МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЧЕРНІГІВСЬКА ПОЛІТЕХНІКА»
WIN-WIN EDIH



Co-funded by
the European Union



Співфінансується
Європейським Союзом



ВІЛЬНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ В ОСВІТІ, НАУЦІ ТА БІЗНЕСІ

FREE/OPEN SOURCE SOFTWARE IN EDUCATION, SCIENCE & BUSINESS (FOSS 2026)

Матеріали II Міжнародної науково-практичної конференції
01–02 жовтня 2026 р.
м. Чернігів

Чернігів 2026

УДК 004.49:[37+001+334.722]
В41

Рекомендовано до друку вченою радою Національного університету «Чернігівська політехніка» (протокол № 12 від 02.11.2026 р.)

Вільне програмне забезпечення в освіті, науці та бізнесі : матеріали ІІ Міжнар. В41 наук.-практ. конф. (м. Чернігів, 01–02 жовтня 2026 р.) / відп. за вип. Д. Б. Мехед. Чернігів : НУ «Чернігівська політехніка», 2026. 156 с.

ISBN 978-617-8584-09-2

Збірник містить матеріали ІІ Міжнародної науково-практичної конференції «Вільне програмне забезпечення в освіті, науці та бізнесі» (FOSS 2026), що відбулася онлайн 01–02 жовтня 2026 року в Національному університеті «Чернігівська політехніка». Розглянуто питання використання вільного та відкритого програмного забезпечення в освіті, науці, бізнесі, ІТ-інфраструктурі, кібербезпеці, аналізі даних і штучному інтелекті. Матеріали подано українською та англійською мовами.

Для науковців, викладачів, здобувачів освіти, фахівців ІТ-галузі та всіх, хто цікавиться вільним програмним забезпеченням.

Матеріали подано в авторській редакції. Відповідальність за зміст, достовірність фактів, цитат і посилань несуть автори.

Конференцію проведено в межах проєкту WIN-WIN EDIH (грантова угода № 101191647), співфінансованого Європейським Союзом у межах програми Digital Europe. Висловлені погляди та думки належать виключно автору(ам) і не обов'язково відображають позицію Європейського Союзу чи Європейської Комісії.

Відповідальний за випуск: Мехед Дмитро Борисович

ISBN 978-617-8584-09-2

© НУ «Чернігівська політехніка», 2026

ОРГАНІЗАЦІЙНИЙ КОМІТЕТ

Голова організаційного комітету:
Анатолій Приступа – проректор з наукової роботи, к.т.н., доцент.
Заступник голови:
Володимир Базилевич – директор ННІ електронних та інформаційних технологій, к.е.н., доцент.
Члени організаційного комітету:
Ірина Білоус – завідувач кафедри інформаційних технологій та програмної інженерії, к.т.н., доцент.
Андрій Роговенко – завідувач кафедри інформаційних та комп'ютерних систем, к.т.н., доцент.
Андрій Хижняк – старший викладач кафедри інформаційних та комп'ютерних систем.
Дмитро Мехед – доцент кафедри кібербезпеки та математичного моделювання, к.п.н., доцент.
Катерина Максьом – завідувачка кафедри соціальної роботи, к.п.н., доцент.
Євген Патлань – Engineering Manager, Percona.
Юлія Дайнеко – голова правління Chernihiv.IT.
Влад Носовець – координатор Chernihiv.IT.

Відповідальний координатор конференції:
Мехед Дмитро Борисович, тел. (063) 6908000, e-mail: d.mekhed@stu.cn.ua https://feit.stu.cn.ua/foss/

*За зміст матеріалів, викладених у тезах доповідей, персональну відповідальність несуть автори

ЗМІСТ

Абдразаков І. В. ЕКОНОМІЧНА ОЦІНКА ПЕРЕХОДУ МАЛИХ І СЕРЕДНІХ ПІДПРИЄМСТВ НА ВІДКРИТЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ МОДЕЛІ СУКУПНОЇ ВАРТОСТІ ВОЛОДІННЯ 6

Бакало В. І. ПЕРЕТВОРЕННЯ ЯКІСНОЇ ОЦІНКИ КІБЕРРИЗИКІВ У КІЛЬКІСНУ ЗАСОБАМИ ВІЛЬНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 9

Березинський І. В., Чурілова І. С. ВІДКРИТІСТЬ «OPEN SOURCE AI». АВТОРСЬКЕ ПРАВО ТА ПРИВАТНІСТЬ 13

Величко В. В. РЕАЛІЗАЦІЯ ПРИНЦИПІВ АРХІТЕКТУРИ ZERO TRUST У VPN-ШЛЮЗАХ НА ОСНОВІ ДИНАМІЧНОЇ ОЦІНКИ КОНТЕКСТУ СЕСІЙ 17

Велігорський Б. О., Хоменко М. А. ОПТИМІЗАЦІЯ КЕРУВАННЯ MICROGRID ЗА ДОПОМОГОЮ БІБЛІОТЕКИ CVXPY 19

Гричина О. О. FILE NOTIFICATION ATTACK — НОВА АТАКА ЧЕРЕЗ БІЧНИЙ КАНАЛ ПРОТИ ВСІХ ОПЕРАЦІЙНИХ СИСТЕМ / FILE NOTIFICATION ATTACK: A NEW SIDE-CHANNEL ATTACK AGAINST ALL MAJOR OPERATING SYSTEMS 23

Денісова О. О. ДОСЛІДЖЕННЯ ВІДТВОРЮВАНИХ ДОСЛІДНИЦЬКИХ ПАЙПЛАЙНІВ ЄВРОПЕЙСЬКОГО ДОСЛІДНИЦЬКОГО ПРОСТОРУ 26

Живило Є. О. АПАРАТНИЙ МЕТОД АДАПТИВНОЇ НЕЙТРАЛІЗАЦІЇ РАДІОЕЛЕКТРОННИХ ЗАВАД У ЗАСОБАХ ТЗІ В УМОВАХ РАДІОЕЛЕКТРОННОГО ПРИДУШЕННЯ 30

Калашнікова Ю. В. ДЕТЕРМІНОВАНА АРХІТЕКТУРА ЗАСОБІВ ТЗІ ДЛЯ МІНІМІЗАЦІЇ ВІКНА ВРАЗЛИВОСТІ ОКІ 34

Клочко К. М., Роговенко А. І. РОЗРОБКА ТА ДОСЛІДЖЕННЯ ДЕЦЕНТРАЛІЗОВАНОЇ ВЕБСИСТЕМИ КАТАЛОГІЗАЦІЇ ЕЛЕКТРОННИХ ПУБЛІКАЦІЙ ЗАСОБАМИ ELIXIR 39

Кондратюк Д. С., Казимир В. В. СЕМАНТИЧНИЙ КОНТРАКТ ПОДІЙ У ВЗАЄМОДІЇ IoT ТА ERP СИСТЕМ / A SEMANTIC EVENT CONTRACT FOR IoT–ERP INTERACTION 42

Красько Т. О. ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ БРОНЮВАННЯ СТОЛИКІВ У РЕСТОРАНАХ ЗАСОБАМИ МОБІЛЬНОГО ЗАСТОСУНКУ ANDROID 45

Литвин С. О. ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ ВИЯВЛЕННЯ АНОМАЛІЙ У ЛОГАХ ПРОГРАМНИХ СИСТЕМ 47

Максимов О. В., Лисенко Д. Е. ОЦІНЮВАННЯ ВІДКРИТИХ ЛОКАЛЬНИХ МОВНИХ МОДЕЛЕЙ ДЛЯ ГЕНЕРАЦІЇ ЮНІТ-ТЕСТІВ У PYTHON-ПРОЄКТАХ 51

Максьом К. В. ЦИФРОВА МЕДІАГРАМОТНІСТЬ У ДОБУ ГЕНЕРАТИВНОГО ШІ: ПОТЕНЦІАЛ ВІДКРИТИХ ІНСТРУМЕНТІВ / DIGITAL MEDIA LITERACY IN THE AGE OF GENERATIVE AI: THE POTENTIAL OF OPEN-SOURCE TOOLS 54

Панчук М. О. АПАРАТНО-ОРІЄНТОВАНІ МЕТОДИ ЗАБЕЗПЕЧЕННЯ КІБЕРБЕЗПЕКИ ГЕТЕРОГЕННИХ СИСТЕМ PEP НА БАЗІ SoC FPGA 57

Перепечай А. С., Лисенко Д. Е. ІНТЕЛЕКТУАЛЬНА СИСТЕМА ОЦІНЮВАННЯ ТА РАНЖУВАННЯ ПРІОРИТЕТНОСТІ СОЦІАЛЬНИХ ЗВЕРНЕНЬ ГРОМАДЯН 61

Плахтій М. О. НАНОСТРУКТУРНА АРХІТЕКТУРА КВАНТОВИХ НАКОПИЧУВАЧІВ 65

Пода О. Л. АРХІТЕКТУРНО-ІНЖЕНЕРНІ РІШЕННЯ ЗАБЕЗПЕЧЕННЯ МОНІТОРИНГУ ШІ-АСИСТЕНТА ТА БЕЗПЕКИ ДАНИХ В МОБІЛЬНІЙ ІНФОРМАЦІЙНІЙ СИСТЕМІ ПСИХОЕМОЦІЙНОЇ ПІДТРИМКИ / ARCHITECTURE AND EMPIRICAL SECURITY TESTING OF A MOBILE PSYCHO-EMOTIONAL SUPPORT APPLICATION WITH AN AI ASSISTANT 68

Синенко М. А. ВИКОРИСТАННЯ ВІЛЬНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПРИ ВИВЧЕННІ СТЕГАНОГРАФІЇ 72

Содиль М. О. ВПРОВАДЖЕННЯ КОНЦЕПЦІЇ ZERO TRUST В ІНФРАСТРУКТУРУ ПІДПРИЄМСТВА 75

Фесенко Т. М. БАГАТОКАНАЛЬНА АРХІТЕКТУРА РАДІОЧАСТОТНОГО МОНІТОРИНГУ ПОТЕНЦІЙНИХ ТКВІ З ВИКОРИСТАННЯМ БПЛА 78

Чопорова О. В., Чопоров С. В. ВИКОРИСТАННЯ 3D-МОДЕЛЮВАННЯ ТА АЛГОРИТМІВ НЕЙРОННИХ МЕРЕЖ В НАВЧАЛЬНИХ ПРОЄКТАХ ІНЖЕНЕРНО-ТЕХНІЧНОГО СПРЯМУВАННЯ / USE OF 3D MODELING AND NEURAL NETWORK ALGORITHMS IN EDUCATIONAL PROJECTS OF ENGINEERING AND TECHNICAL DIRECTION 82

Чуб М. В. ВИЯВЛЕННЯ ТА ПРОТИДІЯ АДАПТИВНИМ ВЕБ-АТАКАМ ЗА ДОПОМОГОЮ МОДЕЛЕЙ ГЛИБОКОГО НАВЧАННЯ 86

Шевченко Ю. О. ПОРІВНЯЛЬНЕ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ POSTGRESQL ТА INFLUXDB ДЛЯ ЗБЕРІГАННЯ ДАНИХ ФІЗИЧНОЇ АКТИВНОСТІ / COMPARATIVE STUDY OF POSTGRESQL AND INFLUXDB PERFORMANCE FOR STORING PHYSICAL ACTIVITY DATA 89

Al-Akwaa K. H. M. A. OPEN-SOURCE DATA SCIENCE FOR SUSTAINABLE DEVELOPMENT: A REPRODUCIBLE ANALYTICAL FRAMEWORK 93

Borysenko O. A. AN ARCHITECTURAL APPROACH TO LLM-BASED ROOT CAUSE ANALYSIS FOR CONTAINER ORCHESTRATION INCIDENTS 100

Burmaka I. LOCAL, OPEN, AND ENERGY-EFFICIENT: HARNESSING X86 NPUS FOR EVERYDAY AI TASKS 102

Byvoino T. P., Byvoino P. G. SIMULATION MODELING AND STATISTICAL ANALYSIS OF THE EFFICIENCY OF AN EMERGENCY MEDICAL SERVICE STATION 106

Didusenko O. A., Bula Y. I., Shymchuk G. V. REPRODUCIBLE FOSS FRAMEWORK FOR RESILIENCE EVALUATION OF CONTAINERIZED DISTRIBUTED SYSTEMS 109

Didusenko O. A., Bula Y. I., Shymchuk G. V. REPRODUCIBLE FOSS TESTBED FOR FAULT-TOLERANT MULTIPATH ROUTING IN IP NETWORKS 112

Ebirim V., Jadama A. F. R. RESILIENT DIGITAL EDUCATION IN DISRUPTED ENVIRONMENTS: THE CCDS APPROACH 115

Fraya DIGITAL TRANSFORMATION OF BUSINESS THROUGH FREE AND OPEN SOURCE SOFTWARE: OPPORTUNITIES, BARRIERS, AND STRATEGIC PATHWAYS 119

Kala P. RUN MONGODB YOUR WAY: A PRACTICAL OPEN-SOURCE STACK FOR SELF-MANAGED DEPLOYMENTS 121

Lukin O. A MODERN C++ KMIP CLIENT BASED ON A REUSABLE PROTOCOL LIBRARY 124

Mekhed D. B. DEVELOPING STUDENTS' CYBER HYGIENE SKILLS THROUGH THE USE OF FREE SPECIALIZED SOFTWARE 127

Nahornyi B. I., Lysenok D. V., Savenko A. O. AN OPEN BROWSER-BASED CO-SIMULATOR OF STM32 FIRMWARE AND ELECTRICAL CIRCUITS AS A FREE ALTERNATIVE TO PROTEUS 130

Romanenko M. V. FOSS APPROACH IN MOBILE DEVELOPMENT: CREATING AN OPEN-SOURCE ANDROID APPLICATION WITH THE INTEGRATION OF READY-MADE LIBRARIES FOR STYLISTIC RECOMMENDATIONS 134

Shevchenko N. A., Kibitkin D. S., Shymchuk G. V. REPRODUCIBLE OPEN-SOURCE BENCHMARK OF CONCEPT-DRIFT DETECTORS FOR STREAMING REGRESSION 137

Shevchenko N. A., Kibitkin D. S., Shymchuk G. V. INTERPRETABLE OPEN-SOURCE MACHINE LEARNING FOR NATURAL GAS CONSUMPTION FORECASTING WITH WEATHER AND LAG FEATURES 140

Shivashankar C. ARCHITECTING DATABASE DEPLOYMENTS ON KUBERNETES: PATTERNS, TRADE-OFFS AND EXPERIENCES 143

Shymchuk G. V., Yatskiv P. O. DETECTION OF ANOMALOUS STATES IN IP NETWORKS USING MACHINE LEARNING AND OPEN-SOURCE SOFTWARE TOOLS 146

Shymchuk G. V., Yatskiv P. O. OPEN-SOURCE SOFTWARE ENVIRONMENT FOR MODELING FAULT-TOLERANT MULTIPATH ROUTING IN IP NETWORKS 149

Silva F. HOSTING SOMEONE ELSE'S APP: IFRAME CHOREOGRAPHY IN PMM 152

Šupić A. ZERO-DOWNTIME MIGRATION WITH PERCONA CLUSTERSYNC FOR MONGODB 154

Абдразаков І. В.
Національний університет «Чернігівська політехніка», кафедра економіки, обліку і оподаткування, м. Чернігів, Україна
Контактний автор: Ігор Абдразаков, e-mail: iabdrazakov@stu.cn.ua, ORCID: 0009-0008-5831-7525

ЕКОНОМІЧНА ОЦІНКА ПЕРЕХОДУ МАЛИХ І СЕРЕДНІХ ПІДПРИЄМСТВ НА ВІДКРИТЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ МОДЕЛІ СУКУПНОЇ ВАРТОСТІ ВОЛОДІННЯ

Анотація. Досліджено економічну доцільність переходу малих і середніх підприємств на відкрите програмне забезпечення. Запропоновано модель сукупної вартості володіння (TCO), яка, крім ліцензійних та інфраструктурних витрат, враховує витрати на міграцію, навчання персоналу та втрати продуктивності в період адаптації. Модель апробовано на умовних підприємствах з 20 та 50 робочими місцями, що переходять з Microsoft 365 на LibreOffice та Nextcloud. За тарифом Business Standard перехід окупається менш ніж за рік, а п'ятирічна економія становить 53,4 % та 63,9 % відповідно. Порівняно з тарифом Business Basic перехід для невеликого підприємства не окупається, тобто ефект відкритого ПЗ визначається рівнем пропрієтарного тарифу та масштабом підприємства.

Ключові слова: відкрите програмне забезпечення; сукупна вартість володіння; малі та середні підприємства; цифрова трансформація; економічна ефективність.

Вступ

Повномасштабна війна посилила тиск на малі та середні підприємства (МСП), для яких цифровізація стала водночас умовою виживання та джерелом додаткових витрат. Розвиток цифрового сектору нарощує ресурсний потенціал економіки, проте водночас стає середовищем становлення нових форм нерівності [1], зокрема між великими компаніями та МСП, для яких валютні ліцензійні платежі є відчутним навантаженням. Відкрите програмне забезпечення (FOSS) часто сприймають як безкоштовну альтернативу, однак підприємство бере на себе розгортання, адміністрування, навчання персоналу та тимчасову втрату продуктивності, і без оцінки цих витрат рішення про міграцію ухвалюються інтуїтивно.

Мета дослідження – розробити модель сукупної вартості володіння (TCO) для оцінки переходу МСП на відкрите офісне та колаборативне ПЗ і визначити умови, за яких такий перехід є економічно доцільним.

Матеріали та методи

Об'єктом моделювання є умовне підприємство сфери послуг з офісним пакетом, файловим сховищем, спільним редагуванням документів, корпоративною поштою та внутрішніми комунікаціями. Базовий сценарій передбачає використання Microsoft 365 Business Standard with Copilot (з Teams) за 23,5 дол. США за користувача на місяць при річній підписці; для аналізу чутливості розглянуто тариф Business Basic за 7 дол. США, який містить лише веб- та мобільні версії застосунків [2]. Альтернативний сценарій передбачає перехід на LibreOffice 26.8 [3] та Nextcloud Server 36 [4] з модулями Nextcloud Office і Nextcloud Talk на орендованому сервері (Ubuntu Server 24.04 LTS, 4 vCPU, 8 ГБ RAM, 200 ГБ SSD) у дата-центрі ЄС; корпоративну пошту винесено до хостинг-провайдера.

Сукупну вартість володіння на горизонті T років розраховано за формулою:

$$TCO(T) = C_{впр} + \Sigma(t = 1 \ldots T)\,(C_{ліц,t} + C_{інф,t} + C_{адм,t}),$$

де $C_{впр}$ – одноразові витрати на впровадження (розгортання та міграція даних, навчання персоналу, втрати продуктивності в період адаптації); $C_{ліц,t}$, $C_{інф,t}$, $C_{адм,t}$ – річні витрати на ліцензії, інфраструктуру та адміністрування. Додатково розраховано строк окупності та чисту приведену вартість (NPV) переходу за ставкою дисконтування 15 %, що відображає високу вартість капіталу в умовах воєнного стану.

Припущення моделі: вартість години роботи системного адміністратора – 20 дол. США, праці працівника – 6 дол. США; адміністрування – 2 год на місяць у базовому сценарії та 5

год в альтернативному; розгортання та міграція – 40 год; навчання – 4 год на працівника в групах по 10 осіб; втрати продуктивності – 5 % робочого часу протягом двох місяців; сервер з резервним копіюванням – 50 дол. США на місяць, пошта – 1,5 дол. США за скриньку на місяць. Для 50 робочих місць: сервер – 80 дол. США на місяць, адміністрування – 3 та 8 год, міграція – 60 год. Розрахунки виконано в доларах США на п'ятирічному горизонті.

Результати

Для підприємства з 20 робочими місцями річні витрати базового сценарію становлять 6 120 дол. США, з них 92 % – ліцензії; в альтернативному сценарії вони знижуються до 2 160 дол. США, тобто на 64,7 %. Одноразові витрати на впровадження складають 3 456 дол. США, з яких 2 016 дол. – втрати продуктивності в період адаптації, тому перехід окупається приблизно за 10,5 місяця. П'ятирічна ТСО становить 30 600 проти 14 256 дол. США, економія – 53,4 %, NPV – 9 819 дол. США.

Для підприємства з 50 робочими місцями ефект масштабу посилює результат: річні витрати знижуються з 14 820 до 3 780 дол. США (на 74,5 %), одноразові витрати становлять 7 840 дол. США, строк окупності – близько 8,5 місяця, п'ятирічна економія – 63,9 %, NPV – 29 168 дол. США.

За тарифом Business Basic результат принципово інший: для 20 робочих місць річні витрати обох сценаріїв збігаються (2 160 дол. США) і перехід не окупається, а для 50 робочих місць річна економія становить лише 1 140 дол. США при строку окупності 6,9 року.

Отже, перехід на FOSS замінює змінні витрати на ліцензії, пропорційні кількості користувачів, умовно-постійними витратами на сервер та адміністрування, що й зумовлює ефект масштабу. Результат переходу визначається насамперед рівнем тарифу, від якого відмовляється підприємство, а найбільшою статтею одноразових витрат є не технології, а люди: втрати продуктивності становлять 58–64 % витрат на впровадження.

Обговорення

Економічна доцільність переходу на відкрите ПЗ не є універсальною і залежить від пропрієтарного рішення, з яким його порівнюють. Включення Copilot до тарифу Business Standard підвищує вартість ліцензії і привабливість відкритих альтернатив, проте порівняння не є повністю симетричним: відкритий стек у моделі не містить аналогічного ШІ-асистента, тому для підприємств, які покладаються на ці функції, розрахована економія частково завищена. Модель також не монетизує стратегічні переваги FOSS – незалежність від постачальника, контроль над даними та передбачуваність витрат без валютних платежів, – важливі для операційної стійкості бізнесу в умовах війни.

До обмежень моделі належать умовний характер підприємства, неврахування витрат на операційні системи та облікове ПЗ, ризики несумісності форматів документів із контрагентами, а також чутливість до вартості й доступності кваліфікованого адміністратора. Оскільки основним бар'єром є одноразові витрати на впровадження, перспективним механізмом підтримки є державні та міжнародні грантові програми, спрямовані на покриття витрат на міграцію та навчання персоналу.

Висновки

Запропонована модель ТСО дозволяє оцінити перехід МСП на відкрите ПЗ з урахуванням прихованих витрат, насамперед втрат продуктивності в період адаптації персоналу. За актуальним тарифом Microsoft 365 Business Standard міграція на LibreOffice та Nextcloud окупається менш ніж за рік і забезпечує п'ятирічну економію 53,4–63,9 %, тоді як порівняно з Business Basic для невеликого підприємства вона не окупається, що підтверджує потребу в розрахунку для конкретного тарифу й масштабу. Подальші дослідження передбачають апробацію моделі на реальних підприємствах і поширення її на облікові та ERP-системи.

Список використаних джерел

1. Олійник К. Д. Трансформаційний потенціал цифрового сектору як середовища становлення нових форм нерівності: роль ІКТ у нарощенні ресурсного потенціалу країни. Економічний простір, 2025, №198, с. 288–296. URL: https://economic-prostir.com.ua/wp-content/uploads/2025/03/198-288-296-olijnyk.pdf (дата звернення: 26.09.2026).
2. Microsoft 365 for business: compare plans. URL: https://www.microsoft.com/en-us/microsoft-365/business (дата звернення: 26.09.2026).
3. LibreOffice. URL: https://www.libreoffice.org (дата звернення: 26.09.2026).
4. Nextcloud Server. URL: https://github.com/nextcloud/server (дата звернення: 26.09.2026).

Бакало В. І.
Львівський національний університет імені Івана Франка, Львів, Україна
Контактний автор: Владислав Бакало, e-mail: Vladyslav.Bakalo@lnu.edu.ua, ORCID: 0009-0004-2228-0995

ПЕРЕТВОРЕННЯ ЯКІСНОЇ ОЦІНКИ КІБЕРРИЗИКІВ У КІЛЬКІСНУ ЗАСОБАМИ ВІЛЬНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Анотація. У роботі описано задачу формалізації результатів якісного оцінювання ризиків інформаційної безпеки для їх використання у математичних моделях. Запропонований підхід базується на алгоритмі перетворення якісних лінгвістичних оцінок, отриманих під час аудиту інформаційних активів, у кількісні показники ризику. Визначаються цінність обраного активу та рівень його вразливості, після чого розраховуються середня вразливість та нормований показник ризику. З використанням інструментів Python, CustomTkinter і SQLite створено програмний модуль та проведено експеримент на синтетичному наборі даних про активи. Отримані результати підтвердили існування монотонної залежності отриманого значення ризику від цінності обраного активу та рівня його вразливості. Запропонований підхід може використовуватися як основа для задач підтримки прийняття рішень.

Ключові слова: інформаційна безпека; кіберризик; оцінювання ризику; кількісна оцінка; вільне ПЗ; Python.

Вступ

Управління ризиками включає ідентифікацію наявних активів та пов'язаних із ними загроз, оцінювання й оброблення ризику. Міжнародний стандарт ISO/IEC 27005, який є однією з методологічних основ такого процесу, пропонує рекомендації щодо управління ризиками інформаційної безпеки [1]. Часто результати оцінювання подаються як лінгвістичні категорії, наприклад, «низький», «середній», «високий» рівень ризику. Для експертного аналізу такий формат зручний, проте ускладнює використання цих даних у математичних моделях [2]. Інтерпретація відповідей аудитора також може впливати на якісні оцінки, тому доцільно формалізувати перехід від текстових оцінок до числового представлення. Основна ідея запропонованого підходу полягає у представленні експертного оцінювання у формі, придатній для автоматизованого аналізу. Метою роботи є розробка та перевірка алгоритму перетворення якісних результатів оцінювання кіберризиків у кількісні показники засобами вільного програмного забезпечення. Він дозволить провести етап якісного оцінювання та автоматизувати такі етапи, як числове представлення та розрахунок ризику. Такий модуль може використовуватися як один із модулів системи підтримки прийняття рішень.

Матеріали та методи

Програмну реалізацію виконано мовою Python [3] у середовищі розробки PyCharm 2025.2.1.1. Для створення графічного інтерфейсу та реалізації основних функцій застосовано бібліотеки Python, зокрема CustomTkinter [4], PyQt6, SQLite та засоби оброблення даних і математичних розрахунків (NumPy, Matplotlib та Seaborn). Розробка та тестування виконувалися у локальному середовищі; параметри математичної моделі задавалися безпосередньо у програмному коді.

Результати

Математична модель. Розглядаємо інформаційну систему, яка складається з таких активів: апаратне забезпечення, програмне забезпечення, мережеві ресурси, персонал та дані [1]. Для кожного з них аудитор задає значення цінності та відповідає на блок запитань, пов’язаних із рівнем захищеності. Результати початкового оцінювання мають якісну форму, але передаються алгоритму через дискретну числову шкалу. Цінність активу (C_i) визначається за наступною шкалою: допоміжний актив – 1, важливий актив – 2, критичний актив – 3. Для оцінювання параметрів захищеності використовується обернена шкала, де більший бал означає більшу вразливість: 1 – надійний захист, 2 – частковий захист, 3 – відсутній або недостатній захист.

Нехай для -го активу задано m_i оцінок параметрів захищеності: $x_{ij} \in \{1,2,3\}$, де x_{ij} – числове представлення відповіді на j -те питання. Тоді середня вразливість активу визначаємо як:

$$V_i = \frac{1}{m_i} \sum_{j=1}^{m_i} x_{ij}.$$

Для формування кількісного показника ризику використовуємо добуток цінності активу та його середньої вразливості: $R_i = C_i \cdot V_i$. За шкали 1-3 максимальне теоретичне значення показника дорівнює $R_{\max} = 3 \cdot 3 = 9$. Тому для знаходження нормованого показника використовуємо формулу:

$$R_i^{norm} = \frac{R_i}{9}.$$

Максимально можливому ризику відповідає значення 1. Водночас мінімальне значення в межах цієї шкали становить $1/9$, оскільки мінімальні значення C_i та V_i дорівнюють одиниці. Проте така модель є спрощеним кількісним представленням результатів експертного оцінювання. Тому цей числовий показник не означає ймовірність ризику. Використовуємо його для стандартизації результатів оцінювання та підготовки даних для подальшого аналізу.

Алгоритм перетворення. Процес оброблення одного профілю активу складається з наступних етапів. Спочатку аудитор обирає категорію активу та відповідає на питання (рисунок 1).

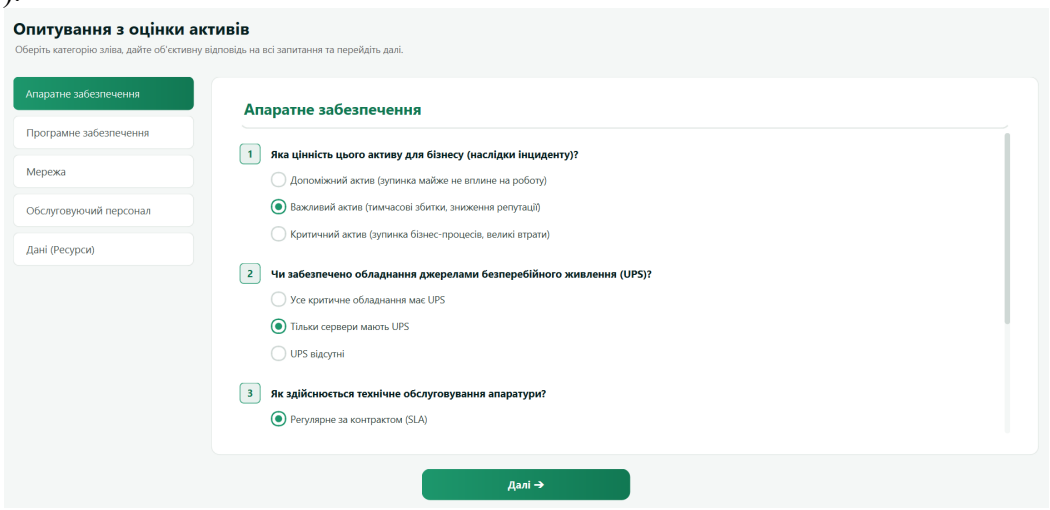


Рис. 1 – Інтерфейс модуля збору даних якісних оцінок активів

Далі якісні характеристики автоматично замінюються відповідними числовими значеннями відповідно до встановленої шкали. На третьому етапі для активу обчислюється середня вразливість V_i. Після цього визначається абсолютний показник ризику R_i та його нормована форма R_i^{norm}. Результат зберігається разом із початковими оцінками, що дає змогу відтворити послідовність обчислень. Узагальнений алгоритм можна подати таким чином:

$$\text{Якісна оцінка} \rightarrow \text{числове кодування} \rightarrow V_i \rightarrow R_i \rightarrow R_i^{norm}.$$

Експериментальна перевірка. Для перевірки математичної моделі сформовано синтетичний контрольний набір із п’яти профілів, що відповідають основним категоріям активів. Для кожного активу використано шість параметрів оцінювання захищеності. Вхідні дані мають такий вигляд (таблиця 1).

Таблиця 1 – Вхідні дані синтетичного набору даних

Актив	C_i	Оцінки захищеності x_{ij}
Апаратне забезпечення	3	3, 3, 2, 3, 2, 3
Програмне забезпечення	3	2, 2, 3, 2, 3, 2
Мережа	2	2, 3, 2, 2, 3, 2
Персонал	2	2, 1, 2, 1, 2, 1
Дані	3	3, 3, 3, 2, 3, 3

Для кожного активу розраховуємо середню вразливість, абсолютний ризик та нормований показник (таблиця 2).

Таблиця 2 – Розрахунок показників для активів

Актив	Цінність C_i	Середня вразливість V_i	Ризик R_i	Нормований ризик R_i^{norm}
Апаратне забезпечення	3	2,67	8,01	0,890
Програмне забезпечення	3	2,33	6,99	0,777
Мережа	2	2,33	4,67	0,519
Персонал	2	1,50	3,00	0,333
Дані	3	2,83	8,49	0,943

Найвище значення нормованого показника отримано для категорії «Дані» - 0,943 через одночасно високу цінність активу ($C = 3$) та високий середній рівень вразливості ($V = 2{,}83$). Найменше значення отримано для персоналу (0,333) через нижчий середній показник вразливості. Поведінку запропонованої моделі також можна дослідити зміною кожного з двох вхідних параметрів. Для трьох можливих значень цінності активу та трьох граничних значень середньої вразливості отримуємо (рисунок 2).

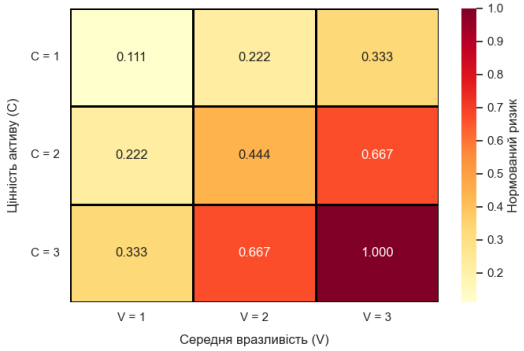


Рис. 2 – Теплова карта нормованого ризику залежно від цінності активу та середньої вразливості

Отримані значення демонструють монотонне зростання показника при збільшенні вразливості незалежно від фіксованого значення цінності. Аналогічно, при однаковій вразливості збільшення цінності активу призводить до збільшення значення ризику.

Обговорення

Проведений експеримент показав, що запропонована процедура забезпечує перехід від якісних оцінок до кількісного представлення ризику. Основною перевагою підходу є не заміна експертної оцінки математичним обчисленням, а формалізація результатів цього оцінювання в такому форматі, який може бути використаний для подальших розрахунків. Разом з тим запропонована модель має низку обмежень. По-перше, початкові відповіді залежать від експертного оцінювання, тому числова формалізація не усуває суб'єктивність вихідних даних. Вона лише забезпечує спосіб їх подальшої математичної обробки. По-друге, використання простого добутку не враховує окремо ймовірність реалізації конкретної загрози, величину потенційних збитків, взаємозалежності між активами та можливі часові зміни ризику. Тому отриманий показник слід розглядати як нормовану оцінку ризику, а не як повну модель очікуваного фінансового збитку. Для демонстрації роботи використано трирівневу шкалу. Доцільно застосовувати багаторівневі шкали або ввести вагові коефіцієнти для окремих параметрів. Розроблений модуль може розглядатися як початковий етап системи підтримки прийняття рішень. Окремою перевагою запропонованої реалізації є використання вільного програмного забезпечення. Це дає можливість створити автономний інструмент із локальним зберіганням даних без використання комерційних платформ.

Висновки

У роботі розроблено та формалізовано підхід до перетворення якісних результатів оцінювання кіберризиків у кількісні показники. Запропонована процедура передбачає числове представлення важливості активу та рівня його вразливості, обчислення середньої вразливості, формування показника ризику та його нормалізацію. Практичну реалізацію виконано з використанням вільного програмного забезпечення на базі Python, CustomTkinter

та SQLite. Експеримент підтвердив відповідність результатів заданій математичній моделі та продемонстрував монотонне зростання ризику при збільшенні цінності активу або рівня його вразливості. Отримані показники можуть бути використані для порівняння активів та формування вхідної інформації для подальших задач підтримки прийняття рішень.

Список використаних джерел

1. ДСТУ ISO/IEC 27005:2023. Інформаційна безпека, кібербезпека та захист конфіденційності. Настанова керування ризиками інформаційної безпеки. Чинний від 2024-01-01. Київ : ДП «УкрНДНЦ», 2023. 66 с.
2. Родін Є. С. Процесні підходи до моделювання у сфері управління ризиками інформаційної безпеки. Математичні машини і системи. 2012. № 4. С. 142–148.
3. Python Software Foundation. Python 3.13 Documentation. URL: https://docs.python.org/3.13/.
4. Tom Schimansky. CustomTkinter: A modern and customizable Python UI-library based on Tkinter. GitHub repository. URL: https://github.com/TomSchimansky/CustomTkinter.

Березинський І. В., Чурілова І. С.
Військовий інститут телекомунікацій та інформатизації імені Героїв Крут, науковий центр зв'язку та інформатизації, м. Київ, Україна
Контактний автор: Ігор Березинський, e-mail: lambert0301@ukr.net, ORCID: 0009-0004-2736-3837
Ірина Чурілова, e-mail: iryna.churilova@viti.edu.ua, ORCID: 0009-0007-6195-9265

ВІДКРИТІСТЬ «OPEN SOURCE AI». АВТОРСЬКЕ ПРАВО ТА ПРИВАТНІСТЬ

Анотація. «Open Source AI» (штучний інтелект із відкритим кодом) — системи штучного інтелекту, чий вихідний код, архітектура, ваги моделі (параметри навчання) та інструменти є вільними для перегляду, вивчення, аналізу, навчання, модифікації та розповсюдження.

Загалом перевага таких моделей полягає у відкритості як для споживача так і для автора-розробника, корегування виявлених помилок та неточностей і як наслідок це може сприяти покращенню самої моделі та її адаптації. Така модель може значно скоротити терміни експериментальних випробувань та час для їх проведення.

Ключові слова: Open Source AI, авторське право, приватність, модель, модифікація, ШІ, WIPO, AI Bills of Materials, OSI, AI Act.

Вступ

Технології ШІ все частіше впливають на сенс нашого життя як через призму позитивного досвіду так і через певні негативні фактори. Позитивна риса проявляється в оперативному вирішенні поставлених завдань та виконання певних функцій в найкоротший термін.

Водночас існує і ряд запитань щодо використання «Open Source AI» і зокрема як таке використання може вплинути на приватність та авторські права? Представники Open Source Initiative (OSI) — великої некомерційної організації, створеної для просування, популяризації та захисту програмного забезпечення з відкритим вихідним кодом та заснованої 08 лютого 1998 року Брюсом Перенсом (Bruce Perens) та Еріком Реймондом (Eric S. Raymond) у Пало-Альто, Каліфорнія інтенсивно вивчають дану модель і головне - чи має вона обов'язково відкривати всі дані, враховуючи авторське право та приватність [1].

Загалом передумовою для здійснення певних свобод є доступ до бажаної форми внесення змін до системи та до засобів її використання. Проте разом з тим постає дедалі частіше питання: хто власник такого коду, створеного за допомогою AI, чи можна його використовувати в бізнес-проєктах та не зачіпає це авторські та суміжні права?

Однозначної відповіді наразі немає. Правила залежать від багатьох факторів: юрисдикції, умов конкретного AI-сервісу, характеру людського внеску, джерела використаного коду, ліцензій залежностей і внутрішніх правил компанії. WIPO (Всесвітня організація інтелектуальної власності) прямо зазначає, що AI створює питання щодо авторства, власності, винагороди правовласників і використання AI у творчому процесі, а правові підходи продовжують розвиватися. Тому безпечна стратегія для розробника — не сприймати AI як "автоматичного власника" або "автоматичного автора", а розглядати його як інструмент у процесі створення програмного забезпечення [2].

Прикладом таких моделей є Llama від Meta або Mistral, які дозволяють розробникам завантажувати та запускати їх на власних серверах, модифікувати для специфічних завдань або інтегрувати у власні застосунки без значних ліцензійних обмежень. Це відкриває широкі можливості для створення нових AI-інструментів, проведення досліджень та навчання, роблячи технології ШІ доступнішими для ширшого кола користувачів.

Матеріали та методи

Визначення «Open Source AI» щодо авторського права або приватності не має чіткого доступу або обмеження у використанні. Водночас з огляду на національне законодавство можливо забезпечити даний доступ ліцензією чи іншим документом. Такі моделі є захищеними авторським правом, оскільки державна правова система має більше можливостей для вирішення цього питання. У будь-якому випадку, нам потрібне чітке

твердження, що супроводжує розповсюдження таких моделей, яке гарантує, що вони вільно доступні для всіх.

«Open Source AI» надає розробникам доступ до ресурсів, вирішальних для розробки різноманітних застосунків, що відображають суспільне середовище та дозволяють створювати моделі, узгоджені з різними системами цінностей.

Результати

Моделі «Open Source AI» вимагають інших понять та термінології, ніж програмне забезпечення з відкритим кодом, зокрема, оскільки вони складніші за своїм складом. Такі моделі базуються на кількох компонентах (від коду до вагових коефіцієнтів та навчальних даних) і часто залучають кількох учасників. Тому необхідно зрозуміти, що саме означає ШІ з відкритим кодом, і які правові наслідки пов'язаної з ним ліцензії на всю модель ШІ [3].

В Україні триває розробка законодавства у сфері штучного інтелекту з урахуванням європейського Регламенту AI Act. Майбутній закон має встановити правила використання ШІ, забезпечити захист прав людей та водночас створити умови для розвитку інновацій. Документ має охопити весь життєвий цикл систем штучного інтелекту — від розробки та навчання до аудиту і виведення з експлуатації. Також планується визначити відповідальність провайдерів та користувачів ШІ-систем, забезпечити прозорість алгоритмів і встановити правила реагування на інциденти, які можуть завдати шкоди людям або майну.

Також пропонується визначити заборонені практики використання ШІ. До них можуть належати маніпулятивні методи впливу, використання вразливості людей через вік, інвалідність або соціально-економічне становище, прогнозування злочинів виключно на основі профілювання, розпізнавання емоцій у сфері праці та освіти, біометрична категоризація за чутливими ознаками. Окремо пропонується обмежити дистанційну біометричну ідентифікацію в режимі реального часу у громадських місцях, за винятком окремих правоохоронних та безпекових випадків.

Наразі передбачається, що Міністерство цифрової трансформації України забезпечуватиме реалізацію державної політики у сфері штучного інтелекту, аналізуватиме практику застосування законодавства, сприятиме розвитку ШІ-грамотності та міжнародній співпраці.

Цифровий регулятор здійснюватиме розробку підзаконних актів і методичних рекомендацій, проводитиме ринковий нагляд за універсальними моделями ШІ. Окремо експерти працюють над питаннями регулювання наборів даних для навчання ШІ, винятками для дослідницьких систем, а також правилами застосування штучного інтелекту у секторі безпеки й оборони. Таке питання є особливо важливим, оскільки європейський AI Act не поширюється на військову сферу. Також триває узгодження термінології, зокрема понять «дипфейк», «впроваджувач», «ШІ-грамотність» та «валідаційні дані».

Наразі в роботі перебуває розділ щодо високоризикових систем штучного інтелекту. До таких систем пропонується віднести технології біометричної ідентифікації та категоризації, системи управління критичною інфраструктурою, використання ШІ в освіті, під час найму та управління персоналом, у сфері доступу до державних і фінансових послуг, правосуддя та правоохоронної діяльності. Також обговорюється статус систем, що обробляють екстрені виклики громадян та визначають медичні заклади для госпіталізації. Такий Закон має наблизити Україну до стандартів ЄС. Очікується, що ухвалення закону про штучний інтелект дозволить Україні синхронізувати національне законодавство із нормами Євросоюзу, створити умови для виходу українських ІТ-компаній на міжнародні ринки та сформувати прозорі правила використання ШІ [4].

Обговорення

Тож наскільки відкритим має бути штучний інтелект з відкритим кодом? Таке обговорення інтенсивно триває серед OSI, Free Software Foundation, Software Freedom Conservancy, Open Source Declaration , Digital Public Goods Alliance та інших. Багато хто стверджує, що для того, щоб бути справді відкритими, системи штучного інтелекту повинні бути повністю прозорими, від коду до даних. Зрештою, ці дані формують рішення та

упередження ШІ, і без доступу до них розробники та кінцеві користувачі мають дуже мало розуміння того, як або чому ШІ поводиться так, як він поводиться.

Однак інші більш обережні. Вони зазначають, що розголошення конфіденційних або службових даних може мати негативні наслідки, зокрема порушення конфіденційності або розкриття інтелектуальної власності конкурентам.

Такий приклад «взірця відкритого штучного інтелекту» називають деякі спеціалісти EleutherAI. Він розширює межі відкритого штучного інтелекту, не лише випускаючи моделі, але й інструменти для їх відтворення. Проте в даному випадку виникають додаткові питання: як ми можемо гарантувати, що ці моделі не увічнюють шкідливі упередження або не спричиняють непередбачуваних наслідків, якщо їхні навчальні дані є недосконалими або неповними? Водночас великі корпорації, такі як OpenAI, Meta та Anthropic, не пов'язані цими обмеженнями, і це може загальмувати розвиток більш різноманітних моделей ШІ з відкритим кодом [5].

Для програмного коду ситуація є особливо дискусійною, адже в Україні закон прямо визнає комп’ютерні програми об’єктами авторського права та охороняє їх як літературні твори, однак якщо значна частина коду була автоматично згенерована без суттєвої творчої участі розробника, виникає проблема доведення авторства людини. У сфері комерційного використання AI-контенту важливу роль відіграють договори. На практиці права на тексти, код чи дизайн, створені за допомогою нейромереж, часто визначаються умовами договору між сторонами та правилами використання AI-сервісу щодо договорів і розпорядження майновими правами інтелектуальної власності. Незабаром планують дозволити використовувати контент ШІ без згоди автора і встановити право особливого роду (sui generis) на неоригінальні об’єкти, згенеровані системами штучного інтелекту.

Ключовим актом в нашій державі є Закон України «Про авторське право і суміжні права» № 2811-IX.

Відповідно до статті 1 Закону, автором є фізична особа, яка своєю творчою діяльністю створила твір. Аналогічно стаття 6 передбачає, що авторське право поширюється на твори у сфері літератури, мистецтва, науки, які є результатом творчої діяльності автора.

Водночас стаття 33 Закону окремо вводить право особливого роду (sui generis) на неоригінальні об’єкти, згенеровані комп’ютерною програмою. Закон прямо визначає, що йдеться про результат, створений без безпосередньої участі фізичної особи. У такому випадку класичне авторське право не виникає, а застосовується право особливого роду.

Суб’єктами такого права можуть бути правовласники комп’ютерної програми, користувачі або інші особи — залежно від умов використання конкретної моделі ШІ та договорів між сторонами.

Тобто, якщо людина творчо використовувала нейромережу як інструмент і суттєво впливала на результат, автором зазвичай вважається саме вона. Водночас у випадках, коли контент був автоматично згенерований без творчої участі людини, закон може розглядати його як неоригінальний об’єкт із режимом права sui generis [6].

Висновки

Вивчення та дослідження «Open Source AI» дає змогу зробити висновки, що чіткого вирішення проблеми вільного доступу та приватності допоки не існує.

Наразі спостерігається криза правового регулювання. Open Source AI дає світу технологічну незалежність та прозорість алгоритмів, але вимагає радикально нових юридичних інструментів — наприклад, впровадження «AI Bills of Materials» (паспортів даних) та криптографічного захисту прав авторів [7].

Список використаних джерел

1. Open Source Initiative (OSI). Офіційний сайт URL: https://opensource.org/ (дата звернення: 28.09.2026).

2. Етика, авторське право та інтелектуальна власність при використанні генеративного AI в коді: https://thepetrov.com/ai-code-ethics-copyright-ip/. 08.09.2026 року. Михайло Петров. URL: https://thepetrov.com/ai-code-ethics-copyright-ip/ (дата звернення: 28.09.2026).

3. Open Source AI – definition and selected legal challenges. Yaniv Benhamou (University of Geneva). 15.04.2024. URL: https://legalblogs.wolterskluwer.com/copyright-blog/open-source-ai-definition-and-selected-legal-challenges/ (дата звернення: 28.09.2026).
4. «Від ChatGPT до ШІ в освіті, судах і на роботі: Україна готує закон про контроль штучного інтелекту». Судово-юридична газета. 08.09.2026. URL: https://sud.ua/uk/news/laws/371648-vid-chatgpt-do-shi-v-osviti-sudakh-i-na-roboti-ukraina-hotuie-zakon-pro-kontrol-shtuchnoho-intelektu (дата звернення: 28.09.2026).
5. «AI and Open Source: Defining the New Era». Dotan Horovits 30.01.2025. URL: https://horovits.medium.com/ai-and-open-source-defining-the-new-era-5ba89e11c26d (дата звернення: 28.09.2026).
6. Як законодавство та судова практика регулюють авторське право на ШІ-контент. Судово-юридична газета. 11.05.2026. URL:https://sud.ua/uk/news/publication/360630-kak-zakonodatelstvo-i-sudebnaya-praktika-reguliruyut-avtorskoe-pravo-na-ii-kontent (дата звернення: 28.09.2026).
7. Штучний інтелект та право на приватність: актуальні виклики цифрової епохи. Науковий вісник Ужгородського національного університету. Т.П. Попович, О.В. Маслюк, А.А. Мацола. Том 5 №90 (2025). URL: https://visnyk-pravo.uzhnu.edu.ua/issue/view/19766 (дата звернення: 28.09.2026).

Величко В. В.
Національний університет «Чернігівська політехніка», м. Чернігів, Україна
e-mail: vladik1701004@gmail.com

РЕАЛІЗАЦІЯ ПРИНЦИПІВ АРХІТЕКТУРИ ZERO TRUST У VPN-ШЛЮЗАХ НА ОСНОВІ ДИНАМІЧНОЇ ОЦІНКИ КОНТЕКСТУ СЕСІЙ

Анотація. Традиційні VPN-шлюзи перевіряють легітимність користувача лише один раз – у момент встановлення тунелю. Якщо файл конфігурації або сертифікат клієнта викрадено, зловмисник отримує безперешкодний доступ до внутрішньої мережі підприємства під виглядом співробітника. У роботі розглянуто практичний підхід до реалізації моделі Zero Trust для відкритого сервера OpenVPN. Замість ресурсомісткого аналізу зашифрованого трафіку ми аналізуємо метадані журналу підключень у реальному часі. Визначено чотири базові критерії компрометації облікових даних: одночасні сесії під одним сертифікатом, вхід поза робочим графіком, нетипова швидкість зміни географічних зон IP-адрес та повторювані помилки TLS-рукостискання. Такий аналіз дозволяє виявляти аномалії без встановлення стороннього пропрієтарного ПЗ на клієнтські пристрої.

Ключові слова: архітектура Zero Trust; OpenVPN; безпека віддаленого доступу; аналіз журналів; виявлення компрометації ключів; відкрите ПЗ.

Вступ

Масовий перехід організацій на гібридний та віддалений режим роботи зробив VPN-шлюзи основним входом до внутрішньої корпоративної інфраструктури. Більшість таких систем функціонують за класичним принципом периметрової безпеки: клієнт проходить перевірку автентичності під час первинного рукостискання, після чого отримує мережевий доступ без подальшої перевірки легітимності самої сесії.

Головний ризик такої схеми – компрометація автентифікаційних артефактів. Якщо файл конфігурації (.ovpn) та закритий ключ сертифіката X.509 скопійовано сторонньою особою, шлюз сприймає несанкціоноване з’єднання як штатного користувача.

Архітектура нульової довіри, сформульована у стандарті NIST SP 800-207, вимагає принципу «ніколи не довіряй, завжди перевіряй». Це означає, що перевірка має тривати впродовж усього часу з'єднання. Мета роботи – розробити модель безперервної оцінки безпекового контексту VPN-підключень для виявлення ознак компрометації облікових даних за допомогою штатних механізмів журналювання відкритого ПЗ.

Матеріали та методи

Базою для дослідження обрано сервер OpenVPN 2.6 під управлінням Linux. OpenVPN фіксує події життєвого циклу сесій у журналах аудиту (status.log) та підтримує керування активними клієнтами через протокол TCP/Unix-сокет (Management Interface).

Концепція NIST SP 800-207 визначає розподіл на точку прийняття рішень Policy Decision Point та точку застосування політик Policy Enforcement Point. У нашій схемі сервіс OpenVPN виступає як PEP, розриває тунель або змінює маршрути, а окремий модуль аналізу подій – як PDP.

Для оцінки контексту відібрано 4 метрики, які можна вилучити безпосередньо з текстового журналу сервера:

– Колізія ідентифікаторів (дублювання сесії): фіксація двох або більше одночасно активних тунелів з однаковим іменем сертифіката (Common Name, CN), але різними зовнішніми IP-адресами.

– Часова відповідність: перевірка моменту автентифікації на відповідність дозволеному робочому інтервалу конкретної групи користувачів, наприклад, штатний день з 08:00 до 20:00.

– Оцінка просторової несуперечливості: порівняння координат двох послідовних підключень клієнта за публічними базами IP-геолокації GeoLite2 відносно часу між ними, індикатор зміни мережевого маршруту, підключення через комерційний VPN, Tor або викрадення конфігурації.

– Частота помилок TLS: сплеск невдалих спроб узгодження сесії з одного IP за короткий проміжок часу, ознака автоматизованого сканування чи підбору.

Така логіка дозволяє не навантажувати сервер дешифруванням мережевих пакетів: весь моніторинг здійснюється виключно за метаданими заголовків сесії, які OpenVPN записує у звичайний текстовий файл.

Обговорення

Запропоноване рішення має низку практичних відмінностей:

– На відміну від утиліт на зразок Fail2Ban, які реагують лише на невдалі спроби авторизації, наш підхід контролює дії клієнта вже після успішного проходження перевірки сертифіката.

– На відміну від важких комерційних систем ZTNA, наприклад, Cisco або Palo Alto Networks, описаний принцип реалізується виключно засобами відкритого ПЗ та не вимагає встановлення сторонніх агентів стеження на робочі станції співробітників.

Обмеження підходу: Точність географічного критерію напряму залежить від повноти баз GeoIP та поведінки мобільних провайдерів (CGNAT), тому цей показник не можна використовувати як єдину підставу для блокування користувача без додаткових ознак, наприклад, паралельної активної сесії. Подальший розвиток проєкту полягає у підтримці протоколу WireGuard та перевірці цифрового відбитка пристрою.

Висновки

Проаналізовано проблему довіри за замовчуванням у традиційних VPN-тунелях та показано практичну можливість реалізації вимог Zero Trust без використання закритих комерційних систем.

Сформульовано перелік із чотирьох контекстних метрик моніторингу подій підключення, що дозволяє фіксувати компрометацію ключів та викрадення конфігурацій без розкриття вмісту шифрованого трафіку.

Показано спосіб інтеграції легкого модуля прийняття рішень із відкритим шлюзом OpenVPN через штатний інтерфейс управління.

Список використаних джерел

1. Rose S., Borchert O., Mitchell R., Connelly S. Zero Trust Architecture. NIST Special Publication 800-207, 2020. 59 p. DOI: https://doi.org/10.6028/NIST.SP.800-207

2. OpenVPN Community Software. OpenVPN Management Interface Guide. URL: https://openvpn.net/community-resources/management-interface/ (дата звернення: 01.10.2026).

3. Stafford V. Zero Trust Architecture. Journal of Research of the National Institute of Standards and Technology, 2020. Vol. 125, pp. 1–12. DOI: https://doi.org/10.6028/jres.125.034

4. Barth A. Security Architecture of Remote Access Systems. Journal of Computer Security, 2021. Vol. 29(4), pp. 411–428.

Велігорський Б. О., Хоменко М. А.
Національний університет «Чернігівська політехніка», гр. АСД-ЕІ25, м. Чернігів, Україна
Національний університет «Чернігівська політехніка», Кафедра електроніки, автоматики, робототехніки та мехатроніки, м. Чернігів, Україна
Контактний автор: Богдан Велігорський, e-mail: bogdan_vel@stu.cn.ua, ORCID: 0009-0008-4492-4156

ОПТИМІЗАЦІЯ КЕРУВАННЯ MICROGRID ЗА ДОПОМОГОЮ БІБЛІОТЕКИ CVXPY

Анотація. З появою та широким впровадженням відновлювальних джерел електроенергії актуальною стає задача оптимізації керування режимами роботи мікромереж, які включають у себе альтернативні джерела генерації з метою забезпечення цілодобового та надійного електропостачання. Удосконалення керування дає змогу заощаджувати, в першу чергу, ресурси з централізованої мережі, а отже й зменшувати вартість імпортованої електроенергії. У даній роботі розглянуто програмний підхід оптимізації керування Microgrid, реалізований на основі відкритої бібліотеки CVXPY для знаходження мінімуму цільової функції простої моделі мікромережі, що включає всі основні її компоненти. Отримані результати дають можливість виявити закономірності, за яких досягнути потреб попиту можливо без імпорту електроенергії з централізованої мережі, а отже й заощадивши при цьому фінансові ресурси.

Ключові слова: відкрите ПЗ; електроенергетика; мікромережі; Microgrid; CVXPY; Python.

Вступ

Задача керування Microgrid включає в себе урахування попиту на електроенергію P_{LOAD}, а також генерацію з відновлювальних джерел електроенергії P_{PV} [1]. З метою забезпечення попиту споживання без залучення централізованої мережі необхідно, щоб потужність генерації з відновлювальних фотоелектричних джерел дорівнювала або переважала попит. Однак, характер генерації з відновлювального джерела і попиту, як правило, є мінливими впродовж дня. У деякі години може бракувати потужності з альтернативних джерел, а в деякі – попит значно менший за генерацію і виникає питання збереження надлишків. Тому, в моделях Microgrid для цього використовується батарея P_{BAT}. Якщо ресурсів з відновлювального джерела достатньо, то батарея заряджається, якщо ж ні – розряджається. Однак, Microgrid може працювати і в мережевому режимі, коли задіюється потужність з централізованого джерела P_{GRID}. Ураховуючи всі вищезазначені складники, виходить наступне рівняння балансу потужностей:

$$P_{LOAD} = P_{PV} + P_{BAT} + P_{GRID}; \qquad P_{GRID} = P_{LOAD} - P_{PV} - P_{BAT} \tag{1}$$

Окрім вирішення задачі балансу (1) потужності, оптимальне керування режимами роботи мікромережі має економічний зміст. У задачах, пов'язаних із Microgrid, метою оптимізації управління може бути мінімізація загальної вартості спожитої електроенергії з централізованої мережі. Відповідна функція витрат без урахування експортування електроенергії, що потребує мінімізації, матиме наступний вигляд:

$$J = \sum_{k=0}^{N} c_{BUY} \cdot P_{GRID}(k) \tag{2}$$

де c_{BUY} – вартість купівлі електроенергії, N – кількість розподілів споживання (зазвичай N=24), $P_{GRID}(u_k)$ - спожита потужність у часовий розподіл (годину) k.

Якщо взяти до уваги, що купівля електроенергії стосується тільки мережевого режиму роботи Microgrid (2), то можливою задачею, що вирішує бібліотека CVXPY, є знаходження таких особливостей генерації та попиту на електроенергію, за яких імпорт електроенергії буде найменшим з урахуванням зазначених обмежень.

Основними обмеженнями в даній моделі, які можна представити математично, є:

- невід'ємний попит ($P_{LOAD} \geq 0$);
- невід'ємна генерація з відновлювальних фотоелектричних джерел ($P_{PV} \geq 0$);
- невід'ємний, але обмежений імпорт електроенергії ($0 \leq P_{GRID} \leq P_{GRID_MAX}$).

Ці обмеження є обов'язковими для подальшої задачі. Крім того, при моделюванні попиту та генерації з відновлювального джерела задля побудови більш точної моделі потрібно також враховувати додаткові особливості, як-от максимум виробленої електроенергії опівдні, нульові значення генерації в нічний час тощо.

Матеріали та методи

Вільне програмне забезпечення (Free open source software - FOSS) є поширеною філософією надання та розповсюдження програмного забезпечення, яка дозволяє користувачам застосовувати його у будь-яких цілях, а у разі потреби – модифікувати елементи програми таким чином, щоб вона виконувала потрібні задачі [2]. До того ж, удосконалене програмне забезпечення також можна поширювати, що надає можливість іншим користувачам вирішувати ширший спектр задач за допомогою обраної програми. Крім того, досить часто таке програмне забезпечення є безкоштовним та загальнодоступним, що суттєво спрощує проведення наукового дослідження або використання його в навчальних цілях.

Для демонстрації вирішення задачі опуклої оптимізації при керуванні режимами роботи мікромережі ми використаємо відкриту бібліотеку CVXPY, мову програмування Python (версія 3.12.7), середовище розробки Pycharm (версія 2026.2.2), а також бібліотеку Matplotlib (версія 3.11.2).

CVXPY – це мова моделювання з відкритим вихідним кодом, вбудована в Python та призначена для вирішення задач опуклої оптимізації: знаходження мінімумів та максимумів цільової функції з урахуванням вказаних обмежень (constraints). Вона дає змогу представляти задачу оптимізації у математичній формі, а не в обмежувальній стандартній, яка потрібна для розв'язувачів [3]. CVXPY є загальнодоступним проєктом, в який кожен може зробити свій внесок з метою поліпшення.

