УДК 004.056 : 004.738.5 https://doi.org/10.15407/pp2025.04.041

С.В. Поперешняк, Б.Д. Горох

ПІДХІД ДО ПОШУКУ ВРАЗЛИВОСТЕЙ ВЕБСАЙТІВ НА ОСНОВІ СТАТИЧНОГО Й ДИНАМІЧНОГО АНАЛІЗУ

У статті запропоновано підхід до автоматизованого пошуку вразливостей вебсайтів на основі поєднання статичного та динамічного аналізу в межах модульної архітектури сканера. Дослідження зумовлене зростанням кількості параметризованих URL у сучасних вебзастосунках і, як наслідок, надлишковість обходів та висока вартість багатоваріантних перевірок при обмежених часових і ресурсних бюджетах у сценаріях DevSecOps/CI/CD. Підхід базується на двоетапному конвеєрі: попередній статичний аналіз вебресурсу, що включає побудову карти сайту пошуковим роботом із контролем глибини, вилучення кінцевих точок, параметрів і форм введення, а також нормалізацію URL-шаблонів через узагальнення динамічних ідентифікаторів; динамічне тестування вразливостей для нормалізованої множини тестових точок із паралельним виконанням ізольованих перевірок та агрегацією результатів у формати, придатні для автоматизованого оброблення. Запропоновано метрики оцінювання якості та продуктивності, зокрема precision/recall, коефіцієнт редукції запитів і пропускну здатність, що дозволяють кількісно оцінити ефект попередньої нормалізації та ефективність багатопотокової обробки. Реалізацію виконано як CLI-утиліту на Java з плагінною моделлю тестів, що спрощує розширення системи під нові класи вразливостей без модифікації ядра. Експериментальну перевірку проведено на еталонних вразливих застосунках OWASP Juice Shop і OWASP WebGoat та на власних проєктах; показано суттєве скорочення часу роботи пошукового робота та досягнення прийнятної пропускної здатності залежно від умов розгортання. Отримані результати підтверджують доцільність комбінування статичного структурування простору пошуку з цільовими динамічними перевірками для підвищення масштабованості та відтворюваності аналізу безпеки вебресурсів.
Ключові слова: веббезпека; вразливості; статичний аналіз; динамічний аналіз; URL-нормалізація; crawler; модульна архітектура; багатопотокове сканування; IoT, DevSecOps.

S. Popereshnyak, B. Horokh

AN APPROACH TO WEBSITE VULNERABILITY DETECTION BASED ON STATIC AND DYNAMIC ANALYSIS

This paper proposes an approach to automated website vulnerability detection based on the combination of static and dynamic analysis within a modular scanner architecture. The motivation for this study arises from the growing number of parameterized URLs in modern web applications and, as a consequence, redundant crawling and the high cost of multi-variant testing under limited time and resource budgets in DevSecOps/CI/CD scenarios. The proposed approach is built on a two-stage pipeline: preliminary static analysis of a web resource, which includes sitemap construction by a crawler with depth control, extraction of endpoints, parameters, and input forms, as well as URL template normalization through the generalization of dynamic identifiers; and dynamic vulnerability testing for a normalized set of test points with parallel execution of isolated checks and aggregation of results into machine-readable formats. Quality and performance evaluation metrics are proposed, including precision/recall, the request reduction ratio, and throughput, which enable quantitative assessment of the impact of preliminary normalization and the efficiency of multithreaded processing. The implementation is realized as a Java-based CLI utility with a plugin-based testing model, facilitating extensibility for new vulnerability classes without modification of the core system. Experimental validation was conducted using benchmark vulnerable applications OWASP Juice Shop and OWASP WebGoat, as well as proprietary projects; the results demonstrate a significant reduction in crawler execution time and the achievement of acceptable throughput depending on deployment conditions. The obtained results confirm the effectiveness of combining static structuring of the search space with targeted dynamic checks to improve the scalability and reproducibility of web security analysis.
Key words: web security; vulnerabilities; static analysis; dynamic analysis; URL normalization; crawler; modular architecture; multithreaded scanning; IoT; DevSecOps.

Вступ

Вебзастосунки та вебсайти залишаються одними з основних цілей кібератак через високу динамічність коду, залежність від сторонніх компонентів і складні ланцюги обробки користувацького введення. Типові класи вразливостей, зокрема, ін'єкції, XSS, помилки конфігурації, слабкі HTTP-заголовки та SSRF проявляються як на рівні логіки застосунку, так і на рівні розгортання та мережевої взаємодії, що зумовлює необхідність поєднання різних методів аналізу.

Поширення практик DevSecOps і CI/CD актуалізує потребу в автоматизованих і відтворюваних інструментах оцінювання безпеки вебресурсів. Водночас на практиці сканери стикаються з надлишковим обходом великої кількості параметризованих URL та високою вартістю багатоваріантних перевірок для різних класів вразливостей за обмежених часових і ресурсних бюджетів.

У роботі запропоновано підхід до пошуку вразливостей вебсайтів на основі комбінування статичного і динамічного аналізу в межах модульної архітектури сканера. Підхід передбачає попередню побудову карти сайту за допомогою crawler'а з обмеженням глибини, нормалізацію URL-шаблонів для зменшення дублювання логічно еквівалентних сторінок, багатопотоковий запуск ізольованих тестів вразливостей та агрегування результатів у форматі, придатному для подальшого аналізу й інтеграції.

Актуальність розроблення таких систем зумовлена зростанням кількості й складності атак на вебресурси, які є критичними компонентами сучасної інформаційної інфраструктури та обробляють конфіденційні дані. Автоматизовані засоби пошуку вразливостей дозволяють зменшити трудомісткість ручного аудиту, підвищити регулярність перевірок і сприяють впровадженню безпечних практик розроблення програмного забезпечення, що є ключовим чинником забезпечення стійкості вебзастосунків.

Аналіз останніх досліджень і публікацій.

