UDC 004.056 **doi: 10.26906/SUNZ.2026.1.122**

Є. В. Челак, Г. В. Гейко

Національний технічний університет “Харківський політехнічний інститут”, Харків, Україна

КЛАСИФІКАЦІЯ ВРАЗЛИВОСТЕЙ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ФОРМУВАННЯ ОЗНАК ЇХ НАЯВНОСТІ НА ОСНОВІ МЕТОДІВ МАШИННОГО НАВЧАННЯ

Анотація. Об’єктом дослідження є процес виявлення вразливостей у програмному забезпеченні. **Предметом дослідження** є підходи до класифікації вразливостей та методи формування ознак їх наявності, придатних для використання в алгоритмах машинного навчання. **Метою роботи** є систематизація існуючих типів дефектів безпеки та розробка методу витягу і попередньої обробки характеристик, що забезпечують можливість їх формалізованого представлення у вигляді числового простору ознак. У роботі проаналізовано основні принципи класифікації вразливостей за причинами виникнення, способом експлуатації, рівнем прояву та впливом на властивості безпеки. Запропоновано узагальнену систему структурних, поведінкових і контекстних параметрів, які можуть виступати індикаторами потенційних дефектів. Розроблено процедури перетворення різнорідних даних, що включають нормалізацію числових величин, кодування категоріальних характеристик, бінаризацію логічних ознак та зменшення розмірності. **Отримані результати** створюють методичну основу для подальшої розробки інтелектуальних систем автоматизованого виявлення вразливостей програмного забезпечення.

Ключові слова: вразливості програмного забезпечення, класифікація, інформативні ознаки, попередня обробка даних, машинне навчання, інформаційна безпека.

Вступ

Сучасне програмне забезпечення є складними багаторівневими системами, що функціонують у динамічному середовищі та взаємодіють із великою кількістю зовнішніх сервісів, мережевих компонентів і користувачів. Зростання обсягів коду, використання сторонніх бібліотек і постійне оновлення функціональності призводять до підвищення ймовірності появи дефектів безпеки, які можуть бути використані зловмисниками для порушення конфіденційності, цілісності та доступності інформації. У зв’язку з цим проблема своєчасного виявлення вразливостей програмного забезпечення є однією з ключових у сфері комп’ютерної інженерії.

Традиційні підходи до аналізу безпеки, що базуються на експертних перевірках, сигнатурному пошуку або ручному аудиті коду, потребують значних ресурсів та не завжди здатні забезпечити необхідний рівень масштабованості. Крім того, багато сучасних вразливостей мають складну природу та проявляються лише за певних умов виконання програм, що ускладнює їх формалізацію у вигляді жорстких правил. Це стимулює розвиток підходів, орієнтованих на автоматизований аналіз даних та використання методів машинного навчання.

Ефективність застосування алгоритмів машинного навчання безпосередньо залежить від якості опису об’єктів дослідження у просторі ознак. Тому поряд із розробкою моделей класифікації важливою задачею є систематизація типів вразливостей і визначення характеристик, що можуть свідчити про їх наявність у програмному забезпеченні. Формування таких ознак дозволяє перейти від якісного експертного аналізу до кількісних методів оцінювання та створює основу для побудови інтелектуальних систем підтримки прийняття рішень.

Таким чином, актуальним є дослідження, спрямоване на побудову узагальненої класифікації вразливостей програмного забезпечення та визначення набору інформативних параметрів, придатних для подальшого використання у моделях машинного навчання.

Об’єктом дослідження є процес виявлення вразливостей у програмному забезпеченні.

Предметом дослідження є методи класифікації вразливостей та підходи до формування ознак їх наявності, що можуть бути використані алгоритмами машинного навчання.

Метою роботи є систематизація вразливостей програмного забезпечення та розробка підходу до формування інформативного простору ознак для забезпечення можливості їх автоматизованого виявлення із застосуванням методів машинного навчання.

Для досягнення поставленої мети необхідно вирішити такі задачі:

1. Визначити типові групи дефектів безпеки та їх характерні прояви;
2. Сформувати перелік потенційно інформативних ознак для кожного класу;
3. Обґрунтувати можливість використання сформованих ознак у задачах машинного навчання

Класифікація вразливостей

Проблема систематизації вразливостей програмного забезпечення залишається актуальною протягом тривалого часу, оскільки від коректності класифікації залежить ефективність методів їх виявлення, аналізу ризиків та вибору механізмів захисту [1]. Існуючі підходи до поділу вразливостей базуються на різних принципах: джерелі виникнення, способі експлуатації, рівні прояву, типі порушуваних властивостей безпеки або особливостях життєвого циклу програмного забезпечення [2].

