$

$NPV > 0$- проєкт вигідний. Строк окупності:

$$\text{PP} = \frac{\text{TCO}}{\text{Savings}/12} \approx 5{,}75 \text{ місяців}.$$

4. Сценарний аналіз і порівняння з ручним аудитом

Таблиця 4

Сценарний аналіз і порівняння з ручним аудитом

Сценарій	Інциденти/рік	Економія (USD)	Очікуваний ROI
Оптимістичний	4 → 1	360 000	213 %
Базовий	3 → 1	240 000	108 %
Песимістичний	2 → 1	120 000	4 %

Навіть у песимістичному сценарії проєкт не збитковий. Для порівняння: ручний аудит коштує ~150 000 USD/рік, FPR 25-35 %, перевірка займає тижні, тоді як ІКС-АВВЗ зменшує витрати до 115 000 USD/рік, FPR - 8 %, час перевірки - до кількох годин.

Міжнародний науково-технічний журнал
«Вимірювальна та обчислювальна техніка в технологічних процесах»
ISSN 2219-9365

Експериментально доведено статистично значущу перевагу гібридної архітектури GNN + LLM над класичними методами. Підвищення F1-score до 0,90, ROC-AUC до 0,95 й зниження FPR до 0,08 пояснюються синергією структурного аналізу (AST, CFG, PDG) та контекстуального розуміння логіки. Гібридна модель формально описується як ансамбль $\hat{y} = \alpha y_{\text{GNN}} + (1-\alpha) y_{\text{LLM}}$, що зменшує дисперсію оцінки й забезпечує узагальнюючу похибку < 1,1 %.

ВИСНОВКИ З ДАНОГО ДОСЛІДЖЕННЯ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ РОЗВІДОК У ДАНОМУ НАПРЯМІ

У роботі вирішено практичну задачу підвищення ефективності автоматизованого виявлення вразливостей веб-застосунків та класифікації загроз шляхом створення інтегрованої інтелектуальної системи.

Системний аналіз методів і таксономій. Проаналізовано підходи SAST, DAST, SCA та таксономії CWE, CVE, CAPEC. Показано, що фрагментованість класичних інструментів і статичність метрик CVSS спричиняють перевантаження DevSecOps-команд хибними сповіщеннями.

Запропоновано Гібридна модель класифікації, що поєднує графовий аналіз потоків даних (GNN) із семантичною інтерпретацією коду на базі великих мовних моделей (LLM). Такий підхід реалізовано вперше для задач безпеки веб-застосунків.

Експерименти показали приріст F1-оцінки до 0,90 (на 13 %) та збільшення ROC-AUC до 0,95. Частка хибних спрацювань зменшилася з 18 % до 8 %, що становить 55,5 % зниження порівняно з Random Forest.

Розроблено модель оцінювання, динамічний ризик-скоринг , який доповнює CVSS гнучкими коефіцієнтами експозиції та бізнес-важливості, дозволяючи формувати пріоритети виправлення вразливостей.

Комплексний аналіз економічної ефективності підтвердив високий економічний ефект: ROI понад 108 %, чиста приведена вартість перевищує 430 тис. USD, а строк окупності - менше 6 місяців.

Новизна полягає у формалізації оцінки узагальнюючої похибки для задач виявлення вразливостей та вдосконаленні методів агрегування даних. Практичний ефект полягає у готовності алгоритмів до інтеграції в CI/CD-конвеєри та корпоративні системи управління ризиками, що здатне зменшити фінансові втрати від кіберінцидентів.

Література

1. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. [Уведено вперше ; чинний від 2016-07-01]. Київ : ДП «УкрНДНЦ», 2016. 17 с. URL: https://nv-oneu.com.ua/downloads/dstu-8302-2015.pdf

2. Кашталян А., Лисенко С., Савенко Б., Сохор Т., Кисіль Т. Принцип та метод синтезу систем децепції для виявлення шкідливого програмного забезпечення та комп'ютерних атак. *Радіоелектронні і комп'ютерні системи*. 2023. № 4. С. 112–151. https://doi.org/10.32620/reks.2023.4.10

3. OWASP Top 10:2021. The Ten Most Critical Web Application Security Risks. *OWASP Foundation*. URL: https://owasp.org/www-project-top-ten/

4. Білодід Д. В. Методика запобігання вразливостей front-end частини web-застосунків : кваліфікаційна робота магістра : 121 Інженерія програмного забезпечення. Київ : ДУІКТ, 2025. 84 с. URL: https://surl.li/vzrnlw

5. Метод пошуку вразливостей вебзастосунків з використанням API ChatGPT / В. В. Поліщук та ін. *Сучасні інформаційні технологіі в будівництві та архітектурі*. 2024. № 318240. URL: https://smarttech.knuba.edu.ua/article/view/318240

6. Kashtalian A., Lysenko S., Sachenko A. Evaluation criteria of centralization options in the architecture of multicomputer systems with traps and baits. *Radioelectronic and Computer Systems*. 2025. No. 1. P. 264–297.

7. Alshomrani M. Survey of Transformer-Based Malicious Software Detection Systems. *Electronics*. 2024. Vol. 13, No. 4677. DOI: https://doi.org/10.3390/electronics13234677

8. Modern Approaches to Software Vulnerability Detection: A Survey of Machine Learning, Deep Learning, and Large Language Models / T. Zeng et al. *Electronics*. 2025. Vol. 14, No. 22. URL: https://www.mdpi.com/2079-9292/14/22/4449

9. Varga A. A Machine Learning-enhanced web-crawler for vulnerability detection: A binary classification approach : Master's thesis : AI and Machine Learning. Karlskrona : Blekinge Institute of Technology, 2025. 62 p. URL: https://www.diva-portal.org/smash/get/diva2%3A1962633/FULLTEXT01.pdf

10. Web Vulnerabilities using Machine Learning for Prevention and Detection: A Critical Review *International Journal on Emerging Technologies*, Vol. 16, Issue 2, pp. 120–139 (2025) URL: https://www.researchtrend.net/ijet/pdf/Web-Vulnerabilities-using-Machine-Learning-for-Prevention-and-Detection-A-Critical-Review-Oduleye-BE-18.pdf

11. Artificial Intelligence-Driven Supervised Classification Algorithm for Website Vulnerability Detection Using MITRE NVD CVE Scores / J. Doe et al. *Preprints.org*. 2026. URL: https://www.preprints.org/manuscript/202602.0475

12. Busko V. Automatic exploit assessment based on deep learning methods. *Ontology of Designing*. 2024. Vol. 14, No. 2. P. 156–170. URL: https://doi.org/10.18287/2223-9537-2024-14-3-408-420

13. Automated Vulnerability Assessment Using Machine Learning / R. Sharma et al. *ResearchGate*. 2024. URL: https://www.researchgate.net/publication/382918034

14. AI-Driven Automated Vulnerability Scanning for Real-Time Threat Detection and Mitigation / K. Modi et al. *International Journal of Innovative Research in Science, Engineering and Technology*. 2025. Vol. 14, No. 3. https://doi.org/10.15680/IJIRSET.2025.1403324

15. Boucena A. Leveraging Large Language Models For Automated Software Vulnerability Detection and Analysis : Master's thesis. Guelma : University of Guelma, 2025. 95 p. URL: https://dspace.univ-guelma.dz/jspui/bitstream/123456789/18272/1/F5_8_BOUCENA_AMINA_1752072283.pdf

16. Automatic Source Code Vulnerability Detection, Classification, and Prioritization Using Deep Learning / S. Jalowski URL: https://assets-eu.researchsquare.com/files/rs-7423339/v1_covered_14df0d01-894d-4139-86d7-37ff5eae83a7.pdf

17. Савенко О. С., Лисенко С. М., Нічепорук А. О. Виявлення бот-мереж на основі аналізу поведінки мережевих об'єктів у розподілених системах. *Комп'ютерні системи та мережі*. 2020. № 19. С. 190–198. URL: https://computingonline.net/computing/article/view/1761

18. Common Weakness Enumeration (CWE). *MITRE Corporation*. 2025. URL: https://cwe.mitre.org/

19. Gyamfi N. K., Goranin N. Automated System-Level Malware Detection Using Machine Learning: A Comprehensive Review. *Applied Sciences*. 2023. Vol. 13, No. 11908. URL: https://www.mdpi.com/2076-3417/13/21/11908

20. Phishing Website Detection Using Machine Learning / A. Kumar et al. *Dialnet*. 2025. URL: https://dialnet.unirioja.es/descarga/articulo/9930098.pdf

References

1. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. [Уведено вперше ; чинний від 2016-07-01]. Київ : ДП «УкрНДНЦ», 2016. 17 с. URL: https://nv-oneu.com.ua/downloads/dstu-8302-2015.pdf

2. Кашталян А., Лисенко С., Савенко Б., Сохор Т., Кисіль Т. Принцип та метод синтезу систем децепції для виявлення шкідливого програмного забезпечення та комп'ютерних атак. *Радіоелектронні і комп'ютерні системи*. 2023. № 4. С. 112–151. DOI: https://doi.org/10.32620/reks.2023.4.10

3. OWASP Top 10:2021. The Ten Most Critical Web Application Security Risks. *OWASP Foundation*. URL: https://owasp.org/www-project-top-ten/

4. Білодід Д. В. Методика запобігання вразливостей front-end частини web-застосунків : кваліфікаційна робота магістра : 121 Інженерія програмного забезпечення. Київ : ДУІКТ, 2025. 84 с. URL: https://surl.li/vzmlw

5. Метод пошуку вразливостей вебзастосунків з використанням API ChatGPT / В. В. Поліщук та ін. *Сучасні інформаційні технології в будівництві та архітектурі*. 2024. № 318240. URL: https://smarttech.knuba.edu.ua/article/view/318240

6. Kashtalian A., Lysenko S., Sachenko A. Evaluation criteria of centralization options in the architecture of multicomputer systems with traps and baits. *Radioelectronic and Computer Systems*. 2025. No. 1. P. 264–297.

7. Alshomrani M. Survey of Transformer-Based Malicious Software Detection Systems. *Electronics*. 2024. Vol. 13, No. 4677. https://doi.org/10.3390/electronics13234677

8. Modern Approaches to Software Vulnerability Detection: A Survey of Machine Learning, Deep Learning, and Large Language Models / T. Zeng et al. *Electronics*. 2025. Vol. 14, No. 22. URL: https://www.mdpi.com/2079-9292/14/22/4449

9. Varga A. A Machine Learning-enhanced web-crawler for vulnerability detection: A binary classification approach : Master's thesis : AI and Machine Learning. Karlskrona : Blekinge Institute of Technology, 2025. 62 p. URL: https://www.diva-portal.org/smash/get/diva2%3A1962633/FULLTEXT01.pdf

10. Web Vulnerabilities using Machine Learning for Prevention and Detection: A Critical Review *International Journal on Emerging Technologies*, Vol. 16, Issue 2, pp. 120–139 (2025) URL: https://www.researchtrend.net/ijet/pdf/Web-Vulnerabilities-using-Machine-Learning-for-Prevention-and-Detection-A-Critical-Review-Oduleye-BE-18.pdf

11. Artificial Intelligence-Driven Supervised Classification Algorithm for Website Vulnerability Detection Using MITRE NVD CVE Scores / J. Doe et al. *Preprints.org*. 2026. URL: https://www.preprints.org/manuscript/202602.0475

12. Busko V. Automatic exploit assessment based on deep learning methods. *Ontology of Designing*. 2024. Vol. 14, No. 2. P. 156–170. URL: https://doi.org/10.18287/2223-9537-2024-14-3-408-420

13. Automated Vulnerability Assessment Using Machine Learning / R. Sharma et al. *ResearchGate*. 2024. URL: https://www.researchgate.net/publication/382918034

14. AI-Driven Automated Vulnerability Scanning for Real-Time Threat Detection and Mitigation / K. Modi et al. *International Journal of Innovative Research in Science, Engineering and Technology*. 2025. Vol. 14, No. 3. DOI: 10.15680/IJIRSET.2025.1403324 URL: https://www.ijirset.com/upload/2025/march/324_AI-Driven.pdf

15. Boucena A. Leveraging Large Language Models For Automated Software Vulnerability Detection and Analysis : Master's thesis. Guelma : University of Guelma, 2025. 95 p. URL: https://dspace.univ-guelma.dz/jspui/bitstream/123456789/18272/1/F5_8_BOUCENA_AMINA_1752072283.pdf

16. Automatic Source Code Vulnerability Detection, Classification, and Prioritization Using Deep Learning / S. Jalowski URL: https://assets-eu.researchsquare.com/files/rs-7423339/v1_covered_14df0d01-894d-4139-86d7-37ff5eae83a7.pdf

17. Савенко О. С., Лисенко С. М., Нічепорук А. О. Виявлення бот-мереж на основі аналізу поведінки мережевих об'єктів у розподілених системах. *Комп'ютерні системи та мережі*. 2020. № 19. С. 190–198. URL: https://computingonline.net/computing/article/view/1761

18. Common Weakness Enumeration (CWE). *MITRE Corporation*. 2025. URL: https://cwe.mitre.org/

19. Gyamfi N. K., Goranin N. Automated System-Level Malware Detection Using Machine Learning: A Comprehensive Review. *Applied Sciences*. 2023. Vol. 13, No. 11908. URL: https://www.mdpi.com/2076-3417/13/21/11908

20. Phishing Website Detection Using Machine Learning / A. Kumar et al. *Dialnet*. 2025. URL: https://dialnet.unirioja.es/descarga/articulo/9930098.pdf

ДОДАТОК В

(обов'язковий)

Лістинг програмного забезпечення ІКС АВВЗ

Цей код не дублює весь код ІКС АВВЗ, а подає зведений варіант основних модулів системи, які прямо описані в роботі:

$$ingest \rightarrow correlate \rightarrow normalize \rightarrow classify \rightarrow risk \rightarrow policy \rightarrow report \rightarrow store$$

