COMPUTER SYSTEMS AND NETWORKS

Vol. 7, No. 1, 2025

DOI: https://doi.org/10.23939/csn2025.01.195

УДК 004.02; 004.05; 004.09

ОГЛЯД СУЧАСНИХ ІНСТРУМЕНТІВ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ БЕЗПЕКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

М. В. Максимович, Л. З. Мичуда

Національний університет “Львівська політехніка”,
кафедра безпеки інформаційних технологій
E-mail: maksym.v.maksymovych@lpnu.ua, lesia.z.mychuda@lpnu.ua

На сьогодні зі стрімким розвитком сучасних технологій в галузі розробки програмного забезпечення, активною цифровізацією та міграцією багатьох послуг в онлайн як ніколи постає питання забезпечення безпеки сервісів, які надають такі послуги в контексті цілісності, конфіденційності та доступності інформації. Рівень захищеності додатків прямо залежить від інвестицій в безпеку в процесі розробки додатків, тому важливо щоб пріоритетом розробників була не тільки реалізація функціональних вимог, а й активна робота над нефункціональними вимогами в контексті безпеки додатків. Забезпечення безпеки додатків полягає у використанні різноманітних інструментів, які дають змогу мінімізувати імовірність успішного проведення різних атак. Сьогодні існує безліч таких інструментів - комерційних чи з відкритим кодом. Проаналізовано сучасні інструменти автоматизованого тестування безпеки програмного забезпечення, порівняння їхніх можливостей, вартості впровадження та складності інтеграції у життєвий цикл розробки. Визначення переваг та недоліків комерційних рішень та інструментів з відкритим кодом, а також встановлення перспективних напрямів подальших досліджень щодо спрощення впровадження безкоштовних інструментів у практику забезпечення безпеки додатків.

Ключові слова: автоматизація, захист, кібербезпека, контейнеризація, програмне забезпечення, тестування.

Вступ

У сучасному цифровому середовищі безпека програмного забезпечення відіграє критично важливу роль, адже кількість кіберзагроз постійно зростає. Від особистих даних користувачів до корпоративних систем, вразливості у програмному коді можуть стати причиною серйозних інцидентів, які призводять до фінансових втрат і репутаційних ризиків. У відповідь на зростаючі загрози кіберзлочинності активізуються дослідження у сфері кібербезпеки, а ринок продуктів, що пропонує ефективні рішення для захисту, продовжує стрімко розширюватися. Серед таких рішень особливе місце займають інструменти автоматизованого тестування безпеки програмного забезпечення, які впроваджуються безпосередньо в процес розробки для виявлення та усунення вразливостей на ранніх етапах.

Сучасні інструменти автоматизованого тестування безпеки допомагають виявляти потенційні загрози та вразливості ще на етапі розробки, що дає змогу значно зменшити ризики експлуатації коду зловмисниками. Вони забезпечують ефективний аналіз, швидке виявлення проблем і сприяють підвищенню рівня безпеки додатків.

Сьогодні на ринку представлено багато інструментів для підвищення безпеки додатків, серед яких як комерційні рішення, так й інструменти з відкритим кодом - кожен із них має свої переваги та недоліки. У цій статті ми розглянемо сучасні інструменти автоматизованого тестування безпеки програмного забезпечення, їхні можливості, сильні та слабкі сторони, а також їхній вплив на якість і надійність програмних рішень.

Огляд літературних джерел

Як зазначається у статті [1], кількість кібератак на програмне забезпечення має тенденцію до збільшення. Досить часто системи мають вразливості в безпеці, які можуть бути використані в зловмисних цілях.

Згідно зі звітами Cybersecurity Ventures, наслідки кібератак щороку стають дедалі масштабнішими, що призводить до суттєвих фінансових втрат для компаній, урядів і користувачів. Експерти прогнозують, що глобальні витрати, спричинені кібератаками, зростатимуть експоненційно, оскільки хакерські методи стають більш витонченими, а кількість цифрових загроз невпинно збільшується. Основними чинниками такого зростання є поширення програм-вимагачів, витоки конфіденційної інформації, фінансові шахрайства та атаки на критично важливу інфраструктуру. Наприклад, у своїх прогнозах Cybersecurity Ventures вказує, що до 2025 року глобальні фінансові збитки від кіберзлочинності можуть сягнути 10,5 трильйона доларів щорічно, порівняно з 3 трильйонами у 2015 році [2].