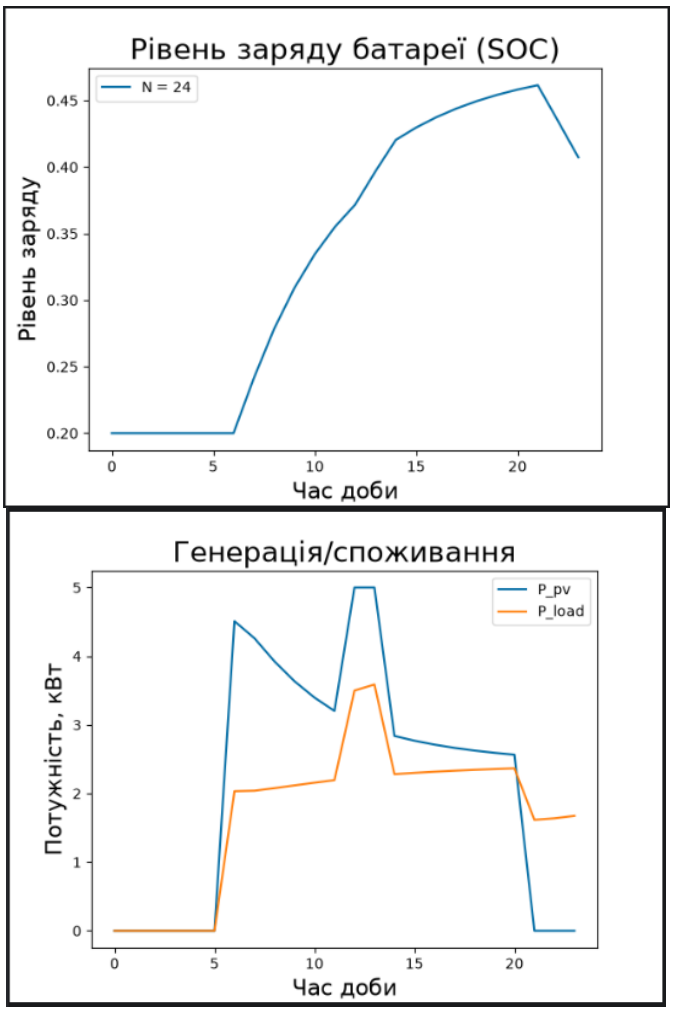
Python – це інтерпретована високорівнева об’єктно-орієнтована мова програмування із суворою динамічною типізацією, розроблена Гвідо ван Россумом на початку 1990-х [4]. Відкритість коду, можливість використання Python у діалоговому режимі, наявність у стандартному дистрибутиві модуля для розробки програм з графічним інтерфейсом та адаптивність під математичні задачі роблять цю мову програмування зручним засобом для швидкого вирішення задач, де потрібно візуалізувати математичні дані.

Інтегроване середовище розробки Pycharm від JetBrains призначене для створення програм на мові Python [5]. Воно надає засоби для написання та аналізу коду, відлагоджувач, інструмент для запуску модульних тестів та підтримує веб-розробку з використанням фреймворку Django. Однією з ключових переваг даного рішення є кросплатформенність, що дозволяє створювати програми в будь-якій популярній операційній системі: Windows, Mac OS X або Linux.

Бібліотека Matplotlib призначена для візуалізації наборів даних у вигляді графіків. Спершу її створив нейробіолог Джон Гантер для своїх потреб, а згодом її почали використовувати та розвивати численні фахівці з найрізноманітніших галузей [6]. У програмі можливим є створення кількох різних графіків завдяки даній бібліотеці, так що можливим є графічне представлення кількох різних величин за одне виконання програми.

Результати

За допомогою мови Python реалізовано спрощену програмну модель Microgrid, яка моделює споживчий попит на наступні 24 години, вироблення електроенергії з відновлювальних фотоелектричних джерел та заряджання/розряджання батареї залежно від того, переважає генерація попит чи навпаки. Крім того, програма здійснює розрахунок вартості спожитої з централізованої мережі електроенергії. Завдяки застосуванню бібліотеки CVXPY здійснено оптимізацію керування режимами роботи мікромережі таким чином, щоб вартість купівлі електроенергії була найменшою. Це зроблено шляхом мінімізації функції витрат, у даному випадку – підбору таких параметрів попиту та генерації впродовж доби, за яких не потрібно буде залучати централізоване джерело електроенергії. Візуалізацію отриманих результатів здійснено з використанням бібліотеки Matplotlib (див. рисунок 1).



а б

Рисунок 1 – Візуалізація даних: зміна рівня заряду (SOC) батареї (а) та згенерованої/спожитої електроенергії (б) впродовж 24 годин

З рисунка 1 видно, що для мінімально можливого споживання електроенергії з централізованої мережі (c_{BUY} = 0) необхідно, щоб попит не перевищував потужність виробленої енергії з фотоелектричних джерел (див. рисунок 1, б). До 20-ї години, поки ця умова виконувалася, було зафіксовано зростання рівня заряду (SOC) батареї (див. рисунок 1, а), тоді як після 20:00 – зменшення. Це пояснюється необхідністю використання акумульованої електроенергії для задоволення споживчих потреб, адже рівень заряду батареї дозволяє це робити без необхідності імпорту електроенергії з централізованого джерела. Тому вартість купівлі електроенергії впродовж усієї доби залишається нульовою.

Обговорення

Порівняно з комерційними рішеннями, як-от MATLAB та Simulink, бібліотека CVXPY потребує встановлення додаткових компонентів таких як Python і Matplotlib для візуалізації результатів, а також уміння програмувати. Водночас, попри деякі труднощі під час встановлення, CVXPY є відкритим програмним забезпеченням, яке кожен може покращити, а перевірка досліджень є прозорою та доступною кожному. Крім того, CVXPY дозволяє вирішувати більш складні задачі, такі як оптимізація керування, а не просте моделювання роботи мікромережі за допомогою конструктивних блоків, як у випадку з Simulink.

Перспективним є застосування бібліотеки CVXPY для оптимізації керування режимами роботи Microgrid у разі застосування графіків погодинних відключень електроенергії, оскільки механізм накладення обмежень на цільову функцію є дуже зручним для представлення у програмному коді, так що можна додавати обмеження з точністю до годин.

Висновки

Оптимізація керування режимами роботи електричної мікромережі дає змогу більш раціонально використовувати доступні ресурси, задовольняючи при цьому основні потреби споживачів. За допомогою мови програмування Python реалізовано програмну модель Microgrid, яка складається з прогнозованого попиту, генерації з відновлювальних фотоелектричних джерел, централізованої мережі, а також батареї, що заряджається/розряджається залежно від того, достатньо відновлювальної генерації для задоволення споживацьких потреб чи ні. До представленої моделі застосовано можливості відкритої бібліотеки CVXPY з метою знаходження таких особливостей генерації та споживання впродовж доби, які би давали змогу звести до мінімуму імпорт електроенергії з централізованої мережі й забезпечувати всі потреби завдяки відновлювальній генерації та батареї.

Майбутні напрями роботи включають побудову більш складної моделі мікромережі, яка буде враховувати невизначеність, а також реалізацію прогнозного керування режимами роботи Microgrid за допомогою бібліотеки CVXPY.

Список використаних джерел

1. P. Zhang. Microgrids. Theory and Practice. John Wiley & Sons, Incorporated, 2024.
2. L. Fortunato, M. Galassi. The case for free and open source software in research and scholarship Philosophical Transactions of the Royal Society A, 2021, vol.(379).
3. CVXPY. URL: https://www.cvxpy.org/ (дата звернення: 17.09.2026).
4. D. Kuhlman. A Python Book: Beginning Python, Advanced Python, and Python Exercises. 2011, Section 1.1.
5. Pycharm. URL: https://www.jetbrains.com/pycharm/ (дата звернення: 17.09.2026).
6. Matplotlib. URL: https://matplotlib.org/stable/#about-us (дата звернення: 17.09.2026).

Гричина О. О.
Національний університет «Чернігівська політехніка», кафедра кібербезпеки та математичного моделювання, м. Чернігів, Україна
Науковий керівник – Петренко Т. А., кандидат технічних наук, доцент кафедри кібербезпеки та математичного моделювання
Контактний автор: Олег Гричина, e-mail: oleh.grichina@gmail.com

FILE NOTIFICATION ATTACK — НОВА АТАКА ЧЕРЕЗ БІЧНИЙ КАНАЛ ПРОТИ ВСІХ ОПЕРАЦІЙНИХ СИСТЕМ / FILE NOTIFICATION ATTACK: A NEW SIDE-CHANNEL ATTACK AGAINST ALL MAJOR OPERATING SYSTEMS

Анотація. У тезах розглянуто клас атак через бічний канал, що використовують підсистеми сповіщення про зміни у файловій системі на операційних системах Windows, macOS, Linux та Android. На основі аналізу дослідження Інституту інформаційної безпеки Технічного університету Граца, оприлюдненого у вересні 2026 року, показано, що непривілейований процес може отримувати час і шляхи файлових подій інших користувачів і програм. Продемонстровані наслідки охоплюють відстеження відвідуваних вебсайтів у Windows, атаку на міжклавішні інтервали та підроблену форму введення пароля в Linux, а також моніторинг медіафайлів WhatsApp в Android. Оцінено практичну загрозу: атака малопридатна для масового шкідливого програмного забезпечення, проте є зручним прихованим інструментом для цільових атак рівня APT. Розглянуто стан виправлень і запропоновані дослідниками заходи захисту.

Ключові слова: атака через бічний канал; сповіщення файлової системи; APT; конфіденційність.

Вступ

У вересні 2026 року дослідники з Інституту інформаційної безпеки Технічного університету Граца (Австрія) оприлюднили статтю [1], в якій описано новий метод атаки через бічний канал, що працює на всіх популярних операційних системах (ОС): Windows, macOS, Linux та Android. Публікацію статті заплановано на конференції ACM CCS 2026. Окремо дослідники перевірили систему контейнеризації Docker, де змогли атакувати інші контейнери, розміщені на тому самому хості.

Вразливість існує на рівні операційної системи, а перші примітиви для її реалізації було додано ще на початку цього століття: у Windows 2000, Mac OS X Leopard 10.5 (2007), Linux 2.6.13 (2005) та Android (2008) [1, 3].

Мета роботи — проаналізувати механізм атаки, її прояви в різних операційних системах та оцінити реальний рівень загрози для користувачів і організацій.

Матеріали та методи

Ключовим механізмом атаки є підсистема сповіщення про зміни у файловій системі. Вона дає змогу програмам реагувати на події без участі користувача: наприклад, коли файл переміщують в одній програмі, інша програма, в якій відкрито цей каталог, бачить зміни й оновлює інтерфейс без будь-якої взаємодії користувача з нею. Цим механізмом користуються текстові редактори, файлові менеджери, клієнти синхронізації та антивіруси [2]. Відповідні інтерфейси: ReadDirectoryChangesW (Windows), FSEvents (macOS), inotify (Linux) та FileObserver (Android) [3].

Проблема полягає в тому, що цю підсистему може використовувати будь-яка програма, щоб дізнаватися про зміни в будь-якій частині файлової системи, якщо вона має право читання на файл або на каталог, у якому він знаходиться. Самі події не розкривають вмісту файлів, проте імена файлів і час подій утворюють бічний канал, з якого можна відтворити дії користувача, програм і системи [2]. У різних ОС це має різні наслідки та можливості для отримання інформації.

Методом дослідження є аналіз оригінальної статті [1], опублікованого авторами коду proof-of-concept та повідомлень про реакцію розробників ОС [2–4].

Результати

Windows. Якщо програма спробує відстежувати каталог, до якого не має доступу, система відмовить, але якщо застосувати моніторинг до кореня диска, ОС повідомляє повний шлях кожного зміненого файлу без перевірки прав доступу, включно з файлами та операціями інших користувачів [2]. За допомогою цього дослідники показали, що стороння програма, запущена від імені іншого користувача, може в реальному часі визначати, які вебсайти відвідуються у Firefox, з точністю 97,8 % [1].

macOS. У macOS ситуація краща: обходу для отримання подій із приватних каталогів не виявлено, тому приватної інформації можна отримати значно менше [1]. Проте атакувальник досі може стежити за поведінкою ОС, програм і користувача, зокрема за запуском застосунків і зміною налаштувань [2].

Linux. Спостереження за доступним для читання каталогом дає події навіть про файли в ньому, які атакувальник прочитати не може. Відстежуючи /dev/input, дослідники отримували сповіщення про кожне натискання клавіші й реалізували атаку на міжклавішні інтервали з точністю 93,1–100 % для семи користувачів, а також віддалений варіант через SSH [3]. Крім того, за шрифтами, які Firefox завантажує для сторінки, вдалося визначати сайт серед 100 найпопулярніших з точністю 87,9 % [1]. Відстежуючи запуск /usr/bin/pkexec, можна визначити момент появи вікна автентифікації Polkit і відобразити поверх нього підроблену форму для введення пароля; це продемонстровано на KDE Plasma 5 та 6 з Wayland [4].

Android. Атака дає змогу дізнаватися, які файли використовують інші програми, причому застосунку-атакувальнику не потрібні жодні дозволи. На прикладі WhatsApp і програми, що виконувалась у фоні на смартфонах Google Pixel та Samsung Galaxy, дослідники отримали інформацію про те, коли і які медіафайли було отримано, надіслано чи відкрито користувачем [1, 2].

Обговорення

Атака потребує вже отриманого доступу до пристрою жертви (можливості виконувати код, зазвичай від імені окремого облікового запису), а здобутою інформацією зазвичай є лише час події та шлях до файлу [2]. Тому звичайне шкідливе програмне забезпечення для викрадення чи шифрування даних навряд чи використовуватиме цей вектор. Натомість цільові атаки проти конкретних осіб, наприклад з боку угруповань Advanced Persistent Threat (APT), маючи більше контексту про жертву та нагод застосувати цю атаку, можуть використати її, залишаючись непоміченими.

Реакція виробників ОС суттєво різниться. У Linux проблему частково усунуто (CVE-2025-68788): виправлення увійшло до ядер 5.10.248, 5.15.198, 6.1.160, 6.6.120, 6.12.65 та 6.18.3, проте атака через pkexec лишається актуальною [4]. Microsoft вважає поведінку Windows передбаченою дизайном, оскільки розкриваються лише імена та шляхи файлів, а не їхній вміст, і посилається на захисні налаштування, які адміністратори можуть увімкнути [2]. Для Android і macOS виправлень на момент публікації не було [2]. Дослідники пропонують заборонити моніторинг усього диска в Windows та запровадити в ядрах Windows і macOS систему дозволів з урахуванням контексту, прав доступу та власника файлів і каталогів [1].

Висновки

Атаки через сповіщення файлової системи демонструють, що навіть метадані без доступу до вмісту файлів можуть стати джерелом витоку конфіденційної інформації. Вразливість має архітектурний характер, існує близько двох десятиліть і досі не усунена повністю в жодній із розглянутих ОС. Перспективними напрямами подальшої роботи є розроблення методів виявлення аномального використання API сповіщень і оцінка застосовності атаки в контейнерних та багатокористувацьких середовищах.

Список використаних джерел

1. Neela S. R., Zhao X., Wultsch J. A., Weissteiner H., Draschbacher F., Gast S., Gruss D. File Notification Attacks: Templating and Exploiting Side-Channel Leakage from the File-Notification Systems on Linux, Windows, and macOS. ACM Conference on Computer and Communications Security (CCS 2026), The Hague, 2026.
2. Windows, Linux, Android File Notification Systems Leak User Activity. SecurityWeek, 2026. URL: https://securityweek.com/windows-linux-android-file-notification-systems-leak-user-activity (дата звернення: 01.10.2026).
3. ‘Decades-old’ bugs found affecting Windows, Android, macOS and Linux. TechRadar, 2026. URL: https://techradar.com/pro/security/decades-old-bugs-found-affecting-windows-android-macos-and-linux-but-the-os-makers-dont-see-it-as-a-big-deal (дата звернення: 01.10.2026).
4. Research into file-notification attacks on Linux. LWN.net, 2026. URL: https://lwn.net/Articles/1096431/ (дата звернення: 01.10.2026).

Денісова О. О.
Київський національний економічний університет імені Вадима Гетьмана, кафедра комп'ютерних наук, Київ, Україна
Контактний автор: Ольга Денісова, e-mail: denisova@kneu.edu.ua, ORCID: 0000-0001-5397-4018

ДОСЛІДЖЕННЯ ВІДТВОРЮВАНИХ ДОСЛІДНИЦЬКИХ ПАЙПЛАЙНІВ ЄВРОПЕЙСЬКОГО ДОСЛІДНИЦЬКОГО ПРОСТОРУ

Анотація. Досліджено практики побудови відтворюваних дослідницьких пайплайнів у Європейському дослідницькому просторі. На основі міждоменного картування виокремлено п'ять архітектурних форм пайплайнів і десять функціональних категорій засобів їх підтримки. Запропоновано розглядати базовою одиницею пайплайну контрольований перехід між версіями даних, моделей і результатів із фіксацією операції, виконавця, критерію приймання та результату перевірки. Підхід забезпечує інтеграцію автоматизованих, експертних і ШІ-підтримуваних операцій, наскрізну простежуваність та виявлення залежних результатів, що потребують повторної перевірки після змін у дослідницькому процесі.

Ключові слова: відтворювані дослідницькі пайплайни; вільне ПЗ; EOSC; FAIR; штучний інтелект.

Вступ

Зростання обсягів і різнорідності дослідницьких даних, ускладнення аналітичних процедур, використання різних програмних засобів і обчислювальних середовищ, а також поєднання автоматизованих, експертних і підтримуваних штучним інтелектом (ШІ) операцій суттєво ускладнюють відтворення сучасних досліджень. За таких умов особливого значення набуває організація дослідницьких процедур у формі пайплайнів, які явно визначають входи, операції, виходи та залежності між окремими етапами й забезпечують можливість їх повторного виконання та, де це можливо, автоматизації [1]. Метою такого підходу є перетворення набору розрізнених процедур на керований і простежуваний процес, в якому можна відтворити програмне середовище, встановити походження результатів та визначити, які з них потребують повторного виконання або перегляду після зміни вхідних даних, програмного коду, параметрів, методики чи проміжних результатів, отриманих за участю ШІ.

Водночас відтворюваність дослідження не може обмежуватися доступністю програмного коду. FAIR-принципи поширюють вимоги щодо можливості пошуку та ідентифікації, доступності, інтероперабельності й повторного використання як на дослідницьке програмне забезпечення, так і на обчислювальні робочі процеси та їх складові [2, 3]. Однак дотримання цих принципів і технічна повторюваність не гарантують наукової коректності, оскільки формально відтворюваний процес може повторювати методологічно необґрунтоване рішення. Це особливо суттєво для пайплайнів, що містять експертні або підтримувані ШІ операції, для яких необхідно фіксувати підстави прийняття рішень та результати їх перевірки. Особливої уваги потребує координація переходів між етапами або операціями – перевірка відповідності виходу попереднього етапу вимогам наступного, узгодження суперечностей, передавання результатів, отримання погоджень та опрацювання винятків [4]. Хоча цей висновок одержано стосовно корпоративних процесів, їх структурна логіка є релевантною для дослідницьких пайплайнів.

Наявні засоби підтримують виконання, фіксацію походження, реєстрацію та документування окремих складових пайплайнів, однак недостатньо формалізованими залишаються їх наскрізна інтеграція з експертними й ШІ-підтримуваними операціями та контроль залежностей між результатами. Європейський дослідницький простір є придатною основою для аналізу таких практик, оскільки п'ять наукових кластерів Європейської хмари відкритої науки (European Open Science Cloud, EOSC) охоплюють різні предметні області та дослідницькі інфраструктури. Метою роботи є визначення архітектурних форм

відтворюваних дослідницьких пайплайнів і функціональних категорій вільного програмного забезпечення для їх реалізації, а також обґрунтування інтеграції автоматизованих, експертних і ШІ-підтримуваних операцій на основі наскрізної простежуваності контрольованих переходів між версіями дослідницьких даних, моделей і результатів.

Матеріали та методи

Дослідження виконано як структуроване міждоменне картування європейських практик із подальшим функціональним і концептуальним синтезом. Корпус сформовано за трьома рівнями: наукові кластери EOSC, горизонтальні проєкти EOSC, прикладні та інформаційні інфраструктури. Він забезпечує структурне міждоменне охоплення досліджуваних практик, але не використовується для статистичного оцінювання поширеності окремих програмних продуктів або підходів у дослідницьких організаціях ЄС. Повторне виконання всіх проаналізованих пайплайнів і порівняльне тестування програмних засобів не проводилися; для забезпечення відтворюваності огляду фіксувалися першоджерело, репозиторій, функціональна категорія, дата перевірки та, за наявності, параметри програмного середовища.

Результати

За результатами дослідження було виділено п’ять архітектурних форм дослідницьких пайплайнів за домінантним способом організації процесу: виконуваний обчислювальний робочий процес (EOSC-Life, ESCAPE, EuroScienceGateway); відтворюване віртуальне дослідницьке середовище (PaNOSC); федеративний аналітичний пайплайн (BY-COVID); документований людино-машинний робочий процес (SSH Open Marketplace); пайплайн походження даних і організації знань (OpenAIRE Graph). Ці форми не є взаємовиключними й доповнюються горизонтальними механізмами інтероперабельності розподілених сервісів (ENVRI-FAIR), ідентифікації, метаданих та оцінювання FAIR-відповідності (FAIR-IMPACT), якості дослідницького програмного забезпечення (EVERSE) і поєднання та динамічного розгортання сервісів (EOSC Beyond).

Порівняння практик показує, що відтворюваність забезпечується функціональним поєднанням спеціалізованих засобів і процедур, а не одним програмним продуктом. У розподілених інфраструктурах вона додатково залежить від стандартизованих метаданих і сумісності сервісів, а в інформаційних пайплайнах – від постійної агрегації, очищення, дедуплікації, збагачення та зв’язування наукових цифрових об’єктів. Показовим для наскрізної інтеграції є BY-COVID – його пайплайн поєднує формулювання дослідницького питання, причинне моделювання, визначення вимог до даних, спільну модель даних, синтетичні дані, локальний аналіз у захищених середовищах і порівняння агрегованих результатів, а репозиторій містить RO-Crate, сценарії Quarto, код R та зафіксоване середовище renv, що демонструє включення до відтворюваного процесу не лише програмних, а й концептуальних, семантичних та експертних переходів.

EOSC-Life демонструє одну з найбільш повних архітектур обчислювальної відтворюваності – мова Common Workflow Language використовується як стандарт опису інструментів і робочих процесів, що їх можуть виконувати різні рушії, зокрема Nextflow і Snakemake. Для реєстрації, опису й публікації робочих процесів було створено WorkflowHub. На проміжному етапі проєкту в ньому було зареєстровано 250 робочих процесів, що охоплювали понад 100 дослідницьких груп і проєктів [5]. ENVRI-FAIR представляє іншу модель, де відтворюваність забезпечується не лише повторним виконанням обчислень, а й стандартизованими метаданими, сумісністю FAIR-сервісів, базою знань та інтеграцією розподілених дослідницьких інфраструктур з EOSC.

Відтворюваний дослідницький пайплайн потребує підтримки на основі десяти основних функціональних категорій засобів, реалізованих переважно на основі вільного програмного забезпечення: документування дослідницьких процедур (прикладом є eLabFTW), підготовки даних (OpenRefine), керування версіями коду, даних і програмного середовища (Git, DVC,

renv), оркестрації залежних операцій (Nextflow, Snakemake, Galaxy, AiiDA), відтворення середовища виконання (Apptainer, Conda, Binder), фіксації походження й пакування метаданих (AiiDA, RO-Crate), тестування та контролю якості (LifeMonitor, засоби безперервної інтеграції), реєстрації і публікації повторно використовуваних об'єктів (WorkflowHub, Zenodo, Quarto), підтримки експертного аналізу (ASReview, QualCoder) та організації знань (Wikibase, бази знань із підтримкою ШІ). Категорії частково перекриваються, оскільки окремі засоби реалізують кілька функцій одночасно. Отже, класифікація відображає функції, які мають бути забезпечені в пайплайні, а не взаємовиключні класи програмних продуктів. Окремою проблемою залишається простежуваність експертних операцій. Так, ASReview може істотно скорочувати обсяг ручного перегляду публікацій, однак алгоритмічне ранжування не замінює фіксації критеріїв і підстав експертного рішення [6].

Окремо проаналізовано засоби організації знань із підтримкою штучного інтелекту. Вони не включалися до основного корпусу практик ЄС без підтвердженого застосування у відповідних проєктах, а використовувалися як технологічна вибірка для аналізу нових можливостей. Порівняння таких систем, як second-brain-os, LLM Wiki, llmwiki-marimo, Holt, HomeKB, показує, що для наукового застосування недостатньо автоматичного формування пов'язаних сторінок. Відтворювана база знань повинна відокремлювати первинні джерела від похідного машинного синтезу і зберігати походження тверджень. Такий процес доцільно представити як послідовність джерело → фрагмент → запропоноване твердження → перевірка → підтверджене твердження → синтезований результат. Перевірка може фіксуватися журналом виконання, результатом тесту, версією критеріїв або системи кодування, експертною міткою чи записом перевірки ШІ-сформованого твердження.

Обговорення

Результати дослідження показують, що практика дослідницьких пайплайнів ЄС ґрунтується не на використанні одного універсального програмного засобу, а на компонуванні спеціалізованих функціональних рівнів. Відповідно, добір вільного програмного забезпечення доцільно здійснювати за повнотою покриття критичних переходів конкретного дослідження. Для кожного такого переходу мають бути зафіксовані версії входу й виходу, операція, виконавець, критерій приймання та результат перевірки. Граф залежностей дає змогу визначати операції, які потребують повторного виконання після зміни даних, коду або параметрів; аналогічний принцип доцільно поширити на експертні та знаннєві об'єкти. Зміна критеріїв включення, системи кодів або джерела має виявляти залежні рішення, фрагменти, твердження та висновки. Для ШІ збереження запиту, ідентифікатора моделі та відповіді забезпечує лише технічну простежуваність, тому відповідь моделі доцільно трактувати як «запропоновану» до перевірки першоджерела або іншого критерію валідації. Така модель підтримує поширення неактуальності за графом залежностей, поєднуючи простежуваність способу отримання і підстав прийняття результату.

Висновки

Дослідження практик ЄС дозволило виокремити основні архітектурні форми відтворюваних дослідницьких пайплайнів і функціональні категорії вільного програмного забезпечення для їх підтримки. Запропоновано розглядати базовою одиницею пайплайну контрольований перехід між версіями даних, моделей і результатів із визначеними операцією, виконавцем, критерієм приймання та результатом перевірки. Такий підхід дає змогу інтегрувати автоматизовані, експертні та ШІ-підтримувані операції, забезпечити наскрізну простежуваність і виявляти залежні результати, що потребують повторної перевірки після змін. Подальша робота має бути спрямована на експериментальну перевірку запропонованої моделі у конкретному дослідницькому пайплайні.

Список використаних джерел

1. Stoudt S., Vásquez V.N., Martinez C.C. Principles for data analysis workflows. PLoS Computational Biology, 2021, vol. 17(3), e1008770. DOI: 10.1371/journal.pcbi.1008770.
2. Barker M., Chue Hong N.P., Katz D.S. et al. Introducing the FAIR Principles for research software. Scientific Data, 2022, vol. 9, article 622. DOI: 10.1038/s41597-022-01710-x.
3. Wilkinson S.R., Aloqalaa M., Belhajjame K. et al. Applying the FAIR Principles to computational workflows. Scientific Data, 2025, vol. 12, article 328. DOI: 10.1038/s41597-025-04451-9.
4. McKinsey & Company. Cutting the ‘coordination tax’: How agentic AI can reshape workflows. 2026. URL: https://www.mckinsey.com/industries/industrials/our-insights/cutting-the-coordination-tax-how-agentic-ai-can-reshape-workflows (дата звернення: 24.09.2026).
5. EOSC-Life. Tools & Workflows. URL: https://www.eosc-life.eu/tools-workflows/ (дата звернення: 24.09.2026).
6. Ferdinands G., Schram R., de Bruin J. et al. Performance of active learning models for screening prioritization in systematic reviews: a simulation study into the Average Time to Discover relevant records. Systematic Reviews, 2023, vol. 12, article 100. DOI: 10.1186/s13643-023-02257-7.

Живило Є. О.
Національний університет «Полтавська політехніка імені Юрія Кондратюка», кафедра комп'ютерних та інформаційних технологій і систем, м. Полтава, Україна
Контактний автор: Євген Живило, e-mail: ziviloevgenij651@gmail.com, ORCID: 0000-0003-4077-7853

АПАРАТНИЙ МЕТОД АДАПТИВНОЇ НЕЙТРАЛІЗАЦІЇ РАДІОЕЛЕКТРОННИХ ЗАВАД У ЗАСОБАХ ТЗІ В УМОВАХ РАДІОЕЛЕКТРОННОГО ПРИДУШЕННЯ

Анотація. У роботі досліджується проблема забезпечення доступності моніторингу технічних каналів витоку інформації (ТКВІ) в умовах радіоелектронного придушення (РЕП), яке може спричиняти насичення приймального тракту, маскування корисного сигналу та збільшення часу його виявлення. Запропоновано апаратний метод адаптивної нейтралізації радіоелектронних завад на базі SoC FPGA із поєднанням алгоритмів LMS та RLS. Критичні за часом операції реалізовано у програмованій логіці із застосуванням DSP48 та конвеєрної обробки. Експериментальна реалізація на Zynq-7020 забезпечила скорочення затримки обробки з 120 до 2,3 мкс, підвищення доступності моніторингу з 87,3 до 99,94 %, збільшення SNR з 4,2 до 11,7 дБ та зменшення енергоспоживання з 3,8 до 1,2 Вт. Отримані результати підтверджують ефективність апаратної реалізації адаптивної фільтрації для зменшення часової складової вікна вразливості та підвищення стійкості функції моніторингу ТКВІ до завадового впливу.

Ключові слова: технічний захист інформації, радіоелектронне придушення, радіоелектронні завади, адаптивна фільтрація, вікно вразливості, доступність моніторингу, детермінована латентність, LMS, RLS, SoC FPGA, Zynq-7020.

Вступ

Сучасні засоби технічного захисту інформації (ТЗІ) значною мірою залежать від здатності систем радіочастотного моніторингу своєчасно виявляти сигнали, потенційно пов'язані з ТКВІ. В умовах РЕП функціонування таких систем може істотно погіршуватися внаслідок насичення приймального тракту, маскування корисного сигналу та підвищення ймовірності його пропуску. Унаслідок цього порушується доступність функції моніторингу на фізичному рівні та формується додаткове вікно вразливості засобів ТЗІ.

Одним із визначальних чинників зазначеної проблеми є латентність обробки сигналів. За програмної реалізації адаптивної фільтрації на ARM-ядрі час реакції залежить не лише від обчислювальної складності алгоритму, а й від планування задач, доступу до пам'яті та особливостей виконання програмного коду. У досліджуваній конфігурації така реалізація характеризується затримкою до 120 мкс, протягом якої корисний сигнал може залишатися невиявленим. Отже, зменшення та стабілізація часу реакції є необхідною умовою підвищення доступності моніторингу в умовах РЕП.

Водночас завадове середовище може характеризуватися як стаціонарними, так і нестаціонарними змінами. Тому для забезпечення адаптивності системи доцільним є поєднання алгоритмів LMS та RLS: перший забезпечує оперативне коригування параметрів за змінних характеристик завади, тоді як другий характеризується вищою швидкістю збіжності за відносно стаціонарного завадового впливу.

З огляду на зазначене виникає необхідність перенесення критичних за часом операцій із програмного середовища ARM до детермінованої апаратної реалізації у програмованій логіці SoC FPGA. Такий підхід дозволяє сформувати фіксовану структуру обчислювального конвеєра та, відповідно, зменшити часову невизначеність реакції системи.

Метою роботи є розроблення та експериментальна перевірка апаратного методу адаптивної нейтралізації радіоелектронних завад на базі SoC FPGA, спрямованого на зменшення латентності обробки, звуження вікна вразливості та підвищення доступності моніторингу ТКВІ в умовах РЕП.

Матеріали та методи

Дослідження виконано на основі апаратно-програмного підходу до адаптивної обробки радіосигналів, у межах якого програмну реалізацію на ARM-ядрі використано як базовий варіант для порівняння з реалізацією критичних операцій у програмованій логіці SoC FPGA.

Для адаптації до нестаціонарних завад застосовано алгоритм LMS (Least Mean Squares) [1], оновлення вектора вагових коефіцієнтів якого визначається співвідношенням:

$$w(n+1) = w(n) + \mu \cdot e(n) \cdot x(n) \quad (1)$$

де $w(n)$ – вектор вагових коефіцієнтів адаптивного фільтра; $x(n)$ – вектор відліків вхідного сигналу; $e(n) = \mathrm{d}(n) - w^T(n) \cdot x(n)$ – похибка адаптації; μ – коефіцієнт кроку адаптації, значення якого масштабується відповідно до дисперсії вхідного сигналу, що забезпечує адаптацію алгоритму до зміни статистичних характеристик завади.

Для обробки завад із відносно стаціонарними статистичними характеристиками використано рекурсивний алгоритм найменших квадратів RLS (Recursive Least Squares). Вектор коефіцієнтів підсилення визначається як:

$$K(n) = \frac{P(n-1) \cdot x(n)}{\lambda + x^T(n) \cdot P(n-1) \cdot x(n)} \quad (2)$$

де $P(n-1)$ – матриця оцінювання попередніх станів, λ – коефіцієнт забування. Оновлення вектора вагових коефіцієнтів здійснюється за співвідношенням:

$$w(n) = w(n-1) + K(n) \cdot e(n) \quad (3)$$

Застосування RLS у складі гібридного алгоритмічного контуру забезпечує швидке уточнення параметрів фільтра за умов, коли статистичні характеристики завадового впливу змінюються повільніше.

Апаратну реалізацію критичних операцій виконано на платформі SoC FPGA Zynq-7020 із використанням блоків DSP48, конвеєрної організації обчислень та потокового інтерфейсу AXI4-Stream [2]. Тактова частота апаратного тракту становить 200 МГц, а пропускна здатність конвеєра – один відлік за такт. У складі апаратно-програмного контуру передбачено взаємодію блоків адаптивної фільтрації, FFT, CORDIC та DMA.

Критичний контур оновлення коефіцієнтів LMS реалізовано мовою SystemVerilog. Синхронне оновлення регістрових коефіцієнтів, конвеєрна структура та відсутність залежних від даних розгалужень забезпечують фіксовану послідовність апаратних операцій і, відповідно, детерміновану часову характеристику обробки.

Для оцінювання ефективності порівнюваних реалізацій використано чотири основні показники: доступність моніторингу, час обробки як часову складову вікна вразливості, співвідношення сигнал/шум (SNR) та енергоспоживання.

Результати

Для кількісного оцінювання впливу затримки обробки на доступність моніторингу у роботі використано показник вікна вразливості засобів ТЗІ:

$$W_v = \tau_{det} \cdot P_{miss} \quad (4)$$

де τ_{det} – час виявлення загрози, P_{miss} – імовірність пропуску цільового сигналу. Таким чином, показник враховує одночасно часову та ймовірнісну складові недоступності функції моніторингу.

Для встановлення залежності вразливості від параметрів процесу виявлення використано функціональне представлення:

$$T_{vuln} = f(\tau_{det} \cdot P_{miss}) \quad (5)$$

Це співвідношення використано як функціональну модель зв’язку між часом виявлення, імовірністю пропуску та інтегральною характеристикою вразливості. У такому представленні скорочення часу реакції безпосередньо зменшує часову складову інтервалу, протягом якого цільовий сигнал може залишатися невиявленим.

Експериментальне порівняння продемонструвало скорочення затримки обробки з 120 до 2,3 мкс, тобто приблизно у 52 рази. Отже, апаратна реалізація забезпечила істотне звуження часової складової вікна вразливості порівняно з програмним виконанням на ARM-ядрі.

Для кількісної оцінки функціональної доступності моніторингу використано співвідношення:

$$A = 1 - \frac{T_{blind}}{T_{obs}} \tag{6}$$

де T_{blind} – сумарна тривалість «сліпих» інтервалів, зумовлених завадовим насиченням приймального тракту, а T_{obs} – загальний час спостереження.

Застосування апаратної реалізації забезпечило збільшення коефіцієнта доступності моніторингу з 87,3 до 99,94 %, тобто на 12,64 процентного пункту. Отриманий результат узгоджується зі зменшенням латентності, оскільки скорочення часу реакції зменшує тривалість інтервалів, протягом яких функція моніторингу є недоступною.

Порівняльні результати наведено в табл. 1.

Таблиця 1 – Порівняння характеристик програмної та апаратної реалізацій

Показник	Програмна реалізація (ARM)	Апаратна реалізація (PL)	Ефект
Коефіцієнт доступності моніторингу, %	87,3	99,94	+12,64 в.п.
Затримка обробки, мкс	120	2,3	-98,08 %
SNR, дБ	4,2	11,7	+7,5 дБ
Енергоспоживання, Вт	3,8	1,2	-68,4 %

Отримані дані демонструють комплексне покращення характеристик апаратної реалізації. Окрім 52-кратного скорочення затримки, зафіксовано збільшення SNR з 4,2 до 11,7 дБ, що відповідає приросту на 7,5 дБ. Це свідчить про покращення умов виділення корисного сигналу після адаптивної обробки.

Додатково енергоспоживання зменшилося з 3,8 до 1,2 Вт, тобто на 68,4 %. Таким чином, апаратне виконання критичного контуру забезпечує не лише часову, але й енергетичну перевагу порівняно з програмною реалізацією.

За результатами синтезу апаратний модуль використовує 10,9 % ресурсів DSP48 та 3,5 % LUT платформи Zynq-7020. Такий рівень використання програмованої логіки формує ресурсний резерв для подальшого розширення функціональності, зокрема реалізації апаратних модулів класифікації завад та оцінювання параметрів загроз.

Обговорення

Отримані результати свідчать, що ефективність систем моніторингу ТКВІ в умовах РЕП доцільно оцінювати не лише за здатністю адаптивного фільтра послаблювати завадовий компонент, а й за часовою характеристикою реакції. Для засобів ТЗІ суттєвим є не тільки досягнення необхідного SNR, але й забезпечення своєчасного виявлення цільового сигналу, оскільки надмірна латентність безпосередньо збільшує інтервал потенційної недоступності моніторингу.

Порівняння ARM- та PL-реалізацій демонструє, що основною перевагою запропонованого апаратного підходу є не лише прискорення обчислень, а й формування детермінованої часової поведінки критичного контуру. На відміну від програмного виконання, час проходження сигналу через апаратний конвеєр визначається його структурою та фіксованою кількістю тактів, що зменшує залежність латентності від планування задач і доступу до пам’яті.

Поєднання LMS та RLS додатково забезпечує адаптацію до різних режимів завадового середовища. LMS доцільний для оперативного реагування на нестаціонарні зміни, тоді як RLS забезпечує швидке уточнення параметрів за відносно стаціонарного характеру завади. У поєднанні з апаратною реалізацією це формує основу для побудови адаптивного контуру моніторингу з передбачуваними часовими характеристиками.

Водночас отримані результати необхідно інтерпретувати з урахуванням обмежень дослідження. Експериментальну перевірку виконано на конкретній платформі Zynq-7020, тому абсолютні значення латентності, енергоспоживання та використання апаратних ресурсів не можуть безпосередньо переноситися на інші FPGA/SoC-платформи без

додаткової верифікації. Крім того, для повної оцінки робастності методу необхідне розширення набору тестових сценаріїв із різними типами, інтенсивністю та спектральними характеристиками РЕП.

Висновки

Розроблено апаратний метод адаптивної нейтралізації радіоелектронних завад у засобах ТЗІ, який поєднує алгоритми LMS та RLS із реалізацією критичних за часом операцій у програмованій логіці SoC FPGA.

Формалізовано показник вікна вразливості, що враховує часову та ймовірнісну складові недоступності моніторингу, а також коефіцієнт функціональної доступності системи моніторингу ТКВІ.

Експериментальна реалізація на Zynq-7020 забезпечила скорочення затримки обробки з 120 до 2,3 мкс, підвищення доступності моніторингу з 87,3 до 99,94 %, збільшення SNR з 4,2 до 11,7 дБ та зменшення енергоспоживання з 3,8 до 1,2 Вт.

Отримані результати підтверджують, що перенесення критичних операцій адаптивної фільтрації з ARM-ядра до програмованої логіки дозволяє одночасно зменшити латентність, звузити часову складову вікна вразливості та підвищити доступність моніторингу ТКВІ за досліджуваних умов РЕП.

Подальші дослідження доцільно спрямувати на параметризацію математичної моделі вікна вразливості, розширення набору сценаріїв РЕП, статистичну валідацію результатів та інтеграцію апаратних модулів класифікації завад і оцінювання параметрів загроз.

Список використаних джерел

1. Lu, Q., Qin, H., He, F., et al. Wideband Interference Cancellation System Based on a Fast and Robust LMS Algorithm. Sensors. 2023. Vol. 23, No. 18. Article 7871. DOI: 10.3390/s23187871.

2. Digital Signal Processing – DSP Slice // Zynq 7000 SoC Technical Reference Manual (UG585) / AMD Technical Information Portal. 2026. URL: https://docs.amd.com/r/en-US/ug585-zynq-7000-SoC-TRM/Digital-Signal-Processing-DSP-Slice (дата звернення: 15.09.2026).

Калашнікова Ю. В.
Національний університет «Полтавська політехніка імені Юрія Кондратюка», кафедра комп'ютерних та інформаційних технологій і систем, м. Полтава, Україна
Контактний автор: Юлія Калашнікова, e-mail: kalashjulia74@gmail.com, ORCID: 0000-0001-9899-4784

ДЕТЕРМІНОВАНА АРХІТЕКТУРА ЗАСОБІВ ТЗІ ДЛЯ МІНІМІЗАЦІЇ ВІКНА ВРАЗЛИВОСТІ ОКІ

Анотація. Забезпечення конфіденційності, цілісності та доступності інформації на об'єктах критичної інфраструктури (ОКІ) потребує не лише підвищення точності виявлення загроз, а й мінімізації часової затримки між їх виникненням, виявленням і формуванням реакції засобів технічного захисту інформації (ТЗІ). Запропоновано детерміновану архітектуру ТЗІ, спрямовану на зменшення часового вікна вразливості за рахунок багатоканального апаратного паралелізму, потокового цифрового оброблення та детермінованої міжканальної синхронізації.

Критичні операції моніторингу, цифрового оброблення та первинного виявлення реалізуються паралельно у програмованій логіці SoC FPGA, тоді як ARM/PS-компонент забезпечує формування керуючих рішень і реєстрацію результатів. Для кількісного оцінювання ефективності архітектури використано показники часу вразливості, частки часу неповного контролю, ймовірності пропуску події та інтегрального показника вразливості. Експериментальне дослідження на платформі Zynq-7020 показало зростання ймовірності виявлення з 0,45 до 0,97, зменшення частки часу неповного контролю з 35 % до менш ніж 3 % та підвищення пропускної здатності цифрового тракту з 327 до 980 МБ/с. При цьому інтегральний показник вразливості зменшився з 0,1925 до менш ніж 0,0009.

Ключові слова: технічний захист інформації, критична інфраструктура, вікно вразливості, детермінована архітектура, апаратний паралелізм, потокове цифрове оброблення, багатоканальний моніторинг, радіомоніторинг, SoC FPGA.

Вступ

Забезпечення конфіденційності, цілісності та доступності інформації на ОКІ потребує не лише підвищення точності виявлення загроз, а й забезпечення прогнозованого часу їх виявлення та нейтралізації. Для інформаційно-технічних систем із безперервним моніторингом часові характеристики приймального, цифрового та керуючого трактів безпосередньо визначають своєчасність реагування на події інформаційної безпеки.

У традиційних архітектурах засобів ТЗІ моніторинг може реалізовуватися шляхом послідовного опитування сенсорів, часового мультиплексування каналів і програмного формування керуючих рішень. Зі збільшенням кількості контрольованих каналів така організація зумовлює виникнення інтервалів неповного контролю, протягом яких окремі просторово-частотні області інформаційного середовища залишаються поза поточним спостереженням. У результаті формується часове вікно вразливості, протягом якого інформаційна подія може залишитися невиявленою або її нейтралізація може бути несвоєчасною.

Для кількісного опису часових витрат системи від моменту виникнення події до формування реакції введемо часовий інтервал вразливості:

$$T_v = T_d + T_p + T_r, \quad (1)$$

де T_d – час виявлення потенційної загрози; T_p – час первинного цифрового оброблення та формування рішення; T_r – час формування та реалізації реакції засобу ТЗІ.

Особливого значення мінімізація цього інтервалу набуває під час моніторингу динамічних джерел радіовипромінювання, зокрема безпілотних літальних апаратів із псевдовипадковою (стрибкоподібною) перебудовою робочої частоти (СППЧ). Короткотривалість інформаційних подій і просторово-частотна мінливість таких джерел підвищують вимоги до безперервності, швидкодії та часової детермінованості моніторингу.

Для високошвидкісних багатоканальних систем додатковою умовою є синхронність функціонування окремих приймальних трактів. Зокрема, у JESD204B Subclass 1 сигнал

SYSREF використовується як системна часова опора для забезпечення детермінованої затримки та узгодження моментів оброблення між пристроями [1].

Метою роботи є розроблення детермінованої архітектури засобів ТЗІ для мінімізації часового вікна вразливості ОКІ шляхом поєднання багатоканального апаратного паралелізму, потокового цифрового оброблення, детермінованого міжканального синхронізування та скорочення часу реакції на потенційну загрозу.

Матеріали та методи

Методологічною основою дослідження є перехід від послідовного опитування каналів до просторового та функціонального апаратного паралелізму. Для кількісного оцінювання безперервності моніторингу введено нормовану частку часу неповного контролю:

$$S_{inc} = \frac{T_{inc}}{T_{obs}}, \qquad (2)$$

де T_{inc} – сумарний час неповного контролю, T_{obs} – загальний час спостереження.

Для врахування ймовірності пропуску інформаційної події додатково визначено:

$$P_{miss} = 1 - P_{det}. \qquad (3)$$

де P_{miss} – ймовірність виявлення потенційної події.

Тоді інтегральний показник вразливості може бути представлений як

$$I_v = S_{inc} \cdot P_{miss}, \qquad (4)$$

Зменшення I_v відповідає одночасному підвищенню безперервності моніторингу та зниженню ймовірності пропуску події. Такий підхід дає змогу пов’язати архітектурні характеристики засобу ТЗІ з кількісно вимірюваними параметрами його функціонування.

Запропонована архітектура (Рис. 1) містить багатоканальну антенну підсистему, синхронізовані приймальні тракти, аналого-цифрові перетворювачі, високошвидкісні інтерфейси передавання даних, паралельні блоки цифрового оброблення, апаратні модулі виявлення та попередньої класифікації, підсистему просторового аналізу, ARM/PS-компонент формування керуючих рішень і підсистему реєстрації та передавання результатів.

На рівні програмованої логіки критичні операції виконуються паралельно для всіх каналів. Детермінованість часових характеристик забезпечується синхронізацією тактових сигналів, фіксованою структурою конвеєрів, синхронним формуванням відліків і контролем заповнення буферів. Для високошвидкісних приймальних трактів використовується інтерфейс JESD204B із сигналом SYSREF. У режимі Subclass 1 сигнал SYSREF використовується для узгодження локальних багатофреймових тактових подій (LMFC), що забезпечує повторюваність часових характеристик передавання даних [1].

Водночас синхронізація сама по собі не гарантує просторової узгодженості каналів.

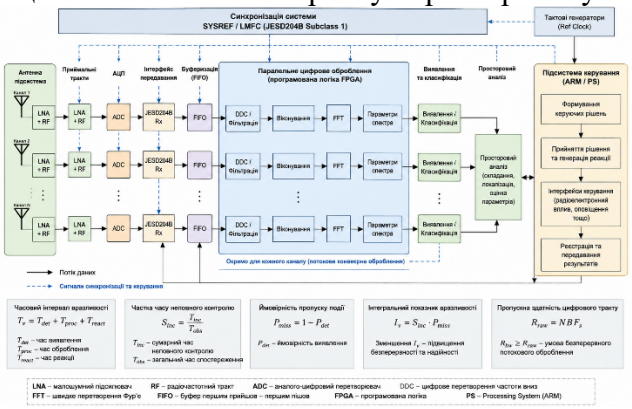


Рис. 1 – Детермінована багатоканальна архітектура ТЗІ

Фазова точність залежить від характеристик антен, аналогових приймальних трактів, тактових джерел і результатів калібрування. Тому синхронізацію розглянуто як складову комплексного забезпечення детермінованої роботи багатоканальної системи.

Для (N) приймальних каналів із частотою дискретизації (F_s) та розрядністю одного цифрового відліку (B) біт сумарна швидкість необробленого цифрового потоку визначається як

$$R_{raw} = NBF_s. \qquad (5)$$

де R_{raw} – сумарна швидкість необробленого цифрового потоку, N – кількість каналів, B – кількість бітів на один відлік одного каналу, F_s – частота дискретизації.

Для забезпечення безперервного потокового моніторингу гарантована пропускна здатність цифрового тракту повинна задовольняти умову

$$R_{hw} \geq R_{raw}, \qquad (6)$$

де R_{hw} – гарантована пропускна здатність апаратного тракту цифрового оброблення. Порушення умови (6) спричиняє накопичення даних у буферах, збільшення затримки оброблення та потенційну втрату відліків. Отже, достатня пропускна здатність цифрового тракту є необхідною умовою безперервного функціонування засобу ТЗІ.

Середовище відтворення

Експериментальне оцінювання запропонованої архітектури виконано на платформі SoC FPGA Zynq-7020 (XC7Z020). Вибір платформи зумовлений наявністю програмованої логіки FPGA та ARM/PS-компонента, що дає змогу реалізувати паралельні апаратні модулі цифрового оброблення і програмне формування керуючих рішень в межах єдиної системи. Підтримка пристрою XC7Z020 засобами Vivado документована виробником [2].

Для забезпечення відтворюваності цифрової частини дослідження використано програмне та апаратне середовище, наведене в табл. 1.

Таблиця 1 – Середовище відтворення дослідження

Компонент	**Конфігурація**	**Призначення**
FPGA	AMD/Xilinx Zynq-7020, XC7Z020	апаратна реалізація
Vivado	2024.2.2	synthesis, implementation, bitstream
Vitis	2024.2	ARM/PS software component
Verilator	5.052	FOSS RTL simulation/lint
Yosys	OSS CAD Suite / стабільна версія	FOSS RTL synthesis/аналіз
GTKWave	3.3.128	перегляд часових діаграм
HDL	Verilog/SystemVerilog	опис паралельних модулів
ОС	Ubuntu 24.04 LTS	середовище FOSS-інструментів

Застосування Verilator, Yosys та GTKWave дозволяє перенести значну частину RTL-верифікації, синтезу та аналізу часових діаграм у відкрите програмне середовище, тоді як Vivado/Vitis використовуються для цільової реалізації на платформі Zynq-7000 [2, 3].

Результати

Для верифікації запропонованого підходу виконано порівняння базової архітектури з послідовним обслуговуванням каналів та запропонованої багатоканальної архітектури з апаратним паралелізмом на платформі SoC FPGA Zynq-7020. Основні результати порівняння наведено в табл. 2.

Таблиця 2 – Порівняльні характеристики архітектур

Показник	Базова архітектура	Запропонована архітектура
Ймовірність виявлення P_{det}	0,45	0,97
Частка часу неповного контролю S_{inc}	35 %	< 3 %
Пропускна здатність цифрового тракту R_{hw}	327 МБ/с	980 МБ/с
Інтегральний показник I_v	0,1925	< 0,0009

Отримані результати свідчать про покращення всіх досліджуваних характеристик. Ймовірність виявлення зросла з 0,45 до 0,97, тоді як частка часу неповного контролю зменшилася з 35 % до менш ніж 3 %. Таким чином, апаратний паралелізм забезпечує підвищення безперервності моніторингу та зменшення ймовірності пропуску інформаційної події.

Пропускна здатність цифрового тракту зросла з 327 до 980 МБ/с, тобто приблизно утричі. Це підвищує спроможність системи обробляти багатоканальний потік у реальному часі та зменшує ризик накопичення даних у буферах за умови виконання вимоги (6).

Найбільш виражена зміна спостерігається для інтегрального показника вразливості. Для базової архітектури отримано $I_v = 0,35(1-0,45) = 0,1925$, тоді як для запропонованої – $I_v < 0,03(1-0,97) < 0,0009.$ Отже, за наведеними межами значення показника зменшується понад у 213 разів.

Таким чином, результати підтверджують, що зменшення часової складової вразливості визначається не лише збільшенням кількості каналів, а комплексною організацією їх синхронного функціонування, потокового цифрового оброблення та апаратного виявлення. Саме поєднання цих механізмів забезпечує одночасне підвищення повноти контролю, пропускної здатності та часової детермінованості засобу ТЗІ.

Наукова новизна

Наукова новизна полягає в удосконаленні архітектури засобів ТЗІ шляхом інтеграції просторового апаратного паралелізму, незалежного потокового цифрового оброблення сигналів, детермінованого міжканального синхронізування та формалізованого оцінювання часової складової вразливості. На відміну від послідовної організації моніторингу, запропонований підхід встановлює безпосередній зв'язок між структурною організацією апаратного тракту та кількісними показниками безперервності контролю, ймовірності виявлення подій і інтегрального рівня вразливості.

Практичне значення

Практичне значення одержаних результатів полягає у можливості застосування запропонованого архітектурного принципу під час модернізації засобів моніторингу ОКІ, для яких критичними є безперервність контролю, детермінованість часу оброблення та висока пропускна здатність цифрового тракту. Архітектура може масштабуватися за кількістю приймальних каналів за умови відповідного забезпечення ресурсів FPGA, пропускної здатності міжз'єднань і енергетичних характеристик платформи.

Обговорення

Отримані результати дають підстави розглядати часове вікно вразливості не лише як характеристику алгоритму виявлення, а як комплексний архітектурний параметр, сформований часовою поведінкою приймального, цифрового та керуючого трактів. Відповідно до (1), його величина визначається сукупністю часу виявлення, первинного цифрового оброблення та реалізації реакції.

У послідовній архітектурі зі збільшенням кількості каналів може зростати тривалість циклу їх обслуговування, що зумовлює збільшення інтервалів неповного контролю. Тому саме збільшення кількості сенсорів не гарантує зменшення часового вікна вразливості. Запропонований підхід послаблює цю залежність на критичних ділянках тракту завдяки паралельному виконанню операцій моніторингу та цифрового оброблення.

Водночас ефективність апаратного паралелізму обмежується пропускною здатністю всього тракту. За умови порушення (6) виникає накопичення даних у буферах, що збільшує затримку оброблення та може спричинити втрату відліків. Отже, апаратний паралелізм має розглядатися у взаємозв'язку з пропускною здатністю тракту, а умова (6) є необхідною для забезпечення безперервного потокового моніторингу.

Окремим фактором детермінованості є міжканальна синхронізація. У JESD204B Subclass 1 сигнал SYSREF використовується для формування системної часової опори та узгодження LMFC, що забезпечує повторюваність часових характеристик багатоканального тракту [1]. Водночас синхронізація не усуває фазових похибок, зумовлених характеристиками антен, аналогових приймальних трактів, тактових джерел і калібруванням. Тому часову та просторову узгодженість каналів доцільно розглядати як взаємопов'язані, але не тотожні характеристики.

Подальші дослідження доцільно спрямувати на масштабування кількості каналів, експериментальне визначення залежності R_{raw} від N, B та F_s , вимірювання фактичної латентності кожного конвеєра, оцінювання використання ресурсів FPGA та енергоспоживання, а також визначення впливу калібрувальних похибок на просторовий аналіз. Окремим напрямом є експериментальна перевірка запропонованої архітектури для мобільних джерел радіовипромінювання та сигналів зі СППЧ.

Висновки

У роботі запропоновано детерміновану архітектуру засобів технічного захисту інформації для мінімізації часового вікна вразливості об’єктів критичної інфраструктури. Рішення ґрунтується на багатоканальному апаратному паралелізмі, потоковому цифровому обробленні, детермінованій міжканальній синхронізації та реалізації критичних операцій у програмованій логіці SoC FPGA.

Результати підтверджують, що скорочення вікна вразливості визначається не лише алгоритмом виявлення, а й архітектурою тракту моніторингу. Перехід до синхронного багатоканального потокового оброблення забезпечує передумови для підвищення безперервності контролю та скорочення часу реакції.

Подальші дослідження доцільно зосередити на масштабуванні кількості каналів, оптимізації ресурсів FPGA та енергоспоживання, калібруванні, визначенні детермінованої латентності та експериментальній перевірці архітектури для динамічних джерел радіовипромінювання, зокрема сигналів зі СППЧ.

Список використаних джерел

1. Analog Devices. JESD204B Subclasses-Part 1: An Introduction to JESD204B Subclasses and Deterministic Latency. Analog Devices: Technical Articles. 2019. URL: https://www.analog.com/en/resources/technical-articles/jesd204b-subclasses-part1-an-introduction-to-jesd204b-subclasses-and-deterministic-latency.html (дата звернення: 16.09.2026).

2. AMD. Supported Devices. UG973: Vivado Design Suite User Guide: Release Notes, Installation, and Licensing (v2024.2). 2024. URL: https://docs.amd.com/r/2024.2-English/ug973-vivado-release-notes-install-license/Supported-Devices (дата звернення: 16.09.2026).

3. Snyder W. Overview. Verilator 5.052 documentation. 2026. URL: https://veripool.org/guide/latest/overview.html (дата звернення: 16.09.2026).

Клочко К. М., Роговенко А. І.
Національний університет «Чернігівська політехніка», кафедра інформаційних та комп'ютерних систем, м. Чернігів, Україна
Контактний автор: Костянтин Клочко, e-mail: kostya_klochko@ukr.net, ORCID: 0009-0003-8555-8723

РОЗРОБКА ТА ДОСЛІДЖЕННЯ ДЕЦЕНТРАЛІЗОВАНОЇ ВЕБСИСТЕМИ КАТАЛОГІЗАЦІЇ ЕЛЕКТРОННИХ ПУБЛІКАЦІЙ ЗАСОБАМИ ELIXIR

Анотація. У тезах розглядається розробка децентралізованої вебсистеми каталогізації наукових публікацій та збору метаданих (DPI) з нульовою довірою між автономними вузлами. Проаналізовано обмеження традиційних централізованих бібліографічних баз даних (SPOF, комерційний вендор-лок, затримка верифікації та ризики спотворення даних). Запропоновано комплексне архітектурне та криптографічне рішення на основі акторної парадигми BEAM (Elixir), декларативного фреймворку Ash 3.0 та вебтехнології Phoenix LiveView. Реалізовано асиметричну автентифікацію міжвузлових HTTP-запитів на базі підписів Ed25519 із захистом Anti-Replay, конвеєр збору даних із каскадною дедуплікацією (ORCID, DOI, ISBN, відстань Левенштейна), модель відкладеної модерації (Change Proposals / JSON-дельти) та механізм аварійної ізоляції скомпрометованих вузлів («Unplug Kill-Switch»).

Ключові слова: децентралізовані системи; наукові публікації; Elixir; Ash Framework; Phoenix LiveView; Ed25519; дедуплікація; ORCID; Oban; PostgreSQL.

Вступ

Сучасний розвиток відкритої науки (Open Science) вимагає безперервного, оперативного та стійкого обліку наукометричної інформації. Традиційні комерційні та централізовані платформи (Scopus, Web of Science, Google Scholar) мають низку системних вад: наявність єдиної точки відмови (SPOF), закритість або висока вартість API, тривала адміністративна затримка при модерації та відсутність криптографічних гарантій авторства при розподіленому вводі даних.

