Таблиця 1

Оцінювання результатів виявлення ризикових станів

Показник	Базовий моніторинг	Запропонований підхід
F1-score	0,78	0,89
False Positive Rate	1,00	0,74
Час інтерпретації інциденту	1,00	0,68
Explainability Coverage	-	0,86

Висновки. Запропоновано формалізовану схему пояснюваного AI/ML-виявлення аномалій у мікросервісних та мультихмарних середовищах. Нормовані компоненти E_rec, D_top та R_inc роблять оцінку S(v,t) інтерпретованою і придатною для порівняння вузлів. На практиці це зменшує хибні спрацювання, підвищує якість виявлення інцидентів і пояснює причини ризикового стану.

1. Chandola V., Banerjee A., Kumar V. Anomaly detection: A survey. ACM Computing Surveys. - 2009. - Vol. 41, №3. - P. 1-58.
2. Ahmed M., Mahmood A.N., Hu J. A survey of network anomaly detection techniques. Journal of Network and Computer Applications. - 2016. - Vol. 60. - P. 19-31.
3. Ying R., Bourgeois D., You J., Zitnik M., Leskovec J. GNNExplainer: Generating Explanations for Graph Neural Networks. Advances in Neural Information Processing Systems. - 2019. - P. 9240-9251.

Метрикове оцінювання зменшення технічного боргу JavaScript-коду як передумови підвищення безпеки програмних систем

УДК 004.41:004.8

Ірина Замрій [1], Олексій Кулаков[2]

Державний університет інформаційно-комунікаційних технологій,
[1]i.zamrii@duikt.edu.ua, [2]o.kulakov@stud.duikt.edu.ua

У сучасних JavaScript-проєктах технічний борг проявляється як зниження зв'язності модулів, зростання міжмодульних залежностей та ускладнення трасування змін. У контексті кібербезпеки такі властивості є непрямими чинниками ризику, оскільки ускладнюють аудит коду, статичний аналіз, локалізацію потенційно вразливих ділянок і безпечне внесення виправлень. Роботи з LLM-рефакторингу підкреслюють перспективність автоматизованих змін, але вказують на потребу їх метрикової та функціональної валідації [1–3].

Метою роботи є розроблення компактної процедури метрикового оцінювання зменшення технічного боргу JavaScript-коду після LLM-згенерованих змін для підвищення безпеки програмних систем. Наукова новизна полягає у використанні інтегрального показника, який поєднує зв'язність, залежність і трасувальну невпорядкованість, а рішення про прийняття зміни пов'язує з додатним приростом якості. Такий підхід

узгоджується з розвитком спеціалізованих моделей для коду [4]. Для кількісної перевірки прийнятності кожної LLM-зміни технічний борг подано як інтегральний індекс $S(k)$, що агрегує три нормовані характеристики структури коду:

$$S(k) = 0{,}4 \cdot (1 - Coh(k)) + 0{,}4 \cdot Dep(k) + 0{,}2 \cdot TrEnt(k), \quad (1)$$

де $S(k)$ − інтегральний індекс технічного боргу на k-тій ітерації, $Coh(k)$ − внутрішня зв'язність, $Dep(k)$ − нормована залежність між модулями, $TrEnt(k)$ − ентропія трасування вимог або тестів до коду. Менше значення $S(k)$ відповідає нижчому рівню боргу та кращій структурній готовності коду до аудиту, тестування і безпечного супроводу. Така інтерпретація узгоджується з практикою вимірювання підтримуваності [5].

Експериментальна процедура передбачала такі етапи: 1) обчислення початкових метрик; 2) формування промпту з описом структурних недоліків; 3) отримання LLM-рекомендацій; 4) внесення змін без зміни зовнішньої поведінки; 5) повторне обчислення $S(k)$. Для перевірки використано відкриті бібліотеки axios, commander.js і qs-main.

