УДК 004.056.5

Декалюк Б.О., здобувач,
Ханін Н.В., здобувач,
Чешун Д.В., викладач
Хмельницький фаховий економіко-технологічний коледж УЕП

ЗАСТОСУВАННЯ ТЕХНОЛОГІЙ ВИЯВЛЕННЯ ВРАЗЛИВОСТЕЙ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Вибір методу виявлення вразливостей залежить від багатьох факторів, зокрема від етапу життєвого циклу, наявності вихідного коду та типу вразливостей, які очікується знайти. Існують три основні автоматизовані підходи (SAST, DAST, IAST) та ручне тестування [1-3].

Для класифікації загроз часто використовують загальновизнані стандарти, такі як OWASP Top 10 [4].

Ці стандарти забезпечують єдину мову та методологію для ідентифікації, класифікації та реагування на загрози, роблячи процес тестування безпеки додатків (Application Security Testing) більш структурованим та результативним, а також дозволяють робити порівняльну оцінку ефективності методів виявлення проти класів вразливостей (таблиця 1).

Таблиця 1 – Оцінка ефективності методів проти класів вразливостей

Клас вразливостей	SAST	DAST	IAST	Ручне тестування
Вразливості коду: SQL Ін'єкція, XSS(Cross-Site Scripting) [5]	Висока	Середня / Висока	Середня / Висока	Висока
Вразливості часу виконання, помилки конфігурації сервера, слабке управління сесіями.	Низька	Висока	Висока	Висока
Маніпуляції цінами, обхід етапів авторизації, IDOR [6]	Низька	Низька	Низька	Висока

Базуючись на аналізі методів та враховуючи технічні можливості інфраструктури можна сформулювати практичні пропозиції та рекомендації для ефективного виявлення вразливостей програмного забезпечення, що полягають у впровадженні циклічного процесу тестування безпеки.

Практичне застосування DAST є першим кроком для розгорнутого застосунку. В умовах лабораторії це реалізується розгортанням навмисно вразливого вебзастосунку у ізольованій «пісочниці» Кіберполігону.

Використання утиліти OWASP ZAP, яка надсилає тисячі тестових запитів до всіх виявлених точок входу, намагаючись знайти поширені вразливості є другим етапом. Після завершення роботи утиліти, фахівець має вручну перевірити кожну знайдену вразливість "високого" та "середнього" ризику, підтвердивши її реальну експлуатованість.

Застосування SAST та IAST буде ефективним, якщо є доступ до вихідного коду досліджуваного ПЗ. Паралельно запускається SAST-інструмент SonarQube або Snyk. Це дозволяє знайти вразливості, які DAST міг пропустити, що найважливіше – точно вказати на проблемний рядок коду, що значно прискорює виправлення.

Для запобігання XSS всі дані, що надходять від користувача, мають проходити валідацію на дозволені символи. Всі дані, що відображаються на сторінці, мають проходити контекстно-залежне кодування HTML, URL та JavaScript.

Автоматизація запуску SAST та DAST-сканерів у процесі безперервної інтеграції дозволяє виявляти вразливості при кожній зміні коду, що відповідає сучасним підходам до кіберзахисту.

Список використаних джерел:

1. Understanding Industry Perspectives of Static Application Security Testing (SAST) / Yuan Li et al. Proceedings of the ACM on Software Engineering. Volume 2, Issue FSE. Article № FSE134. P.3033-3056. DOI: https://dl.acm.org/doi/abs/10.1145/3729404.

2. DAST: Difficulty-Adaptive Slow-Thinking for Large Reasoning Models / Yi Shen et al. arXiv. 2025. №2503.04472v2. DOI: https://arxiv.org/abs/2503.04472.

3. Anoop Gupta, Sivakumar Ponnusamy. Cybersecurity and Ethical Hacking Harnessing AI. IJGIS. November 2024. DOI: https://doi.org/10.21428/e90189c8.35100e4b.

4. System approach to web application security: analysis of threats and methods of cyber protection. A. Ilienko et al. Ukrainian Information Security Research Journal. 2024. Vol. 26, №2. P. 277-293. URL: https://h7.cl/1iBLW.

5. Advancing XSS Detection in IoT over 5G: A Cutting-Edge Artificial Neural Network Approach / Rabee Alqura'n et al. IoT. 2024. №5(3). P. 478-508. DOI: https://www.mdpi.com/2624-831X/5/3/22.

6. Bhutani V., Ghassemi Toosi F., Buckley J. Analysing the Analysers: An Investigation of Source Code Analysis Tools. Applied Computer Systems. 2024. Vol. 29, Is. 1. DOI: https://h7.cl/1iC7i.