Спроби вирішення цих проблем через федеративні каталоги стикаються з новими викликами: відсутність довіри між автономними серверами, загрози ін'єкції сфальсифікованих метаданих (Metadata Poisoning), атаки повторного відтворення (Replay Attacks) та маніпуляції ідентифікаторами. Метою дослідження є проєктування, розробка та аналіз децентралізованої вебсистеми DPI (Decentralized Publication Index), яка забезпечує автоматизований збір, дедуплікацію та взаємну синхронізацію метаданих електронних публікацій за моделлю нульової довіри (Zero-Trust) без використання ресурсомістких блокчейн-консенсусів або центральних регуляторів.

Матеріали та методи

Для побудови системи обрано архітектурний патерн «модульний моноліт» на базі акторної моделі віртуальної машини Erlang (BEAM) та мови Elixir [1, 2]. Система складається з чотирьох основних шарів:

1. Шар користувацького інтерфейсу (Presentation Layer): реактивний вебінтерфейс на Phoenix LiveView [2, 3] із передачею мінімальних бінарних diff-пакетів через WebSockets та стилізацією Tailwind CSS / Flowbite.

2. Декларативний доменний шар (Domain Resource Layer): побудований на Ash Framework 3.0 [4], де сутності публікацій (Publication), авторів (Author), ролей (AuthorRole), поліморфних ідентифікаторів (PublicationId) та серверів-партнерів (DPIServer) описуються декларативними ресурсами з матрицею безпеки Ash.Policy.Authorizer.

3. Шар криптографічної безпеки та міжвузлового обміну (Federation & Security Layer): забезпечення автентичності та фонової синхронізації через транзакційні черги Oban [5] у PostgreSQL [6].

4. Шар інгестії та дедуплікації (Ingestion Layer): адаптери зовнішніх API (ORCID API v3.0, Scopus), нормалізатори ідентифікаторів та каскадний алгоритм зіставлення записів.

Криптографічний захист міжвузлового обміну спирається на асиметричні підписи Ed25519 (RFC 8032) [7]. Кожен вузол при відправці HTTP-пакета підписує приватним ключем sk_node зведений геш:

σ = Sign_sk(SHA-512(ID_server || request_timestamp || Payload))

Отримувач виконує двоетапну верифікацію:

1. перевірку часового вікна Anti-Replay (|timestamp_now-request_timestamp| ≤ 300 с)

2. перевірку підпису σ публічним ключем pk_node.

Конвеєр збору та дедуплікації включає алгоритмічну валідацію контрольних цифр (ORCID, EAN-13 для ISBN), нормалізацію DOI та каскадну схему виявлення дублікатів:

1. Пошук точного збігу за канонічним DOI;

2. Пошук точного збігу за нормалізованим ISBN-13;

3. Нечітке зіставлення (Fuzzy Matching) назв за модифікованою відстанню Левенштейна [8] у комбінації з роком виходу та прізвищем першого автора.

Модель безпечної модерації реалізовано через стейт-машину пропозицій змін (ChangeProposal). Зміни неавторизованих користувачів зберігаються як JSON-дельти у стані pending та застосовуються атомарно лише після схвалення модератором. Захисний механізм «Unplug Kill-Switch» дозволяє адміністратору миттєво перевести сервер у стан :unplug, зупиняючи черги Oban, блокуючи вхідні запити (403 Forbidden) та каскадно видаляючи чужорідний спам без втрати локальних даних.

Результати

Розроблений програмний комплекс DPI забезпечує повнофункціональну каталогізацію наукових праць. Локальне збереження реплікованих даних у базі PostgreSQL гарантує субмілісекундну швидкість відгуку при пошуку та фільтрації незалежно від доступності зовнішніх мережевих вузлів.

Обговорення

Запропонований підхід доводить можливість створення стійкої федеративної системи без використання блокчейн-консенсусу, який вимагає високих обчислювальних витрат. Застосування підписів Ed25519 та часових вікон Anti-Replay забезпечує невідмовність транзакцій та захист від сурогатних вузлів. Завдяки високій продуктивності платформи BEAM та згортці реактивного стану в Phoenix LiveView, накладні витрати на серверне обладнання мінімізовані.

Основним обмеженням системи є необхідність первинної довіри при обміні публічними ключами pk_node між адміністраторами партнерських вузлів та орієнтація на почесну синхронізацію джерел істини. Подальший розвиток системи передбачає проведення навантажувального тестування у кластерах із сотень вузлів, впровадження семантичного пошуку на основі векторних ембеддінгів (LLM / pgvector) та інтеграцію протоколу W3C ActivityPub для сумісності з Fediverse.

Висновки

1. Розроблено децентралізовану вебсистему каталогізації наукових публікацій на основі Elixir, Ash 3.0 та Phoenix LiveView, яка поєднує інституційну автономність з високою швидкістю локального пошуку.

2. Реалізовано криптографічний протокол автентифікації на еліптичних кривих Ed25519 із захистом Anti-Replay, який забезпечує принципи Zero-Trust між автономними вузлами.

3. Створено конвеєр збору та каскадної дедуплікації даних на основі нормалізації ідентифікаторів та нечіткого порівняння назв публікацій за відстанню Левенштейна.

4. Впроваджено модель неруйнівної модерації змін ChangeProposal та захисний перемикач «Unplug Kill-Switch» для оперативної ізоляції скомпрометованих серверів та видалення спаму.

Список використаних джерел

1. The Elixir programming language. URL: https://elixir-lang.org/ (дата звернення: 24.09.2026).

2. Tate B., DeBenedetto S. Programming Phoenix LiveView: Interactive Elixir web programming without writing any JavaScript. Pragmatic Bookshelf, 2025. 320 p.
3. Phoenix Framework. URL: https://www.phoenixframework.org/ (дата звернення: 24.09.2026).
4. Ash Framework. URL: https://ash-hq.org/ (дата звернення: 24.09.2026).
5. Oban Pro: Robust job processing for Elixir. URL: https://oban.pro/ (дата звернення: 24.09.2026).
6. PostgreSQL documentation. URL: https://www.postgresql.org/ (дата звернення: 24.09.2026).
7. Josefsson S., Liusvaara I. Edwards-Curve Digital Signature Algorithm (EdDSA). IETF RFC 8032, 2017. URL: https://www.rfc-editor.org/info/rfc8032 (дата звернення: 24.09.2026).
8. Tkaczyk D., Szostek P., Fedoryszak M., et al. CERMINE: automatic extraction of structured metadata from scientific literature. International Journal on Digital Libraries, 2015. Vol. 16, No. 3-4. P. 247–266.

Кондратюк Д. С., Казимир В. В.
Національний університет «Чернігівська політехніка», кафедра інформаційних та комп'ютерних систем, Чернігів, Україна
Контактний автор: Данило Кондратюк, e-mail: kondratiuk2001@gmail.com

СЕМАНТИЧНИЙ КОНТРАКТ ПОДІЙ У ВЗАЄМОДІЇ IoT ТА ERP СИСТЕМ / A SEMANTIC EVENT CONTRACT FOR IoT–ERP INTERACTION

Анотація. У взаємодії систем Інтернету речей (IoT) та планування ресурсів підприємства (ERP) до корпоративного рівня зазвичай надходить бінарна подія (можлива відмова) з оцінкою ризику, якої достатньо для заявки на обслуговування, але не для вибору конкретної дії. У роботі подію формалізовано як семантичний контракт, де є мінімальний набір атрибутів (тип події, ризик, причина, політика рішення), достатній, щоб ERP обрала правильну дію. На двох відкритих датасетах прогнозного обслуговування, AI4I 2020 та Microsoft Azure PdM, показано, що тип події відновлюється за тими самими ознаками, що й ризик, у 93–98 % випадків і додає близько 120 байтів на подію. Виведено умову окупності типізованої події, яка пов'язує її вигоду з часткою хибних тривог детектора. Усі обчислення виконано вільним програмним забезпеченням і оформлено відтворюваним скриптом.

Ключові слова: Інтернет речей; ERP-системи; прогнозне обслуговування; семантика подій; edge–cloud; відкрите ПЗ; scikit-learn.

Вступ

У попередніх моїх роботах взаємодію IoT та ERP подано як конвеєр edge–cloud–ERP, де до ERP надходить не телеметрія, а потік подій [1] [2]. Проте зміст події залишався поза увагою, адже вона бінарна («можлива відмова» з оцінкою ризику). Цього досить для інформування, проте не для вибору дії: замінити інструмент, перевірити охолодження чи скоригувати режим. Я віддавав перевагу локальній діагностиці та вирішенню проблеми на місці. Мета цієї роботи — формалізувати зміст події між хмарним і корпоративним рівнями та кількісно оцінити, коли її збагачення окупається.

Матеріали та методи

Контракт події. Подію розглянемо як семантичний контракт (узгоджений між рівнями набір атрибутів), достатній для вибору дії ERP [3]. Подамо її як кортеж

$$e = \langle m, t, \tau, r, c, \pi \rangle$$

Де: m — одиниця обладнання; t — час; τ — тип події (режим відмови); r — оцінка ризику; c — причина, тобто k ознак із найбільшим відхиленням; π — політика рішення (поріг θ, вартості C_{FN} і C_{FP}, версія моделі). Тип τ задає відображення $\tau \to a(\tau)$ у конкретну дію ERP (табл. 1). Причина c робить рішення перевірним для інженера, а політика π — простежуваним.

Двоступенева хмара та дані. Хмару розділено на два ступені: детектор (Random Forest; поріг θ мінімізує витрати $J(\theta) = C_{FN} \cdot FN(\theta) + C_{FP} \cdot FP(\theta)$ [1]) і новий типізатор, що приписує події тип. Перевірку виконано на двох датасетах: AI4I 2020 — 10 000 зразків фрезерного верстата з позначкою режиму відмови [4]; Azure PdM — 100 машин, рік погодинної телеметрії, позначка компонента, що відмовив [5]. На другому групова крос-валідація за машинами, тож моделі перевіряються на «незнайомому» обладнанні, адже поріг θ щоразу добирають лише за навчальними фолдами, тож дані оцінювання в нього не потрапляють. Скрізь $C_{FN} = 10$, $C_{FP} = 1$.

Відтворюваність. Усі обчислення виконано вільним програмним забезпеченням: Python 3 зі scikit-learn, pandas і matplotlib; обидва набори даних відкриті, а весь експеримент оформлено одним скриптом із фіксованими початковими значеннями генератора випадкових чисел, тож наведені нижче числа відтворюються повторним запуском [6].

Результати

Тип відновлюється за тими самими ознаками, що й ризик: 98 % на AI4I 2020 і 93 % на Azure PdM проти 35–36 % у найчастішого класу (табл. 2). За режимами картина неоднорідна

(табл. 1): тепловідвід, потужність і перевантаження типізуються майже безпомилково, а знос інструмента слабкий — до ERP доходять лише 5 подій із 46, і правильно названо 3 з них. Цей режим за побудовою набору не детермінований вимірюваннями, як і суто випадкові відмови. Контракт наповнюваний і коштує мало: типізована подія займає 171 байт JSON проти 54 у бінарної події.

Таблиця 1 – Відображення «тип події → дія ERP» та якість типізації (AI4I 2020)

Тип події τ	**Дія ERP a(τ)**	**Передано до ERP**	**Правильно типізовано**
TWF — знос інструмента	заміна інструмента	5 / 46	3 / 5
HDF — тепловідвід	обслуговування охолодження	114 / 115	100 %
PWF — відхилення потужності	діагностика приводу	95 / 95	98 %
OSF — перевантаження	корекція режиму	98 / 98	98 %

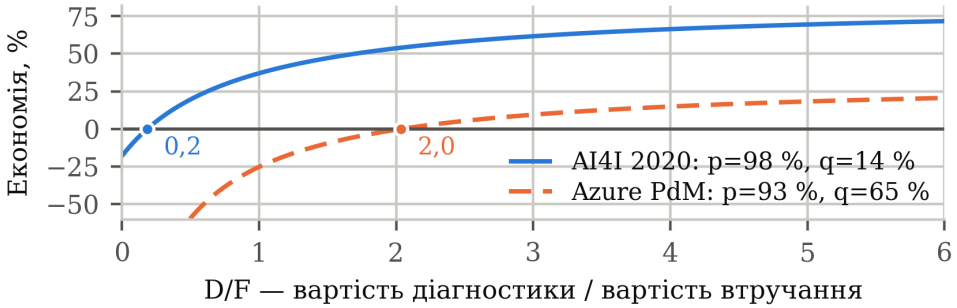
Виграш оцінимо через дії ERP. Нехай p — частка правильно типізованих подій, а q — частка хибних тривог серед переданих. За бінарною подією виконують діагностику (D) і, за підтвердження, усунення (F); за типізованою — усунення одразу, тож хибний тип чи хибна тривога додають марне втручання. Порівняння очікуваних витрат дає умову окупності

$$D/F > [q/(1 - q) + (1 - p)]/p$$

За тієї самої структури вартостей на AI4I поріг $D/F > 0{,}2$ і економія 37–62 % при $D/F = 1…3$, а на Azure PdM — уже $D/F > 2{,}0$, тож за дешевої діагностики типізоване рішення програє бінарному (табл. 2, рис. 1).

Таблиця 2 – Наповнення контракту подій на двох еталонних наборах

Показник	**AI4I 2020**	**Azure PdM**
Типи подій τ	TWF, HDF, PWF, OSF	comp1 – comp4
Подій до ERP (хибних тривог q)	334 (14 %)	1 736 (65 %)
Правильно типізовано p	98 ± 2 %	93 ± 1 %
Поріг окупності D/F	0,2	2,0



Обговорення

Вирішальним для контракту виявляється не якість типізатора, а влучність детектора: типізація на обох датасетах висока й майже однакова, тоді як поріг окупності відрізняється на порядок саме через частку хибних тривог (14 % проти 65 %). Причина асиметрії проста: за бінарною подією хибна тривога коштує лише діагностики, а за типізованою — ще й марного втручання. Звідси практичний висновок: збагачувати подію типом варто там, де діагностика дорога відносно втручання, або де детектор «розумний»; інакше раціональнішою є гібридна політика — адресна дія за високої впевненості та діагностика за низької.

Обмеження. Числа отримано за однієї структури вартостей ($C_{FN} = 10$, $C_{FP} = 1$), а вартості D та F вважаються однаковими для всіх типів події. Контракт може нести лише ту семантику, яка детермінована спостережуваними ознаками: режими зносу інструмента та випадкових відмов у наборі AI4I 2020 цього не мають і типізації не піддаються. Обидва набори залишаються імітаційними стендами, тож перенесення на реальну потокову телеметрію потребує окремої перевірки.

Висновки

Подію доцільно розглядати як семантичний контракт: хмара відповідає не лише за виявлення проблеми, а й за її найменування, а ERP отримує готову підставу для дії. Обидва

датасети показали, що контракт наповнюється (p = 93–98 %) і майже нічого не коштує, проте діє лише за влучного детектора — умова окупності пов’язує семантику події з якістю попереднього рівня. Надалі власна, консервативніша робоча точка типізованої політики, гібридне рішення ERP за низької впевненості та довірений реєстр подій.

Список використаних джерел

1. Kondratiuk D., Kazymyr V. A Hybrid Model of Interaction Between IoT and ERP Systems in the Context of Industry 4.0. ACIT 2026 (подано до друку).
2. Кондратюк Д. С., Казимир В. В. Багаторівневий edge–cloud конвеєр у взаємодії IoT та ERP систем. XXVI Міжнародна науково-практична конференція, 2026.
3. Tang Ji, Xu X. Exploring the Integration of Cloud Manufacturing and Cyber-Physical Systems in the Era of Industry 4.0 – An OPC UA Approach. Robotics and Computer-Integrated Manufacturing, 2025, 93, 102927.
4. Matzka S. AI4I 2020 Predictive Maintenance Dataset. UCI Machine Learning Repository, 2020. DOI: 10.24432/C5HS5C.
5. Microsoft Azure Predictive Maintenance. URL: https://www.kaggle.com/datasets/arnabbiswas1/microsoft-azure-predictive-maintenance (дата звернення: 19.09.2026).
6. scikit-learn: Machine Learning in Python. URL: https://scikit-learn.org (дата звернення: 19.09.2026).

Красько Т. О.
Сумський державний університет, кафедра комп'ютерних наук, Суми, Україна
e-mail: timakrasko2004@gmail.com

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ БРОНЮВАННЯ СТОЛИКІВ У РЕСТОРАНАХ ЗАСОБАМИ МОБІЛЬНОГО ЗАСТОСУНКУ ANDROID

Анотація. У роботі досліджено процес автоматизації бронювання столиків у закладах громадського харчування за допомогою сучасних мобільних технологій. Проблематика традиційних методів бронювання полягає в їхній низькій ефективності, ймовірності людських помилок та відсутності гнучких інструментів для користувачів на українському ринку. Метою проєкту є розробка кросплатформового мобільного застосунку під операційну систему Android, що забезпечує пошук закладів, перегляд доступних столів у реальному часі, керування передзамовленнями та систему відгуків. Як основні інструменти розробки використано мову програмування Kotlin, декларативний фреймворк Jetpack Compose, архітектурний патерн Clean Architecture у поєднанні з MVVM, а також систему впровадження залежностей Koin. Практичним результатом є готовий до розгортання програмний продукт, який оптимізує внутрішні процеси ресторанів та покращує якість взаємодії з клієнтами.

Ключові слова: мобільний застосунок; Android; Kotlin; Jetpack Compose; бронювання столиків; Clean Architecture; ресторанний бізнес.

Вступ

Сучасний ресторанний бізнес стрімко цифровізується, впроваджуючи інформаційні технології для підвищення якості обслуговування. Традиційні способи бронювання (телефонні дзвінки, паперові журнали) стають неефективними через незручність для клієнтів та ризики втрати даних. Метою даної роботи є розробка інформаційної технології та мобільного застосунку для автоматизації процесу бронювання столиків і передзамовлення страв у мережі ресторанів засобами платформи Android.

Матеріали та методи

Для реалізації проєкту використано виключно сучасне вільне та відкрите програмне забезпечення (FOSS) і рекомендовані Google технологічні стеки:

– Мова програмування: Kotlin (забезпечує лаконічність, безпеку типів null-safety та підтримку асинхронних корутин).

– Побудова інтерфейсу (UI): Декларативний фреймворк Jetpack Compose.

– Архітектура: Комбінація Clean Architecture та патерну MVVM (Model-View-ViewModel) для чіткого розмежування бізнес-логіки та UI.

– Управління залежностями: Фреймворк Koin.

– Навігація: Jetpack Navigation Component v3.

– Середовище розробки та контроль версій: Android Studio (JDK 17), емулятор AVD (Pixel 9, API 36) та система контролю версій Git / GitHub.

Результати

У ході виконання практики розроблено повнофункціональний мобільний застосунок, що включає такі ключові модулі:

– Головний екран та каталог локацій (HomeScreen / LocationDetails): Дозволяє користувачам без обов'язкової реєстрації переглядати інформацію про заклади, меню, рейтинги та популярні страви.

– Модуль аутентифікації (LoginScreen / RegistrationScreen): Забезпечує безпечний вхід та реєстрацію з перевіркою складності паролів, а також інтеграцію соціальних сервісів.

– Система бронювання (BookTableScreen): Фільтрація в реальному часі за локацією, датою, часом та кількістю гостей із подальшим вибором доступних столів.

– Управління резерваціями (ReservationsScreen): Перегляд поточних і завершених бронювань, можливість їх редагування, скасування та залишення відгуків.

– Профіль користувача (ProfileScreen): Керування персональними даними, зміна пароля та аватарки.

Обговорення

Порівняння розробленого рішення з відомими світовими аналогами (OpenTable, TheFork, Resy) показало, що закордонні платформи не мають партнерських закладів в Україні або недоступні для завантаження в регіоні. Запропонований застосунок адаптований під локальні потреби ринку. Враховуючи відсутність конкуренції, застосунок має великі перспективи.

Висновки

Створено мобільний додаток для автоматизації бронювання столиків у ресторанах, який повністю виконує поставлені завдання. Продемонстровано ефективність застосування сучасних інструментів Android-розробки. Отримані результати закладають міцний фундамент для завершення магістерського дослідження та впровадження програмного продукту в комерційну експлуатацію.

Список використаних джерел

1. Gilski P., Stefanski J. Android OS: A Review // TEM Journal. 2015. T. 4, № 1.
2. Ardito L. Effectiveness of Kotlin vs. Java in android app development tasks // Inf. Softw. Technol. 2020. T. 127.
3. Fajri A. R., Rani S. Penerapan Design Pattern MVVM dan Clean Architecture pada Pengembangan Aplikasi Android // Automata. 2022. T. 3, № 2.
4. Arriola C. Meet Android Studio Android Developers // Android Studio. 2024.

Литвин С. О.
Національний університет «Чернігівська політехніка», кафедра інформаційних та комп'ютерних систем, Чернігів, Україна
Контактний автор: Сергій Литвин, e-mail: seregalytvin@gmail.com, ORCID: 0009-0004-0902-2828

ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ ВИЯВЛЕННЯ АНОМАЛІЙ У ЛОГАХ ПРОГРАМНИХ СИСТЕМ

Анотація. Автоматизоване виявлення аномалій у логах програмних систем є важливим завданням моніторингу, діагностики та забезпечення надійності програмного забезпечення. У роботі експериментально досліджено вплив способу представлення логових даних та алгоритму машинного навчання на якість виявлення аномалій. Експеримент виконано для трьох наборів логів – HDFS, BGL та OpenStack – із використанням трьох способів представлення: Count Vector, TF-IDF та Binary Event Presence. Порівняно чотири методи без учителя: Isolation Forest, One-Class SVM, Autoencoder та Local Outlier Factor. Загалом виконано 36 комбінацій «набір даних – представлення – алгоритм». Якість оцінювалася за Precision, Recall, F1 та FPR. Результати показали, що ефективність методів істотно залежить від набору даних та способу представлення логів, тому універсального поєднання алгоритму і представлення для всіх досліджених систем не виявлено.

Ключові слова: аналіз логів; виявлення аномалій; машинне навчання; HDFS; BGL; OpenStack; Count Vector; TF-IDF; Binary Event Presence; Isolation Forest; One-Class SVM; Autoencoder; Local Outlier Factor.

Вступ

Журнали подій програмних систем містять інформацію про роботу програмного забезпечення, виконання операцій, помилки та інші події, що виникають під час функціонування системи. Аналіз великих обсягів подій вручну є складним, тому автоматизовані методи їх обробки мають практичне значення для моніторингу та виявлення відхилень у роботі програмних систем.

Одним із підходів до автоматизації такого аналізу є застосування методів машинного навчання для виявлення аномальних логових послідовностей. При цьому результат може залежати не лише від самого алгоритму, але й від способу перетворення послідовності подій у числове представлення. Одна й та сама логова інформація, представлена як кількість поява подій, TF-IDF або бінарна ознака наявності події, може формувати різну структуру простору ознак.

Метою дослідження є експериментальне визначення впливу способу представлення логових даних та алгоритму машинного навчання на ефективність виявлення аномалій у програмних системах.

Для досягнення мети проведено порівняльний експеримент на трьох наборах логів із використанням чотирьох алгоритмів машинного навчання та трьох способів представлення даних.

Матеріали та методи

У дослідженні використано три набори логів програмних систем: HDFS, BGL та OpenStack. Вони відрізняються масштабом, структурою логових подій та способом формування еталонної розмітки аномалій.

Для HDFS використано підготовлений набір логових послідовностей, у якому об'єктом аналізу є блок файлової системи (BlockId). Кожному блоку відповідає одна послідовність подій та еталонна мітка Normal або Anomaly. Загалом набір містить 575 061 об'єкт, з яких 558 223 нормальних та 16 838 аномальних. Для HDFS додатковий парсинг логів у межах цього дослідження не виконувався, оскільки використовувалася вже структурована послідовність Event ID.

Для BGL (Blue Gene/L) використано логовий набір, структуризація якого виконувалася за допомогою парсера на основі алгоритму Drain, реалізованого у проєкті LogHub. У процесі обробки текстові записи були перетворені на структуровані події з відповідними Event ID. Для подальшого аналізу логові записи групувалися у 10-секундні часові вікна. Вікно

вважалося аномальним, якщо хоча б один запис у ньому мав аномальну мітку. Після формування та підготовки вибірки для експериментів отримано 17 005 об'єктів, серед яких 11 423 нормальні та 5 582 аномальні.

Для OpenStack використано набір даних, підготовлений на основі датасету OpenStack від ParisaKalaki. Для цілей цього дослідження використано вдосконалену та адаптовану версію набору, у якій виконано додаткову фільтрацію та встановлення відповідності між розміченими екземплярами і логовими послідовностями. Для ідентифікації екземплярів використовувалися InstanceId, а за його відсутності – однозначний request_id. Неоднозначні та такі, що не мали відповідності, записи не включалися до подальшого аналізу. У фінальній експериментальній вибірці отримано 371 об'єкт, з яких 191 нормальний та 180 аномальних.

Таким чином, загальна вибірка охоплює 592 066 об'єктів із трьох різних програмних систем, серед яких 574 837 нормальних та 22 600 аномальних об'єктів.

Для кожного набору даних досліджено три способи представлення послідовностей подій:

– Count Vector (Event Count Vector) – кожна ознака відповідає кількості появ відповідного Event ID у послідовності;

– TF-IDF – вагове представлення Event ID з урахуванням частоти події в конкретній послідовності та її поширеності у навчальній вибірці;

– Binary Event Presence – бінарне представлення, де значення 0 означає відсутність події, а 1 – її наявність незалежно від кількості появ.

Для кожного набору даних використовувався фіксований простір Event ID. Для TF-IDF параметри IDF визначалися тільки за нормальною навчальною вибіркою. Це дозволило уникнути використання еталонних міток тестової вибірки під час побудови моделі.

Експериментальна постановка відповідає підходу unsupervised/one-class anomaly detection. Навчання моделей здійснювалося лише на нормальних послідовностях. Тестова вибірка містила як нормальні, так і аномальні послідовності. Еталонні мітки не використовувалися для навчання baseline-моделей і застосовувалися лише після отримання прогнозів для оцінювання якості.

Для визначення порога аномальності нормальна навчальна вибірка поділялася на 80 % для навчання та 20 % для валідації. Модель навчалася на першій частині, після чого для нормальної валідаційної вибірки обчислювався anomaly score. Порогом визначалося 95-е процентильне значення валідаційних оцінок. Після фіксації порога модель застосовувалася до тестової вибірки.

Порівняно чотири алгоритми:

– Isolation Forest (IF);

– One-Class SVM (OCSVM);

– Autoencoder (AE) на основі MLPRegressor;

– Local Outlier Factor (LOF) з режимом novelty=True.

Для оцінювання використано Precision, Recall, F1 та False Positive Rate (FPR). Основною метрикою порівняння визначено F1, оскільки вона враховує одночасно повноту виявлення аномалій та точність позитивних прогнозів. Додатково вимірювалися час навчання та час інференсу.

Експериментальна реалізація виконана мовою Python із використанням вільного програмного забезпечення та бібліотек екосистеми Python для машинного навчання й аналізу даних.

Результати

Загальна експериментальна матриця містила 3 набори даних, 3 способи представлення, 4 алгоритми, тобто 36 експериментів.

Основні результати за метрикою F1 наведено в табл. 1.

Таблиця 1 – Значення F1 для досліджуваних комбінацій

Набір даних	Представлення	IF	OCSVM	AE	LOF
HDFS	Count Vector	0,1742	0,8336	0,8637	0,9955
HDFS	TF-IDF	0,8245	0,8866	0,8562	0,9956

HDFS	Binary Event Presence	0,3273	0,9423	0,9159	0,9605
BGL	Count Vector	0,0908	0,1665	0,8101	0,8565
BGL	TF-IDF	0,0499	0,9298	0,9076	0,8992
BGL	Binary Event Presence	0,0918	0,3960	0,9496	0,9167
OpenStack	Count Vector	0,5078	0,9704	0,6165	0,7365
OpenStack	TF-IDF	0,9704	0,9756	0,9756	0,7616
OpenStack	Binary Event Presence	0,4701	0,9783	0,9753	0,9755

Результати демонструють істотну залежність якості від поєднання способу представлення та алгоритму. На HDFS Local Outlier Factor показав F1 = 0,9955 для Count Vector та F1 = 0,9956 для TF-IDF. Для цих конфігурацій Precision становив 0,9915, Recall – 0,9996 та 0,9997 відповідно, а FPR – 0.0013.

Для HDFS також спостерігається значний вплив представлення на Isolation Forest. При переході від Count Vector до TF-IDF F1 збільшився з 0,1742 до 0.8245. Для One-Class SVM найвище значення F1 отримано при Binary Event Presence – 0,9423, а для Autoencoder – також при Binary Event Presence, де F1 становив 0.9159.

На BGL результати алгоритмів відрізнялися значно сильніше. Isolation Forest показав низькі значення F1 для всіх трьох способів представлення: 0,0908 для Count Vector, 0,0499 для TF-IDF та 0,0918 для Binary Event Presence. Натомість One-Class SVM з TF-IDF досяг F1 = 0.9298. Найвище значення F1 на BGL отримано для Autoencoder з Binary Event Presence – 0,9496, при Precision = 0,9066 та Recall = 0.9968.

Для OpenStack усі три представлення забезпечили високі результати для One-Class SVM: F1 становив 0,9704, 0,9756 та 0,9783 для Count Vector, TF-IDF та Binary Event Presence відповідно. Для Isolation Forest найбільшою була різниця між представленнями: F1 збільшився від 0,5078 для Count Vector до 0,9704 для TF-IDF. Для Autoencoder F1 становив 0,6165 для Count Vector та близько 0,976 для TF-IDF і Binary Event Presence.

Окрім якості класифікації, алгоритми відрізнялися за обчислювальними витратами. Найбільш помітною ця відмінність була для LOF на HDFS. Час навчання LOF становив приблизно 2586–3537 с, а час інференсу – приблизно 938–1238 с залежно від представлення. Для порівняння, час навчання Autoencoder на HDFS становив близько 8–17 с, а інференсу – менше 0,04 с. Таким чином, висока якість LOF супроводжувалася значно більшими обчислювальними витратами.

Обговорення

Отримані результати підтверджують, що спосіб представлення логів є важливим фактором для anomaly detection. Водночас його вплив не є однаковим для різних алгоритмів та наборів даних.

Показовим прикладом є BGL. Для Isolation Forest перехід до TF-IDF не покращив якість: F1 знизився з 0,0908 для Count Vector до 0.0499. Водночас для One-Class SVM на тому самому представленні TF-IDF забезпечив F1 = 0.9298. Отже, одна й та сама структура ознак може по-різному використовуватися різними алгоритмами машинного навчання.

Подібна залежність спостерігається на OpenStack. Для Isolation Forest TF-IDF забезпечив F1 = 0,9704 порівняно з 0,5078 для Count Vector, тоді як для LOF різниця між Count Vector і TF-IDF була значно меншою: 0,7365 проти 0.7616. Це демонструє, що вплив представлення не можна розглядати незалежно від алгоритму.

Окремо слід відзначити Binary Event Presence. На BGL цей спосіб представлення забезпечив для Autoencoder F1 = 0,9496, що є найвищим результатом Autoencoder серед трьох наборів даних та досліджених представлень для BGL. На HDFS Binary Event Presence також підвищив F1 Autoencoder до 0,9159 та F1 One-Class SVM до 0.9423.

При цьому результати не дозволяють зробити висновок про існування одного універсально оптимального алгоритму або представлення. Наприклад, найвищі значення F1 для HDFS отримано з LOF, для BGL – з Autoencoder, а для OpenStack – з One-Class SVM.

Таким чином, практичний вибір методу залежить від характеристик конкретного набору логів.

Важливим обмеженням є обчислювальна складність LOF для великих вибірок. Хоча на HDFS він забезпечив F1 близько 0,996, час виконання був значно більшим порівняно з іншими досліджуваними методами. Крім того, Autoencoder для BGL не досяг повної збіжності за встановленої кількості ітерацій, що слід враховувати під час інтерпретації відповідних результатів.

Іншим обмеженням є те, що Count Vector, TF-IDF та Binary Event Presence є представленнями типу bag-of-events. Вони характеризують наявність або частоту Event ID, але безпосередньо не моделюють порядок подій у логовій послідовності. Дослідження послідовнісних представлень може бути окремим напрямом подальшої роботи.

Висновки

У роботі проведено експериментальне порівняння чотирьох методів без учителя для виявлення аномалій у логах програмних систем із використанням трьох способів представлення логових послідовностей. Для наборів HDFS, BGL та OpenStack виконано 36 експериментів.

Результати показали, що якість виявлення аномалій залежить одночасно від набору даних, способу представлення та алгоритму машинного навчання. Для різних наборів даних найвищі значення F1 були отримані для різних комбінацій методів: LOF на HDFS, Autoencoder з Binary Event Presence на BGL та One-Class SVM з Binary Event Presence на OpenStack.

Отримані результати підтверджують доцільність порівняння не лише алгоритмів машинного навчання, а й способів представлення логових даних. Водночас висока якість не завжди відповідає мінімальним обчислювальним витратам, що особливо помітно для LOF на HDFS.

Подальше дослідження передбачає детальніший аналіз структури аномальних послідовностей, оцінювання стабільності результатів та дослідження представлень, здатних враховувати порядок подій у логах.

Список використаних джерел

1. Liu F. T., Ting K. M., Zhou Z.-H. Isolation Forest. 2008 Eighth IEEE International Conference on Data Mining. IEEE, 2008. P. 413–422.
2. Schölkopf B., Platt J. C., Shawe-Taylor J., Smola A. J., Williamson R. C. Estimating the Support of a High-Dimensional Distribution. Neural Computation. 2001. Vol. 13, No. 7. P. 1443–1471.
3. Breunig M. M., Kriegel H.-P., Ng R. T., Sander J. LOF: Identifying Density-Based Local Outliers. Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data. 2000. P. 93–104.
4. scikit-learn developers. scikit-learn: Machine Learning in Python. URL: https://scikit-learn.org/.
5. Apache Software Foundation. Apache Hadoop. URL: https://hadoop.apache.org/.
6. Liang Y., Zhang Y., Xiong H., Sahoo R. Failure Prediction in IBM Blue Gene/L Event Logs. Proceedings of the 2007 IEEE International Conference on Data Mining. IEEE, 2007.
7. Du M., Li F., Zheng G., Srikumar V. DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning. Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2017. P. 1285–1298.
8. LogHub. A Large Collection of System Log Datasets for AI-driven Log Analytics. GitHub. URL: https://github.com/logpai/loghub.
9. LogPAI. LogParser: A Machine Learning Toolkit for Log Parsing. GitHub. URL: https://github.com/logpai/logparser.
10. LogPAI. Drain - an online log parsing approach with fixed depth tree. GitHub. URL: https://github.com/logpai/logparser/tree/main/logparser/Drain.
11. Kalaki P. OpenStack Loglizer - Anomaly Detection in OpenStack Logs. GitHub. URL: https://github.com/ParisaKalaki/OpenStack.
12. Kalaki P. OpenStack Logs. GitHub. URL: https://github.com/ParisaKalaki/OpenStack-logs.

Максимов О. В., Лисенко Д. Е.
Національний університет «Чернігівська політехніка», кафедра інформаційних та комп'ютерних систем, Чернігів, Україна
Контактний автор: Олексій Максимов, e-mail: maksimov98gmy@gmail.com, ORCID: 0009-0008-4257-6549

ОЦІНЮВАННЯ ВІДКРИТИХ ЛОКАЛЬНИХ МОВНИХ МОДЕЛЕЙ ДЛЯ ГЕНЕРАЦІЇ ЮНІТ-ТЕСТІВ У PYTHON-ПРОЄКТАХ

Анотація. Генерація юніт-тестів за допомогою великих мовних моделей (LLM) може суттєво зменшити ручні витрати на тестування, однак хмарні сервіси часто недоступні через вимоги конфіденційності вихідного коду. У роботі оцінено, чи здатні відкриті локальні моделі класу 7B – Mistral 7B, Code Llama 7B та DeepSeek Coder 6.7B – пройти повний цикл «генерація – компіляція – виконання» для Python-проєкту itsdangerous в офлайн-середовищі, де оркестратор не може змістовно покращувати згенерований код. Запропоновано відтворюваний протокол на основі вільних інструментів (pytest, coverage.py, py_compile) із забороною змін production-коду та однією ітерацією виправлення. Жодна з моделей не забезпечила успішного запуску згенерованих тестів; найкращу частку компільованих файлів після мінімальної санітизації показала Mistral 7B (5/5). Основні причини відмов – змішування коду з природною мовою, Markdown-обгортки та застарілі імпорти.

Ключові слова: відкрите ПЗ; локальні мовні моделі; генерація юніт-тестів; Python; pytest; автоматизація тестування.

Вступ

Юніт-тестування є одним із базових засобів забезпечення якості програмного забезпечення, проте підготовка якісних тестів потребує значних витрат часу та глибокого розуміння логіки модуля. Класичні підходи до автоматизації (випадкове та пошукове тестування, символьне виконання) останніми роками доповнюються генерацією тестів за допомогою LLM безпосередньо з вихідного коду [1–4]. Більшість таких досліджень, однак, спирається на хмарні комерційні моделі, використання яких неприйнятне для закритих кодових баз, освітніх і наукових середовищ з обмеженим бюджетом або для організацій, що вимагають повністю відтворюваного локального процесу.

Поява відкритих (open-weight) моделей для роботи з кодом – Mistral 7B [5], Code Llama [6], DeepSeek-Coder [7] – та засобів їх запуску на звичайній робочій станції створює передумови для приватної генерації тестів. Відкритим залишається питання, чи здатні такі моделі помірного розміру самостійно створювати тести, які не лише «схожі на код», а й компілюються та виконуються. Мета роботи – експериментально оцінити придатність відкритих локальних LLM класу 7B для генерації юніт-тестів мовою Python і виявити типові причини відмов.

Матеріали та методи

Об'єктом дослідження обрано відкриту бібліотеку itsdangerous (ліцензія BSD-3-Clause) [8] із п'ятьма цільовими модулями: encoding.py, serializer.py, signer.py, timed.py та url_safe.py. Протокол забороняв зміну production-коду (src/itsdangerous/) і базових тестів; згенеровані файли зберігалися в окремому каталозі для кожної моделі, сирі відповіді моделей архівувалися, а всі етапи логувалися.

Моделі Mistral 7B, Code Llama 7B та DeepSeek Coder 6.7B запускалися локально через API-сервер LM Studio з OpenAI-сумісною кінцевою точкою, що забезпечувало єдиний спосіб виклику. Параметри генерації були фіксованими: temperature = 0,2, top_p = 0,95, max_tokens = 1600, seed = 1. Промпт вимагав виводити лише коректний Python-код у стилі pytest, без мережевого доступу й запису у файлову систему, з покриттям нормальних, граничних і помилкових випадків.

Конвеєр оцінювання включав: 1) базовий прогін наявних тестів за допомогою pytest [9] і coverage.py [10]; 2) одноразову генерацію тестів для кожного модуля; 3) перевірку синтаксису через python -m py_compile; 4) запуск pytest; 5) за потреби – одну repair-ітерацію тією самою моделлю. Єдиною дозволеною постобробкою було вилучення зовнішніх

Markdown-огорож (```), якщо саме вони перешкоджали компіляції. Метрики: частка компільованих файлів, код виходу pytest, журнали помилок і базове покриття.

Результати

Базовий прогін підтвердив коректність стенда: 297 штатних тестів пройшли успішно за 0,55 с, звіт coverage зафіксував 322 інструкції без пропусків (100 % покриття). Отже, подальші відмови можна пов'язувати саме зі згенерованими артефактами, а не із середовищем. Узагальнені результати для локальних моделей наведено в табл. 1.

Таблиця 1 – Результати оцінювання локальних моделей

Модель	**Компіляція (найкраща)**	**Код виходу pytest**	**Основні причини відмов**
Mistral 7B	5/5*	2	Супровідний текст і Markdown-обгортки; імпорти через src.*; некоректні escape-послідовності; repair-запит повертав HTTP 400
Code Llama 7B	1/6	2	Незакриті дужки, незавершені рядкові літерали, code fences усередині файлів, нестабільне завершення генерації
DeepSeek Coder 6.7B	3/4*	2	Не-ASCII символ у bytes-літералі; імпорт застарілого TimedJSONWebSignatureSerializer; проза/Markdown у первинному виході

* після вилучення лише зовнішніх Markdown-огорож

Mistral 7B згенерувала файли для всіх п'яти модулів, але у первинному вигляді вони не компілювалися через вставки на кшталт «Here's a sample test suite…». Після мінімальної санітизації всі файли компілювалися, проте pytest завершувався помилками імпорту та лексичними дефектами, а repair-етап не відпрацював через відповідь HTTP 400. Code Llama 7B продемонструвала найнижчу стабільність: компілювався лише 1 файл із 6. DeepSeek Coder 6.7B показала проміжний результат (3/4 доступних файлів), однак виконання зупинялося на не-ASCII символі в bytes-літералі та на імпорті API, відсутнього в актуальній версії бібліотеки.

Обговорення

Аналіз журналів дозволив виокремити чотири групи відмов: 1) природномовний текст і Markdown-огорожі у файлах, що мали містити лише код; 2) помилкові або застарілі імпорти (шлях src.*, неіснуючі символи API); 3) лексичні дефекти Python (не-ASCII bytes-літерали, некоректні escape-послідовності); 4) нестабільність repair-ланки. Таким чином, ключовим обмеженням локальних моделей помірного розміру є не лише семантична повнота тестів, а насамперед дотримання жорсткого формату виходу та знання актуального API. Мінімальна постобробка підвищує частку компільованих файлів, але не усуває структурних помилок на рівні залежностей.

На відміну від досліджень із великими хмарними моделями [1–3], де значна частка тестів виконується успішно, у суворо офлайн-сценарії з моделями 7B повний цикл не було пройдено жодного разу. Обмеженнями роботи є один об'єкт дослідження, фіксований seed, одна repair-ітерація та використання LM Studio, яке, на відміну від решти стека, не є вільним ПЗ; для повністю відкритого конвеєра його доцільно замінити на llama.cpp або Ollama з тією самою OpenAI-сумісною кінцевою точкою. Практично локальні 7B-моделі наразі доцільно застосовувати як генератор чернеток і тестових ідей, а не як автономний генератор готових тестів.

Висновки

Запропоновано та апробовано відтворюваний протокол оцінювання відкритих локальних LLM для генерації юніт-тестів, побудований на вільних інструментах і без зміни production-коду. Жодна з трьох моделей класу 7B не забезпечила повного циклу «генерація – компіляція – запуск» без ручного втручання; найкращі показники компіляції після мінімальної санітизації становили 5/5 для Mistral 7B, 3/4 для DeepSeek Coder 6.7B та 1/6 для Code Llama

7В. Подальші дослідження передбачають обмежене граматикою декодування (grammar-constrained decoding), AST-орієнтовану санітизацію, адаптери імпортів для актуальних версій бібліотек, розширення вибірки моделей і проєктів, а також оцінювання приросту покриття та мутаційного тестування.

Список використаних джерел

1. Schäfer M., Nadi S., Eghbali A., Tip F. An empirical evaluation of using large language models for automated unit test generation. IEEE Transactions on Software Engineering. 2024. Vol. 50, № 1. P. 85–105. DOI: 10.1109/TSE.2023.3334955.
2. Siddiq M. L., Santos J. C. S., Tanvir R. H. et al. Using large language models to generate JUnit tests: An empirical study. Proc. of EASE 2024. ACM, 2024. P. 313–322. DOI: 10.1145/3661167.3661216.
3. Yang L., Yang C., Gao S. et al. On the evaluation of large language models in unit test generation. Proc. of ASE 2024. ACM, 2024. P. 1607–1619. DOI: 10.1145/3691620.3695529.
4. Wang J., Huang Y., Chen C. et al. Software testing with large language models: Survey, landscape, and vision. IEEE Transactions on Software Engineering. 2024. Vol. 50, № 4. P. 911–936. DOI: 10.1109/TSE.2024.3368208.
5. Jiang A. Q., Sablayrolles A., Mensch A. et al. Mistral 7B. arXiv:2310.06825. 2023. URL: https://arxiv.org/abs/2310.06825
6. Rozière B., Gehring J., Gloeckle F. et al. Code Llama: Open foundation models for code. arXiv:2308.12950. 2023. URL: https://arxiv.org/abs/2308.12950
7. Guo D., Zhu Q., Yang D. et al. DeepSeek-Coder: When the large language model meets programming. arXiv:2401.14196. 2024. URL: https://arxiv.org/abs/2401.14196
8. itsdangerous. Pallets Projects. URL: https://github.com/pallets/itsdangerous
9. pytest: helps you write better programs. URL: https://github.com/pytest-dev/pytest
10. Coverage.py: code coverage measurement for Python. URL: https://github.com/nedbat/coveragepy

Максьом К. В.
Національний університет «Чернігівська політехніка», кафедра соціальної роботи, м. Чернігів, Україна
Контактний автор: Катерина Максьом, e-mail: maxsemka@stu.cn.ua, ORCID: 0000-0002-4338-0344

ЦИФРОВА МЕДІАГРАМОТНІСТЬ У ДОБУ ГЕНЕРАТИВНОГО ШІ: ПОТЕНЦІАЛ ВІДКРИТИХ ІНСТРУМЕНТІВ / DIGITAL MEDIA LITERACY IN THE AGE OF GENERATIVE AI: THE POTENTIAL OF OPEN-SOURCE TOOLS

Анотація. Генеративний штучний інтелект (ШІ) змінює інформаційне середовище: з'являються синтетичні тексти й зображення, правдоподібні, але вигадані факти, нові форми дезінформації. Тому цифрова медіаграмотність має охоплювати й уміння критично працювати з контентом, створеним ШІ. У тезах розглянуто потенціал відкритих ШІ-інструментів для розвитку таких умінь у межах вибіркового навчального курсу «AI & LLM Literacy: етика, критичне мислення та відповідальне використання генеративного ШІ». Запропоновано набір інструментів з відкритим кодом (Ollama, GPT4All, Hugging Face, моделі з відкритими вагами) та типи навчальних завдань, згрупованих навколо ключових навичок, що відносять до медіаграмотності. Показано, що відкритість інструментів сама по собі відповідає принципам медіаграмотності, а також окреслено інфраструктурні умови їх використання в закладах вищої освіти.

Ключові слова: цифрова медіаграмотність; генеративний штучний інтелект; AI-грамотність; відкрите програмне забезпечення; критичне мислення; вища освіта.

Вступ

Генеративний ШІ за короткий час став частиною повсякденної роботи людини з інформацією. Чат-боти відповідають на запитання, пишуть тексти, створюють зображення. Для користувача такий контент часто виглядає достовірним, хоча може містити вигадані факти, спотворення чи упередження. Водночас ті самі технології використовують і для масового виробництва дезінформації. Для України, яка перебуває в умовах війни, в т.ч. інформаційної, це питання стає особливо гостро.

Медійна та інформаційна грамотність традиційно передбачає вміння шукати, оцінювати й відповідально поширювати інформацію. ЮНЕСКО вже включає тему ШІ до навчальних програм із медійної та інформаційної грамотності [5] і наголошує на потребі формувати компетентності щодо ШІ в освіті [3; 4]. У науковій літературі AI-грамотність розглядають як здатність розуміти, критично оцінювати та відповідально використовувати технології ШІ [1; 2]. Отже, у добу генеративного ШІ AI-грамотність стає необхідною складовою цифрової медіаграмотності.

Більшість популярних сервісів генеративного ШІ є закритими: невідомо, на яких даних навчено моделі, а запити користувачів обробляються на серверах відповідних компаній. Такі сервіси складно досліджувати з освітньою метою. Відкриті ШІ-інструменти, навпаки, дозволяють «зазирнути всередину» технології, зрозуміти як це працює.

Метою даної роботи є обґрунтування потенціалу відкритих ШІ-інструментів для розвитку цифрової медіаграмотності здобувачів вищої освіти.

Матеріали та методи

Основою дослідження став вибірковий курс «AI & LLM Literacy: етика, критичне мислення та відповідальне використання генеративного ШІ», що розроблено на кафедрі соціальної роботи Національного університету «Чернігівська політехніка» для здобувачів бакалаврського рівня різних спеціальностей. Використано методи аналізу наукових джерел і міжнародних рекомендацій, порівняльного аналізу програмних засобів та педагогічного проєктування.

Інструменти відбирали за такими критеріями: відкрита ліцензія; можливість локального запуску без передавання даних стороннім сервісам; прийнятні вимоги до апаратного забезпечення; підтримка української мови; простота використання студентами без IT-підготовки. Розглядалися версії програм, актуальні станом на вересень 2026 р.; посилання на репозиторії наведено у списку джерел [6–8].

Результати

За результатами аналізу для курсу запропоновано чотири групи відкритих інструментів (табл. 1). Їх розглянуто не з технічного погляду, а з огляду на те, які навички медіаграмотності вони допомагають розвивати.

Таблиця 1 – Відкриті ШІ-інструменти, запропоновані для курсу, та їхній внесок у розвиток цифрової медіаграмотності

Інструмент	**Ліцензія**	**Внесок у розвиток цифрової медіаграмотності**
Ollama	MIT	Локальний запуск мовних моделей: студенти бачать, що «розумна» відповідь – це результат роботи конкретної програми на конкретному комп'ютері
GPT4All	MIT	Простий графічний інтерфейс для студентів без IT-підготовки; відповіді на основі власних документів (LocalDocs) дозволяють порівнювати згенерований текст із джерелом
Hugging Face (платформа та бібліотека Transformers)	Apache 2.0 (бібліотека)	Картки моделей і наборів даних: аналіз походження, авторства, обмежень та умов використання – тобто перевірка «джерела» самої технології
Моделі з відкритими вагами (напр., Mistral 7B)	Apache 2.0	Порівняння відповідей різних моделей на однаковий запит, виявлення помилок, упереджень і вигаданих фактів

На основі цих інструментів спроєктовано навчальні завдання, які відповідають чотирьом ключовим навичкам цифрової медіаграмотності.

Перевірка фактів. Здобувачі освіти отримують відповідь моделі, знаходять у ній твердження, що потребують перевірки, і звіряють їх із першоджерелами. Так вони вчаться помічати «галюцинації» – правдоподібні, але вигадані факти.

Аналіз джерела та походження. Здобувачі освіти вивчають картки моделей: хто створив модель, на яких даних її навчено, які обмеження визначили розробники. Це аналог запитання «хто автор і чи можна йому довіряти», яке медіаграмотна людина ставить до будь-якого повідомлення.

Розпізнавання синтетичного контенту. Студенти порівнюють тексти, які написані людиною та моделлю, досліджують, як формулювання запиту змінює відповідь, і знайомляться з підходами до маркування згенерованих матеріалів, зокрема з відкритим стандартом C2PA [9].

Етика створення та поширення. Студенти обговорюють упередження у відповідях моделей, питання авторства і відповідальності за використання та поширення згенерованого контенту.

Обговорення

Відкритість інструментів сама по собі є принципом медіаграмотності. Якщо ми вчимо студентів перевіряти джерело будь-якої інформації, логічно, щоб і технологія, з якою вони працюють, була прозорою: відомо, хто її створив, на яких даних і з якими обмеженнями. Крім того, під час локального запуску запити здобувачів не передаються стороннім компаніям/особам, що саме по собі є прикладом відповідального ставлення до даних.

Невеликі локальні моделі помиляються частіше, ніж великі комерційні системи. З погляду медіаграмотності це перевага: помилки легше помітити, тож їх зручно використовувати саме для тренування критичного мислення.

Водночас, відкрите програмне забезпечення має безкоштовну ліцензію, але його не можна назвати цілковито безкоштовним. Локальні моделі потребують комп'ютерів із достатнім обсягом пам'яті, компетентного технічного супроводу, а в умовах відключень електроенергії – ще й резервного живлення. Тобто відкриті інструменти не усувають

витрати, а замінюють постійні платежі за підписки вкладеннями в інфраструктуру, яка залишається в університеті. Для закладу вищої освіти це важливо з погляду сталості: налаштовані інструменти та обладнання продовжують працювати після завершення фінансування. Зменшити витрати можна, розмістивши модель на одному сервері центру замість потужних комп'ютерів у кожній аудиторії.

Серед інших обмежень – нижча якість відповідей українською мовою у невеликих моделях та потреба в підготовці викладачів. Слід також розрізняти програмне забезпечення з відкритим кодом і моделі з відкритими вагами, тобто моделі, параметри яких опубліковано для вільного завантаження та використання. Дані, на яких навчено такі моделі, не завжди відомі, а їхні ліцензії можуть обмежувати використання. Комерційні сервіси залишаються частиною реального досвіду студентів, тому доцільним є поєднаний підхід: відкриті інструменти – для розуміння технології та експериментів, комерційні – для порівняння і критичного аналізу.

Висновки

У добу генеративного ШІ цифрова медіаграмотність має охоплювати вміння критично працювати з контентом, створеним ШІ. Відкриті ШІ-інструменти мають для цього значний потенціал: вони роблять технологію прозорою, дозволяють досліджувати її помилки та походження і відповідають принципам відповідального ставлення до даних. За умови продуманого інфраструктурного забезпечення вони можуть стати сталою складовою навчання цифрової медіаграмотності у закладах вищої освіти. Подальший напрям роботи – пілотне тестування курсу та запропонованих завдань і оцінювання їхнього впливу на рівень цифрової медіаграмотності здобувачів.

Подяки

Тези підготовлено в межах проєкту «Навчання цифрової медіаграмотності в умовах суспільних змін в Україні» (TeDiMeLi, програма Erasmus+ KA2; номер проєкту: 101237182). Проєкт TeDiMeLi співфінансується Європейським Союзом, проте висловлені погляди та думки належать лише авторам цього проєкту і не обов'язково відображають погляди Європейського Союзу чи Європейського виконавчого агентства з питань освіти та культури. Ні Європейський Союз, ні грантодавець не можуть нести за них відповідальність.

Список використаних джерел

1. Long D., Magerko B. What is AI Literacy? Competencies and Design Considerations. Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems. 2020. P. 1–16.
2. Ng D. T. K., Leung J. K. L., Chu S. K. W., Qiao M. S. Conceptualizing AI literacy: An exploratory review. Computers and Education: Artificial Intelligence. 2021. Vol. 2. 100041.
3. UNESCO. Guidance for generative AI in education and research. Paris : UNESCO, 2023.
4. UNESCO. AI competency framework for students. Paris : UNESCO, 2024.
5. Grizzle A., Wilson C., Tuazon R. et al. Media and Information Literate Citizens: Think Critically, Click Wisely! Paris : UNESCO, 2021.
6. Ollama. URL: https://github.com/ollama/ollama (дата звернення: 21.09.2026).
7. GPT4All. URL: https://github.com/nomic-ai/gpt4all (дата звернення: 21.09.2026).
8. Hugging Face Transformers. URL: https://github.com/huggingface/transformers (дата звернення: 21.09.2026).
9. Coalition for Content Provenance and Authenticity (C2PA). URL: https://c2pa.org (дата звернення: 21.09.2026).

Панчук М. О.
Національний університет «Полтавська політехніка імені Юрія Кондратюка», кафедра комп'ютерних та інформаційних технологій і систем, м. Полтава, Україна
Контактний автор: Микита Панчук, e-mail: panchuknikita181@gmail.com, ORCID: 0009-0004-9139-3395

АПАРАТНО-ОРІЄНТОВАНІ МЕТОДИ ЗАБЕЗПЕЧЕННЯ КІБЕРБЕЗПЕКИ ГЕТЕРОГЕННИХ СИСТЕМ РЕР НА БАЗІ SoC FPGA

Анотація. Розглянуто проблему забезпечення кібербезпеки гетерогенних систем радіоелектронної розвідки (РЕР) на базі SoC FPGA в умовах високошвидкісного потокового оброблення сигналів. Запропоновано багаторівневу апаратно-орієнтовану модель захисту, що охоплює зниження інформативності побічного енергетичного каналу, криптографічний захист внутрішніх потоків даних та протидію компрометації конфігурації і фізичному втручанню. Для першого рівня обґрунтовано застосування DVFS і Clock Gating. Другий рівень передбачає інтеграцію AES-GCM у тракт AXI4-Stream. Для третього рівня запропоновано використання PUF та механізму Tamper-Response на основі відстані Махаланобіса. Розрахунки показали, що за вхідного потоку 980 МБ/с пропускна здатність одного конвеєрного AES-128 оцінюється приблизно у 1000 МБ/с. Запропонований підхід формує апаратний контур захисту, адаптований до специфіки гетерогенних SoC FPGA.

Ключові слова: кібербезпека, SoC FPGA, апаратна безпека, РЕР, побічні канали, AES-GCM, PUF, Tamper-Response.

Вступ

Перехід портативних комплексів РЕР, зокрема приладу «Чуйка», до гетерогенної архітектури SoC FPGA забезпечує істотне підвищення продуктивності потокового оброблення сигналів. За наведеними даними, час обчислення спектра зменшується з 14,2 мс до 4,59 мкс, що відповідає прискоренню приблизно у 3090 разів. Одночасно з підвищенням обчислювальної продуктивності зростає інтенсивність внутрішніх потоків даних, яка для розглянутої конфігурації досягає 980 МБ/с [1, 2]. За таких умов збільшується роль внутрішніх апаратних трактів та інтерфейсів AXI4-Stream, через які передаються необроблені й проміжні результати оброблення сигналів.

Зростання швидкості оброблення та обсягів внутрішнього обміну даними розширює поверхню потенційних апаратних атак. До основних ризиків належать витік інформації через побічні канали, отримання доступу до внутрішніх потоків даних, компрометація конфігурації ПЛІС та фізичне втручання в апаратну платформу [1]. Отже, захист гетерогенної системи РЕР повинен охоплювати не лише зовнішні інтерфейси, а й внутрішні обчислювальні та конфігураційні тракти.

Метою роботи є розроблення та математичне обґрунтування багаторівневої апаратно-орієнтованої моделі кіберзахисту гетерогенної системи РЕР на базі SoC FPGA, спрямованої на зниження інформативності побічних каналів, захист внутрішніх потоків даних та протидію компрометації конфігурації і фізичному втручанню.

Матеріали та методи

Запропонована модель передбачає три взаємопов'язані рівні захисту. Перший рівень спрямовано на зниження інформативності побічного енергетичного каналу. Другий рівень забезпечує конфіденційність та автентичність внутрішніх потоків даних. Третій рівень формує апаратний корінь довіри та механізм реагування на фізичне втручання.

Перший рівень базується на керованій зміні енергетичного профілю програмованої логіки за допомогою DVFS та Clock Gating. Конвеєрні обчислювачі ШПФ і CORDIC можуть формувати варіації енергоспоживання, статистично пов'язані з внутрішніми операціями, що створює передумови для застосування диференційного аналізу потужності (DPA) [1, 2]. Тому керування напругою живлення, тактовою частотою та активністю логічних елементів розглядається як засіб зниження інформативності побічного каналу.

Повну потужність системи доцільно представити залежністю

$$P_{tot} = P_{stat} + \alpha C_L V^2 f + P_{leak}(T) \quad (1)$$

де P_{static} – статична складова споживаної потужності; α – коефіцієнт активності перемикань; C_L – еквівалентна ємність навантаження; V – напруга живлення; f – тактова частота; $P_{leak}(T)$ – складова, пов'язана з витоками та температурним режимом.

Відповідно до наведеної залежності, зміна напруги живлення має квадратичний вплив на динамічну складову потужності, тоді як зміна тактової частоти впливає на інтенсивність перемикань логічних елементів. Clock Gating додатково зменшує активність перемикань у функціонально неактивних ділянках логіки. Сукупна зміна зазначених параметрів трансформує статистичні характеристики енергетичного профілю та потенційно знижує його кореляцію з внутрішніми станами обчислювального тракту.

Для первинного оцінювання інформативності енергетичного каналу використовується відношення сигнал/шум:

$$SNR_{DPA} = \frac{\Delta P_{signal}}{\sigma_{noise}} \quad (2)$$

де ΔP_{signal} – різниця потужності для досліджуваних станів, а σ_{noise} – середньоквадратичне відхилення шумової складової. При цьому зниження середньої потужності не може розглядатися як самостійний критерій стійкості до DPA, оскільки результативність атаки визначається статистичною залежністю між вимірюваним енергетичним профілем та внутрішніми операціями системи. Отже, DVFS і Clock Gating доцільно трактувати як засоби зниження інформативності побічного каналу, а не як самостійний криптографічний механізм.