Такі масштаби збитків пояснюються не лише зростанням кількості атак, а й їхньою складністю. Хакери дедалі частіше використовують методи соціальної інженерії, штучний інтелект та автоматизовані атаки, що дозволяє їм обходити традиційні системи захисту. У відповідь компанії змушені інвестувати дедалі більше ресурсів у кібербезпеку, зокрема у автоматизоване тестування безпеки, яке допомагає виявляти вразливості на ранніх етапах розробки програмного забезпечення [3]. Згідно з дослідженням “Grand View Research” [4] - компанія, яка спеціалізується на проведенні маркетингових досліджень і аналізі ринків, ринок кібербезпеки невпинно зростає, і оцінюється в 245 мільярдів доларів у 2024 році, з ростом у 12.9 % до 2030 року (рис. 1), що свідчить про актуальність проблеми та активні інвестиції у дослідження у сфері кіберзахисту.

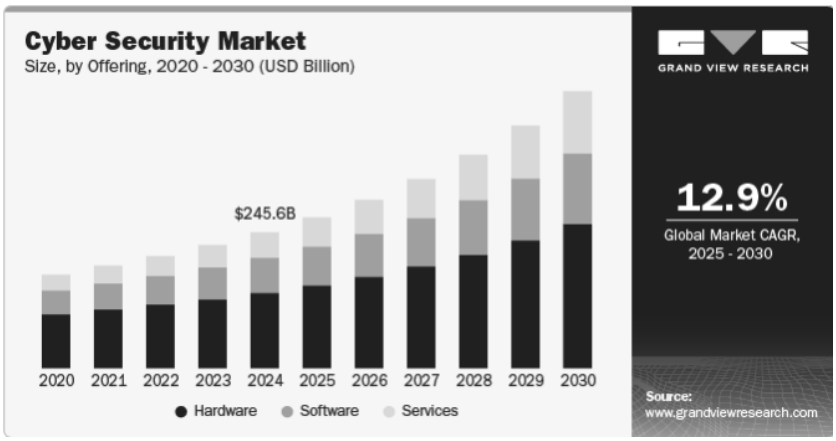

Рис. 1. Динаміка росту ринку кіберзахисту

Як зазначається у статті [5], програмне забезпечення, має відповідати вимогам замовника, а також бути захищеним і вільним від вразливостей, тому забезпечення його безпеки є насамперед відповідальністю розробників. Аналіз, впровадження та підтримка вимог до безпеки систем, що інтенсивно використовують програмне забезпечення, і створення справді безпечного програмного забезпечення потребує планування безпеки з нуля та постійного забезпечення безпеки протягом життєвого циклу програмного забезпечення та навіть після розгортання, коли програмне забезпечення розвивається [6].

Також у статті [7] автори зазначають, що сучасні проєкти з розробки програмного забезпечення мають враховувати безпеку як одну з важливих характеристик, які повинно мати ПЗ. Однак надмірні витрати на безпеку можуть призвести до збільшення вартості програмного забезпечення та часто до його затримок.

Зважаючи на велику кількість наявних інструментів, залишається важливим також правильно обирати та застосовувати інструменти для досягнення найбільшої ефективності та водночас з мінімальними затратами ресурсів, тому в межах цієї статті буде оглянуто найпопулярніші інструменти забезпечення безпеки, комерційні чи відкритого коду, буде проведена їхня порівняльна характеристика, аналіз ефективності використання та необхідних ресурсів для впровадження та підтримки.

Постановка задачі

Ця стаття присвячена аналізу та порівнянню сучасних інструментів автоматизованого тестування безпеки програмного забезпечення, включаючи комерційні рішення та інструменти з відкритим кодом.

Метою дослідження є визначення особливостей їх впровадження, вартості, складності інтеграції у життєвий цикл розробки та оцінка практичної ефективності. Крім того, важливим аспектом роботи є встановлення переваг і недоліків наявних підходів, а також формулювання перспективних напрямів для подальших досліджень щодо спрощення процесу інтеграції та підтримки безкоштовних інструментів тестування безпеки в життєвому циклі розробки програмного забезпечення..

