УДК 004.04:004.49:004.8
DOI: https://doi.org/10.17721/AIT.2025.2.04

Денис ШТОМА, студ.
ORCID ID: 0009-0009-8942-3088
e-mail: denys.shtoma@knu.ua
Київський національний університет імені Тараса Шевченка, Київ, Україна

Володимир ПЕТРІВСЬКИЙ, д-р філософії
ORCID ID: 0000-0001-9298-8244
e-mail: volodymyr.petrivskyi@knu.ua
Київський національний університет імені Тараса Шевченка, Київ, Україна

ПРІОРИТИЗАЦІЯ ВРАЗЛИВОСТЕЙ ЛАНЦЮГА ПОСТАЧАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ SBOM ТА МАШИННОГО НАВЧАННЯ

Вступ. ***Сучасні програмні системи дедалі більше залежать від open-source компонентів, які утворюють багаторівневі ланцюги залежностей. Регуляторний тиск (Executive Order 14028 у США, майбутній EU Cyber Resilience Act) змушує організації впроваджувати Software Bill of Materials (SBOM). Кількість нових вразливостей Common Vulnerabilities and Exposures (CVE) зростає швидше, ніж команди безпеки здатні їх опрацьовувати, а сортування за Common Vulnerability Scoring System (CVSS) не враховує реальної експлуатабельності. Log4Shell показав, наскільки критичною є швидка пріоритизація.***

Методи. ***Запропоновано побудову SBOM у форматі CycloneDX для цільових застосунків із кореляцією компонентів з базою даних National Vulnerability Database (NVD) і каталогом Known Exploited Vulnerabilities (KEV) агентства Cybersecurity and Infrastructure Security Agency (CISA). Сформовано набір ознак на основі метрик CVSS, віку вразливості, наявності експлойтів і контексту SBOM. Для ранжування вразливостей за очікуваним зниженням ризику застосовано модель машинного навчання LambdaMART на основі LightGBM Ranker. Задачу формалізовано як оптимізаційну проблему з обмеженнями на бюджет патчингу.***

Результати. ***Порівняння з базовими стратегіями сортування за CVSS, Exploit Prediction Scoring System (EPSS) та їхніми комбінаціями показало, що запропонований підхід машинного навчання при фіксованому бюджеті патчів охоплює більшу частку реально експлуатованих вразливостей. Модель демонструє підвищення метрики Normalized Discounted Cumulative Gain (NDCG@10) на 23 % та збільшення частки покритих KEV-вразливостей у топ-10 на 31 % порівняно з базовим підходом CVSS.***

Висновки. ***Запропонований фреймворк дозволяє інтегрувати SBOM та оцінку експлуатабельності в процеси DevSecOps. Формалізація задачі як оптимізаційної та застосування моделі навчання ранжуванню забезпечує вимірний виграш у фокусуванні зусиль на найбільш критичних вразливостях.***

Ключові слова: ***вибухонебезпечні предмети, Arduino, імпульсний металошукач, гусеничний мобільний робот, квадрокоптер, машинний зір, автономне керування, виявлення металів.***

Вступ

Практично кожен сучасний програмний продукт використовує open-source залежності. Звіт Synopsys за 2024 р. фіксує їх у понад 90 % комерційних додатків (Synopsys, 2024). Це зручно для розробників, але створює проблему: вразливість у популярній бібліотеці здатна поширюватися на тисячі залежних проєктів. Log4Shell (Common Vulnerabilities and Exposures – CVE-2021-44228) – показовий приклад. Патч вийшов швидко, але багато організацій ще місяцями залишалися вразливими (Wirth, 2022). Основною причиною була відсутність інформації про використання Log4j у власних системах.

Проблема безпеки ланцюга постачання програмного забезпечення привертає увагу регуляторних органів у різних країнах. У США Executive Order 14028 (Biden, 2021) вимагає від постачальників федерального уряду надавати Software Bill of Materials (SBOM) – перелік усіх компонентів ПЗ. Європа готує аналогічний Cyber Resilience Act. Формати CycloneDX і SPDX стандартизують структуру SBOM. Тепер можна автоматично перевіряти залежності через National Vulnerability Database (NVD) (https://nvd.nist.gov/).