У сучасних дослідженнях із безпеки вебзастосунків простежується стійка тенденція до інтеграції засобів автоматизованого аналізу у процеси DevSecOps та CI/CD, а також до поєднання статичних і динамічних методів виявлення вразливостей. Значна кількість робіт зосереджена на подоланні обмежень окремих підходів, зокрема високого рівня хибнопозитивних спрацювань статичного аналізу та ресурсомісткості динамічного тестування.

У роботі [1] проведено системний аналіз проблеми хибнопозитивних спрацювань у статичному аналізі, окреслено сучасні підходи до її зменшення та підкреслено доцільність комбінування SAST із іншими видами аналізу для підвищення практичної цінності результатів. Ефективність динамічного аналізу вебзастосунків досліджується в роботі [2], де показано, що результативність чорних сканерів істотно залежить від якості виявлення кінцевих точок і параметрів, а також від обмежень на кількість HTTP-запитів.

Концептуальні засади поєднання статичного й динамічного аналізу узагальнено у [3], де обґрунтовано підхід змішаного аналізу безпеки як способу підвищення масштабованості та відтворюваності перевірок безпеки вебзастосунків. Практичні аспекти застосування шаблонного статичного аналізу для виявлення типових вразливостей OWASP Top 10 розглянуто у роботі [4], що підтверджує доцільність використання легковагових статичних методів як попереднього фільтра.

Порівняльний огляд сучасних параметрів безпеки вебзастосунків і тенденцій їх розвитку наведено у дослідженні [5], де підкреслюється необхідність переходу до модульних і адаптивних засобів аналізу безпеки. Роботи [6], [7] присвячені застосуванню інтелектуальних систем і методів машинного навчання для виявлення вторгнень та аномалій, що демонструє потенціал таких підходів для зменшення кількості хибних спрацювань і підвищення точності детекції.

Питання інтерпретації та візуалізації результатів динамічного тестування у складних багатопроєктних середовищах розглянуто в роботі [8], а використання методів машинного навчання для виявлення вебатак – у дослідженні [9]. Ці роботи підкреслюють актуальність поєднання класичних методів аналізу з інтелектуальними підходами та структурованим поданням результатів.

Таким чином, аналіз сучасних публікацій показує, що актуальною науково-практичною задачею залишається інженерно обґрунтоване комбінування статичного структурування простору пошуку з цільовими динамічними перевірками в межах модульних і масштабованих архітектур. Запропонований у даній роботі підхід відповідає цим тенденціям і розвиває їх у напрямку зменшення надлишкових перевірок та підвищення ефективності автоматизованого аналізу безпеки вебресурсів.

Мета та завдання дослідження. Метою роботи є розроблення та обґрунтування підходу до пошуку вразливостей вебсайтів на основі поєднання статичного та динамічного аналізу в модульній архітектурі сканера, який зменшує надлишкові запити за рахунок нормалізації URL-шаблонів, забезпечує масштабоване багатопотокове виконання тестів та формує результати у вигляді, придатному для інтеграції з процесами забезпечення якості та безпеки програмного забезпечення.

Виклад основного матеріалу

Вимоги до програмного забезпечення та обґрунтування технологічних рішень. Сучасні вебзастосунки є базовим середовищем реалізації бізнес-процесів у сферах електронної комерції, фінансових послуг, електронного урядування та цифрових сервісів. Водночас зростання функціональної складності вебресурсів супроводжується підвищенням рівня ризиків, зумовлених як технічними чинниками, так і людським фактором. Незважаючи на значний розвиток засобів захисту, найбільш поширеними залишаються такі класи вразливостей, як SQL-ін’єкції, міжсайтове виконання сценаріїв (XSS), підробка міжсайтових запитів (CSRF), а також помилки в механізмах автентифікації та авторизації. Ефективне виявлення зазначених вразливостей потребує використання спеціалізованих програмних засобів, огляд яких було здійснено в попередньому розділі.

Аналіз наявних інструментів показує, що більшість із них або орієнтовані на вузьке коло задач, або характеризуються високою ресурсомісткістю та обмеженою гнучкістю налаштувань. У зв’язку з цим виникає необхідність формування узагальнених системних вимог до програмного забезпечення, яке поєднує ефективність автоматизованого аналізу з можливістю адаптації до різних сценаріїв використання. Ключовими вимогами в цьому контексті є висока точність сканування, помірні витрати обчислювальних ресурсів, гнучкі механізми конфігурації, підтримка інтеграції з іншими засобами безпеки через стандартизовані формати даних, а також можливість подальшого розширення функціональності.

На основі зазначених міркувань сформульовано такі системні вимоги до розроблюваного програмного забезпечення: забезпечення кросплатформної роботи з підтримкою основних операційних систем; використання модульної архітектури, що дозволяє додавати нові типи перевірок без модифікації ядра; мінімальний поріг входу для інтеграції із зовнішніми інструментами та процесами; підтримка декількох форматів звітності, зокрема, HTML і JSON для подальшого аналізу результатів.

Відповідно до визначених вимог обґрунтовано вибір технологічних рішень. За основний спосіб взаємодії з користувачем обрано інтерфейс командного рядка, що забезпечує простоту розгортання, зручність автоматизації та інтеграції з іншими системами, а також відсутність потреби у додатковій серверній інфраструктурі. Для реалізації програмного забезпечення використано мову програмування Java, яка гарантує кросплатформність, надає розвинені засоби для паралельного виконання завдань і містить зрілу екосистему бібліотек для роботи з HTTP-протоколом та модульною архітектурою.