Надані основні блоки основних модулів системи в вигляді: доменні моделі, *InMemoryRepository, IngestionModule, CorrelationModule, NormalizationModule з* $N_1..N_6$, *KnowledgeBase, HybridClassifier, RiskModule, PolicyModule, ReportingModule і оркестратор IKSAVVZSystem*

```
from __future__ import annotations

from collections import Counter, defaultdict
from dataclasses import dataclass, field, asdict
from enum import Enum
import hashlib
import html
import json
import math
import re
from typing import Any


# ============================================================
# ENUMS AND CORE MODELS
# ============================================================


class ScanSource(str, Enum):
    SAST = "SAST"
    DAST = "DAST"
    IAST = "IAST"
    SCA = "SCA"
    LOG = "LOG"
    HTTP = "HTTP"
    CONFIG = "CONFIG"
    CONTAINER = "CONTAINER"
    MANUAL = "MANUAL"
    CORRELATED = "CORRELATED"


class ThreatClass(str, Enum):
    SQL_INJECTION = "SQL_INJECTION"
```

```
    XSS = "XSS"
    BROKEN_ACCESS_CONTROL = "BROKEN_ACCESS_CONTROL"
    SECURITY_MISCONFIGURATION = "SECURITY_MISCONFIGURATION"
    VULNERABLE_COMPONENT = "VULNERABLE_COMPONENT"
    SSRF = "SSRF"
    UNKNOWN = "UNKNOWN"


class RiskLevel(str, Enum):
    LOW = "LOW"
    MEDIUM = "MEDIUM"
    HIGH = "HIGH"
    CRITICAL = "CRITICAL"


class DecisionMode(str, Enum):
    PRIMARY = "PRIMARY"
    ABSTAIN_UNKNOWN = "ABSTAIN_UNKNOWN"


@dataclass
class RawFinding:
    source: ScanSource
    application_id: str
    title: str
    description: str
    tenant_id: str = "default-tenant"
    location: str | None = None
    severity: str | None = None
    cvss_score: float | None = None
    cwe_id: str | None = None
    cve_id: str | None = None
    extra: dict[str, str] = field(default_factory=dict)


@dataclass
class NormalizationStageTrace:
    stage_id: str
    stage_name: str
    summary: str


@dataclass
class NormalizedFinding:
    finding_id: str
    correlation_key: str
    tenant_id: str
    application_id: str
    source: ScanSource
    primary_source: ScanSource
    correlated_sources: list[ScanSource]
    source_count: int
    title: str
```

```
    description: str
    location: str
    severity: str
    cvss_score: float
    cwe_id: str | None
    cve_id: str | None
    tokens: list[str]
    business_criticality: float
    exposure: float
    data_sensitivity: float
    mitigation_strength: float
    corroboration_score: float
    normalization_version: str
    normalization_trace: list[NormalizationStageTrace]
    supporting_raw_findings: list[RawFinding] =
field(default_factory=list)


@dataclass
class AlternativeHypothesis:
    threat_class: ThreatClass
    confidence: float
    rank: int


@dataclass
class KnowledgeReference:
    kind: str
    identifier: str
    title: str
    summary: str
    source: str
    url: str


@dataclass
class ClassificationResult:
    finding_id: str
    threat_class: ThreatClass
    confidence: float
    class_probabilities: dict[str, float]
    model_family: str
    primary_hypothesis: ThreatClass
    primary_hypothesis_confidence: float
    decision_margin: float
    decision_mode: DecisionMode
    abstained: bool
    abstain_reason: str | None
    alternative_hypotheses: list[AlternativeHypothesis]
    component_predictions: dict[str, str]
    component_confidences: dict[str, float]
    explanation: str
    owasp_category: str
```

```
    owasp_title: str
    suggested_cwe: str
    capec: str
    knowledge_references: list[KnowledgeReference]


@dataclass
class RiskFactorWeights:
    cvss: float = 0.30
    confidence: float = 0.20
    class_impact: float = 0.15
    business_criticality: float = 0.10
    exposure: float = 0.10
    corroboration: float = 0.05
    data_sensitivity: float = 0.10
    mitigation_strength: float = 0.10
    nonlinear_boost: float = 0.20


@dataclass
class RiskUncertaintySettings:
    n_classes: int = 7
    lambda_epistemic: float = 0.30
    low_threshold: float = 0.20
    medium_threshold: float = 0.50
    confident_threshold: float = 0.35
    high_risk_threshold: float = 7.0


@dataclass
class RiskProfile:
    profile_id: str
    name: str
    version: int
    description: str = ""
    source: str = "builtin"
    is_active: bool = True
    weights: RiskFactorWeights =
field(default_factory=RiskFactorWeights)
    uncertainty: RiskUncertaintySettings =
field(default_factory=RiskUncertaintySettings)
    class_impacts: dict[str, float] = field(
        default_factory=lambda: {
            ThreatClass.SQL_INJECTION.value: 0.95,
            ThreatClass.XSS.value: 0.75,
            ThreatClass.BROKEN_ACCESS_CONTROL.value: 0.90,
            ThreatClass.SECURITY_MISCONFIGURATION.value: 0.60,
            ThreatClass.VULNERABLE_COMPONENT.value: 0.70,
            ThreatClass.SSRF.value: 0.85,
            ThreatClass.UNKNOWN.value: 0.40,
        }
    )
```

```
@dataclass
class RiskResult:
    finding_id: str
    threat_class: ThreatClass
    confidence: float
    epistemic_uncertainty: float
    risk_score_base: float
    risk_score_composite: float
    risk_interval_min: float
    risk_interval_max: float
    uncertainty_label: str
    review_recommended: bool
    final_risk_score: float
    risk_level: RiskLevel
    explanation: str
    profile_id: str


@dataclass
class PolicyDecision:
    quadrant: str
    action: str
    rationale: str
    requires_human_review: bool
    requires_ticket: bool
    sla_hours: int


@dataclass
class FullAnalysisResult:
    normalized: NormalizedFinding
    classification: ClassificationResult
    risk: RiskResult
    policy: PolicyDecision


@dataclass
class BatchSummary:
    total_findings: int
    total_correlated: int
    applications: int
    risk_levels: dict[str, int]
    threat_classes: dict[str, int]
    policy_actions: dict[str, int]
    sources: dict[str, int]
    avg_risk_score: float
    review_queue_size: int
    top_findings: list[str]


# ============================================================
# IN-MEMORY STORAGE
# ============================================================
```

```
class InMemoryRepository:
    """Спрощене сховище appendix-рівня замість production SQLite
contour."""

    def __init__(self) -> None:
        self.raw_findings: list[RawFinding] = []
        self.normalized_findings: list[NormalizedFinding] = []
        self.classification_results: list[ClassificationResult] = []
        self.risk_results: list[RiskResult] = []
        self.final_results: list[FullAnalysisResult] = []
        self.active_risk_profile = RiskProfile(profile_id="default-
v1", name="default", version=1)

    def persist_raw(self, items: list[RawFinding]) -> None:
        self.raw_findings.extend(items)

    def persist_result(self, result: FullAnalysisResult) -> None:
        self.normalized_findings.append(result.normalized)
        self.classification_results.append(result.classification)
        self.risk_results.append(result.risk)
        self.final_results.append(result)

    def get_active_risk_profile(self) -> RiskProfile:
        return self.active_risk_profile

    def summarize(self) -> BatchSummary:
        results = list(self.final_results)
        if not results:
            return BatchSummary(
                total_findings=0,
                total_correlated=0,
                applications=0,
                risk_levels={},
                threat_classes={},
                policy_actions={},
                sources={},
                avg_risk_score=0.0,
                review_queue_size=0,
                top_findings=[],
            )

        risk_counter = Counter(item.risk.risk_level.value for item
in results)
        class_counter =
Counter(item.classification.threat_class.value for item in results)
        policy_counter = Counter(item.policy.action for item in
results)
        source_counter =
Counter(item.normalized.primary_source.value for item in results)
        avg_score = round(sum(item.risk.final_risk_score for item in
results) / len(results), 2)
```

```
        top_findings = [item.normalized.title for item in
sorted(results, key=lambda x: x.risk.final_risk_score,
reverse=True)[:5]]
        review_queue = sum(1 for item in results if
item.risk.review_recommended or item.policy.requires_human_review)
        correlated = sum(1 for item in results if
item.normalized.source_count > 1)

        return BatchSummary(
            total_findings=len(results),
            total_correlated=correlated,
            applications=len({item.normalized.application_id for
item in results}),
            risk_levels=dict(risk_counter),
            threat_classes=dict(class_counter),
            policy_actions=dict(policy_counter),
            sources=dict(source_counter),
            avg_risk_score=avg_score,
            review_queue_size=review_queue,
            top_findings=top_findings,
        )


# ============================================================
# INGESTION
# ============================================================


class IngestionModule:
    SOURCE_ALIASES = {
        "SAST": ScanSource.SAST,
        "SEMGREP": ScanSource.SAST,
        "DAST": ScanSource.DAST,
        "ZAP": ScanSource.DAST,
        "IAST": ScanSource.IAST,
        "SCA": ScanSource.SCA,
        "LOG": ScanSource.LOG,
        "HTTP": ScanSource.HTTP,
        "CONFIG": ScanSource.CONFIG,
        "CONTAINER": ScanSource.CONTAINER,
        "MANUAL": ScanSource.MANUAL,
    }

    def _parse_source(self, value: Any, default: ScanSource) ->
ScanSource:
        if not value:
            return default
        return self.SOURCE_ALIASES.get(str(value).strip().upper(),
default)

    def _ensure_list(self, payload: Any) -> list[Any]:
        if payload is None:
            return []
```

```
        return payload if isinstance(payload, list) else [payload]

    def ingest_report_payload(
        self,
        payload: Any,
        application_id: str,
        tenant_id: str = "default-tenant",
        default_source: ScanSource = ScanSource.MANUAL,
    ) -> list[RawFinding]:
        records = self._ensure_list(payload)
        findings: list[RawFinding] = []

        for item in records:
            if not isinstance(item, dict):
                continue

            source = self._parse_source(item.get("source"),
default_source)
            findings.append(
                RawFinding(
                    source=source,
                    tenant_id=tenant_id,
                    application_id=application_id,
                    title=str(item.get("title") or item.get("name")
or "Untitled finding"),
                    description=str(item.get("description") or
item.get("message") or "No description"),
                    location=item.get("location") or
item.get("path"),
                    severity=item.get("severity"),

cvss_score=self._coerce_float(item.get("cvss_score")),
                    cwe_id=item.get("cwe_id"),
                    cve_id=item.get("cve_id"),
                    extra={str(k): str(v) for k, v in
item.get("extra", {}).items()} if isinstance(item.get("extra"),
dict) else {},
                )
            )
        return findings

    @staticmethod
    def _coerce_float(value: Any) -> float | None:
        try:
            return None if value in (None, "") else float(value)
        except (TypeError, ValueError):
            return None


# ============================================================
# CORRELATION
# ============================================================
```

```
class CorrelationModule:
    """Спрощена дедуплікація і зведення мультиджерельних finding-
ів."""

    @staticmethod
    def build_correlation_key(raw: RawFinding) -> str:
        material = "|".join(
            [
                raw.application_id,
                raw.location or "unknown",
                raw.cwe_id or "",
                raw.cve_id or "",
                raw.title.lower().strip(),
            ]
        )
        return hashlib.sha256(material.encode("utf-
8")).hexdigest()[:16]

    def correlate_raw_findings(self, raw_findings: list[RawFinding])
-> list[RawFinding]:
        grouped: dict[str, list[RawFinding]] = defaultdict(list)
        for item in raw_findings:
            grouped[self.build_correlation_key(item)].append(item)

        correlated: list[RawFinding] = []
        for key, group in grouped.items():
            if len(group) == 1:
                item = group[0]
                item.extra.setdefault("correlation_key", key)
                item.extra.setdefault("source_count", "1")
                item.extra.setdefault("correlated_sources",
item.source.value)
                correlated.append(item)
                continue

            merged_sources = ",".join(sorted({item.source.value for
item in group}))
            best = max(group, key=lambda item:
self._severity_rank(item.severity))
            merged_extra = dict(best.extra)
            merged_extra["correlation_key"] = key
            merged_extra["source_count"] = str(len(group))
            merged_extra["correlated_sources"] = merged_sources
            merged_extra["merged_titles"] = " | ".join(item.title
for item in group)

            correlated.append(
                RawFinding(
                    source=ScanSource.CORRELATED,
                    tenant_id=best.tenant_id,
                    application_id=best.application_id,
                    title=best.title,
                    description=best.description,
```

```
                    location=best.location,
                    severity=best.severity,
                    cvss_score=best.cvss_score,
                    cwe_id=best.cwe_id,
                    cve_id=best.cve_id,
                    extra=merged_extra,
                )
            )
        return correlated

    @staticmethod
    def _severity_rank(value: str | None) -> int:
        ranks = {"LOW": 1, "MEDIUM": 2, "HIGH": 3, "CRITICAL": 4}
        return ranks.get((value or "MEDIUM").upper(), 2)


# ============================================================
# NORMALIZATION N1..N6
# ============================================================


class NormalizationModule:
    SEVERITY_MAP = {
        "info": "LOW",
        "low": "LOW",
        "medium": "MEDIUM",
        "high": "HIGH",
        "critical": "CRITICAL",
    }
    NORMALIZATION_VERSION = "N1-N6.v1"

    def normalize_finding(self, raw: RawFinding) ->
NormalizedFinding:
        state = self._n1_parse_and_sanitize(raw)
        state = self._n2_unify_schema(state)
        state = self._n3_enrich_taxonomy(state)
        state = self._n4_contextualize(state)
        state = self._n5_prepare_correlation(state)
        return self._n6_finalize(state)

    def _n1_parse_and_sanitize(self, raw: RawFinding) -> dict[str,
Any]:
        return {
            "raw": raw,
            "tenant_id": raw.tenant_id or "default-tenant",
            "title": (raw.title or "").strip(),
            "description": (raw.description or "").strip(),
            "location": (raw.location or "unknown").strip(),
            "cwe_id": raw.cwe_id,
            "cve_id": raw.cve_id,
            "trace": [NormalizationStageTrace("N1",
"parse_and_sanitize", "Sanitized raw finding fields and tenant
context.")],
```

```
        }

    def _n2_unify_schema(self, state: dict[str, Any]) -> dict[str,
Any]:
        raw: RawFinding = state["raw"]
        correlation_key = raw.extra.get("correlation_key") or
CorrelationModule.build_correlation_key(raw)
        severity = self.SEVERITY_MAP.get((raw.severity or
"medium").lower(), "MEDIUM")
        correlated_sources = [ScanSource[item.strip()] for item in
raw.extra.get("correlated_sources", raw.source.value).split(",") if
item.strip() in ScanSource.__members__]
        state.update(
            {
                "finding_id":
hashlib.sha256(f"{correlation_key}|{raw.source.value}|{raw.cwe_id}|{
raw.cve_id}".encode("utf-8")).hexdigest()[:16],
                "correlation_key": correlation_key,
                "severity": severity,
                "primary_source": correlated_sources[0] if
correlated_sources else raw.source,
                "correlated_sources": correlated_sources or
[raw.source],
                "source_count": max(1,
int(raw.extra.get("source_count", str(len(correlated_sources) or
1)))),
            }
        )
        state["trace"].append(NormalizationStageTrace("N2",
"unify_schema", "Canonicalized identifiers, source metadata, and
severity scale."))
        return state

    def _n3_enrich_taxonomy(self, state: dict[str, Any]) ->
dict[str, Any]:
        raw: RawFinding = state["raw"]
        fallback = {"LOW": 3.0, "MEDIUM": 5.5, "HIGH": 8.0,
"CRITICAL": 9.5}
        title_desc = f"{state['title']} {state['description']}
{state['location']}".lower()
        tokens = [token for token in re.sub(r"[^a-zA-Z0-9_/\- ]+", "
", title_desc).split() if len(token) > 2]
        state.update({"cvss_score": raw.cvss_score if raw.cvss_score
is not None else fallback[state["severity"]], "tokens": tokens})
        state["trace"].append(NormalizationStageTrace("N3",
"enrich_taxonomy", "Prepared lexical and taxonomy-ready features."))
        return state

    def _n4_contextualize(self, state: dict[str, Any]) -> dict[str,
Any]:
        raw: RawFinding = state["raw"]
        text = f"{raw.title} {raw.description}".lower()
```

```
        extra_text = " ".join(f"{k}:{v}" for k, v in
raw.extra.items()).lower()
        business_criticality = 0.9 if any(key in text for key in
("payment", "auth", "admin", "login", "token")) else 0.5
        exposure = 0.8 if raw.source in {ScanSource.DAST,
ScanSource.IAST, ScanSource.CONTAINER, ScanSource.CORRELATED} else
0.5
        data_sensitivity = 0.9 if any(key in text for key in
("password", "credential", "session", "cookie", "user data")) else
0.5
        mitigation_strength = 0.2 if "waf" in extra_text or
"sanitized" in extra_text else 0.0
        corroboration_score = min(1.0, 0.45 + 0.2 *
int(raw.extra.get("source_count", "1"))) if
raw.extra.get("source_count") else (0.8 if raw.cwe_id or raw.cve_id
else 0.5)
        state.update(
            {
                "business_criticality": business_criticality,
                "exposure": exposure,
                "data_sensitivity": data_sensitivity,
                "mitigation_strength": mitigation_strength,
                "corroboration_score": corroboration_score,
            }
        )
        state["trace"].append(NormalizationStageTrace("N4",
"contextualize", "Derived contextual risk factors from source and
content."))
        return state

    def _n5_prepare_correlation(self, state: dict[str, Any]) ->
dict[str, Any]:
        state["supporting_raw_findings"] = [state["raw"]]
        state["trace"].append(NormalizationStageTrace("N5",
"prepare_correlation", "Prepared provenance bundle for downstream
correlation and audit."))
        return state

    def _n6_finalize(self, state: dict[str, Any]) ->
NormalizedFinding:
        raw: RawFinding = state["raw"]
        state["trace"].append(NormalizationStageTrace("N6",
"finalize", "Materialized normalized finding artifact for
classification and risk scoring."))
        return NormalizedFinding(
            finding_id=state["finding_id"],
            correlation_key=state["correlation_key"],
            tenant_id=state["tenant_id"],
            application_id=raw.application_id,
            source=raw.source,
            primary_source=state["primary_source"],
            correlated_sources=state["correlated_sources"],
            source_count=state["source_count"],
```

```
            title=state["title"],
            description=state["description"],
            location=state["location"],
            severity=state["severity"],
            cvss_score=state["cvss_score"],
            cwe_id=state["cwe_id"],
            cve_id=state["cve_id"],
            tokens=state["tokens"],
            business_criticality=state["business_criticality"],
            exposure=state["exposure"],
            data_sensitivity=state["data_sensitivity"],
            mitigation_strength=state["mitigation_strength"],
            corroboration_score=state["corroboration_score"],
            normalization_version=self.NORMALIZATION_VERSION,
            normalization_trace=state["trace"],

supporting_raw_findings=state["supporting_raw_findings"],
        )


# ============================================================
# KNOWLEDGE AND CLASSIFICATION
# ============================================================


class KnowledgeBase:
    KB = {
        ThreatClass.SQL_INJECTION: [
            KnowledgeReference("OWASP", "A03:2021", "Injection",
"Injection flaws remain critical in web systems.", "OWASP",
"https://owasp.org/Top10/A03_2021-Injection/"),
            KnowledgeReference("CWE", "CWE-89", "SQL Injection",
"Improper neutralization in SQL query construction.", "MITRE",
"https://cwe.mitre.org/data/definitions/89.html"),
        ],
        ThreatClass.XSS: [
            KnowledgeReference("OWASP", "A03:2021", "Injection",
"Cross-site scripting remains a major browser-side threat.",
"OWASP", "https://owasp.org/Top10/A03_2021-Injection/"),
            KnowledgeReference("CWE", "CWE-79", "Cross-site
Scripting", "Improper neutralization of input during web page
generation.", "MITRE",
"https://cwe.mitre.org/data/definitions/79.html"),
        ],
        ThreatClass.BROKEN_ACCESS_CONTROL: [
            KnowledgeReference("OWASP", "A01:2021", "Broken Access
Control", "Missing authorization checks enable privilege abuse.",
"OWASP", "https://owasp.org/Top10/A01_2021-Broken_Access_Control/"),
        ],
        ThreatClass.SSRF: [
            KnowledgeReference("OWASP", "A10:2021", "SSRF", "Server-
side request forgery enables internal resource access.", "OWASP",
```

```
"https://owasp.org/Top10/A10_2021-Server-
Side_Request_Forgery_%28SSRF%29/"),
        ],
        ThreatClass.SECURITY_MISCONFIGURATION: [
            KnowledgeReference("OWASP", "A05:2021", "Security
Misconfiguration", "Unsafe defaults and exposed services create a
broad attack surface.", "OWASP", "https://owasp.org/Top10/A05_2021-
Security_Misconfiguration/"),
        ],
        ThreatClass.VULNERABLE_COMPONENT: [
            KnowledgeReference("OWASP", "A06:2021", "Vulnerable and
Outdated Components", "Components with known vulnerabilities expose
inherited risk.", "OWASP", "https://owasp.org/Top10/A06_2021-
Vulnerable_and_Outdated_Components/"),
        ],
        ThreatClass.UNKNOWN: [
            KnowledgeReference("NOTE", "UNKNOWN", "Manual Review
Required", "The finding should be manually reviewed or enriched with
more evidence.", "IKS-RD", "about:blank"),
        ],
    }

    def lookup(self, threat_class: ThreatClass) ->
list[KnowledgeReference]:
        return list(self.KB.get(threat_class,
self.KB[ThreatClass.UNKNOWN]))


class HybridClassifier:
    """
    Спрощений appendix-рівень варіант hybrid classifier.
    У production-системі це відповідає
text/context/source/structural/LLM fusion.
    """

    KEYWORDS = {
        ThreatClass.SQL_INJECTION: ("sql", "select", "union",
"database", "query", "injection"),
        ThreatClass.XSS: ("xss", "script", "javascript", "dom",
"reflected", "stored"),
        ThreatClass.BROKEN_ACCESS_CONTROL: ("idor", "access",
"admin", "authorization", "permission", "privilege"),
        ThreatClass.SECURITY_MISCONFIGURATION: ("misconfig",
"debug", "exposed", "default", "directory listing", "open port"),
        ThreatClass.VULNERABLE_COMPONENT: ("dependency", "package",
"library", "component", "cve", "version"),
        ThreatClass.SSRF: ("ssrf", "callback", "metadata", "internal
host", "169.254", "fetch"),
    }

    def __init__(self, knowledge_base: KnowledgeBase) -> None:
        self.knowledge_base = knowledge_base
```

```
    def classify_finding(self, finding: NormalizedFinding) ->
ClassificationResult:
        text_distribution = self._text_signal(finding)
        context_distribution = self._context_signal(finding)
        source_distribution = self._source_signal(finding)
        structural_distribution = self._structural_signal(finding)
        semantic_distribution = self._semantic_signal(finding)

        final = self._weighted_fusion(
            [
                (text_distribution, 0.30),
                (context_distribution, 0.20),
                (source_distribution, 0.10),
                (structural_distribution, 0.15),
                (semantic_distribution, 0.25),
            ]
        )

        ranked = sorted(final.items(), key=lambda item: item[1],
reverse=True)
        top_label, top_conf = ranked[0]
        second_conf = ranked[1][1] if len(ranked) > 1 else 0.0
        decision_margin = round(top_conf – second_conf, 4)

        if top_conf < 0.45 or decision_margin < 0.08:
            final_label = ThreatClass.UNKNOWN
            decision_mode = DecisionMode.ABSTAIN_UNKNOWN
            abstained = True
            abstain_reason = "Low confidence or insufficient
separation between competing hypotheses."
        else:
            final_label = ThreatClass(top_label)
            decision_mode = DecisionMode.PRIMARY
            abstained = False
            abstain_reason = None

        alternatives = [
            AlternativeHypothesis(threat_class=ThreatClass(label),
confidence=round(score, 4), rank=index + 1)
            for index, (label, score) in enumerate(ranked[1:4])
        ]
        knowledge = self.knowledge_base.lookup(final_label)

        return ClassificationResult(
            finding_id=finding.finding_id,
            threat_class=final_label,
            confidence=round(top_conf, 4),
            class_probabilities={label: round(score, 4) for label,
score in final.items()},
            model_family="hybrid_weighted_ensemble",
            primary_hypothesis=ThreatClass(top_label),
            primary_hypothesis_confidence=round(top_conf, 4),
            decision_margin=decision_margin,
```

```
            decision_mode=decision_mode,
            abstained=abstained,
            abstain_reason=abstain_reason,
            alternative_hypotheses=alternatives,
            component_predictions={
                "text": max(text_distribution,
key=text_distribution.get),
                "context": max(context_distribution,
key=context_distribution.get),
                "source": max(source_distribution,
key=source_distribution.get),
                "structural": max(structural_distribution,
key=structural_distribution.get),
                "semantic": max(semantic_distribution,
key=semantic_distribution.get),
            },
            component_confidences={
                "text": round(max(text_distribution.values()), 4),
                "context": round(max(context_distribution.values()),
4),
                "source": round(max(source_distribution.values()),
4),
                "structural":
round(max(structural_distribution.values()), 4),
                "semantic":
round(max(semantic_distribution.values()), 4),
            },
            explanation=f"Hybrid fusion over
text/context/source/structural/semantic signals; top={top_label};
margin={decision_margin:.4f}.",
            owasp_category=self._owasp_category(final_label),
            owasp_title=self._owasp_title(final_label),
            suggested_cwe=self._suggested_cwe(final_label),
            capec=self._capec(final_label),
            knowledge_references=knowledge,
        )

    def _blank_distribution(self) -> dict[str, float]:
        return {item.value: 0.01 for item in ThreatClass}

    def _text_signal(self, finding: NormalizedFinding) -> dict[str,
float]:
        text = f"{finding.title} {finding.description}
{finding.location}".lower()
        result = self._blank_distribution()
        for threat_class, markers in self.KEYWORDS.items():
            result[threat_class.value] += sum(0.13 for marker in
markers if marker in text)
        return self._normalize_distribution(result)

    def _context_signal(self, finding: NormalizedFinding) ->
dict[str, float]:
        result = self._blank_distribution()
```

```
        if finding.business_criticality >= 0.8 and "admin" in
finding.title.lower():
            result[ThreatClass.BROKEN_ACCESS_CONTROL.value] += 0.45
        if finding.data_sensitivity >= 0.8 and "cookie" in
finding.description.lower():
            result[ThreatClass.XSS.value] += 0.35
        if finding.cve_id:
            result[ThreatClass.VULNERABLE_COMPONENT.value] += 0.50
        return self._normalize_distribution(result)

    def _source_signal(self, finding: NormalizedFinding) ->
dict[str, float]:
        result = self._blank_distribution()
        if finding.primary_source in {ScanSource.SCA,
ScanSource.CONFIG}:
            result[ThreatClass.VULNERABLE_COMPONENT.value] += 0.35
            result[ThreatClass.SECURITY_MISCONFIGURATION.value] +=
0.25
        if finding.primary_source in {ScanSource.DAST,
ScanSource.HTTP}:
            result[ThreatClass.SQL_INJECTION.value] += 0.25
            result[ThreatClass.XSS.value] += 0.20
            result[ThreatClass.SSRF.value] += 0.20
        return self._normalize_distribution(result)

    def _structural_signal(self, finding: NormalizedFinding) ->
dict[str, float]:
        result = self._blank_distribution()
        joined = " ".join(finding.tokens)
        if "query" in joined or "database" in joined:
            result[ThreatClass.SQL_INJECTION.value] += 0.50
        if "permission" in joined or "access" in joined:
            result[ThreatClass.BROKEN_ACCESS_CONTROL.value] += 0.45
        return self._normalize_distribution(result)

    def _semantic_signal(self, finding: NormalizedFinding) ->
dict[str, float]:
        result = self._blank_distribution()
        text = f"{finding.title} {finding.description}".lower()
        if "metadata" in text or "callback" in text:
            result[ThreatClass.SSRF.value] += 0.55
        if "dependency" in text or "library" in text:
            result[ThreatClass.VULNERABLE_COMPONENT.value] += 0.40
        if "misconfig" in text or "default credentials" in text:
            result[ThreatClass.SECURITY_MISCONFIGURATION.value] +=
0.45
        return self._normalize_distribution(result)

    def _weighted_fusion(self, parts: list[tuple[dict[str, float],
float]]) -> dict[str, float]:
        fused = {item.value: 0.0 for item in ThreatClass}
        for distribution, weight in parts:
            for key, value in distribution.items():
```

```
                fused[key] += value * weight
        return self._normalize_distribution(fused)

    @staticmethod
    def _normalize_distribution(values: dict[str, float]) ->
dict[str, float]:
        total = sum(max(value, 0.0) for value in values.values()) or
1.0
        return {key: max(value, 0.0) / total for key, value in
values.items()}

    @staticmethod
    def _owasp_category(threat_class: ThreatClass) -> str:
        mapping = {
            ThreatClass.SQL_INJECTION: "A03:2021",
            ThreatClass.XSS: "A03:2021",
            ThreatClass.BROKEN_ACCESS_CONTROL: "A01:2021",
            ThreatClass.SECURITY_MISCONFIGURATION: "A05:2021",
            ThreatClass.VULNERABLE_COMPONENT: "A06:2021",
            ThreatClass.SSRF: "A10:2021",
            ThreatClass.UNKNOWN: "MANUAL-REVIEW",
        }
        return mapping[threat_class]

    @staticmethod
    def _owasp_title(threat_class: ThreatClass) -> str:
        return {
            ThreatClass.SQL_INJECTION: "Injection",
            ThreatClass.XSS: "Injection",
            ThreatClass.BROKEN_ACCESS_CONTROL: "Broken Access
Control",
            ThreatClass.SECURITY_MISCONFIGURATION: "Security
Misconfiguration",
            ThreatClass.VULNERABLE_COMPONENT: "Vulnerable and
Outdated Components",
            ThreatClass.SSRF: "Server-Side Request Forgery",
            ThreatClass.UNKNOWN: "Manual Review Required",
        }[threat_class]

    @staticmethod
    def _suggested_cwe(threat_class: ThreatClass) -> str:
        return {
            ThreatClass.SQL_INJECTION: "CWE-89",
            ThreatClass.XSS: "CWE-79",
            ThreatClass.BROKEN_ACCESS_CONTROL: "CWE-284",
            ThreatClass.SECURITY_MISCONFIGURATION: "CWE-16",
            ThreatClass.VULNERABLE_COMPONENT: "CWE-1104",
            ThreatClass.SSRF: "CWE-918",
            ThreatClass.UNKNOWN: "TBD",
        }[threat_class]

    @staticmethod
    def _capec(threat_class: ThreatClass) -> str:
```

```
        return {
            ThreatClass.SQL_INJECTION: "CAPEC-66",
            ThreatClass.XSS: "CAPEC-63",
            ThreatClass.BROKEN_ACCESS_CONTROL: "CAPEC-233",
            ThreatClass.SECURITY_MISCONFIGURATION: "CAPEC-673",
            ThreatClass.VULNERABLE_COMPONENT: "CAPEC-470",
            ThreatClass.SSRF: "CAPEC-664",
            ThreatClass.UNKNOWN: "N/A",
        }[threat_class]


# ============================================================
# RISK, UNCERTAINTY, POLICY
# ============================================================


class RiskModule:
    def rank_risk(self, finding: NormalizedFinding, classification:
ClassificationResult, profile: RiskProfile) -> RiskResult:
        r_base = self._calculate_base(finding, classification,
profile)
        entropy =
self._normalized_entropy(classification.class_probabilities,
profile.uncertainty.n_classes)
        r_comp = round(r_base * (1.0 +
profile.uncertainty.lambda_epistemic * entropy), 2)
        r_min = round(r_base, 2)
        r_max = round(r_comp, 2)
        uncertainty_label = self._uncertainty_label(entropy,
profile)
        review = entropy >= profile.uncertainty.medium_threshold or
classification.abstained
        risk_level = self._risk_level_from_score(r_comp)
        explanation = (
            f"R_a={r_base:.2f}; H~={entropy:.3f}; "
            f"R_comp={r_comp:.2f} in [{r_min:.2f}, {r_max:.2f}];
profile={profile.profile_id}"
        )
        return RiskResult(
            finding_id=finding.finding_id,
            threat_class=classification.threat_class,
            confidence=classification.confidence,
            epistemic_uncertainty=round(entropy, 4),
            risk_score_base=r_base,
            risk_score_composite=r_comp,
            risk_interval_min=r_min,
            risk_interval_max=r_max,
            uncertainty_label=uncertainty_label,
            review_recommended=review,
            final_risk_score=r_comp,
            risk_level=risk_level,
            explanation=explanation,
            profile_id=profile.profile_id,
```

```
        )

    def _calculate_base(self, finding: NormalizedFinding,
classification: ClassificationResult, profile: RiskProfile) ->
float:
        weights = profile.weights
        class_impact =
profile.class_impacts.get(classification.threat_class.value, 0.40)
        cvss_norm = finding.cvss_score / 10.0
        score = 10.0 * (
            weights.cvss * cvss_norm
            + weights.confidence * classification.confidence
            + weights.class_impact * class_impact
            + weights.business_criticality *
finding.business_criticality
            + weights.exposure * finding.exposure
            + weights.corroboration * finding.corroboration_score
            + weights.data_sensitivity * finding.data_sensitivity
            – weights.mitigation_strength *
finding.mitigation_strength
        )
        nonlinear_boost = 1.0 + weights.nonlinear_boost * cvss_norm
* finding.business_criticality * finding.exposure
        return round(max(0.0, min(10.0, score * nonlinear_boost)),
2)

    @staticmethod
    def _normalized_entropy(probabilities: dict[str, float],
n_classes: int) -> float:
        values = [max(value, 1e-12) for value in
probabilities.values()]
        entropy = -sum(value * math.log(value) for value in values)
        return entropy / math.log(max(n_classes, 2))

    @staticmethod
    def _risk_level_from_score(score: float) -> RiskLevel:
        if score >= 9.0:
            return RiskLevel.CRITICAL
        if score >= 7.0:
            return RiskLevel.HIGH
        if score >= 4.0:
            return RiskLevel.MEDIUM
        return RiskLevel.LOW

    @staticmethod
    def _uncertainty_label(value: float, profile: RiskProfile) ->
str:
        if value < profile.uncertainty.low_threshold:
            return "Низька"
        if value < profile.uncertainty.medium_threshold:
            return "Помірна"
        return "Висока"
```

```
class PolicyModule:
    def build_policy_decision(self, risk: RiskResult) ->
PolicyDecision:
        if risk.risk_level == RiskLevel.CRITICAL:
            return PolicyDecision("Q4", "AUTO-ESCALATE", "Critical
risk requires immediate escalation.", True, True, 4)
        if risk.risk_level == RiskLevel.HIGH:
            return PolicyDecision("Q3", "VERIFY_AND_ESCALATE", "High
risk requires analyst verification and escalation.", True, True, 8)
        if risk.review_recommended:
            return PolicyDecision("Q2", "MANUAL-VERIFY", "Elevated
uncertainty requires manual review.", True, False, 24)
        return PolicyDecision("Q1", "TRACK_AND_MONITOR", "Risk is
controlled by standard remediation flow.", False, False, 72)


# ============================================================
# REPORTING
# ============================================================


class ReportingModule:
    def to_json_report(self, results: list[FullAnalysisResult],
summary: BatchSummary) -> str:
        payload = {
            "summary": asdict(summary),
            "findings": [self._serialize_result(item) for item in
results],
        }
        return json.dumps(payload, ensure_ascii=False, indent=2)

    def to_html_report(self, results: list[FullAnalysisResult],
summary: BatchSummary) -> str:
        rows = []
        for item in results:
            rows.append(
                "<tr>"

f"<td>{html.escape(item.normalized.finding_id)}</td>"
                f"<td>{html.escape(item.normalized.title)}</td>"

f"<td>{html.escape(item.classification.threat_class.value)}</td>"

f"<td>{html.escape(item.risk.risk_level.value)}</td>"
                f"<td>{item.risk.final_risk_score:.2f}</td>"
                f"<td>{html.escape(item.policy.action)}</td>"
                "</tr>"
            )

        return f"""<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="utf-8">
```

```
  <title>ІКС АВВЗ – Консолідований звіт</title>
  <style>
    body {{ font-family: 'Times New Roman', serif; margin: 32px;
color: #111; }}
    h1, h2 {{ margin-bottom: 8px; }}
    table {{ width: 100%; border-collapse: collapse; margin-top:
16px; }}
    th, td {{ border: 1px solid #333; padding: 8px; text-align:
left; vertical-align: top; }}
    .meta {{ margin-bottom: 20px; }}
    .meta div {{ margin: 4px 0; }}
  </style>
</head>
<body>
  <h1>ІКС АВВЗ</h1>
  <h2>Консолідований appendix-звіт</h2>
  <div class="meta">
    <div><strong>Знайдено записів:</strong>
{summary.total_findings}</div>
    <div><strong>Скорельовано:</strong>
{summary.total_correlated}</div>
    <div><strong>Застосунків:</strong> {summary.applications}</div>
    <div><strong>Середній ризик:</strong>
{summary.avg_risk_score:.2f}</div>
    <div><strong>Черга ручного перегляду:</strong>
{summary.review_queue_size}</div>
  </div>
  <table>
    <thead>
      <tr>
        <th>Finding ID</th>
        <th>Назва</th>
        <th>Клас загрози</th>
        <th>Рівень ризику</th>
        <th>Фінальний бал</th>
        <th>Policy Action</th>
      </tr>
    </thead>
    <tbody>
      {"".join(rows)}
    </tbody>
  </table>
</body>
</html>
"""

    @staticmethod
    def _serialize_result(result: FullAnalysisResult) -> dict[str,
Any]:
        data = asdict(result)
        data["normalized"]["source"] =
result.normalized.source.value
```

```
        data["normalized"]["primary_source"] =
result.normalized.primary_source.value
        data["normalized"]["correlated_sources"] = [item.value for
item in result.normalized.correlated_sources]
        data["classification"]["threat_class"] =
result.classification.threat_class.value
        data["classification"]["primary_hypothesis"] =
result.classification.primary_hypothesis.value
        data["classification"]["decision_mode"] =
result.classification.decision_mode.value
        data["risk"]["threat_class"] =
result.risk.threat_class.value
        data["risk"]["risk_level"] = result.risk.risk_level.value
        return data


# ============================================================
# SYSTEM ORCHESTRATION
# ============================================================


class IKSAVVZSystem:
    def __init__(self) -> None:
        self.repository = InMemoryRepository()
        self.ingestion = IngestionModule()
        self.correlation = CorrelationModule()
        self.normalization = NormalizationModule()
        self.knowledge = KnowledgeBase()
        self.classifier = HybridClassifier(self.knowledge)
        self.risk = RiskModule()
        self.policy = PolicyModule()
        self.reporting = ReportingModule()

    def analyze_payload(
        self,
        payload: Any,
        application_id: str,
        tenant_id: str = "default-tenant",
        default_source: ScanSource = ScanSource.MANUAL,
    ) -> tuple[list[FullAnalysisResult], BatchSummary]:
        raw_findings = self.ingestion.ingest_report_payload(
            payload=payload,
            application_id=application_id,
            tenant_id=tenant_id,
            default_source=default_source,
        )
        self.repository.persist_raw(raw_findings)

        correlated =
self.correlation.correlate_raw_findings(raw_findings)
        results: list[FullAnalysisResult] = []
        profile = self.repository.get_active_risk_profile()
```