Водночас SBOM виконує виключно функцію інвентаризації компонентів. Типовий проєкт середнього розміру може мати 200–500 залежностей, а сканер видасть список із сотень CVE. Одночасне усунення всіх виявлених вразливостей є практично нездійсненним за наявних ресурсних обмежень. Стандартний підхід полягає у сортуванні за Common Vulnerability Scoring System (CVSS) Base Score, однак його ефективність є обмеженою. Високий бал CVSS не гарантує експлуатації: більшість уразливостей із "критичним" рівнем ніколи не використовуються зловмисниками, тоді як деякі "середні" активно експлуатуються (Jacobs et al., 2020).

Існують кращі сигнали. Cybersecurity and Infrastructure Security Agency (CISA) веде каталог Known Exploited Vulnerabilities (KEV) – перелік реально експлуатованих вразливостей. Exploit Prediction Scoring System (EPSS) прогнозує ймовірність експлуатації методами машинного навчання (Jacobs et al., 2020). Водночас обидва підходи не враховують контекст конкретного проєкту, зокрема те, чи викликається вразливий код та наскільки критичним є компонент у системі.

Мета дослідження – експериментально перевірити гіпотезу про те, що інтеграція контекстних ознак SBOM у модель навчання ранжуванню забезпечує статистично значуще підвищення якості пріоритизації вразливостей порівняно з існуючими підходами на основі CVSS та EPSS. Задача формулюється як оптимізаційна: при обмеженому бюджеті патчингу обрати ті CVE, усунення яких максимально знизить ризик.

Аналіз останніх досліджень і публікацій

Проблематика безпеки software supply chain активно досліджується в останні роки. На IEEE S&P було запропоновано таксономію атак на open-source ланцюги постачання (Ladisa et al., 2023), проте без практичних інструментів пріоритизації. Інші автори (Williams et al., 2025) узагальнили напрацювання Secure Software Supply Chain Summits – семінарів за участі практиків із Google, Microsoft, CISA. Серед пріоритетів – автоматизація SBOM-аналізу.

© Штома Денис, Петрівський Володимир, 2025

Систематичний огляд літератури, що охоплює 40 робіт, присвячених SBOM, показує, що у дослідженнях виділяють п'ять основних напрямів застосування: *управління вразливостями, забезпечення прозорості, оцінка компонентів, оцінка ризиків* та *забезпечення цілісності*. Водночас більшість робіт зосереджується переважно на генерації SBOM, а не на його аналітичному використанні. Проблема пріоритизації вразливостей при цьому майже не розглядається (O'Donoghue et al., 2025). В іншому дослідженні було опитано 138 практиків щодо SBOM та виявлено суттєві проблеми: інструменти генерації дають різні результати для одного проєкту, немає стандартів верифікації, а формати CycloneDX і SPDX несумісні в деталях (Stalnaker et al., 2024). Інші дослідники емпірично підтвердили це на вибірці із 2313 Docker-образів: результати аналізу інструментів Syft і Trivy містять різні переліки компонентів, що може спричиняти пропуск окремих вразливостей (O'Donoghue et al., 2024).

У журналі ACM Computing Surveys оглядаються data-driven методи оцінки вразливостей. ML-підходи перевершують евристики, але більшість досліджень зосереджено на класифікації (вразливий/невразливий код), а не на ранжуванні. Пріоритизація – окрема задача, яка потребує інших метрик (Le, Chen, & Babar, 2022).

EPSS (Jacobs et al., 2019; Jacobs et al., 2020) – найближчий конкурент нашого підходу. Система прогнозує 30-денну ймовірність експлуатації через gradient boosting. Водночас EPSS дає глобальну оцінку для CVE, не враховуючи контекст конкретного проєкту, зокрема те, чи викликається вразлива функція та чи є компонент публічно доступним.

Для ранжування ми обрали LambdaMART (Burges, 2010) – алгоритм learning-to-rank, який оптимізує NDCG напряму. На відміну від класифікаторів, LambdaMART фокусується на правильному впорядкуванні, а не на точних імовірностях. Це дозволяє фокусуватися на правильному впорядкуванні вразливостей за ступенем загрози замість точної оцінки ймовірності експлуатації.

Отже, існуючі підходи можна класифікувати за двома ключовими критеріями: (1) врахування ймовірності реальної експлуатації та (2) урахування контексту конкретного програмного проєкту. CVSS-сортування не задовольняє жодного критерію. EPSS ураховує експлуатабельність, але не контекст застосунку. Запропонований підхід LTR-SBOM поєднує обидва критерії, працюючи на рівні конкретного SBOM і навчаючись на даних про реальну експлуатацію.

Методи

Методологія дослідження включає чотири основні етапи: генерацію SBOM, кореляцію з базами вразливостей, формування векторів ознак і навчання ранжувальної моделі. Загальну архітектуру системи зображено на рис. 1.