Типи тестування безпеки програмного забезпечення

У контексті автоматизованого тестування безпеки виділяють 4 підходи: статичний, динамічний, інтерактивний, композитний аналізи. Розглянемо коротко кожен з них:

Статичний аналіз (SAST) – метод тестування безпеки, що аналізує вихідний код без його виконання. Допомагає усунути вразливості на етапі розробки, але не виявляє проблеми у зовнішніх сервісах чи API [8].

Динамічний аналіз (DAST) – тестування запущених додатків за методом “чорної скриньки”. Виявляє вразливості через сканування вебінтерфейсу та імітацію атак [9].

Інтерактивний аналіз (IAST) – поєднує SAST і DAST, перевіряючи безпеку в процесі роботи додатка під час тестування чи взаємодії з ним [10].

Композитний аналіз (SCA) – оцінює безпеку сторонніх бібліотек і залежностей, допомагаючи виявляти відомі вразливості у використаних модулях.

Ці підходи є реалізовані сучасними інструментами, комерційними чи інструментами відкритого коду, які інтегруються в життєвий цикл розробки програмного забезпечення, та виконуються автоматично під час кожної потенційної версії програмного забезпечення.

Сучасні інструменти тестування безпеки

Вступ. На сьогодні існує безліч інструментів, які реалізують тестування безпеки програмного забезпечення, серед яких можуть бути комерційні чи інструменти на основі відкритого коду. Кожна категорія реалізовує в якомусь вигляді вищезгадані методології тестування, динамічне, статичне чи інтерактивне тестування безпеки. У цьому розділі ми розглянемо кожну з категорій, проведемо аналіз інструментів на основі даних з відкритих джерел.

Комерційні інструменти – інструменти тестування безпеки, які розробили провідні компанії в галузі кіберзахисту, пропонують легку інтеграцію автоматизованого тестування в життєвий цикл розробки програмного забезпечення. Такі інструменти пропонують на основі платних підписок, на основі обраного плану, що залежить від рівня компанії, де буде використовуватись інструмент, кількості проведених тестувань, часу виконання, обраних інструментів тестування, обраних інтеграцій тощо.

Як приклад комерційного інструменту можна навести **JIT Security**. JIT – це відкрита платформа Application Security Posture Management (ASPM), яка автоматизує перевірки безпеки на всіх етапах життєвого циклу розробки програмного забезпечення [11].

JIT інтегрується з різними засобами контролю безпеки та інструментами відкритого коду для сканування, що забезпечує комплексне покриття всіх етапів життєвого циклу програмного забезпечення та сприяє впровадженню тестування безпеки в робочі процеси розробників. Цей інструмент поєднує в собі методи статичного, композитного та динамічного аналізу та підтримує інтеграцію з популярними середовищами розробки, зокрема GitHub, AWS і GCP, що робить її гнучким та ефективним рішенням для автоматизації безпеки.

Як зазначалось раніше, JIT базується на безкоштовних інструментах відкритого коду, ефективно поєднуючи їх та надаючи розробникам можливість швидко та легко інтегрувати в цикл розробки. Можна зробити висновок, що основним продуктом компанії є інструменти швидкої, зручної та легкої інтеграції автоматизованого тестування безпеки, ніж розробка самих інструментів забезпечення безпеки.

Цінова політика використання базується на основі потреби використання та передбачає 3 плани (рис. 2):

- **Community** - безкоштовний план, для проєктів до 3 розробників, який передбачає використання в межах знайомства з продуктом;
- **Growth** - для невеликих продуктів чи стартапів від 4 розробників, вартість місячної підписки коштує 50$ на місяць за 1 розробника;
- **Enterprise** - для продуктів Enterprise рівня, вартість використання на основі індивідуальних домовленостей та не розголошується.

Рис. 2. Цінова політика інструменту JIT Security

Всі плани охоплюють методи статичного, динамічного та композитного аналізів, проте вартість використання та інтеграції динамічного аналізу оцінюється окремо, так само окремо може оцінюватись вартість сканувань вебдодатків чи точок доступу серверів (API).

Також варто зазначити що цей інструмент не пропонує методів інтерактивного тестування безпеки, що може бути більш ефективним щодо швидкості виконання та використання ресурсів.

Серед інших інструментів також можна виділити такі, як Veracode, Fortify, які пропонують методи статичного, динамічного чи композитного аналізу та зручну інтеграцію на основі вибраного плану. Цінова політика даних компаній не розголошується, вартість використання є індивідуальною, проте цілком зрозуміло що вартість використання оцінюється десятками тисяч доларів щорічно, з поступовим зростанням залежно від динаміки росту компанії, за базовий функціонал для компаній малого рівня типу стартапів, що може не входити в рамки бюджету, і отже, безпека може втратити пріоритет на початкових етапах, що може мати негативний вплив на роботу компанії.