У більшості досліджень відзначається, що універсальної класифікаційної схеми не існує, оскільки одна й та сама вразливість може одночасно належати до декількох категорій [3]. Тому на практиці часто

використовуються багаторівневі або ієрархічні структури, які дозволяють поєднати різні аспекти аналізу.

Класифікація за причинами виникнення. Одним із найбільш поширених підходів є поділ вразливостей відповідно до природи помилки, що призвела до їх появи [1, 2]. У межах цього підходу зазвичай виділяють: помилки проєктування архітектури; дефекти реалізації алгоритмів; некоректну обробку виняткових ситуацій; недоліки механізмів автентифікації та авторизації; помилки конфігурації та розгортання. Такий поділ дозволяє встановити взаємозв'язок між етапами життєвого циклу розробки та потенційними загрозами безпеці.

Класифікація за способом експлуатації. Інший підхід орієнтований на механізм використання вразливості зловмисником [3, 4]. У цьому випадку розглядаються: ін'єкційні атаки; переповнення буфера; міжсайтове виконання сценаріїв (XSS); підвищення привілеїв; віддалене виконання коду.

Подібна систематизація є зручною для побудови моделей загроз і оцінювання можливих наслідків атак.

Класифікація за рівнем прояву. Вразливості можуть також групуватися відповідно до рівня програмної або апаратної інфраструктури, на якому вони виникають [4]. Зазвичай виділяють: рівень вихідного коду; рівень програмних компонентів та бібліотек; рівень операційної системи; мережевий рівень; рівень конфігурацій та політик безпеки.

Такий підхід дозволяє точніше визначати відповідальність між розробниками, адміністраторами та операторами систем.

Класифікація за впливом на властивості безпеки. З точки зору результату експлуатації, вразливості традиційно співвідносяться з порушенням конфіденційності, цілісності або доступності інформації [5].

У межах цього підходу вони можуть призводити до: витоку даних; модифікації або знищення інформації; блокування роботи сервісів; отримання несанкціонованого контролю над системою.

Подібна класифікація широко використовується під час аналізу ризиків та визначення пріоритетів усунення дефектів.

Незважаючи на різноманіття існуючих схем, більшість із них орієнтовані на експертний аналіз і не завжди придатні для безпосереднього використання у задачах машинного навчання. Причиною цього є відсутність формалізованих параметрів, які можна було б кількісно виміряти або автоматично отримати з процесу функціонування програмного забезпечення [1–5]. Тому актуальною задачею є перехід від описових категорій до таких характеристик, які можуть бути представлені у вигляді набору інформативних ознак. Саме це дозволяє трансформувати класифікацію вразливостей у прикладну основу для побудови інтелектуальних систем їх автоматизованого виявлення.

Формування ознак наявності вразливостей програмного забезпечення

На рис. 1 представлено ієрархію ознак, які можуть бути згруповані на структурні, поведінкові та контекстні ознаки.

Рис 1. Групи ознак наявності вразливостей в програмному забезпеченні

Перехід до використання методів машинного навчання у задачах виявлення вразливостей потребує формалізації характеристик програмного забезпечення у вигляді числових або категоріальних параметрів. На відміну від експертних підходів, де аналіз часто виконується якісно, алгоритми класифікації потребують чітко визначеного простору ознак, що відображає потенційні прояви дефектів безпеки.

Формування таких ознак є складною задачею, оскільки більшість вразливостей мають прихований характер і можуть проявлятися лише під час виконання певних сценаріїв роботи програм. У зв'язку з цим доцільним є використання багаторівневого підходу, який враховує структурні, поведінкові та контекстні характеристики програмного забезпечення. Структурні ознаки описують внутрішні властивості

програмного коду та архітектури системи. Вони можуть бути отримані за допомогою статичного аналізу без виконання програми. Подібні параметри дозволяють виявляти потенційно проблемні ділянки, що статистично частіше пов'язані з наявністю дефектів безпеки.

На відміну від структурних, поведінкові ознаки формуються під час виконання програмного забезпечення та відображають його взаємодію з операційною системою, мережею та іншими компонентами інфраструктури. Поведінкові параметри особливо інформативні для виявлення вразливостей, що не можуть бути встановлені лише шляхом аналізу вихідного коду.

Значна частина ризиків безпеки залежить від умов експлуатації програмного забезпечення. Тому додатково враховуються контекстні характеристики, пов'язані з конфігурацією середовища, версіями компонентів та політиками доступу. У багатьох випадках саме комбінація структурних і контекстних факторів створює передумови для реалізації атаки.

Для ефективного використання у моделях машинного навчання сформований набір параметрів повинен відповідати ряду вимог:

– мати вимірюваний або формалізований характер;
– забезпечувати відтворюваність результатів;
– бути чутливим до змін стану безпеки;
– мінімізувати надлишковість та кореляцію між показниками.