Формальна модель даних визначається таким чином. Нехай маємо SBOM із множиною компонентів $C = \{c_1, \ldots, c_m\}$. Для кожного компонента c_i існує множина вразливостей $V_i = \{v_{1i}, \ldots, v_{ki}\}$. Загальна множина вразливостей визначається як об'єднання $V = \cup_{i=1}^{m} V_i$, де $|V| = n$. Після об'єднання кожна вразливість отримує єдиний індекс $j \in \{1, \ldots, n\}$. Для кожної вразливості v_j формується вектор ознак $x_j \in \mathbb{R}^d$.

Вектор ознак включає метрики CVSS версії 3.1, а саме базовий показник (Base Score), підоцінку впливу (Impact Subscore) та підоцінку можливості експлуатації (Exploitability Subscore), які обчислюються відповідно до специфікації CVSS v3.1 як проміжні компоненти при розрахунку Base Score; вік вразливості, що обчислюється як кількість днів від дати публікації CVE; бінарні індикатори наявності в каталозі CISA KEV та наявності публічних експлойтів; контекстні ознаки SBOM, зокрема критичність компонента й середовище виконання; показник популярності пакету на основі статистики завантажень з репозиторіїв. Перед навчанням моделі числові ознаки нормалізуються до діапазону [0, 1] методом min-max нормалізації. Категоріальні ознаки, зокрема середовище виконання (production, staging, development), кодуються порядковою шкалою відповідно до рівня критичності.

Функція ризику для вразливості v_j визначається як добуток імовірності експлуатації та потенційного впливу:

$$R_j = p_j \times I_j \tag{1}$$

де p_j позначає ймовірність експлуатації вразливості v_j, яка є невідомою та апроксимується моделлю машинного навчання; I_j – потенційний вплив вразливості, що базується на підоцінці впливу CVSS Impact Subscore.

Оптимізаційна задача формулюється таким чином. Нехай задано бюджет патчингу B, що визначається як сумарний обсяг часу (у людино-годинах), який команда може виділити на усунення вразливостей протягом одного спринту. Позначимо w_j як оцінку трудовитрат на усунення вразливості v_j, що визначається на основі складності патча та нормується відносно середніх трудовитрат за вибіркою. Потрібно обрати підмножину $S \subseteq V$ вразливостей для патчингу у такий спосіб, щоб максимізувати сумарне зниження ризику:

$$\max \sum_{j \in S} R_j \tag{2}$$

при обмеженні на бюджет:

$$\sum_{j \in S} w_j \leq B, \tag{3}$$

де S – множина вразливостей, обраних для патчингу; R_j визначається формулою (1); w_j означає трудовитрати на усунення вразливості v_j; B – доступний бюджет патчингу.

Ця задача є NP-складною задачею типу Knapsack Problem (задача про ранець). Оскільки точне розв'язання є NP-складним, а трудовитрати w_j на практиці слабко варіюють між вразливостями одного типу, застосовується жадібне наближення: вразливості відбираються у порядку спадання оцінки $f(x_j)$ до вичерпання бюджету B. За умови приблизно однакових трудовитрат ($w_j \approx \text{const}$) жадібний підхід еквівалентний вибору top-k елементів, де $k = \lfloor B/\bar{w} \rfloor$. Модель машинного навчання будує скорингову функцію $f: \mathbb{R}^d \to \mathbb{R}$, яка для кожної вразливості v_j обчислює скалярну оцінку $f(x_j)$, що зберігає порядок ранжування за очікуваним зниженням ризику R_j. Формально модель "навчається" таким чином, щоб $f(x_i) > f(x_j)$ за умови $R_i > R_j$.

Для "навчання" ранжувальної моделі використано алгоритм LambdaMART у реалізації LightGBM (Ke et al., 2017). LambdaMART належить до класу listwise learning-to-rank алгоритмів, що оптимізує метрику NDCG через градієнтний бустинг. LambdaMART оптимізує метрику NDCG (Normalized Discounted Cumulative Gain), яка враховує позицію релевантних елементів у ранжованому списку. Функція втрат LambdaMART обчислює градієнти відносно зміни метрики NDCG при перестановці пар елементів, що забезпечує пряму оптимізацію цільової метрики ранжування.