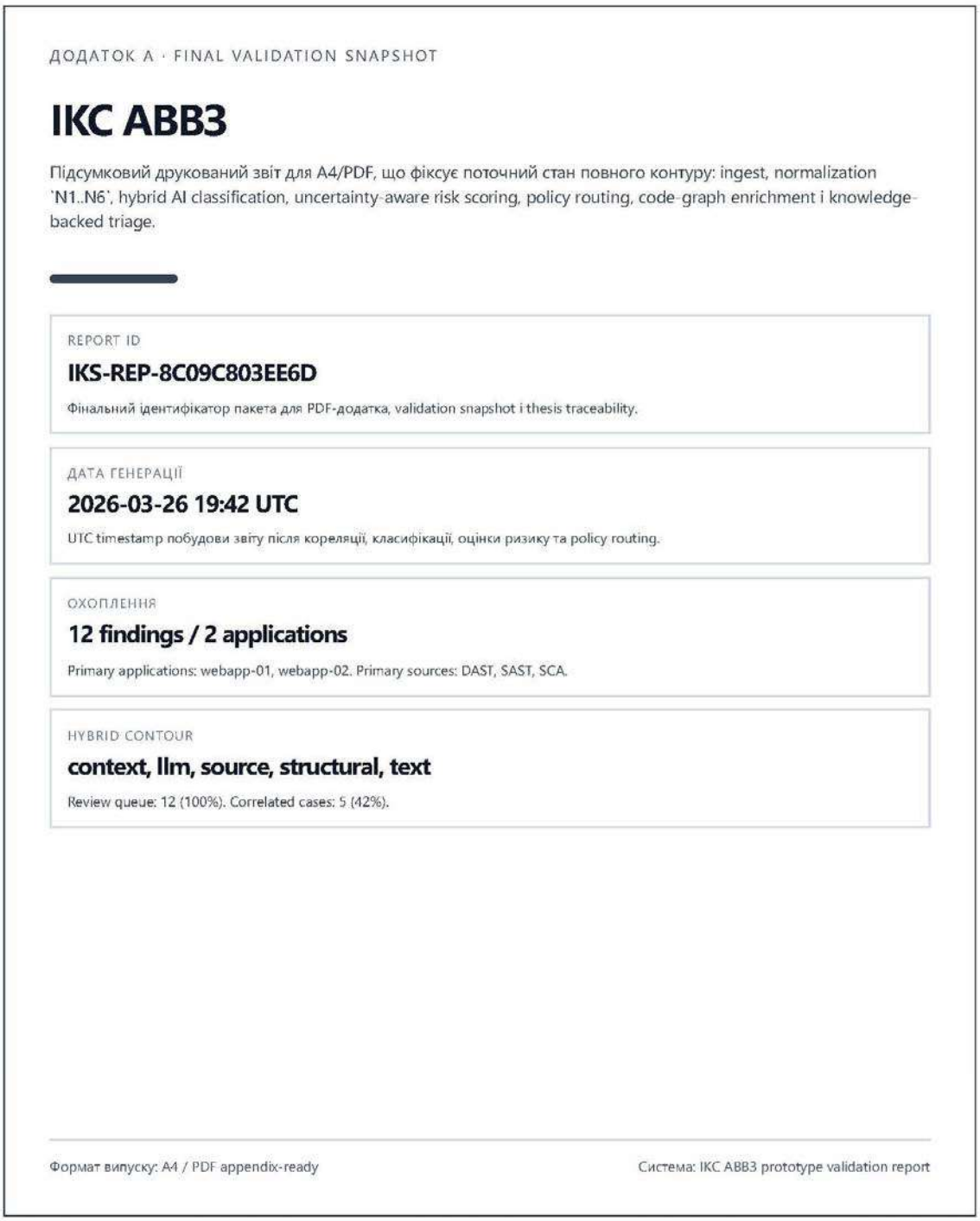
```
        for raw in correlated:
            normalized = self.normalization.normalize_finding(raw)
            classification =
self.classifier.classify_finding(normalized)
            risk = self.risk.rank_risk(normalized, classification,
profile)
            policy = self.policy.build_policy_decision(risk)
            final = FullAnalysisResult(normalized=normalized,
classification=classification, risk=risk, policy=policy)
            self.repository.persist_result(final)
            results.append(final)

        results.sort(key=lambda item: item.risk.final_risk_score,
reverse=True)
        return results, self.repository.summarize()


# ============================================================
# DEMO ENTRYPOINT
# ============================================================


def build_demo_payload() -> list[dict[str, Any]]:
    return [
        {
            "source": "DAST",
            "title": "SQL injection in login form",
            "description": "Unsanitized query parameter reaches
SELECT statement in login endpoint.",
            "location": "/login",
            "severity": "high",
            "cvss_score": 8.9,
            "cwe_id": "CWE-89",
        },
        {
            "source": "HTTP",
            "title": "Reflected script execution in search",
            "description": "User supplied script payload is rendered
in browser without output encoding.",
            "location": "/search",
            "severity": "high",
            "cwe_id": "CWE-79",
        },
        {
            "source": "SCA",
            "title": "Outdated dependency with known CVE",
            "description": "Backend library version has a published
CVE and vulnerable component marker.",
            "location": "requirements.txt",
            "severity": "medium",
            "cve_id": "CVE-2025-0001",
        },
        {
```

```
            "source": "CONFIG",
            "title": "Default credentials and debug mode enabled",
            "description": "Administrative debug interface exposed
with default credentials.",
            "location": "deployment.yaml",
            "severity": "critical",
            "cwe_id": "CWE-16",
        },
    ]


def main() -> None:
    system = IKSAVVZSystem()
    results, summary = system.analyze_payload(
        payload=build_demo_payload(),
        application_id="demo-webapp",
        tenant_id="thesis-demo",
        default_source=ScanSource.MANUAL,
    )

    print("=== IKS AVVZ APPENDIX DEMO ===")
    print(f"Findings: {summary.total_findings}")
    print(f"Correlated: {summary.total_correlated}")
    print(f"Average risk score: {summary.avg_risk_score}")
    print()

    for item in results:
        print(
            f"[{item.risk.risk_level.value}] "
            f"{item.normalized.title} -> "
            f"{item.classification.threat_class.value} -> "
            f"{item.policy.action}
({item.risk.final_risk_score:.2f})"
        )

    print()
    print("JSON report preview:")
    print(system.reporting.to_json_report(results, summary))


if __name__ == "__main__":
    main()
```