Зниження інформативності побічного каналу не усуває ризику доступу до даних, що передаються внутрішніми апаратними трактами. Тому другий рівень моделі спрямовано на забезпечення конфіденційності та автентичності потоків даних між функціональними блоками ПЛІС. Для захисту ділянки AXI4-Stream між блоком ШПФ та DMA пропонується інтегрувати конвеєрний апаратний модуль AES-GCM. У такій конфігурації криптографічне перетворення виконується безпосередньо в апаратному тракті, що зменшує обсяг незахищених проміжних даних, доступних у межах системи.

Основною умовою застосування криптографічного модуля є забезпечення пропускної здатності, не меншої за інтенсивність вхідного потоку:

$$R_{AES} \geq R_{in} \quad (3)$$

Для розглянутої конфігурації сумарна швидкість потоку трьох каналів АЦП становить $R_{in} = 980$ МБ / с .

За умови оброблення одного байта за такт пропускна здатність AES-128 визначається залежністю $R_{AES} = f_{clk} b$

де f_{clk} – тактова частота криптографічного рушія, а b – кількість байтів, що обробляються за один такт. За наведеними параметрами розрахункова пропускна здатність одного конвеєрного AES-128 становить близько 1000 МБ/с. Таким чином, вона перевищує інтенсивність вхідного потоку приблизно на 2 %:

$$\frac{1000 - 980}{980} \cdot 100\% \approx 2{,}04\%$$

Отриманий результат свідчить про принципову можливість інтеграції криптографічного тракту, однак практична працездатність повної реалізації AES-GCM потребує додаткової RTL-верифікації з урахуванням FIFO, AXI4-Stream, DMA, затримок конвеєра та накладних витрат механізму автентифікації.

Оцінювання доступних апаратних ресурсів Zynq-7020 також вказує на принципову можливість інтеграції криптографічного модуля. За наведеними оцінками, апаратний рушій AES потребує близько 2500 LUT та 18 DSP48, тоді як доступний резерв становить 37 450 LUT та 94 DSP48. Остаточна оцінка потребує врахування BRAM, регістрових ресурсів, тактової інфраструктури, маршрутизації та накладних витрат повного RTL-проєкту.

Оскільки криптографічний захист внутрішніх потоків не вирішує проблему компрометації самої конфігурації ПЛІС, третій рівень моделі спрямовано на формування апаратного кореня довіри та протидію фізичному втручанню. Для цього пропонується використання Physical Unclonable Function (PUF), пов'язаної з конкретним фізичним екземпляром FPGA. Ентропія відгуку PUF оцінюється за Шенноном:

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i \quad (4)$$

де p_i – ймовірність появи відповідного значення відгуку.

За достатніх показників ентропії, стабільності та міжекземплярної унікальності PUF може використовуватися для формування ключового матеріалу, пов'язаного з конкретним фізичним пристроєм [3, 4]. Такий підхід ускладнює перенесення захищеної конфігурації на інший екземпляр FPGA. Водночас ентропія PUF не повинна безпосередньо ототожнюватися з імовірністю клонування, оскільки стійкість визначається сукупністю фізичних, криптографічних та системних характеристик реалізації.

Для виявлення фізичних впливів, зокрема fault injection та clock glitching, пропонується механізм Tamper-Response, що аналізує вектор фізичних і часових параметрів системи. З урахуванням взаємної кореляції параметрів використовується відстань Махаланобіса:

$$D_M(\vec{S}) = \sqrt{(\vec{S} - \vec{\mu})^{\mathrm{T}} \Sigma^{-1} (\vec{S} - \vec{\mu})} \quad (5)$$

де $\vec{\mu}$ – вектор математичних сподівань параметрів у легітимному режимі, а Σ – їх коваріаційна матриця.

Умова формування події фізичного втручання визначається співвідношенням

$$D_M(\mathbf{x}) > \tau_T \quad (6)$$

де τ_T – порогове значення Tamper-Response. Його необхідно визначати на репрезентативній вибірці штатних режимів роботи з урахуванням температурних варіацій, режимів DVFS та перехідних процесів. Перевищення встановленого порога ініціює перехід системи у захищений стан, блокування або стирання ключового матеріалу у захищеному сховищі. Таким чином, механізм спрямовано на скорочення часу доступності конфігураційних секретів після виявлення фізичного впливу.

Результати та обговорення

За результатами дослідження сформовано багаторівневу апаратно-орієнтовану модель кіберзахисту гетерогенної системи РЕР на базі SoC FPGA. Перший рівень забезпечує зниження інформативності побічного енергетичного каналу шляхом керованої зміни параметрів енергоспоживання за допомогою DVFS та Clock Gating. Другий рівень передбачає криптографічний захист внутрішніх потоків шляхом інтеграції AES-GCM у тракт AXI4-Stream. Розрахункова пропускна здатність одного конвеєрного AES-128 становить близько 1000 МБ/с за інтенсивності вхідного потоку 980 МБ/с. Третій рівень базується на PUF та механізмі Tamper-Response з використанням відстані Махаланобіса для виявлення аномальних відхилень фізичних і часових параметрів.

Отримані результати свідчать, що запропоновані механізми охоплюють різні класи апаратних загроз і можуть інтегруватися безпосередньо в апаратний тракт SoC FPGA. Водночас результати розрахунку пропускної здатності AES-128 та оцінювання ресурсів Zynq-7020 слід розглядати як попереднє обґрунтування принципової реалізовності. Остаточне підтвердження ефективності моделі потребує експериментальної RTL-верифікації повного тракту, статистичного оцінювання стійкості до DPA та визначення порога Tamper-Response за експериментальними даними.

Висновки

Запропоновано багаторівневу апаратно-орієнтовану модель забезпечення кібербезпеки гетерогенних систем РЕР на базі SoC FPGA. Модель поєднує три взаємодоповнювальні механізми: зниження інформативності побічних енергетичних каналів, криптографічний захист внутрішніх потоків даних та апаратне реагування на фізичне втручання.

Для першого рівня обґрунтовано використання DVFS і Clock Gating як засобів керування енергетичним профілем. Для другого рівня визначено умову пропускної здатності криптографічного тракту та показано, що розрахункова продуктивність одного AES-128 на рівні близько 1000 МБ/с перевищує заданий потік 980 МБ/с приблизно на 2 %. Для третього рівня запропоновано використання PUF для формування апаратно-зв’язаного ключового матеріалу та Tamper-Response на основі відстані Махаланобіса.

Подальші дослідження доцільно спрямувати на експериментальне оцінювання стійкості до DPA за статистичними критеріями, RTL-верифікацію повного AES-GCM-тракту, визначення оптимального порога Tamper-Response та дослідження можливості використання доступних ресурсів DSP48 для реалізації апаратних детекторів аномалій на основі TinyML.

Список використаних джерел

1. Zhang J., Qu G. Recent Attacks and Defenses on FPGA-based Systems. ACM Transactions on Reconfigurable Technology and Systems. 2019. Vol. 12, No. 3. P. 1–24. DOI: 10.1145/3340557.
2. Drimer S. Security for volatile FPGAs. Cambridge : University of Cambridge, Computer Laboratory, 2009. 169 p. Technical Report UCAM-CL-TR-763. DOI: 10.48456/tr-763.
3. Zhyvylo Y., Kuchma Y. Mathematical modeling of intellectual and cryptographic protection of authentication keys. Information Technology and Security. 2025. Vol. 13, No. 2. P. 162–177. DOI: 10.20535/2411-1031.2025.13.2.344591.
4. Фесенко Т., Калашнікова Ю. Математичні аспекти комбінованого застосування алгоритму AES та стеганографічних методів у захисті ключів автентифікації. Інформаційні технології та безпека. 2025. № 13 (2). С. 178–191. DOI: 10.20535/2411-1031.2025.13.2.344592.

Перепечай А. С., Лисенко Д. Е.
Національний університет «Чернігівська політехніка», кафедра інформаційних та комп'ютерних систем, м. Чернігів, Україна
Контактний автор: Андрій Перепечай, e-mail: andreyperepechay@gmail.com

ІНТЕЛЕКТУАЛЬНА СИСТЕМА ОЦІНЮВАННЯ ТА РАНЖУВАННЯ ПРІОРИТЕТНОСТІ СОЦІАЛЬНИХ ЗВЕРНЕНЬ ГРОМАДЯН

Анотація. Зростання кількості звернень громадян до соціальних служб створює високе навантаження на операторів та підвищує ризик затримок під час реагування на критичні ситуації. У роботі запропоновано концепцію та архітектуру інтелектуальної вебсистеми підтримки прийняття рішень для автоматизованого оцінювання та ранжування пріоритетності таких звернень. Дослідження базується на поєднанні семантичного аналізу незаструктурованого тексту за допомогою методів опрацювання природної мови (NLP) та багатокритеріального оцінювання формалізованих характеристик звернення. Система передбачається у вигляді вебзастосунку на основі вільного та відкритого програмного забезпечення (Python, FastAPI та SQLite). Запропонований комбінований підхід має забезпечити врахування контекстної інформації зі змісту звернення, формування впорядкованої черги опрацювання та надання працівнику пояснення щодо визначеного рівня пріоритету. Практичне значення роботи полягає у створенні інструменту підтримки прийняття рішень під час опрацювання соціальних звернень.

Ключові слова: соціальні звернення громадян; пріоритетність; ранжування; інтелектуальна система; опрацювання природної мови; багатокритеріальне оцінювання; підтримка прийняття рішень.

Вступ

Зі збільшенням кількості соціальних звернень громадян зростає навантаження на працівників, які здійснюють їх опрацювання та визначають черговість реагування. За великої кількості звернень ручний аналіз кожного випадку може потребувати значних часових ресурсів, а визначення пріоритетності залежить від багатьох факторів, зокрема змісту звернення, рівня терміновості, кількості осіб, яких стосується проблема, соціальних обставин та тривалості ситуації. Особливу складність становлять звернення у вільній текстовій формі, оскільки важливі ознаки проблеми можуть бути не представлені у вигляді формалізованих параметрів [1].

Перспективним підходом до вирішення цієї проблеми є використання методів опрацювання природної мови для аналізу текстового змісту звернень у поєднанні з багатокритеріальним оцінюванням їх структурованих характеристик. Такий підхід дає змогу враховувати як інформацію, що міститься безпосередньо в тексті, так і формалізовані критерії оцінювання [2, 3].

Метою роботи є розроблення інтелектуальної вебсистеми для автоматизованого оцінювання та ранжування пріоритетності соціальних звернень громадян на основі поєднання семантичного аналізу тексту та багатокритеріального оцінювання. Для досягнення поставленої мети передбачається розробити алгоритм визначення пріоритетності, реалізувати програмний прототип вебсистеми та провести експериментальне порівняння окремого семантичного, критеріального та комбінованого підходів.

Матеріали та методи

Для проведення дослідження та розроблення програмного прототипу інтелектуальної системи оцінювання та ранжування пріоритетності соціальних звернень громадян планується використати набір даних, що міститиме текстові описи звернень та їх формалізовані характеристики. Текстова частина даних використовуватиметься для семантичного аналізу змісту звернення, а структуровані характеристики — для багатокритеріального оцінювання його пріоритетності. До таких характеристик можуть належати рівень терміновості, кількість осіб, яких стосується проблема, наявність соціально вразливих категорій, тривалість проблемної ситуації та рівень потенційного ризику. На етапі дослідження передбачається формування тестового набору звернень із визначеними експертними або заздалегідь

встановленими рівнями пріоритетності, який використовуватиметься для порівняння різних підходів.

Програмна частина системи планується до реалізації мовою Python з використанням вільного та відкритого програмного забезпечення [4]. Для створення серверної частини вебсистеми передбачається використання фреймворку FastAPI, який забезпечуватиме оброблення запитів, взаємодію між вебінтерфейсом, базою даних та алгоритмічними модулями [5]. Вебінтерфейс планується реалізувати за допомогою HTML, CSS та JavaScript без використання надмірно ресурсоємних клієнтських фреймворків.

Для зберігання інформації про соціальні звернення та результати їх опрацювання планується використати SQLite [6]. Вибір цієї системи керування базою даних зумовлений її простотою розгортання, невисокими вимогами до обчислювальних ресурсів та достатністю функціональних можливостей для дослідницького прототипу. У базі даних передбачається зберігати текст звернення, його структуровані характеристики, проміжні результати оцінювання, визначений рівень пріоритетності та сформований рейтинг.

Для оброблення та аналізу даних планується використати бібліотеки NumPy та Pandas, а для реалізації методів машинного навчання та класифікації — scikit-learn. Семантичний аналіз тексту передбачається реалізувати із застосуванням методів опрацювання природної мови (NLP). На початковому етапі доцільно використовувати відносно прості та відтворювані методи представлення тексту, зокрема TF-IDF, у поєднанні з алгоритмами класифікації [7]. Такий підхід не потребує використання великих мовних моделей або потужних графічних процесорів і відповідає вимогам до ресурсомісткості проєкту.

Методика дослідження передбачає порівняння трьох варіантів визначення пріоритетності соціальних звернень. Перший варіант базуватиметься лише на багатокритеріальному оцінюванні структурованих характеристик звернення. Другий використовуватиме лише семантичну інформацію, отриману з тексту звернення. Третій, комбінований варіант поєднуватиме результати обох підходів.

Загальну схему комбінованого оцінювання на етапі проєктування можна представити як:

$$\text{FinalScore} = \alpha \cdot \text{SemanticScore} + \beta \cdot \text{CriteriaScore},$$

де SemanticScore — оцінка, отримана в результаті аналізу тексту звернення, CriteriaScore — результат багатокритеріального оцінювання структурованих характеристик, а α і β — вагові коефіцієнти. При цьому передбачається, що $\alpha + \beta = 1$. Конкретні значення ваг не фіксуються заздалегідь, а мають бути обґрунтовані під час експериментального дослідження.

Після отримання підсумкової оцінки планується виконувати класифікацію звернень за рівнями пріоритетності, наприклад високий, середній та низький, а також формувати впорядковану чергу звернень. Тобто, система має забезпечити не лише визначення категорії окремого звернення, а й можливість порівняння декількох звернень між собою [8].

Для оцінювання запропонованих підходів передбачається провести експериментальне порівняння критерійного, семантичного та комбінованого методів. Залежно від сформованого набору даних для оцінювання якості класифікації можуть використовуватися стандартні метрики, зокрема accuracy, precision, recall та F1-score [9]. Для аналізу правильності сформованого порядку звернень планується додатково застосувати метрики ранжування, вибір яких буде визначено після формування набору даних та еталонної оцінки пріоритетності.

Розроблення та тестування програмного прототипу планується виконувати в середовищі Visual Studio Code на персональному комп'ютері без використання спеціалізованого серверного або графічного обладнання. Застосування Python, FastAPI та SQLite дає можливість розгорнути систему локально та забезпечити відтворюваність експериментів. Для візуалізації результатів дослідження передбачається використання Matplotlib або Plotly, зокрема для побудови графіків порівняння якості окремих методів [10].

Результати

У результаті проведеного дослідження сформовано концепцію інтелектуальної вебсистеми для автоматизованого оцінювання та ранжування пріоритетності соціальних

звернень громадян. Визначено основну послідовність оброблення звернення: реєстрація та підготовка вхідних даних, семантичний аналіз текстового опису проблеми, оцінювання структурованих характеристик, визначення підсумкового рівня пріоритетності та формування впорядкованої черги звернень.

Запропонована структура системи передбачає одночасне використання двох груп інформації. Перша група представлена текстовим змістом звернення, на основі якого планується визначати семантичні ознаки потенційної терміновості або критичності ситуації. Друга група містить формалізовані характеристики звернення, зокрема рівень терміновості, кількість осіб, яких стосується проблема, соціальні обставини, тривалість ситуації та потенційний ризик.

На основі отриманих оцінок передбачається формування комбінованого показника пріоритетності. Узагальнена схема запропонованого підходу наведена на рис. 1.

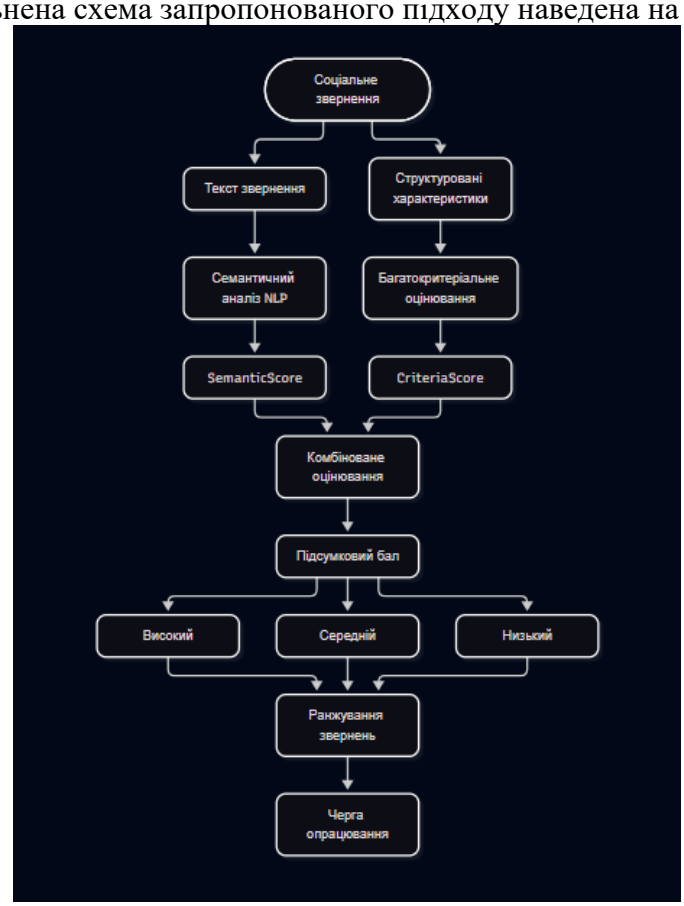


Рисунок 1 – Структурна схема комбінованого оцінювання пріоритетності соціальних звернень

Сформована модель також передбачає можливість порівняння трьох підходів до визначення пріоритетності: критеріального, семантичного та комбінованого. Це створює основу для подальшого експериментального дослідження ефективності запропонованого методу. У таблиці 1 наведено порівнювані підходи до оцінювання пріоритетності звернень громадян.

Таблиця 1 – Порівнювані підходи до оцінювання пріоритетності

Підхід	**Вхідні дані**	**Результат**
Критеріальний	Структуровані характеристики звернення	CriteriaScore
Семантичний	Текст звернення	SemanticScore
Комбінований	Текст + структуровані характеристики	FinalScore

Отримана модель дозволяє перейти від простого зберігання соціальних звернень до формування автоматизованої рекомендації щодо черговості їх опрацювання. При цьому остаточне рішення щодо реагування залишається за працівником соціальної служби, а система виконує функцію підтримки прийняття рішення.

Обговорення

Запропонований підхід орієнтований на поєднання інформації, яка міститься безпосередньо в тексті соціального звернення, із формалізованими характеристиками проблемної ситуації. На відміну від підходу, що використовує лише структуровані критерії, семантичний аналіз дає можливість враховувати додаткову контекстну інформацію, яка може бути відсутня серед заданих полів форми. Водночас використання лише текстового аналізу не враховує частину явно заданих характеристик звернення, тому комбінування двох джерел інформації розглядається як перспективний напрям дослідження.

Обмеженням запропонованого підходу є залежність якості оцінювання від якості та репрезентативності набору даних, коректності визначених критеріїв та наявності достовірної еталонної оцінки пріоритетності. Крім того, автоматичний аналіз тексту може мати труднощі з неоднозначними формулюваннями та контекстом окремих звернень.

Подальший розвиток роботи передбачає програмну реалізацію запропонованої моделі, формування набору тестових даних та експериментальне порівняння семантичного, критеріального і комбінованого підходів. Перспективним також є розширення набору критеріїв та вдосконалення механізму пояснення сформованої рекомендації працівнику соціальної служби.

Висновки

У роботі сформовано концепцію інтелектуальної вебсистеми для автоматизованого оцінювання та ранжування пріоритетності соціальних звернень громадян. Запропоновано підхід, що поєднує семантичний аналіз текстового змісту звернення з багатокритеріальним оцінюванням його структурованих характеристик. Визначено основну послідовність оброблення звернення: від реєстрації та аналізу вхідних даних до визначення рівня пріоритетності та формування впорядкованої черги опрацювання.

Запропонована модель дає змогу сформувати основу для експериментального порівняння семантичного, критеріального та комбінованого підходів. Подальша робота передбачає реалізацію програмного прототипу на основі Python, FastAPI та SQLite, формування тестового набору даних і проведення експериментального оцінювання якості запропонованих методів.

Список використаних джерел

1. Kim, N., Hong, S. Automatic classification of citizen requests for transportation using deep learning: Case study from Boston city. Information Processing & Management. 2021. Vol. 58, No. 1. Article 102410. DOI: 10.1016/j.ipm.2020.102410. Access mode: https://www.sciencedirect.com/science/article/pii/S0306457320309055
2. Zhang, J., Geng, Q., Jin, J. EKLI-Attention: An integrated attention mechanism for classifying citizen requests in government–citizen interactions. Information Processing & Management. 2025. Vol. 62, No. 6. Article 104237. DOI: 10.1016/j.ipm.2025.104237. Access mode: https://www.sciencedirect.com/science/article/abs/pii/S0306457325001785
3. Greco, S., Słowiński, R., Wallenius, J. Fifty years of multiple criteria decision analysis: From classical methods to robust ordinal regression. European Journal of Operational Research. 2025. Vol. 323, No. 2. P. 351–377. DOI: 10.1016/j.ejor.2024.07.038.
4. Python Software Foundation. Python 3 Documentation [Електронний ресурс]. URL: https://docs.python.org/3/ (дата звернення: 25.09.2026).
5. FastAPI. FastAPI Documentation [Електронний ресурс]. URL: https://fastapi.tiangolo.com/ (дата звернення: 25.09.2026).
6. SQLite. SQLite Documentation [Електронний ресурс]. URL: https://www.sqlite.org/docs.html (дата звернення: 25.09.2026).
7. scikit-learn developers. TfidfVectorizer — scikit-learn documentation [Електронний ресурс]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html (дата звернення: 25.09.2026).
8. Purohit, H., Castillo, C., Pandey, R. Ranking and grouping social media requests for emergency services using serviceability model. Social Network Analysis and Mining. 2020. Vol. 10. Article 22. DOI: 10.1007/s13278-020-0633-3. Access mode: https://link.springer.com/article/10.1007/s13278-020-0633-3
9. scikit-learn developers. Metrics and scoring: quantifying the quality of predictions — scikit-learn documentation [Електронний ресурс]. URL: https://scikit-learn.org/stable/modules/model_evaluation.html (дата звернення: 25.09.2026).
10. Matplotlib. Matplotlib Documentation [Електронний ресурс]. URL: https://matplotlib.org/stable/ (дата звернення: 25.09.2026).

Плахтій М. О.
Приватний вищий навчальний заклад «Університет сучасних технологій», кафедра комп'ютерних наук, м. Київ, Україна
e-mail: mp@ukr.net, ORCID: 0000-0003-3805-0591

НАНОСТРУКТУРНА АРХІТЕКТУРА КВАНТОВИХ НАКОПИЧУВАЧІВ

Анотація. У роботі розглянуто наноструктурну архітектуру квантових накопичувачів як багаторівневу гетеросистему, в якій енергетичні та когерентні властивості визначаються просторовою організацією наношарів, характеристиками міжфазних меж і взаємодією з навколишнім середовищем. Обґрунтовано доцільність застосування гетероструктур та методів атомарно контрольованого синтезу для формування заданих потенціальних профілів і локалізації квантових станів. Для кількісного оцінювання стабільності запропоновано використовувати характерні часи енергетичної релаксації ( ) та фазової когерентності ( ). Показано, що контроль дефектності, геометрії наношарів і міжфазних переходів створює передумови для зменшення каналів енергетичних втрат і декогеренції та, відповідно, підвищення стабільності квантових станів.

Ключові слова: наноструктура, квантові накопичувачі, гетероструктура, квантова яма, когерентність, декогеренція, наношари, молекулярно-променева епітаксія.

Вступ

Розвиток квантових технологій потребує матеріальних платформ, здатних не лише формувати дискретні енергетичні стани, а й забезпечувати їхню стабільність у процесі взаємодії з навколишнім середовищем. Для наноструктурних платформ квантових накопичувачів принциповим є забезпечення керованого формування та переходів між енергетичними станами за одночасного обмеження процесів енергетичної релаксації та фазової декогеренції. У цьому контексті перспективним напрямом є використання наноструктурованих гетеросистем, у яких геометрія шарів і властивості міжфазних меж дають змогу цілеспрямовано формувати потенціальний профіль та енергетичний спектр системи.

Основна науково-технічна проблема полягає у необхідності одночасного забезпечення просторової локалізації квантових станів, енергетичної стабільності та зменшення взаємодії з каналами декогеренції. Відповідно, метою роботи є обґрунтування наноструктурної архітектури квантового накопичувача та визначення фізичних параметрів, придатних для кількісного оцінювання стабільності сформованих квантових станів.

Матеріали та методи

Наноструктурну архітектуру доцільно розглядати як багатошарову гетеросистему з керованими параметрами товщини та послідовності шарів, потенціального профілю і властивостей міжфазних меж. У загальному випадку теоретичну модель такої системи можна представити гамільтоніаном

$$\hat{H} = \hat{H}_0 + \hat{H}_{\text{int}} + \hat{H}_{\text{env}} \qquad (1)$$

де $\hat{H}_0$ характеризує власну енергетичну структуру наносистеми, $\hat{H}_{\text{int}}$ визначає взаємодію між її компонентами, а $\hat{H}_{\text{env}}$ описує взаємодію системи з навколишнім середовищем.

Керування геометрією та складом наношарів на субнанометровому рівні може здійснюватися методом молекулярно-променевої епітаксії. Такий підхід забезпечує високий ступінь контролю складу та геометрії епітаксійних структур і тому придатний для формування багатошарових квантових систем [1]. Експериментальні дослідження ультратонких $TiSe_2$-плівок також демонструють можливість близького до пошарового росту та встановлюють залежність кореляційних властивостей від концентрації поверхневих дефектів [2].

Для кількісного оцінювання енергетичної стабільності використаємо експоненційну модель релаксації

$$E(t) = E_0 \exp\left(-\frac{t}{\tau_E}\right) \qquad (2)$$

де E_0 – початкова енергія накопиченого стану, а τ_E – характерний час енергетичної релаксації. При τ_E маємо $E(\tau_E) = \frac{E_0}{e} \approx 0,368E_0$, тобто протягом одного часу релаксації зберігається приблизно 36,8 % початкової енергетичної складової.

Аналогічним чином фазову стабільність можна описати залежністю

$$C(t) = C_0 \exp\left(-\frac{t}{\tau_\phi}\right) \qquad (3)$$

де C_0 – початковий рівень когерентності, а τ_ϕ – характерний час фазової когерентності. Отже, параметри τ_E і τ_ϕ можуть використовуватися як взаємодоповнювальні часові характеристики стабільності наноструктурної архітектури.

Результати

Запропонована архітектура ґрунтується на просторовому формуванні наношарів із заданими електронними та енергетичними характеристиками. Варіювання товщини шарів і властивостей міжфазних переходів дає змогу формувати потенціальні профілі та квантові ями, у яких забезпечується просторова локалізація носіїв і керування дискретними енергетичними станами. Таким чином, геометричні параметри гетероструктури безпосередньо пов’язуються з умовами формування та стабілізації квантових станів.

Важливим фактором при цьому є структурна однорідність наношарів. Вакансії, міжвузлові атоми та інші дефекти змінюють локальний потенціал і можуть формувати додаткові канали розсіювання та декогеренції. Зокрема, експериментальні дослідження ультратонких $TiSe_2$-плівок показують зменшення кореляційної довжини електронних станів зі збільшенням концентрації поверхневих дефектів [2]. Цей результат є експериментальним прикладом взаємозв’язку між структурною дефектністю та кореляційними властивостями ультратонких квантових систем і підтверджує доцільність контролю дефектності під час формування наноструктурної архітектури.

Однак стабільність квантового стану визначається не лише внутрішніми властивостями наноструктури. Як відкритій квантовій системі, їй притаманна взаємодія з навколишнім середовищем, яка може зумовлювати як енергетичну релаксацію, так і фазову декогеренцію. Тому для комплексного оцінювання архітектури доцільно розглядати співвідношення характерних часових масштабів

$$R_\tau = \frac{\tau_\phi}{\tau_E} \qquad (4)$$

Параметр (R_τ) дає змогу зіставити характерні часові масштаби фазової когерентності та енергетичної релаксації. Наприклад, за умови ($R_\tau \gg 1$) фазова когерентність характеризується значно довшим часовим масштабом, ніж енергетична релаксація. Водночас таке співвідношення є критерієм порівняльного аналізу архітектур, а не самостійним експериментальним показником ефективності конкретної наноструктури.

Отже, стабільність наноструктурної архітектури визначається сукупністю взаємопов’язаних параметрів: товщиною та послідовністю шарів, властивостями міжфазних меж, рівнем структурної дефектності, формою потенціального профілю та характеристиками взаємодії з навколишнім середовищем. Комплексне керування цими параметрами створює передумови для формування відтворюваних енергетичних станів і підвищення їхньої стабільності.

Обговорення

Отримані аналітичні співвідношення свідчать, що стабільність наноструктурної архітектури не може бути пов’язана лише з однією характеристикою матеріалу. Вона формується сукупною дією просторових, енергетичних і динамічних параметрів системи.

Формування квантових ям забезпечує локалізацію станів, однак сама локалізація не гарантує їх тривалої стабільності. Водночас зменшення дефектності та контроль характеристик міжфазних меж потенційно дають змогу обмежити додаткові канали розсіювання, що є важливим чинником збереження когерентних властивостей.

Застосування гетероструктур також узгоджується із загальним напрямом досліджень гібридних квантових систем, у яких поєднання різних фізичних платформ розглядається як засіб формування керованих квантових властивостей і масштабованих технологічних структур [3]. Водночас наведене узгодження не слід трактувати як безпосередню експериментальну верифікацію запропонованої архітектури.

Основним обмеженням роботи є відсутність експериментальних даних для конкретного складу матеріалів, товщини наношарів, концентрації дефектів і температурного режиму. Тому запропоновані залежності та часові критерії слід розглядати як теоретичну основу для подальшого чисельного моделювання та експериментальної перевірки. Наступним етапом дослідження має стати встановлення кількісного зв'язку між параметрами наноструктури та значеннями і .

Висновки

Наноструктурну архітектуру квантового накопичувача доцільно представляти як багаторівневу гетеросистему, у якій геометрія наношарів, характеристики міжфазних меж і структурна дефектність пов'язані з енергетичними та когерентними властивостями квантових станів. На основі аналітичної моделі обґрунтовано використання характерних часів енергетичної релаксації ( ) та фазової когерентності ( ) як взаємодоповнювальних параметрів оцінювання стабільності архітектури.

Показано, що експоненційні залежності енергетичної складової та рівня когерентності дають змогу формалізувати часові характеристики втрати енергії та фазової стабільності. Встановлено доцільність використання співвідношення ( ) для порівняльного аналізу різних наноструктурних архітектур. При цьому отримані результати мають теоретичний характер і не замінюють експериментальної оцінки конкретних матеріалів та конструкцій.

Подальші дослідження доцільно спрямувати на чисельне моделювання конкретних гетероструктур, визначення залежностей ( ) і ( ) від товщини наношарів, характеристик міжфазних меж і концентрації дефектів, а також на експериментальну перевірку запропонованих часових критеріїв.

Список використаних джерел

1. M. B. Panish, Molecular Beam Epitaxy. Science 208, 916-922 (1980). DOI:10.1126/science.208.4446.916.

2. Peng, J.-P., Guan, J.-Q., Zhang, H.-M., Song, C.-L., Wang, L., He, K., Xue, Q.-K., & Ma, X.-C. (2015). Molecular beam epitaxy growth and scanning tunneling microscopy study of $TiSe_2$ ultrathin films. Physical Review B, 91(12), 121113. https://doi.org/10.1103/PhysRevB.91.121113.

3. Xiang, Z.-L., Ashhab, S., You, J. Q., & Nori, F. (2013). Hybrid quantum circuits: Superconducting circuits interacting with other quantum systems. Reviews of Modern Physics, 85(2), 623–653. https://doi.org/10.1103/RevModPhys.85.623.

Пода О. Л.
Національний університет «Чернігівська політехніка», кафедра інформаційних та комп'ютерних систем, Чернігів, Україна
Контактний автор: Ольга Пода, e-mail: olya.poda@gmail.com, ORCID: 0009-0002-2275-1465

АРХІТЕКТУРНО-ІНЖЕНЕРНІ РІШЕННЯ ЗАБЕЗПЕЧЕННЯ МОНІТОРИНГУ ШІ-АСИСТЕНТА ТА БЕЗПЕКИ ДАНИХ В МОБІЛЬНІЙ ІНФОРМАЦІЙНІЙ СИСТЕМІ ПСИХОЕМОЦІЙНОЇ ПІДТРИМКИ / ARCHITECTURE AND EMPIRICAL SECURITY TESTING OF A MOBILE PSYCHO-EMOTIONAL SUPPORT APPLICATION WITH AN AI ASSISTANT

Анотація. Стрімке зростання попиту на послуги у сфері психічного здоров'я на фоні дефіциту фахівців зумовлює впровадження чат-ботів, побудованих на основі великих мовних моделей (LLM) як засобів оперативного психоемоційного супроводу. Однак ймовірнісна природа LLM породжує специфічні проблеми для розробників: нестабільність структури відповідей та недостатньо розвинені системи безпеки даних. Основною метою дослідження є аналіз та розробка архітектурно-інженерних рішень для мобільної системи підтримки з інтегрованим ШІ-асистентом, які забезпечують надійність, конфіденційність і контроль безпеки відповідей. У результаті запропонована серверлес-архітектура на основі Firebase Cloud Functions із окремим модулем моніторингу кризових станів. Валідність правил доступу до даних підтверджено 20 тестами, а три провідні LLM пройшли порівняльні тести бенчмарків на точність розпізнавання кризових маркерів. Результати дослідження підтверджують необхідність архітектурного відокремлення функцій безпеки від основної логіки формування відповіді.

Ключові слова: штучний інтелект; психоемоційна підтримка; serverless-архітектура; тестування систем з ШІ; Firebase; безпека даних; LLM

Вступ

Проблеми психічного здоров'я залишаються особливо актуальними: за оцінками, кожна восьма людина у світі бореться з таким розладом, а понад 70% не отримують професійної допомоги через нестачу фахівців та обмежене фінансування інфраструктури, пов'язаної з наданням таких послуг. Чат-боти на основі LLM розглядаються як допоміжний інструмент цілодобової підтримки, проте їхня розробка супроводжується низкою специфічних інженерних викликів: традиційне тестування програмного забезпечення спирається на детермінізм, тоді як поведінка LLM залежить від коду, інференції моделі та формулювання запиту [1]. Задокументований досвід інтеграції LLM у продакшн-застосунки фіксує стійкі категорії відмов — від порушення формату відповіді до нестабільності поведінки поміж різних моделей [2]. Наявні механізми контролю поведінки LLM-агентів здебільшого реактивні — виявляють небезпечну поведінку лише тоді, коли вона вже настала або є неминучою [3].

Метою роботи є розробка архітектурно-інженерних рішень для мобільної інформаційної системи психоемоційної підтримки з інтегрованим ШІ-асистентом, що забезпечують надійність, захист персональних даних та моніторинг безпечності відповідей моделі. Завданнями дослідження є: обґрунтувати архітектуру серверлес-декомпозиції з відокремленим модулем моніторингу; верифікувати модель доступу до даних; порівняти LLM за здатністю розпізнавати кризові стани користувача.

Матеріали та методи

Клієнтську частину реалізовано на Flutter (BSD-3-Clause, відкритий SDK) з архітектурним патерном BLoC. Серверна логіка побудована як серверлес-декомпозиція засобами Firebase Cloud Functions: AI-проксі функція оркеструє запит до зовнішнього LLM API та зчитує контекст розмови з Firestore, окрема функція моніторингу аналізує відповідь на

кризові ознаки перед поверненням результату користувачу. Автентичність клієнта підтверджується Firebase App Check; ключ доступу до LLM API зберігається в Secret Manager і не потрапляє на клієнт.

Перевірку правил доступу до даних Firestore проведено засобами автоматизованого тестування: Firebase Local Emulator Suite (firebase-tools, відкритий код), бібліотека rules-unit-testing (Apache-2.0) та тестовий фреймворк Mocha під Node.js. Розроблено 21 тест, які охоплюють розмежування доступу за власником, захист від підміни ідентифікатора в тілі запиту, незмінність потоку повідомлень та валідацію схеми документа, враховуючи обмеження довжини тексту та заборона недекларованих полів.

Для порівняння LLM за здатністю розпізнавання кризових станів сформовано набір із 18 тестових фраз трьох категорій: нейтральні, стрес та кризові маркери, кожну з яких передано 3 LLM-провайдерам через уніфікований класифікаційний промт із фіксованим лімітом токенів відповіді. Для кожного запиту зафіксовано точність класифікації, затримку та орієнтовну вартість. Вихідний код тестів та бенчмарків доступний у репозиторії: https://github.com/olhapoda/ai-support-system-testing.

Результати

Розроблена архітектура поділена на три зони довіри: клієнт, керований Firebase-бекенд та зовнішній LLM-сервіс з чітким розділенням відповідальності між AI-проксі-функцією та модулем моніторингу за принципом єдиної відповідальності. Дана архітектура вказана на рисунку 1. Послідовність обробки запиту передбачає розгалуження на emergency-режим: у разі виявлення кризових ознак система повертає заздалегідь підготовлену відповідь із ресурсами допомоги, не покладаючись на подальшу генерацію LLM. Дана послідовність наведена на рисунку 2.

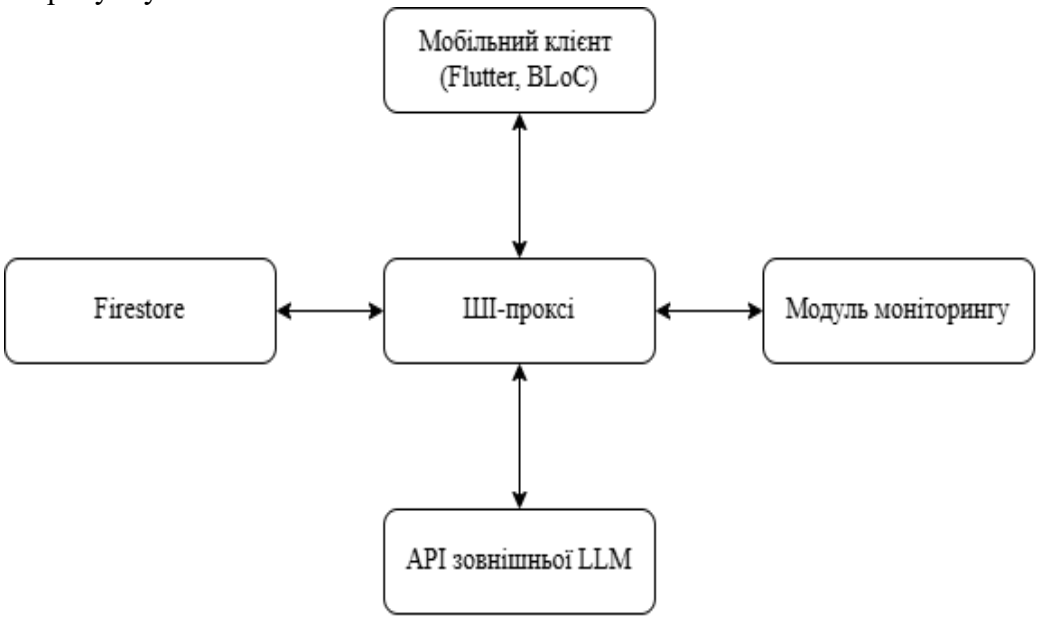


Рисунок 1 – Схема архітектури системи

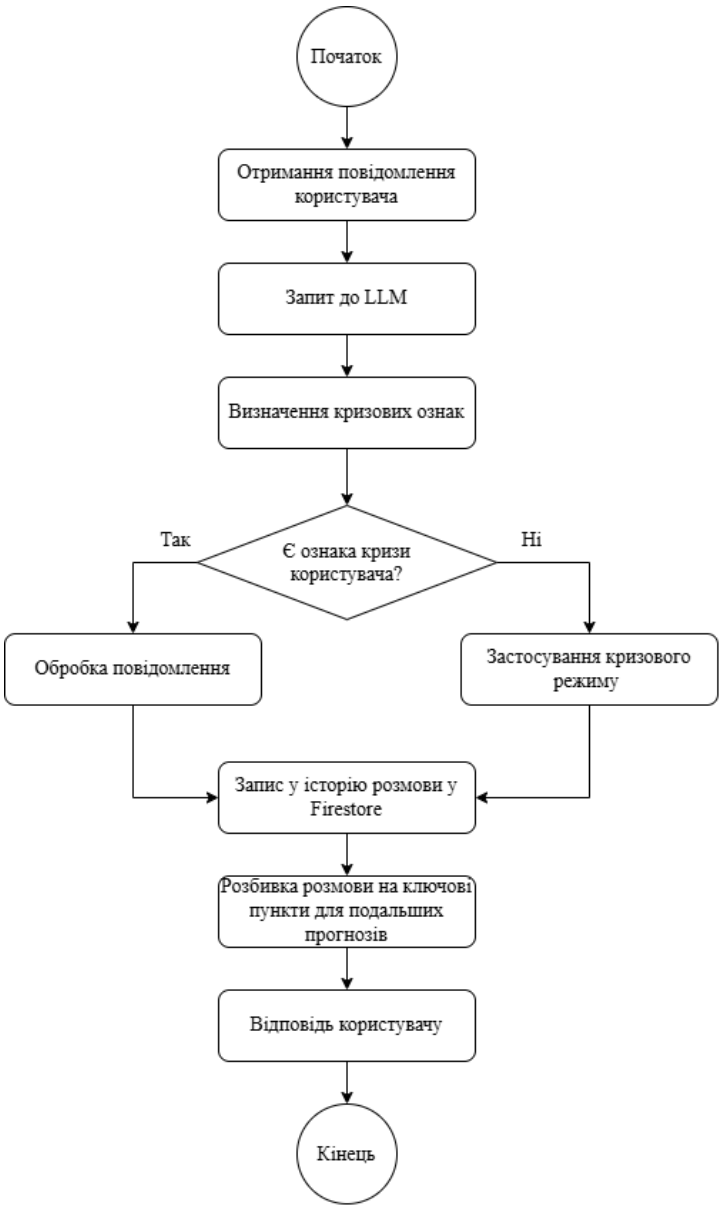


Рисунок 2 – Схема послідовності обробки запиту

Усі 20 тестових сценаріїв правил доступу пройшли успішно; середній час обробки запиту становив 97,7 мс.

Порівняльні тести LLM-моделей на основі набору з 18 фраз показали точність класифікації від 79,2% до 87,5%; найбільш критичним результатом є кількість пропущених кризових сигналів серед 6 фраз кризової категорії для кожної моделі: модель GPT 6 Luna пропустила 2 такі сигнали.

Обговорення

Архітектурне відокремлення модуля моніторингу від основної логіки генерації відповіді дозволяє приймати рішення про валідацію результату незалежно від ймовірнісного виводу LLM, який узгоджується з принципом гарантованого детермінованого fallback для критичних сценаріїв [2]. Виявлена варіативність поведінки моделей при однаковому запиті, в тому числі залежність від ліміту токенів відповіді для reasoning-моделей емпірично підтверджує задокументовану в літературі нестабільність поведінки різних LLM [2].

Слід зазначити обмеження дослідження: вибірка з 24 тестових фраз є пілотною і недостатньою для статистично значущих висновків. Використані LLM-сервіси є пропрієтарними, тоді як методологія розробки й тестування архітектури повністю базується на FOSS-інструментарії, такому як Flutter, Node.js, Firebase CLI та Mocha, що забезпечує відтворюваність дослідження незалежно від постачальника моделі. Наявні механізми

моніторингу LLM-агентів здебільшого реактивні; перспективним напрямом є перехід до проактивного прогнозування ризику на основі ймовірнісного моделювання траєкторії стану [3].

Висновки

У ході цього дослідження було запропоновано архітектуру серверлес-декомпозиції мобільної системи психоемоційної підтримки яка включає модуль моніторингу безпеки відповідей ШІ-асистента. Верифікація моделі доступу до даних Firestore показала успішне проходження всіх 21 тесту. Порівняльні тести 3 LLM-моделей виявили значні відмінності в точності розпізнавання кризових станів користувача, що підкреслює важливість вибору моделі для таких цілей.

Подальші дослідження можуть зосереджуватися на переході до проактивного моніторингу на основі ймовірнісного прогнозування, а також на розробці гібридної архітектури, яка б частково використовувала локальні моделі для зменшення обсягу чутливих даних, що передаються стороннім сервісам.

Список використаних джерел

1. Dobslaw F., Feldt R., Yoon J., Yoo S. Challenges in Testing Large Language Model Based Software: A Faceted Taxonomy. arXiv, 2025. DOI: 10.48550/arXiv.2503.00481.
2. Oliveira W. Less Is More: Engineering Challenges of On-Device Small Language Model Integration in a Mobile Application. arXiv, 2026. DOI: 10.48550/arXiv.2604.24636.
3. Wang H., Poskitt C. M., Wei J., Sun J. ProbGuard: Proactive Runtime Monitoring for LLM Agent Safety via Probabilistic Prediction. Proc. of the 41st IEEE/ACM ASE, 2026 (до друку). arXiv:2508.00500.
4. Firebase CLI (firebase-tools). URL: https://github.com/firebase/firebase-tools (дата звернення: 29.09.2026).
5. Flutter SDK. URL: https://flutter.dev (дата звернення: 29.09.2026).

Синенко М. А.
Національний університет «Чернігівська політехніка», кафедра кібербезпеки та математичного моделювання, м. Чернігів, Україна
Контактний автор: Марина Синенко, e-mail: mara.a.snnk@gmail.com, ORCID: 0000-0002-8961-533X

ВИКОРИСТАННЯ ВІЛЬНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПРИ ВИВЧЕННІ СТЕГАНОГРАФІЇ

Анотація. Стеганографія, як наука та мистецтво приховування секретної інформації, є важливою складовою підготовки кіберзахисників. При вивченні стеганографії для використання доступне різноманітне вільне та відкрите програмне забезпечення (FOSS), за допомогою якого можна краще зрозуміти базові стеганографічні алгоритми, отримати практичні навички приховування інформації та навчитися виявляти приховані дані (стегоаналіз). Зокрема, для приховування даних корисними можуть бути наступні інструменти: Steghide, OpenStego, Virtual Steganographic Laboratory (VSL). Для більш глибокого занурення у стеганографічні алгоритми корисно також практикувати написання власних простих скриптів, наприклад, на Python. У роботі наведено покроковий алгоритм реалізації методу найменш значущого біта, для подальшої реалізації на мові Python, що дозволить здобувачам вищої освіти детально проаналізувати алгоритм та провести певні експерименти з його параметрами.

Ключові слова: відкрите ПЗ; стеганографія; стеганоаналіз; метод найменш значущого біта.

Вступ

Стеганографія, як наука та мистецтво приховування секретної інформації, є важливою складовою підготовки кіберзахисників. Стрімкий розвиток цифрових та телекомунікаційних технологій створює передумови для розробки та удосконалення різноманітних методів захисту інформації. Для забезпечення конфіденційності секретної інформації, що передається відкритими каналами зв'язку, на сьогодні використовують два підходи. Перший підхід полягає у використанні криптографічних методів, тобто блокування несанкціонованого доступу шляхом шифрування повідомлення, інший підхід – стеганографічний – полягає у прихованні самого факту існування секретної інформації. Слід зазначити, що на практиці часто використовують комбінацію обох підходів, тобто інформація, яка потребує захисту, спочатку шифрується, а потім приховується у файлах-контейнерах (будь-яких файлах довільної природи). Загалом за допомогою стеганографічних методів успішно вирішують наступні задачі:

– захист конфіденційної інформації від несанкціонованого доступу;
– захист авторських прав на деякі види інтелектуальної власності;
– подолання систем моніторингу та керування мережевими ресурсами;
– камуфляж програмного забезпечення;
– створення прихованих каналів витоку інформації від законного власника.

Матеріали та методи

При вивченні стеганографії для використання доступне різноманітне вільне та відкрите програмне забезпечення (FOSS), за допомогою якого можна краще зрозуміти базові стеганографічні алгоритми, отримати практичні навички приховування інформації та навчитися виявляти приховані дані (стегоаналіз). Зокрема, для приховування даних корисними можуть бути наступні інструменти: Steghide, OpenStego, Virtual Steganographic Laboratory (VSL).

Steghide — консольна програма, яка дозволяє ховати дані у зображення (JPEG, BMP) та аудіофайли (WAV, AU) за допомогою алгоритму заміни. Також використовує шифрування для додаткового захисту. Для встановлення програми з терміналу Linux використовують команду:

apt-get install steghide

Для приховування даних вводять команду:

steghide embed -ef secretFile -cf coverFile -sf outputFile -z compressionLevel -e scheme;

-ef – указує шлях до файлу, який потрібно приховати.

-cf – це файл, у якому будуть приховані дані. Можна використовувати наступні формати зображень чи аудіо-файлів: BMP, JPEG, WAV, AU.

-sf – необов'язковий аргумент, який указує вихідний файл.

-z – указує рівень стиснення від 1 до 9. Якщо файл стискати не потрібно, використовують аргумент -Z.

-e – указує тип шифрування.

Після виконання команди, буде запропоновано встановити пароль, за допомогою якого можна буде витягнути приховані дані. Для витягнення даних використовують команду:

steghide extract -sf stegoFile -xf outputFile

OpenStego — програма з графічним інтерфейсом (GUI) на базі Java. Підтримує роботу із зображеннями (в основному PNG та BMP).

Основні характеристики.

– Приховування даних всередині зображення або вилучення даних із зображення;

– Додавання водяного знака;

– Підтримує алгоритми шифрування AES128, AES256;

– Використовує архітектуру на основі плагінів, де можна створювати різні плагіни для різних стеганографічних інструментів чи алгоритмів водяного знака.

Virtual Steganographic Laboratory (VSL) — графічний інструмент, створений саме для навчання. Він дозволяє будувати візуальні блоки (пайплайни) для застосування різних стеганографічних методів до зображень.

Інструменти для виявлення прихованого (Стегоаналіз), наприклад, Zsteg — потужний консольний інструмент для пошуку прихованих даних у файлах форматів PNG та BMP. Автоматично перевіряє LSB, колірні канали та інші популярні методи приховування.

Однак для більш глибокого занурення у стеганографічні алгоритми корисно практикувати написання власних простих скриптів, наприклад, на Python (використовуючи бібліотеку Pillow для роботи з пікселями зображень).

Результати

Як приклад розглянемо реалізацію методу LSB (Least Significant Bit). Покроковий опис алгоритму має наступний вигляд:

1. Вхідні дані: Завантажується оригінальне зображення-контейнер та текстове повідомлення.

2. Додавання стоп-маркера: До тексту додається умовний маркер кінця (наприклад, нуль-символ \x00 або унікальна послідовність на кшталт #####).

3. Конвертація в бінарний вид: Кожен символ тексту перетворюється на 8-бітну послідовність з нулів та одиниць (наприклад, літера 'A' → 01000001).

4. Перевірка місткості: Розраховується кількість доступних пікселів (W × H × кількість каналів). Якщо бітів у тексті більше, ніж пікселів у картинці, програма видає помилку.

5. Модифікація бітів (Суть LSB): Програма йде циклом по кожному біту повідомлення та по кожному байту кольору пікселя:

– Щоб скинути останній біт байта в 0, застосовується побітова кон'юнкція з числом 254 (11111110 у двійковій системі): byte & 254.

– Далі додається біт повідомлення за допомогою побітової диз'юнкції: (byte & 254) | message_bit.

6. Збереження: Оновлений масив пікселів збирається назад у формат зображення та зберігається без втрати якості (обов'язково у форматі PNG або BMP).



а)



б)



в)

Рисунок 1 – Оригінальне зображення у форматі BMP та результати застосування методу LSB

На рисунку 1 показані результати реалізації алгоритму LSB на мові Python: а) – оригінальне зображення (файл-контейнер); б) контейнер з прихованим текстом, де для приховання було використано два останні біти; в) контейнер з прихованим текстом, де для приховання було використано три останні біти – у цьому випадку візуально можна помітити незначні зміни.

Обговорення

Таким чином, використання мови Python для написання власного скрипта реалізації методу LSB не тільки сприяє більш глибокому розумінню алгоритму, але дозволяє здобувачам вищої освіти визначити оптимальну глибину приховування секретного повідомлення.

Висновки

Використання вільного програмного забезпечення при вивченні стеганографії має ряд важливих переваг, серед яких варто відзначити передусім відсутність витрат на ліцензії, відкритий вихідний код, який дозволяє вивчити алгоритми зсередини, високий рівень безпеки, оскільки код перевіряється спільнотою на наявність бекдорів, незалежність від розробника — софт працюватиме, навіть якщо компанія закриється. Однак слід також вказати на певні недоліки, а саме, відсутність офіційної підтримки, можливий складний інтерфейс, оскільки при розробці ПЗ акцент направлений на функціонал.

Список використаних джерел

1. Конахович Г., Прогонов Д., Пузиренко О. Комп’ютерна стеганографічна обробка й аналіз мультимедійних даних [підручник]. — К.: «Центр навчальної літератури», 2018. — 558 с.
2. Самборський, І. Сучасний стан та перспективи розвитку стеганографії у телекомунікаційних системах / Самборський Іван Іванович, Толстова Анастасія Вадимівна // Information Technology and Security. – 2022. – Vol. 10, Iss. 1 (18). – Pp. 27–38.
3. Urmila Pilania, Manoj Kumar. Unveiling the Art of Steganography: A Modern Approach. – CRC Press, 2026. – 198 p.

Содиль М. О.
Національний університет «Чернігівська політехніка», кафедра кібербезпеки та математичного моделювання, Чернігів, Україна
e-mail: nick.sodyl@stu.cn.ua, ORCID: 0009-0009-3744-0089

ВПРОВАДЖЕННЯ КОНЦЕПЦІЇ ZERO TRUST В ІНФРАСТРУКТУРУ ПІДПРИЄМСТВА

Анотація. У темі доповіді розглянуто проблеми трансформації традиційної периметрової моделі захисту в концепцію Zero Trust Architecture (ZTA). Розглянуто базові принципи, ключові компоненти та етапи розгортання Zero Trust для мінімізації ризиків розповсюдження загроз усередині корпоративної мережі.

Ключові слова: кібербезпека, Zero Trust Architecture, автентифікація, IAM, ZTNA, BYOD, мікросегментація.

Вступ

Традиційна модель мережевої безпеки, що базується на довірі до внутрішнього периметра, втратила ефективність через масовий перехід на гібридну модель роботи, використання хмарних сервісів (SaaS/IaaS) та концепцію BYOD. Отримавши первинний доступ до внутрішньої мережі через скомпрометовані облікові дані або уразливості, зловмисники отримують можливість для розгортання атак вже всередині мережі.

Концепція Zero Trust («Нікому не довіряй, завжди перевіряй») змінює цю парадигму. Довіра не надається за замовчуванням жодному користувачеві, пристрою чи сесії, незалежно від їхнього фізичного або мережевого розташування.

Матеріали та методи

Упровадження ZTA вимагає дотримання трьох основних принципів побудови.

Перший принцип – явна перевірка (Verify Explicitly): автентифікація та авторизація здійснюються на основі всіх доступних точок даних (ідентичність, пристрій, геолокація, контекст, аномалії поведінки).

Другим принципом побудови є надання найменших привілеїв (Least Privilege Access). Обмеження прав користувачів за допомогою моделей JIT (Just-In-Time) та JEA (Just-Enough-Access), а також адаптивної політики контролю доступу (ABAC).

Третій важливий принцип побудови це припущення про компрометацію (Assume Breach). Мінімізація радіуса ураження за допомогою мікросегментації, наскрізного шифрування трафіку (TLS 1.3/IPsec) та постійного аналізу дій у режимі реального часу [3].

Для реалізації ZTA на підприємстві необхідна інтеграція низки компонентів. Потрібна інтеграція Identity & Access Management (IAM). Централізоване управління з багатофакторною автентифікацією (MFA). Zero Trust Network Access (ZTNA) це заміна класичних VPN на контекстно-залежні тунелі до конкретних застосунків, а не до мережі загалом. Динамічна сегментація мережі на рівні навантажень та віртуальних машин для ізоляції критичних сегментів. Endpoint Detection and Response (EDR / XDR) це постійна оцінка стану безпеки кінцевих пристроїв перед наданням доступу. Також повинна бути налаштована політика для адміністрування яка приймає та здійснює рішення про надання чи блокування доступу на основі політик [6].

Результати

Впровадження Zero Trust є безперервним процесом, а не одноразовою закупівлею ПЗ. Впровадження архітектури Zero Trust потрібно починати з інвентаризації та визначення в першу чергу критичних ресурсів, в наступну чергу всі інші ресурси:

– Дані. Класифікація даних за рівнем конфіденційності (персональні дані, комерційна таємниця, вихідний код, фінансові звіти).

– Активи. Інвентаризація фізичних та віртуальних серверів, маршрутизаторів, IoT-пристроїв, робочих станцій, мобільних пристроїв.

– Застосунки. Складання реєстру всіх корпоративних та хмарних застосунків (ERP, CRM, внутрішні веб-сервіси).

– Сервіси. Визначення критичних мережевих сервісів (DNS, DHCP, Active Directory / LDAP, PKI).

Ідентичність стає новим «периметром» безпеки. Важливо налаштувати управління ідентичністю (Identity) та контроль кінцевих точок (Endpoints). Впровадження єдиного постачальника ідентичності (Identity Provider, IdP) із підтримкою сучасних протоколів (SAML 2.0, OpenID Connect, OAuth 2.0). Контекстна та безпарольна MFA. Перехід від SMS/OTP до контекстної багатофакторної автентифікації на базі ключів FIDO2/WebAuthn або push-сповіщень із прив'язкою до геопозиції та пристрою. Розгортання EDR/XDR (Endpoint Detection and Response) на всіх робочих станціях та серверах. Налаштування перевірки стану пристрою перед наданням доступу (наявність шифрування диска BitLocker/FileVault, актуальний стан ОС, активний антивірус) [5].

Мікросегментація та криптографічний захист мережі. Поділ мережі на ізольовані мікросегменти навколо кожного елемента Protect Surface за допомогою Software-Defined Networking (SDN) або програмних агентів на рівнях хостів. Встановлення правил контролю трафіку за принципом Default Deny (все, що явно не дозволено - заблоковано). Впровадження наскрізного шифрування для всього внутрішнього трафіку (TLS 1.3 для web/API, IPsec / mTLS для міжсерверної взаємодії) [5].

Впровадження архітектури ZTNA та контроль застосунків. Заміна традиційних мережевих VPN на контекстно-залежний доступ до конкретних застосунків. Впровадження ZTNA-шлюзів (Software-Defined Perimeter, SDP). Застосунок залишається «невидимим» для зовнішнього та внутрішнього сканування мережі доти, доки користувач та його пристрій успішно не пройдуть верифікацію на Policy Decision Point (PDP). Впровадження політик на основі атрибутів (Attribute-Based Access Control). Моделей JIT (Just-In-Time) надання підвищених привілеїв лише на обмежений час виконання завдання. Обов’язково потрібна відмова від незахищених протоколів у внутрішній мережі (HTTP, Telnet, FTP, SMBv1) [4].

Zero Trust передбачає, що мережа постійно знаходиться під загрозою, тому контроль має бути безперервним. Збір та телеметрія всіх подій авторизації, мережевого трафіку та стану пристроїв у SIEM (Security Information and Event Management). Запуск UEBA (User and Entity Behavior Analytics) для виявлення аномалій (наприклад, спроба завантаження великого обсягу даних користувачем у нетиповий час). Налаштування автоматичних сценаріїв реагування при виявленні високого рівня ризику пристрою або облікового запису. Автоматичне відкликання сесії доступу, ізоляція хоста від мережі або запит додаткового MFA [2].