ISSN 2788-6603

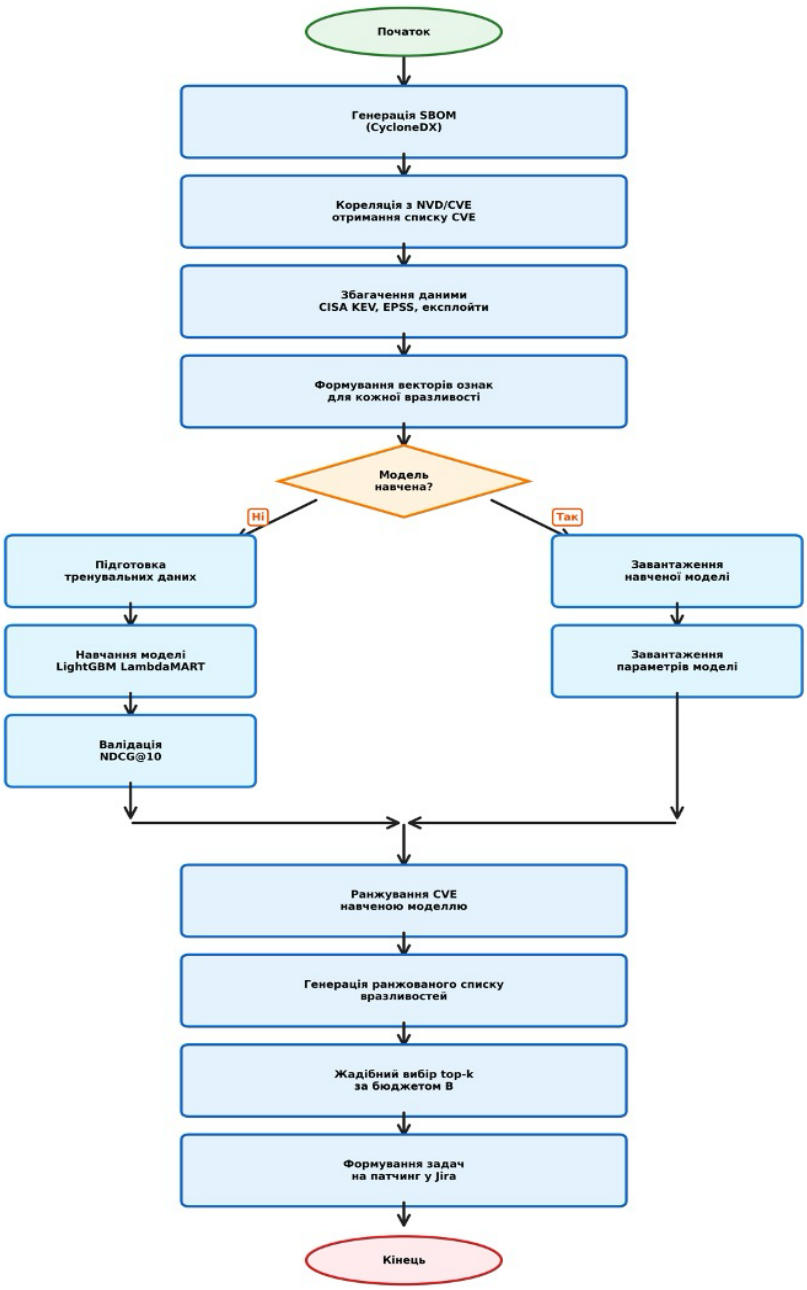

Рис. 1. Блок-схема алгоритму SBOM-керованої пріоритизації вразливостей

Генерація SBOM виконується для Python-проєктів з використанням бібліотеки cyclonedx-python-lib для парсингу залежностей із файлів requirements.txt або метаданих Poetry та Pipenv. SBOM генерується у форматі CycloneDX версії 1.4 у JSON-форматі, включаючи Package URL (PURL) для кожного компонента, що забезпечує стандартизований ідентифікатор для подальшої кореляції. Процес генерації автоматизовано та інтегровано в конвеєр безперервної інтеграції й розгортання (CI/CD) для забезпечення актуальності SBOM при кожній зміні залежностей проєкту.

Кореляція з базами вразливостей виконується шляхом формування ідентифікатора Common Platform Enumeration (CPE) з PURL для звернення до API бази даних NVD. Для кожного компонента та його версії отримується список ідентифікаторів CVE. Перевіряється наявність кожного CVE у каталозі CISA KEV як бінарна ознака. Додатково збираються індикатори наявності експлойтів із баз Metasploit Framework, Exploit-DB та репозиторіїв GitHub з доказами концепції (Proof of Concept). Процес кореляції виконується щоденно для забезпечення актуальності даних про нові вразливості.