ДОДАТОК Г

(додатковий)

Приклад сформованого звіту розробленої системи ІКС АВВЗ



ДОДАТОК A · FINAL VALIDATION SNAPSHOT

ІКС АВВЗ

Підсумковий друкований звіт для A4/PDF, що фіксує поточний стан повного контуру: ingest, normalization `N1..N6`, hybrid AI classification, uncertainty-aware risk scoring, policy routing, code-graph enrichment і knowledge-backed triage.

REPORT ID

IKS-REP-8C09C803EE6D

Фінальний ідентифікатор пакета для PDF-додатка, validation snapshot і thesis traceability.

ДАТА ГЕНЕРАЦІЇ

2026-03-26 19:42 UTC

UTC timestamp побудови звіту після кореляції, класифікації, оцінки ризику та policy routing.

ОХОПЛЕННЯ

12 findings / 2 applications

Primary applications: webapp-01, webapp-02. Primary sources: DAST, SAST, SCA.

HYBRID CONTOUR

context, llm, source, structural, text

Review queue: 12 (100%). Correlated cases: 5 (42%).

Формат випуску: A4 / PDF appendix-ready

Система: ІКС АВВЗ prototype validation report

ІКС АВВ3 - фінальний операційний звіт аналізу вразливостей

Звіт зібрано у delivery-формі для triage, DevSecOps handoff і thesis-grade демонстрації повного контуру: ingest, normalization N1..N6, hybrid AI classification, uncertainty-aware risk scoring, policy routing та knowledge enrichment.

Report ID: IKS-REP-8C09C803EE6D | Сформовано: 2026-03-26 19:42 UTC | Знахідок: 12 | Застосунків: 2

Найвищий рівень: CRITICAL

Hybrid contour: context, llm, source, structural, text

Manual review queue: 12 (100%)

Correlated findings: 5 (42%)

Classifier abstentions: 2

Операційний підсумок

Це executive-level зріз поточного пакета. Він дає картину навантаження на triage, масштабу кореляції між каналами та фактичної інтенсивності ризику вже після роботи hybrid runtime.

ВСЬОГО ЗНАХІДОК	СЕРЕДНІЙ РИЗИК	MANUAL REVIEW
12	**8.73**	**12**
Після кореляції і злиття мультиджерельних підтверджень	Композитна risk-оцінка для всього пакета	Поточна частка review = 100%

CORRELATED CASES	ABSTAIN / UNKNOWN	POLICY ACTION TYPES
5	**2**	**2**
Частка мультиканальних кейсів = 42%	Сигнал для подальшого dataset expansion і selective gating	Різні operational outcomes, згенеровані policy engine

Метод і контур

Професійний формат звіту має явно показувати не лише finding list, а й метод побудови рішення.

- Нормалізація виконується як formal pipeline N1..N6 з trace metadata.
- Classifier decision базується на hybrid fusion компонентів context, llm, source, structural, text.
- Knowledge layer використовує OWASP, CWE, CAPEC, CVE і CVSS enrichment.
- Risk contour враховує confidence, uncertainty interval, policy quadrant і SLA.

Пріоритети черги

Перші позиції реагування, уже відсортовані за risk + confidence + policy contours.

1. Runtime SQL Injection in login endpoint [CRITICAL/VERIFY_AND_ESCALATE]
2. IDOR in user profile endpoint [CRITICAL/VERIFY_AND_ESCALATE]
3. SQL Injection via search API [CRITICAL/VERIFY_AND_ESCALATE]
4. Horizontal privilege escalation in orders API [CRITICAL/VERIFY_AND_ESCALATE]
5. Stored XSS in comment field [CRITICAL/VERIFY_AND_ESCALATE]

Operational readiness

Поточний пакет уже придатний для handoff у зовнішні контури і ручний triage.

- Policy engine сформував 2 типи дій для цього звіту.

- Найвищий ризик у пакеті - CRITICAL.
- Кореляція покрила 5 кейсів із кількох каналів спостереження.

Розподіл ризиків і класів

Два ключові зрізи: operational severity і taxonomy-level структура загроз.

CRITICAL 7 HIGH 3 MEDIUM 2

Threat taxonomy

BROKEN_ACCESS_CONTROL 2 SECURITY_MISCONFIGURATION 2 SQL_INJECTION 2 UNKNOWN 2 VULNERABLE_COMPONENT 2 SSRF 1 XSS 1

Джерела і policy matrix

Тут видно, які канали реально дали сигнал і які operational рішення породив policy layer.

CORRELATED 5 DAST 3 SAST 2 SCA 2

Policy matrix

VERIFY_AND_ESCALATE 8 MANUAL-VERIFY 4

Черга пріоритетного реагування

Компактний executive-реєстр для першої хвилі дій без переходу в повні finding dossier.

#	ЗНАХІДКА	КЛАС	РИЗИК	POLICY	SLA
1	Runtime SQL Injection in login endpoint webapp-01 · /app/auth/login.py:47	SQL_INJECTION	CRITICAL · 10.00	VERIFY_AND_ESCALATE	8h
2	SQL Injection via search API webapp-02 · /api/products/search	SQL_INJECTION	CRITICAL · 10.00	VERIFY_AND_ESCALATE	8h
3	IDOR in user profile endpoint webapp-01 · /api/users/*/profile	BROKEN_ACCESS_CONTROL	CRITICAL · 10.00	VERIFY_AND_ESCALATE	8h
4	Horizontal privilege escalation in orders API webapp-02 · /api/orders/*	BROKEN_ACCESS_CONTROL	CRITICAL · 10.00	VERIFY_AND_ESCALATE	8h
5	Stored XSS in comment field webapp-02 · /app/comments/views.py:88	UNKNOWN	CRITICAL · 9.42	VERIFY_AND_ESCALATE	8h
6	SSRF in webhook URL parameter webapp-01 · /api/webhooks/register	SSRF	CRITICAL · 9.41	VERIFY_AND_ESCALATE	8h

Табличний огляд знахідок

Головна таблиця поєднує технічний контекст, AI decision output, risk interval, uncertainty і policy-routing в одному компактному реєстрі.