Функціональну логіку роботи системи формалізовано за допомогою BPMN-

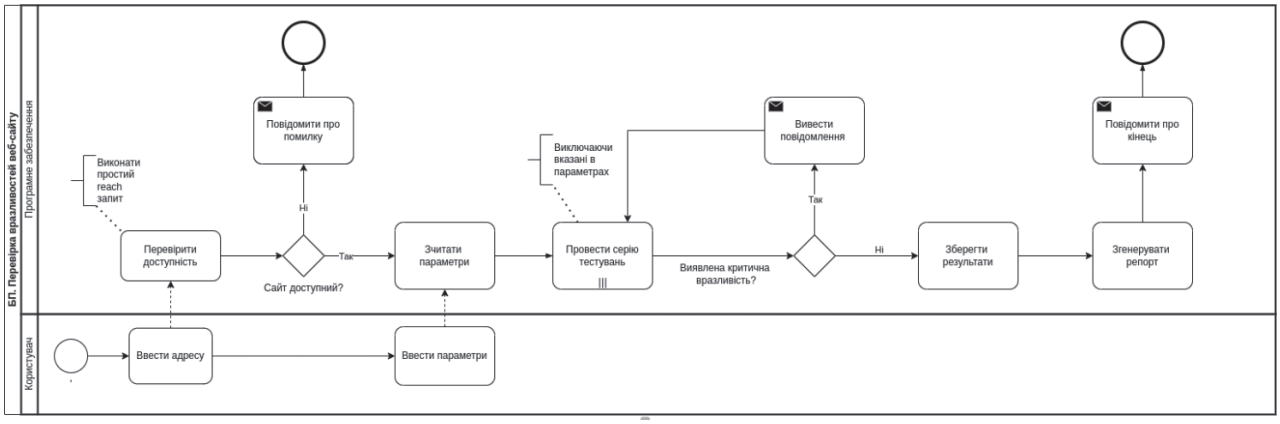

Рис. 1. Схема бізнес-процесу «Пошук вразливостей вебсайту»

моделі бізнес-процесу «Пошук вразливостей веб-сайту» (Рис.1).

Відповідно до цієї моделі користувач ініціює процес, задаючи адресу вебресурсу для аналізу та, за потреби, додаткові параметри, що визначають пріоритети або обмеження тестування. У разі некоректності введених даних або недоступності ресурсу система формує повідомлення про помилку. Далі здійснюється послідовне або паралельне виконання набору перевірок на наявність вразливостей, під час якого користувач може отримувати попереджувальні повідомлення про виявлення критичних ризиків. Завершальним етапом є формування звіту з результатами аналізу та інформування користувача про місце його збереження.

Запропонована сукупність вимог і технологічних рішень створює основу для реалізації підходу, що відповідає меті дослідження та забезпечує практичну придатність системи для автоматизованого пошуку вразливостей вебсайтів.

Комбінування статичного і динамічного аналізу. Запропонований у роботі підхід до пошуку вразливостей вебсайтів базується на поєднанні методів статичного та динамічного аналізу в межах єдиного керованого процесу сканування. Таке комбінування дозволяє зменшити надлишковість динамічних перевірок, підвищити відтворюваність результатів та оптимізувати використання обчислювальних і мережевих ресурсів без втрати повноти аналізу.

На першому етапі застосовується статичний аналіз вебресурсу, який виконується без активної взаємодії з логікою сервера. У межах цього етапу здійснюється побудова мапи сайту за допомогою пошукового робота, аналіз структури HTML-документів, форм введення даних, параметризованих URL та HTTP-заголовків. Результатом статичного аналізу є множина виявлених кінцевих точок, які потенційно можуть бути вразливими до експлуатації.

Нехай U – множина всіх URL-адрес, виявлених у процесі обходу вебресурсу. З метою зменшення надлишкового сканування виконується нормалізація URL-шаблонів, у ході якої динамічні параметри (числові ідентифікатори, UUID, хеші тощо) замінюються узагальненими маркерами. У результаті формується зменшена множина логічно унікальних шаблонів:

$$U_n \subseteq U.$$

Для кожного нормалізованого шаблону $u \in U_n$ визначається множина параметрів $P(u)$, що використовуються у запитах або формах введення даних. Тоді множина тестових точок для динамічного аналізу визначається як:

$$T = \bigcup_{u \in U_n} P(u).$$

Таким чином, динамічні перевірки виконуються не для всіх виявлених URL, а лише для узагальнених шаблонів і пов'язаних із ними параметрів, що істотно скорочує кількість HTTP-запитів.

Ефективність такого поєднання статичного та динамічного аналізу може бути кількісно оцінена за допомогою коефіцієнта редукції запитів:

$$R = 1 - \frac{| U_n |}{| U |},$$

де $| U | -$ кількість первинно виявлених URL, а $| U_n | -$ кількість нормалізованих URL-шаблонів. Значення цієї метрики характеризує ступінь зменшення надлишкових перевірок за рахунок попереднього статичного аналізу.

На другому етапі здійснюється динамічний аналіз, який полягає у виконанні активних тестів вразливостей (DAST) для кожної точки $t \in T$. Динамічні перевірки реалізуються у вигляді ізольованих асинхронних задач, що виконуються паралельно та використовують різні корисні навантаження (payloads) залежно від класу вразливості. Такий підхід дозволяє поєднати глибину аналізу з прийнятним часом виконання та контрольованим навантаженням на цільовий ресурс.

Загальний конвеєр комбінування статичного і динамічного аналізу в розробленій системі пошуку вразливостей вебсайтів наведено на рис. 2. Представлена схема ілюструє послідовність переходу від первинного збору інформації про вебресурс до виконання цільових динамічних перевірок та агрегування результатів у єдиний звіт.

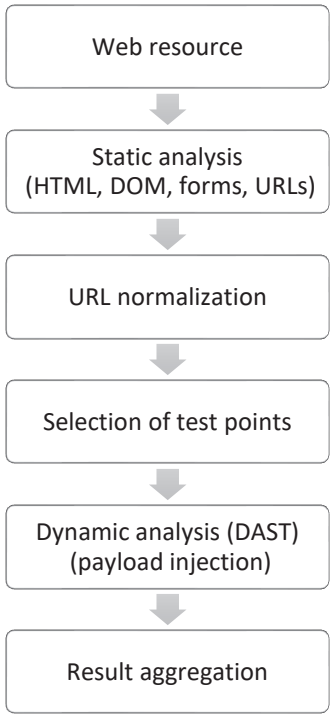

Рис. 2. Конвеєр комбінування статичного та динамічного аналізу у системі пошуку вразливостей вебсайтів

Узагальнено процес комбінування статичного та динамічного аналізу може бути поданий у вигляді композиції операторів:

$$A = DAST(SAST(W)),$$

де W — цільовий вебресурс, $SAST(\cdot) -$ оператор статичного аналізу, що формує структуру тестових точок, а DAST(·) – оператор динамічного тестування, який виконує активні перевірки на вразливості.