Висновки

Основними бар'єрами під час переходу на Zero Trust є наявність застарілих систем, які не підтримують сучасні протоколи автентифікації, висока складність первинного налаштування політик та ризик погіршення досвіду користувачів (UX) у разі непродуманої конфігурації. Перехід на Zero Trust є необхідним кроком для забезпечення кіберстійкості сучасного підприємства. ZTA не скасовує базові засоби захисту, а систематизує їх, дозволяючи мінімізувати ризики складних атак та внутрішніх загроз.

Майбутнім напрямом у розвитку архітектури Zero Trust на підприємстві є інтеграція штучного інтелекту та машинного навчання. Розробка та імплементація алгоритмів ML для розрахунку поточного індексу ризику (Continuous Risk Score) на основі моніторингу поведінки користувачів та сутностей (UEBA). Застосування моделей глибокого навчання (Deep Learning) для аналізу зашифрованого трафіку (наприклад, TLS 1.3 без розшифрування) з метою виявлення прихованих каналів управління (C2) та нетипових сплесків передачі даних.

Список використаних джерел

1. Yeoh W., Liu M., Shore M., Jiang F. Zero trust cybersecurity: Critical success factors and a maturity assessment framework. Computers & Security. 2023. Vol. 130.
2. Pashikanti S. Implementing Zero Trust Architecture across multi-cloud environments: A security framework. International Journal of Leading Research Publication. 2023. Vol. 4, Issue 9.
3. Gambo M. L., Almulhem A. Zero Trust Architecture: A systematic literature review. 2025. URL: https://arxiv.org/abs/2503.11659
4. Min C. H. A policy-driven zero trust architecture aligned with NIST framework. Electronics. 2025. Vol. 14, No. 20
5. Weinberg, A., & Cohen, K. (2024). Zero trust implementation in the emerging technologies era: A survey. Complex Engineering Systems, 4. URL: https://www.researchgate.net/publication/384451867_Zero_trust_implementation_in_the_emerging_technologies_era_a_survey/citation/download (дата звернення: 22.08.2026)
6. Sunkara, G. (2025). Implementing Zero Trust architecture in modern enterprise networks. SAMRIDDHI: A Journal of Physical Sciences, Engineering and Technology, 17(11). URL: https://www.researchgate.net/publication/393185151_Implementing_Zero_Trust_Architecture_in_Modern_Enterprise_Networks (дата звернення: 19.08.2026)

Фесенко Т. М.
Національний університет «Полтавська політехніка імені Юрія Кондратюка», кафедра комп'ютерних та інформаційних технологій і систем, м. Полтава, Україна
Контактний автор: Тетяна Фесенко, e-mail: tanifesenko@gmail.com, ORCID: 0009-0006-1698-3795

БАГАТОКАНАЛЬНА АРХІТЕКТУРА РАДІОЧАСТОТНОГО МОНІТОРИНГУ ПОТЕНЦІЙНИХ ТКВІ З ВИКОРИСТАННЯМ БПЛА

Анотація. Розглянуто проблему підвищення безперервності радіочастотного моніторингу потенційних технічних каналів витоку інформації (ТКВІ) під час використання безпілотних літальних апаратів (БПЛА) як мобільних приймальних платформ. Встановлено, що послідовне опитування просторових приймальних каналів збільшує тривалість циклу моніторингу та формує часові інтервали неповного просторово-частотного контролю. Запропоновано багатоканальну архітектуру на базі SoC FPGA, що поєднує просторовий апаратний паралелізм, незалежне конвеєрне цифрове оброблення та детерміноване міжканальне часове узгодження із використанням JESD204B і SYSREF. Експериментальне оцінювання триканальної структури показало збільшення ймовірності виявлення потенційного радіосигналу з 0,45 до 0,97 та скорочення частки часу неповного контролю з 35 % до менш ніж 3 %. Максимальна зафіксована швидкість потокового оброблення становила 980 МБ/с без втрат пакетів (кадрів).

Ключові слова: технічні канали витоку інформації, радіочастотний моніторинг, безпілотний літальний апарат, багатоканальна архітектура, апаратний паралелізм, цифрове потокове оброблення, детермінована синхронізація, SoC FPGA.

Вступ

Сучасні БПЛА, оснащені широкосмуговими радіоприймальними засобами, можуть використовуватися як мобільні платформи радіочастотного моніторингу побічних електромагнітних випромінювань технічних засобів. Аналіз таких випромінювань дає змогу виявляти радіосигнали, характеристики яких потенційно можуть бути пов'язані з проявами ТКВІ.

Специфіка радіочастотного моніторингу з використанням БПЛА визначається динамічною зміною просторового положення приймальної платформи, параметрів сигналів та умов їх поширення. За послідовного опитування декількох приймальних каналів збільшення їх кількості призводить до зростання тривалості циклу обслуговування. Унаслідок цього виникають часові інтервали неповного просторово-частотного контролю, протягом яких радіосигнал, потенційно пов'язаний із ТКВІ, може залишатися поза активною областю спостереження. Така особливість безпосередньо впливає на своєчасність виявлення сигналів.

Отже, актуальним науково-технічним завданням є підвищення безперервності багатоканального радіочастотного моніторингу шляхом одночасного апаратного оброблення сигналів декількох просторових каналів, скорочення тривалості критичного циклу їх обслуговування та мінімізації часових інтервалів неповного контролю.

Метою роботи є підвищення ефективності виявлення радіосигналів, потенційно пов'язаних із ТКВІ, шляхом удосконалення архітектури засобу радіочастотного моніторингу на основі просторового апаратного паралелізму приймальних каналів, конвеєрного цифрового оброблення та детермінованого міжканального часового узгодження.

Метод дослідження

Для порівняльного оцінювання послідовної та паралельної архітектур використано показники ймовірності виявлення сигналу, безперервності просторово-частотного контролю та пропускної здатності цифрового тракту.

Для системи з (N) приймальними каналами, за припущення статистичної незалежності подій виявлення сумарна ймовірність виявлення сигналу хоча б одним каналом визначається як

$$P_{det}^{(N)} = 1 - \prod_{i=1}^{N} (1 - P_{det,\,i}), \quad (1)$$

де P_{det} – ймовірність виявлення сигналу (i)-м приймальним каналом за заданих умов спостереження. Для триканальної системи вираз (1) набуває вигляду:

$$P_{det}^{(3)} = 1-(1-P_{det,1})(1-P_{det,2})(1-P_{det,3}) \quad (2)$$

Якщо приймальні канали мають однакову ймовірність виявлення P_{det}, отримаємо:

$$P_{det}^{(3)} = 1-(1-P_{det})^3 \quad (3)$$

Отримана модель характеризує теоретичний ефект збільшення кількості одночасно функціонуючих приймальних каналів та використовується як модель першого наближення для порівняльного аналізу архітектур. Водночас у реальному багатоканальному приймальному тракті статистична незалежність може порушуватися внаслідок спільних джерел шуму, взаємного впливу каналів, особливостей тактової системи та інших апаратних факторів.

Для кількісного оцінювання безперервності моніторингу введено відносну частку часу неповного контролю

$$\eta_{nc} = \frac{T_{nc}}{T_{obs}}, \quad (4)$$

де T_{nc} – сумарна тривалість інтервалів, протягом яких окремі просторово-частотні сегменти не перебувають під активним контролем, T_{obs} – загальна тривалість спостереження. Зменшення (η_{nc}) характеризує підвищення безперервності моніторингу та скорочення часової області потенційного пропуску сигналів.

Запропонована архітектура

Для зменшення (η_{nc}) запропоновано багатоканальну апаратну архітектуру на базі SoC FPGA, у якій послідовне часове мультиплексування критичної ділянки приймального тракту замінено просторовим апаратним паралелізмом [1]. Експериментальний варіант системи містить три незалежні приймальні канали. Кожен канал формує окремий цифровий потік і обробляється власним конвеєром цифрового оброблення. Така організація забезпечує одночасне приймання, синхронізацію та первинне цифрове оброблення сигналів декількох просторових каналів без їх послідовного перемикання.

Функціонально (i)-й приймальний канал може бути представлений послідовністю $x_i[n] \to ADC_i \to Sync_i \to Proc_i \to Det_i$, де $x_i[n]$ – аналоговий сигнал (i)-го просторового каналу, ADC_i – аналого-цифрове перетворення, $Sync_i$ – часова синхронізація, $Proc_i$ – первинне цифрове оброблення, Det_i – процедура виявлення сигналу.

Для паралельної триканальної структури відповідні операції виконуються одночасно, що усуває необхідність послідовного перемикання приймальних трактів на критичній ділянці оброблення. За послідовної архітектури тривалість циклу визначається сумою часу оброблення окремих каналів та витрат на їх перемикання. У паралельній архітектурі тривалість критичного етапу за умови достатньої апаратної пропускної здатності визначається переважно найбільш тривалим каналом: $T_{parallel} \approx \max(T_1, T_2, \ldots, T_N)$.

Таким чином, збільшення кількості одночасно контрольованих просторових напрямків не потребує пропорційного збільшення тривалості циклу послідовного опитування.

Детерміноване часове узгодження

Апаратний паралелізм приймальних каналів зумовлює необхідність їх часової узгодженості, оскільки міжканальне порівняння параметрів сигналів потребує визначеного співвідношення часових позицій відповідних відліків. Для забезпечення детермінованого часового узгодження приймальних трактів використано інтерфейс JESD204B та сигнал SYSREF [2].

Часову узгодженість двох каналів можна представити умовою

$$|t_i - t_j| \le \Delta t_{max}, \quad i, j \in 1,2,3, \quad (5)$$

де t_i та t_j – часові позиції відповідних відліків або кадрів даних, а Δt_{max} – допустиме міжканальне розузгодження, визначене вимогами алгоритму подальшого просторового аналізу.

Детерміноване часове узгодження є необхідною передумовою коректного міжканального порівняння параметрів сигналів. Водночас воно не визначає повністю точність просторового оцінювання, яка також залежить від геометрії антенної системи, амплітудно-фазової нерівномірності приймальних трактів, калібрування та алгоритму визначення напрямку приходу сигналу.

Пропускна здатність цифрового тракту

Поряд із часовою узгодженістю критичним обмеженням багатоканальної архітектури є сумарна швидкість надходження цифрових даних. Із збільшенням кількості паралельних каналів вона зростає відповідно до індивідуальних швидкостей сформованих потоків:

$$B_{sum} = \sum_{i=1}^{N} B_i \qquad (6)$$

де B_i – швидкість цифрового потоку (i)-го приймального каналу.

Для забезпечення безперервного потокового оброблення ефективна пропускна здатність апаратного тракту B_{eff} повинна задовольняти умову $B_{eff} \geq B_{sum}$

Невиконання цієї умови може спричинити накопичення даних у буферах та їх втрату, що безпосередньо впливає на своєчасність виявлення сигналів. Тому для підвищення надійності доцільно реалізувати апаратний контроль стану синхронізації, службових маркерів, переповнення буферів та цілісності потокових даних. Такий контроль дає змогу виявляти порушення функціонування потокового тракту до передавання результатів до програмної підсистеми моніторингу.

Експериментальні результати

Експериментальне оцінювання модернізованого комплексу «Чуйка» виконано за показниками ймовірності виявлення, відносної частки часу неповного контролю, пропускної здатності апаратного тракту та часової узгодженості багатоканального оброблення.

У досліджуваному експериментальному сценарії перехід від послідовної до запропонованої триканальної архітектури забезпечив збільшення ймовірності виявлення потенційного радіосигналу з 0,45 до 0,97. Абсолютне збільшення становило 0,52, а відносне збільшення визначається як $\frac{0,97-0,45}{0,45}\cdot 100\% \approx 115,6\%$. Відповідно, ймовірність пропуску сигналу зменшилася з 0,55 до 0,03.

Частку часу неповного контролю зменшено з 35 % до менш ніж 3 %, що відповідає абсолютному скороченню більш ніж на 32 процентні пункти та відносному скороченню більш ніж на 91,4 %. Оскільки фактичне значення після модернізації є меншим за 3 %, наведена оцінка відносного скорочення має консервативний характер.

За досліджуваних умов зафіксовано швидкість оброблення цифрового потоку до 980 МБ/с. Під час проведеного тестування втрат пакетів (кадрів) не зафіксовано, що свідчить про відповідність реалізованого потокового тракту вимогам досліджуваного сценарію за пропускною здатністю.

Отримані результати свідчать, що ефективність запропонованої архітектури визначається сукупною дією трьох взаємопов’язаних факторів: одночасного функціонування приймальних каналів, синхронного потокового цифрового оброблення та забезпечення необхідної пропускної здатності цифрового тракту.

Наукова новизна та практичне значення

Наукова новизна роботи полягає в удосконаленні архітектурного методу радіочастотного моніторингу сигналів, потенційно пов’язаних із ТКВІ, шляхом поєднання просторового апаратного паралелізму приймальних каналів, незалежного потокового цифрового оброблення та детермінованого міжканального часового узгодження.

На відміну від послідовної організації приймання, запропонований підхід забезпечує одночасний контроль декількох просторових напрямків та зменшує часові інтервали неповного просторово-частотного спостереження. Для кількісного порівняння архітектур використано показники ймовірності виявлення, частки часу неповного контролю та пропускної здатності цифрового тракту.

Практичне значення одержаних результатів полягає у можливості застосування запропонованої архітектури під час модернізації мобільних засобів радіочастотного моніторингу та систем технічного захисту інформації, що потребують одночасного багатоканального контролю та безперервного потокового оброблення даних.

Висновки

Запропоновано триканальну SoC FPGA-архітектуру радіочастотного моніторингу потенційних ТКВІ, основу якої становлять просторовий апаратний паралелізм приймальних каналів, незалежне конвеєрне цифрове оброблення та детерміноване міжканальне часове узгодження. Математична модель показує принципову перевагу одночасного функціонування декількох каналів за умови незалежності подій виявлення, а введений показник частки часу неповного контролю дає змогу кількісно оцінювати безперервність моніторингу.

Експериментальні результати для досліджуваного сценарію характеризуються збільшенням ймовірності виявлення потенційного радіосигналу з 0,45 до 0,97, скороченням частки часу неповного контролю з 35 % до менш ніж 3 % та забезпеченням швидкості потокового оброблення до 980 МБ/с без зафіксованих втрат пакетів (кадрів).

Подальший розвиток запропонованого підходу доцільно пов’язати з масштабуванням кількості приймальних каналів, дослідженням взаємного впливу каналів та корельованих шумів, уточненням вимог до міжканального часового узгодження, а також експериментальною оцінкою впливу геометрії антенної системи, калібрування та алгоритмів просторового аналізу на точність виявлення потенційних ТКВІ.

Список використаних джерел

1. Bae, Y., Lee, H., Park, H., Jang, W.-h., & Jang, B.-J. (2026). Edge CA-CFAR Data Reduction for Bandwidth-Efficient Real-Time Wideband Spectrum Sensing on Low-Cost SDRs. Sensors, 26(14), 4468. https://doi.org/10.3390/s26144468.

2. Intel. F-Tile JESD204B IP Deterministic Latency Implementation Guidelines : Intel FPGA IP User Guide. 2024. URL: https://docs.altera.com/r/docs/723907/current (дата звернення: 16.09.2026).

Чопорова О. В., Чопоров С. В.
1 – Київський національний економічний університет імені Вадима Гетьмана, кафедра комп'ютерних наук, Київ, Україна
2 – Європейський університет, Київ, Україна
Контактний автор: Оксана Чопорова, e-mail: choporova.oksana@kneu.edu.ua, ORCID: 0000-0003-3167-7869

ВИКОРИСТАННЯ 3D-МОДЕЛЮВАННЯ ТА АЛГОРИТМІВ НЕЙРОННИХ МЕРЕЖ В НАВЧАЛЬНИХ ПРОЄКТАХ ІНЖЕНЕРНО-ТЕХНІЧНОГО СПРЯМУВАННЯ / USE OF 3D MODELING AND NEURAL NETWORK ALGORITHMS IN EDUCATIONAL PROJECTS OF ENGINEERING AND TECHNICAL DIRECTION

Анотація. Робота присвячена огляду класичних методів розрахунку напружено-деформованого стану конструкцій під час вивчення навчальної дисципліни «Системи автоматизованого інжинірингу». Запропоновано поєднання 3D-моделювання з вже розробленими штучними нейронними мережами для прогнозування деформацій пластин різної форми, включаючи пластини з отворами, циліндричні та конічні оболонки. Результатом є створення навчального середовища, де студенти можуть проєктувати й оптимізувати елементи авіаційної і машинобудівної техніки, замінивши тривалі розрахунки на швидкі нейромережеві обчислення. Значущість такого підходу полягає у підготовці ІТ-фахівців, суттєвій економії обчислювальних ресурсів та формуванні навичок впровадження ШІ у високотехнологічне виробництво.

Ключові слова: штучні нейронні мережі, параметричне 3D-моделювання, системи автоматизованого інжинірингу, прогнозування деформацій, просторові оболонки.

Вступ

Сучасна підготовка ІТ-фахівців вимагає інтеграції передових обчислювальних методів у процес вивчення прикладних інженерних дисциплін. В межах навчальної дисципліни «Системи автоматизованого інжинірингу» на кафедрі комп'ютерних наук Київського національного економічного університету ім. Гетьмана студенти комплексно опановують різноманітні програмні середовища (зокрема, системи параметричного 3D-моделювання) для автоматизованого проєктування об'єктів складної просторової форми. Особлива увага під час навчання приділяється створенню геометричних моделей, що є базовими елементами реальних інженерних конструкцій і мають широке застосування в авіаційній та машинобудівній галузях.

Традиційно оцінка надійності та прогнозування поведінки таких просторових елементів під навантаженням здійснюється за допомогою класичних ресурсомістких систем комп'ютерного моделювання (зокрема, на основі методу скінченних елементів). Однак у навчальному процесі такий підхід часто знижує інтерактивність через високі апаратні вимоги та тривалий час розрахунків. З огляду на це, актуальним стає впровадження та аналіз моделей різного типу за допомогою попередньо навчених штучних нейронних мереж. Такий підхід дозволяє миттєво прогнозувати деформації конструкцій, змінюючи ключові параметри, перетворюючи складний розрахунок на швидкий та наочний інструмент сучасного інжинірингу без використання важких класичних симуляцій.

Матеріали та методи

Метою роботи є інтеграція результатів наукових досліджень із нейромережевого прогнозування напружено-деформованого стану просторових конструкцій у навчальний курс із систем автоматизованого інжинірингу. Запропонований підхід має на меті продемонструвати студентам інноваційні методи проєктування складної техніки (зокрема, ракет, літаків та іншого машинобудівного обладнання, а також певних їх деталей), де використання штучного інтелекту дозволяє скоротити часові та обчислювальні ресурси на етапах розробки, розрахунків та попередньої апробації виробів.

Навчальний процес розбивається на взаємопов'язані етапи, що поєднують комп'ютерний дизайн та машинне навчання. На початковому етапі студенти, використовуючи системи 3D-моделювання, створюють цифрові моделі елементів інженерних конструкцій. Це можуть

бути як маленькі деталі у вигляді прямокутних або квадратних пластин з отворами і без, так і об’ємні деталі циліндричної та конічної форми, які використовують для моделювання спрощені моделі секцій фюзеляжу літака, корпусів ракет або елементів турбін. Студенти вчаться параметризувати об'єкти (задавати товщину, геометричні розміри, властивості матеріалу), розуміючи, як ці характеристики впливають на фізичне призначення деталі.

Здобувачам вищої освіти пропонується застосувати готові розроблені архітектури штучних нейронних мереж. Геометричні та фізичні параметри створеної 3D-моделі передаються на вхід нейромережі, яка миттєво генерує високоточний прогноз деформацій та напружень конструкції під заданими навантаженнями.

Оскільки нейромережа видає результат за мілісекунди, студенти отримують можливість у реальному часі проводити оптимізацію конструкції. Вони можуть багаторазово змінювати параметри 3D-моделі у САПР, досягаючи, наприклад, мінімальної ваги оболонки при збереженні її міцності та наочно бачити отримані результати.

Такий підхід ілюструє, як поєднання автоматизованого проєктування та штучного інтелекту трансформує інженерію. Студенти на практиці бачать, як готові бази розрахунків (використані для навчання нейромережі) здатні замінити важкі обчислення на етапі ескізного проєктування нової техніки, заощаджуючи ресурси підприємств до початку етапу фізичного виготовлення та випробувань.

Результати

Після вибору програми для автоматизованого проєктування об’єктів відбувається створення геометричних фігур. Спочатку можуть бути обрані довільні параметри, а потім накладаються певні обмеження по ширині, товщині, радіусам та інші. Якщо, наприклад розглядається прямокутна пластина з круглим отвором, то необхідно враховувати, щоб діаметр отвору не виходив за межі пластини. Моделі пластин зображені на рисунках 1, 2.

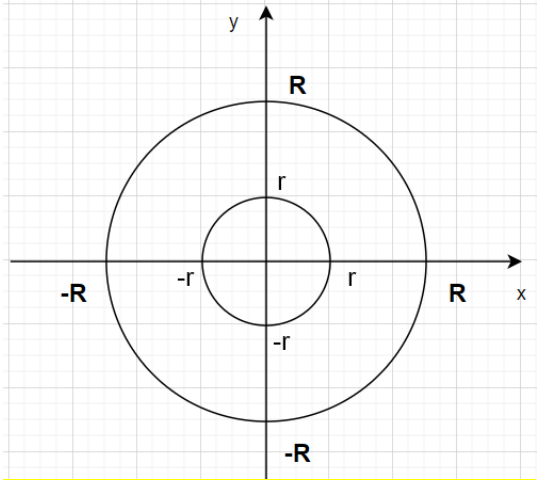


Рисунок 1 – Модель круглої пластини з отвором

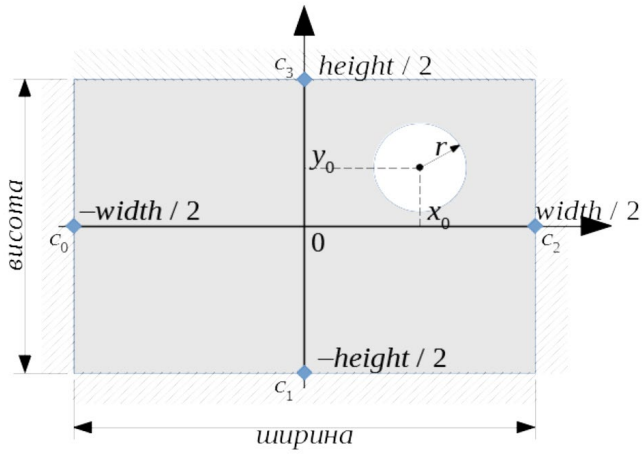


Рисунок 2 – Модель прямокутної пластини з отвором

Для визначення напружено-деформованого стану круглої пластини необхідні значення таких параметрів як: a – радіус пластини, h – товщина пластини, E – модуль пружності матеріалу, ν – коефіцієнт Пуассона, q – поперечна сила. Параметри пластинки генеруються випадковим чином з певними обмеженнями: a∈[0.1;4 (м)]; h∈[0.01;0.05 (м)]; E∈[50000;300000] (MPa); ν∈[0;0.45]; q∈[0.01;0.1] (MPa).

Результати розрахунків максимального прогину зображені на рисунках 3-4.

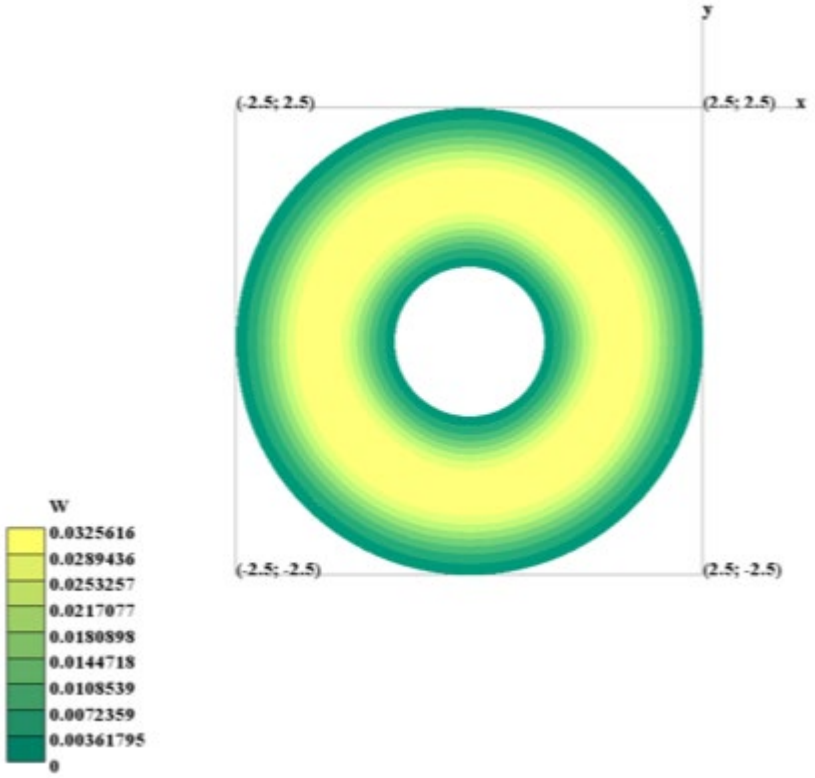


Рисунок 3 – Зміна значень прогину круглої пластини

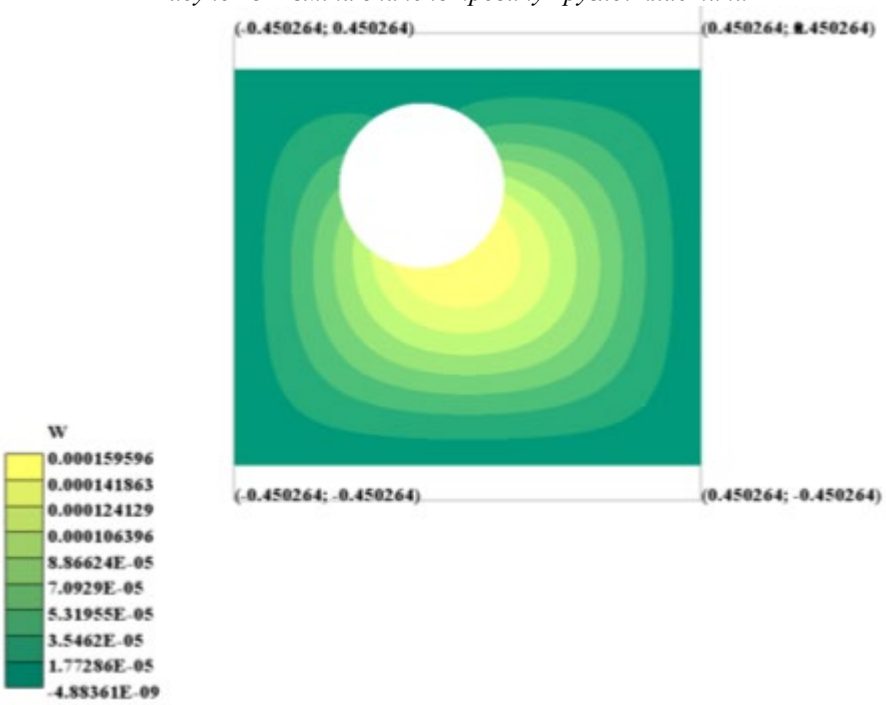


Рисунок 4 – Зміна значень прогину прямокутної пластини

Обговорення

Традиційне вивчення напружено-деформованого стану базується на використанні важких комерційних CAE-систем, що реалізують метод скінченних елементів (МСЕ). Цей підхід вимагає знання принципів побудови сіток, ідеалізації моделей та наявності потужних

робочих станцій. Натомість запропонований нейромережевий підхід дозволяє абстрагуватися від складної механіки з допомогою легких програмних рішень (наприклад, Python-скриптів, що з'єднують 3D-модель з навченою мережею). Це робить процес дружнім для ІТ-фахівців і дозволяє проводити складні обчислення на базовому комп'ютерному обладнанні.

Висновки

Отже, впровадження освітньої методики, що інтегрує системи параметричного 3D-моделювання та штучні нейронні мережі модернізує інженерну ІТ-освіту. Використання ШІ для прогнозування деформацій конструкцій (пластин, циліндричних та конічних оболонок) дозволяє подолати певні недоліки класичних CAE-систем. Це створює інтерактивне навчальне середовище, де студенти миттєво бачать результати змін у проєкті, що стимулює пошук оптимальних інженерних рішень у реальному часі. Здобувачі освіти отримують практичний досвід роботи з моделями та базовими елементами цифрових двійників, що є необхідним для сучасного авіаційного та машинобудівного секторів.

Список використаних джерел

1. Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, 378, 686-707.
2. Li, Y., Zhang, X., & Wang, H. (2025). Real-time AI-CAE surrogate modeling for parametric design of aerospace structures. Engineering Applications of Artificial Intelligence, 133, 108-119.
3. Magana, A. J., & Silva, H. (2024). Digital Twins and Surrogate Modeling in STEM Education: Enhancing Interactive CAD/CAE Learning Environments. International Journal of Engineering Education, 40(2), 312-325.
4. Lisnyak, A. O., & Choporova, O. V. (2020). Using machine learning to predict the stress-strain state of a rectangular plate with a circular cut-out. CEUR Workshop Proceedings, 2791, 1–6. https://www.scopus.com/record/display.uri?eid=2-s2.0-85098720535

Чуб М. В.
КПІ ім. Ігоря Сікорського, кафедра інформаційної безпеки, м. Київ, Україна
Контактний автор: Микита Чуб, e-mail: chub.m.v.-fb52mp@edu.kpi.ua, ORCID: 0009-0000-1914-7019

ВИЯВЛЕННЯ ТА ПРОТИДІЯ АДАПТИВНИМ ВЕБ-АТАКАМ ЗА ДОПОМОГОЮ МОДЕЛЕЙ ГЛИБОКОГО НАВЧАННЯ

Анотація. У роботі запропоновано архітектуру інтегрованого аналітичного ядра для виявлення аномальної поведінки в HTTP-трафіку на основі LSTM Autoencoder. Модель навчається на профілі легітимного трафіку та визначає послідовності запитів із високим рівнем аномальності. Для виявлених аномалій формується набір ознак, який передається модулю WAF-адаптера для створення правил блокування через REST API. Експериментальну перевірку проведено на реальних журналах HTTP-трафіку. Отримані результати демонструють можливість автоматизованого виявлення нетипових поведінкових патернів та подальшого формування захисних правил для систем класу WAF.

Ключові слова: машинне навчання; LSTM Autoencoder; адаптивні веб-атаки; поведінковий аналіз; кібербезпека; WAF; HTTP-трафік.

Вступ

У зв’язку зі стрімким розвитком технологій штучного інтелекту (ШІ) та великих мовних моделей кіберзлочинці дедалі більше використовують їх для автоматизації атак на вебзастосунки та мережеву інфраструктуру. Традиційні засоби захисту класу Web Application Firewall (WAF) та сигнатурні аналізатори виявляються неефективними проти адаптивних атак, оскільки атакуючі агенти під управлінням ШІ здатні динамічно модифікувати структуру HTTP-запитів, обходити сигнатурні правила та міняти поведінкові патерни в режимі реального часу. Метою цієї роботи є розробка архітектури інтегрованого аналітичного ядра для поведінкового аналізу HTTP-трафіку, що забезпечує автоматизоване виявлення інцидентів кібербезпеки та реагування на них шляхом автоматичного створення політик блокування для WAF.

Матеріали та методи

Запропоноване рішення базується на трирівневій модульній архітектурі, інтегрованій у наявну інфраструктуру моніторингу та захисту (рис. 1).

Модуль перехоплення та первинної обробки використовує агреговані логи HTTP-трафіку, що надходять від SIEM, після чого здійснює первинну обробку запитів та виділення ключових ознак (заголовки, часові інтервали між запитами, ентропія корисного навантаження, структура URI, коди відповідей, IP).

Модуль аналітичного ядра здійснює поведінковий аналіз за допомогою нейронної мережі на базі LSTM Autoencoder. Він виявляє відхилення від профілю легітимної поведінки HTTP-сесій, зокрема поведінкові патерни, які характерні для адаптивних автоматизованих атак керованих ШІ.

Модуль WAF-адаптерів трансформує результати виявлення аномалій аналітичним ядром у політики блокування. Через REST API він передає згенеровані правила безпосередньо до WAF для оперативного застосування та блокування шкідливого трафіку на шляху до вебзастосунку.

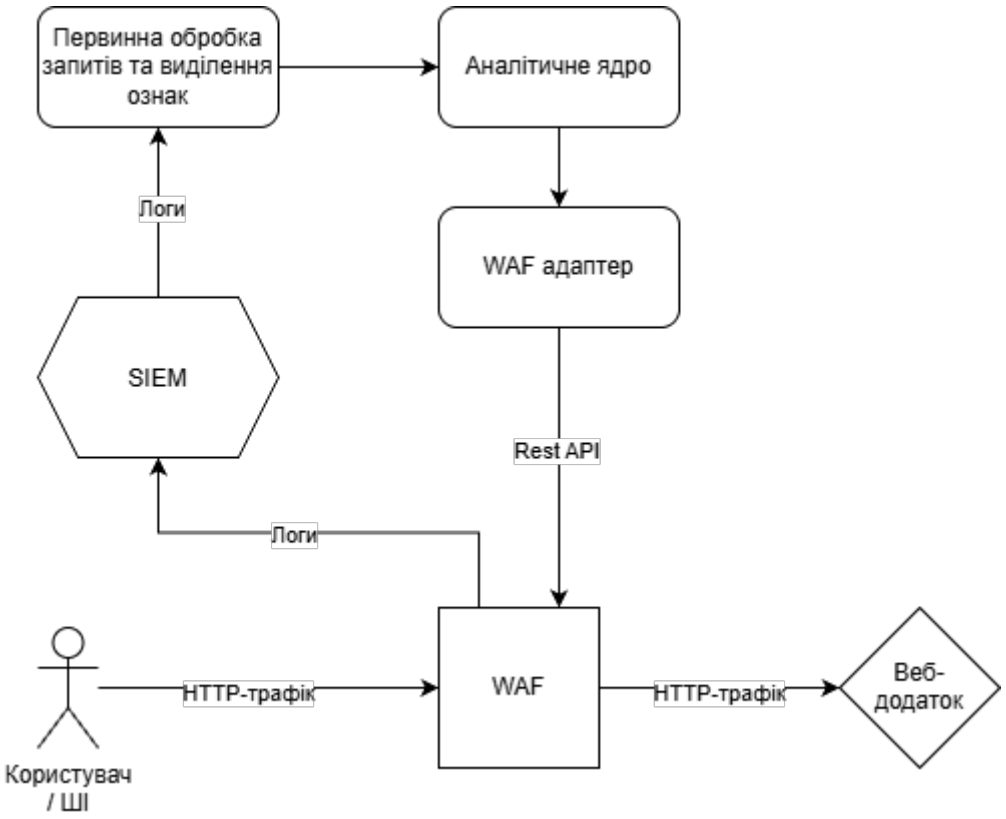


Рис. 1 – Концептуальна схема потоків даних та взаємодії компонентів аналітичної системи з WAF

Результати

Апробацію розробленого аналітичного ядра проведено на базі реальних журналів HTTP-трафіку. Для виявлення поведінкових патернів використано архітектуру нейронної мережі LSTM Autoencoder, реалізовану за допомогою фреймворку PyTorch із застосуванням апаратного прискорення CUDA.

Процес формування профілю легітимного трафіку здійснювався на навчальній вибірці обсягом більше ніж 2 мільйони запитів. В якості вхідних векторів ознак аналізувалися таймінги запитів, глибина сесій та метрики поліморфізму. Протягом 30 епох навчання функція втрат стабільно знизилася з 0,0052 до 0,0019, що свідчить про успішну екстракцію нормальних поведінкових ознак. Межу визначення аномалій встановлено на 1% найбільш відхилених від норми запитів, де поріг аномальності становив 0.0187.

Етап виявлення аномальних поведінкових патернів, які характерні для ШІ-керованих атак, проводився на тестовому наборі зі 100 тисяч нерозмічених запитів. У результаті роботи аналітичного ядра було автоматично ізольовано 41 IP-адресу з аномальною поведінкою. Два кластери запитів продемонстрували критичні відхилення, де показники аномальності становили 0,250 та 0,182, що більш ніж у 9 разів перевищує допустимий поріг аномальності (табл. 1).

Таблиця 1 – Результати виявлення аномалій моделлю LSTM Autoencoder

IP-адреса джерела	**Показник аномальності**	**Перевищення порогу аномальності**
58.20.x.x	0,250958	у 13,4 рази
47.250.x.x	0,182378	у 9,7 разів
193.193.x.x	0,108021	у 5,7 разів
212.90.x.x	0,042057	у 2,2 рази
93.183.x.x	0,040958	у 2,1 рази

Обговорення

Ручна верифікація показала, що частина виявлених аномалій відповідає ознакам автоматизованої шкідливої активності. Згенерований перелік ознак було успішно передано до модуля адаптера для подальшого застосування динамічних правил на рівні WAF. Це

підтверджує здатність системи оперативно реагувати на загрози, які обходять статичні правила безпеки.

Висновки

Розроблена архітектура інтеграції аналітичного ШІ-ядра з WAF та SIEM довела свою практичну ефективність у виявленні та протидії складним адаптивним загрозам. Використання моделі LSTM Autoencoder дозволило автоматизувати процес виявлення аномалій у поведінці HTTP-трафіку з високою точністю, виділяючи атаки, що не визначаються класичними сигнатурними методами.

Список використаних джерел

1. Masoodi F. S., Bamhdi A. Deep Learning for Intrusion Detection: Techniques and Applications. Wiley, 2025, 304 p.
2. Michelucci U. Applied Deep Learning with TensorFlow 2: Learn to Implement Advanced Deep Learning Techniques with Python. Apress, 2022, 380 p.
3. OWASP Top 10:2025. URL: https://top10.owasp.org/2025/ (дата звернення: 13.08.2026).
4. MITRE ATT&CK Framework. Enterprise Techniques: Application Layer Protocol. URL: https://attack.mitre.org/techniques/T1071/ (дата звернення: 13.08.2026).
5. Brownlee J. A Gentle Introduction to LSTM Autoencoders. Machine Learning Mastery, 2020. URL: https://machinelearningmastery.com/lstm-autoencoders/ (дата звернення: 15.08.2026).

Шевченко Ю. О.
Харківський національний університет ім. В.Н. Каразіна, Харків, Україна
Контактний автор: Юлія Шевченко, e-mail: yuliia.shevchenko@student.karazin.ua, ORCID: 0009-0002-7442-1775

ПОРІВНЯЛЬНЕ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ POSTGRESQL ТА INFLUXDB ДЛЯ ЗБЕРІГАННЯ ДАНИХ ФІЗИЧНОЇ АКТИВНОСТІ / COMPARATIVE STUDY OF POSTGRESQL AND INFLUXDB PERFORMANCE FOR STORING PHYSICAL ACTIVITY DATA

Анотація. У роботі виконано порівняльне дослідження продуктивності реляційної СУБД PostgreSQL і спеціалізованої часово-рядової бази даних InfluxDB для зберігання й аналізу даних фізичної активності. На основі публічного датасету PAMAP2 побудовано пайплайн. Він починає свою роботу з очищення даних, після чого всі записи завантажено в обидві СУБД і виконано п'ять семантично зіставних запитів, кожен з яких запускався по 10 разів. Встановлено, що PostgreSQL суттєво швидший на індексованих запитах вибірки за діапазоном часу та на фільтрації за значеннями полів. В свою чергу InfluxDB виграє на типових часово-рядових операціях – зниження розмірності вибірки та отриманні останнього значення для кожного суб'єкта. PostgreSQL також суттєво швидший при масовому завантаженні даних. Результати дають практичні орієнтири для вибору СУБД відповідно до профілю навантаження систем моніторингу фізичної активності.

Ключові слова: PostgreSQL, InfluxDB, часові ряди, IMU-дані, PAMAP2, продуктивність баз даних, фізична активність, порівняльний аналіз.

Вступ

Розвиток носимих пристроїв і систем моніторингу фізичної активності супроводжується стрімким зростанням обсягів даних, що надходять із IMU-сенсорів (акселерометр, гіроскоп, магнітометр) та фізіологічних датчиків з високою частотою дискретизації [1]. Такі дані за своєю природою є часовими рядами, прив'язаними до конкретного суб'єкта та моменту часу, що ставить перед розробниками питання вибору сховища даних: використовувати універсальну реляційну СУБД, з якою команда вже добре знайома, чи спеціалізовану часово-рядову СУБД (Time Series Database), що орієнтована саме на такий тип навантаження.

У літературі неодноразово зазначається, що спеціалізовані TSDB за рахунок стовпцевого зберігання, стиснення та вбудованих функцій агрегації здатні суттєво випереджати класичні реляційні СУБД на типових часово-рядових операціях (даунсемплінг, останнє значення, агрегація за тегами) [2]. А реляційні СУБД мають перевагу у виконанні складних довільних запитів, при коректному індексуванні [3]. Проте характеристики такої переваги залежать від конкретного набору даних, схеми та профілю запитів, тому емпіричне дослідження на реальних даних фізичної активності є актуальним [4].

Метою роботи є кількісне порівняння продуктивності PostgreSQL та InfluxDB на реальному датасеті фізичної активності PAMAP2 за показниками часу масового завантаження даних та часу виконання типових аналітичних запитів.

Матеріали та методи

Дані. Використано публічний датасет PAMAP2 (Physical Activity Monitoring), що містить записи 9 суб'єктів з трьох IMU-сенсорів та пульсометра з частотою дискретизації IMU 100 Гц [5]. Вихідні файли містять по 54 стовпці без заголовків.

Модуль підготовки даних, реалізований на Python з використанням pandas/NumPy. Він виконує послідовно зчитування .dat-файлів і додавання ідентифікатора суб'єкта, видалення невалідних колонок, видалення рядків перехідного стану, сортування за суб'єктом і часом, лінійну інтерполяцію пропусків окремо для кожного суб'єкта з обмеженням довжини пропуску 1 с, видалення рядків із пропусками [6], що залишилися після інтерполяції, видалення викидів акселерометра, формування умовної часової мітки, додавання текстової назви активності, фінальну перевірку та збереження результату у форматі Parquet.

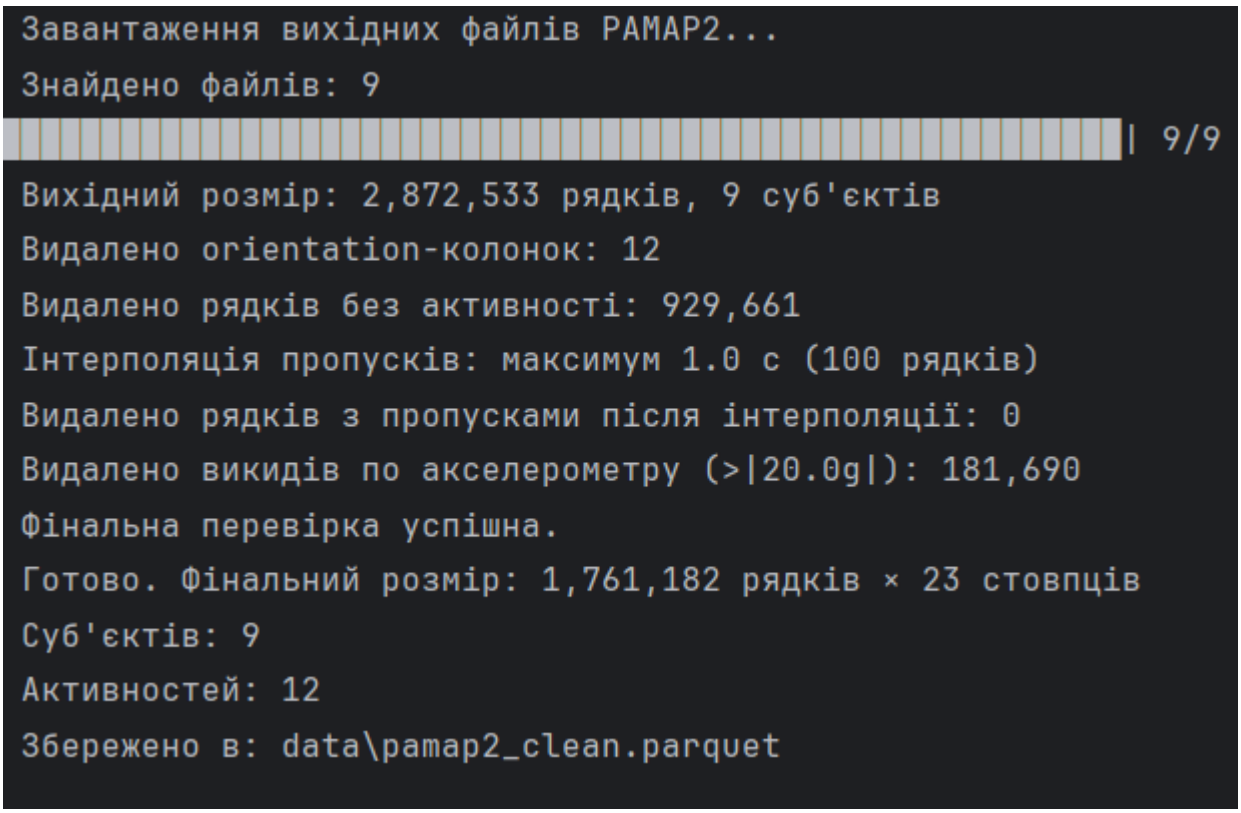


Рисунок 1 – Результати препроцесингу даних

Середовище та СУБД. PostgreSQL: дані завантажено через скрипт у таблицю activity_data, після чого створено індекси за time та activity_id. InfluxDB 2.x: дані завантажено пакетно у вимірювання activity, де subject_id, activity_id, activity_name – теги, а heart_rate та IMU-ознаки – поля.

Бенчмаркінг. Сформовано 5 семантично зіставних пар запитів SQL/Flux: Q1 – вибірка сирих показань акселерометра за 5-хвилинний інтервал для одного суб'єкта; Q2 – AVG/MIN/MAX пульсу за видами активності; Q3 – даунсемплінг акселерометра до 1-хвилинних інтервалів; Q4 – останнє значення пульсу для кожного суб'єкта; Q5 – фільтрація пульсу за діапазоном значень і конкретною активністю. Кожен запит виконується з фіксуванням часу виконання і кількості повернених рядків.

Результати

Час масового завантаження. PostgreSQL завантажив всі рядки за 45,59 с, створення трьох індексів зайняло ще 2,98 с (разом ≈ 48,6 с). InfluxDB завантажив той самий обсяг точок через пакетний Write API за 166,78 с – приблизно у 3,6 раза довше.

Контроль коректності. Для кожного з запитів кількість повернених рядків повністю збігалася між СУБД. PostgreSQL виявився швидшим на запитах Q1 та Q5, InfluxDB – на запитах Q2, Q3 і Q4, при цьому перевага InfluxDB на Q4 є найбільш вираженою серед усіх вимірювань (у 52 рази).

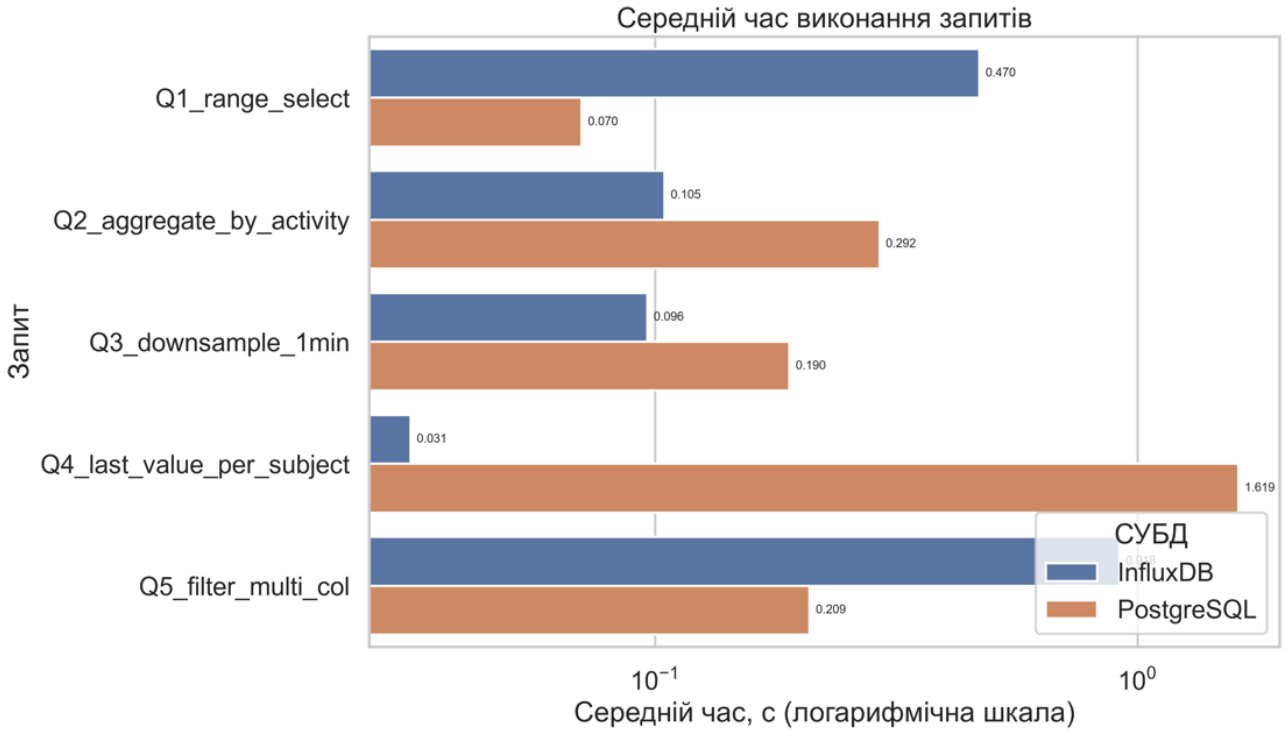


Рисунок 2 – Графік порівняння середнього часу виконання запитів у PostgreSQL та InfluxDB

Обговорення

Отримані результати узгоджуються з архітектурними особливостями обох систем. PostgreSQL демонструє перевагу там, де можна ефективно скористатися індексами: точковий діапазон часу для конкретного суб'єкта та фільтрація за конкретними значеннями поля разом з тегом активності. InfluxDB виграє на операціях, для яких він спроєктований: агрегація за тегом-групою, вбудований даунсемплінг та отримання останньої точки на серію – рушій TSM зберігає дані впорядковано за часом у межах серії, тому операція last() виконується практично за сталий час, тоді як еквівалентний SELECT DISTINCT ON у PostgreSQL без спеціального індексу під цей патерн змушений виконувати повне сортування/сканування, що і пояснює 52-кратну різницю.

Суттєва різниця в часі масового завантаження (45,6 с проти 166,8 с) частково пояснюється не лише архітектурою СУБД, а й способом завантаження: PostgreSQL використовував пакетний серверний COPY, тоді як для InfluxDB точки формувалися по одному рядку засобами клієнтської бібліотеки Python перед пакетною відправкою, що додає клієнтські накладні витрати. Це обмеження методики варто врахувати: результат відображає продуктивність конкретного способу завантаження, а не теоретичну межу можливостей InfluxDB.

Інші обмеження дослідження: тестування проведено на одному локальному вузлі без реплікації, обсяг даних є помірним порівняно з промисловими системами моніторингу активності, індекси PostgreSQL використано без додаткового покращення.

Висновки

Проведене порівняльне дослідження на датасеті PAMAP2 показало, що жодна з двох СУБД не є універсально кращою: вибір визначається профілем навантаження. PostgreSQL доцільний, коли переважають точкові вибірки за індексованим діапазоном часу та багатоатрибутна фільтрація за значеннями. У свою чергу InfluxDB доцільний для типових часово-рядових операцій – агрегації за тегами, даунсемплінгу та отримання останнього значення на серію. PostgreSQL також суттєво швидший за використаний спосіб масового завантаження даних.

Перспективні напрями подальшої роботи: тестування СУБД на більшому обсязі даних і в кластерній конфігурації, оцінка обсягу дискового простору та ефективності стиснення.

Список використаних джерел

1. Grzesik P., Mrozek D. Comparative Analysis of Time Series Databases in the Context of Edge Computing for Low Power Sensor Networks. SpringerLink. URL: https://doi.org/10.1007/978-3-030-50426-7_28.
2. InfluxDB vs PostgreSQL. InfluxData. URL: https://www.influxdata.com/comparison/influxdb-vs-postgres/.
3. Multiple time series database on microservice architecture for IoT-based sleep monitoring system - Journal of Big Data. SpringerLink. URL: https://link.springer.com/article/10.1186/s40537-022-00658-4.
4. Low Cost UGC CARE Approved Journal Publication and Scopus Indexed Standard. URL: https://www.ijnrd.org/papers/IJNRD2605098.pdf.
5. UCI Machine Learning Repository. UCI Machine Learning Repository. URL: https://archive.ics.uci.edu/dataset/231/pamap2+physical+activity+monitoring.
6. Clerke D. PostgreSQL for Data Engineers: Indexes, Bulk Loads, and the Patterns That Actually Matter. DEV Community. URL: https://dev.to/de_clerke/postgresql-for-data-engineers-indexes-bulk-loads-and-the-patterns-that-actually-matter-25fl.

Al-Akwaa K. H. M. A.

OPEN-SOURCE DATA SCIENCE FOR SUSTAINABLE DEVELOPMENT: A REPRODUCIBLE ANALYTICAL FRAMEWORK

Abstract. Open-source data science is increasingly used to study sustainable development, yet an available repository does not necessarily make an analysis reproducible. A workflow may depend on a changing API response, an undocumented filter, an unpinned software package, or a manual step that is not visible in the final figure. These problems are particularly important in Sustainable Development Goal (SDG) research, where official indicators are often combined with external data and reused by people who were not involved in the original analysis.

This paper presents a practical framework for reproducible open-source data science in sustainable-development research. The framework links six elements: the semantic definition of an indicator, data and access documentation, explicit transformations, environment and workflow control, provenance from inputs to claims, and responsible release practices. It is designed as an implementation profile rather than a replacement for existing open-science or FAIR standards.

A working pilot was developed around the United Nations SDG indicator EG_ACS_ELEC, the proportion of the population with access to electricity. The pipeline queried the United Nations SDG API, recorded the request and retrieval details, filtered and validated the data, produced a summary table and chart, and stored file hashes in a provenance manifest. The initial run received 19,000 API records and retained 309 observations for 13 geographic areas covering 2000–2024. The workflow was then executed again in a clean Windows environment. Both automated tests passed in the clean rerun, and the same analytical outputs were produced. The results show that a small open-source project can preserve more than code alone: it can also preserve the source request, data assumptions, software environment, validation evidence, and generated outputs.

Keywords: open-source software, sustainable development, SDG data, reproducibility, open science, FAIR data, research software, data provenance

1. Introduction

Data-driven analysis is now part of many discussions about sustainable development. Researchers and practitioners use data to examine energy access, health, education, poverty, climate change, food security, and urban development. The United Nations Sustainable Development Goals provide a common structure for many of these questions. However, the existence of an official indicator does not automatically make the analysis that uses it reproducible.

In a typical data-science project, the analyst may download an API response, remove missing values, select a geographic dimension, convert a year field, calculate a summary, and create a chart. Some of these decisions may appear in code, while others remain implicit. If the source API changes, if the software environment is rebuilt on another computer, or if a second researcher tries to understand the chart, the original result may be difficult to reproduce.

Open-source software reduces this problem by making code available for inspection, modification, and reuse. It does not solve the whole problem. A public repository can still lack a record of the data request, the indicator definition, the transformation rules, or the computational environment. A dataset can be findable and downloadable while the analysis that produced a published conclusion remains unclear.

This paper addresses that gap by proposing an operational framework for reproducible open-source SDG analytics. The framework is intentionally modest. It does not claim that one project can solve every issue in data governance, statistical validity, or sustainable development. Instead, it asks a practical question:

What should an open-source SDG workflow preserve so that another researcher can understand it, rerun it, inspect its outputs, and reuse it responsibly?

The paper makes four contributions. First, it identifies a gap between existing open-science guidance and the day-to-day requirements of an executable SDG workflow. Second, it defines six

connected layers for that workflow. Third, it demonstrates the framework through a real implementation using the UN SDG API. Fourth, it evaluates the implementation through a local run and a second clean-environment rerun.

2. Background and Related Work

2.1 Open science and sustainable development

UNESCO presents open science as a set of principles and practices that make scientific knowledge more accessible, inclusive, equitable, and sustainable [1]. Its Recommendation on Open Science links openness with transparency, scrutiny, reproducibility, accountability, collaboration, and collective benefit. These values are relevant to SDG research because the results are often used across institutions, countries, disciplines, and levels of government.

The policy framework is important, but a policy statement does not tell a researcher exactly how to record an API request or how to show that a figure can be regenerated. There is still a practical layer between open-science principles and a working analytical repository.

The United Nations Statistics Division provides an important foundation for this layer. The SDG Global Database and its metadata repository publish official indicator data, definitions, and methodological information [10] [11]. A reproducible workflow should therefore record not only the name of an indicator, but also its series code, definition, source endpoint, retrieval time, dimensions, units, and any filtering decisions.

2.2 FAIR data and research software

The FAIR Guiding Principles organise good data practice around findability, accessibility, interoperability, and reusability [3]. They emphasise persistent identifiers, rich metadata, clear access protocols, licences, provenance, and community standards. These concepts are useful when SDG workflows combine official data with local or contextual sources.

FAIR is not, by itself, a complete reproducibility method. It does not guarantee that a transformation is valid, that a statistical claim is correct, or that an analysis can be executed in a new environment. It provides a strong data-management foundation, but the workflow still needs explicit computational and analytical documentation.

The Turing Way provides practical guidance for reproducible computational research. It recommends version control, testing, reproducible environments, and research compendia that bring code, data, methods, metadata, and outputs together [6] [7]. It also treats research scripts as software that require versioning, quality assurance, licensing, and maintenance [8]. These practices are closely connected to the goals of Free and Open Source Software because they turn an analysis into a reusable and inspectable artefact.

2.3 Reproducibility as an execution problem

Previous research has shown that code availability is not equivalent to successful reproduction. Studies of computational notebooks have identified dependency failures, hidden state, execution-order problems, and notebooks that do not run outside their original environment [12] [13]. Artifact-review approaches distinguish between documentation, consistency, completeness, and executability [15]. Provenance research further shows the value of recording inputs, algorithms, parameters, transformations, dependencies, and execution environments [16].

These contributions are valuable, but they remain distributed across different communities. UNESCO addresses open-science values. FAIR addresses data and metadata. Official statistics organisations maintain indicators and revisions. Research-software communities address code, environments, and compendia. The researcher building an SDG workflow must still connect these pieces.

3. Research Gap and Questions

3.1 Research gap

Existing guidance does not yet provide a simple, evaluated, end-to-end reference implementation for open-source SDG analytics. In particular, it does not consistently connect the following elements in one operational chain:

– the official meaning and version of an indicator;
– the source API request and data-access conditions;
– external or local data used in the analysis;
– transformations, filters, joins, and missing-data rules;
– the software environment and execution command;
– the provenance linking outputs to inputs and runs; and
– the release of reusable code, documentation, and lawful data derivatives.

The contribution of this paper is therefore deliberately bounded. It does not claim to invent the first reproducibility framework. It proposes a practical SDG-specific integration of established open-science, FAIR, official-statistics, and research-software practices, and tests the integration with an executable pilot.

3.2 Research questions

RQ1. What minimum technical and documentation components are required for an SDG analytical result to be independently rerunnable and auditable?

RQ2. How can official SDG data and metadata be connected to an executable open-source workflow without losing information about source, version, filtering, and transformation?

RQ3. Does a clean-environment rerun reproduce the pipeline outputs and automated validation results produced during the original run?

RQ4. Which parts of the reproducibility chain remain dependent on external conditions, such as API availability or changing source data?

4. The Reproducible Analytical Framework

The proposed framework has six layers. Each layer addresses a different failure point in an analytical workflow.