#ID / ЛОКАЦІЯ	ЗНАХІДКА / КОМПОНЕНТ	КЛАС / OWASP / CWE/CVE	ДЖЕРЕЛА	РИЗИК	POLICY	ML / UNCERTAINTY
435d876bafb43f4e /app/auth/login.py:47	Runtime SQL Injection in login endpoint webapp-01	SQL_INJECTION A03:2021 · CWE CWE-89 · CVE -	SAST IAST primary=SAST · count=2	CRITICAL · 10.00 CVSS 9.80 · interval 9.28-10.00	VERIFY_AND_ESCALATE Q=B · SLA 8h · review Так	57% Висока · H~ 0.743

#ID / ЛОКАЦІЯ	ЗНАХІДКА / КОМПОНЕНТ	КЛАС / OWASP / CWE/CVE	ДЖЕРЕЛА	РИЗИК	POLICY	ML / UNCERTAINTY
6d9068c1104e9320 /api/users/*/profile	IDOR in user profile endpoint webapp-01	BROKEN_ACCESS_CONTROL A01:2021 · CWE CWE-284 · CVE -	DAST primary=DAST · count=1	CRITICAL · 10.00 CVSS 8.10 · interval 8.21-10.00	VERIFY_AND_ESCALATE Q=B · SLA 8h · review Так	48% Висока · H~ 0.828
6406c34662803512 /api/products/search	SQL Injection via search API webapp-02	SQL_INJECTION A03:2021 · CWE CWE-89 · CVE -	DAST primary=DAST · count=1	CRITICAL · 10.00 CVSS 9.10 · interval 9.27-10.00	VERIFY_AND_ESCALATE Q=B · SLA 8h · review Так	52% Висока · H~ 0.795
846917fa66078564 /api/orders/*	Horizontal privilege escalation in orders API webapp-02	BROKEN_ACCESS_CONTROL A01:2021 · CWE CWE-284 · CVE -	DAST HTTP primary=DAST · count=2	CRITICAL · 10.00 CVSS 8.30 · interval 8.25-10.00	VERIFY_AND_ESCALATE Q=B · SLA 8h · review Так	45% Висока · H~ 0.859
43d2f0d7c87989f6 /app/comments/views.py:88	Stored XSS in comment field webapp-02	UNKNOWN A03:2021 · CWE CWE-79 · CVE -	SAST LOG primary=SAST · count=2	CRITICAL · 9.42 CVSS 8.00 · interval 7.52-9.42	VERIFY_AND_ESCALATE Q=B · SLA 8h · review Так	46% Висока · H~ 0.842
e858f560a7025a8d /api/webhooks/register	SSRF in webhook URL parameter webapp-01	SSRF A10:2021 · CWE CWE-918 · CVE -	DAST primary=DAST · count=1	CRITICAL · 9.41 CVSS 8.60 · interval 7.70-9.41	VERIFY_AND_ESCALATE Q=B · SLA 8h · review Так	57% Висока · H~ 0.741
e36b799b00c2f1e3 pom.xml:line 45	Vulnerable dependency: log4j 2.14.1 webapp-01	VULNERABLE_COMPONENT A06:2021 · CWE CWE-937 · CVE CVE-2021-44228	SCA primary=SCA · count=1	CRITICAL · 9.07 CVSS 10.00 · interval 7.29-9.07	VERIFY_AND_ESCALATE Q=B · SLA 8h · review Так	50% Висока · H~ 0.816
7ac1342c3e5a36c0 /search?q=	Reflected XSS in search parameter webapp-01	XSS A03:2021 · CWE CWE-79 · CVE -	DAST HTTP primary=DAST · count=2	HIGH · 8.63 CVSS 7.40 · interval 7.04-8.63	VERIFY_AND_ESCALATE Q=B · SLA 8h · review Так	54% Висока · H~ 0.754
0a6e7ba698dc3f44 requirements.txt:3	Outdated library: Django 3.0.7 webapp-01	VULNERABLE_COMPONENT A06:2021 · CWE CWE-937 · CVE CVE-2021-35042	SCA primary=SCA · count=1	HIGH · 7.92 CVSS 7.50 · interval 6.31-7.92	MANUAL-VERIFY Q=D · SLA 24h · review Так	44% Висока · H~ 0.849
57cd724c9e8c5333 /config/settings.py:12	Debug mode enabled in production config webapp-01	SECURITY_MISCONFIGURATION A05:2021 · CWE CWE-16 · CVE -	SAST CONFIG primary=SAST · count=2	HIGH · 7.43 CVSS 5.30 · interval 6.01-7.43	MANUAL-VERIFY Q=D · SLA 24h · review Так	52% Висока · H~ 0.790
3762da33148057fb /app/middleware/session.py:23	Missing HttpOnly flag on session cookie webapp-01	UNKNOWN A05:2021 · CWE CWE-16 · CVE -	SAST primary=SAST · count=1	MEDIUM · 6.87 CVSS 5.40 · interval 5.40-6.87	MANUAL-VERIFY Q=D · SLA 24h · review Так	37% Висока · H~ 0.909
a1068d5ac34d0a50 /app/middleware/headers.py	Missing X-Frame-Options header webapp-02	SECURITY_MISCONFIGURATION A05:2021 · CWE CWE-16 · CVE -	SAST primary=SAST · count=1	MEDIUM · 6.01 CVSS 4.30 · interval 4.68-6.01	MANUAL-VERIFY Q=D · SLA 24h · review Так	32% Висока · H~ 0.947

Деталізовані картки знахідок

Повні finding dossier для ручного triage, audit trail і зовнішнього handoff. Тут зібрано decision rationale, normalization trace, enrichment і remediation advice.

FINDING DOSSIER

CRITICAL

Runtime SQL Injection in login endpoint

webapp-01 · /app/auth/login.py:47 · 435d876bafb43f4e

THREAT CLASS
SQL_INJECTION

OWASP
A03:2021 · Injection

SOURCES
SAST IAST

POLICY
VERIFY_AND_ESCALATE · SLA 8h

RISK INTERVAL
9.28-10.00

UNCERTAINTY
Висока · H~ 0.743

CLASSIFIER ENSEMBLE

hybrid_weighted_ensemble · context=SECURITY_MISCONFIGURATION/0.38, llm=SQL_INJECTION/0.25, source=SQL_INJECTION/0.49, structural=SQL_INJECTION/0.68, text=SQL_INJECTION/0.53

CLASSIFIER DECISION

mode=PRIMARY · final=SQL_INJECTION · primary=SQL_INJECTION/0.57 · margin=0.48

alternatives=-

RISK PROFILE

thesis-default:v1 · lambda=0.30 · high=7.00 · review=0.50

NORMALIZATION PIPELINE

N1-N6.v1 · N1 -> N2 -> N3 -> N4 -> N5 -> N6

N1
parse_and_sanitize
Sanitized raw finding fields and tenant context.

N2
unify_schema
Canonicalized identifiers, source metadata, and severity scale.

N3
enrich_taxonomy
Prepared lexical and taxonomy-ready features.

N4
contextualize
Derived contextual risk factors from source and content.

N5
prepare_correlation
Prepared provenance bundle with 2 supporting raw finding(s).

N6
finalize
Materialized normalized finding artifact for downstream classification and risk scoring.

ОПИС

Unparameterized SQL query in login handler: SELECT * FROM users WHERE username='<input>' allows attacker to bypass authentication via SQL injection.

LLM-ASSISTED SEMANTIC LAYER

Семантичний шар інтерпретує знахідку як SQL_INJECTION. Ключовий сигнал формується з опису, контексту SAST+1 та

SSRF=0.08. Пріоритетним кроком слід вважати усунення кореневої причини в коді або конфігурації, а не лише компенсаційний контроль.

backend=template · prompt=semantic-assist-v1

ПОЯСНЕННЯ МОДЕЛІ

Ensemble-класифікатор визначив клас SQL_INJECTION з довірою 0.57. Агрегатор F поєднав p_txt, p_ctx, p_gnn, p_src і p_llm (text=SQL_INJECTION:0.53, context=SECURITY_MISCONFIGURATION:0.38, structural=SQL_INJECTION:0.68, source=SQL_INJECTION:0.49, llm=SQL_INJECTION:0.25). Альтернативні гіпотези: суттєвих альтернатив не зафіксовано. Структурний граф: code graph gnn nodes=22, edges=30, source=synthetic, node_types=FILE:2, MODULE:2, IMPORT:1, FUNCTION:3, CFG_ENTRY:3, CFG_EXIT:3, CFG_STATEMENT:5, SYMBOL:3, edge_types=CONTAINS:5, AST_CHILD:3, AST_BODY:5, AST_IMPORT:1, CFG_ENTRY:3, CFG_FLOW:11, CALLS:2. Ключові ознаки: sql, injection, query, select. Семантичний шар інтерпретує знахідку як SQL_INJECTION. Ключовий сигнал формується з опису, контексту SAST+1 та таксономічних прив'язок CWE-89, CAPEC-66. Рівень довіри моделі становить 0.57, а найближчі альтернативи: XSS=0.09, SSRF=0.08. Пріоритетним кроком слід вважати усунення кореневої причини в коді або конфігурації, а не лише компенсаційний контроль.

ОЦІНКА РИЗИКУ

R_a=9.28 (CVSS=9.8, confidence=0.57); Ĥ=0.743 [Висока невизначеність]; R_comp=10.00 ∈ [9.28, 10.00]; profile=thesis-default:v1; ⚠ рекомендується ручний перегляд

CVE/CWE/CAPEC ENRICHMENT

CWE-89 · Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

seed-knowledge

User-controlled input reaches dynamic SQL construction and can modify query semantics or disclose data.

CAPEC-66 · SQL Injection

seed-knowledge

An attacker manipulates database queries through crafted input and abuses insufficient query parameterization.

РЕКОМЕНДОВАНА РЕМЕДІАЦІЯ

Перевести доступ до БД на параметризовані запити, заборонити string concatenation у SQL та додати негативні тести на ін'єкції.

FINDING DOSSIER

CRITICAL

IDOR in user profile endpoint

webapp-01 · /api/users/*/profile · 6d9068c1104e9320

THREAT CLASS
BROKEN_ACCESS_CONTROL

OWASP
A01:2021 · Broken Access Control

SOURCES
DAST

POLICY
VERIFY_AND_ESCALATE · SLA 8h

RISK INTERVAL
8.21-10.00

UNCERTAINTY
Висока · H~ 0.828

CLASSIFIER ENSEMBLE

hybrid_weighted_ensemble · context=BROKEN_ACCESS_CONTROL/0.71, llm=BROKEN_ACCESS_CONTROL/0.17, source=BROKEN_ACCESS_CONTROL/0.44, structural=BROKEN_ACCESS_CONTROL/0.34, text=BROKEN_ACCESS_CONTROL/0.45

CLASSIFIER DECISION

mode=PRIMARY · final=BROKEN_ACCESS_CONTROL · primary=BROKEN_ACCESS_CONTROL/0.48 · margin=0.38

alternatives=-

RISK PROFILE

thesis-default:v1 · lambda=0.30 · high=7.00 · review=0.50

NORMALIZATION PIPELINE

N1-N6.v1 · N1 -> N2 -> N3 -> N4 -> N5 -> N6

N1

parse_and_sanitize

Sanitized raw finding fields and tenant context.

N2
unify_schema
Canonicalized identifiers, source metadata, and severity scale.

N3
enrich_taxonomy
Prepared lexical and taxonomy-ready features.

N4
contextualize
Derived contextual risk factors from source and content.

N5
prepare_correlation
Prepared provenance bundle with 1 supporting raw finding(s).

N6
finalize
Materialized normalized finding artifact for downstream classification and risk scoring.

ОПИС

Changing the user_id parameter in GET /api/users/{id}/profile allows access to other users' personal data without authorization check.

LLM-ASSISTED SEMANTIC LAYER

Семантичний шар інтерпретує знахідку як BROKEN_ACCESS_CONTROL. Ключовий сигнал формується з опису, контексту DAST та таксономічних прив'язок CWE-284, CAPEC-233. Рівень довіри моделі становить 0.48, а найближчі альтернативи: SQL_INJECTION=0.10, XSS=0.10. Пріоритетним кроком слід вважати усунення кореневої причини в коді або конфігурації, а не лише компенсаційний контроль.

backend=template · prompt=semantic-assist-v1

ПОЯСНЕННЯ МОДЕЛІ

Ensemble-класифікатор визначив клас BROKEN_ACCESS_CONTROL з довірою 0.48. Агрегатор F поєднав p_txt, p_ctx, p_gnn, p_src і p_llm (text=BROKEN_ACCESS_CONTROL:0.45, context=BROKEN_ACCESS_CONTROL:0.71, structural=BROKEN_ACCESS_CONTROL:0.34, source=BROKEN_ACCESS_CONTROL:0.44, llm=BROKEN_ACCESS_CONTROL:0.17). Альтернативні гіпотези: суттєвих альтернатив не зафіксовано. Структурний граф: code graph gnn nodes=16, edges=20, source=synthetic, node_types=FILE:2, MODULE:2, IMPORT:1, FUNCTION:2, CFG_ENTRY:2, CFG_EXIT:2, CFG_STATEMENT:3, SYMBOL:2, edge_types=CONTAINS:4, AST_CHILD:2, AST_BODY:3, AST_IMPORT:1, CFG_ENTRY:2, CFG_FLOW:7, CALLS:1. Ключові ознаки: idor, access, authorization. Семантичний шар інтерпретує знахідку як BROKEN_ACCESS_CONTROL. Ключовий сигнал формується з опису, контексту DAST та таксономічних прив'язок CWE-284, CAPEC-233. Рівень довіри моделі становить 0.48, а найближчі альтернативи: SQL_INJECTION=0.10, XSS=0.10. Пріоритетним кроком слід вважати усунення кореневої причини в коді або конфігурації, а не лише компенсаційний контроль.

ОЦІНКА РИЗИКУ

R_a=8.21 (CVSS=8.1, confidence=0.48); Ĥ=0.828 [Висока невизначеність]; R_comp=10.00 ∈ [8.21, 10.00]; profile=thesis-default:v1; ⚠ рекомендується ручний перегляд

CVE/CWE/CAPEC ENRICHMENT

CWE-284 · Improper Access Control

seed-knowledge

Authorization is enforced inconsistently, allowing subjects to reach resources or actions outside intended policy.

CAPEC-233 · Privilege Escalation

seed-knowledge

The attacker abuses missing or weak authorization boundaries to execute actions above the intended privilege level.

РЕКОМЕНДОВАНА РЕМЕДІАЦІЯ

Винести авторизацію в централізований middleware/policy layer, перевіряти ownership/tenant isolation і додати інтеграційні тести на BOLA/IDOR.

FINDING DOSSIER

CRITICAL

SQL Injection via search API

webapp-02 · /api/products/search · 6406c34662803512

THREAT CLASS
SQL_INJECTION

OWASP
A03:2021 · Injection

SOURCES
DAST

POLICY
VERIFY_AND_ESCALATE · SLA 8h

RISK INTERVAL
9.27-10.00

UNCERTAINTY
Висока · H~ 0.795

CLASSIFIER ENSEMBLE

hybrid_weighted_ensemble · context=SECURITY_MISCONFIGURATION/0.41, llm=SQL_INJECTION/0.18, source=SQL_INJECTION/0.51, structural=SQL_INJECTION/0.67, text=SQL_INJECTION/0.43

CLASSIFIER DECISION

mode=PRIMARY · final=SQL_INJECTION · primary=SQL_INJECTION/0.52 · margin=0.43
alternatives=-

RISK PROFILE

thesis-default:v1 · lambda=0.30 · high=7.00 · review=0.50

NORMALIZATION PIPELINE

N1-N6.v1 · N1 -> N2 -> N3 -> N4 -> N5 -> N6

N1
parse_and_sanitize
Sanitized raw finding fields and tenant context.

N2
unify_schema
Canonicalized identifiers, source metadata, and severity scale.

N3
enrich_taxonomy
Prepared lexical and taxonomy-ready features.

N4
contextualize
Derived contextual risk factors from source and content.

N5
prepare_correlation
Prepared provenance bundle with 1 supporting raw finding(s).

N6
finalize
Materialized normalized finding artifact for downstream classification and risk scoring.

ОПИС

GET /api/products/search?name= parameter is vulnerable to UNION-based SQL injection. Attacker can extract credentials and payment data from the database.