Запропонований підхід дозволяє поєднати переваги статичного аналізу, зокрема, масштабованість і низьку інвазивність, із точністю та практичною значущістю динамічного тестування. Це забезпечує зменшення надлишкових запитів, підвищення ефективності сканування та придатність системи до використання в автоматизованих процесах забезпечення якості та безпеки вебзастосунків.

Архітектура програмного забезпечення системи пошуку вразливостей. Для реалізації запропонованого підходу до автоматизованого пошуку вразливостей вебсайтів обрано модульну архітектуру програмного забезпечення з підтримкою розширення функціональності через плагіни. Така архітектура забезпечує гнучкість системи, її адаптацію до різних класів вразливостей і можливість еволюційного розвитку без модифікації ядра.

Архітектурні рішення ґрунтуються на використанні усталених проєктних патернів, що дозволяє досягти балансу між розширюваністю, продуктивністю та зручністю супроводу. Кожен тип перевірки вразливостей реалізується як окремий плагін, який динамічно підвантажується під час виконання, що спрощує додавання нових алгоритмів статичного та динамічного аналізу. Реалізація плагінного механізму базується на Java ServiceLoader API та забезпечує ізоляцію модулів тестування і зниження ризику поширення помилок між компонентами.

Взаємодія користувача з системою організована через інтерфейс командного рядка із застосуванням патерну Command, що дозволяє відокремити обробку команд від бізнес-логіки та спростити розширення набору функцій. Реалізація механізмів тес-

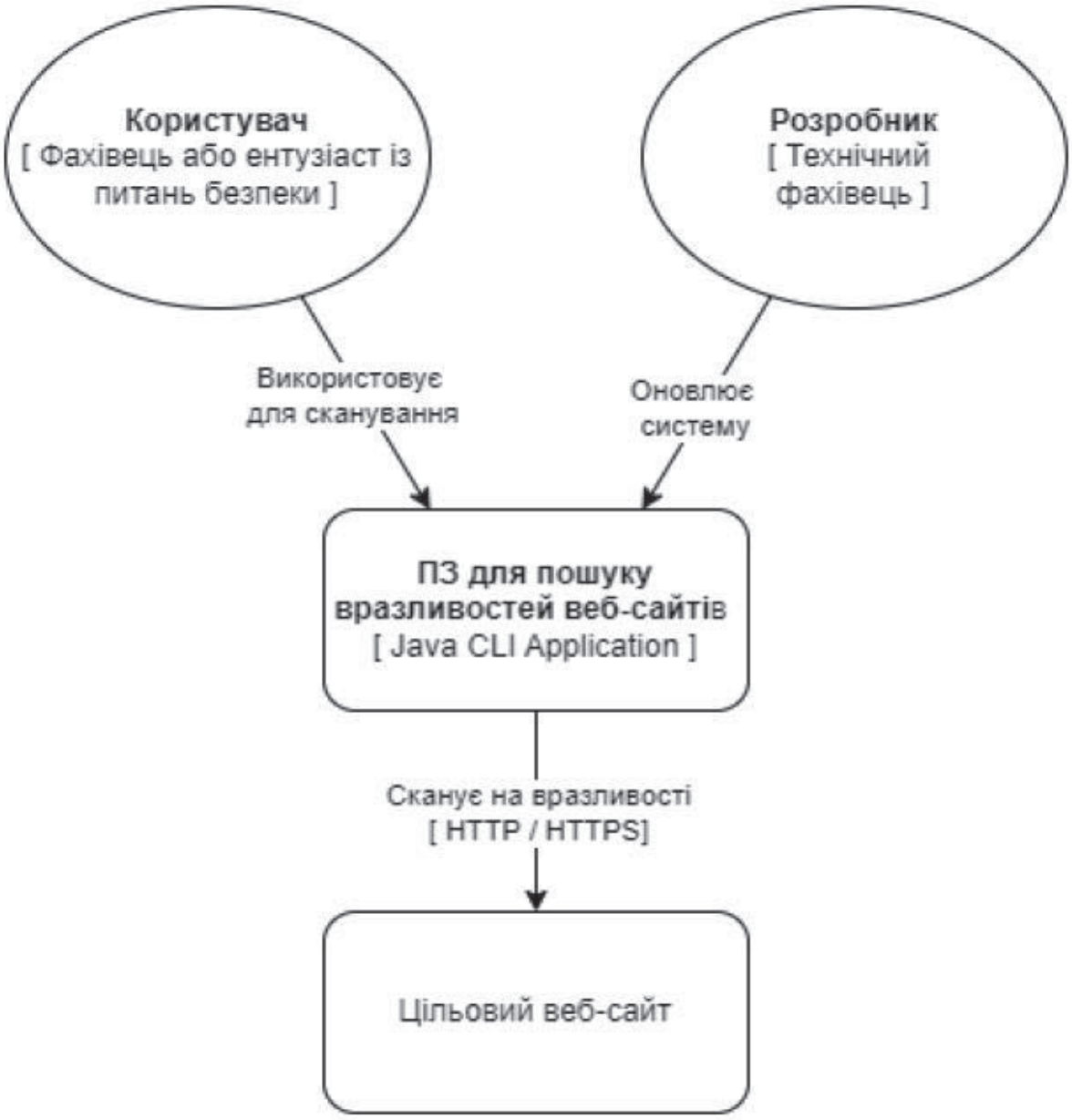

Рис. 3. Діаграма C4 Model першого рівня

тування вразливостей ґрунтується на патерні Strategy, який забезпечує взаємозамінність алгоритмів виявлення та підтримує комбінування статичних і динамічних методів аналізу в межах одного процесу сканування.