Таблиця 1

Зміна інтегрального індексу технічного боргу

Проєкт	$Coh_0 \rightarrow Coh_2$	$Dep_0 \rightarrow Dep_2$	$S_0 \rightarrow S_2$	Зменшення S
axios	0,252→0,254	0,101→0,098	0,348→0,346	0,5%
commander.js	0,082→0,097	0,385→0,353	0,521→0,502	3,6%
qs-main	0,104→0,120	0,125→0,077	0,408→0,383	6,2%

Отримані значення показують, що вплив LLM-згенерованих змін залежить від початкового стану системи. Для axios, який має відносно впорядковану структуру, зменшення S є мінімальним. Для commander.js та qs-main ефект виразніший: зменшується Dep, а Coh зростає. У безпековому контексті це означає послаблення непрямих передумов кіберризику, пов'язаних із надмірною зв'язаністю компонентів.

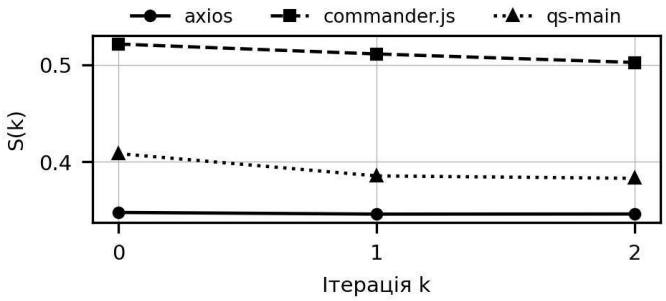

Рис. 1. Динаміка інтегрального індексу $S(k)$ після LLM-згенерованих змін

Практична значущість підходу полягає в тому, що LLM-зміни не приймаються автоматично, тобто кожна ітерація проходить кількісну перевірку,

а негативний або нульовий приріст $\Delta S(k) = S(k-1) - S(k)$ є підставою для відхилення зміни. Це зменшує ризик формального рефакторингу, коли код виглядає простішим локально, але створює нові залежності або ускладнює безпековий аудит.

Висновки. Запропонована процедура дозволяє формалізувати зменшення технічного боргу як вимірюваний процес, у якому LLM виконує роль генератора кандидатних змін, а метрики − роль критерію прийняття. Пілотні результати засвідчили монотонне або майже монотонне зменшення $S(k)$, причому найбільший ефект спостерігається у проєктах з вищою початковою залежністю модулів. Отже, підхід може розглядатися як допоміжний засіб підвищення кіберстійкості, оскільки спрощення структури коду полегшує аудит, тестування і контроль безпечних змін.

1. Martinez S., Xu L., Elnaggar M., Alomar E.A. Software Refactoring Research with Large Language Models: A Systematic Literature Review. Journal of Systems and Software. 2025. URL: https://www.sciencedirect.com/science/article/abs/pii/S0164121225004315.
2. Tornhill A., Borg M., Hagatulah N., Söderberg E. ACE: Automated Technical Debt Remediation with Validated Large Language Model Refactorings. FSE Companion '25: Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering. 2025. P. 1318–1324. DOI: 10.1145/3696630.3730565.
3. Cordeiro J., Noei S., Zou Y. LLM-Driven Code Refactoring: Opportunities and Limitations. 2025 IEEE/ACM Second IDE Workshop (IDE). 2025. P. 32–36. DOI: 10.1109/IDE66625.2025.00011.
4. Rozière B., et al. Code Llama: Open Foundation Models for Code. arXiv:2308.12950. 2023. URL: https://arxiv.org/abs/2308.12950.
5. SonarSource. Understanding measures and metrics. SonarQube Server Documentation. 2026. URL: https://docs.sonarsource.com/sonarqube-server/user-guide/code-metrics/metrics-definition.

Ризик-орієнтований підхід до захисту IoT-пристроїв у муніципальних системах

УДК 004.056:004.738.5:352.07

Олександр Голотенко[1], Сергій Кульчицький[2], Данило Стухляк[3]

Тернопільський національний технічний університет імені Івана Пулюя,
[1]golotenko@gmail.com, [2]serhii_kulchytskyi0212@tntu.edu.ua,
[3]itaniumua@gmail.com

Активне впровадження IoT-пристроїв у муніципальних системах є важливим напрямом цифрової трансформації громад. Такі пристрої застосовуються для екологічного моніторингу, керування освітленням, обліку ресурсів, контролю енергоспоживання, транспортних потоків і роботи комунальних об'єктів. Водночас їхня значна кількість, територіальна