Екстракція та попередня обробка ознак для задач машинного навчання

Ефективність застосування алгоритмів машинного навчання у задачах виявлення вразливостей програмного забезпечення безпосередньо залежить від способу представлення первинної інформації. Дані, що отримуються зі статичного або динамічного аналізу, як правило, мають неоднорідний характер, можуть містити текстові, числові, булеві або категоріальні значення та відрізнятися за масштабом вимірювання.

У зв'язку з цим необхідною є процедура їх перетворення у формалізований вектор ознак, придатний для подальшої автоматизованої обробки.

Нехай програмний об'єкт описується множиною сирих параметрів:

$$R = \{r_1, r_2, \dots, r_m\}, \quad (1)$$

де кожен елемент може належати до різного типу даних.

Завданням попередньої обробки є відображення множини R у простір машинного навчання:

$$\Phi : R \to X, \quad (2)$$

де $X = \{x_1, x_2, \dots, x_n\}$ – вектор числових характеристик фіксованої довжини.

Першим кроком буде перетворення числових параметрів. Для кількісних показників (час виконання, обсяг пам'яті, інтенсивність мережевих запитів) застосовується нормалізація, що забезпечуватиме порівнюваність масштабів різних величин. Одним з поширених підходів є мін-макс перетворення:

$$x_i = \frac{r_i - r_i^{min}}{r_i^{max} - r_i^{min}}. \quad (3)$$

У випадках наявності викидів доцільним може бути використання стандартизації:

$$x_i = \frac{r_i - \mu_i}{\sigma_i}, \quad (4)$$

де μ_i та σ_i – середнє значення та стандартне відхилення відповідної ознаки.

Другим кроком є перетворення категоріальних параметрів, що можуть описувати типи операцій, режими доступу або види використовуваних протоколів. Такі параметри не можуть бути безпосередньо використані більшістю алгоритмів машинного навчання. Тому вони перетворюються у числове подання за допомогою кодування. Нехай параметр r_j може приймати k різних значень. Тоді формується вектор:

$$x_j = \{b_1, b_2, \dots, b_k\}, \quad (5)$$

де $b_t = \begin{cases} 1, \text{якщо реалізовано } t \text{ категорію;} \\ 0, \text{інакше.} \end{cases}$

Такий підхід дозволяє уникнути хибного введення порядкових залежностей між категоріями.

Третім типом будуть перетворення текстових та логічних характеристик. Для таких параметрів, які відображають наявність або відсутність певних властивостей, використовується бінарне відображення. У випадку текстових описів або журналів подій можливе використання частотного представлення, при якому кожен елемент словника відображає кількість його поява у межах аналізованого об'єкта.

Останнім етапом буде формування підсумкового вектору та зменшення розмірності з усуненням надлишковості. Після всіх перетворень буде сформовано єдиний простір ознак *X*. Саме такий формат є стандартним для більшості сучасних алгоритмів класифікації, від нейронних мереж до дерев рішень та систем з підкріпленням.

Оскільки велика кількість параметрів може призводити до перенавчання моделей та збільшення обчислювальної складності, доцільним є виконання процедури відбору найбільш інформативних ознак. Узагальнено це можна представити як перетворення:

$$X \to X', \text{де } \dim(X') < \dim(X). \quad (6)$$

У результаті підвищується стабільність навчання та узагальнююча здатність алгоритмів.

Таким чином, запропонований підхід до витягу та попередньої обробки забезпечує уніфіковане представлення різнорідних характеристик програмного забезпечення у вигляді числового вектора, що створює основу для подальшого застосування методів машинного навчання у задачах автоматизованого виявлення вразливостей.

Висновки

У роботі розглянуто проблему систематизації вразливостей програмного забезпечення в контексті підготовки даних для застосування методів машинного навчання. Проведений аналіз існуючих підходів до класифікації дозволив виділити основні групи дефектів безпеки за причинами виникнення, спосо-

бом експлуатації, рівнем прояву та впливом на базові властивості інформаційних систем.

Показано, що більшість традиційних класифікацій орієнтовані переважно на експертне використання та мають описовий характер, що ускладнює їх безпосередню інтеграцію у автоматизовані процедури аналізу. У зв'язку з цим запропоновано підхід до переходу від якісного представлення вразливостей до формування формалізованого простору ознак, придатного для алгоритмічної обробки. Розроблено метод витягу та попередньої обробки ознак, що забезпечує перетворення різнорідних даних у єдине числове представлення фіксованої розмірності. Запропоновані процедури нормалізації, кодування та зменшення розмірності дозволяють підвищити стабільність подальшого навчання моделей і зменшити вплив надлишкової або некорельованої інформації.

Отримані результати формують методичну основу для побудови автоматизованих систем виявлення вразливостей програмного забезпечення та можуть бути використані під час розробки класифікаційних, прогнозних і аналітичних моделей безпеки.