Для підвищення продуктивності система підтримує багатопотокове виконання перевірок із використанням механізмів Java Concurrency API. Паралельне виконання незалежних тестів дозволяє скоротити час аналізу та забезпечити масштабованість на багатоядерних обчислювальних платформах.

Зберігання сигнатур вразливостей, шаблонів тестування та результатів сканування реалізовано у форматі структурованих JSON-файлів без використання зовнішніх СУБД. Таке рішення підвищує портативність програмного забезпечення та спрощує його розгортання й оновлення. Архітектура системи узгоджується із принципом єдиної відповідальності та передбачає чіткий розподіл функцій між модулем командного інтерфейсу, ядром керування процесом аналізу, підсистемою плагінів, модулями сканування та формування звітів, що

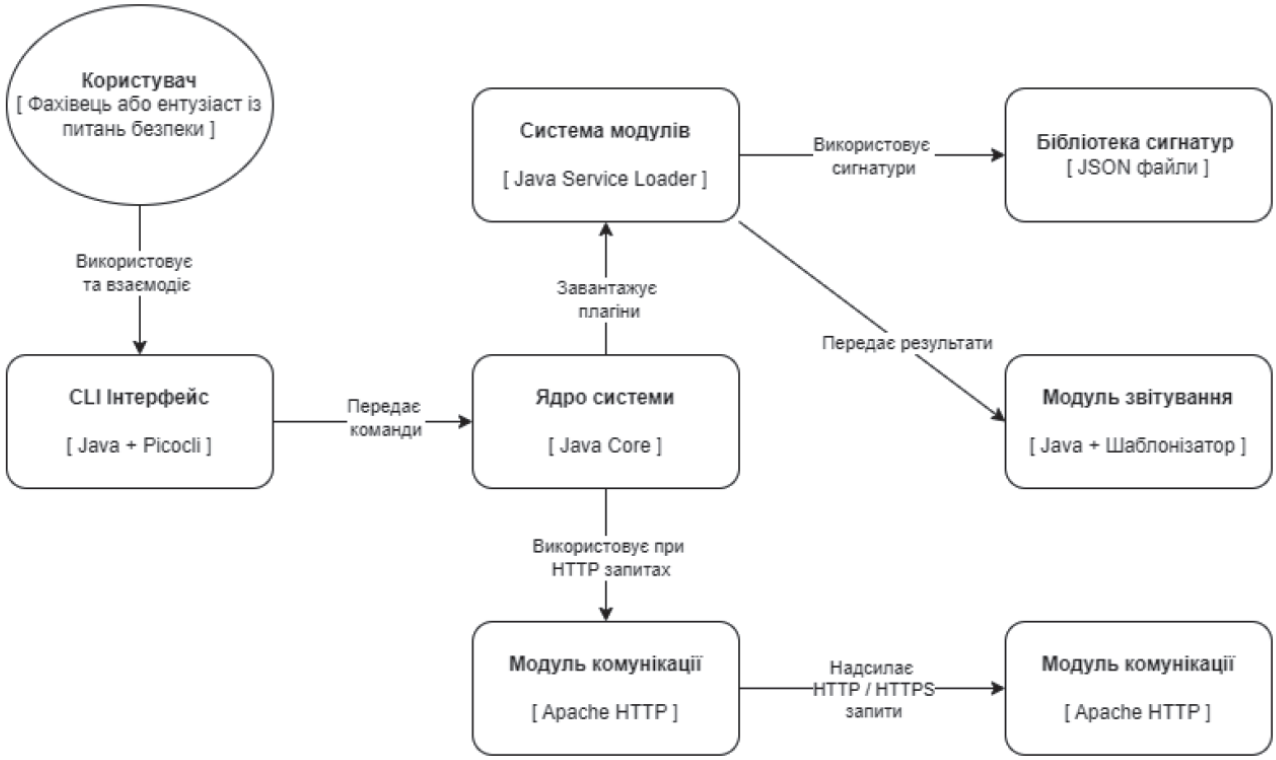

Рис. 4. Діаграма C4 Model другого рівня

спрощує тестування, супровід і подальший розвиток системи.

Загальна структура системи та взаємодія її компонентів відображені за допомогою діаграм C4 Model першого та другого рівнів (рисунки 3 і 4).

Для наочного подання структури програмного забезпечення та взаємодії його складових у роботі використано нотацію C4 Model, яка дозволяє поетапно деталізувати архітектурні рішення від загального контексту до рівня окремих компонентів.

Діаграма першого рівня (Context Diagram) відображає систему пошуку вразливостей як єдиного програмного комплексу, що взаємодіє з користувачем та зовнішніми вебресурсами, які підлягають аналізу. На цьому рівні акцент зроблено на ролі системи в загальному процесі забезпечення безпеки вебсайтів та її інтеграції в робочі сценарії використання.

Діаграма другого рівня (Container Diagram) деталізує внутрішню структуру системи, виокремлюючи основні контейнери та модулі, зокрема, командний інтерфейс, ядро керування процесом сканування, підсистему плагінів, модуль виконання тестів і підсистему формування звітів, а також їхні інформаційні потоки. Така декомпозиція дозволяє чітко простежити розподіл відповідальності між компонентами та обґрунтовує вибір модульної архітектури як основи для реалізації запропонованого підходу до пошуку вразливостей вебсайтів.

Конструювання програмного забезпечення. Програмна реалізація системи пошуку вразливостей вебсайтів базується на сукупності алгоритмічних та інженерних рішень, спрямованих на підвищення ефективності виявлення вразливостей і оптимізацію використання обчислювальних ресурсів. Реалізація виконана у вигляді набору взаємодіючих компонентів, інкапсульованих у відповідні класи з перевизначенням базових методів та модифікацією традиційних підходів до аналізу вебресурсів. До ключових функціональних компонентів системи належать інтелектуальний пошуковий робот, набір тестів вразливостей та механізми запобігання циклічному виконанню процесів сканування. Основні компоненти та відповідні алгоритмічні рішення системи узагальнено в табл. 3.1.