LLM-ASSISTED SEMANTIC LAYER

Семантичний шар інтерпретує знахідку як SQL_INJECTION. Ключовий сигнал формується з опису, контексту DAST та таксономічних прив'язок CWE-89, CAPEC-66. Рівень довіри моделі становить 0.52, а найближчі альтернативи: XSS=0.09, BROKEN_ACCESS_CONTROL=0.09. Пріоритетним кроком слід вважати усунення кореневої причини в коді або конфігурації, а не лише компенсаційний контроль.

backend=template · prompt=semantic-assist-v1

ПОЯСНЕННЯ МОДЕЛІ

source=SQL_INJECTION:0.51, llm=SQL_INJECTION:0.18). Альтернативні гіпотези: суттєвих альтернатив не зафіксовано. Структурний граф: code graph gnn nodes=22, edges=30, source=synthetic, node_types=FILE:2, MODULE:2, IMPORT:1, FUNCTION:3, CFG_ENTRY:3, CFG_EXIT:3, CFG_STATEMENT:5, SYMBOL:3, edge_types=CONTAINS:5, AST_CHILD:3, AST_BODY:5, AST_IMPORT:1, CFG_ENTRY:3, CFG_FLOW:11, CALLS:2. Ключові ознаки: sql, injection, database, union. Семантичний шар інтерпретує знахідку як SQL_INJECTION. Ключовий сигнал формується з опису, контексту DAST та таксономічних прив'язок CWE-89, CAPEC-66. Рівень довіри моделі становить 0.52, а найближчі альтернативи: XSS=0.09, BROKEN_ACCESS_CONTROL=0.09. Пріоритетним кроком слід вважати усунення кореневої причини в коді або конфігурації, а не лише компенсаційний контроль.

ОЦІНКА РИЗИКУ

R_a=9.27 (CVSS=9.1, confidence=0.52); Ĥ=0.795 [Висока невизначеність]; R_comp=10.00 ∈ [9.27, 10.00]; profile=thesis-default:v1; ⚠ рекомендується ручний перегляд

CVE/CWE/CAPEC ENRICHMENT

CWE-89 · Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

seed-knowledge

User-controlled input reaches dynamic SQL construction and can modify query semantics or disclose data.

CAPEC-66 · SQL Injection

seed-knowledge

An attacker manipulates database queries through crafted input and abuses insufficient query parameterization.

РЕКОМЕНДОВАНА РЕМЕДІАЦІЯ

Перевести доступ до БД на параметризовані запити, заборонити string concatenation у SQL та додати негативні тести на ін'єкції.

FINDING DOSSIER

CRITICAL

Horizontal privilege escalation in orders API

webapp-02 · /api/orders/* · 846917fa66078564

THREAT CLASS
BROKEN_ACCESS_CONTROL

OWASP
A01:2021 · Broken Access Control

SOURCES
DAST HTTP

POLICY
VERIFY_AND_ESCALATE · SLA 8h

RISK INTERVAL
8.25-10.00

UNCERTAINTY
Висока · H~ 0.859

CLASSIFIER ENSEMBLE

hybrid_weighted_ensemble · context=BROKEN_ACCESS_CONTROL/0.72, llm=BROKEN_ACCESS_CONTROL/0.18, source=BROKEN_ACCESS_CONTROL/0.44, structural=BROKEN_ACCESS_CONTROL/0.31, text=BROKEN_ACCESS_CONTROL/0.38

CLASSIFIER DECISION

mode=PRIMARY · final=BROKEN_ACCESS_CONTROL · primary=BROKEN_ACCESS_CONTROL/0.45 · margin=0.35
alternatives=-

RISK PROFILE

thesis-default:v1 · lambda=0.30 · high=7.00 · review=0.50

NORMALIZATION PIPELINE

N1-N6.v1 · N1 -> N2 -> N3 -> N4 -> N5 -> N6

N1
parse_and_sanitize
Sanitized raw finding fields and tenant context.

N2
unify_schema
Canonicalized identifiers, source metadata, and severity scale.

N3
enrich_taxonomy

N4
contextualize
Derived contextual risk factors from source and content.

N5
prepare_correlation
Prepared provenance bundle with 2 supporting raw finding(s).

N6
finalize
Materialized normalized finding artifact for downstream classification and risk scoring.

ОПИС

Authenticated user can access orders of other users by modifying order_id in GET /api/orders/{order_id}. No ownership check performed.

LLM-ASSISTED SEMANTIC LAYER

Семантичний шар інтерпретує знахідку як BROKEN_ACCESS_CONTROL. Ключовий сигнал формується з опису, контексту DAST+1 та таксономічних прив'язок CWE-284, CAPEC-233. Рівень довіри моделі становить 0.45, а найближчі альтернативи: SQL_INJECTION=0.10, SECURITY_MISCONFIGURATION=0.10. Пріоритетним кроком слід вважати усунення кореневої причини в коді або конфігурації, а не лише компенсаційний контроль.
backend=template · prompt=semantic-assist-v1

ПОЯСНЕННЯ МОДЕЛІ

Ensemble-класифікатор визначив клас BROKEN_ACCESS_CONTROL з довірою 0.45. Агрегатор F поєднав p_txt, p_ctx, p_gnn, p_src і p_llm (text=BROKEN_ACCESS_CONTROL:0.38, context=BROKEN_ACCESS_CONTROL:0.72, structural=BROKEN_ACCESS_CONTROL:0.31, source=BROKEN_ACCESS_CONTROL:0.44, llm=BROKEN_ACCESS_CONTROL:0.18). Альтернативні гіпотези: суттєвих альтернатив не зафіксовано. Структурний граф: code graph gnn nodes=16, edges=20, source=synthetic, node_types=FILE:2, MODULE:2, IMPORT:1, FUNCTION:2, CFG_ENTRY:2, CFG_EXIT:2, CFG_STATEMENT:3, SYMBOL:2, edge_types=CONTAINS:4, AST_CHILD:2, AST_BODY:3, AST_IMPORT:1, CFG_ENTRY:2, CFG_FLOW:7, CALLS:1. Ключові ознаки: access, privilege. Семантичний шар інтерпретує знахідку як BROKEN_ACCESS_CONTROL. Ключовий сигнал формується з опису, контексту DAST+1 та таксономічних прив'язок CWE-284, CAPEC-233. Рівень довіри моделі становить 0.45, а найближчі альтернативи: SQL_INJECTION=0.10, SECURITY_MISCONFIGURATION=0.10. Пріоритетним кроком слід вважати усунення кореневої причини в коді або конфігурації, а не лише компенсаційний контроль.

ОЦІНКА РИЗИКУ

R_a=8.25 (CVSS=8.3, confidence=0.45); Ĥ=0.859 [Висока невизначеність]; R_comp=10.00 ∈ [8.25, 10.00]; profile=thesis-default:v1; ⚠ рекомендується ручний перегляд

CVE/CWE/CAPEC ENRICHMENT

CWE-284 · Improper Access Control
seed-knowledge
Authorization is enforced inconsistently, allowing subjects to reach resources or actions outside intended policy.

CAPEC-233 · Privilege Escalation
seed-knowledge
The attacker abuses missing or weak authorization boundaries to execute actions above the intended privilege level.

РЕКОМЕНДОВАНА РЕМЕДІАЦІЯ

Винести авторизацію в централізований middleware/policy layer, перевіряти ownership/tenant isolation і додати інтеграційні тести на BOLA/IDOR.

FINDING DOSSIER

CRITICAL

Stored XSS in comment field

webapp-02 · /app/comments/views.py:88 · 43d2f0d7c87989f6

THREAT CLASS
UNKNOWN

OWASP
A03:2021 · Injection

SOURCES
SAST LOG

POLICY
VERIFY_AND_ESCALATE · SLA 8h

RISK INTERVAL
7.52-9.42

UNCERTAINTY
Висока · H~ 0.842

CLASSIFIER ENSEMBLE

hybrid_weighted_ensemble · context=BROKEN_ACCESS_CONTROL/0.36, llm=XSS/0.21, source=XSS/0.30, structural=SQL_INJECTION/0.67, text=XSS/0.43

CLASSIFIER DECISION

mode=ABSTAIN_UNKNOWN · final=UNKNOWN · primary=XSS/0.46 · margin=0.32 · abstain=component_disagreement
alternatives=SQL_INJECTION=0.14

RISK PROFILE

thesis-default:v1 · lambda=0.30 · high=7.00 · review=0.50

NORMALIZATION PIPELINE

N1-N6.v1 · N1 -> N2 -> N3 -> N4 -> N5 -> N6

N1
parse_and_sanitize
Sanitized raw finding fields and tenant context.

N2
unify_schema
Canonicalized identifiers, source metadata, and severity scale.

N3
enrich_taxonomy
Prepared lexical and taxonomy-ready features.

N4
contextualize
Derived contextual risk factors from source and content.

N5
prepare_correlation
Prepared provenance bundle with 2 supporting raw finding(s).

N6
finalize
Materialized normalized finding artifact for downstream classification and risk scoring.

ОПИС

User-supplied comment text is stored in database and rendered without escaping in the admin panel. Stored XSS allows session hijacking of administrators.

LLM-ASSISTED SEMANTIC LAYER

Семантичний шар фіксує абстенцію класифікатора: фінальне рішення переведено в UNKNOWN через розбіжність між ensemble-підмодулями. Провідна гіпотеза лишається XSS з довірою 0.46, а найближчі альтернативи: SQL_INJECTION=0.14. Такий запис слід передати на ручну верифікацію перед автоматичною ескалацією або ticketing-процедурою.
backend=template · prompt=semantic-assist-v1

ПОЯСНЕННЯ МОДЕЛІ

Ensemble-класифікатор не зафіксував остаточний клас і перевів запис у UNKNOWN. Провідна гіпотеза: XSS з довірою 0.46; поріг спрацював через суттєва розбіжність між підмодулями ensemble при margin=0.32. Альтернативи: SQL_INJECTION=0.14. Агрегатор F поєднав p_txt, p_ctx, p_gnn, p_src і p_llm (text=XSS:0.43, context=BROKEN_ACCESS_CONTROL:0.36, structural=SQL_INJECTION:0.67, source=XSS:0.30, llm=XSS:0.21). Структурний граф: code graph gnn nodes=22, edges=30, source=synthetic, node_types=FILE:2, MODULE:2, IMPORT:1, FUNCTION:3, CFG_ENTRY:3,

CFG_FLOW:11, CALLS:2. Ключові ознаки: xss. Семантичний шар фіксує абстенцію класифікатора: фінальне рішення переведено в UNKNOWN через розбіжність між ensemble-підмодулями. Провідна гіпотеза лишається XSS з довірою 0.46, а найближчі альтернативи: SQL_INJECTION=0.14. Такий запис слід передати на ручну верифікацію перед автоматичною ескалацією або ticketing-процедурою.

ОЦІНКА РИЗИКУ

R_a=7.52 (CVSS=8.0, confidence=0.46); Ĥ=0.842 [Висока невизначеність]; R_comp=9.42 ∈ [7.52, 9.42]; profile=thesis-default:v1; ⚠ рекомендується ручний перегляд; classifier_abstained=true; primary_hypothesis=XSS; abstain_reason=component_disagreement

CVE/CWE/CAPEC ENRICHMENT

CWE-79 · Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

seed-knowledge

Untrusted input is returned to the browser without sufficient output encoding or context-aware sanitization.

CAPEC-63 · Cross-Site Scripting

seed-knowledge

Malicious script content is reflected or stored so that the victim browser executes attacker-controlled code.

РЕКОМЕНДОВАНА РЕМЕДІАЦІЯ

Передати знахідку на ручну triage-валідацію, розширити навчальні дані та за потреби додати новий клас загроз у модель.

FINDING DOSSIER

CRITICAL

SSRF in webhook URL parameter

webapp-01 · /api/webhooks/register · e858f560a7025a8d

THREAT CLASS
SSRF

OWASP
A10:2021 · Server-Side Request Forgery

SOURCES
DAST

POLICY
VERIFY_AND_ESCALATE · SLA 8h

RISK INTERVAL
7.70-9.41

UNCERTAINTY
Висока · H~ 0.741

CLASSIFIER ENSEMBLE

hybrid_weighted_ensemble · context=SSRF/0.77, llm=SSRF/0.41, source=SSRF/0.35, structural=BROKEN_ACCESS_CONTROL/0.31, text=SSRF/0.53

CLASSIFIER DECISION

mode=PRIMARY · final=SSRF · primary=SSRF/0.57 · margin=0.48
alternatives=-

RISK PROFILE

thesis-default:v1 · lambda=0.30 · high=7.00 · review=0.50

NORMALIZATION PIPELINE

N1-N6.v1 · N1 -> N2 -> N3 -> N4 -> N5 -> N6

N1
parse_and_sanitize
Sanitized raw finding fields and tenant context.

N2
unify_schema
Canonicalized identifiers, source metadata, and severity scale.

N3
enrich_taxonomy
Prepared lexical and taxonomy-ready features.

N4
contextualize
Derived contextual risk factors from source and content.

N5
prepare_correlation
Prepared provenance bundle with 1 supporting raw finding(s).

N6
finalize
Materialized normalized finding artifact for downstream classification and risk scoring.

ОПИС

The webhook URL parameter is fetched server-side without validation. Attacker can specify internal URLs to probe internal services and access AWS metadata endpoint.

LLM-ASSISTED SEMANTIC LAYER

Семантичний шар інтерпретує знахідку як SSRF. Ключовий сигнал формується з опису, контексту DAST та таксономічних прив'язок CWE-918, CAPEC-664. Рівень довіри моделі становить 0.57, а найближчі альтернативи: BROKEN_ACCESS_CONTROL=0.10, XSS=0.08. Пріоритетним кроком слід вважати усунення кореневої причини в коді або конфігурації, а не лише компенсаційний контроль.

backend=template · prompt=semantic-assist-v1

ПОЯСНЕННЯ МОДЕЛІ

Ensemble-класифікатор визначив клас SSRF з довірою 0.57. Агрегатор F поєднав p_txt, p_ctx, p_gnn, p_src і p_llm (text=SSRF:0.53, context=SSRF:0.77, structural=BROKEN_ACCESS_CONTROL:0.31, source=SSRF:0.35, llm=SSRF:0.41). Альтернативні гіпотези: суттєвих альтернатив не зафіксовано. Структурний граф: code graph gnn nodes=16, edges=20, source=synthetic, node_types=FILE:2, MODULE:2, IMPORT:1, FUNCTION:2, CFG_ENTRY:2, CFG_EXIT:2, CFG_STATEMENT:3, SYMBOL:2, edge_types=CONTAINS:4, AST_CHILD:2, AST_BODY:3, AST_IMPORT:1, CFG_ENTRY:2, CFG_FLOW:7, CALLS:1. Ключові ознаки: ssrf, server-side, internal, url. Семантичний шар інтерпретує знахідку як SSRF. Ключовий сигнал формується з опису, контексту DAST та таксономічних прив'язок CWE-918, CAPEC-664. Рівень довіри моделі становить 0.57, а найближчі альтернативи: BROKEN_ACCESS_CONTROL=0.10, XSS=0.08. Пріоритетним кроком слід вважати усунення кореневої причини в коді або конфігурації, а не лише компенсаційний контроль.

ОЦІНКА РИЗИКУ

R_a=7.70 (CVSS=8.6, confidence=0.57); Ĥ=0.741 [Висока невизначеність]; R_comp=9.41 ∈ [7.70, 9.41]; profile=thesis-default:v1; ⚠ рекомендується ручний перегляд

CVE/CWE/CAPEC ENRICHMENT

CWE-918 · Server-Side Request Forgery (SSRF)

seed-knowledge

The application issues outbound requests to attacker-controlled destinations and can be used to reach internal services.

CAPEC-664 · Server-Side Request Forgery

seed-knowledge

Attacker-controlled URLs or destinations coerce the server into making unintended outbound requests.

РЕКОМЕНДОВАНА РЕМЕДІАЦІЯ

Обмежити вихідні запити allowlist-політикою, заблокувати internal IP/file schemes, додати URL validation та мережеву сегментацію.

FINDING DOSSIER

CRITICAL

Vulnerable dependency: log4j 2.14.1

webapp-01 · pom.xml:line 45 · e36b799b00c2f1e3

THREAT CLASS
VULNERABLE_COMPONENT

OWASP
A06:2021 · Vulnerable and Outdated Components

SOURCES
SCA

POLICY
VERIFY_AND_ESCALATE · SLA 8h

RISK INTERVAL

UNCERTAINTY

CLASSIFIER ENSEMBLE

hybrid_weighted_ensemble · context=VULNERABLE_COMPONENT/0.80, llm=VULNERABLE_COMPONENT/0.20, source=VULNERABLE_COMPONENT/0.84, structural=VULNERABLE_COMPONENT/0.35, text=VULNERABLE_COMPONENT/0.35

CLASSIFIER DECISION

mode=PRIMARY · final=VULNERABLE_COMPONENT · primary=VULNERABLE_COMPONENT/0.50 · margin=0.40
alternatives=-

RISK PROFILE

thesis-default:v1 · lambda=0.30 · high=7.00 · review=0.50

NORMALIZATION PIPELINE

N1-N6.v1 · N1 -> N2 -> N3 -> N4 -> N5 -> N6

N1
parse_and_sanitize
Sanitized raw finding fields and tenant context.