Інструменти відкритого коду - це програмні рішення, доступні безкоштовно для використання, модифікації та розповсюдження. Вони дають змогу розробникам інтегрувати безпеку без необхідності у великих фінансових витратах на ліцензії чи комерційні продукти. Такі інструменти використовуються для статичного аналізу коду (SAST), динамічного тестування безпеки (DAST), аналізу залежностей (SCA) чи інтерактивного тестування безпеки додатку (IAST). Існує багато інструментів, які покривають кожен тип тестування, проте не існує єдиного інструменту відкритого коду, який би покривав всі типи тестування, а отже, прийняття рішень про використання певних інструментів, впровадження та підтримка стає тягарем розробників, що може бути ресурсозатратним завданням.

Одним з найвідоміших інструментів динамічного тестування є **OWASP ZAP** - відкритий, безкоштовний інструмент, розроблений для допомоги в тестуванні безпеки вебзастосунків і виявленні їхніх вразливостей. ZAP активно використовується для виявлення загроз, таких як SQL-ін'єкції, XSS (міжсайтові скриптові атаки) та інших типових вразливостей вебдодатків [12].

Checkmarx Interactive Application Security Testing (CxIAST) - це динамічне та безперервне рішення для тестування безпеки, яке виявляє вразливості у запущеній програмі шляхом використання існуючих заходів функціонального тестування [13]. CxIAST, на відміну від традиційного DAST або OWASP ZAP, фокусується на інтеграції тестування безпеки безпосередньо у процеси тестування функціональності. Він дає змогу проводити тестування безпеки без переривання життєвого циклу розробки, оскільки працює паралельно з іншими тестами, виявляючи вразливості в коді, який вже працює. Це дозволяє виявляти проблеми без затримок та інтегрувати тестування безпеки швидше в цикл розробки.

Інструменти статичного аналізу є найлегшими в інтеграції, оскільки не потребують виконання програми для перевірки, а працюють виключно шляхом аналізу коду щодо вразливостей, наприклад наявність таємних ключів у коді, недосяжного чи непотрібного коду, порушень безпекових стандартів, аналізують залежності, оцінюють складність і ефективність коду, а також допомагають дотримуватись правил програмування та безпеки. Вибір SAST інструментів залежить від конкретної мови програмування, однак якщо у проєкті використовуються популярні мови програмування, такі як Java, C#, JavaScript чи Python, можна вибирати серед безлічі інструментів [14], наприклад, **Mobile Security Framework** для Java, Kotlin, Objective-C, та Swift, **PHPStan** – найбільш популярний для PHP, **Bandit** – широко застосовується у Python, або **Semgrep**, яка є агностичною до мови програмування.

Також в сучасних проєктах широко застосовуються сторонні бібліотеки, тому дуже важливим є застосування композитного аналізу безпеки, в межах якого проводиться перевірка цих бібліотек на предмет безпеки. Серед основних інструментів композитного аналізу можна виділити такі: **OWASP Dependency-check**, що працює з такими мовами, як Java, .NET, JavaScript, Ruby, **RetireJS** – може перевіряти залежності виключно в рамках мови Javascript та **Safety** – який працює з Python [15].

Основним недоліком використання інструментів відкритого коду є складність впровадження в життєвий цикл розробки продукту, відсутність зручних інтеграцій чи, як зазначалось раніше, відсутність єдиного інструменту, який би покривав всі методології тестування безпеки. Отож під час використання інструментів відкритого коду компанія може економити значну частину фінансових ресурсів, проте буде витрачати набагато більше часу інженерів на впровадження та підтримку, ніж у разі використання комерційних інструментів.

Звідси можна зробити висновок, що актуальними є дослідження в сфері застосування та оптимізації використання інструментів тестування безпеки на основі відкритого коду, результатом яких може бути рішення, що полегшує впровадження та підтримку даних інструментів в життєвому циклі розробки ПЗ.