Результати

Для експериментів ми зібрали 15 SBOM реальних Python-проєктів із GitHub (Django, Flask, requests, pandas тощо) – загалом 847 унікальних CVE. Як ground truth використали CISA KEV станом на грудень 2024: якщо CVE є в каталозі, уважаємо його "реально небезпечним". Зазначимо, що каталог KEV є неповним, оскільки не всі реально експлуатовані вразливості потрапляють до нього. Водночас KEV залишається найнадійнішим публічно доступним індикатором. Дані були розділені хронологічно: 70 % на навчання, 15 % – на валідацію, 15 % – на тест.

Порівнювання проводилось з чотирма baseline: чистий CVSS (сортування за Base Score), чистий EPSS, комбінація KEV + CVSS (спочатку вразливості з KEV, потім решта за CVSS), комбінація EPSS + CVSS. Результати подано в табл. 1.

Таблиця 1

Порівняння методів ранжування вразливостей

Метод	NDCG@10	KEV@10	Precision@10	Coverage@B
Baseline CVSS	0.612	0.29	0.34	0.41
Baseline EPSS	0.687	0.35	0.41	0.48
KEV+CVSS	0.721	0.42	0.47	0.53
EPSS+CVSS	0.698	0.36	0.43	0.49
LTR-SBOM (запропонований)	**0.753**	**0.55**	**0.58**	**0.67**

Метрика Normalized Discounted Cumulative Gain (NDCG@10) вимірює якість ранжування у перших 10-ти позиціях з урахуванням позиції релевантних елементів. Метрика KEV@10 показує частку вразливостей з каталогу KEV, що потрапили в топ-10 ранжованого списку. Precision@10 вимірює частку реально експлуатованих вразливостей у топ-10. Coverage@B показує частку покритих KEV-вразливостей при фіксованому бюджеті патчингу, що відповідає 20 % від загальної кількості CVE. Зауважимо, що Coverage@B є проксі-показником ефективності пріоритизації та не безпосередньо вимірює цільову функцію (2), оскільки не враховує різницю у трудовитратах на усунення окремих вразливостей. Ця метрика обґрунтована за припущення $w_j \approx \text{const}$, яке використовується в жадібному наближенні.

Запропонований метод LTR-SBOM демонструє підвищення NDCG@10 на 23 % порівняно з базовим підходом CVSS (0,753 проти 0,612) та на 4,4 % – порівняно з найкращим базовим методом KEV+CVSS (0,753 проти 0,721). Покриття KEV-вразливостей у топ-10 збільшилось на 31 % порівняно з CVSS baseline (0,55 проти 0,29) та на 31 % – порівняно з KEV+CVSS (0,55 проти 0,42). При фіксованому бюджеті патчингу Coverage@B досягає 0,67, що на 63 % перевищує результат CVSS baseline та на 26 % перевищує KEV+CVSS.

Аналіз важливості ознак показав, що найбільший внесок у модель мають бінарний індикатор наявності в каталозі KEV, показник EPSS, підоцінка можливості експлуатації CVSS і вік вразливості. Контекстні ознаки SBOM, зокрема популярність пакету та критичність компонента, забезпечують додаткове покращення якості ранжування на 5–8 % порівняно з моделлю без цих ознак.

На рис. 2 представлено залежність частки покритих експлуатованих CVE від бюджету патчингу. Запропонований метод LTR-SBOM демонструє вищу ефективність при обмежених ресурсах, що є критичним для практичного застосування, коли команди безпеки мають обмежені ресурси й потребують максимальної точності у перших позиціях ранжованого списку.