N2
unify_schema
Canonicalized identifiers, source metadata, and severity scale.

N3
enrich_taxonomy
Prepared lexical and taxonomy-ready features.

N4
contextualize
Derived contextual risk factors from source and content.

N5
prepare_correlation
Prepared provenance bundle with 1 supporting raw finding(s).

N6
finalize
Materialized normalized finding artifact for downstream classification and risk scoring.

ОПИС

Log4j 2.14.1 is vulnerable to remote code execution via JNDI lookup injection (Log4Shell). Attacker can execute arbitrary code on server.

LLM-ASSISTED SEMANTIC LAYER

Семантичний шар інтерпретує знахідку як VULNERABLE_COMPONENT. Ключовий сигнал формується з опису, контексту SCA та таксономічних прив'язок CVE-2021-44228, CWE-937, CAPEC-468. Рівень довіри моделі становить 0.50, а найближчі альтернативи: SSRF=0.10, XSS=0.09. Пріоритетним кроком слід вважати усунення кореневої причини в коді або конфігурації, а не лише компенсаційний контроль.
backend=template · prompt=semantic-assist-v1

ПОЯСНЕННЯ МОДЕЛІ

Ensemble-класифікатор визначив клас VULNERABLE_COMPONENT з довірою 0.50. Агрегатор F поєднав p_txt, p_ctx, p_gnn, p_src і p_llm (text=VULNERABLE_COMPONENT:0.35, context=VULNERABLE_COMPONENT:0.80, structural=VULNERABLE_COMPONENT:0.35, source=VULNERABLE_COMPONENT:0.84, llm=VULNERABLE_COMPONENT:0.20). Альтернативні гіпотези: суттєвих альтернатив не зафіксовано. Структурний граф: code graph gnn nodes=16, edges=20, source=synthetic, node_types=FILE:2, MODULE:2, IMPORT:1, FUNCTION:2, CFG_ENTRY:2, CFG_EXIT:2, CFG_STATEMENT:3, SYMBOL:2, edge_types=CONTAINS:4, AST_CHILD:2, AST_BODY:3, AST_IMPORT:1, CFG_ENTRY:2, CFG_FLOW:7, CALLS:1. Ключові ознаки: dependency. Семантичний шар інтерпретує знахідку як VULNERABLE_COMPONENT. Ключовий сигнал формується з опису, контексту SCA та таксономічних прив'язок CVE-2021-44228, CWE-937, CAPEC-468. Рівень довіри моделі становить 0.50, а найближчі альтернативи: SSRF=0.10, XSS=0.09. Пріоритетним кроком слід вважати усунення кореневої причини в коді або конфігурації, а не лише компенсаційний контроль.

ОЦІНКА РИЗИКУ

R_a=7.29 (CVSS=10.0, confidence=0.50); Ĥ=0.816 [Висока невизначеність]; R_comp=9.07 ∈ [7.29, 9.07]; profile=thesis-default:v1; Ф рекомендується ручний перегляд

CVE/CWE/CAPEC ENRICHMENT

CVE-2021-44228 · Apache Log4j2 remote code execution via JNDI lookup

seed-knowledge

A crafted log message can trigger remote code execution in affected Log4j2 versions through attacker-controlled JNDI lookups.

CWE-937 · Using Components with Known Vulnerabilities

seed-knowledge

A dependency or runtime component contains a published vulnerability that is already known to the public domain.

CAPEC-468 · Exploitation of Vulnerable Components

seed-knowledge

Known-vulnerable libraries or images are leveraged as the practical entry point for compromise.

РЕКОМЕНДОВАНА РЕМЕДІАЦІЯ

Оновити залежність або контейнерний образ, зафіксувати мінімальну безпечну версію, увімкнути dependency audit і контроль SBOM у пайплайні.

FINDING DOSSIER

HIGH

Reflected XSS in search parameter

webapp-01 · /search?q= · 7ac1342c3e5a36c0

THREAT CLASS

XSS

OWASP

A03:2021 · Injection

SOURCES

DAST HTTP

POLICY

VERIFY_AND_ESCALATE · SLA 8h

RISK INTERVAL

7.04-8.63

UNCERTAINTY

Висока · H~ 0.754

CLASSIFIER ENSEMBLE

hybrid_weighted_ensemble · context=XSS/0.69, llm=XSS/0.27, source=XSS/0.41, structural=SQL_INJECTION/0.66, text=XSS/0.60

CLASSIFIER DECISION

mode=PRIMARY · final=XSS · primary=XSS/0.54 · margin=0.38

alternatives=-

RISK PROFILE

thesis-default:v1 · lambda=0.30 · high=7.00 · review=0.50

NORMALIZATION PIPELINE

N1-N6.v1 · N1 -> N2 -> N3 -> N4 -> N5 -> N6

N1

parse_and_sanitize

Sanitized raw finding fields and tenant context.

N2

unify_schema

Canonicalized identifiers, source metadata, and severity scale.

N3

enrich_taxonomy

Prepared lexical and taxonomy-ready features.

N4

contextualize

Derived contextual risk factors from source and content.

N5

prepare_correlation

Prepared provenance bundle with 2 supporting raw finding(s).

N6
finalize
Materialized normalized finding artifact for downstream classification and risk scoring.

ОПИС

Search query parameter is reflected in response without HTML encoding. Payload <script>alert(1)</script> executes in victim browser.

LLM-ASSISTED SEMANTIC LAYER

Семантичний шар інтерпретує знахідку як XSS. Ключовий сигнал формується з опису, контексту DAST+1 та таксономічних прив'язок CWE-79, CAPEC-63. Рівень довіри моделі становить 0.54, а найближчі альтернативи: SQL_INJECTION=0.16, SSRF=0.09. Пріоритетним кроком слід вважати усунення кореневої причини в коді або конфігурації, а не лише компенсаційний контроль.
backend=template · prompt=semantic-assist-v1

ПОЯСНЕННЯ МОДЕЛІ

Ensemble-класифікатор визначив клас XSS з довірою 0.54. Агрегатор F поєднав p_txt, p_ctx, p_gnn, p_src і p_llm (text=XSS:0.60, context=XSS:0.69, structural=SQL_INJECTION:0.66, source=XSS:0.41, llm=XSS:0.27). Альтернативні гіпотези: суттєвих альтернатив не зафіксовано. Структурний граф: code graph gnn nodes=22, edges=30, source=synthetic, node_types=FILE:2, MODULE:2, IMPORT:1, FUNCTION:3, CFG_ENTRY:3, CFG_EXIT:3, CFG_STATEMENT:5, SYMBOL:3, edge_types=CONTAINS:5, AST_CHILD:3, AST_BODY:5, AST_IMPORT:1, CFG_ENTRY:3, CFG_FLOW:11, CALLS:2. Ключові ознаки: xss, script, html. Семантичний шар інтерпретує знахідку як XSS. Ключовий сигнал формується з опису, контексту DAST+1 та таксономічних прив'язок CWE-79, CAPEC-63. Рівень довіри моделі становить 0.54, а найближчі альтернативи: SQL_INJECTION=0.16, SSRF=0.09. Пріоритетним кроком слід вважати усунення кореневої причини в коді або конфігурації, а не лише компенсаційний контроль.

ОЦІНКА РИЗИКУ

R_a=7.04 (CVSS=7.4, confidence=0.54); Ĥ=0.754 [Висока невизначеність]; R_comp=8.63 ∈ [7.04, 8.63]; profile=thesis-default:v1; ⚠ рекомендується ручний перегляд

CVE/CWE/CAPEC ENRICHMENT

CWE-79 · Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
seed-knowledge
Untrusted input is returned to the browser without sufficient output encoding or context-aware sanitization.

CAPEC-63 · Cross-Site Scripting
seed-knowledge
Malicious script content is reflected or stored so that the victim browser executes attacker-controlled code.

РЕКОМЕНДОВАНА РЕМЕДІАЦІЯ

Екранувати вивід, увімкнути CSP, заборонити небезпечні DOM API та додати автотести на reflected/stored XSS.

FINDING DOSSIER

HIGH

Outdated library: Django 3.0.7

webapp-01 · requirements.txt:3 · 0a6e7ba698dc3f44

THREAT CLASS	OWASP
VULNERABLE_COMPONENT	A06:2021 · Vulnerable and Outdated Components
SOURCES	POLICY
SCA	MANUAL-VERIFY · SLA 24h
RISK INTERVAL	UNCERTAINTY
6.31-7.92	Висока · H~ 0.849

CLASSIFIER ENSEMBLE

hybrid_weighted_ensemble · context=VULNERABLE_COMPONENT/0.34, llm=SQL_INJECTION/0.16, source=VULNERABLE_COMPONENT/0.75, structural=SQL_INJECTION/0.67, text=VULNERABLE_COMPONENT/0.32

CLASSIFIER DECISION

mode=PRIMARY · final=VULNERABLE_COMPONENT · primary=VULNERABLE_COMPONENT/0.44 · margin=0.27
alternatives=-

RISK PROFILE

thesis-default:v1 · lambda=0.30 · high=7.00 · review=0.50

NORMALIZATION PIPELINE

N1-N6.v1 · N1 -> N2 -> N3 -> N4 -> N5 -> N6

N1
parse_and_sanitize
Sanitized raw finding fields and tenant context.

N2
unify_schema
Canonicalized identifiers, source metadata, and severity scale.

N3
enrich_taxonomy
Prepared lexical and taxonomy-ready features.

N4
contextualize
Derived contextual risk factors from source and content.

N5
prepare_correlation
Prepared provenance bundle with 1 supporting raw finding(s).

N6
finalize
Materialized normalized finding artifact for downstream classification and risk scoring.

ОПИС

Django 3.0.7 has multiple known security fixes including SQL injection and open redirect vulnerabilities patched in later versions.

LLM-ASSISTED SEMANTIC LAYER

Семантичний шар інтерпретує знахідку як VULNERABLE_COMPONENT. Ключовий сигнал формується з опису, контексту SCA та таксономічних прив'язок CVE-2021-35042, CWE-937, CAPEC-468. Рівень довіри моделі становить 0.44, а найближчі альтернативи: SQL_INJECTION=0.17, XSS=0.09. Пріоритетним кроком слід вважати усунення кореневої причини в коді або конфігурації, а не лише компенсаційний контроль.

backend=template · prompt=semantic-assist-v1

ПОЯСНЕННЯ МОДЕЛІ

Ensemble-класифікатор визначив клас VULNERABLE_COMPONENT з довірою 0.44. Агрегатор F поєднав p_txt, p_ctx, p_gnn, p_src і p_llm (text=VULNERABLE_COMPONENT:0.32, context=VULNERABLE_COMPONENT:0.34, structural=SQL_INJECTION:0.67, source=VULNERABLE_COMPONENT:0.75, llm=SQL_INJECTION:0.16). Альтернативні гіпотези: суттєвих альтернатив не зафіксовано. Структурний граф: code graph gnn nodes=22, edges=30, source=synthetic, node_types=FILE:2, MODULE:2, IMPORT:1, FUNCTION:3, CFG_ENTRY:3, CFG_EXIT:3, CFG_STATEMENT:5, SYMBOL:3, edge_types=CONTAINS:5, AST_CHILD:3, AST_BODY:5, AST_IMPORT:1, CFG_ENTRY:3, CFG_FLOW:11, CALLS:2. Ключові ознаки: outdated, library, version. Семантичний шар інтерпретує знахідку як VULNERABLE_COMPONENT. Ключовий сигнал формується з опису, контексту SCA та таксономічних прив'язок CVE-2021-35042, CWE-937, CAPEC-468. Рівень довіри моделі становить 0.44, а найближчі альтернативи: SQL_INJECTION=0.17, XSS=0.09. Пріоритетним кроком слід вважати усунення кореневої причини в коді або конфігурації, а не лише компенсаційний контроль.

ОЦІНКА РИЗИКУ

R_a=6.31 (CVSS=7.5, confidence=0.44); Ĥ=0.849 [Висока невизначеність]; R_comp=7.92 ∈ [6.31, 7.92]; profile=thesis-default:v1; ⚠ рекомендується ручний перегляд

CVE/CWE/CAPEC ENRICHMENT

CVE-2021-35042 · CVE-2021-35042

seed-knowledge

Reference metadata is available via external taxonomy refresh when network access is enabled.

CWE-937 · Using Components with Known Vulnerabilities

A dependency or runtime component contains a published vulnerability that is already known to the public domain.

CAPEC-468 · Exploitation of Vulnerable Components

seed-knowledge

Known-vulnerable libraries or images are leveraged as the practical entry point for compromise.

РЕКОМЕНДОВАНА РЕМЕДІАЦІЯ

Оновити залежність або контейнерний образ, зафіксувати мінімальну безпечну версію, увімкнути dependency audit і контроль SBOM у пайплайні.

FINDING DOSSIER

HIGH

Debug mode enabled in production config

webapp-01 · /config/settings.py:12 · 57cd724c9e8c5333

THREAT CLASS
SECURITY_MISCONFIGURATION

OWASP
A05:2021 · Security Misconfiguration

SOURCES
SAST CONFIG

POLICY
MANUAL-VERIFY · SLA 24h

RISK INTERVAL
6.01-7.43

UNCERTAINTY
Висока · H~ 0.790

CLASSIFIER ENSEMBLE

hybrid_weighted_ensemble · context=SECURITY_MISCONFIGURATION/0.53, llm=SSRF/0.22, source=SECURITY_MISCONFIGURATION/0.51, structural=SECURITY_MISCONFIGURATION/0.78, text=SECURITY_MISCONFIGURATION/0.44

CLASSIFIER DECISION

mode=PRIMARY · final=SECURITY_MISCONFIGURATION · primary=SECURITY_MISCONFIGURATION/0.52 · margin=0.41
alternatives=-

RISK PROFILE

thesis-default:v1 · lambda=0.30 · high=7.00 · review=0.50

NORMALIZATION PIPELINE

N1-N6.v1 · N1 -> N2 -> N3 -> N4 -> N5 -> N6

N1
parse_and_sanitize
Sanitized raw finding fields and tenant context.

N2
unify_schema
Canonicalized identifiers, source metadata, and severity scale.

N3
enrich_taxonomy
Prepared lexical and taxonomy-ready features.

N4
contextualize
Derived contextual risk factors from source and content.

N5
prepare_correlation
Prepared provenance bundle with 2 supporting raw finding(s).

N6
finalize
Materialized normalized finding artifact for downstream classification and risk scoring.

ОПИС

LLM-ASSISTED SEMANTIC LAYER

Семантичний шар інтерпретує знахідку як SECURITY_MISCONFIGURATION. Ключовий сигнал формується з опису, контексту SAST+1 та таксономічних прив'язок CWE-16, CAPEC-13. Рівень довіри моделі становить 0.52, а найближчі альтернативи: SSRF=0.11, VULNERABLE_COMPONENT=0.08. Пріоритетним кроком слід вважати усунення кореневої причини в коді або конфігурації, а не лише компенсаційний контроль.

backend=template · prompt=semantic-assist-v1

ПОЯСНЕННЯ МОДЕЛІ

Ensemble-класифікатор визначив клас SECURITY_MISCONFIGURATION з довірою 0.52. Агрегатор F поєднав p_txt, p_ctx, p_gnn, p_src і p_llm (text=SECURITY_MISCONFIGURATION:0.44, context=SECURITY_MISCONFIGURATION:0.53, structural=SECURITY_MISCONFIGURATION:0.78, source=SECURITY_MISCONFIGURATION:0.51, llm=SSRF:0.22). Альтернативні гіпотези: суттєвих альтернатив не зафіксовано. Структурний граф: code graph gnn nodes=15, edges=18, source=synthetic, node_types=FILE:2, MODULE:2, IMPORT:1, FUNCTION:2, CFG_ENTRY:2, CFG_EXIT:2, CFG_STATEMENT:2, SYMBOL:2, edge_types=CONTAINS:4, AST_CHILD:2, AST_BODY:2, AST_IMPORT:1, CFG_ENTRY:2, CFG_FLOW:6, CALLS:1. Ключові ознаки: debug, exposed. Семантичний шар інтерпретує знахідку як SECURITY_MISCONFIGURATION. Ключовий сигнал формується з опису, контексту SAST+1 та таксономічних прив'язок CWE-16, CAPEC-13. Рівень довіри моделі становить 0.52, а найближчі альтернативи: SSRF=0.11, VULNERABLE_COMPONENT=0.08. Пріоритетним кроком слід вважати усунення кореневої причини в коді або конфігурації, а не лише компенсаційний контроль.