Оцінка ефективності застосування інструментів тестування безпеки

Як зазначалось раніше, один з найпопулярніших комерційних інструментів тестування безпеки, який працює на основі інструментів відкритого коду, пропонує рішення простої інтеграції автоматизації безпеки на основі обраного плану. Крім того, цей інструмент не пропонує методів інтерактивного тестування безпеки, що є великим недоліком, оскільки методи інтерактивного тестування реалізовують швидке динамічне тестування, тоді як методи динамічного тестування, які пропонуються інструментом, можуть займати десятки годин.

У межах статті [16] було проаналізовано час виконання інструментів різного типу. У межах дослідження було створено проєкт зі спеціально підготовленими вразливостями (рис. 3), до яких застосовувались інструменти тестування. Було використано найбільш відомі на теперішній час вразливості за версією документа "Топ-10 OWASP", який створила компанія OWASP (Open Web Application security project), що описує найвідоміші атаки на сьогодні.

CWE	Vulnerability Types	Number of Test Cases
78	Command Injection	40
643	Xpath Injection	20
79	Cross Site Scripting (XSS)	40
90	LDAP Injection	20
22	Path Traversal	40
89	SQL Injection	40
614	Secure Cookie flag	20
501	Trust Boundary Violation	20
330	Weak Randomness	40
327	Weak Cryptographic	20
328	Weak Hashing	20
	Total	320

Рис. 3. Вразливості, включені в проєкт

У межах дослідження було виявлено середній час виконання тестування з використанням цих інструментів. Отже, у результаті дослідження було виявлено середній час виконання тестування з використанням кожного з методів (табл. 1).

Таблиця 1

Час тестування проєкту з використанням вказаних типів

Інструмент	Час на тестування
SAST	19- 22 хв
DAST	36 г
IAST	2 хв

Враховуючи результати цього дослідження, можна зробити висновок, що найбільш ефективним є поєднання статичного та інтерактивного типів тестування, на відміну від статичного та динамічного типів, яке пропонує інструмент JIT security, що свідчить про те, що в контексті доцільності більш практичним може бути самостійний вибір та застосування інструментів на основі відкритого коду, ніж використання комерційних продуктів та повна залежність від них.

Як підсумок можна провести порівняльну характеристику застосування підходів на основі комерційних та безкоштовних рішень, що наведена у табл. 2.

Таблиця 2

Порівняльний аналіз застосування комерційних та безкоштовних інструментів тестування безпеки

Критерій	Комерційні рішення	Інструменти з відкритим кодом
Вартість	Висока, платні підписки	Безкоштовні або умовно безкоштовні
Функціональність	SAST, DAST, IAST, SCA (залежно від плану)	Окремі інструменти для кожного типу тестування
Гнучкість	Обмежена тарифами	Повна, можливість кастомізації
Інтеграція в CI/CD	Легка, з готовими плагінами	Потребує ручного налаштування
Підтримка	Офіційна підтримка, оновлення	Спільнота або самостійне налаштування

Отже, підсумовуючи, як видно з табл. 2, комерційні рішення забезпечують просту інтеграцію та комплексний підхід до тестування безпеки, проте їхня висока вартість та обмеження тарифних планів можуть бути перешкодою для гнучкого використання. Інструменти з відкритим кодом, навпаки, пропонують безкоштовний доступ і можливість кастомізації, але потребують значних зусиль для налаштування та підтримки. Отож вибір між цими підходами залежить від потреб компанії, наявних ресурсів та рівня підготовки команди до самостійного впровадження засобів безпеки.

Враховуючи це, можна зробити висновок, що дослідження є актуальними для подальшого вдосконалення підходів до автоматизованого тестування безпеки в життєвому циклі розробки програмного забезпечення. Зокрема, перспективним напрямом є створення ефективних методів інтеграції інструментів з відкритим кодом, які б спрощували налаштування, підтримку та масштабування процесів безпеки. Це дасть змогу розробникам отримати гнучке та економічно вигідне рішення без необхідності значних фінансових витрат на комерційні продукти.

Результати дослідження

У межах дослідження було проаналізовано найбільш популярні інструменти тестування безпеки програмного забезпечення, що реалізують тестування в межах визначених типів. Розглядалися як комерційні рішення, так й інструменти з відкритим кодом, зокрема їхні базові версії, особливості інтеграції та цінова політика.

Аналіз показав, що комерційні інструменти пропонують швидку та зручну інтеграцію в життєвий цикл розробки, проте висока вартість обмежує їхнє широке використання. Крім того, деякі сервіси не дають змогу комбінувати різні підходи, нав’язуючи додаткову оплату за окремі методи тестування, що знижує гнучкість для користувачів.