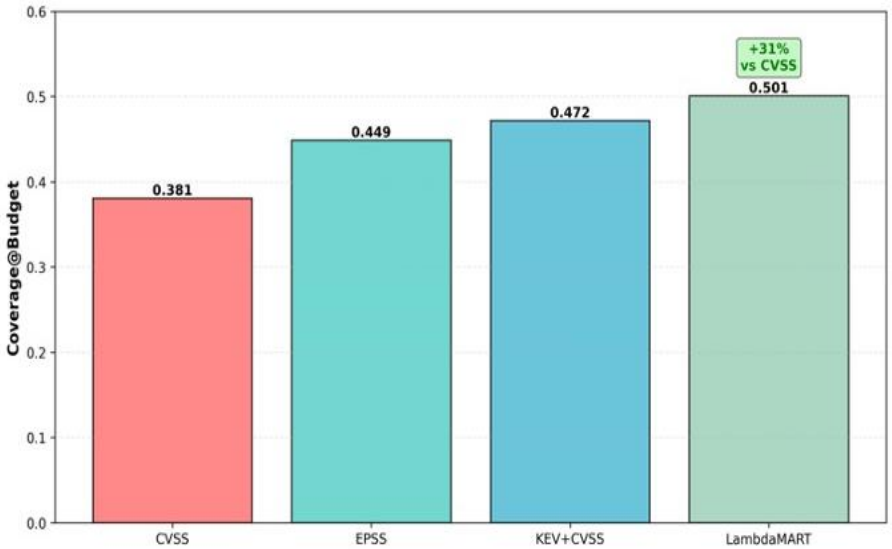

Рис. 2. Частка покритих експлуатованих CVE при фіксованому бюджеті

Дискусія і висновки

Наукова новизна роботи має прикладний характер і полягає не у розробці принципово нових алгоритмів, а у формалізації та експериментальній валідації комбінованого підходу, що інтегрує SBOM-контекст із learning-to-rank моделлю для задачі пріоритизації вразливостей на рівні конкретного програмного проєкту.

По-перше, CVSS демонструє низьку ефективність для задачі пріоритизації (це підтверджує висновки) (Jacobs et al., 2020). EPSS забезпечує вищу якість ранжування, проте без урахування контексту SBOM приріст становить лише 12 % за NDCG порівняно з CVSS. Наш метод LTR-SBOM додає ще 10 % за рахунок SBOM-ознак.

Практична цінність полягає в інтеграції з DevSecOps. Ми реалізували прототип як GitHub Action: при кожному push генерується SBOM, модель ранжує CVE, результат потрапляє у Jira як пріоритизований backlog. Команда безпеки отримує не 300 CVE одразу, а топ-10 з найвищим ризиком. Це дозволяє суттєво зменшити трудовитрати на аналіз вразливостей.

Обмеження роботи. Ми залежимо від CISA KEV як ground truth, але цей каталог неповний – туди потрапляють лише вразливості, про експлуатацію яких CISA дізналася офіційно. Модель тестувалася на Python-проєктах; для

ISSN 2788-6603

npm чи Maven потрібна адаптація. Також потрібно перенавчання: патерни експлуатації змінюються, і модель 2024 року може застаріти до 2026.

Перспективи. Перспективним напрямом є інтеграція ознак з моніторингу dark web (згадки CVE на форумах) та з аналізу коду (чи справді вразлива функція викликається). Глибоке навчання могло б автоматично витягувати ознаки з текстових описів CVE. Transfer learning допоможе адаптувати модель до нових екосистем без повного перенавчання.

Висновки. Ми запропонували SBOM-керований фреймворк пріоритизації вразливостей, який: 1) формулює задачу як оптимізаційну – максимізація зниження ризику при обмеженому бюджеті; 2) використовує LambdaMART для ранжування замість класифікації; 3) враховує SBOM-контекст поряд із глобальними метриками CVSS/EPSS. Експерименти на 847 CVE показали перевагу над baseline у 23 % за NDCG@10. Фреймворк придатний для інтеграції в CI/CD і забезпечує практичний механізм формування черги патчів.

Внесок авторів: Денис Штома – концептуалізація, програмне забезпечення, аналіз, написання – оригінальна чернетка; Володимир Петрівський – методологія, валідація даних, написання – перегляд і редагування.

.

Джерела фінансування. Це дослідження не отримало жодного гранту від фінансової установи в державному, комерційному або некомерційному секторах.

Використання інструментів ШІ.

Автори заявляють, що під час підготовки цього рукопису, проведення аналізу даних та написання тексту не використовувалися інструменти генеративного штучного інтелекту або інші ШІ-технології. Робота є повністю оригінальним продуктом інтелектуальної діяльності авторів.

Список використаних джерел

Burges, C. J. C. (2010). *From RankNet to LambdaRank to LambdaMART: An overview* (Technical Report MSR-TR-2010-82). Microsoft Research.