Table 1 – Layers of the proposed reproducible analytical framework

Layer	Purpose	Evidence produced
Semantic anchor	Preserve the exact meaning of the indicator	Series code, definition, unit, dimensions, metadata source
Data closure	Document where the inputs came from and how they were obtained	API endpoint, query, retrieval time, page count, access conditions
Transformation contract	Make data-processing decisions visible	Filters, conversions, validation rules, missing-value handling
Environment and workflow closure	Make execution possible in another environment	Python version, dependencies, commands, tests, output paths
Provenance and claim closure	Link outputs to data and execution evidence	Manifest, hashes, output files, claim-to-evidence links
Release and stewardship	Support responsible reuse and maintenance	README, licence, documentation, limitations, archive or release

4.1 Semantic anchor

The semantic anchor prevents the analysis from becoming detached from the indicator itself. In this pilot, the anchor is the UN SDG series code EG_ACS_ELEC, defined as the proportion of the population with access to electricity. The project also records the related SDG context, the data source, the year range, and the geographic dimension used for filtering.

4.2 Data closure

Data closure means that another researcher can identify the source and understand how the data were retrieved. The manifest records the UN SDG API endpoint, the series code, the page size, the

number of pages downloaded, the number of records received, the retrieval timestamp, and the ALLAREA location filter.

This approach is important because the phrase “downloaded the data” is too vague for a reproducible study. A future researcher needs to know what was requested and what was retained.

4.3 Transformation contract

The transformation contract states what happened between the API response and the analytical files. In this pilot, records were filtered to the selected location dimension, records without usable values were removed, years were converted into an analytical year field, values were treated as percentages, and duplicate country-year observations and invalid ranges were checked.

The contract also records a limitation: the latest available observation is not necessarily from the same year for every geographic area. This prevents the summary output from being interpreted as a strict same-year comparison.

4.4 Environment and workflow closure

The project was executed with Python 3.12.4 on Windows 10. The dependencies were installed in a virtual environment. The workflow was run from the command line, and two automated tests were executed with pytest.

A clean rerun is important because a successful run on the developer’s computer does not prove reproducibility. In the second run, the project was copied to a clean project folder, a new virtual environment was created, the dependencies were installed again, and the pipeline and tests were executed without relying on the original environment.

4.5 Provenance and claim closure

The project creates provenance/run_manifest.json. The manifest records the source request, data counts, year range, software information, output paths, and SHA-256 hashes. The raw data, processed data, summary table, and chart are therefore connected to a specific run.

The project also separates supported claims from claims that require further evidence. For example, local execution and clean rerun success are supported by terminal output and manifests. A claim about general reproducibility across all SDG projects would not be supported by this pilot.

4.6 Release and stewardship

The project includes a README describing installation and execution. It also documents the main limitation of the pilot and states how the source data were obtained. The code and documentation can be released under an open-source licence, while the source data remain subject to the terms of the original provider.

5. Methodology

5.1 Case selection

The pilot uses the UN SDG indicator EG_ACS_ELEC, the proportion of the population with access to electricity. The indicator was selected because it has a direct connection to SDG 7, has an official API series code, and can be retrieved through a documented public endpoint.

The study is not intended to rank countries or regions. The purpose is to test whether the workflow can preserve the data request, validate the values, generate outputs, and repeat the process in a separate environment.

5.2 Pipeline implementation

The pipeline performs the following steps:

1. Request the paginated series data from the UN SDG API.
2. Record the URL, series code, page size, page count, status code, and retrieval time.
3. Filter the returned records using the selected location dimension.
4. Retain usable observations and convert the analytical fields into consistent types.
5. Validate the required columns, year range, percentage range, and duplicate keys.
6. Save the raw and processed CSV files.

7. Produce a summary table based on the latest available observation for each geographic area.
8. Create a chart using geographic-area names rather than numeric codes.
9. Calculate SHA-256 hashes and store them in the run manifest.

5.3 Validation

The test suite includes two tests. The first verifies that a valid data frame passes the validation function. The second verifies that a value outside the expected percentage range is rejected. These tests are small, but they demonstrate that the validation rules are executable and can fail when the data violate the declared assumptions.

5.4 Clean rerun

The second execution followed the documented procedure in a separate clean project folder. A new virtual environment was used. The dependencies were installed again. The pipeline was run, and the test suite was executed from the clean environment.

The clean rerun was recorded in provenance/clean_rerun_terminal.txt. The evaluation focuses on whether the workflow can be rebuilt and executed without undocumented manual changes.

6. Results

6.1 Initial execution

The initial execution completed successfully. The pipeline queried 19 pages from the UN SDG API, with 1,000 records requested per page. The API returned 19,000 records before filtering. After filtering and validation, 309 observations remained for 13 geographic areas. The available years ranged from 2000 to 2024.

The API returned HTTP status code 200. The run manifest recorded the retrieval time as 19 September 2026 at 02:33:27 UTC. The software environment was Python 3.12.4 on Windows 10, with pandas 2.3.3.

The pipeline generated four main output artefacts: the raw data file, the processed data file, a summary CSV file, and a chart. The manifest stored SHA-256 hashes for each file. The raw and processed files had the same hash, which is expected because the current processing stage does not apply a substantive alteration to the retained rows. Recording the equality is still useful because it makes the relationship between the two files explicit.

6.2 Data summary

The summary table contains the latest available observation for each geographic area. Twelve areas had a latest observation in 2024, while one area had a latest observation in 2008. This difference is a property of the source data and does not indicate a pipeline failure.

The result means that the table should not be interpreted as a strict 2024 comparison for all areas. The framework makes this limitation visible instead of hiding it behind a single label such as “latest year.” This is a small but important example of why provenance and metadata matter in SDG analysis.

6.3 Visualisation

The first chart version used numeric geographic-area codes on the vertical axis. Although the chart was technically generated, it was not suitable for interpretation because the labels were difficult for a reader to understand. The chart was then corrected to use geographic-area names.

This change did not alter the analytical values. It improved the readability and auditability of the output. A researcher reviewing the figure can now connect the displayed labels to the records in the processed dataset without consulting a separate code list.

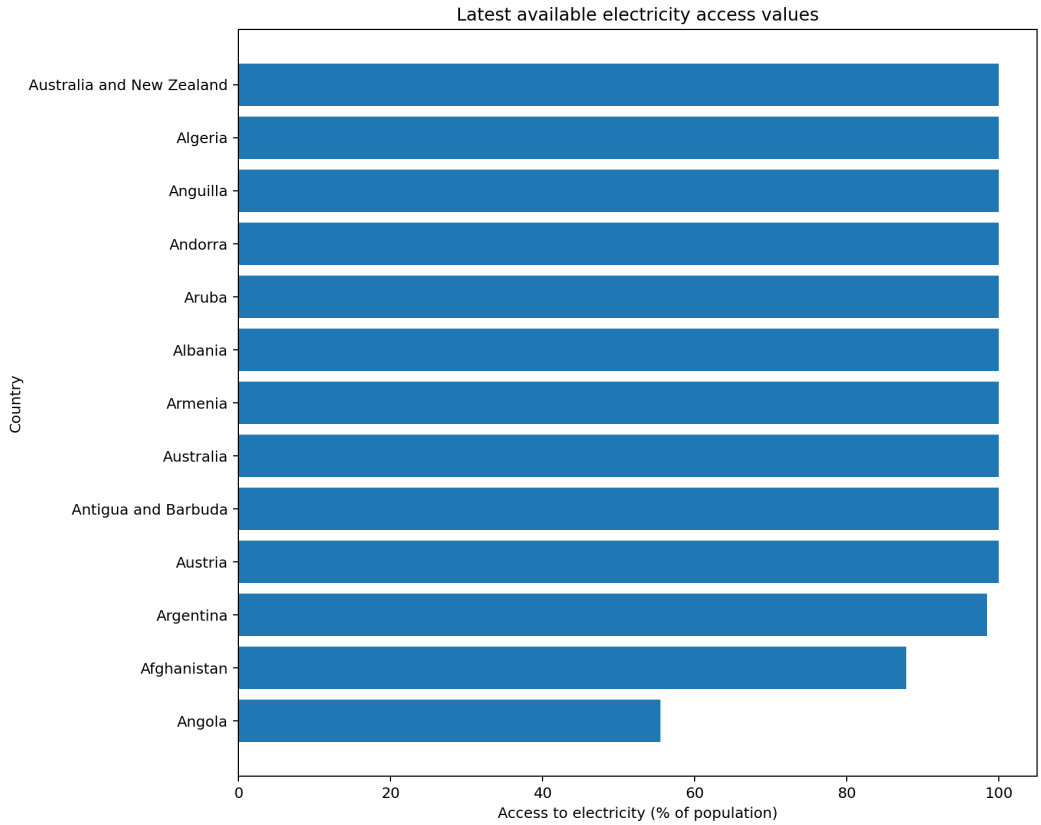


Figure 1 – Latest available electricity-access values for the 13 geographic areas retained by the pipeline. The figure reports the latest available observation for each area; the observation year is therefore not necessarily identical across all areas.

6.4 Test results

The original run collected two automated tests, and both passed. The clean rerun also collected two tests, and both passed in 2.17 seconds. The terminal displayed 14 matplotlib deprecation warnings during the clean rerun. These warnings did not stop the pipeline and did not cause a test failure. They are recorded as environment messages rather than treated as analytical errors.

The clean rerun therefore reproduced the core execution behaviour: the environment was created, the pipeline ran, the main analytical outputs were generated, and the validation tests passed. This statement refers to the substantive outputs of the workflow, not to a claim that every file was byte-for-byte identical across runs. Because the pipeline depends on an external API, response content, ordering, or file hashes may change when the source is updated.

6.5 Reproducibility outcome

After the clean rerun, the project can reasonably be described as reproduced in a second Windows environment under the documented procedure. This claim is limited to the pilot implementation and its particular source conditions. It means that the documented workflow and core analytical outputs were reproduced; it does not imply byte-for-byte identity for every file or future API response.

The reason for this qualification is that the source API may change. A later request could return revised data, a different release, or a changed response order. The manifest and retrieval information make such changes detectable. They do not freeze an external API that the project does not control.

7. Discussion

The pilot shows that reproducibility is not a single property. It has several parts. The source request was documented. The data were filtered according to an explicit rule. The values were validated. The workflow ran in a clean environment. The tests passed twice. The outputs were saved and hashed. Each of these facts provides a different type of evidence.

The pilot also shows why a human review step remains necessary. The first chart used numeric area codes, which were acceptable for computation but poor for communication. The code did not fail, and the tests did not detect the presentation problem. A researcher still had to inspect the output and recognise that the figure was not readable enough for a paper.

The difference between the initial and clean runs also illustrates the limits of reproducibility. The project can reproduce the workflow, but the external API may change over time. This is why a good manifest should preserve the request, timestamp, record counts, and hashes. For a stronger archival release, the project should also preserve a lawful snapshot or a documented source version when the provider's terms allow it.

Another limitation concerns the data scope. The final dataset contains 13 geographic areas, and the latest observation year is not uniform. The pilot should therefore not be used to make a broad conclusion about global electricity access. Its purpose is methodological: to show how a small SDG analysis can be made inspectable and rerunnable.

The framework could be extended in several ways. A future version could connect the indicator to its official metadata record, add a second licensed contextual dataset, record a formal data contract, and use a workflow engine or container. A larger study could evaluate multiple SDG indicators and compare notebook-based, script-based, and workflow-engine-based projects.

8. Conclusion

This paper proposed and tested a practical framework for reproducible open-source data science in sustainable-development research. The framework connects semantic definitions, source documentation, transformations, computational environments, provenance, and responsible release practices.

The pilot used the UN SDG indicator EG_ACS_ELEC. It retrieved 19,000 records from the official API, retained 309 validated observations for 13 geographic areas, generated a summary table and chart, and recorded the run details and file hashes. The same project was then executed in a clean Windows environment. The pipeline completed successfully and both automated tests passed again.

The results do not prove that all SDG analyses are reproducible. They demonstrate something more specific and useful: a small open-source workflow can preserve enough information for a second researcher to rebuild and run it, while also making important limitations visible. That is a practical starting point for more reliable and reusable sustainable-development analytics.

References

1. UNESCO Recommendation on Open Science. https://www.unesco.org/en/open-science/about
2. UNESCO Open Science Toolkit. https://www.unesco.org/en/open-science/toolkit
3. The FAIR Guiding Principles for scientific data management and stewardship. https://www.nature.com/articles/sdata201618
4. CODATA Introduction to FAIR Data Stewardship. https://codata.org/introduction-to-fair-data-stewardship/
5. CODATA WorldFAIR+. https://codata.org/initiatives/making-data-work/worldfair-2/
6. The Turing Way Guide for Reproducible Research. https://book.the-turing-way.org/reproducible-research/reproducible-research/
7. The Turing Way Research Compendia. https://book.the-turing-way.org/reproducible-research/compendia/
8. The Turing Way Software Management Plans. https://book.the-turing-way.org/reproducible-research/rdm/rdm-smp/
9. The Turing Way Open Source Software. https://book.the-turing-way.org/reproducible-research/open/open-source/
10. United Nations SDG Global Database. https://unstats.un.org/sdgs/dataportal/database
11. United Nations SDG Indicators Metadata Repository. https://unstats.un.org/sdgs/metadata/
12. Understanding and improving the quality and reproducibility of Jupyter notebooks. https://pmc.ncbi.nlm.nih.gov/articles/PMC8106381/
13. Computational reproducibility of Jupyter notebooks from biomedical publications. https://arxiv.org/html/2308.07333v1
14. ACM Artifact Review and Badging. https://www.acm.org/publications/policies/artifact-review-and-badging-current
15. Improving Reproducibility of Data Science Pipelines through Transparent Provenance Capture. https://davisjam.github.io/files/publications/RupprechtDavisArnoldGurBhagwat-Ursprung-VLDB20.pdf

Borysenko O. A.
Kyiv National Economic University named after Vadym Hetman, Kyiv, Ukraine
Corresponding author e-mail: compstotion@gmail.com
Scientific advisor: Yaroslav Vozniuk, PhD in Economics

AN ARCHITECTURAL APPROACH TO LLM-BASED ROOT CAUSE ANALYSIS FOR CONTAINER ORCHESTRATION INCIDENTS

Modern cloud-native applications are deployed as dozens or hundreds of loosely coupled microservices running under a container orchestrator such as Kubernetes. The reliability of such systems no longer depends on any single component but on the correctness of interactions between them. A single misconfigured resource limit, a slow downstream dependency, or a transient network partition can trigger a chain of symptoms that surface far from their actual origin, forcing an on-call engineer to manually reconstruct the causal sequence from dozens of uncorrelated dashboards, logs, and alerts.

Traditional observability stacks address this by collecting metrics, logs, and traces, but the correlation between them is still performed by a human. Static alerting rules and dashboards flag individual anomalies without explaining why they occurred or how they relate to one another. This creates alert fatigue and directly increases mean time to diagnosis (MTTD), a problem that recent work has started to address by applying large language model (LLM) agents to incident analysis, using structured telemetry rather than free-form text as the primary evidence source [4, 5].

The architecture proposed in this work introduces an LLM-based agent as a diagnostic layer on top of an existing observability stack, rather than as a replacement for it. Three telemetry sources are collected at the cluster level: the Kubernetes Events API, which reports state transitions such as OOMKilled or CrashLoopBackOff [1]; Prometheus, which exposes resource metrics and firing alerts [2]; and OpenTelemetry traces, which record the path of a request across service boundaries [3]. A telemetry aggregator correlates these three streams by time window and by resource labels (pod, namespace, deployment), producing a unified context object for a given incident window instead of three disconnected feeds.

Consider a representative scenario: a memory leak in a checkout-service pod eventually triggers an OOMKilled event; the resulting restart causes transient 5xx responses; the API gateway, several hops downstream, reports elevated latency and a spike in retries. In a conventional setup these three facts appear as three independent alerts. In the proposed pipeline, the agent receives the correlated context and, through a small set of grounded tools (get_pod_events, query_prometheus_range, get_trace_by_id), retrieves the specific evidence needed to explain the chain, then produces a ranked, natural-language explanation such as: gateway latency is a downstream symptom of an OOMKill on pod X at 14:32, whose configured memory limit of 256Mi is below the peak RSS observed over the preceding week.

A key design constraint is that the agent must not free-generate an explanation from prior knowledge alone: each factual claim in the output is required to be traceable to a specific tool-call result, with the corresponding timestamp and metric or log reference attached. This tool-grounding is what keeps the explanation auditable and substantially reduces the risk of a plausible-sounding but incorrect diagnosis, which is the main failure mode reported for ungrounded LLM use in operational settings.

The resulting causal chain is rendered in the monitoring dashboard as a short sequence of linked events rather than a wall of text, so an engineer can inspect the exact evidence path the agent used before accepting or rejecting its conclusion (Fig. 1).

The approach has clear boundaries. LLM inference introduces a delay of several seconds, which makes it unsuitable for hard real-time control loops such as horizontal pod autoscaling; the agent is therefore positioned strictly as an advisory diagnostic layer above the existing alerting and autoscaling mechanisms, not as their replacement. The quality of its output is also bounded by the completeness and consistency of the underlying telemetry: poorly labeled resources or missing trace context degrade the correlation step before the agent is ever invoked.

Even within these limits, shifting from a set of uncorrelated alerts to a single synthesized incident narrative meaningfully reduces the cognitive load of the initial triage phase, which is typically the largest and least automatable part of incident response. Based on architectural estimates and preliminary trials on a small simulated Kubernetes cluster, such a pipeline is expected to shorten mean time to diagnosis by roughly 25-35% compared to manual log and metric correlation, primarily by removing the initial evidence-gathering step from the human workflow.

Fig. 1. LLM-based root cause analysis pipeline for a containerized cluster

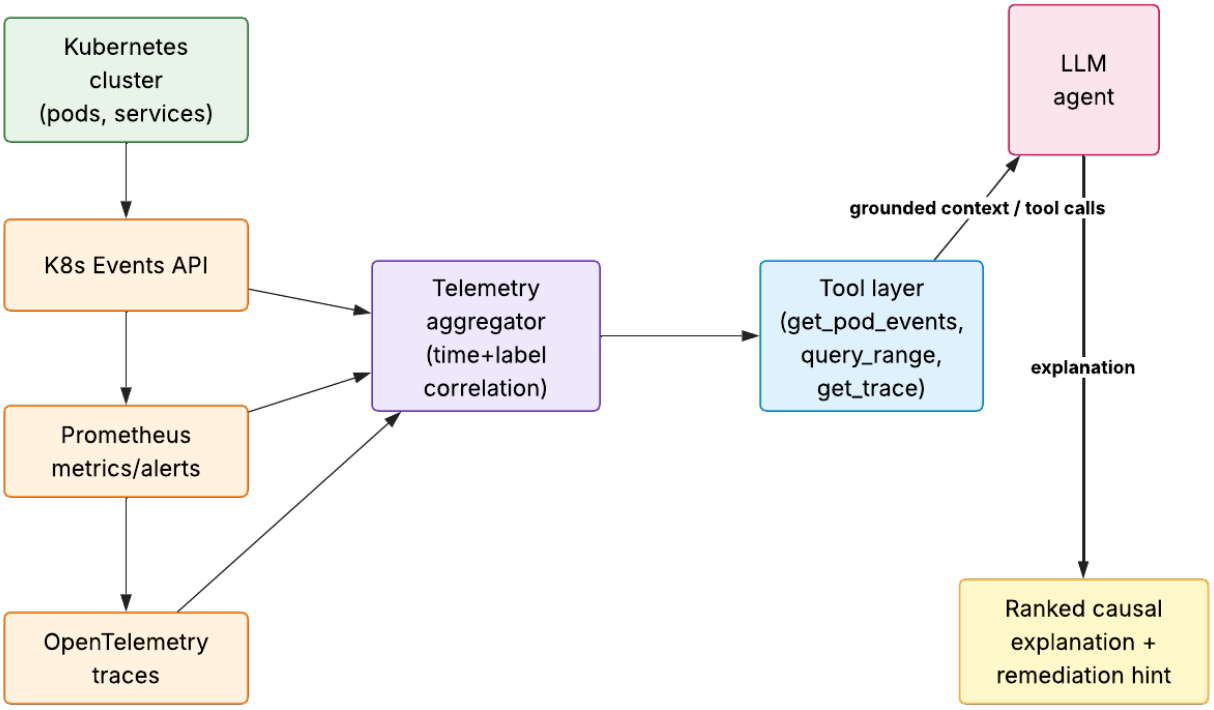


Figure 1 – LLM-based root cause analysis pipeline for a containerized cluster

In conclusion, positioning an LLM agent as a tool-grounded diagnostic layer over standard Kubernetes observability data offers a practical path toward reducing on-call cognitive load without displacing existing alerting or autoscaling mechanisms, and the architecture described here is intended as a foundation for the automated remediation layer developed in the broader scope of the author's research.

References

1. Alerting Overview. Prometheus Documentation. URL: https://prometheus.io/docs/alerting/latest/overview/ (accessed: 22.09.2026)
2. Debug Running Pods. Kubernetes Documentation. URL: https://kubernetes.io/docs/tasks/debug/debug-application/debug-running-pod/ (accessed: 22.09.2026)
3. Multi-Agent Systems for Root Cause Analysis in Microservices. arXiv. URL: https://arxiv.org/html/2605.03505v1 (accessed: 22.09.2026)
4. Simplifying Root Cause Analysis in Kubernetes with StateGraph and LLM. arXiv. URL: https://arxiv.org/pdf/2506.02490 (accessed: 22.09.2026)
5. Traces. OpenTelemetry Documentation. URL: https://opentelemetry.io/docs/concepts/signals/traces/ (accessed: 22.09.2026)

Burmaka I.
Chernihiv Polytechnic National University, Department of Information Technologies and Software Engineering, Chernihiv, Ukraine
Corresponding author: Ivan Burmaka, e-mail: Ivan.Bourmaka@stu.cn.ua, ORCID: 0000-0002-7476-5757

LOCAL, OPEN, AND ENERGY-EFFICIENT: HARNESSING X86 NPUS FOR EVERYDAY AI TASKS

Abstract. This thesis examines the architecture, development frameworks, and practical applications of Neural Processing Units in modern laptop processors, specifically focusing on Intel Core Ultra 200V and AMD Ryzen AI platforms. The study provides a comprehensive analysis of NPU hardware architectures, comparing Intel's automated, hardware-managed approach against AMD's programmable XDNA architecture. The research explores the open-source software ecosystem supporting NPU acceleration, including OpenVINO, AMD IRON, ONNX Runtime, and Windows ML frameworks. Through practical experimentation on AMD Ryzen AI 7 350 processor, the thesis demonstrates NPU advantages in real-world applications including video segmentation and image generation. Performance testing reveals that NPUs achieve superior power efficiency compared to integrated GPUs. The findings establish NPUs as critical components for energy-efficient, privacy-preserving on-device AI inference in battery-powered computing platforms.

Keywords: Neural Processing Unit (NPU), AI PC, energy-efficient inference, on-device machine learning, laptop AI acceleration, generative AI hardware.

Laptop NPU Architecture and Computational Possibilities

Intel's Neural Processing Unit targets machine vision and deep learning inference in AI PCs, delivering high throughput with exceptional power efficiency. The Core Ultra 200V series supports INT4, INT8, and FP16 data types for flexible precision inference.

The architecture comprises two subsystems: Host Control and the Deep Learning Accelerator (Fig. 1 (a)). Host Control uses a 64-bit RISC-V microcontroller managing job submission through a PCIe interface [1]. The Deep Learning Accelerator features six Neural Compute Engine (NCE) Tiles totaling 12,288 INT8 MACs, 12 DSP processors, and 9 MB of near-compute memory. An M2I (Media-to-Inference) engine accelerates preprocessing transforms including cropping, scaling, and colorspace conversion. The architecture balances programmability with automation, using hardware-managed synchronization to optimize inference workloads for power-constrained platforms.

AMD's Ryzen AI NPU, derived from Versal ACAP technology, employs a flexible tile-based AI Engine (AIE) array (Fig. 1 (b)). Each AIE tile contains vector processors optimized for ML operations, local scratchpad memory, and dedicated DMA engines, interconnected via a packet-switched network-on-chip. This architecture emphasizes programmability, allowing developers to explicitly define data movement and computational workflows.

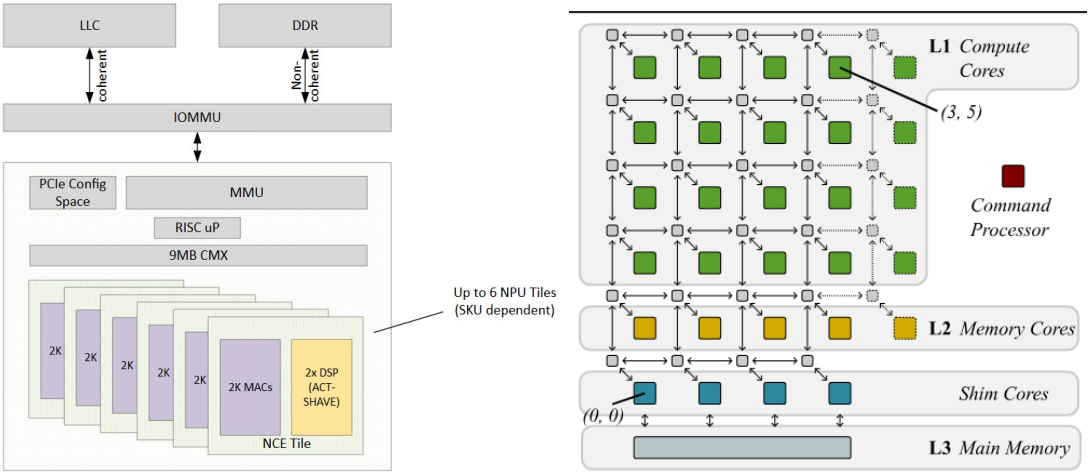


Figure 1 – Generic architecture of Intel NPU IP (a) and AMD XDNA NPU (b)

AMD offers both high-level tool-flows (Ryzen AI Software for PyTorch/TensorFlow) and the open-source IRON toolflow for low-level programming. IRON enables developers to describe NPU layouts via Python scripts, with compute kernels in C++ using the AIE API, generating optimized configurations for specific problem sizes [2].

Both NPUs excel at accelerating matrix multiplication, convolution, and transformer operations crucial for modern AI workloads. Intel's approach prioritizes ease of use through automated compiler optimization and hardware-managed scheduling, while AMD offers deeper programmability for specialized applications. Both platforms support mixed-precision computation (FP16/INT8) and emphasize on-chip SRAM execution to minimize power consumption—critical for battery-powered AI PCs. Both approaches enable efficient on-device inference that preserves privacy and reduces latency.

NPU development frameworks

The open-source ecosystem is foundational for unlocking the energy-efficient capabilities of modern x86 NPUs. For Intel architectures, OpenVINO serves as the primary toolkit, providing model optimization, quantization (via NNCF), and deployment tools. AMD's ecosystem relies on the AMD IRON framework (built on MLIR-AIE) alongside the Ryzen AI software stack for targeting XDNA AI Engine tiles.

Bridging these hardware-specific environments is ONNX Runtime, a universal execution engine. By leveraging dedicated Execution Providers—Vitis AI EP for AMD and OpenVINO EP for Intel—ONNX Runtime abstracts hardware complexities, allowing seamless deployment across both platforms using a unified API.

Intel provides a fully open-source pipeline from OpenVINO framework to Linux kernel drivers, while AMD's ecosystem uses a mixed approach: open-source drivers (amdxdna) and runtimes (XRT), but proprietary Vitis AI compiler and firmware components.

Microsoft's Windows ML framework offers OS-level NPU abstraction through DirectML, providing hardware-agnostic acceleration across Intel, AMD, and Qualcomm NPUs via standard Windows APIs, trading low-level optimization control for deployment simplicity.

Applied Open-Source Solutions for NPU Acceleration

While operating systems integrate NPU capabilities directly into native APIs (such as Windows Studio Effects), the independent open-source software (FOSS) community is actively developing standalone tools. These projects allow developers and users to harness hardware acceleration directly across different platforms, including Linux, without relying on proprietary OS ecosystems.

Audacity with OpenVINO Plugins. For advanced audio processing, Intel has developed a suite of AI plugins for the open-source audio editor Audacity, built entirely on the OpenVINO toolkit. This integration allows users to run complex machine learning models locally on Intel NPUs to perform tasks like noise suppression, vocal isolation (stem separation), and even local music generation.

FastFlowLM (FLM) is a breakthrough open-source runtime designed exclusively for AMD Ryzen AI processors. FastFlowLM allows users to run popular large language models in INT4 quant and vision-language models locally using only the NPU. It operates with an ultra-lightweight footprint, and exposes an OpenAI-compatible REST API. It is not showing very high speeds but NPU based software is much cooler than GPU one (when LLM runs on AMD iGPU it heats system to maximum available temperature and causes throttling) [4].

AMD Ryzen AI Software Workflows for Generative Models. To enable custom application development, AMD provides foundational FOSS workflows for generative AI pipelines. Through the Ryzen AI Software stack, developers can deploy open-source generative models using ONNX Runtime coupled with the Vitis AI Execution Provider. This allows developers to seamlessly run models like Stable Diffusion or various LLMs directly on the NPU's spatial architecture. Furthermore, with the introduction of tools like AMD IRON, researchers gain low-level, hardware-agnostic control over the AI Engine tiles. This specific ecosystem empowers the community to build custom, highly efficient inference pipelines [5].

Wide number of APIs allows to write a custom software. As an example, human-background segmentation app for web-camera video processing was developed based on Windows ML framework. Also, not every neural network can effectively work on NPU, because some operations can not be performed on NPU, at least with Windows ML. Non compatible layers are usually offloaded to the CPU, but it degrades performance and increases energy consumption, making NPU offload useless. But Mediapipe_selfie_segmentation and PP-HumanSeg models work perfectly, using about 10% of NPU resources on Ryzen AI 7 350 CPU.

Performance test and efficiency

To test neural models performance on NPU and iGPU of Ryzen AI 7 350 processor few tasks were prepared. All experiments were performed when maximal energy saving mode was turned on (Motherboard Eco mode and Windows power saving, to decrease CPU boosting impact on power consumption). First one is Video segmentation model PP-Human-seg (Table 1). On both devices we have superior performance for such small model. But GPU power consumption is slightly higher.

Table 1 – Background detection model performance characteristics

	NPU	**GPU**
Power used for 1.5 min	0.34 Wh	0.44Wh
Temperature	44 C	50 C
Device load	12%	7%

Second experiment was related to image generation with StableDiffusion XL (Table 2), On Windows there are some performance issues with DirectML and Vulkan backend causing performance degradation (16 times slower than the NPU) and high energy consumption. ROCm backend inside ComfyUI performs better, but still far from NPU (because NPU uses optimized model quants).

Table 2 – Stable Diffusion XL performance on NPU

	NPU	**GPU Vulkan**	**GPU ROCm**
Power used for 1024x1024 image 50 steps	0.73 Wh	5 Wh	3Wh
Temperature	55C	72C (hits temp. limit)	58C
Device load	97+%	Up to 100%	93%
Generation speed	0.6 sec/it	8 sec/it	5.5 sec/it

Conclusions

The integration of Neural Processing Units in laptop processors enables practical on-device AI inference with substantial energy savings. Experimental results on Ryzen AI 7 350 demonstrate that NPUs consume 23% less power than integrated GPUs for video segmentation tasks and an impressive 75% less for image generation. This translates to 2-4 hours of extended battery life in AI-intensive workflows, while maintaining 20-28°C lower operating temperatures that prevent thermal throttling.

Performance results reveal surprising capabilities: while NPUs predictably excel at lightweight tasks like real-time webcam segmentation, they also deliver competitive performance for complex generative workloads. For affordable everyday tasks such as background removal and noise suppression, NPU performance proves more than adequate with minimal power draw.

Beyond efficiency gains, NPU-accelerated on-device inference eliminates cloud dependency, preserving user privacy for sensitive applications like voice and video processing while reducing latency to enable truly real-time interactive AI experiences. The maturity of open-source frameworks demonstrates that NPUs have transitioned from specialized hardware to accessible platforms for community-driven innovation, establishing them as essential components for the emerging AI PC category.

References

1. Intel Core Ultra 200V Series Processors https://edc.intel.com/content/www/us/en/secure/design/confidential/products/platforms/details/lunar-lake-mx/core-ultra-200v-series-processors-datasheet-volume-1-of-2
2. Rösti, A., & Franz, M. (2025). Unlocking the AMD Neural Processing Unit for ML Training on the Client Using Bare-Metal-Programming Tools . arXiv. https://doi.org/10.48550/ARXIV.2504.03083
3. OpenVINO 2026.4 — OpenVINOTM documentation. (n.d.). https://docs.openvino.ai/2026/index.html
4. FastFlowLM · FastFlowLM. (n.d.). https://fastflowlm.com/
5. Ryzen AI Software — Ryzen AI Software 1.8.0 documentation. (n.d.). https://ryzenai.docs.amd.com/en/latest/index.html

Byvoino T. P., Byvoino P. G.
Chernihiv Polytechnic National University, Department of Information Technologies and Software Engineering, Chernihiv, Ukraine
Chernihiv Polytechnic National University, Department of Information and Computer Systems, Chernihiv, Ukraine
Corresponding author: Taras Byvoino, e-mail: dec-fpo@stu.cn.ua, ORCID: 0000-0001-6914-5441

SIMULATION MODELING AND STATISTICAL ANALYSIS OF THE EFFICIENCY OF AN EMERGENCY MEDICAL SERVICE STATION

Abstract. This paper presents the results of the development and study of an object-oriented simulation model of an urban emergency medical service station designed to analyze operational efficiency and justify an appropriate number of medical teams. The model was implemented in Java using a discrete-event approach and the specialized Simulation framework. The developed computer application visualizes the dynamics of call-handling processes, enables the study of transient states and the collection of statistical characteristics, and supports series of computer experiments using regression and analysis of variance. The simulation results were used to investigate the effect of the number of medical teams on the average length of the call queue, response time, patient transportation time to the hospital, and team idle time. For the test model, the use of eleven teams was found to provide maximum economic efficiency and minimize staff idle time, whereas deploying at least twelve teams enables a higher quality of service for the population. The results confirm that the developed application can be used as a decision-support tool for planning and optimizing the resources of emergency medical services.

Keywords: simulation modeling, discrete-event simulation, emergency medical care, queuing system, medical team, statistical analysis, regression analysis, Java.

Introduction. Optimizing the structure and resource management of urban emergency medical services is a critically important systems-analysis task, made significantly more difficult by contemporary instability and dynamic migration-related fluctuations in population size. Since call arrivals, ambulance movements, and medical examinations are distinctly stochastic processes, classical analytical models of queuing theory cannot fully account for nonlinear delays and transient system states. This makes the development of a specialized simulation tool for multi-scenario studies of emergency services particularly relevant [1].

The aim of this study is to implement an object-oriented simulation model of an urban emergency medical service station and perform statistical and regression analyses of its output characteristics in order to justify an appropriate number of medical teams under high workload conditions.

Materials and Methods. The model was implemented in Java using the discrete-event Simulation framework [2]. Process coordination is based on interactions among autonomous active objects (the CallGenerator and medical teams represented by Brigade), parameterized transaction queues, and state recorders of the Store class.

A standalone computer application was developed to conduct the study. Its main purpose is the comprehensive analysis and visualization of processes in queuing systems. The application makes it possible to observe in real time changes in queue length and the workload of medical teams in the model, investigate the nature and duration of transient processes in queues, and automatically obtain summary statistics for the system's output performance indicators. In addition, the application architecture supports large batches of computer experiments to examine in detail how input factors affect service efficiency, performing automated regression and analysis-of-variance analyses of the stochastic simulation results. Figure 1 shows the application interface while investigating the effect of the number of teams on the average call-queue length.

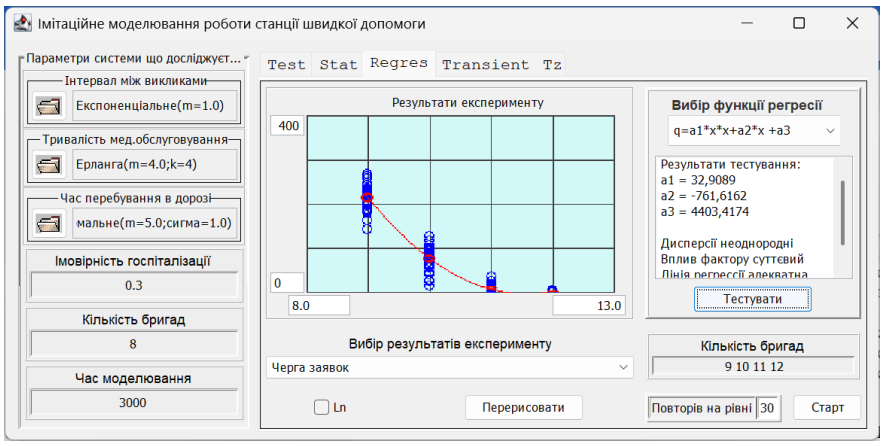


Figure 1 – User interface for investigating the effect of the number of teams on the average request-queue length

Results. At the initial stage, the system's transient behavior was studied using the TransProcessManager component (the Transient tab). The duration of the non-stationary warm-up period from an empty initial state was found to be approximately twenty-five mean interarrival intervals between requests. This made it possible to demonstrate mathematically that a simulation duration of 3,000 hours reduces the initial bias in the output data to a negligible one percent.

To investigate the effect of the number of medical teams on the model's performance, a multifactor experimental design was implemented using the ExperimentManager component (the Regres tab), with thirty repeated runs at each level of the team-count factor (Fig. 1). The experiment examined the factor's influence on the average length of the request queue and the queue of teams waiting for a call, the average response time, the average interval between a call and the patient's arrival at the hospital, and the average team idle time.

The selected sample size and simulation duration made it possible to obtain regression models for the studied parameters that were statistically adequate according to Fisher's criterion. The heterogeneity of variances detected by Cochran's test is explained by the operating characteristics of stochastic queuing systems, in which data dispersion decreases sharply as the system moves from a critically overloaded state to a state in which resources are fully sufficient.

The regression studies made it possible to justify the optimal number of medical teams depending on the emergency service's management priorities. To maximize economic efficiency and minimize staff idle time, eleven teams are optimal for the test model; however, to achieve the highest quality of service for the population, it is reasonable to deploy at least twelve medical teams.

Analysis of variance was performed for the model operating with twelve ambulance teams. With this staffing level, at a 95% confidence level, the average call-queue length remains within the narrow range of 1.49 to 1.78 (not exceeding two calls), while the response time remains consistently between 5.53 and 5.86 minutes.

The empirical distributions of service-performance indicators obtained on the Stat tab are presented as frequency histograms (Fig. 2).

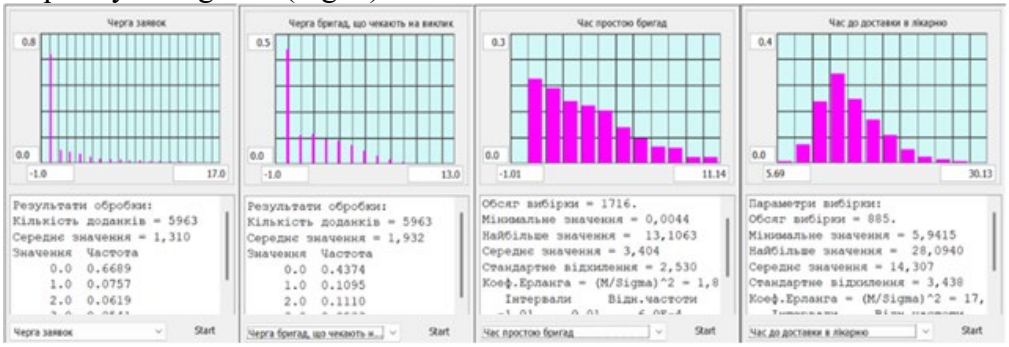


Figure 2 – Frequency histograms of the system's stochastic output indicators when operating with twelve medical teams

Analysis of the resulting distributions of random variables confirms a high level of service quality. The first histogram clearly shows that the most frequent call-queue length is zero, with a frequency of 0.6689. This means that in nearly 67% of cases, new requests from citizens are handled immediately and assigned to available ambulances without any waiting. The mean queue length is only 1.31 calls.

At the same time, analysis of the histograms for staff idle time and travel time confirms the absence of critically overloaded queuing-system states and the rational use of the medical institution's resources. This demonstrates the practical value of the developed application as a digital decision-support tool for management.

Conclusions. The developed computer application has demonstrated high effectiveness as a systems-analysis tool for the process of handling emergency ambulance calls. Healthcare managers can use the application as a reliable analytical basis for risk forecasting and optimizing the technical resources of emergency services.

A promising direction for further development of the application is the creation of a subsystem for adaptive parameterization of interactive data-entry elements. Practical analysis of the model's use has demonstrated the need for an intelligent gateway that can automatically transform intuitive expert estimates of boundary conditions (the optimistic minimum, pessimistic maximum, and most likely everyday operation time) into theoretical characteristics of nonlinear probability distributions. Implementing algorithms for automatic structural identification of distributions based on skewness coefficients will allow end users to account flexibly for the heavy-tail effects of extreme call-handling delays without complicating the program's graphical interface, thereby substantially improving simulation accuracy.

References

1. Strategy for the Digitalization of Ukraine's Healthcare System: Order of the Cabinet of Ministers of Ukraine dated 20 August 2024, No. 782-r. Available at: mozu.gov.ua (accessed 12 July 2026).
2. Byvoino, P. H., Byvoino, T. P., & Pavlovskyi, V. I. Development of Simulation Models in the Java Programming Language: Textbook for Higher Education Students in Specialty F7 "Computer Engineering". Chernihiv: National University "Chernihiv Polytechnic", 2025. 135 p.

Didusenko O. A., Bula Y. I., Shymchuk G. V.
Ternopil Ivan Puluj National Technical University, Department of Computer Science, Ternopil, Ukraine
Corresponding author: Grigorii Shymchuk, e-mail: gorych@gmail.com, ORCID: 0000-0003-2362-7386

REPRODUCIBLE FOSS FRAMEWORK FOR RESILIENCE EVALUATION OF CONTAINERIZED DISTRIBUTED SYSTEMS

Abstract. A reproducible open-source framework for evaluating the resilience of containerized distributed systems is proposed. The framework combines k3s/Kubernetes, Chaos Mesh, Prometheus, and Python and focuses on failures that are common in distributed services: pod termination, node unavailability, increased network latency, and packet loss. Four deployment configurations were defined, from a single replica to three replicas with health probes, anti-affinity, and a PodDisruptionBudget. A pilot stochastic workload model with 30 repetitions per configuration was used to validate the metric pipeline before cluster experiments. Median modeled recovery time decreased from 4.54 s for a single replica to 0.48 s for the most resilient configuration; mean error rate decreased from 7.80% to 0.90%, while availability increased from 92.00% to 98.55%. The work provides a compact FOSS methodology for reproducible resilience experiments and clearly separates preliminary model-based results from future execution in a real k3s cluster.

Keywords: open-source software; Kubernetes; k3s; distributed systems; resilience; chaos engineering; fault injection; microservices.

Introduction

Container orchestration has become a standard mechanism for deploying distributed applications, but replication alone does not guarantee service resilience. Failures may occur at several layers: a process can become unresponsive, a pod can be terminated, a worker node can disappear, or network communication can experience latency and loss. Kubernetes provides readiness and liveness probes to remove unready pods from service traffic and restart failed containers, while ReplicaSets maintain a desired number of replicas [1]. These mechanisms create a practical basis for controlled resilience experiments.

Recent research treats fault injection and chaos engineering as systematic techniques for evaluating microservice robustness. Singh et al. assessed Kubernetes microservices under chaos and measured response time, errors, resource use, and request throughput [2]. Wang et al. proposed Kubernetes architecture graphs for selecting resilience experiments rather than relying only on random single-point faults [3]. A 2025 systematic review identified fault-injection configuration, execution, test oracles, and system-under-test selection as core design decisions for microservice testing [4]. Digital-twin approaches have also been used to reduce the cost of testing Kubernetes applications under computing and network faults [5].

The objective of this work is to define a small, fully open-source framework that can be repeated in university laboratories and used as a baseline for further research. The proposed implementation uses k3s as a lightweight conformant Kubernetes distribution [6], Chaos Mesh for fault injection [7], Prometheus-compatible metrics, and Python for experiment orchestration and analysis. A pilot model is used only to verify the metric definitions and expected direction of effects before running the framework on real nodes.

Materials and Methods

The target testbed consists of one k3s server and three worker nodes. A simple HTTP microservice is deployed behind a Kubernetes Service, while a load generator sends requests at a constant target rate. Prometheus records request counters, latency, pod state, and node state. Chaos Mesh is responsible for controlled pod failures and network disturbances. The logical workflow is shown in Figure 1. The framework is designed so that manifests, failure schedules, and analysis scripts can be stored in a repository and replayed without commercial cloud services.

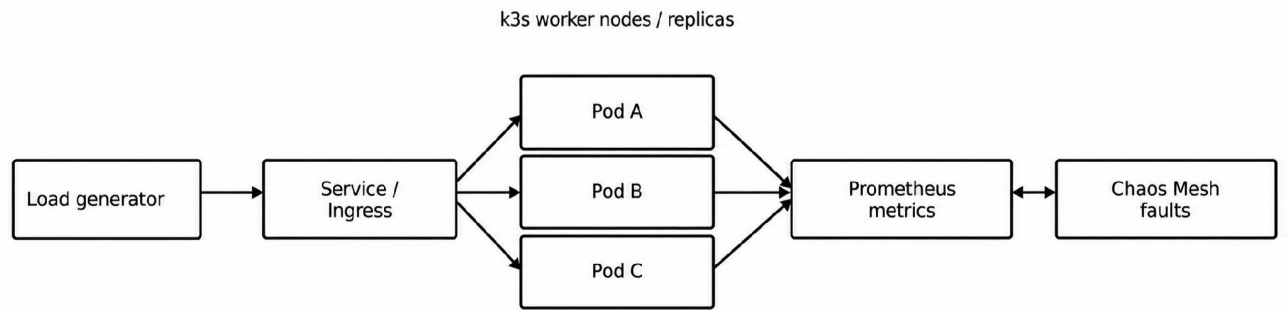


Figure 1 – Reproducible FOSS workflow for container resilience experiments

Four deployment configurations were considered. C1 contains one application replica without dedicated probes. C2 contains three replicas distributed by the scheduler. C3 uses three replicas with readiness and liveness probes. C4 adds pod anti-affinity and a PodDisruptionBudget (PDB) so that replicas are preferentially separated and planned disruptions cannot remove too many healthy instances simultaneously. Readiness probes are important because an unready pod is excluded from Service traffic even if its process is still running [1].

Table 1 – Fault scenarios for the resilience experiment

Scenario	Fault	Duration / level	Observed effect
C-POD	Pod termination	one active pod	rescheduling/restart
C-NODE	Worker unavailable	30 s	replica relocation
C-LAT	Network latency	+250 ms	tail latency
C-LOSS	Packet loss	5%	errors/retries

The pilot workload model generated 30 repetitions for each configuration with a fixed seed. Each repetition included a steady-state period followed by one fault episode. Recovery time was measured from fault activation to restoration of the target request success rate. Availability over the observation window was defined as

$$A = \frac{(T_{total} - T_{unavailable})}{T_{total}} * 100\%. \quad (1)$$

The request error rate was calculated as

$$R_{err} = \frac{N_{error}}{N_{requests}} * 100\%. \quad (2)$$

The p95 latency was used to capture tail behavior. For the future cluster experiment, the same metrics can be derived from HTTP counters and histogram buckets exported to Prometheus. The model does not emulate the internal Kubernetes controller loop; its purpose is to test the analysis workflow and produce falsifiable expectations for the real FOSS testbed.

Results

Table 2 presents the pilot results aggregated over all four fault types. Replication alone reduced the modeled recovery interval and error rate, but health probes produced a further improvement by preventing traffic from being sent to instances that had not yet recovered. The configuration combining probes, anti-affinity, and a PDB had a median recovery time of 0.48 s, a mean error rate of 0.90%, and mean availability of 98.55%. The single-replica baseline was substantially more sensitive to the same injected conditions.

Table 2 – Pilot model-based resilience results

Configuration	Median recovery, s	P95 recovery, s	Error rate, %	p95 latency, ms	Availability, %
C1: 1 replica	4.54	6.09	7.80	422.88	92.00
C2: 3 replicas	1.72	2.12	2.89	321.48	95.59
C3: 3 replicas + probes	0.79	1.03	1.56	259.74	97.38
C4: probes + anti-affinity + PDB	0.48	0.57	0.90	220.47	98.55

Discussion

The modeled trend is consistent with the design of Kubernetes fault-tolerance mechanisms. Multiple replicas reduce dependence on one pod, readiness probes protect the request path from

instances that cannot currently serve traffic, and liveness probes provide automatic restart for processes that enter an unrecoverable state [1]. Anti-affinity reduces the probability that all replicas share one worker, while a PDB constrains voluntary disruptions. These mechanisms address different failure modes and should therefore be tested together rather than treated as interchangeable.

Chaos engineering is useful because it evaluates the system during controlled violations of its steady state. Chaos Mesh is an open-source platform that supports orchestration of pod and network abnormalities [7], which makes it suitable for reproducing C-POD, C-LAT, and C-LOSS. The framework also aligns with current research that combines Kubernetes with controlled faults and evaluates response time, errors, and availability [2-5]. K3s reduces deployment overhead while remaining Kubernetes-conformant, making it appropriate for small laboratories, edge testbeds, and teaching environments [6].

The principal limitation is the synthetic nature of Table 2. The values should not be interpreted as measured k3s performance. Real recovery time depends on probe periods, scheduler load, image availability, node resources, CNI behavior, and application startup time. Consequently, the next stage must deploy the same four configurations on identical nodes, inject each fault repeatedly with Chaos Mesh, and calculate confidence intervals from Prometheus traces. The testbed should also separate planned disruptions from unplanned node loss, because a PDB does not prevent every failure type.

Conclusions

A reproducible FOSS framework for evaluating resilience of containerized distributed systems was proposed. The framework combines k3s/Kubernetes, Chaos Mesh, Prometheus-compatible monitoring, and Python and defines common pod, node, latency, and packet-loss faults. In the pilot model, the transition from one replica to a configuration with probes, anti-affinity, and a PDB reduced median recovery time from 4.54 s to 0.48 s and the mean request error rate from 7.80% to 0.90%.

Future work will execute the scenarios in a physical or virtual k3s cluster, compare application-level and infrastructure-level faults, study the influence of probe parameters on false recovery decisions, and extend the experiment to stateful services and multi-service dependency chains.

References

1. Kubernetes Authors. Configure Liveness, Readiness and Startup Probes. Kubernetes Documentation. https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/ (accessed 01.09.2026).
2. Singh S., Muntean C. H., Gupta S. Boosting Microservice Resilience: An Evaluation of Istio's Impact on Kubernetes Clusters Under Chaos. 2024 9th International Conference on Fog and Mobile Edge Computing (FMEC), 2024, pp. 245-252. https://doi.org/10.1109/FMEC62297.2024.10710237.
3. Wang H., Liu L., Yue C., Wang L. et al. Cloud-native systems resilience assessments based on Kubernetes architecture graph. Service Oriented Computing and Applications, 2024. https://doi.org/10.1007/s11761-024-00406-x.
4. Yu S., Wu H., Niu X., Nie C. A Systematic Literature Review on Fault Injection Testing of Microservice Systems. IEEE Transactions on Services Computing, 2025, pp. 1-22. https://doi.org/10.1109/TSC.2025.3621564.
5. Zaccarini M., Borsatti D., Cerroni W., Foschini L., Grabarnik G. Ya., Manca L. et al. TELKA: Twin-Enhanced Learning for Kubernetes Applications. IEEE ISCC, 2024, pp. 1-6. https://doi.org/10.1109/ISCC61673.2024.10733736.
6. K3s Project. K3s – Lightweight Kubernetes. https://docs.k3s.io/ (accessed 01.09.2026).
7. Chaos Mesh Authors. Chaos Mesh Overview, version 2.8.4. https://chaos-mesh.org/docs/ (accessed 01.09.2026).
8. Figueira J., Coutinho C. Developing self-adaptive microservices. Procedia Computer Science, 2024, 232, 264-273. https://doi.org/10.1016/j.procs.2024.01.026.

Didusenko O. A., Bula Y. I., Shymchuk G. V.
Ternopil Ivan Puluj National Technical University, Department of Computer Science, Ternopil, Ukraine
Corresponding author: Grigorii Shymchuk, e-mail: gorych@gmail.com, ORCID: 0000-0003-2362-7386

REPRODUCIBLE FOSS TESTBED FOR FAULT-TOLERANT MULTIPATH ROUTING IN IP NETWORKS

Abstract. A reproducible open-source framework for studying fault-tolerant multipath routing in IP networks is proposed. The target testbed combines Mininet, Open vSwitch, FRRouting, iperf3, and Python so that topology, routing policy, failure injection, and metric collection can be reproduced without proprietary network equipment. A pilot model-based experiment with 50 failure episodes per strategy compared conventional single-path OSPF reconvergence, ECMP re-hashing, a precomputed backup path, and a health-aware multipath policy. The median modeled recovery time decreased from 2410.6 ms for the single-path baseline to 61.7 ms for health-aware multipath; mean packet loss decreased from 2.703% to 0.109%. The preliminary results do not substitute for packet-level emulation, but they define measurable hypotheses and a reproducible FOSS workflow for subsequent Mininet/FRRouting validation.

Keywords: open-source software; IP networks; multipath routing; fault tolerance; ECMP; FRRouting; Mininet; Open vSwitch.

Introduction

IP networks are expected to preserve connectivity despite link, interface, or router failures. Traditional shortest-path routing can restore reachability after control-plane reconvergence, but the interruption interval may still be relevant for interactive, industrial, and distributed services. The IP Fast Reroute framework therefore distinguishes mechanisms that precompute repair paths from ordinary post-failure convergence; equal-cost multipath (ECMP) and loop-free alternatives are among the basic repair options [1]. Resiliency mechanisms also emphasize path diversity and the use of disjoint paths so that a single failure does not affect all forwarding alternatives [2].

Recent research continues to treat multipath forwarding as both a resilience and traffic-engineering problem. Zhang and Geng showed that providing multiple routing choices can improve network availability [3], while programmable ECMP work demonstrated that conventional hash-based ECMP may be insufficient when a flow must be moved quickly away from a problematic path [4]. At the implementation level, FRRouting supports ECMP and weighted ECMP mechanisms, and Open vSwitch provides flow-based multipath actions [5, 6]. These components can be combined with Mininet, which runs real Linux networking and switch/application code in a virtual topology [7].

The objective of this work is to define a reproducible FOSS testbed and a quantitative evaluation procedure for comparing fault-recovery strategies in an IP network. The present contribution is deliberately positioned as a pilot stage: recovery metrics are first checked in a transparent stochastic model, after which the same scenarios are intended to be reproduced with real TCP/UDP flows in Mininet and FRRouting.

Materials and Methods

The planned software stack contains Mininet for topology orchestration, Open vSwitch for forwarding, FRRouting for OSPF/BGP control-plane functions, iperf3 for TCP/UDP traffic generation, and Python for experiment control and analysis. The current iperf3 project released version 3.22 on 29 September 2026, immediately before preparation of this work [8]. Figure 1 shows the compact topology used for the pilot design. Three internally separated paths connect R1 and R5, allowing link failures to be introduced without removing all end-to-end connectivity.

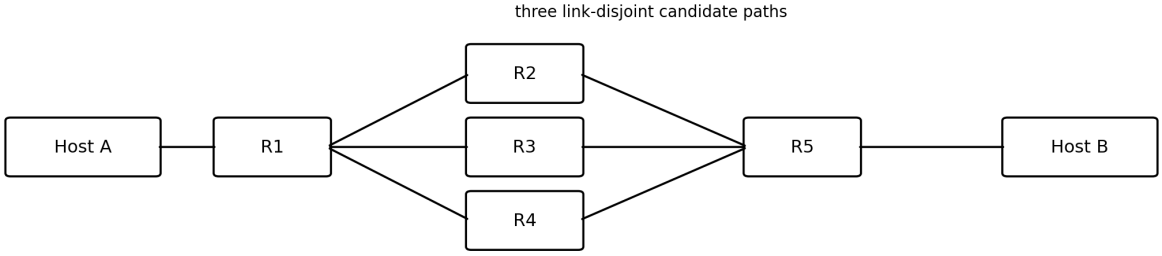


Figure 1 – Multipath topology for the reproducible FOSS testbed

Four strategies were compared: S1 – ordinary single-path forwarding followed by OSPF reconvergence; S2 – ECMP with flow re-hashing after a failed member is removed; S3 – a precomputed primary/backup path; and S4 – a health-aware multipath policy that ranks available next hops by recent reachability and delay. For the pilot experiment, 50 independent failure episodes were generated for each strategy with a fixed random seed. The model included failure-detection delay, route activation delay, transient packet loss, and post-recovery throughput retention. The numerical values are therefore model-based and are used to verify the comparison protocol rather than to claim production-network timing.

Table 1 – Failure scenarios and measured indicators

Scenario	Injected event	Primary metric	Secondary metrics
F1	Failure of one path link	Recovery time	Loss, throughput
F2	Failure near ingress router	Recovery time	Availability, RTT
F3	Failure under active load	Packet loss	Recovery, throughput
F4	Repeated path degradation	Stability	Switches, loss

Recovery time was defined as the interval between the injected failure and restoration of successful end-to-end forwarding:

$$T_{rec} = t_{restore} - t_{failure}. \quad (1)$$

Packet loss ratio was calculated as

$$PLR = \frac{N_{lost}}{N_{sent}} * 100\%. \quad (2)$$

For a 60-s observation window, service availability was estimated as $A = \frac{T_{total} - T_{down}}{T_{total}} * 100\%$, where T_{down} is the modeled outage interval. Throughput retention was the ratio of mean post-failure throughput to the no-failure baseline. Median and 95th-percentile recovery time were reported because recovery distributions are asymmetric.

Results

The pilot results are summarized in Table 2. All multipath strategies reduced the modeled interruption relative to ordinary single-path convergence. ECMP decreased median recovery time by approximately one order of magnitude, while precomputed and health-aware alternatives reduced the interruption further. The health-aware policy achieved a median recovery time of 61.7 ms and a 95th percentile of 96.0 ms. In the same model, mean packet loss was 0.109% and throughput retention was 98.36%.

Table 2 – Pilot model-based fault-recovery results

Method	Median Trec, ms	P95 Trec, ms	Loss, %	Availability, %	Throughput retention, %
OSPF single path	2410.6	2744.9	2.703	96.030	83.94
ECMP re-hash	303.1	445.3	0.538	99.479	94.00
Precomputed backup	102.7	138.3	0.182	99.836	97.35
Health-aware multipath	61.7	96.0	0.109	99.897	98.36

Discussion

The results support the experimental hypothesis that path diversity is useful only when an alternative can be activated quickly. Single-path routing preserves eventual reachability but depends on failure detection and control-plane convergence. ECMP already maintains several equal-cost next hops and can therefore shorten interruption, yet flow hashing does not by itself guarantee precise movement away from a degraded path. This limitation is consistent with recent work on programmable ECMP [4]. Precomputed backup and health-aware selection reduce the modeled recovery interval because the forwarding alternative is available before full reconvergence.

The main contribution is reproducibility rather than a claim about absolute timing. Every element of the planned testbed is open-source: Mininet can instantiate the topology, Open vSwitch can implement multipath forwarding, FRRouting can provide dynamic routing, iperf3 can generate measured TCP/UDP load, and Python can inject faults and aggregate results. The same

configuration files and random seeds can therefore be shared and repeated in teaching laboratories or research environments without specialized routers.

The limitation is that Table 2 is obtained from a controlled model, not from packet-level execution of FRRouting daemons. Absolute recovery values must not be transferred to production networks. The next validation stage should execute the four strategies under identical Mininet topologies, vary link delay and offered load, repeat each failure at least 30 times, and compare measured confidence intervals. Additional experiments should consider simultaneous failures and unequal-cost paths.

Conclusions

A reproducible FOSS methodology for evaluating fault-tolerant multipath routing was developed. The proposed stack combines Mininet, Open vSwitch, FRRouting, iperf3, and Python and defines common failure scenarios and metrics. In the pilot model, moving from ordinary single-path convergence to health-aware multipath reduced median recovery time from 2410.6 ms to 61.7 ms and packet loss from 2.703% to 0.109%. These values are preliminary, but they provide explicit hypotheses for subsequent emulation.