ОЦІНКА РИЗИКУ

R_a=6.01 (CVSS=5.3, confidence=0.52); Ĥ=0.790 [Висока невизначеність]; R_comp=7.43 ∈ [6.01, 7.43]; profile=thesis-default:v1; ⚠ рекомендується ручний перегляд

CVE/CWE/CAPEC ENRICHMENT

CWE-16 · Configuration

seed-knowledge

Improper or insecure configuration choices expose the application to avoidable runtime risk.

CAPEC-13 · Configuration Abuse

seed-knowledge

The attacker exploits insecure runtime or deployment settings that broaden the application's attack surface.

РЕКОМЕНДОВАНА РЕМЕДІАЦІЯ

Застосувати hardened baseline конфігурацій, закрити debug/verbose режими, додати security headers та policy-as-code перевірки в CI/CD.

FINDING DOSSIER

MEDIUM

Missing HttpOnly flag on session cookie

webapp-01 · /app/middleware/session.py:23 · 3762da33148057fb

THREAT CLASS
UNKNOWN

OWASP
A05:2021 · Security Misconfiguration

SOURCES
SAST

POLICY
MANUAL-VERIFY · SLA 24h

RISK INTERVAL
5.40-6.87

UNCERTAINTY
Висока · H~ 0.909

CLASSIFIER ENSEMBLE

hybrid_weighted_ensemble · context=BROKEN_ACCESS_CONTROL/0.41, llm=SECURITY_MISCONFIGURATION/0.18, source=SECURITY_MISCONFIGURATION/0.28, structural=XSS/0.30, text=SECURITY_MISCONFIGURATION/0.21

CLASSIFIER DECISION

mode=ABSTAIN_UNKNOWN · final=UNKNOWN · primary=SECURITY_MISCONFIGURATION/0.37 · margin=0.23 · abstain=component_disagreement

alternatives=BROKEN_ACCESS_CONTROL=0.14

RISK PROFILE

thesis-default:v1 · lambda=0.30 · high=7.00 · review=0.50

NORMALIZATION PIPELINE

N1
parse_and_sanitize
Sanitized raw finding fields and tenant context.

N2
unify_schema
Canonicalized identifiers, source metadata, and severity scale.

N3
enrich_taxonomy
Prepared lexical and taxonomy-ready features.

N4
contextualize
Derived contextual risk factors from source and content.

N5
prepare_correlation
Prepared provenance bundle with 1 supporting raw finding(s).

N6
finalize
Materialized normalized finding artifact for downstream classification and risk scoring.

ОПИС

Session cookie is set without HttpOnly and Secure flags, making it accessible via JavaScript and transmittable over unencrypted connections.

LLM-ASSISTED SEMANTIC LAYER

Семантичний шар фіксує абстенцію класифікатора: фінальне рішення переведено в UNKNOWN через розбіжність між ensemble-підмодулями. Провідна гіпотеза лишається SECURITY_MISCONFIGURATION з довірою 0.37, а найближчі альтернативи: BROKEN_ACCESS_CONTROL=0.14. Такий запис слід передати на ручну верифікацію перед автоматичною ескалацією або ticketing-процедурою.
backend=template · prompt=semantic-assist-v1

ПОЯСНЕННЯ МОДЕЛІ

Ensemble-класифікатор не зафіксував остаточний клас і перевів запис у UNKNOWN. Провідна гіпотеза: SECURITY_MISCONFIGURATION з довірою 0.37; поріг спрацював через суттєва розбіжність між підмодулями ensemble при margin=0.23. Альтернативи: BROKEN_ACCESS_CONTROL=0.14. Агрегатор F поєднав p_txt, p_ctx, p_gnn, p_src і p_llm (text=SECURITY_MISCONFIGURATION:0.21, context=BROKEN_ACCESS_CONTROL:0.41, structural=XSS:0.30, source=SECURITY_MISCONFIGURATION:0.28, llm=SECURITY_MISCONFIGURATION:0.18). Структурний граф: code graph gnn nodes=16, edges=20, source=synthetic, node_types=FILE:2, MODULE:2, IMPORT:1, FUNCTION:2, CFG_ENTRY:2, CFG_EXIT:2, CFG_STATEMENT:3, SYMBOL:2, edge_types=CONTAINS:4, AST_CHILD:2, AST_BODY:3, AST_IMPORT:1, CFG_ENTRY:2, CFG_FLOW:7, CALLS:1. Ключові ознаки: загальний контекст. Семантичний шар фіксує абстенцію класифікатора: фінальне рішення переведено в UNKNOWN через розбіжність між ensemble-підмодулями. Провідна гіпотеза лишається SECURITY_MISCONFIGURATION з довірою 0.37, а найближчі альтернативи: BROKEN_ACCESS_CONTROL=0.14. Такий запис слід передати на ручну верифікацію перед автоматичною ескалацією або ticketing-процедурою.

ОЦІНКА РИЗИКУ

R_a=5.40 (CVSS=5.4, confidence=0.37); Ĥ=0.909 [Висока невизначеність]; R_comp=6.87 ∈ [5.40, 6.87]; profile=thesis-default:v1; ⚠ рекомендується ручний перегляд; classifier_abstained=true; primary_hypothesis=SECURITY_MISCONFIGURATION; abstain_reason=component_disagreement

CVE/CWE/CAPEC ENRICHMENT

CWE-16 · Configuration

seed-knowledge

Improper or insecure configuration choices expose the application to avoidable runtime risk.

CAPEC-13 · Configuration Abuse

seed-knowledge

The attacker exploits insecure runtime or deployment settings that broaden the application's attack surface.

РЕКОМЕНДОВАНА РЕМЕДІАЦІЯ

Передати знахідку на ручну triage-валідацію, розширити навчальні дані та за потреби додати новий клас загроз у модель.

FINDING DOSSIER

MEDIUM

Missing X-Frame-Options header

webapp-02 · /app/middleware/headers.py · a1068d5ac34d0a50

THREAT CLASS
SECURITY_MISCONFIGURATION

OWASP
A05:2021 · Security Misconfiguration

SOURCES
SAST

POLICY
MANUAL-VERIFY · SLA 24h

RISK INTERVAL
4.68-6.01

UNCERTAINTY
Висока · H~ 0.947

CLASSIFIER ENSEMBLE

hybrid_weighted_ensemble · context=SECURITY_MISCONFIGURATION/0.50, llm=SECURITY_MISCONFIGURATION/0.17, source=SECURITY_MISCONFIGURATION/0.34, structural=SECURITY_MISCONFIGURATION/0.79, text=SECURITY_MISCONFIGURATION/0.31

CLASSIFIER DECISION

mode=PRIMARY · final=SECURITY_MISCONFIGURATION · primary=SECURITY_MISCONFIGURATION/0.32 · margin=0.19
alternatives=-

RISK PROFILE

thesis-default:v1 · lambda=0.30 · high=7.00 · review=0.50

NORMALIZATION PIPELINE

N1-N6.v1 · N1 -> N2 -> N3 -> N4 -> N5 -> N6

N1
parse_and_sanitize
Sanitized raw finding fields and tenant context.

N2
unify_schema
Canonicalized identifiers, source metadata, and severity scale.

N3
enrich_taxonomy
Prepared lexical and taxonomy-ready features.

N4
contextualize
Derived contextual risk factors from source and content.

N5
prepare_correlation
Prepared provenance bundle with 1 supporting raw finding(s).

N6
finalize
Materialized normalized finding artifact for downstream classification and risk scoring.

ОПИС

Application does not set X-Frame-Options or Content-Security-Policy frame-ancestors, allowing clickjacking attacks.

LLM-ASSISTED SEMANTIC LAYER

Семантичний шар інтерпретує знахідку як SECURITY_MISCONFIGURATION. Ключовий сигнал формується з опису, контексту SAST та таксономічних прив'язок CWE-16, CAPEC-13. Рівень довіри моделі становить 0.32, а найближчі альтернативи: BROKEN_ACCESS_CONTROL=0.13, XSS=0.12. Пріоритетним кроком слід вважати усунення кореневої причини в коді або конфігурації, а не лише компенсаційний контроль.

ПОЯСНЕННЯ МОДЕЛІ

Ensemble-класифікатор визначив клас SECURITY_MISCONFIGURATION з довірою 0.32. Агрегатор F поєднав p_txt, p_ctx, p_gnn, p_src і p_llm (text=SECURITY_MISCONFIGURATION:0.31, context=SECURITY_MISCONFIGURATION:0.50, structural=SECURITY_MISCONFIGURATION:0.79, source=SECURITY_MISCONFIGURATION:0.34, llm=SECURITY_MISCONFIGURATION:0.17). Альтернативні гіпотези: суттєвих альтернатив не зафіксовано. Структурний граф: code graph gnn nodes=15, edges=18, source=synthetic, node_types=FILE:2, MODULE:2, IMPORT:1, FUNCTION:2, CFG_ENTRY:2, CFG_EXIT:2, CFG_STATEMENT:2, SYMBOL:2, edge_types=CONTAINS:4, AST_CHILD:2, AST_BODY:2, AST_IMPORT:1, CFG_ENTRY:2, CFG_FLOW:6, CALLS:1. Ключові ознаки: headers. Семантичний шар інтерпретує знахідку як SECURITY_MISCONFIGURATION. Ключовий сигнал формується з опису, контексту SAST та таксономічних прив'язок CWE-16, CAPEC-13. Рівень довіри моделі становить 0.32, а найближчі альтернативи: BROKEN_ACCESS_CONTROL=0.13, XSS=0.12. Пріоритетним кроком слід вважати усунення кореневої причини в коді або конфігурації, а не лише компенсаційний контроль.

ОЦІНКА РИЗИКУ

R_a=4.68 (CVSS=4.3, confidence=0.32); Ĥ=0.947 [Висока невизначеність]; R_comp=6.01 ∈ [4.68, 6.01]; profile=thesis-default:v1; ⚠ рекомендується ручний перегляд

CVE/CWE/CAPEC ENRICHMENT

CWE-16 · Configuration

seed-knowledge

Improper or insecure configuration choices expose the application to avoidable runtime risk.

CAPEC-13 · Configuration Abuse

seed-knowledge

The attacker exploits insecure runtime or deployment settings that broaden the application's attack surface.

РЕКОМЕНДОВАНА РЕМЕДІАЦІЯ

Застосувати hardened baseline конфігурацій, закрити debug/verbose режими, додати security headers та policy-as-code перевірки в CI/CD.

Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Дмитро МИКУЛЯК

Співавтор:

Назва: Інтелектуальна комп'ютерна система автоматичного виявлення вразливостей вебзастосунків та класифікації загроз

Експерт: Олег САВЕНКО

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:4.86%

Коефіцієнт подібності 2:1.56%

Мікропробіли: 0

Заміна букв: 15

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2026-04-27 08:21:49.0

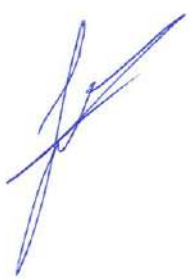
Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:



2026-04-27 Доцент Андрій Нічепорук

Дата експерт

Anti-Plagiarism (http://ap.km.ua) v-15.701

Максимальне співпадіння з одним документом 0.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 17%**

	Документ		Сумарний збіг по Базі Даних	
ID: 270695 Назва: МКР Інтелектуальна комп'ютерна система автоматичного виявлення вразливостей вебзастосунків та класифікації загроз Додано в БД: 2026-04-27 Автора: Дмитро МИКУЛЯК Керівники: Олег САВЕНКО Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	177616	1328	2666 (2%)	43 (3%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

Здобувач: Дмитро МИКУЛЯК

Тема: Інтелектуальна комп'ютерна система автоматичного виявлення вразливостей вебзастосунків та класифікації загроз

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи магістра:

Кількість листів креслень _–_; кількість сторінок записки 118

1. Короткий зміст роботи та прийнятих рішень У роботі запропоновано метод автоматичної класифікації безпекових знахідок на основі гібридного класифікаційного контуру, що поєднує текстові, структурні, контекстні, джерельні, графові та трансформерно-семантичні ознаки.

2. Висновок про відповідність роботи дипломному завданню Кваліфікаційна робота магістра відповідає виданому завданню

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи:

У вступі обґрунтовано актуальність теми, визначено об'єкт, предмет, мету, завдання, методи дослідження, наукову новизну та практичне значення роботи.

У першому розділі проведено аналіз сучасних підходів до виявлення вразливостей вебзастосунків, класифікації кіберзагроз, стандартів безпеки та методів застосування штучного інтелекту в кібербезпеці.

У другому розділі розроблено архітектуру, модель та метод інтелектуальної комп'ютерної системи автоматичного виявлення вразливостей вебзастосунків і класифікації загроз.

У третьому розділі розроблено алгоритмічне забезпечення системи, зокрема алгоритми збору, нормалізації, класифікації, оцінювання ризику та ранжування загроз.

У четвертому розділі здійснено програмну реалізацію прототипу системи, проведено експериментальну верифікацію та оцінено ефективність запропонованих рішень.

У висновках узагальнено основні результати роботи, підтверджено досягнення поставленої мети та визначено напрями подальшого розвитку системи

4. Позитивні сторони роботи: розроблена інтелектуальна комп'ютерна система дозволяє автоматизувати процес виявлення вразливостей вебзастосунків та класифікації кіберзагроз, забезпечити інтеграцію результатів різних засобів аналізу безпеки, зменшити кількість дубльованих і хибнопозитивних повідомлень, підвищити обґрунтованість ризик-орієнтованого ранжування вразливостей,

скоротити час первинного аналізу результатів сканування, а також надати фахівцям із кібербезпеки структуровані звіти для прийняття рішень щодо пріоритезації усунення загроз.

5. Негативні сторони роботи: _____

_____

6. Оцінка графічного оформлення та пояснювальної записки роботи: – _____

_____

_____

_____

_____

_____

7. Відгук про роботу в цілому: В загальному робота виконана на високому рівні. _____

_____

_____

_____

_____

8. Інші зауваження: – _____

_____

_____

_____

_____

_____

9. Оцінка кваліфікаційної роботи магістра:

Розглянувши позитивні та негативні сторони представленої кваліфікаційної роботи магістра вважаю, що робота заслуговує оцінки «відмінно» 95.00 (А) _____

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) ____ д.т.н., професор, Мартинюк В.В., професор кафедри автоматизації, комп'ютерно-інтегрованих технологій та робототехніки _____

"1 травня" _____2026р. _____

Зав. кафедри КІІС
д-р. філософії Ользі ПАВЛОВІЙ

Дмитро МИКУЛЯК

ПІБ здобувача вищої освіти

ФІТ, 2 курсу, групи КІ2м-24-2

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів академічної відповідальності, ознайомлений (а). Про використання спеціалізованих програмних засобів (СПЗ) StrikePlagiarism та Anti-Plagiarism для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений (а). Надаю університету право на передачу моєї роботи для обробки та збереження в базах даних СПЗ і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються СПЗ.

Також надаю свою згоду на обробку й збереження університетом моєї роботи в Інституційному репозитарії Хмельницького національного університету.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

1 травня 2026 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Назва кваліфікаційної роботи Інтелектуальна комп'ютерна система автоматичного виявлення вразливостей вебзастосунків та класифікація загроз
Автор Дмитро МИКУЛЯК
Освітня програма Комп'ютерна інженерія та програмування
Рівень вищої освіти другий (магістерський)
Спеціальність 123 Комп'ютерна інженерія
Науковий керівник: д. тех. наук, професор Олег САВЕНКО

На основі аналізу кваліфікаційної роботи на дотримання вимог академічної доброчесності (у т.ч. відсутності ознак академічного плагіату) з урахуванням результатів перевірки роботи спеціалізованим програмним засобом(ами) комісія зробила такий висновок:

№	Висновок	Позначка про відповідність
1	Ознаки академічного плагіату	
1.1	Запозичення, виявлені в роботі, є законними і не є академічним плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних, якщо потрібно). Робота приймається до захисту.	відповідає
1.2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована.	
1.3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота може бути допущена до захисту після того як буде відкоригована та доопрацьована і успішно пройде повторну перевірку на академічний плагіат.	
1.4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
2	Інші види порушень академічної доброчесності	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) усі запозичення фрагментарні, або мають належним чином оформленні посилання;

2) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з джерелами на один фрагмент речення;

3) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

4) значна частина знайденого плагіату відноситься до списку використаних джерел

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 4,86% та системою Anti-Plagiarism складає 0%, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

30.04.2026

Завідувач кафедри _____ Ольга ПАВЛОВА
Підпис Ім'я, ПРІЗВИЩЕ

Гарант освітньої програми _____ Олег САВЕНКО
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи _____ Олег САВЕНКО
Підпис Ім'я, ПРІЗВИЩЕ