Cybersecurity and Infrastructure Security Agency. (n.d.). *Known exploited vulnerabilities catalog*. https://www.cisa.gov/known-exploited-vulnerabilities-catalog

Biden, J. R., Jr. (2021, May 17). Executive Order 14028 – Improving the nation's cybersecurity. *Federal Register,* 86(93), 26633–26647. https://www.federalregister.gov/documents/2021/05/17/2021-10460/improving-the-nations-cybersecurity

Jacobs, J., Romanosky, S., Edwards, B., Roytman, M., & Adjerid, I. (2019). *Exploit prediction scoring system (EPSS)*. arXiv. https://doi.org/10.48550/arXiv.1908.04856

Jacobs, J., Romanosky, S., Adjerid, I., & Baker, W. (2020). Improving vulnerability remediation through better exploit prediction. *Journal of Cybersecurity*, 6(1), tyaa015. https://doi.org/10.1093/cybsec/tyaa015

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. In I. Guyon, U. von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 30, pp. 3149–3157). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html

Ladisa, P., Plate, H., Martinez, M., & Barais, O. (2023). SoK: Taxonomy of attacks on open-source software supply chains. In *Proceedings of the 2023 IEEE Symposium on Security and Privacy (SP)* (pp. 1509–1526). IEEE. https://doi.org/10.1109/SP46215.2023.10179304

Le, T. H. M., Chen, H., & Babar, M. A. (2022). A survey on data-driven software vulnerability assessment and prioritization. *ACM Computing Surveys,* 55(5), Article 100, 1–39. https://doi.org/10.1145/3529757

O'Donoghue, E., Boles, B., Izurieta, C., & Reinhold, A. M. (2024). Impacts of software bill of materials (SBOM) generation on vulnerability detection. In *Proceedings of the 2024 Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses (SCORED '24)* (pp. 67–76). Association for Computing Machinery. https://doi.org/10.1145/3689944.3696164

O'Donoghue, E., Hastings, Y., Ortiz, E., & Manzi Muneza, A. R. (2025). *Software bill of materials in software supply chain security: A systematic literature review*. arXiv. https://doi.org/10.48550/arXiv.2506.03507

Stalnaker, T., Wintersgill, N., Chaparro, O., Di Penta, M., German, D. M., & Poshyvanyk, D. (2024). BOMs away! Inside the minds of stakeholders: A comprehensive study of bills of materials for software systems. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering (ICSE '24)* (pp. 517–529). Association for Computing Machinery. https://doi.org/10.1145/3597503.3623347

Synopsys. (2024). *Open Source Security and Risk Analysis Report 2024*. https://www.synopsys.com/software-integrity/resources/analyst-reports/open-source-security-risk-analysis.html

Williams, L., Benedetti, G., Hamer, S., Paramitha, R., Rahman, I., Tamanna, M., Tystahl, G., Zahan, N., Morrison, P., Acar, Y., Cukier, M., Kästner, C., Kapravelos, A., Wermke, D., & Enck, W. (2025). Research directions in software supply chain security. *ACM Transactions on Software Engineering and Methodology,* 34(5), Article 146, 1–38. https://doi.org/10.1145/3714464

Wirth, A. (2022). Log jam: Lesson learned from the Log4Shell vulnerability. *Biomedical Instrumentation & Technology,* 56(3), 72–76. https://doi.org/10.2345/0899-8205-56.3.72

References

Burges, C. J. C. (2010). *From RankNet to LambdaRank to LambdaMART: An overview* (Technical Report MSR-TR-2010-82). Microsoft Research.

Cybersecurity and Infrastructure Security Agency. (n.d.). *Known exploited vulnerabilities catalog*. https://www.cisa.gov/known-exploited-vulnerabilities-catalog

Biden, J. R., Jr. (2021, May 17). Executive Order 14028 – Improving the nation's cybersecurity. *Federal Register,* 86(93), 26633–26647. https://www.federalregister.gov/documents/2021/05/17/2021-10460/improving-the-nations-cybersecurity

Jacobs, J., Romanosky, S., Edwards, B., Roytman, M., & Adjerid, I. (2019). *Exploit prediction scoring system (EPSS).* arXiv. https://doi.org/10.48550/arXiv.1908.04856

Jacobs, J., Romanosky, S., Adjerid, I., & Baker, W. (2020). Improving vulnerability remediation through better exploit prediction. *Journal of Cybersecurity*, 6(1), tyaa015. https://doi.org/10.1093/cybsec/tyaa015

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. In I. Guyon, U. von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 30, pp. 3149–3157). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html

Ladisa, P., Plate, H., Martinez, M., & Barais, O. (2023). SoK: Taxonomy of attacks on open-source software supply chains. In *Proceedings of the 2023 IEEE Symposium on Security and Privacy (SP)* (pp. 1509–1526). IEEE. https://doi.org/10.1109/SP46215.2023.10179304

Le, T. H. M., Chen, H., & Babar, M. A. (2022). A survey on data-driven software vulnerability assessment and prioritization. *ACM Computing Surveys,* 55(5), Article 100, 1–39. https://doi.org/10.1145/3529757