Future work will replace the stochastic timing model with automated Mininet/FRRouting experiments, evaluate path-disjointness under several topologies, and study adaptive weighting that jointly considers reachability, delay, and link utilization.

References

1. Shand M., Bryant S. IP Fast Reroute Framework. RFC 5714, 2010. https://datatracker.ietf.org/doc/html/rfc5714.
2. Filsfils C. et al. Resiliency Use Cases in Source Packet Routing in Networking (SPRING) Networks. RFC 8355, 2018. https://datatracker.ietf.org/doc/html/rfc8355.
3. Zhang W., Geng H. Improving Network Availability through Optimized Multipath Routing and Incremental Deployment Strategies. Computers, Materials & Continua, 2024, 80(1), 427-448. https://doi.org/10.32604/cmc.2024.051871.
4. Liu Y., Xiao Y., Zhang X., Dang W., Liu H., Li X., He Z., Wang J., Kuzmanovic A., Chen A., Miao C. Unlocking ECMP Programmability for Precise Traffic Control. 22nd USENIX NSDI, 2025, pp. 87-106.
5. FRRouting Project. BGP Documentation: Weighted ECMP Using BGP Link Bandwidth. FRRouting 10.2 documentation. https://docs.frrouting.org/en/stable-10.2/bgp.html (accessed 01.09.2026).
6. Open vSwitch Project. Open vSwitch Documentation: Multipath Action. https://docs.openvswitch.org/ (accessed 01.09.2026).
7. Mininet Project. Mininet: An Instant Virtual Network on Your Laptop. https://mininet.org/ (accessed 01.09.2026).
8. ESnet. iperf3 Project News: iperf 3.22 Released, 29 September 2026. https://software.es.net/iperf/news.html (accessed 01.09.2026).

Ebirim V., Jadama A. F. R.
Fuzion Mobile Technologies (FMT), The Gambia

RESILIENT DIGITAL EDUCATION IN DISRUPTED ENVIRONMENTS: THE CCDS APPROACH

Abstract. Digital education has expanded rapidly, but much of its infrastructure remains dependent on continuous internet connectivity, reliable electricity, functioning schools, appropriate devices and stable communication networks. These assumptions become particularly problematic during conflict, displacement, public emergencies and in communities where connectivity remains limited. The challenge is therefore not simply how to digitise education, but how to design digital education systems that remain useful when the conditions supporting conventional digital delivery fail.

This paper examines the Curriculum Content Delivery System (CCDS) as an approach to resilient digital education infrastructure. It adopts a practice-based qualitative and architectural analysis of CCDS development and the iLearn Gambia implementation context, supported by UNESCO and World Bank evidence on digital learning and continuity of education. The analysis considers connectivity dependency, continuity of curriculum access, architectural redundancy and human agency. The Gambia provides the principal implementation context, while international evidence provides comparative grounding for offline delivery.

The central proposition is that connectivity should accelerate digital education - not determine whether digital education is possible.

Keywords: digital education, educational resilience, offline learning, CCDS, curriculum delivery, connectivity, learning continuity, artificial intelligence

1. Introduction

Digital transformation has changed how educational content can be created, distributed and accessed. Learning platforms, mobile applications, digital textbooks, video lessons, virtual classrooms and artificial intelligence increasingly extend education beyond the physical classroom. Yet digital education can become fragile when its delivery architecture assumes that learners are continuously connected.

This dependency is particularly significant in disrupted environments. Conflict can damage schools and communication infrastructure; displacement can separate learners from institutions; emergencies can interrupt normal teaching; and rural or economically disadvantaged communities may remain outside reliable broadband networks. The disruption of physical education therefore exposes a second vulnerability: the assumption that digital education is synonymous with being online.

A resilient approach requires a different starting point. The essential question is not whether a learner has permanent connectivity, but whether the learner can continue to access the curriculum when normal delivery conditions are interrupted. UNESCO's evidence cautions against treating technology as a universal solution and emphasizes appropriate, equitable and sustainable use according to educational context (UNESCO, 2023).

2. Methodological Context

The study adopts a practice-based qualitative and architectural analysis. The principal evidence comprises documentation relating to the CCDS architecture and iLearn Gambia, supplemented by UNESCO documentation of the Strengthening iLearn Gambia Platform project, reports on teacher capacity development, and World Bank evidence concerning continuity of learning during the COVID-19 disruption in The Gambia. International literature on offline digital learning provides comparative context.

The analysis focuses on four dimensions: connectivity dependency, continuity of curriculum access, architectural redundancy and human agency. The sources are treated as implementation, policy and architectural evidence rather than controlled experimental data. Consequently, the paper does not claim causal learning effects. Its purpose is to examine an architectural proposition and identify principles that warrant further empirical investigation.

3. From Connectivity Dependency to Continuity-by-Design

Conventional digital learning systems generally assume a chain of dependencies: a functioning network, electricity, an internet-enabled device, a platform, authentication and access to remotely hosted content. When one or more elements fail, access to learning may also fail.

Continuity-by-design reverses this logic. Resilience is treated as an architectural characteristic rather than an emergency response added after disruption occurs. Online connectivity remains important, but becomes one delivery layer rather than the single condition upon which digital learning depends.

The principle can therefore be expressed simply:

Connectivity should accelerate digital education - not determine whether digital education is possible.

Under this model, the same curriculum can exist across connected and disconnected environments. Online users can receive updated content, participate in communication and generate learning data. Offline users can access locally stored curriculum resources, lessons, textbooks and assessments without requiring continuous internet access. Synchronisation can occur when connectivity becomes available.

4. The CCDS Architecture and Evidence for Offline Delivery

The Curriculum Content Delivery System (CCDS) is designed around curriculum content as the central unit of digital education delivery. Rather than treating a learning management platform as the entire educational system, the architecture connects curriculum digitisation, content management, distribution, learner access, teacher support and monitoring.

International experience provides a basis for this approach. UNESCO has documented offline digital-learning models in which removable media, including SD cards, are used to make educational resources available where internet access is constrained. UNESCO-UNEVOC has similarly documented offline learning initiatives using locally stored digital resources and SD-card-based access. The World Bank's remote-learning guidance identifies offline repositories, portable digital media and pre-installed content as mechanisms for reaching learners in low-connectivity environments (UNESCO, 2018; UNESCO-UNEVOC; World Bank, 2021/2023).

The Gambia provides a particularly relevant national context. During the COVID-19 school closures, the education system adopted multiple approaches to continuity of learning, including radio and television programming and an online repository for lessons. The World Bank's Gambia Emergency Education COVID-19 Response explicitly framed continued access to educational services and resilient learning environments as core objectives (World Bank, 2022).

The subsequent development of iLearn Gambia introduced a more explicit online/offline digital-learning architecture. UNESCO's Project 3240115042, Strengthening iLearn Gambia Platform, implemented from 2024 to 2025, identifies the platform as a digital tool for expanding knowledge sharing and skills development. The project documentation records the use of SD cards and other digital resources to extend access to learning materials where internet connectivity is limited (UNESCO, 2025).

Teacher capacity development formed part of this process. In January 2025, the Directorate of Science and Technology of the Ministry of Basic and Secondary Education conducted a five-day Training of Trainers programme supported through the UNESCO Participation Programme and UNESCO-NATCOM. The training addressed both the online and offline versions of iLearn Gambia, including its SD-card-based delivery model, specifically in relation to extending learning access where connectivity is limited (Foroyaa, 2025).

The significance of these developments is architectural. The SD card is not the digital education system itself. It functions as a portable curriculum-delivery layer within a hybrid system: online infrastructure supports content development, updating, administration and monitoring, while offline resources preserve access at the point of learning when connectivity is unavailable or intermittent.

5. Curriculum Content, Equity and Human Agency

A resilient digital education system requires more than a platform. It requires structured, reusable and distributable curriculum content. The CCDS approach therefore places the curriculum at the centre of the architecture, allowing digital textbooks, lessons, multimedia resources and assessments to be organised according to curriculum structures and distributed through different channels.

The Gambia experience demonstrates the practical value of this principle across different stages of digital education development: emergency remote learning during COVID-19, development of an online/offline learning platform, teacher capacity development and offline curriculum distribution. The evidence does not establish a universal causal effect of SD-card learning; it establishes an operational context in which offline content can form part of a broader digital-learning architecture.

Equity remains central. Offline capability can reduce dependence on continuous connectivity, but it does not eliminate barriers related to devices, electricity, accessible content, teacher capacity, maintenance, language and affordability. Technology must therefore be selected according to local conditions rather than assumed to be universally appropriate.

Teachers remain essential to interpretation, facilitation, assessment and learner support. Digital infrastructure extends their reach and provides additional resources; it does not remove the educational relationship between teacher and learner. The Gambia's teacher-training experience reinforces this principle: infrastructure development and human capacity development must proceed together.

6. Artificial Intelligence as an Intelligence Layer

Artificial intelligence introduces another dimension to resilient digital education. AI can support content adaptation, learner assistance, translation, search, feedback and teacher-facing tools. However, its role should be considered carefully within a resilience architecture.

AI should function as an intelligence layer, not as the foundation of education continuity. If access to an AI service requires permanent connectivity, dependence on that service may reproduce the same vulnerability that offline architecture seeks to address. Core curriculum resources should therefore remain accessible when AI services, cloud platforms or internet connectivity are unavailable.

7. Discussion and Conclusion

The combined evidence indicates that offline digital delivery is not a rejection of digital transformation. It is a recognized response to unequal connectivity and a component of broader hybrid and multi-channel learning strategies.

The CCDS approach extends this principle by making the curriculum itself portable across delivery environments. Online infrastructure provides coordination, updating, administration and monitoring; offline infrastructure provides local access when connectivity cannot be assumed. The resulting redundancy changes the question from whether a learner is continuously connected to whether the education system has another reliable route through which the curriculum can reach that learner.

The Gambia is particularly instructive because the continuity challenge has been addressed through successive layers: broadcast and printed learning during COVID-19, digital platforms, teacher capacity development and offline SD-card-based curriculum delivery. UNESCO and World Bank documentation provides institutional evidence of this evolution, while the iLearn Gambia project demonstrates how offline access can become part of a structured digital-learning architecture.

This principle is increasingly relevant to environments affected by conflict and prolonged educational disruption. Recent UNICEF reporting on Ukraine illustrates the continuing scale of disruption to children's education and the persistence of barriers to classroom-based learning (UNICEF, 2026). Such contexts reinforce the importance of designing continuity mechanisms before conventional delivery systems fail.

Educational resilience does not depend on eliminating disruption. It depends on designing systems capable of continuing through it.

A resilient digital education system does not ask only:

Is the learner online?

It asks:

Can the learner still reach the curriculum?

That is the foundation of continuity-by-design.

References

1. Foroyaa. (2025, January 13). MoBSE Trains Trainers on Utilizing ‘I Learn’ Gambia Platform.
2. Fuzion Mobile Technologies. (2026). CCDS Technical and Digital Education Architecture. Internal technical documentation.
3. Fuzion Mobile Technologies. (2026). iLearn Gambia: Offline and Hybrid Digital Learning Implementation Context. Internal documentation.
4. UNESCO. (2018). Digital Inclusion for Low-skilled and Low-literate People: A Landscape Review. UNESCO-Pearson Initiative for Literacy.
5. UNESCO. (2023). Global Education Monitoring Report 2023: Technology in Education: A Tool on Whose Terms? Paris: UNESCO.
6. UNESCO. (2025). Project 3240115042: Strengthening iLearn Gambia Platform, a digital tool for fostering knowledge sharing and skills development: Revolutionizing Education Access. UNESCO Core Data Portal.
7. UNESCO-UNEVOC. Offline digital learning and skills development: Bibliothèques Sans Frontières.
8. World Bank. (2021). The Gambia Digital Economy Diagnostic. Washington, DC: World Bank.
9. World Bank. (2022). Implementation Completion and Results Report: Gambia Emergency Education COVID-19 Response (P174035). Washington, DC: World Bank.
10. World Bank. (2023). EdTech Toolkit for Remote Learning. Washington, DC: World Bank.
11. UNICEF Ukraine. (2026). Children in Ukraine face a fifth year of assault on their right to learn. UNICEF.

Fraya
Safi Institute of Advanced Study, Calicut (affiliated to University of Calicut), Kerala, India
frayapkl@gmail.com

DIGITAL TRANSFORMATION OF BUSINESS THROUGH FREE AND OPEN SOURCE SOFTWARE: OPPORTUNITIES, BARRIERS, AND STRATEGIC PATHWAYS

Abstract. Digital transformation has become a strategic imperative for organizations seeking resilience, efficiency, and competitiveness. Free and Open Source Software (FOSS) offers a practical foundation for this process, lowering licensing costs, avoiding vendor lock-in, and enabling customization that proprietary alternatives rarely permit. This paper examines how businesses, particularly small and medium-sized enterprises, can use FOSS solutions to modernize core operations such as data management, communication, customer relationship management, and infrastructure. It identifies the principal barriers to adoption — including perceived support risk, integration complexity, and internal skills gaps — and proposes a staged adoption framework that balances cost savings against organizational readiness. The analysis concludes that a deliberate, phased approach to FOSS adoption, supported by community engagement and internal capability building, offers a sustainable pathway to digital transformation without the financial and contractual constraints typical of proprietary ecosystems.

Keywords: digital transformation, open source software, FOSS, business strategy, SME, cost efficiency, vendor lock-in.

1. Introduction

Digital transformation refers to the integration of digital technology into all areas of an organization, fundamentally changing how value is created and delivered to customers. For many enterprises, especially small and medium-sized businesses operating under tight budget constraints, the cost of proprietary software licensing and vendor-controlled infrastructure represents a significant barrier to this transformation. Free and Open Source Software has matured over the past two decades into a viable, often superior, alternative across nearly every category of business software, from operating systems and databases to enterprise resource planning and collaboration tools.

The purpose of this paper is to examine the specific role FOSS can play as an enabler of digital transformation in a business context, to identify the barriers that continue to slow adoption despite clear economic advantages, and to propose a practical, staged framework that organizations can follow to adopt open source tools responsibly.

2. FOSS as an Enabler of Digital Transformation

Open source solutions contribute to digital transformation along several dimensions. First, cost efficiency: the elimination or reduction of per-seat licensing fees frees capital for other transformation initiatives, such as staff training or process redesign. Second, flexibility: access to source code allows organizations to adapt software to their exact workflows rather than reshaping processes to fit rigid proprietary products. Third, interoperability: open standards and open file formats reduce the risk of data being trapped in a single vendor's ecosystem, a critical concern as businesses increasingly rely on data portability across cloud and on-premises systems. Fourth, security transparency: publicly auditable code allows independent verification of security practices, which is increasingly valued in regulated industries.

Common domains where FOSS already substitutes effectively for proprietary tools include operating systems and virtualization (Linux, KVM, Proxmox), databases (PostgreSQL, MariaDB), content and knowledge management (WordPress, Nextcloud), communication and collaboration (Matrix, Jitsi), and business intelligence and analytics (Metabase, Apache Superset). Each of these can serve as a building block in a broader digital transformation roadmap.

3. Barriers to Adoption

Despite the advantages outlined above, adoption of FOSS in business settings remains uneven. Several recurring barriers can be identified. Perceived support risk: decision-makers often equate the absence of a vendor contract with the absence of reliable support, even though commercial

support subscriptions exist for most major open source projects. Integration complexity: legacy systems built around proprietary formats or APIs can make migration technically demanding, requiring investment before savings materialize. Internal skills gaps: many organizations lack staff experienced in configuring, securing, and maintaining open source infrastructure, creating a dependency on external consultants during the transition period. Organizational inertia and risk aversion: procurement processes in many companies are structured around named vendors and service-level agreements, which can disadvantage community-driven projects regardless of technical merit.

4. A Staged Adoption Framework

To translate the advantages of FOSS into practical transformation outcomes while managing the barriers above, this paper proposes a three-stage framework. Stage one, low-risk substitution, targets non-critical, well-supported tools — such as replacing office productivity suites or internal communication platforms — to build organizational familiarity and confidence with minimal operational risk. Stage two, infrastructure modernization, extends adoption to servers, databases, and virtualization layers, typically supported by a commercial support subscription to offset internal skills gaps during the transition. Stage three, strategic differentiation, applies FOSS to core, customer-facing systems where the ability to modify source code becomes a genuine competitive advantage rather than merely a cost-saving measure. Progressing through these stages allows an organization to build internal capability incrementally, rather than attempting a disruptive, all-at-once migration.

5. Conclusion

Free and Open Source Software provides a credible and increasingly mainstream foundation for business digital transformation, offering cost savings, flexibility, and freedom from vendor lock-in. The barriers to adoption are real but manageable, particularly when organizations adopt a staged approach that matches the pace of migration to their internal readiness and risk tolerance. Future research should examine sector-specific adoption patterns and quantify the long-term total cost of ownership of FOSS-based digital transformation compared to proprietary alternatives.

References

1. Fitzgerald, B. (2006). The transformation of open source software. MIS Quarterly, 30(3), 587–598.
2. Gonzalez-Barahona, J. M., Robles, G., & Herraiz, I. (2020). Open Source Software: Concepts, Business Models, and Public Policy Implications. Springer.
3. West, J., & Gallagher, S. (2006). Challenges of open innovation: the paradox of firm investment in open-source software. R&D Management, 36(3), 319–331.
4. European Commission. (2023). Open source software strategy 2023–2027. Publications Office of the European Union.

Kala P.
Percona, Durham, North Carolina, United States
Corresponding author: Puneet Kala, e-mail: speakering+puneet@percona.com

RUN MONGODB YOUR WAY: A PRACTICAL OPEN-SOURCE STACK FOR SELF-MANAGED DEPLOYMENTS

Abstract. Organizations adopting self-managed MongoDB must address security, backup and recovery, migration, and modern search capabilities without relying on a managed database service. This work presents and evaluates an open-source reference architecture based on Percona Server for MongoDB, Percona Backup for MongoDB, Percona ClusterSync for MongoDB, and Percona Search for MongoDB. The architecture is examined through four operational requirements: enterprise security, reliable backup and recovery, low-downtime migration, and vector search for AI-oriented applications. The study analyzes how each component contributes to the deployment lifecycle, as well as its operational requirements and limitations. The resulting architecture demonstrates how organizations can assemble a self-managed MongoDB platform while retaining control over infrastructure, data, security, and technology choices. A practical deployment and migration framework is proposed for teams evaluating self-managed MongoDB environments.

Keywords: MongoDB; open-source software; self-managed databases; database security; backup and recovery; database migration; vector search; Percona Server for MongoDB; database architecture

Introduction

Managed database services simplify infrastructure operations but can also influence where data is stored, how security is configured, which infrastructure providers can be used, and which surrounding technologies are available.

Organizations may therefore choose to operate MongoDB on their own infrastructure to maintain greater control over data, encryption keys, identity management, storage, deployment environments, and application architecture. This flexibility introduces additional operational responsibilities.

A production-ready self-managed MongoDB environment must address several fundamental requirements: enterprise security, reliable backup and recovery, migration of existing data, and increasingly, search and vector capabilities required by AI-enabled applications.

This work investigates how these requirements can be addressed through an open-source stack consisting of Percona Server for MongoDB, Percona Backup for MongoDB, Percona ClusterSync for MongoDB, and Percona Search for MongoDB.

The objective is to develop a practical reference architecture and decision framework for organizations considering a transition from MongoDB Community or managed MongoDB services to a self-managed environment.

Materials and Methods

The proposed architecture consists of four primary open-source components, each addressing a different operational requirement.

Percona Server for MongoDB (PSMDB) provides the primary database layer. The study considers enterprise-oriented security capabilities such as LDAP integration, encryption at rest, and auditing.

Percona Backup for MongoDB (PBM) provides backup and recovery capabilities. The evaluation considers backup configuration, storage, restoration procedures, and methods for verifying that stored backups can successfully support recovery.

Percona ClusterSync for MongoDB (PCSM) provides data migration capabilities. The migration workflow combines initial data cloning with continued synchronization of changes from the source environment, allowing the source database to continue serving application traffic during most of the migration process.

Percona Search for MongoDB provides search functionality, including capabilities relevant to vector-based and AI-oriented applications, without requiring the database itself to be operated as a managed service.

The components are evaluated as parts of a complete deployment lifecycle:
– establish a secure MongoDB foundation;
– configure and validate backup and recovery;
– prepare and execute migration from the existing environment;
– validate the target environment and perform cutover;
– introduce search or vector-search capabilities where required.

Results

The proposed architecture demonstrates that several capabilities commonly associated with managed MongoDB environments can be assembled using independently deployable open-source components.

Percona Server for MongoDB provides the database foundation and security mechanisms required for operating MongoDB within infrastructure controlled by the organization.

Percona Backup for MongoDB addresses data protection by providing mechanisms for creating and restoring MongoDB backups. Importantly, the architecture treats restoration testing as part of the backup strategy rather than considering successful backup creation alone sufficient evidence of recoverability.

Percona ClusterSync for MongoDB addresses the transition from an existing MongoDB environment. By combining data cloning and continued synchronization, most migration activity can occur while the source remains operational, reducing the duration of the final cutover period.

Percona Search for MongoDB extends the architecture with search functionality applicable to conventional search use cases as well as vector-based application patterns.

Together, these components form a reference architecture covering the principal stages of a self-managed MongoDB deployment: secure operation, data protection, migration, and search.

Discussion

Moving from a managed MongoDB service to a self-managed architecture changes the distribution of operational responsibility.

Greater control over infrastructure, data, security configuration, storage, and technology selection is accompanied by responsibility for configuration, monitoring, backups, recovery testing, upgrades, migrations, availability, and troubleshooting.

For this reason, the decision between managed and self-managed MongoDB should not be reduced to infrastructure cost or software functionality alone. Operational expertise and organizational capacity are important architectural considerations.

The proposed stack addresses this challenge through specialized components rather than attempting to provide every capability within the database server itself. This modularity allows organizations to select and operate components according to their requirements, but it also introduces integration and lifecycle-management responsibilities.

Backup and recovery require particular attention. The existence of backups does not guarantee recoverability; organizations should regularly validate restoration procedures and recovery objectives.

Migration similarly requires preparation and monitoring. Continuous synchronization can substantially reduce disruption, but the final transition still requires carefully defined cutover criteria and validation.

Finally, vector search introduces additional workload and infrastructure considerations. Teams should evaluate whether self-managed search capabilities provide sufficient performance and operational characteristics for their specific AI or retrieval workloads.

Conclusions

This work presents a practical open-source reference architecture for organizations seeking to operate MongoDB on their own infrastructure.

The architecture combines Percona Server for MongoDB for the core database and security layer, Percona Backup for MongoDB for data protection, Percona ClusterSync for MongoDB for migration, and Percona Search for MongoDB for search and vector-oriented workloads.

The main contribution is not an individual technology but an integrated decision and deployment framework covering four fundamental questions of self-managed MongoDB operation: how to secure the database, protect and recover data, migrate existing workloads with minimal disruption, and support modern search requirements.

The proposed framework can serve as a starting point for organizations migrating from managed services, operating MongoDB Community deployments, or developing AI applications that require greater control over their database and search infrastructure.

Future work may include quantitative evaluation of migration performance, backup and recovery times, operational overhead, vector-search performance, and comparative analysis between self-managed and managed MongoDB architectures.

References

1. Percona. Percona Server for MongoDB Documentation. Percona Software Documentation.
2. Percona. Percona Server for MongoDB. Open-source project repository.
3. Percona. Percona Backup for MongoDB Documentation. Percona Software Documentation.
4. Percona. Percona Backup for MongoDB. Open-source project repository.
5. Percona. Percona ClusterSync for MongoDB Documentation. Percona Software Documentation.
6. Percona. Percona ClusterSync for MongoDB. Open-source project repository.
7. Percona. Percona Search for MongoDB Documentation. Percona Software Documentation.
8. MongoDB Inc. MongoDB Documentation. MongoDB Inc.
9. MongoDB Inc. MongoDB Security Documentation. MongoDB Inc.

Lukin O.
Percona, Ukraine
Corresponding author: Oleksiy Lukin, e-mail: oleksiy.luikin@percona.com, ORCID: 0009-0006-2030-680X

A MODERN C++ KMIP CLIENT BASED ON A REUSABLE PROTOCOL LIBRARY

Abstract. The Key Management Interoperability Protocol (KMIP) provides a standardized interface between applications and key management systems (KMS). This paper presents kmipclient, an open-source native C++ KMIP client library developed at Percona, together with kmipcore, the underlying KMIP protocol library. The project was motivated by the practical difficulty of extending an existing C-based implementation through a separate C++ wrapper. The new architecture separates the KMIP protocol model from networking and client functionality, making the protocol implementation independently testable and easier to extend. The library supports KMIP 1.4 and 2.0, including protocol negotiation, and is tested against five independent KMIP server implementations. It is already used by Percona Server for MySQL while remaining a general-purpose KMIP client library. The paper also describes AI-assisted development with human-controlled architecture and review.

Keywords: KMIP; key management; C++; KMS; interoperability; open source software; cryptography

Introduction

Encryption protects sensitive data only when the cryptographic keys are adequately protected as well. Keeping keys under a separate security and administrative boundary enables controlled access, key rotation, auditing, and lifecycle management independently of the systems storing encrypted data. Key Management Systems provide this functionality, but different products can expose different interfaces. The Key Management Interoperability Protocol (KMIP), developed by the OASIS KMIP Technical Committee, defines a standardized protocol for communication between key management clients and servers.

The kmipclient project was started to provide a modern, native C++ implementation with a clean API and an architecture that would make further KMIP development easier. The author was already maintaining and extending an existing C-based KMIP implementation used by Percona. The implementation was functional, but adding protocol operations through its C data model and C++ wrapper could require changes in several layers. For example, adding the KMIP Activate and Deactivate operations required implementing functionality in the C library and then adding the corresponding C++ wrapper.

The goal of the new project was therefore not to replace the existing implementation because it was inadequate, but to establish a native C++ architecture in which the protocol model, client functionality, and networking could evolve independently. The practical objective was to make implementing the next required part of the KMIP specification less costly than implementing the previous one.

Materials and Methods

The implementation is organized into two principal libraries. kmipcore is the KMIP protocol library. It contains the KMIP data model, protocol objects, requests and responses, together with TTLV serialization and parsing. It has no dependency on networking or on a particular KMS implementation. kmipclient is the KMIP client library built on top of kmipcore. It provides the higher-level API used by applications, communication with KMS servers, a replaceable NetClient networking abstraction, and connection pooling.

The resulting architecture can be summarized as: Application → kmipclient → kmipcore → KMIP/TTLV → KMS. The separation allows kmipcore to be tested independently of networking and also means that it is not inherently client-specific; the protocol library can serve as a foundation for other KMIP applications, including a future server implementation.

The implementation uses standard C++ mechanisms including RAII, smart pointers, move semantics, and STL facilities for resource management and ownership. The KMIP specification is treated as the primary source of truth. The library supports KMIP 1.4 and 2.0, with protocol negotiation and explicit protocol-version selection.

Testing uses two complementary levels. Unit tests cover the protocol and data model, including serialization, parsing, and specification-conformance behavior. Integration tests exercise the client against five KMIP server implementations: PyKMIP, Cosmian KMS, HashiCorp Vault Enterprise, Fortanix Data Security Manager, and Thales CipherTrust Manager. Every kmipclient release is tested against all five implementations.

Results

The resulting library provides a native C++ KMIP client API while keeping the protocol implementation independent from networking. This architecture reduces the coupling involved in adding protocol functionality and allows protocol components to be tested without a running KMS. The high-level client API can be used by applications while lower-level protocol functionality remains available for specialized operations.

The interoperability test matrix has demonstrated operation with five independent KMIP implementations. This provides a practical complement to specification-based unit testing: the specification defines the required protocol behavior, while independent servers expose implementation differences and compatibility issues that may not appear in isolated tests.

The library has been integrated into Percona Server for MySQL for KMIP-related functionality. At the same time, kmipclient remains a general-purpose library and is not coupled to MySQL-specific functionality. The project has also received contributions and fixes from other Percona engineers. Saikumar VS provided extensive QA support for KMIP functionality, including testing of both the C++ library and the MySQL KMIP keyring implementation.

Discussion

The main architectural trade-off is that the project does not attempt to implement the complete KMIP specification immediately. KMIP is a large protocol, and complete coverage is a substantial undertaking. Instead, the architecture is intended to make incremental implementation practical. Separating kmipcore from kmipclient means that new protocol objects and operations can be developed without introducing corresponding dependencies on the networking layer.

Interoperability testing also has an inherent limitation: successful communication with one KMS does not prove complete conformance to the specification. A server can tolerate input that is accepted by its implementation but does not represent the full protocol requirement. For this reason, specification-driven unit testing remains the primary correctness criterion, while testing against independent servers is used as an additional interoperability and regression check.

AI-assisted development was used extensively for implementation, particularly with GitHub Copilot. The development process was deliberately incremental. First, the KMIP specifications were supplied as the protocol source of truth; the existing C library was then supplied as practical context; finally, a hand-written prototype containing the new API, interfaces, and implementation skeletons was supplied. AI was subsequently used to implement relatively small pieces of the system, including repetitive protocol and serialization code and tests.

AI was also used as an additional review mechanism. Przemyslaw Skibinski reviewed the implementation using several AI models, providing alternative perspectives. Yura Sorokin reviewed the complete library from a human engineering perspective, concentrating on architecture, API, usability, and maintainability. This experience suggests that AI assistance is particularly useful when architectural boundaries and engineering requirements are established before implementation. AI performed a substantial amount of implementation work, while the architecture and engineering decisions remained under human control.

Conclusions

kmipclient provides an open-source native C++ interface to KMIP, built on kmipcore, a reusable implementation of the KMIP protocol itself. The principal contribution is the separation between the protocol model and client functionality, which makes the implementation independently testable and provides a cleaner path for future protocol extensions. Support for KMIP 1.4 and 2.0, automated testing against five independent KMIP implementations, and production use in Percona Server for MySQL provide practical validation of the approach. Future work is focused

on extending protocol coverage while preserving the existing architectural boundaries and interoperability test framework.

Acknowledgements

The author thanks Saikumar VS, Zsolt Parragi (dutow), Przemyslaw Skibinski, Yura Sorokin, and other Percona engineers who contributed testing, reviews, fixes, and improvements.

References

1. OASIS. Key Management Interoperability Protocol (KMIP) Technical Committee. https://www.oasis-open.org/committees/kmip/
2. OASIS. Key Management Interoperability Protocol Specification. https://docs.oasis-open.org/kmip/
3. Percona. kmipclient — Open-source C++ KMIP client library. https://github.com/percona/kmipclient

Mekhed D. B.
Chernihiv Polytechnic National University, Department of Cybersecurity and Mathematical Modeling, Chernihiv, Ukraine
Corresponding author: Dmytro Mekhed, e-mail: d.mekhed@stu.cn.ua, ORCID: 0000-0003-3905-3620

DEVELOPING STUDENTS' CYBER HYGIENE SKILLS THROUGH THE USE OF FREE SPECIALIZED SOFTWARE

Abstract. Students are among the most active and, at the same time, most vulnerable Internet users, yet cyber hygiene is often taught through declarative lectures that rarely change behaviour. This paper proposes a three-level training module in which students acquire protective skills through hands-on work with free/libre and open-source software (FOSS): password management with KeePassXC, private web browsing with Firefox, uBlock Origin and Tor Browser, and secure operating environments with Ubuntu and Tails. For each level, the paper defines practical tasks, expected learning outcomes and an evaluation design based on pre- and post-training assessment. The module requires no licensing costs, and the openness of the source code allows students to understand how protection mechanisms work rather than merely follow rules. The approach can be integrated into information security and digital literacy courses.

Keywords: cyber hygiene; free software; open source; password manager; privacy; Linux; digital literacy; education.

Introduction

Students use dozens of online services every day, including e-mail, social networks, learning platforms and online banking, and often neglect basic security practices such as unique passwords or careful handling of links. In Ukraine this problem is especially pressing. During the full-scale war the country has been under intense cyber pressure: in the first half of 2025 alone, the national CERT-UA team recorded 3,018 cyber incidents, with phishing being the most common incident type (27%) [1].

At the same time, cyber hygiene training is frequently reduced to declarative advice such as “do not click suspicious links” or “create a strong password”. Without practical skills, such knowledge is quickly forgotten and rarely changes user behaviour.

Free software offers a practical alternative. It allows students to use real protection tools legally and at no cost. Because the source code is open, these tools can be independently audited by the community for vulnerabilities, hidden functionality or tracking mechanisms [2]. This substantially reduces the risk of hidden threats and lets learners look inside the protection mechanisms they use.

The aim of this work is to substantiate and design a training module that develops students’ cyber hygiene skills through the practical use of free specialized software on three levels of protection.

Materials and Methods

The module is intended for undergraduate students of any specialty with basic computer literacy; for cybersecurity students it can be extended with source-code analysis tasks. It consists of three consecutive levels, moving from the protection of individual accounts to the protection of the whole working environment.

Software (stable versions as of September 2026): KeePassXC 2.7.12 with the KeePassXC-Browser extension [3]; Mozilla Firefox ESR 153 with the uBlock Origin 1.75 extension [4]; Tor Browser 15.0 [5]; Ubuntu 26.04.1 LTS [6]; Tails 7.12 [7]. The Extended Support Release of Firefox is chosen to keep the laboratory environment stable between classes.

Rationale for tool selection. KeePassXC stores credentials locally in an encrypted KDBX database (AES-256, Twofish or ChaCha20), unlocked by a single master passphrase, so the user controls where the data is kept [3]. uBlock Origin blocks advertising, tracking and known malicious domains by default [4]. Tor Browser relays traffic through a chain of three volunteer-run relays with layered encryption and is designed to resist browser fingerprinting [5]. Ubuntu demonstrates privilege separation between regular and administrative accounts, mandatory access control via AppArmor and cryptographically signed package repositories [6]. Tails boots from a USB drive,

routes all network traffic through Tor and, by default, keeps no data after shutdown unless encrypted persistent storage is explicitly enabled [7].

Environment: a computer laboratory with Ubuntu installed natively or in a virtual machine (for example, VirtualBox or GNOME Boxes), with students working under unprivileged accounts, and USB drives of at least 8 GB for Tails. Step-by-step instructions can be adapted from the Surveillance Self-Defense guides of the Electronic Frontier Foundation [8].

Evaluation design: a questionnaire on security habits (password reuse, use of a password manager, browser privacy settings, update practices) administered before and after training; a practical skills checklist for each level; and a delayed follow-up one to two months later to check whether the practices persist.

Results

The module has not yet been piloted; this section therefore presents its designed structure and the expected learning outcomes for each level (Table 1).

Table 1 – Structure of the three-level training module

Level	FOSS tools	Practical tasks	Expected learning outcome
A. Password management	KeePassXC, KeePassXC-Browser	Create an encrypted database protected by a master passphrase; generate a unique password for each account; configure browser auto-fill; make a backup copy of the database	Stops reusing passwords; understands the difference between local encrypted storage and cloud storage
B. Safe web browsing	Firefox ESR, uBlock Origin, Tor Browser	Configure Enhanced Tracking Protection and HTTPS-Only Mode; review site permissions; install uBlock Origin and inspect blocked trackers; examine a Tor circuit	Recognises common tracking techniques; can configure a privacy-respecting browser; understands onion routing and its limits
C. Secure operating environment	Ubuntu, Tails	Work under an unprivileged account and use sudo; install software only from signed repositories; enable automatic security updates; boot Tails from USB and confirm that no data persists after shutdown	Understands privilege separation and the software supply chain; understands the concept of an amnesic (temporary) environment

The expected overall effect is a shift from declarative knowledge to routine protective behaviour: fewer reused passwords, adoption of a password manager, and consistent use of browser privacy settings and security updates. These outcomes are to be verified in a planned pilot using the evaluation design described above.

Discussion

Compared with lecture-based training, the proposed module moves the emphasis from rules to practice. Its main advantages are the absence of licensing costs, the possibility of legally installing the same tools on students' own devices, which helps transfer skills into everyday life, and the openness of the code, which allows the protection mechanisms themselves to be studied.

The approach also has limitations. First, it has not yet been validated empirically. Second, none of the tools is a universal safeguard: no browser prevents phishing if the user voluntarily enters credentials on a fake site, so the module should be combined with phishing-awareness exercises. Linux reduces but does not eliminate the risk of malware, and the security of a password manager depends on the strength of the master passphrase and the integrity of the device. Third, Tor Browser and Tails involve usability trade-offs, and their use should be accompanied by a discussion of ethical and legal aspects. Finally, the full-featured version of uBlock Origin depends on extension capabilities that Chromium-based browsers have been phasing out, which is an additional reason to base the module on Firefox.

Further work includes piloting the module with students, analysing the results of the pre- and post-training assessment and extending the module with topics such as two-factor authentication and full-disk encryption.

Conclusions

Introducing free software into the educational process makes it possible to move from declarative cyber hygiene instruction to the formation of lasting practical skills. The proposed three-level module (passwords, web browsing, operating environment) provides a gradual increase in complexity and can be integrated into information security, computer science or digital literacy courses. The openness of the source code helps students understand how encryption, traffic filtering and access control actually work, which raises their overall digital literacy, while the institution avoids expenses on commercial licenses.

References

1. Phishing, zero-click exploits, AI attacks: cyber threat trends in Ukraine in 2025. IT Ukraine Association. URL: https://itukraine.org.ua/en/phishing-zero-click-exploits-ai-attacks-cyber-threat-trends-in-ukraine-in-2025/ (accessed: 30.09.2026).
2. What is Free Software? Free Software Foundation. URL: https://www.gnu.org/philosophy/free-sw.html (accessed: 30.09.2026).
3. KeePassXC: Cross-Platform Password Manager. URL: https://keepassxc.org (accessed: 30.09.2026).
4. uBlock Origin. GitHub repository. URL: https://github.com/gorhill/uBlock (accessed: 30.09.2026).
5. The Tor Project. URL: https://www.torproject.org (accessed: 30.09.2026).
6. Ubuntu security documentation. Canonical. URL: https://documentation.ubuntu.com/security/ (accessed: 30.09.2026).
7. Tails: portable operating system. URL: https://tails.net (accessed: 30.09.2026).
8. How to: Use KeePassXC. Surveillance Self-Defense. Electronic Frontier Foundation. URL: https://ssd.eff.org/module/how-use-keepassxc (accessed: 30.09.2026).

Nahornyi B. I., Lysenok D. V., Savenko A. O.
Chernihiv Polytechnic National University, Department of Information and Computer Systems, Chernihiv, Ukraine
Corresponding author: Bohdan Nahornyi, e-mail: bogdan.nahornyy@gmail.com, ORCID: 0009-0004-1379-9787

AN OPEN BROWSER-BASED CO-SIMULATOR OF STM32 FIRMWARE AND ELECTRICAL CIRCUITS AS A FREE ALTERNATIVE TO PROTEUS

Abstract. Microcontroller lab courses mostly rely on the proprietary Proteus VSM simulator; the free Wokwi and QEMU execute firmware without circuit physics, and SPICE computes the circuit without code. We built εmul, an open (Apache-2.0) browser-based platform that co-simulates an ARM Cortex-M4/M7 emulator with STM32F429 and STM32F746 peripherals and a transient circuit model based on modified nodal analysis with a 20 μs step; firmware from STM32CubeIDE runs unmodified. We checked accuracy against independent references: 1074 check values from seven test programs matched a host build with no discrepancies; over 22 quantities in eight circuits the median deviation from ngspice is 0.016 %; the θ-method's numerical damping was derived analytically and confirmed at 55 points with a median discrepancy of 0.05 %. Unmodified lab firmware behaves identically on a physical Open746I-C board and in εmul, and the lab bench runs in the browser in real time (1.00×).

Keywords: free software; microcontroller emulation; STM32; co-simulation; circuit simulation; laboratory course

Introduction

A microcontroller lab course without a board on every desk depends on a bench simulator, and the de facto standard remains Proteus VSM [8]: closed source, a paid licence, Windows only. Free tools each solve part of the problem: Wokwi [12] runs firmware in the browser, but a pin there is either 0 or 1, a project holds a single microcontroller, and its only STM32 is a Cortex-M0+; SimulIDE [10] couples AVR, PIC and 8051 emulation with a circuit model, but as a desktop application and without ARM; QEMU [1] emulates the processor without a circuit; the SPICE family, today ngspice [7], computes the circuit but does not run code. According to a review in Science [2], combining a virtual laboratory with a physical one yields more than either alone, so the firmware should be the same on the board and in the simulator.

The aim of the work is to build a platform in which unmodified firmware from STM32CubeIDE and a physical circuit model with datasheet component limits run in one loop, and to measure its accuracy against references outside the project itself. Scientific novelty: for the first time among the open tools known to the authors, instruction-set-level Cortex-M4/M7 emulation is combined with a transient nonlinear circuit model in the browser; simulator verification is improved: the integrator error is derived analytically and confirmed against an independent reference.

Materials and methods

The platform is written in TypeScript, the code is open [13], and an instance is deployed at emul.digituni.org. In the terminology of the survey [4], this is co-simulation of a discrete-event simulation unit (core with peripherals) and a continuous one (circuit) with a shared step $\Delta t = 20$ μs: in each step the core executes its cycles and sets the pin levels, the solver finds the node voltages, and inputs and the ADC read them in the next step. The core runs in a worker one step ahead of the solver; serial buses (a bit at 115,200 baud is shorter than Δt) are carried as time-exact edges.

We also wanted a low barrier to entry. The student's cycle: open a link, place a board, wire the parts, drag an .elf from STM32CubeIDE onto the board or write code in the built-in Monaco editor with completion over HAL and CMSIS (about 30,000 symbols), press Compile (about a second, GCC diagnostics in the code) and Run. The same editor is a debugger: breakpoints, stepping through the C source, variables, registers and peripheral registers, read from the image's DWARF data. The bench has an oscilloscope, a logic analyser with UART, SPI and I²C decoders, a terminal and meters, and an inspector names the peripheral blocks the firmware touched but the model lacks. A bench is shared as a link, or edited live by several people who see each other's cursors. Nothing is

installed on the student's computer. We wrote our own solver instead of using SPICE because the bench needs the circuit state at every step, component failures per datasheet and a cache of operating points; the fixed step makes runs deterministic, at the cost shown in Figure 1.

The emulator implements ARMv7E-M with the DSP extension and a single- and double-precision FPU, the interrupt controller and low-power modes, and the peripherals of the STM32F429ZI and STM32F746IG from clocks, GPIO and timers to serial buses, DMA, ADC/DAC and the LCD controller; the full list is in the repository. The build service uses GNU Arm Embedded 12.2 with HAL v1.8.5 and v1.3.3 [11].

The circuit is solved by modified nodal analysis [6]. At each step, Newton iteration (1) with the convergence criterion (2), εabs = 10⁻⁶, εrel = 10⁻³, is run for the nonlinear elements; the diode is described by the Shockley equation, the bipolar transistor by the Ebers–Moll model [3], the MOSFET by the Shichman–Hodges model [9]. Capacitors and inductors are replaced by θ-method companion models (3), θ = 1 being implicit Euler and θ = 0.5 the trapezoidal rule:

$$A(x_k)\, x_{k+1} = z(x_k) \qquad (1)$$

$$|x_{k+1} - x_k| < \varepsilon_{\text{abs}} + \varepsilon_{\text{rel}} \cdot |x_{k+1}| \qquad (2)$$

$$i_n = C / (\theta \Delta t) \cdot (v_n - v_{n-1}) - ((1 - \theta) / \theta) \cdot i_{n-1} \qquad (3)$$

The value θ = 0.52 was chosen so that ringing on switching edges decays while resonances lose only a few percent of their quality factor. Its cost can be computed exactly. On a linear circuit the θ-method is equivalent (the stability function of the θ-method [5]) to replacing the Laplace operator s with the discrete operator (4), z = exp(jωΔt), so the steady-state response follows from the same transfer function; for small ωΔt the damping reduces to an equivalent quality factor (5), with N steps per period:

$$s_{\text{eq}}(z) = (1 - z^{-1}) / (\Delta t \cdot (\theta + (1 - \theta)\, z^{-1})) \qquad (4)$$

$$Q_{\text{num}} \approx N / (2\pi(2\theta - 1)), \quad I / I_0 \approx Q_{\text{num}} / (Q + Q_{\text{num}}) \qquad (5)$$

We checked each layer against a reference outside the project. The core: seven C test programs in three builds (Cortex-M4 -O0 and -O2, Cortex-M7 -O2), each leaving 358 values in SRAM, against the same programs built for the host; compiled blocks against the interpreter. The solver: the eight circuits of Table 1 against ngspice 36 [7] with the same model parameters at T = 300 K. The integrator: RLC with Q = 10 at resonance for θ ∈ {0.5, 0.52, 0.6, 0.75, 1} and N from 8 to 256 against (4). The whole bench: one .elf on a physical Open746I-C and in εmul. Speed was measured in the production build, Chromium 151, Intel Core i5-10400, 10 s window, three runs. All measurements refer to release v1.22.1.

Results

All 21 test-program images (1074 check values) matched the host build with no discrepancies; compiled blocks and the interpreter agreed over 506.6 million instructions. The comparison with ngspice is given in Table 1: over 22 measured quantities the median |Δ| is 0.016 %, DC operating points are within 0.07 %, the rectifier within 0.01 %. The only substantial deviation, −7.51 % of the RLC current at resonance, is θ-method damping at 31 steps per period, and (4) puts it at −7.47 %.

Table 1 – The εmul solver versus ngspice 36 on the same models (step 20 µs, θ = 0.52)

Quantity	ngspice	εmul	Δ, %
5.1 V Zener: load voltage, V	5.1255	5.1249	−0.01
CE amplifier (BC547): collector voltage, V	8.9685	8.9671	−0.02
CE amplifier: gain at 1 kHz	1.7613	1.7623	+0.05
Series RLC at resonance, Q = 10: current, mA	99.03	91.59	−7.51
Bridge rectifier with transformer: peak, V	14.363	14.364	+0.01
Bridge rectifier: ripple peak-to-peak, V	1.0291	1.0291	0.00
Current mirror: output current, mA	1.1229	1.1230	+0.01
N-MOSFET, on: drain–source voltage, mV	7.0387	7.0436	+0.07

Integrator damping is shown in Figure 1: the 55 measured points fall on the curves of (4) with a median discrepancy of 0.05 % and a maximum of 0.54 %. At θ = 0.52, $Q_{num} \approx 4N$: a resonance at 0.4 kHz loses 2 % of its response, at 1.6 kHz 7.5 %, at 6.25 kHz 41 %.

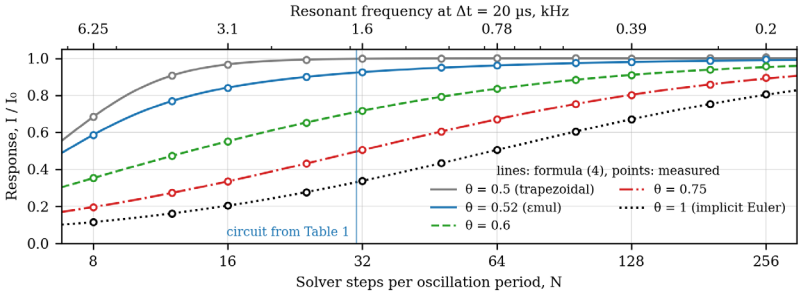


Figure 1 – RLC response (Q = 10) at resonance versus steps per period: lines from (4), points measured in εmul

The whole-bench check used an STM32CubeIDE application written for the course, not for the simulator: joystick interrupts with a software debounce step four LEDs, with the dwell set in 1 s steps. The same .elf, flashed onto the physical Open746I-C and dropped into εmul, behaved identically; in εmul the dwell read 1.000 s for a set 1 s and 5.000 s at the 5 s cap. In the browser the lab bench (F746, 50 MHz) and the two LCD demos at 200 MHz run in real time (0.997–1.001×), firmware with an idle loop at 180 MHz at 0.54–0.85×. On such benches the solver costs almost nothing: in a blink firmware 99.9 % of steps are skipped as settled, so the core sets the speed.

Discussion

The fixed 20 µs step suppresses resonances above about 2 kHz (Figure 1): filters and oscillators need an adaptive step. Benches below real time give the same results, only later, because simulated time is decoupled from the wall clock. The solver holds real time up to ~40 linear or ~15–20 nonlinear unknowns. The comparison with ngspice on identical simplified models verifies the solver and says nothing about how close the models are to silicon; core cycles are counted without a pipeline, and timing on the board was observed, not measured with an instrument. There are no ready-made logic ICs, character displays or sensors, but digital logic can be written in VHDL or Verilog: the build service synthesises it with GHDL and Yosys, and the netlist runs on the bench as a part. Coverage grows through data rather than code: a new chip of the family is a profile of ~35 lines, a component ~25.

Conclusions

A microcontroller lab can run on an unmodified STM32CubeIDE project without a board on every desk and without a proprietary simulator: the firmware and the circuit are simulated together in the browser, and on the benches measured the outcome does not depend on whether the firmware ran on the board or in εmul. The damping of a fixed-step θ-method integrator follows from its stability function, so any simulator of this kind can check its integrator against formula (4) before comparing with a reference. Next: a structured evaluation with students in the course, instrument measurements of UART, PWM and wake-from-Stop timings on the board compared with εmul, an adaptive step for resonant circuits, and 74xx logic ICs.

References

1. Bellard F. QEMU, a fast and portable dynamic translator. USENIX Annual Technical Conference, FREENIX Track, 2005, pp. 41–46. URL: https://www.usenix.org/legacy/event/usenix05/tech/freenix/full_papers/bellard/bellard.pdf (accessed: 25.09.2026).

2. de Jong T., Linn M. C., Zacharia Z. C. Physical and virtual laboratories in science and engineering education. Science, 2013, 340(6130), 305–308. DOI: 10.1126/science.1230579.

3. Ebers J. J., Moll J. L. Large-signal behavior of junction transistors. Proceedings of the IRE, 1954, 42(12), 1761–1772. DOI: 10.1109/JRPROC.1954.274797.

4. Gomes C., Thule C., Broman D., Larsen P. G., Vangheluwe H. Co-simulation: A survey. ACM Computing Surveys, 2018, 51(3), 1–33. DOI: 10.1145/3179993.

5. Hairer E., Wanner G. Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems. Springer, 1996. DOI: 10.1007/978-3-642-05221-7.
6. Ho C.-W., Ruehli A. E., Brennan P. A. The modified nodal approach to network analysis. IEEE Transactions on Circuits and Systems, 1975, 22(6), 504–509. DOI: 10.1109/TCS.1975.1084079.
7. Ngspice, the open source Spice circuit simulator. ngspice.sourceforge.io. URL: https://ngspice.sourceforge.io/ (accessed: 25.09.2026).
8. Proteus Design Suite: simulation (VSM). Labcenter Electronics. URL: https://www.labcenter.com/simulation/ (accessed: 25.09.2026).
9. Shichman H., Hodges D. A. Modeling and simulation of insulated-gate field-effect transistor switching circuits. IEEE Journal of Solid-State Circuits, 1968, 3(3), 285–289. DOI: 10.1109/JSSC.1968.1049902.
10. SimulIDE: real-time electronic circuit simulator. GitHub, SimulIDE, licence GPL-3.0. URL: https://github.com/SimulIDE/SimulIDE (accessed: 25.09.2026).
11. STM32Cube HAL Driver MCU Components: stm32f4xx-hal-driver v1.8.5, stm32f7xx-hal-driver v1.3.3. GitHub, STMicroelectronics. URL: https://github.com/STMicroelectronics (accessed: 25.09.2026).
12. Wokwi: online electronics simulator. wokwi.com. URL: https://wokwi.com/ (accessed: 25.09.2026).
13. ɛmul: circuit simulator for microcontroller labs that runs real STM32 firmware. GitHub, Apache-2.0, release v1.22.1, commit 5bcdc6a. URL: https://github.com/ChernihivPolytechnicNationalUniversity/emul/tree/5bcdc6a (accessed: 25.09.2026).

Romanenko M. V.
Chernihiv Polytechnic National University, Department of Information and Computer Systems, Chernihiv, Ukraine
Corresponding author: Myroslava Romanenko, e-mail: mira.romanenko.3@gmail.com, ORCID: 0009-0008-1897-3122

FOSS APPROACH IN MOBILE DEVELOPMENT: CREATING AN OPEN-SOURCE ANDROID APPLICATION WITH THE INTEGRATION OF READY-MADE LIBRARIES FOR STYLISTIC RECOMMENDATIONS

Abstract. In the contemporary landscape of mobile development, Free and Open-Source Software (FOSS) is acquiring critical significance due to code accessibility, flexibility, and free distribution capabilities. Traditional closed solutions often restrict developers through high licensing costs, a lack of transparency, and vendor dependency, whereas the FOSS approach enables the creation of full-featured Android applications utilizing pre-built libraries and frameworks while ensuring user data confidentiality through on-device processing.

This paper examines the experience of developing an open-source Android application for personalized stylistic recommendations. The application integrates a range of ready-to-use FOSS libraries (Kotlin, Jetpack Compose, Material Design 3, AndroidX Palette, Glide, Retrofit, TensorFlow Lite) and implements color analysis features, weather and seasonal conditions accounting, and event type categorization. Particular attention is paid to on-device data processing, which enhances performance and privacy protection. The results demonstrate that the application of FOSS significantly accelerates development, reduces costs, and promotes free code distribution via open repositories.

Keywords: FOSS, free and open-source software, Android, Jetpack Compose, TensorFlow Lite, mobile development, stylistic recommendations, on-device processing, open source.

Introduction

In the contemporary landscape of rapid mobile technology development and growing demand for personalized digital services, Free and Open-Source Software (FOSS) is gaining critical significance. It provides developers with unhindered access to pre-built libraries, frameworks, and tools, while fostering code transparency, modifiability, and free distribution. This is especially relevant in the field of Android development, where the ecosystem of Kotlin, Jetpack Compose, and numerous open-source projects enables the rapid creation of complex applications with high levels of functionality and data privacy.

However, the context of FOSS adoption in mobile development is not without contradictions. Traditional proprietary solutions are frequently accompanied by high licensing costs, limited customization capabilities, and vendor lock-in. In the segment of stylistic recommendation applications, this issue manifests itself acutely: the majority of existing products are closed-source, focused on monetization through subscriptions or advertising, and rarely offer comprehensive on-device data processing. The lack of open, easily adaptable solutions significantly restricts the opportunities for individual developers and small teams striving to create quality products without substantial financial investments.

The motivation for this research stems from the necessity to demonstrate the practical value of the FOSS approach using a full-scale mobile application as an example. A brief overview of literature and open repositories indicates that while FOSS is actively utilized in general Android development, the number of open solutions remains limited within the domain of personalized stylistic recommendations. Existing projects predominantly either possess narrow functionality or fail to ensure an appropriate level of privacy due to cloud-based data processing.

The aim of this work is to demonstrate the FOSS approach in mobile development by creating an open-source Android application for personalized stylistic recommendations. The application integrates pre-built FOSS libraries, implements color analysis, accounts for weather conditions, seasons, and event types, and ensures maximum data processing directly on the user's device. Such an approach combines development velocity, low cost, code transparency, and the potential for free distribution of the completed solution.

Materials and Methods

To analyze and implement the FOSS approach in mobile development, empirical data from the official documentation of Android, Jetpack Compose, TensorFlow Lite, and other open-source libraries were utilized, alongside an analysis of GitHub open repositories dedicated to stylistic recommendation applications and image processing. Additionally, public developer feedback regarding the advantages and limitations of on-device data processing in mobile applications was taken into account.

The primary tool employed was an open-source Android application developed using a modern FOSS stack. The key technologies include:

– The Kotlin programming language (the official open-source programming language for Android development);

– The declarative UI framework Jetpack Compose combined with Material Design 3;

– The MVVM architectural pattern with elements of Clean Architecture;

– The AndroidX Palette library for analyzing the color palette of images;

– Glide for asynchronous image loading and caching;

– Retrofit for executing network requests to retrieve up-to-date weather data;

– TensorFlow Lite for executing machine learning elements directly on the device.

The development and deployment environment comprised Android Studio (stable versions for 2024–2025), Gradle as the build system, a minimum SDK of 24, and a target SDK of 34+. Testing was conducted on both emulators and physical devices running various Android versions. To ensure reproducibility, fixed dependencies within build.gradle.kts files and clear project configuration instructions were utilized.

Data processing algorithms encompassed:

– Automatic extraction of dominant and accent colors from clothing photographs using AndroidX Palette;

– Generation of harmonious outfits based on color theory (complementary, analogous, and triadic schemes);

– Incorporation of external factors (weather, season, event type) during recommendation generation;

– Interactive user color diagnostics;

– On-device execution of TensorFlow Lite models to enhance performance and preserve privacy.

The project configuration ensured the maximal utilization of local device resources: image processing and recommendation generation were performed without transmitting personal data to external servers. Network requests were applied solely for obtaining public weather data. All source codes, resources, configuration files, and deployment instructions are hosted in the open repository: https://github.com/3Mira3/Mobile_application_Assistant_in_world_of_fashion

This ensures complete research reproducibility: any developer can clone the repository, synchronize Gradle dependencies, and launch the application on an emulator or a physical device.

Results

As a result of implementing the open-source Android application "Fashion Assistant", full compliance with FOSS principles was achieved: all source code, configurations, and resources have been published in an open repository with provisions for free cloning, compilation, and modification. The application successfully integrates pre-built FOSS libraries and implements core functionalities for personalized stylistic recommendations with predominantly on-device data processing.

Comparative Analysis with Existing Mobile Fashion Applications. To evaluate the advantages of the developed solution, a comparison was conducted with popular commercial applications (Acloset, Whering, Cladwell, Indyx, and analogues). The evaluation criteria included: code openness, monetization model, privacy level (on-device vs. cloud), presence of color analysis, consideration of weather/season/event, educational content, and offline functionality.

Table 1 – Comparison of the developed application with existing mobile fashion applications

Criterion	Developed Application	Acloset / Whering / Cladwell / Indyx
Open Source (FOSS)	Yes (full access on GitHub)	No (closed proprietary code)
Monetization Model	Completely free	Subscriptions / freemium / ads
Data Processing	Predominantly on-device	Mostly cloud-based
Image Color Analysis	Yes (AndroidX Palette)	Limited or absent
Weather / Season / Event Accounting	Yes	Partially (depends on the app)
Interactive Color Diagnostics	Yes	Rarely
Educational Content (Fashion History)	Yes	Absent
Offline Mode for Core Features	Yes	Limited or absent
Community Modification & Evolution	Yes	No

The comparison results demonstrate that the developed application possesses significant advantages regarding openness, privacy, and the absence of financial barriers. Unlike commercial solutions, which typically require subscriptions and transmit user data to servers, the FOSS approach ensures complete transparency and the opportunity for further evolution by any developer.

Comparison of Utilized FOSS Libraries with Alternatives. The utilized FOSS libraries demonstrated high efficiency compared to alternatives: TensorFlow Lite provided superior integration with Android and higher on-device processing performance than ML Kit, ONNX Runtime, or PyTorch Mobile; Jetpack Compose proved to be more productive and lightweight than Flutter and React Native; AndroidX Palette provided a stable solution for color analysis without requiring complex custom implementations; Glide exhibited more reliable image caching performance compared to Coil and the legacy Picasso; while Retrofit simplified network requests through convenient typing compared to Ktor or pure OkHttp.

Discussion

The obtained results confirm that the FOSS approach enables the creation of full-featured mobile applications with a high level of functionality, privacy, and openness without significant financial expenses. The comparison with commercial solutions revealed substantial advantages of the developed application in terms of the absence of subscriptions, on-device data processing, and the capability for free code distribution. Simultaneously, certain limitations exist: dependence on input image quality for color analysis, limited recommendation accuracy in the absence of large training datasets, and the need for further functional expansion (e.g., full Room integration or more complex machine learning models).

Compared to traditional proprietary approaches, FOSS provides greater flexibility and transparency, although it requires independent configuration and maintenance by the developer. Application prospects encompass further community-driven development through the open repository, adaptation of the solution for other domains of personalized recommendations, and utilization as an educational example of open-source software implementation in mobile development.