Таблиця 1. Основні компоненти та алгоритмічні рішення системи

Компонент	Призначення	Ключові алгоритмічні / інженерні рішення
URL-нормалізація	Усунення надлишкового сканування	Узагальнення динамічних параметрів URL за шаблонами
Web crawler	Побудова мапи сайту	Рекурсивний обхід з обмеженням глибини та фільтрацією
Модулі тестування	Виявлення вразливостей	Strategy-патерн для різних типів атак
Паралельний виконавець	Прискорення аналізу	ExecutorService, асинхронна агрегація
Модель даних	Зберігання результатів	Незмінні структури, JSON-серіалізація
CLI-інтерфейс	Взаємодія з користувачем	Command-патерн, Picocli

Центральним елементом логіки аналізу є алгоритм нормалізації URL-шаблонів, реалізований у модулі попередньої обробки даних. Його призначення полягає в усуненні надлишкового сканування сторінок з ідентичною структурною логікою, які відрізняються лише значеннями динамічних параметрів. Алгоритм виконує декомпозицію URL-адреси на сегменти та застосовує набір регулярних виразів для виявлення типових змінних ідентифікаторів (UUID, числові та шістнадцяткові значення, хеші), які замінюються узагальненими маркерами. Це дозволяє агрегувати множину фактичних URL в єдиний шаблон для подальшого тестування та істотно скоротити час сканування без втрати повноти аналізу.

Підвищення продуктивності системи досягається за рахунок багатопотокового виконання перевірок із використанням пулу потоків, розмір якого конфігурується відповідно до апаратних можливостей платформи. Кожен тест на вразливість (SQL-ін’єкції, XSS, CSRF тощо) виконується як ізольоване асинхронне завдання з подальшою агрегацією результатів, що забезпечує ефективне використання багатоядерних систем і скорочення загального часу аналізу.

Процес підготовки даних підтримується інтелектуальним пошуковим роботом, який реалізує рекурсивний обхід вебресурсу з обмеженням глибини та фільтрацією статичних ресурсів. Такий підхід дозволяє сформувати мапу сайту, зосереджену на потенційно вразливих елементах, зокрема, сторінках із формами введення та параметризованими запитами.

Програмна структура системи побудована відповідно до принципів об’єктно-орієнтованого програмування з використанням усталених патернів проєктування. Патерн Factory уніфікує створення модулів тестування, Strategy забезпечує взаємозамінність алгоритмів виявлення вразливостей, Builder використовується для гнучкого формування конфігурацій сканування, а Command реалізує взаємодію з користувачем через інтерфейс командного рядка, спрощуючи автоматизацію використання інструмента.

Модель даних системи спроєктована на основі незмінних структур, що гарантує коректність роботи в умовах багатопотокового доступу. Основними сутностями є моделі вразливостей, результати сканування та контекст виконання, який зберігає конфігурацію та стан HTTP-клієнта. Такий підхід дозволяє уникнути станів гонки та забезпечити цілісність даних під час паралельної обробки.

З урахуванням портативного характеру інструмента збереження даних реалізовано без використання реляційних СУБД — на основі файлової системи у форматі

JSON. Результати аналізу зберігаються у структурованому, людиночитабельному вигляді, придатному для подальшого аналізу та експорту. Корисні навантаження для тестів також зберігаються у зовнішніх JSON-ресурсах, що дозволяє оновлювати базу знань системи без перекомпіляції програмного коду.

Технологічний стек реалізації базується на мові програмування Java та бібліотеках з відкритим кодом. Для реалізації CLI-інтерфейсу використано Picocli, мережева взаємодія здійснюється через HTTP-клієнт OkHttp, аналіз HTML-коду — за допомогою Jsoup, а серіалізація даних — бібліотекою Jackson. Логування реалізовано з використанням SLF4J і Logback, візуалізацію прогресу забезпечують Jansi та ProgressBar, а збірка й керування залежностями виконуються за допомогою Gradle.

Аналіз якості програмного забезпечення. З урахуванням функціонального призначення та архітектурних особливостей розробленої системи для пошуку вразливостей вебсайтів якість програмного забезпечення оцінювалася за прикладно-орієнтованими метриками, що відображають як ефективність виявлення загроз, так і продуктивність роботи системи. До них належать точність і повнота, коефіцієнт редукції запитів та пропускна здатність.

Метрики точності (precision) та повноти (recall) визначалися на основі тестування системи на заздалегідь відомих вразливих вебзастосунках і характеризують здатність коректно ідентифікувати реальні вразливості, відрізняти їх від хибних спрацювань і забезпечувати покриття відомих класів загроз у процесі статичного та динамічного аналізу.

Коефіцієнт редукції запитів використано для оцінювання ефективності алгоритму нормалізації URL-шаблонів і характеризує зменшення кількості HTTP-запитів за рахунок усунення дубльованих логічних сутностей без втрати повноти аналізу. Пропускна здатність системи відображає ефективність реалізації багатопотокової архітектури та накладні витрати HTTP-клієнта OkHttp, визначаючи кількість запитів, що обробляються за одиницю часу.

Експериментальна перевірка якості програмного забезпечення виконувалася з використанням еталонних вразливих вебзастосунків OWASP Juice Shop і OWASP WebGoat, а також власних експериментальних проєктів.

У процесі оптимізації алгоритмів та архітектурних рішень було досягнуто істотного покращення продуктивності: час роботи пошукового робота скоротився приблизно з 30 хвилин до 2 хвилин для типового сценарію аналізу. Пропускна здатність системи становить близько 150 запитів за секунду для локально розгорнутих ресурсів і до 10 запитів за секунду для віддалених сайтів. Під час тестування підтверджено здатність платформи ефективно виявляти проблеми конфігурації CORS-заголовків, параметрів, потенційно вразливих до SSRF, а також інші поширені конфігураційні та логічні вади залежно від специфіки цільового ресурсу.