Перспективним напрямом подальших досліджень є експериментальна перевірка інформативності сформованих ознак на реальних наборах даних, а також розробка ансамблевих підходів, здатних адаптуватися до появи нових типів вразливостей і змін у програмному середовищі.

Конфлікт інтересів. Автори декларують, що не мають конфлікту інтересів стосовно даного дослідження, в тому числі фінансового, особистісного характеру, авторства чи іншого характеру, що міг би вплинути на дослідження та його результати, представлені в даній статті.

Використання засобів штучного інтелекту. Автори підтверджують, що не використовували технології штучного інтелекту при створенні представленої роботи.

Список літератури

1. E. Iannone, R. Guadagni, F. Ferrucci, A. De Lucia and F. Palomba, "The Secret Life of Software Vulnerabilities: A Large-Scale Empirical Study," *IEEE Trans. on Software Eng.*, vol. 49, no. 1, pp. 44-63, 2023, doi: https://doi.org/10.1109/TSE.2022.3140868
2. Q. Mao, Z. Li, X. Hu, K. Liu, X. Xia and J. Sun, "Towards Explainable Vulnerability Detection With Large Language Models," in *IEEE Transactions on Software Engineering*, vol. 51, no. 10, pp. 2957-2971, Oct. 2025, doi: https://doi.org/10.1109/TSE.2025.3605442
3. G. Lin, S. Wen, Q. -L. Han, J. Zhang and Y. Xiang, "Software Vulnerability Detection Using Deep Neural Networks: A Survey," in *Proceedings of the IEEE*, vol. 108 (10), pp. 1825-1848, 2020, doi: https://doi.org/10.1109/JPROC.2020.2993293
4. X. Yin, C. Ni and S. Wang, "Multitask-Based Evaluation of Open-Source LLM on Software Vulnerability," in *IEEE Transactions on Software Engineering*, vol. 50, no. 11, pp. 3071-3087, 2024, doi: https://doi.org/10.1109/TSE.2024.3470333
5. X. Zhang *et al*., "Effectively Detecting Software Vulnerabilities via Leveraging Features on Program Slices," in *IEEE Internet of Things Journal*, vol. 12, no. 7, pp. 8033-8048, 1 April1, 2025, doi: https://doi.org/10.1109/JIOT.2025.3541090
6. V. Chelak, O. Hornostal, Ye. Chelak, S. Gavrylenko. Advanced Methods for Classification Quality Assessment Leveraging ROC Analysis and Multidimensional Confusion Matrix. Advanced Information Systems, 2025, Vol 9(1), pp. 24–34, doi: https://doi.org/10.20998/2522-9052.2025.1.03

Received (Надійшла) 17.11.2025
Accepted for publication (Прийнята до друку) 04.02.2026
Publication date (Дата публікації) 27.02.2026

Відомості про авторів / About the Authors

Челак Єгор Володимирович – аспірант кафедри "Комп'ютерна інженерія та програмування", Національний технічний університет "Харківський політехнічний інститут", Харків, Україна;
Yehor Chelak – PhD Student of Department of "Computer Engineering and Programming", National Technical University «Kharkiv Polytechnic Institute», Kharkiv, Ukraine;
e-mail: egor.chelak@gmail.com; ORCID ID: https://orcid.org/0000-0002-3898-4370.

Гейко Геннадій Вікторович – кандидат технічних наук, доцент кафедри "Комп'ютерна інженерія та програмування", Національний технічний університет "Харківський політехнічний інститут", Харків, Україна;
Hennadii Heiko – Candidate of Technical Sciences, Associate Professor of Department of "Computer Engineering and Programming", National Technical University «Kharkiv Polytechnic Institute», Kharkiv, Ukraine;
e-mail: hennadii.heiko@khpi.edu.ua; ORCID ID: https://orcid.org/0000-0001-6958-8306.

Classification of software vulnerabilities and formation of presence signs based on machine learning methods

Yehor Chelak, Hennadii Heiko

Abstract. The object of the study is the process of vulnerability detection in software. **The subject of the study** includes approaches to vulnerability classification and methods for forming indicators of their presence suitable for machine learning algorithms. **The aim of the work** is to systematize existing types of security defects and to develop a method for feature extraction and preprocessing that enables their formal representation within a numerical feature space. The paper analyzes the main principles of vulnerability classification according to root causes, exploitation techniques, manifestation levels, and impact on security properties. A generalized system of structural, behavioral, and contextual parameters that may serve as indicators of potential defects is proposed. Procedures for transforming heterogeneous data are developed, including normalization of numerical values, encoding of categorical attributes, binarization of logical features, and dimensionality reduction. **The obtained results** provide a methodological basis for the further development of intelligent systems for automated software vulnerability detection.

Keywords: software vulnerabilities, classification, informative features, data preprocessing, machine learning, information security.