Conclusions

The study demonstrates the effectiveness of the FOSS approach in mobile development through the creation of an open-source Android application for personalized stylistic recommendations, featuring the integration of pre-built open libraries and on-device data processing. The primary contribution lies in the practical confirmation of the advantages of open-source software regarding development velocity, user privacy, and the capability for free code distribution. Future research may be directed toward expanding the application's functionality, improving machine learning models, and engaging the community in the collaborative development of the project.

References

1. Android Developers. Jetpack Compose Documentation. URL: https://developer.android.com/compose (Date of access: 27.09.2026).
2. TensorFlow. TensorFlow Lite Guide. URL: https://www.tensorflow.org/lite (Date of access: 27.09.2026).
3. Романенко М. В. Mobile application Assistant in the world of fashion [Electronic resource] : open repository. URL: https://github.com/3Mira3/Mobile_application_Assistant_in_the_world_of_fashion (Date of access: 27.09.2026).

Shevchenko N. A., Kibitkin D. S., Shymchuk G. V.
Ternopil Ivan Puluj National Technical University, Department of Computer Science, Ternopil, Ukraine
Corresponding author: Grigorii Shymchuk, e-mail: gorych@gmail.com, ORCID: 0000-0003-2362-7386

REPRODUCIBLE OPEN-SOURCE BENCHMARK OF CONCEPT-DRIFT DETECTORS FOR STREAMING REGRESSION

Abstract. A reproducible open-source benchmark for detecting concept drift in regression-error streams is presented. The benchmark evaluates three lightweight detectors suitable for online monitoring: Page-Hinkley, one-sided CUSUM, and a Kolmogorov-Smirnov sliding-window detector. Controlled streams contained 2,600 observations with a change point at observation 1,300. Sudden and gradual mean shifts were generated, and each scenario was repeated 100 times with fixed pseudo-random seeds. Page-Hinkley produced the shortest mean detection delay for both sudden and gradual drift (18.0 and 114.8 observations, respectively) but generated 5 false alarms in 100 no-drift runs. CUSUM was slightly slower for sudden drift (20.1 observations) and substantially slower for gradual drift (176.7 observations) but produced no false alarms in the no-drift benchmark. The KS-window detector required 34.4 observations for sudden drift and 167.6 for gradual drift and generated 4 false alarms. The results illustrate the practical trade-off between responsiveness and false-alarm control and provide a compact FOSS methodology that can be integrated with streaming-regression models.

Keywords: concept drift; data streams; online learning; Page-Hinkley; CUSUM; Kolmogorov-Smirnov test; regression monitoring; open-source software.

Introduction

Machine-learning models deployed on data streams are often trained under the assumption that the statistical relation between inputs and targets remains sufficiently stable. In practice, seasonal changes, operating regimes, sensor replacement, market behavior, or changes in the physical process can modify the distribution observed after deployment. This phenomenon is commonly described as concept drift. If drift is not identified, a model can remain numerically operational while its predictive error progressively increases [1,2].

Concept-drift research distinguishes sudden, gradual, incremental, and recurring changes. Recent reviews emphasize that detection and adaptation are separate problems: a detector first indicates that the data-generating process or model error has changed, after which the forecasting system may retrain, reset, reweight, or switch models [2,3]. Open-source ecosystems such as River provide incremental learning and drift-detection components for dynamic data streams [1], while recent regression research continues to use Page-Hinkley and adaptive-window mechanisms for online change detection [4].

The objective of this work is to build a small reproducible benchmark that can be executed with Python, NumPy, and SciPy and later connected to any regression model that produces a sequential error statistic. The benchmark intentionally separates detector behavior from a specific forecasting algorithm: the monitored stream can represent absolute error, squared error, normalized residual magnitude, or another operational score. Three detectors with different statistical logic are compared under identical controlled changes.

Materials and Methods

Each benchmark stream contained 2,600 observations. Before the change point, the monitored statistic followed a normal distribution with mean 0.65 and standard deviation 0.12. For sudden drift, the distribution changed at observation 1,300 to mean 1.00 and standard deviation 0.14. For gradual drift, the mean increased linearly from 0.65 to 1.00 during 350 observations and then remained at 1.00. A no-drift scenario preserved the original distribution for all 2,600 observations. One hundred independent runs were generated for each scenario.

Table 1 – Controlled data-stream scenarios

Scenario	Before change	Transition	After change
No drift	$N(0.65, 0.12^2)$	none	same distribution
Sudden	$N(0.65, 0.12^2)$	at t = 1300	$N(1.00, 0.14^2)$
Gradual	$N(0.65, 0.12^2)$	350-point linear shift	$N(1.00, 0.14^2)$

The Page-Hinkley detector tracks cumulative deviations from an online mean. For an observation x_t, the cumulative statistic was updated as

$$m_t = m_{t-1} + x_t - \bar{x}_t - \delta, \quad PH_t = m_t - \min(m_1, \ldots, m_t). \qquad (1)$$

A drift alarm was generated when PH_t exceeded λ. The benchmark used δ = 0.005 and λ = 7.5 after a 50-observation grace period. CUSUM used the first 300 observations to estimate the baseline mean μ_0 and accumulated only positive deviations,

$$S_t = \max[0, S_{t-1} + x_t - \mu^0 - k], \quad alarm\ if\ S_t > h. \qquad (2)$$

with k = 0.08 and h = 5.5. The third detector used a 160-point moving window, compared the first 80 and last 80 observations with the two-sample Kolmogorov-Smirnov test, and raised an alarm for $p < 10^{-4}$. Detection was checked every 10 observations. Figure 1 illustrates one sudden-drift realization; for this seed, Page-Hinkley, CUSUM, and KS-window signaled at observations 1306, 1318, and 1330, respectively.

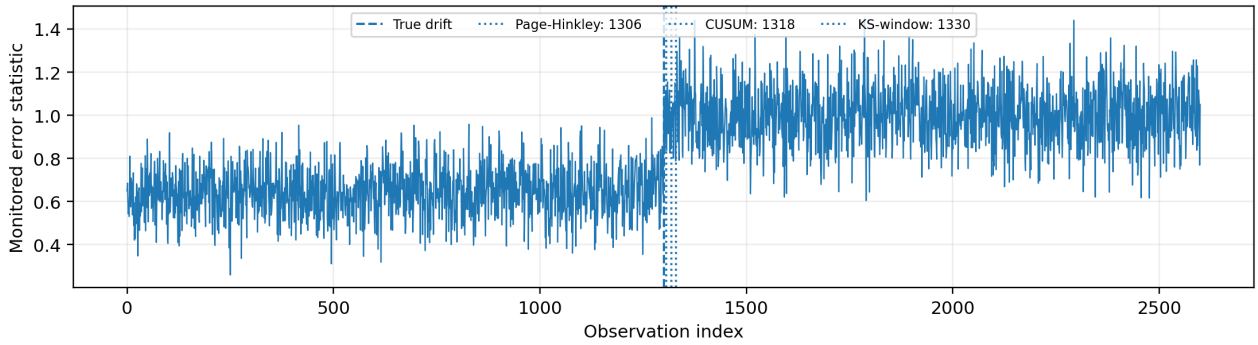


Figure 1 – Example sudden concept drift and detector alarm positions

Two evaluation criteria were used. Detection delay D was measured only for alarms after the true change point, $D = t_{alarm} - t_{change}$. False alarms were alarms before the true change point or any alarm in a no-drift stream. A detector was considered to have detected a scenario if at least one post-change alarm occurred. These measures are widely used because a practical detector should be fast after genuine change but conservative during stationary operation [2,5].

Results

All three detectors identified the injected sudden and gradual shifts in the 100-run benchmark. Table 2 shows the mean detection delay and the no-drift false-alarm count. Page-Hinkley was the most responsive detector, with an 18.0-observation mean delay after sudden drift and 114.8 observations after gradual drift. Its higher sensitivity was accompanied by five false alarms in the no-drift runs. CUSUM required 20.1 observations for sudden drift and 176.7 observations for gradual drift but produced no no-drift false alarms. KS-window was slower than Page-Hinkley for both scenarios and produced four no-drift false alarms.

Table 2 – Detector performance over 100 runs per scenario

Detector	Sudden delay, mean	Gradual delay, mean	Detection rate	False alarms /100 no-drift
Page-Hinkley	18.0	114.8	100%	5
CUSUM	20.1	176.7	100%	0
KS-window	34.4	167.6	100%	4

Discussion

The experiment demonstrates a common drift-monitoring trade-off. Page-Hinkley accumulated evidence rapidly and was therefore effective for both sudden and gradual upward changes, but the same responsiveness increased false-alarm risk. CUSUM was more conservative under the selected parameters: this reduced false alarms but also delayed detection when the change developed gradually. The KS-window detector was distribution-oriented rather than focused specifically on a mean increase, which makes it conceptually more general but less efficient in this benchmark, where the injected drift was dominated by a mean shift.

The results should not be interpreted as a universal ranking. Drift-detector performance depends on change magnitude, noise level, autocorrelation, parameter settings, and the monitored

statistic. Current surveys similarly emphasize that no single detector dominates every stream and that evaluation should reflect the expected operating conditions [2,3,5]. The advantage of the proposed benchmark is reproducibility: stream generators, detector parameters, and metrics are explicit and can be replayed before deployment on real residual streams.

For regression monitoring, an effective workflow is to apply a drift detector to an online error statistic rather than directly to the raw target. This links the alarm to loss of predictive performance. After an alarm, the system can retrain on a recent window, update an incremental model, or activate a backup model. River provides a suitable open-source architecture for this type of streaming workflow [1], while Page-Hinkley-based adaptation has also been evaluated recently in multi-target regression [4].

The main limitation is the controlled Gaussian benchmark. Real errors may be heavy-tailed, autocorrelated, seasonal, and affected by missing data. Future experiments should therefore use residual streams from gas-consumption and network-telemetry models, include recurring and variance-only drift, and compare adaptation policies in addition to alarm quality. Confidence intervals for delay and false-alarm rate should also be reported when the benchmark is expanded.

Conclusions

A reproducible FOSS benchmark of three concept-drift detectors was developed for streaming regression monitoring. Page-Hinkley achieved the shortest mean delay for sudden and gradual changes, CUSUM provided the strongest false-alarm control, and KS-window offered a nonparametric distributional alternative. The benchmark provides explicit stream definitions, detector parameters, and evaluation metrics and can be integrated with online forecasting systems implemented in Python and River.

References

1. Montiel J., Halford M., Mastelini S. M. et al. River: machine learning for streaming data in Python. Journal of Machine Learning Research, 2021, 22(110), 1-8.
2. Arora S., Rani R., Saxena N. A systematic review on detection and adaptation of concept drift in streaming data using machine learning techniques. WIREs Data Mining and Knowledge Discovery, 2024, 14(4), e1536. https://doi.org/10.1002/widm.1536.
3. Hinder F., Vaquet V., Hammer B. One or two things we know about concept drift – a survey on monitoring in evolving environments. Part A: detecting concept drift. Frontiers in Artificial Intelligence, 2024, 7, 1330257. https://doi.org/10.3389/frai.2024.1330257.
4. Stevanoski B., Kostovska A., Panov P., Dzeroski S. Change detection and adaptation in multi-target regression on data streams. Machine Learning, 2024, 113, 8585-8622. https://doi.org/10.1007/s10994-024-06621-z.
5. Lukats D., Zielinski O., Hahn A., Stahl F. T. A benchmark and survey of fully unsupervised concept drift detectors on real-world data streams. International Journal of Data Science and Analytics, 2025, 19, 1-31. https://doi.org/10.1007/s41060-024-00620-y.
6. Hinder F., Vaquet V., Hammer B. One or two things we know about concept drift – a survey on monitoring in evolving environments. Part B: locating and explaining concept drift. Frontiers in Artificial Intelligence, 2024, 7, 1330258. https://doi.org/10.3389/frai.2024.1330258.
7. Virtanen P., Gommers R., Oliphant T. E. et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature Methods, 2020, 17, 261-272. https://doi.org/10.1038/s41592-019-0686-2.
8. Page E. S. Continuous Inspection Schemes. Biometrika, 1954, 41(1/2), 100-115. https://doi.org/10.1093/biomet/41.1-2.100.

Shevchenko N. A., Kibitkin D. S., Shymchuk G. V.
Ternopil Ivan Puluj National Technical University, Department of Computer Science, Ternopil, Ukraine
Corresponding author: Grigorii Shymchuk, e-mail: gorych@gmail.com, ORCID: 0000-0003-2362-7386

INTERPRETABLE OPEN-SOURCE MACHINE LEARNING FOR NATURAL GAS CONSUMPTION FORECASTING WITH WEATHER AND LAG FEATURES

Abstract. An interpretable open-source machine-learning workflow for short-term natural gas consumption forecasting is investigated using historical gas-demand and meteorological observations from 2007-2019. Hourly gas volumes were aggregated into non-overlapping 3-hour intervals and aligned with 3-hour weather records. Models were trained on 2007-2018 and evaluated chronologically on the unseen year 2019. Decision Tree, Random Forest, and Histogram-based Gradient Boosting regressors were compared using R^2, MAE, RMSE, and WAPE. Random Forest achieved the best test performance with $R^2 = 0.9656$, MAE = 4749.6 m^3, RMSE = 7955.9 m^3, and WAPE = 9.52%. Permutation importance showed that the strongest predictive signal was the daily lag Q(t-24 h), followed by Q(t-3 h), Q(t-7 d), outdoor temperature, and cyclic hour features. A reduced eight-feature Random Forest preserved performance (WAPE = 9.46%), while excluding weather increased WAPE to 9.67%. The results demonstrate that an open-source Python/scikit-learn pipeline can combine accurate forecasting with transparent feature-level interpretation.

Keywords: open-source software; natural gas consumption; time-series forecasting; machine learning; Random Forest; explainable AI; permutation importance; weather features.

Introduction

Accurate short-term forecasting of natural gas consumption is important for operational planning, balancing, and demand management. Gas demand is strongly seasonal and depends simultaneously on recent consumption, daily and weekly cycles, and meteorological conditions. Machine-learning methods can represent nonlinear relations between these variables, but operational use also requires an understanding of why a forecast changes. This has increased interest in explainable and interpretable forecasting approaches, especially for energy time series [1-3].

Recent energy-forecasting studies combine predictive models with feature-attribution methods. Eskandari et al. integrated feature selection and interpretable machine learning for energy-consumption forecasting and emphasized the value of a compact feature subset [4]. Neubauer et al. showed that lagged load variables and weather features are central to explainable heating-load forecasting [5]. Mohammadi et al. reported that exogenous variables can improve energy time-series forecasts and that feature perturbation methods provide robust interpretability signals [6]. These findings motivate a reproducible workflow in which forecast accuracy and feature relevance are evaluated together.

The objective of this work is to assess an open-source tree-based forecasting pipeline on a real multiyear gas-consumption dataset and to quantify the contribution of lagged demand, calendar variables, and weather observations. In contrast to a purely predictive comparison, the analysis includes permutation-based feature importance and an ablation test to determine whether a reduced set of variables can preserve forecast quality.

Materials and Methods

The dataset covers 1 January 2007 through 31 December 2019. Hourly gas-consumption values Q were aggregated by summing each non-overlapping block of three consecutive hours, producing 37,984 target observations measured in cubic metres per 3-hour interval. These records were aligned with 3-hour meteorological measurements containing air temperature T, station pressure Po, sea-level pressure P, pressure tendency Pa, relative humidity U, wind speed Ff, precipitation RRR, dew-point temperature Td, and visibility VV. Calendar variables were encoded by sine/cosine transformations of hour-of-day and day-of-year, together with a weekend indicator.

Table 1 – Forecasting feature groups

Group	Variables	Role
Demand lags	Q_lag_1, Q_lag_8, Q_lag_56	3 h, 24 h, 7 d memory
Weather	T, Po, P, Pa, U, Ff, RRR, Td, VV	exogenous conditions
Daily cycle	hour_sin, hour_cos	time-of-day periodicity

Annual / week	doy_sin, doy_cos, weekend	seasonality and weekly effect

Three open-source regressors were evaluated: a CART Decision Tree, Random Forest, and Histogram-based Gradient Boosting, all implemented with scikit-learn. The training interval was 2007-2018, and 2019 was kept completely unseen during fitting. This chronological split avoids information leakage from future observations. The primary relative error measure was weighted absolute percentage error (WAPE),

$$WAPE = \left[\frac{\Sigma|Q_i - \hat{Q}_i|}{\Sigma|Q_i|}\right] \cdot 100\%. \quad (1)$$

where Q_i is the measured 3-hour gas volume and $\hat{Q}_i$ is the corresponding model forecast. R^2, MAE, and RMSE were also calculated. For interpretability, permutation importance was evaluated on a fixed 2019 subset: one feature at a time was randomly permuted and the increase in MAE was recorded. A large increase indicates that the model depends strongly on that variable. This method is model-agnostic and is implemented directly in the scikit-learn inspection module [7].

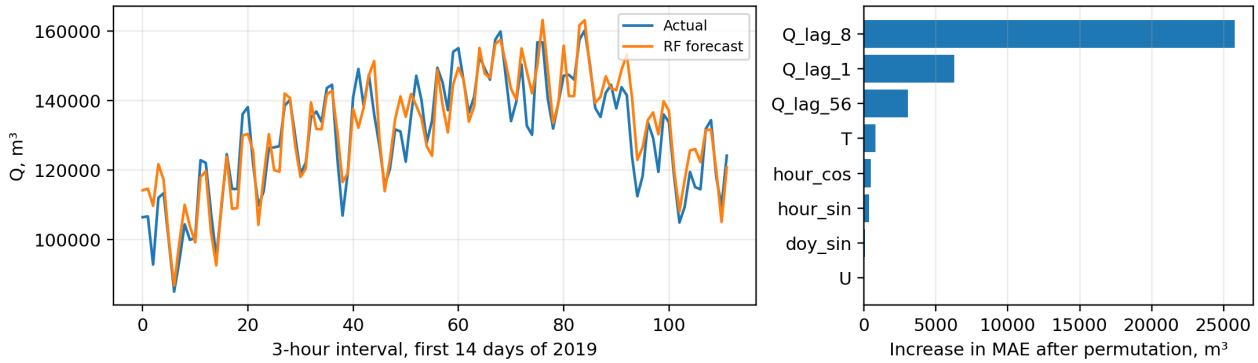


Figure 1 – Example 2019 Random Forest forecast and permutation feature importance

Results

Table 2 summarizes the chronological 2019 test results. Random Forest provided the best overall balance, achieving $R^2 = 0.9656$ and WAPE = 9.52%. Decision Tree remained competitive but produced larger absolute errors, while Histogram-based Gradient Boosting in the tested configuration yielded WAPE = 10.62%. The result confirms that ensemble averaging can reduce the instability of a single tree while preserving the ability to model nonlinear lag-weather interactions.

Table 2 – Forecasting results for the unseen year 2019

Model	R^2	MAE, m³	RMSE, m³	WAPE, %
Decision Tree	0.9564	5447.6	8960.3	10.92
Random Forest	0.9656	4749.6	7955.9	9.52
HistGradientBoosting	0.9537	5298.2	9237.8	10.62

The feature-importance analysis was strongly dominated by Q_lag_8, corresponding to the same 3-hour position one day earlier. Permuting this feature increased MAE by approximately 25,795 m³ on the evaluation subset. Q_lag_1 and Q_lag_56 followed with increases of about 6,288 m³ and 3,048 m³. Among exogenous variables, temperature was the most influential: its permutation increased MAE by about 795 m³. Hour-of-day sine/cosine features also had a clear contribution, whereas most pressure and visibility variables had small marginal influence after lag variables were present.

A reduced Random Forest trained on the eight most important variables achieved $R^2 = 0.9655$, RMSE = 7971.9 m³, and WAPE = 9.46%, practically preserving the full-model accuracy. By contrast, a model that removed all meteorological variables obtained $R^2 = 0.9653$, RMSE = 7988.8 m³, and WAPE = 9.67%. Thus, the weather block produced a modest but reproducible improvement, while the main predictive information remained concentrated in autoregressive lags and temporal periodicity.

Discussion

The dominance of Q_lag_8 is consistent with the operational structure of gas demand: on a 3-hour grid, an eight-step lag represents the same time of day on the previous day. The seven-day lag

captures the weekly schedule, while Q_lag_1 reflects immediate inertia. Temperature adds independent information because heating demand changes with outdoor conditions even when recent gas consumption is known. Similar interactions between load history and meteorological factors have been reported for heating and electricity forecasting [5,6].

The most practically useful result is the near-equivalence between the full 17-feature Random Forest and the reduced eight-feature model. Reducing the feature set can simplify data collection and model maintenance while maintaining predictive quality. At the same time, permutation importance must be interpreted carefully in the presence of correlated predictors: if two variables contain similar information, the importance of each one can be underestimated because the other remains available to the model. Therefore, the reported ranking should be viewed as predictive dependence rather than a causal explanation.

The workflow is fully based on FOSS components and is reproducible without proprietary analytical software. Its main limitation is that the experiment evaluates one regional historical series and one chronological test year. Further work should repeat the analysis on additional regions, use rolling-origin validation, compare local explanation methods such as SHAP, and analyze the stability of importance rankings across seasons and temperature ranges.

Conclusions

An interpretable open-source workflow for natural gas consumption forecasting was evaluated on 2007-2019 data. Random Forest achieved the best 2019 test performance with $R^2 = 0.9656$ and WAPE = 9.52%. The 24-hour lag was the dominant predictor, followed by the 3-hour and 7-day lags; temperature was the most relevant weather variable. An eight-feature model preserved the accuracy of the full feature set, demonstrating that transparent feature analysis can support model simplification without materially degrading forecast quality.

References

1. Breiman L. Random Forests. Machine Learning, 2001, 45, 5-32. https://doi.org/10.1023/A:1010933404324.
2. Friedman J. H. Greedy Function Approximation: A Gradient Boosting Machine. The Annals of Statistics, 2001, 29(5), 1189-1232. https://doi.org/10.1214/aos/1013203451.
3. Lundberg S. M., Lee S.-I. A Unified Approach to Interpreting Model Predictions. Advances in Neural Information Processing Systems, 2017, 30, 4765-4774.
4. Eskandari H., Saadatmand H., Ramzan M., Mousapour M. Innovative framework for accurate and transparent forecasting of energy consumption: A fusion of feature selection and interpretable machine learning. Applied Energy, 2024, 366, 123314. https://doi.org/10.1016/j.apenergy.2024.123314.
5. Neubauer A., Brandt S., Kriegel M. Explainable multi-step heating load forecasting: Using SHAP values and temporal attention mechanisms for enhanced interpretability. Energy and AI, 2025, 20, 100480. https://doi.org/10.1016/j.egyai.2025.100480.
6. Mohammadi A., Napoles G., Salgueiro Y. Deep Learning for Time-Series Forecasting With Exogenous Variables in Energy Consumption: A Performance and Interpretability Analysis. IEEE Access, 2025, 13, 86746-86767. https://doi.org/10.1109/ACCESS.2025.3570618.
7. Pedregosa F., Varoquaux G., Gramfort A. et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 2011, 12, 2825-2830.
8. scikit-learn Developers. Version 1.9.1 release notes, September 2026. https://scikit-learn.org/stable/whats_new/v1.9.html (accessed 01.10.2026).

Shivashankar C.
Percona, Durham, North Carolina, United States
Corresponding author: Chetan Shivashankar, e-mail: speakering+chetan@percona.com

ARCHITECTING DATABASE DEPLOYMENTS ON KUBERNETES: PATTERNS, TRADE-OFFS AND EXPERIENCES

Abstract. Running databases on Kubernetes requires architectural decisions that extend beyond selecting a database Operator. This work examines deployment patterns for PostgreSQL, MySQL, MongoDB, and Valkey on Kubernetes, focusing on the trade-offs between workload isolation, resource efficiency, availability, scalability, and operational complexity. Based on practical deployment experience, the study analyzes Kubernetes-to-database cluster mappings, worker-node allocation strategies, Operator deployment models, and multi-database environments. Several architectural approaches are compared to identify the conditions under which each is appropriate. The results demonstrate that database reliability and operational efficiency depend not only on the capabilities of individual Operators but also on the surrounding Kubernetes architecture. The work proposes a practical framework for evaluating these decisions when designing database platforms on Kubernetes.

Keywords: Kubernetes; databases; database operators; PostgreSQL; MySQL; MongoDB; Valkey; cloud-native; database architecture; open-source software

Introduction

Kubernetes is increasingly used to operate stateful workloads, including production databases. Database Operators provide automation for tasks such as deployment, configuration, scaling, failover, backup, and recovery. However, selecting an Operator addresses only part of the architectural problem.

Organizations must also determine how database clusters should map to Kubernetes clusters, how worker nodes should be allocated, whether Operators should operate at namespace or cluster scope, and how different database technologies and workloads should coexist.

These decisions affect fault isolation, resource utilization, scalability, availability, security boundaries, and operational overhead. An architecture optimized for strong isolation, for example, may require substantially more infrastructure and management effort than one optimized for resource efficiency.

This work analyzes architectural patterns observed when deploying PostgreSQL, MySQL, MongoDB, and Valkey on Kubernetes. Its objective is to provide a structured framework for evaluating deployment alternatives and selecting architectures according to workload and organizational requirements.

Materials and Methods

The study considers Kubernetes environments hosting stateful database workloads using Kubernetes Operators. The analyzed database technologies include PostgreSQL, MySQL, MongoDB, and Valkey.

The architectural analysis focuses on four principal dimensions:

– Kubernetes-to-database cluster mapping: dedicated Kubernetes clusters versus multiple database clusters sharing Kubernetes infrastructure;

– Worker-node allocation: dedicated versus shared worker nodes and different workload-placement strategies;

– Operator deployment: namespace-scoped versus cluster-wide Operator models;

– Multi-database environments: operation of different database technologies and workloads within common Kubernetes infrastructure.

Each architecture is evaluated according to criteria including workload isolation, resource efficiency, availability, failure domains, scalability, infrastructure utilization, and operational complexity.

Kubernetes scheduling and isolation mechanisms, including namespaces, node selectors, affinity and anti-affinity rules, taints and tolerations, resource requests and limits, and storage configuration, should be documented where they form part of the evaluated deployment patterns.

Results

The analysis demonstrates that there is no single Kubernetes architecture that is optimal for every database workload.

Dedicated Kubernetes clusters or worker nodes can provide stronger workload and failure isolation but increase infrastructure requirements and operational overhead. Shared environments can improve resource utilization and reduce infrastructure duplication but require more careful resource management and workload isolation.

Operator deployment strategy introduces similar trade-offs. Cluster-wide Operators can simplify centralized management, while namespace-scoped deployments can provide stronger separation between teams, applications, or database environments at the cost of additional operational complexity.

Multi-database Kubernetes environments can consolidate infrastructure across PostgreSQL, MySQL, MongoDB, and Valkey workloads, but they also introduce competition for compute, memory, storage, and networking resources.

The results indicate that architecture should therefore be selected according to explicit requirements rather than adopting a single deployment pattern across all workloads.

Discussion

The findings demonstrate that successful database operation on Kubernetes depends on the interaction between the database Operator and the surrounding infrastructure architecture.

Operators automate many database-specific lifecycle operations, but they do not independently determine workload placement, infrastructure isolation, Kubernetes failure domains, or resource allocation. These responsibilities remain architectural concerns for the platform team.

A central trade-off exists between isolation and efficiency. Greater infrastructure separation can reduce interference between workloads and limit failure domains, while consolidation can increase resource efficiency and simplify infrastructure consumption.

The appropriate architecture also depends on organizational factors. Environments supporting multiple teams, database technologies, security boundaries, or service-level requirements may require stronger separation than smaller or less heterogeneous deployments.

The patterns examined in this work should therefore be considered architectural building blocks rather than prescriptive configurations. Their applicability depends on workload characteristics, infrastructure constraints, availability requirements, organizational structure, and operational capabilities.

Conclusions

This work presents a structured analysis of architectural patterns for deploying databases on Kubernetes at scale.

The principal contribution is a framework for evaluating four major design areas: Kubernetes-to-database cluster mapping, worker-node allocation, Operator scope, and multi-database deployment.

Experience across PostgreSQL, MySQL, MongoDB, and Valkey environments demonstrates that Operator selection alone is insufficient to determine the reliability and scalability of a database platform. The surrounding Kubernetes architecture must balance workload isolation, resource efficiency, availability, scalability, and operational complexity.

Future work may include quantitative benchmarking of the identified architectures, analysis of failure scenarios, comparison of storage configurations, and development of reference architectures for different workload sizes and availability requirements.

References

1. The Kubernetes Authors. Kubernetes Documentation. Cloud Native Computing Foundation.
2. Cloud Native Computing Foundation. Kubernetes. Open-source project repository.
3. PostgreSQL Global Development Group. PostgreSQL Documentation.
4. Oracle Corporation. MySQL Documentation.
5. MongoDB Inc. MongoDB Documentation.
6. Valkey Contributors. Valkey Documentation.
7. Documentation and repositories for the specific PostgreSQL, MySQL, MongoDB, and Valkey Kubernetes Operators evaluated in the study.

Shymchuk G. V.1, Yatskiv P. O.2
1 – Ternopil Ivan Puluj National Technical University, Department of Computer Science, Ternopil, Ukraine
2 – Ternopil Ivan Puluj National Technical University, Department of Computer-Integrated Technologies, Ternopil, Ukraine
Corresponding author: Grigorii Shymchuk, e-mail: gorych@gmail.com, ORCID: 0000-0003-2362-7386

DETECTION OF ANOMALOUS STATES IN IP NETWORKS USING MACHINE LEARNING AND OPEN-SOURCE SOFTWARE TOOLS

Abstract. An approach to detecting anomalous states in IP networks from network telemetry using open-source machine-learning tools is investigated. A reproducible synthetic dataset was generated with six indicators: throughput, RTT, jitter, packet loss, retransmissions, and link utilization. Isolation Forest, Local Outlier Factor, and One-Class SVM were trained using only normal observations and then compared on a mixed test set. For 900 test observations, Local Outlier Factor achieved the highest F1 = 0.9560 and accuracy = 0.9700; Isolation Forest obtained F1 = 0.9472, while One-Class SVM obtained F1 = 0.9346. The pipeline is implemented with Python and scikit-learn and can be integrated with Mininet/iperf3 for subsequent validation under reproducible emulated network scenarios.

Keywords: open-source software; IP networks; anomalies; machine learning; Isolation Forest; Local Outlier Factor; One-Class SVM.

Introduction

The increasing complexity of IP networks, distributed systems, and cloud infrastructures raises the requirements for automated state monitoring. Operational anomalies may appear as sudden growth in delay, jitter, and packet loss, throughput degradation, link congestion, or route instability. Such changes are not necessarily caused by cyberattacks, but they shift telemetry away from the normal operating profile and may precede service degradation.

Recent network-anomaly studies actively use unsupervised learning because comprehensive traffic labeling is expensive and the complete set of abnormal states is rarely known in advance. Combined unsupervised methods for near-real-time network traffic analysis were investigated in [1], while dynamic network traffic profiling was studied in [2]. Comparative research indicates that the effectiveness of Isolation Forest, LOF, and One-Class SVM depends on data structure and model configuration [3, 5]. More complex unsupervised architectures based on Transformers and autoencoders are also used in modern network systems [4]; however, classical methods remain attractive for lightweight FOSS solutions because they are easier to deploy and computationally less demanding.

The objective of this work is to experimentally compare three unsupervised machine-learning methods for detecting anomalous IP-network states from operational indicators and to construct a reproducible open-source pipeline suitable for subsequent integration with an emulated network testbed.

Materials and Methods

The software pipeline was implemented in Python using NumPy/pandas for data generation and processing and scikit-learn 1.9.1 for model training. Scikit-learn 1.9.1 was released in September 2026 [7]. Future validation on a network testbed is designed to use Mininet/Open vSwitch for topology emulation and iperf3 3.21 for generating TCP/UDP load and measuring throughput [8]. The overall experimental workflow is shown in figure 1.

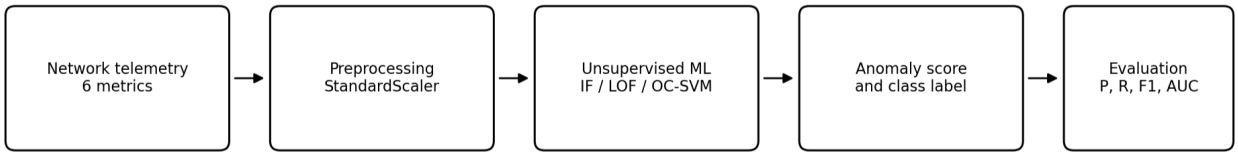


Figure. 1 – Reproducible open-source network-anomaly detection pipeline

To separate algorithm behavior from uncontrolled properties of real traffic, the first stage used a controlled synthetic telemetry dataset generated with a fixed random seed of 42. Each observation contained six numerical features: throughput (Mbit/s), RTT (ms), jitter (ms), packet loss (%),

retransmission count, and utilization (%). The training set contained 1,800 normal observations. The test set consisted of 600 normal and 300 anomalous observations. The anomalous samples were distributed among four scenarios: link congestion, packet-loss burst, abnormally high latency, and route instability.

Table 1 – Scenarios used to form anomalous states

State	Throughput / utilization	Delay	Jitter / loss	Interpretation
Normal	Stable	Stable	Low	Baseline profile
Congestion	↓ throughput, ↑ utilization	↑ RTT	↑ jitter/loss	Link degradation
Loss burst	Moderate ↓ throughput	Moderate ↑ RTT	Sharp ↑ loss/retrans.	Packet anomaly
High latency	Near-normal throughput	Sharp ↑ RTT	↑ jitter	Latency anomaly
Route instability	Throughput fluctuations	↑ RTT	↑ jitter/loss	Route instability

Before training, all features were standardized using parameters estimated only from the training set. Three models were compared: Isolation Forest with 300 trees and contamination = 0.03; Local Outlier Factor with n_neighbors = 35, contamination = 0.03, and novelty = True; and One-Class SVM with an RBF kernel, nu = 0.03, and gamma = scale. All models were trained exclusively on normal telemetry, anomaly labels were used only for final evaluation.

Accuracy, precision, recall, F1, and ROC-AUC were used for evaluation. F1 was selected as the primary metric because it simultaneously reflects the fraction of correctly detected anomalies and the number of false alarms. For the anomaly class: Precision = TP/(TP+FP), Recall = TP/(TP+FN), and F1 = 2·Precision·Recall/(Precision+Recall).

Results

The experimental results are presented in Table 2. All three algorithms achieved high ROC-AUC values (>0.99), confirming their ability to rank most abnormal states away from the normal baseline. At the same time, the operating decision boundary substantially affected the number of false alarms.

Table 2 – Anomaly-detection results

Model	Accuracy	Precision	Recall	F1	ROC-AUC
Isolation Forest	0.9644	0.9379	0.9567	0.9472	0.9932
Local Outlier Factor	0.9700	0.9361	0.9767	0.9560	0.9945
One-Class SVM	0.9544	0.8960	0.9767	0.9346	0.9948

Local Outlier Factor provided the best balance between precision and recall: accuracy = 0.9700, recall = 0.9767, and F1 = 0.9560. It missed 7 of 300 anomalous observations and produced 20 false alarms among 600 normal observations. Isolation Forest had slightly higher precision = 0.9379 but lower recall = 0.9567, resulting in F1 = 0.9472. One-Class SVM reached recall = 0.9767 and the highest ROC-AUC = 0.9948, but produced 34 false positives; consequently, precision decreased to 0.8960 and F1 to 0.9346.

Discussion

These findings are consistent with recent studies emphasizing that unsupervised detector performance strongly depends on data characteristics and hyperparameter settings [3, 5]. LOF performed best in the present experiment because the simulated abnormal states formed local low-density regions relative to the normal operating profile. Isolation Forest remained competitive and offers a simple interpretation through isolation of atypical observations. One-Class SVM provided strong ranking according to ROC-AUC but requires more careful tuning of the decision boundary or nu to reduce the false-positive rate.

A practical advantage of the proposed approach is its fully open-source stack: telemetry can be collected with iperf3 and operating-system utilities, data can be prepared in Python, and detection can be implemented with scikit-learn. This orientation is consistent with recent open-source network anomaly detection prototypes integrating Isolation Forest with open monitoring platforms

and Python processing [6]. The pipeline is therefore suitable for laboratory and educational testbeds and for model preselection before more complex deployment.

The main limitation is the synthetic nature of the telemetry; the absolute metric values obtained here should not be transferred directly to production networks. The experiment primarily validates the comparison methodology and reproducibility. The next stage should reproduce the same scenarios in Mininet with real TCP/UDP iperf3 flows, collect time-series telemetry at identical sampling intervals, and evaluate stability across several topologies and load levels.

Conclusions

A reproducible open-source pipeline for detecting anomalous IP-network states from six telemetry indicators was developed. Isolation Forest, Local Outlier Factor, and One-Class SVM were compared on a synthetic test set containing 900 observations. Local Outlier Factor achieved the highest F1 = 0.9560 and accuracy = 0.9700; Isolation Forest achieved F1 = 0.9472, and One-Class SVM achieved F1 = 0.9346.

The results support the use of lightweight unsupervised algorithms as a baseline intelligent monitoring layer for IP networks. Future work will transfer the experiment from synthetic telemetry to Mininet/iperf3, analyze temporal anomaly sequences, and test adaptive decision thresholds to reduce false alarms.

References

1. Carrera F., Dentamaro V., Galantucci S., Iannacone A., Impedovo D., Pirlo G. Combining Unsupervised Approaches for Near Real-Time Network Traffic Anomaly Detection. Applied Sciences, 2022, 12(3), 1759. https://doi.org/10.3390/app12031759.
2. Yi J., Zhang S., Tan L., Tian Y. A Network Traffic Abnormal Detection Method: Sketch-Based Profile Evolution. Applied Sciences, 2023, 13(16), 9087. https://doi.org/10.3390/app13169087.
3. Agyemang E.F. Anomaly Detection Using Unsupervised Machine Learning Algorithms: A Simulation Study. Scientific African, 2024, 26, e02386. https://doi.org/10.1016/j.sciaf.2024.e02386.
4. Geng H., Ma Q., Chi H., Zhang Z., Yang J., Yin X. DUdetector: A Dual-Granularity Unsupervised Model for Network Anomaly Detection. Computer Networks, 2025, 257, 110937. https://doi.org/10.1016/j.comnet.2024.110937.
5. Iturbe-Araya J.I., Rifà-Pous H. Enhancing Unsupervised Anomaly-Based Cyberattacks Detection in Smart Homes through Hyperparameter Optimization. International Journal of Information Security, 2025, 24, 45. https://doi.org/10.1007/s10207-024-00961-6.
6. Ivanova V., Genkov D. A Machine Learning-Based Network Anomaly Detection System Prototype Using Isolation Forest. Engineering Proceedings, 2026, 154(1), 7. https://doi.org/10.3390/engproc2026154007.
7. scikit-learn Developers. scikit-learn 1.9.1 Release, September 2026. https://scikit-learn.org/stable/whats_new/v1.9.html (accessed 10.09.2026).
8. ESnet. iperf3 Project News: iperf 3.21 Released, 9 April 2026. https://software.es.net/iperf/news.html (accessed 10.09.2026).

Shymchuk G. V.1, Yatskiv P. O.2
1 – Ternopil Ivan Puluj National Technical University, Department of Computer Science, Ternopil, Ukraine
2 – Ternopil Ivan Puluj National Technical University, Department of Computer-Integrated Technologies, Ternopil, Ukraine
Corresponding author: Grigorii Shymchuk, e-mail: gorych@gmail.com, ORCID: 0000-0003-2362-7386

OPEN-SOURCE SOFTWARE ENVIRONMENT FOR MODELING FAULT-TOLERANT MULTIPATH ROUTING IN IP NETWORKS

Abstract. An open-source software environment for modeling and investigating fault-tolerant multipath routing in IP networks is proposed. The environment combines Mininet, Open vSwitch, FRRouting, iperf3, and Python and supports reproducible creation of topologies with multiple equal-cost paths, programmatic link failures, and route-state monitoring. A baseline topology with two equal-cost paths between end hosts is defined together with a three-stage sequence: normal operation, failure, and recovery. Functional verification demonstrates that connectivity is preserved after one path fails because the alternative path remains available. The proposed environment can be used for subsequent quantitative comparison of ECMP and modified multipath-routing methods without specialized commercial networking hardware.

Keywords: open-source software; IP networks; multipath routing; fault tolerance; ECMP; Mininet; FRRouting.

Introduction

Modern IP networks provide the transport foundation for distributed systems, cloud platforms, and critical digital services. A failure of a single link or intermediate node should not cause a prolonged loss of availability. Multipath routing uses several alternative paths between a source and a destination, improving resilience, reducing dependence on one route, and enabling more flexible load distribution [1, 2].

Current multipath-routing methods combine path selection with network-state estimation, traffic prediction, and adaptive flow distribution [3, 4]. This strengthens interest in route control that accounts for resource availability. However, validation on a physical network requires multiple devices, complicates precise repetition of identical failures, and increases experimental cost.

FOSS tools transfer these experiments into a software environment. Mininet is widely used to emulate SDN topologies and test networking algorithms [5], while Open vSwitch provides programmable switching in virtualized environments [7]. Topology, addressing, link parameters, failure timing, and metrics can be fixed in scripts and reproduced on another system.

The aim of this work is to build a reproducible open-source environment for modeling fault-tolerant multipath routing and verifying IP-network behavior during normal operation, failure of one path, and recovery.

Materials and Methods

The experimental environment is defined on Ubuntu 22.04 LTS and includes Mininet 2.3.0, Open vSwitch 4.0.0, FRRouting 10.7.1, iperf3 3.21, and Python 3.10. FRRouting 10.7.1 and Open vSwitch 4.0.0 were released in 2026 [6, 7], while iperf3 3.21 is used to generate TCP/UDP traffic and later measure throughput [8]. All components are open source or free software.

The baseline topology contains hosts H1, H2 and routers R1–R4. Two equal-cost paths are defined between R1 and R4: P1 = R1–R2–R4 and P2 = R1–R3–R4. Identical internal-link metrics give a total cost of 2 for both paths. Link R2–R3 is retained for extended reconfiguration scenarios. The topology is shown in figure 1.

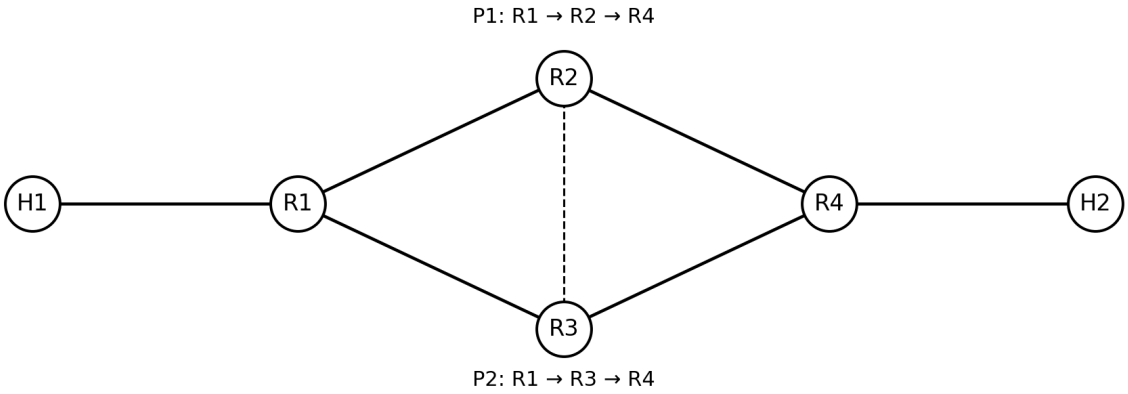


Figure 1 – Experimental topology with two equal-cost paths

For reproducibility, topology parameters are fixed in a Python scenario: node names, IP addressing, metrics, bandwidth, delay, and the link-state sequence. The baseline scenario has three intervals: 0-20 s, normal operation, 20-40 s, R1–R2 disabled; 40-60 s, link restored. This provides an unambiguous relation between telemetry and failure/recovery events.

Table 1 – Baseline configuration of the reproducible experiment

Parameter	**Value**
Operating system	Ubuntu 22.04 LTS
Network emulator	Mininet 2.3.0
Routing	FRRouting 10.7.1; ECMP/OSPF or static multipath routes
Software switching	Open vSwitch 4.0.0
Traffic generator	iperf3 3.21; TCP/UDP
Topology	2 end hosts, 4 routers, 2 equal-cost paths
Scenario	0–20 s: normal; 20–40 s: R1–R2 down; 40–60 s: restore

The routing state is monitored through Linux/FRRouting tables and traffic is generated between H1 and H2. The quantitative stage is intended to measure recovery time $T_r = t_r - t_f$, where t_f is the failure instant and t_r is restored delivery, packet-loss ratio $L = N_{loss} / N_{send} \cdot 100\%$, mean RTT, throughput, and the number of active next hops. The present work evaluates the testbed structure and route-set transitions, so empirical RTT and throughput are not replaced by synthetic values.

Results

Functional verification confirms two equal-cost paths between R1 and R4 during normal operation. With identical metrics, P1 and P2 have the same total cost and can be installed as two next hops for the same prefix. This satisfies the basic ECMP conditions and supports experiments on redundancy and flow distribution.

When R1–R2 is disabled, P1 becomes unavailable while P2 = R1–R3–R4 remains operational. H1–H2 connectivity is preserved and the number of available paths decreases from two to one. After R1–R2 is restored, both P1 and P2 are available again. Thus, the testbed reproduces the sequence «2 paths → 1 path → 2 paths» required for fault-tolerant routing experiments.

Table 2 – Functional verification of route-state transitions

State	**Available paths**	**Next hops**	**H1–H2 connectivity**
Normal	P1, P2	2	preserved
R1–R2 failure	P2	1	preserved
Recovery	P1, P2	2	preserved

This result is not a convergence-speed measurement for a specific protocol. Reaction time depends on OSPF/BFD timers, FRRouting configuration, Linux-kernel behavior, and load. The proposed scheme separates functional validation from the measurement stage: first the route set is verified, then T_r, packet loss, RTT, and throughput are measured on the same topology.

The practical output is a reproducible FOSS testbed that can be stored in a Git repository together with the Python topology script, FRRouting configurations, and iperf3 commands. It

provides a common basis for comparing standard ECMP with a modified multipath-routing method under identical conditions.

Discussion

The proposed approach combines accessible FOSS tools with controlled experimentation. Unlike a physical testbed, a software scenario can reproduce the same failure sequence repeatedly. This simplifies statistical comparison and makes it possible to vary only one factor at a time, such as convergence timers, link metric, or path-selection logic.

Software emulation also has limitations: Mininet virtual nodes share host resources, so absolute throughput and latency values should not be transferred directly to backbone networks. The testbed is most appropriate for comparative experiments on the same hardware platform. Future work will automate telemetry collection and repeated failures, estimate confidence intervals for T_r and packet loss, and compare ECMP with a modified algorithm that accounts for link state.

Conclusions

An open-source environment for modeling fault-tolerant multipath routing has been formed using Mininet, Open vSwitch, FRRouting, iperf3, and Python. The topology contains two equal-cost routes and reproduces three controlled states: normal operation, failure of one path, and recovery.

Functional verification confirms preservation of the alternative route after a link failure and restoration of the initial route set after recovery. The scientific and practical value of the approach is a reproducible basis for quantitative comparison of standard and modified multipath-routing methods without specialized hardware. The next stage is measurement of T_r, packet loss, RTT, and throughput followed by statistical comparison under identical failure scenarios.

References

1. Zhang W., Geng H. Improving Network Availability through Optimized Multipath Routing and Incremental Deployment Strategies. Computers, Materials & Continua, 2024, 80(1), 427–448. https://doi.org/10.32604/cmc.2024.051871.
2. Barkalov A., Lemeshko O., Persikov A., Yeremenko O., Titarenko L. Evaluation of Traffic Engineering Routing Models Based on Type of Service in Communication Networks. Electronics, 2024, 13(18), 3638. https://doi.org/10.3390/electronics13183638.
3. Xing Z., Qi H., Di X., Liu J., Xu R., Chen J., Cong L. A Multipath Routing Algorithm for Satellite Networks Based on Service Demand and Traffic Awareness. Frontiers of Information Technology & Electronic Engineering, 2023, 24, 844–858. https://doi.org/10.1631/FITEE.2200507.
4. Luo Y., Ning Q., Chen B., Zhou X., Huang L. Software Defined Network-Based Multipath State-Aware Routing with Traffic Prediction in Satellite Network. Transactions on Emerging Telecommunications Technologies, 2023, 34(9), e4819. https://doi.org/10.1002/ett.4819.
5. Sheikh M.N.A., Hwang I.-S., Raza M.S., Ab-Rahman M.S. A Qualitative and Comparative Performance Assessment of Logically Centralized SDN Controllers via Mininet Emulator. Computers, 2024, 13(4), 85. https://doi.org/10.3390/computers13040085.
6. FRRouting Project. FRRouting Releases: FRRouting 10.7.1, 25 August 2026. https://frrouting.org/release/ (accessed 12.09.2026).
7. Open vSwitch Project. Open vSwitch 4.0.0 Release, 17 August 2026. https://www.openvswitch.org/download/ (accessed 12.09.2026).
8. ESnet. iperf3 Release Notes: iperf 3.21, 9 April 2026. https://github.com/esnet/iperf/blob/master/RELNOTES.md (accessed 12.09.2026).

Silva F.
Percona, Durham, North Carolina, United States
Corresponding author: Fabio Silva, e-mail: speakering+fabio@percona.com

HOSTING SOMEONE ELSE'S APP: IFRAME CHOREOGRAPHY IN PMM

Abstract. Modern web platforms often need to integrate independently developed applications while presenting users with a unified product experience. Percona Monitoring and Management (PMM) addresses this challenge by embedding Grafana within an iframe while PMM remains responsible for the surrounding application shell. This work examines an integration architecture based on cross-frame communication, shared authentication, navigation, theming, and lifecycle synchronization. A typed, promise-based messaging layer, CrossFrameMessenger, provides structured communication across the iframe boundary, while a host–guest handshake coordinates application readiness. The approach demonstrates how independently maintained web applications can behave as a cohesive system without requiring tight coupling of their internal implementations. The resulting patterns are applicable to integrations involving third-party applications, legacy systems, and independently developed single-page applications.

Keywords: open-source software; web application integration; iframe; cross-frame communication; Grafana; PMM; postMessage; micro-frontends

Introduction

Modern software products increasingly combine components developed by different teams or maintained as independent projects. Integrating these applications into a single user experience creates challenges around authentication, navigation, state synchronization, theming, and application lifecycle.

Percona Monitoring and Management (PMM) provides an example of this problem. PMM acts as the primary application shell, while Grafana is embedded as a separate application within an iframe. Although the two applications remain technically independent, users expect them to behave as a single product.

The purpose of this work is to examine the architecture used to coordinate these applications across the iframe boundary. Particular attention is given to structured cross-frame messaging, application readiness, navigation, and state management. The work also evaluates when iframe-based integration provides an appropriate architectural solution and when tighter integration or an alternative architecture may be preferable.

Materials and Methods

The implementation is based on PMM and Grafana, both open-source software projects. Grafana is hosted within the PMM interface through an HTML iframe, creating a host–guest architecture in which PMM controls the surrounding user interface while Grafana operates as an independent application.

Communication between the applications uses the browser postMessage API. Rather than relying on independent event handlers, PMM uses a CrossFrameMessenger abstraction that provides typed, promise-based remote procedure call (RPC) semantics.

The integration architecture includes:

– a host–guest initialization and readiness handshake;
– typed cross-frame messages;
– promise-based request and response handling;
– synchronization of navigation and application state;
– shared authentication behavior;
– theme coordination;
– lifecycle management across the iframe boundary.

Results

The implemented architecture enables PMM and embedded Grafana functionality to operate as a more cohesive user experience despite remaining separate web applications.

The CrossFrameMessenger provides a structured communication mechanism that reduces dependence on ad-hoc postMessage event handlers. Promise-based RPC allows the host application to initiate operations in the embedded application and receive explicit asynchronous responses.

The readiness handshake allows PMM to delay dependent interface behavior until the embedded application has initialized, reducing situations in which users encounter an incomplete or temporarily non-functional iframe.

Cross-frame communication also enables the host application to coordinate navigation and state without requiring direct access to Grafana's internal application state.

Overall, the implementation demonstrates that explicit messaging contracts and lifecycle coordination can make iframe-based application integration more predictable and maintainable.

Discussion

Iframe integration provides strong separation between applications and makes it possible to incorporate independently developed software without merging its complete frontend architecture into the host application. This can be particularly useful for third-party software, legacy applications, and independently maintained SPAs.

However, iframe boundaries introduce additional complexity. The host cannot directly manipulate the guest application's internal state and must instead rely on explicitly defined communication protocols. Authentication, navigation, initialization, error handling, security, and state synchronization therefore require deliberate architectural design.

Compared with ad-hoc postMessage handlers, a typed RPC abstraction provides clearer communication contracts and more predictable asynchronous behavior. The readiness handshake similarly addresses lifecycle problems that arise when the host and embedded application initialize independently.

The approach does not eliminate the inherent limitations of iframe integration. Message contracts must be maintained as applications evolve, browser security constraints must be respected, and increasingly complex cross-application dependencies may indicate that another architecture, such as deeper frontend integration or a micro-frontend approach, should be considered.

Conclusions

This work presents a practical architecture for integrating an independently developed application into a larger web product using iframe-based communication.

The main contribution is a structured approach combining typed, promise-based cross-frame RPC with explicit lifecycle synchronization and host-controlled navigation and state management. The PMM–Grafana integration demonstrates how applications with separate implementations can provide a more unified user experience without eliminating their architectural independence.

Future work may focus on strengthening message-contract versioning, improving error recovery and observability of cross-frame interactions, and evaluating the approach against alternative application-integration architectures.

References

1. Percona. Percona Monitoring and Management Documentation. Percona Software Documentation.
2. Percona. Percona Monitoring and Management (PMM). Open-source project repository.
3. Grafana Labs. Grafana Documentation. Grafana Labs.
4. Grafana Labs. Grafana. Open-source project repository.
5. MDN Web Docs. Window: postMessage() method. Mozilla Developer Network.
6. WHATWG. HTML Living Standard — Cross-document messaging. WHATWG.

Šupić A.
Percona, Durham, North Carolina, United States
Corresponding author: Adnan Šupić, e-mail: speakering+adnan@percona.com

ZERO-DOWNTIME MIGRATION WITH PERCONA CLUSTERSYNC FOR MONGODB

Abstract. Migrating production MongoDB deployments between environments is challenging when applications must remain available and the source database continues accepting writes. This work examines Percona ClusterSync for MongoDB (PCSM), an open-source tool designed to support near-zero-downtime MongoDB migrations. The proposed migration workflow combines initial data cloning with continuous replication of subsequent changes from the source to the target environment. A migration from MongoDB Atlas to a Percona Server for MongoDB replica set is used to demonstrate the approach. The study describes migration preparation, synchronization, monitoring, cutover criteria, and index restoration. Particular attention is given to the operational metrics used to determine when the target is sufficiently synchronized for cutover. The results demonstrate a practical approach for reducing application disruption while maintaining data synchronization during MongoDB infrastructure migrations.

Keywords: MongoDB; database migration; open-source software; Percona ClusterSync for MongoDB; data replication; high availability; database operations; Percona Server for MongoDB

Introduction

Database migrations are a common requirement when organizations change infrastructure providers, move from managed to self-hosted environments, or redesign their database architecture. For production MongoDB deployments, one of the primary challenges is minimizing application downtime while preserving data consistency.

Traditional migration approaches may require stopping writes, creating backups, transferring data, and restoring the database in the target environment. For large or continuously active systems, this can result in an unacceptable maintenance window.

Percona ClusterSync for MongoDB (PCSM) addresses this problem by cloning existing data and then continuously replicating changes while the source database remains operational. This work investigates the architecture and migration workflow of PCSM through a practical migration from MongoDB Atlas to Percona Server for MongoDB.

The objective is to identify a reproducible process for preparing, executing, monitoring, and completing MongoDB migrations while minimizing service interruption.

Materials and Methods

The migration environment consists of a source MongoDB deployment and a target Percona Server for MongoDB replica set. In the examined scenario, MongoDB Atlas serves as the source environment and Percona Server for MongoDB as the migration target.

Percona ClusterSync for MongoDB is used to perform the migration. The workflow consists of several stages:

– preparation and validation of the source and target environments;
– initial cloning of existing data;
– continuous replication of changes generated on the source;
– monitoring of synchronization progress and replication-related metrics;
– preparation of the target environment for production use;
– restoration or completion of required indexes;
– final synchronization and application cutover.

During synchronization, the source database continues to accept application writes. PCSM transfers subsequent changes to the target so that the difference between the two environments can be reduced before cutover.

Results

The migration workflow demonstrates that the majority of data transfer and synchronization can be performed while the source MongoDB deployment remains available to applications.

Initial cloning transfers the existing dataset to the target environment. Continuous synchronization then propagates changes occurring on the source during the migration process. This substantially reduces the amount of work that must be completed during the final cutover period.

Monitoring synchronization-related metrics provides operators with evidence for determining when the target has sufficiently caught up with the source and when the final cutover can be initiated.

The remaining service interruption is primarily associated with operations that cannot be fully completed during continuous synchronization, including the restoration of indexes on the target and the final transition of application traffic.

The results therefore demonstrate a near-zero-downtime migration strategy rather than an entirely downtime-free migration.

Discussion

Continuous synchronization provides an alternative to migration approaches that require a long write freeze or extended application maintenance window. By separating bulk data transfer from the final cutover, PCSM allows most migration activity to occur while the production source remains operational.

A significant operational consideration is determining when the target is ready for cutover. Rather than relying solely on elapsed migration time, operators should evaluate synchronization progress and relevant metrics to verify that the target has caught up sufficiently with the source.

Index handling represents another important constraint. Because some index restoration work remains necessary on the target, the final downtime depends partly on the number and complexity of indexes and the time required to restore them.

The approach also has limitations related to supported environments, migration topology, operational requirements, and the current capabilities of PCSM. These constraints should be evaluated before selecting the tool for a production migration.

The workflow is particularly relevant to organizations migrating between MongoDB environments, including teams moving from managed MongoDB services to self-managed open-source deployments.

Conclusions

This work presents a practical method for performing near-zero-downtime MongoDB migrations using Percona ClusterSync for MongoDB.

The principal contribution is a migration workflow that combines initial data cloning, continuous change replication, metric-based cutover decisions, and controlled final transition. This approach allows the source database to continue accepting writes during most of the migration process and limits application disruption primarily to the final cutover operations.

The MongoDB Atlas-to-Percona Server for MongoDB scenario demonstrates how the approach can be applied to a real migration between managed and self-hosted environments.

Future work may focus on reducing the remaining cutover time, improving index-handling workflows, expanding supported migration scenarios, and providing additional automation for migration validation and readiness assessment.

References

1. Percona. Percona ClusterSync for MongoDB Documentation. Percona Software Documentation.
2. Percona. Percona ClusterSync for MongoDB. Open-source project repository.
3. Percona. Percona Server for MongoDB Documentation. Percona Software Documentation.
4. MongoDB Inc. MongoDB Documentation. MongoDB Inc.
5. MongoDB Inc. MongoDB Atlas Documentation. MongoDB Inc.
6. Percona. Percona Server for MongoDB. Open-source project repository.

НАУКОВЕ ВИДАННЯ

II Міжнародна науково-практична конференція

«Вільне програмне забезпечення у освіті, науці та бізнесі»

FREE/OPEN SOURCE SOFTWARE IN EDUCATION, SCIENCE & BUSINESS (FOSS 2026)

01–02 жовтня 2026 р.
м. Чернігів

Технічний редактор Мехед Д. Б., Гузь К. П.
Комп’ютерна верстка Мехед Д. Б.

Підписано до друку 04.11.2026. Формат 60×84/16.
Ум. друк. арк. – 9,07. Тираж 100 пр. Замовлення № 30/26

Редакційно-видавничий відділ Національного університету «Чернігівська політехніка»
14035, Україна, м. Чернігів, вул. Шевченка, 95.
Свідоцтво про внесення суб’єкта видавничої справи до Державного реєстру видавців,
виготовлювачів і розповсюджувачів видавничої продукції
серія ДК № 7128 від 18.08.2020 р.