Основні результати експериментальної оцінки узагальнено в табл. 2.

Таблиця 2. Результати оцінювання якості та продуктивності програмного забезпечення

Метрика	Умови тестування	Отримане значення
Час роботи пошукового робота	До оптимізації	~30 хв
Час роботи пошукового робота	Після оптимізації	~2 хв
Пропускна здатність	Локально розгорнутий ресурс	~150 запитів/с
Пропускна здатність	Віддалений вебсайт	~10 запитів/с
Виявлення CORS-вразливостей	OWASP Juice Shop, WebGoat	Успішне
Виявлення параметрів, вразливих до SSRF	Еталонні та власні проєкти	Успішне

Обговорення результатів та перспективи розвитку

Отримані результати підтверджують доцільність запропонованого підходу до пошуку вразливостей вебсайтів на основі поєднання статичного й динамічного аналізу в межах модульної архітектури. Експериментальні дані свідчать, що попередня статична обробка у вигляді побудови мапи сайту та нормалізації URL-шаблонів істотно зменшує надлишковість динамічних перевірок і дозволяє скоротити загальний час сканування без втрати повноти аналізу.

Застосування коефіцієнта редукції запитів показало, що значна частина HTTP-запитів, характерних для традиційних динамічних сканерів, може бути усунена за рахунок агрегування логічно еквівалентних URL. Це особливо важливо для сучасних вебзастосунків із великою кількістю параметризованих посилань, де без оптимізації простір пошуку зростає експоненційно. Тож, запропонований підхід підвищує масштабованість аналізу та зменшує навантаження на цільовий ресурс, що є критичним у середовищах DevSecOps і CI/CD.

Оцінювання пропускної здатності підтвердило ефективність реалізованої багатопотокової архітектури. Досягнуті показники швидкості обробки запитів свідчать про здатність системи виконувати динамічні перевірки у прийнятні часові межі для вебресурсів різного рівня складності. Ізоляція тестів у вигляді асинхронних задач знижує ризик взаємного впливу перевірок і підвищує стабільність роботи системи.

Практичну придатність підходу підтверджено результатами виявлення конфігураційних і параметричних вразливостей, зокрема, пов'язаних із налаштуванням CORS-заголовків та потенційними SSRF-ризиками. Це демонструє, що навіть без залучення складних евристик або моделей машинного навчання можливо досягти прийнятного рівня точності та повноти за рахунок інженерно обґрунтованої організації процесу аналізу.

Водночас експериментальні результати вказують на певні обмеження запропонованого рішення. Ефективність нормалізації URL-шаблонів залежить від якості евристик виділення динамічних параметрів, що в окремих випадках може призводити до неповної редукції дубльованих запитів. Крім того, поточна реалізація динамічних тестів орієнтована переважно на класичні вразливості та конфігураційні помилки й повною мірою не охоплює складні логічні або контекстно-залежні дефекти.

Подальший розвиток підходу доцільно спрямувати на підвищення адаптивності та інтелектуальності системи, зокрема, шляхом удосконалення механізмів планування динамічних перевірок і поглиблення статичного аналізу клієнтського коду. Інтеграція зі стандартами звітності та зовнішніми засобами керування вразливостями також підвищить практичну цінність розробленого рішення для промислового застосування.

Висновки

У статті розроблено та обґрунтовано підхід до пошуку вразливостей вебсайтів на основі поєднання статичного й динамічного аналізу в межах модульної архітектури програмного забезпечення. Запропоноване рішення орієнтоване на підвищення ефективності автоматизованого аналізу безпеки вебресурсів через зменшення надлишкових перевірок, оптимізації використання обчислювальних ресурсів і забезпечення масштабованості системи.

У процесі дослідження сформульовано системні та архітектурні вимоги до програмного забезпечення для пошуку вразливостей вебсайтів з урахуванням сучасних практик DevSecOps та обмежень реальних середовищ експлуатації. Запропоновано алгоритм нормалізації URL-шаблонів, який дозволяє агрегувати логічно еквівалентні сторінки з різними динамічними параметрами та істотно скоротити кількість HTTP-запитів без втрати повноти аналізу. Модульна архітектура сканера з підтримкою плагінів і застосуванням усталених патернів проєктування забезпечує керовану розширюваність та спрощує інтеграцію нових механізмів статичного і динамічного тестування.

Реалізація багатопотокового виконання перевірок як асинхронних задач до-

зволила ефективно використовувати обчислювальні ресурси багатоядерних систем і суттєво зменшити загальний час сканування. Запропонований підхід до оцінювання якості програмного забезпечення, що базується на метриках точності й повноти, коефіцієнта редукції запитів і пропускної здатності, забезпечив об'єктивну експериментальну перевірку ефективності розробленого інструмента.

Результати експериментів, отримані під час тестування на еталонних вразливих вебзастосунках та власних проєктах, підтвердили практичну придатність запропонованого підходу. Було досягнуто істотного покращення продуктивності системи та продемонстровано здатність виявляти низку поширених конфігураційних і параметричних вразливостей у реальних умовах.