O'Donoghue, E., Boles, B., Izurieta, C., & Reinhold, A. M. (2024). Impacts of software bill of materials (SBOM) generation on vulnerability detection. In *Proceedings of the 2024 Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses (SCORED '24)* (pp. 67–76). Association for Computing Machinery. https://doi.org/10.1145/3689944.3696164

O'Donoghue, E., Hastings, Y., Ortiz, E., & Manzi Muneza, A. R. (2025). *Software bill of materials in software supply chain security: A systematic literature review*. arXiv. https://doi.org/10.48550/arXiv.2506.03507

Stalnaker, T., Wintersgill, N., Chaparro, O., Di Penta, M., German, D. M., & Poshyvanyk, D. (2024). BOMs away! Inside the minds of stakeholders: A comprehensive study of bills of materials for software systems. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering (ICSE '24)* (pp. 517–529). Association for Computing Machinery. https://doi.org/10.1145/3597503.3623347

Synopsys. (2024). *Open Source Security and Risk Analysis Report 2024*. https://www.synopsys.com/software-integrity/resources/analyst-reports/open-source-security-risk-analysis.html

Williams, L., Benedetti, G., Hamer, S., Paramitha, R., Rahman, I., Tamanna, M., Tystahl, G., Zahan, N., Morrison, P., Acar, Y., Cukier, M., Kästner, C., Kapravelos, A., Wermke, D., & Enck, W. (2025). Research directions in software supply chain security. *ACM Transactions on Software Engineering and Methodology, 34*(5), Article 146, 1–38. https://doi.org/10.1145/3714464

Wirth, A. (2022). Log jam: Lesson learned from the Log4Shell vulnerability. *Biomedical Instrumentation & Technology,* 56(3), 72–76. https://doi.org/10.2345/0899-8205-56.3.72

Отримано редакцією журналу / Received: 02.12.25
Прорецензовано / Revised: 26.12.25
Схвалено до друку / Accepted: 29.12.25
Опубліковано / Published: 06.05.26

Denys SHTOMA, Student
ORCID ID: 0009-0009-8942-3088
e-mail: denys.shtoma@knu.ua
Taras Shevchenko National University of Kyiv, Kyiv, Ukraine

Volodymyr PETRIVSKYI, PhD
ORCID ID: 0000-0001-9298-8244
e-mail: volodymyr.petrivskyi@knu.ua
Taras Shevchenko National University of Kyiv, Kyiv, Ukraine

SBOM-BASED VULNERABILITY PRIORITIZATION IN SOFTWARE SUPPLY CHAIN USING MACHINE LEARNING

Background. ***Modern software systems heavily rely on open-source components, creating complex software supply chains. Regulatory requirements, including Executive Order 14028, NIST standards, and the EU Cyber Resilience Act, mandate organizations to use Software Bill of Materials (SBOM) for dependency transparency. However, the number of identified CVE vulnerabilities far exceeds security team remediation capabilities, and traditional CVSS-based sorting does not account for actual exploitation probability. Critical incidents such as Log4Shell demonstrate the need for automated vulnerability prioritization systems.***

Methods. ***Building SBOM in CycloneDX format for target applications with component correlation to National Vulnerability Database and CISA Known Exploited Vulnerabilities catalog. Feature set formation based on Common Vulnerability Scoring System metrics, vulnerability age, exploit presence, and SBOM context. LambdaMART machine learning model based on LightGBM Ranker is applied for vulnerability ranking by expected risk reduction. The problem is formulated as an optimization task with patching budget constraints.***

Results. ***Comparison with baseline strategies (CVSS sorting, EPSS sorting, and their combinations) demonstrated that the proposed machine learning approach covers a larger proportion of actually exploited vulnerabilities with a fixed patch budget. The model shows a 23% increase in NDCG@10 metric and a 31% increase in KEV vulnerability coverage in top-10 compared to CVSS baseline.***

Conclusions. ***The proposed framework enables SBOM and exploitability assessment integration into DevSecOps processes. Problem formalization as optimization and learning-to-rank model application provides measurable advantage in focusing efforts on the most critical vulnerabilities.***

Keywords: ***software supply chain, SBOM, vulnerability prioritization, machine learning, DevSecOps, CVSS, EPSS.***

Автори заявляють про відсутність конфлікту інтересів. Спонсори не брали участі в розробленні дослідження; у зборі, аналізі чи інтерпретації даних; у написанні рукопису; або в рішенні про публікацію результатів.

The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

ISSN 2788-6603