Література

1. Guo Z., Tan T., Liu S., Liu X., Lai W., Yang Y., Li Y., Chen L., Dong W., Zhou Y. Mitigating false positive static analysis warnings: Progress, challenges, and opportunities [Електронний ресурс] // *IEEE Transactions on Software Engineering*. – 2023. – Vol. 49, No. 12. – P. 5154–5188. – DOI: 10.1109/TSE.2023.3322561.
2. Althunayyan M., Saxena N., Li S., Gope P. Evaluation of black-box Web application security scanners in detecting injection vulnerabilities [Електронний ресурс] // *Electronics*. – 2022. – Vol. 11, No. 13. – Art. no. 2049. – DOI: 10.3390/electronics11132049.
3. Nunes P. J. C. Blended security analysis for web applications: Techniques and tools : PhD thesis [Електронний ресурс]. – Coimbra : Universidade de Coimbra, 2022. – Режим доступу: https://estudogeral.uc.pt/handle/10316/100340 (дата звернення: 16.12.2025).
4. Kree L., Helmke R., Winter E. Using Semgrep OSS to find OWASP Top 10 weaknesses in PHP applications: A case study [Електронний ресурс] // *Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA 2024)* : Proc. 21st Int. Conf. – Lecture Notes in Computer Science. – Vol. 14828. – Cham : Springer, 2024. – P. 64–83. – DOI: 10.1007/978-3-031-56543-4_4.
5. Shahid J., Hameed M. K., Javed I. T., Qureshi K. N., Ali M., Crespi N. A comparative study of Web application security parameters: Current trends and future directions [Електронний ресурс] // *Applied Sciences*. – 2022. – Vol. 12, No. 8. – Art. no. 4077. – DOI: 10.3390/app12084077.
6. Popereshnyak S., Chornobryvets D., Bakaiev O. AI-driven intelligent platform for freelance services management and monitoring [Електронний ресурс] // *Proceedings of the 1st Workshop Software Engineering and Semantic Technologies (SEST 2025)*. – CEUR Workshop Proceedings. – 2025. – P. 206–217. – Режим доступу: https://ceur-ws.org/Vol-4053/paper12.pdf
7. Popereshnyak S., Vecherkovskaya A., Zhebka V. Intrusion detection based on an intelligent security system using machine learning methods [Електронний ресурс] // *CEUR Workshop Proceedings*. – 2024. – Vol. 3654. – P. 163–178. – Режим доступу: https://ceur-ws.org/Vol-3654/paper14.pdf
8. Sönmez F. Ö., Kiliç B. G. Holistic Web Application Security Visualization for Multi-Project and Multi-Phase Dynamic Application Security Test Results [Електронний ресурс] // *IEEE Access*. – 2021. – Vol. 9. – P. 25858–25884. – DOI: 10.1109/ACCESS.2021.3057044.
9. Sharma S., Zavarsky P., Butakov S. Machine Learning based Intrusion Detection System for Web-Based Attacks [Електронний ресурс] // *Proceedings of the IEEE 6th International Conference on Big Data Security on Cloud, High Performance and Smart Computing, and Intelligent Data and Security*. – Baltimore, USA, 2020. – P. 227–230. – DOI: 10.1109/BigDataSecurity-HPSC-IDS49724.2020.00048.

References

1. Guo, Z., Tan, T., Liu, S., Liu, X., Lai, W., Yang, Y., Li, Y., Chen, L., Dong, W., & Zhou, Y. (2023). Mitigating false positive static analysis warnings: Progress, challenges, and opportunities. *IEEE Transactions on Software Engineering, 49*(12), 5154–5188. https://doi.org/10.1109/TSE.2023.3322561
2. Althunayyan, M., Saxena, N., Li, S., & Gope, P. (2022). Evaluation of black-box Web application security scanners in detecting injection vulnerabilities. *Electronics, 11*(13), 2049. https://doi.org/10.3390/electronics11132049
3. Nunes, P. J. C. (2022). *Blended security analysis for web applications: Techniques and tools* (Doctoral dissertation, Universidade de

Coimbra). Retrieved from https://estudogeral.uc.pt/handle/10316/100340

4. Kree, L., Helmke, R., & Winter, E. (2024). Using Semgrep OSS to find OWASP Top 10 weaknesses in PHP applications: A case study. In *Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA 2024)* (Lecture Notes in Computer Science, Vol. 14828, pp. 64–83). Springer. https://doi.org/10.1007/978-3-031-56543-4_4
5. Shahid, J., Hameed, M. K., Javed, I. T., Qureshi, K. N., Ali, M., & Crespi, N. (2022). A comparative study of Web application security parameters: Current trends and future directions. *Applied Sciences, 12*(8), 4077. https://doi.org/10.3390/app12084077
6. Popereshnyak, S., Chornobryvets, D., Bakaiev, O. (2025). AI-driven intelligent platform for freelance services management and monitoring. In *Proceedings of the 1st Workshop on Software Engineering and Semantic Technologies (SEST 2025)* (pp. 206–217). CEUR Workshop Proceedings. Retrieved from https://ceur-ws.org/Vol-4053/paper12.pdf
7. Popereshnyak, S., Vecherkovskaya, A., & Zhebka, V. (2024). Intrusion detection based on an intelligent security system using machine learning methods. *CEUR Workshop Proceedings, 3654*, 163–178. Retrieved from https://ceur-ws.org/Vol-3654/paper14.pdf
8. Sönmez, F. Ö., & Kiliç, B. G. (2021). Holistic Web Application Security Visualization for Multi-Project and Multi-Phase Dynamic Application Security Test Results. *IEEE Access, 9*, 25858–25884. https://doi.org/10.1109/ACCESS.2021.3057044
9. Sharma, S., Zavarsky, P., & Butakov, S. (2020). Machine learning based intrusion detection system for web-based attacks. In *Proceedings of the IEEE 6th International Conference on Big Data Security on Cloud, High Performance and Smart Computing, and Intelligent Data and Security* (pp. 227–230). IEEE. https://doi.org/10.1109/BigDataSecurity-HPSC-IDS49724.2020.00048

Одержано: 17.12.2025
Внутрішня рецензія отримана: 22.12.2025
Зовнішня рецензія отримана: 24.12.2025

Про авторів:

Поперешняк Світлана Володимирівна,
к.ф.-м.н., доцент
http://orcid.org/0000-0002-0531-9809.

Горох Богдан Дмитрович,
здобувач вищої освіти
http://orcid.org/0009-0001-4143-0029

Місце роботи авторів:

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»,
тел. +38-098-645-54-62
E-mail: spopereshnyak@gmail.com
horokhbohdandmytrovich@gmail.com