З іншого боку, більшість комерційних рішень фактично інтегрують відкриті інструменти, надаючи зручний інтерфейс для їх використання. Це свідчить про те, що розробники можуть самостійно підбирати відповідні інструменти та впроваджувати їх у свої процеси, не покладаючись на комерційні платформи. Однак самостійна інтеграція потребує значних зусиль і знань у сфері DevSecOps, оскільки немає універсального безкоштовного інструменту, що повністю автоматизує процес тестування.

Враховуючи вищесказане, можна зробити висновок, що залишаються актуальними дослідження у сфері впровадження автоматизованого тестування в життєвий цикл розробки програмного забезпечення, результатом яких може бути розроблений підхід, який дозволяє швидко та безкоштовно впроваджувати та підтримувати автоматизоване тестування безпеки різних типів на основі інструментів відкритого коду.

Висновки

Проведене дослідження показало, що сучасні інструменти автоматизованого тестування безпеки є надзвичайно актуальними в умовах постійного зростання складності кіберзагроз та широкої цифровізації процесів. Незважаючи на те, що комерційні рішення пропонують зручну інтеграцію, комплексну підтримку та простоту використання, їх застосування обмежується високою вартістю, що не завжди доступно для малих і середніх компаній.

Альтернативні інструменти з відкритим кодом є безкоштовними і дають змогу гнучко адаптувати процес тестування до конкретних потреб, однак їх впровадження та подальше використання потребують значних ресурсів: часу, спеціальних знань і додаткових зусиль розробників. Складність налаштування та підтримки цих рішень часто стримує команди від їх впровадження в повній мірі, навіть попри очевидні переваги з точки зору економії коштів.

Отже, перспективним напрямом для подальших досліджень є розробка нових підходів і технологій, які б спрощували процес інтеграції та підтримки відкритих рішень тестування безпеки. Це допомогло б забезпечити ефективний захист програмного забезпечення за мінімальних витрат ресурсів і зробило б автоматизоване тестування безпеки більш доступним та практичним для широкого кола розробників і компаній.

Список літератури

1. *Tauqeer O. B., Jan S., Khadidos A. O., Khadidos A. O., Khan F. Q., Khattak S. Analysis of security testing techniques. Intelligent Automation and Soft Computing. 2021. 29(1). 291–306. Scopus. Doi: https://doi.org/10.32604/iasc.2021.017260.*
2. *Magazine C. Cybercrime To Cost The World $10.5 Trillion Annually By 2025. Cybercrime Magazine. 2018, December 8. URL: https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/.*
3. *Li J. Vulnerabilities mapping based on OWASP-SANS: A survey for static application security testing (SAST). Annals of Emerging Technologies in Computing. 2020. 4(3). 1–8. Scopus. Doi: https://doi.org/10.33166/AETiC.2020.03.001.*
4. *Cyber Security Market Size, Share | Industry Report, 2030. (n.d.). March 10, 2025. URL: https://www.grandviewresearch.com/industry-analysis/cyber-security-market.*
5. *Pathirathna P., Ayesha V., Imihira W., Wasala W., Kodagoda N., Edirisinghe T. Security testing as a service with docker containerization. 2017. p. 7. Doi: https://doi.org/10.1109/SKIMA.2017.8294109.*
6. *Mirakhorli M., Galster M., Williams L. Understanding Software Security from Design to Deployment. ACM SIGSOFT Software Engineering Notes. 2020. 45(2). 25–26. Doi: https://doi.org/10.1145/3385678.3385687.*
7. *Tøndel I. A., Cruzes D. S., Jaatun M. G. Achieving "Good Enough" Software Security: The Role of Objectivity. Proceedings of the Evaluation and Assessment in Software Engineering. 2020. 360–365. Doi: https://doi.org/10.1145/3383219.3383267.*
8. *Yang J., Tan L., Peyton J., A Duer K. Towards Better Utilizing Static Application Security Testing. 2019. 51–60. Scopus. Doi: https://doi.org/10.1109/ICSE-SEIP.2019.00014.*
9. *Singh R., Kumar Gupta M., Patil D. R., Maruti Patil S. Analysis of Web Application Vulnerabilities using Dynamic Application Security Testing. 2024 IEEE 9th International Conference for Convergence in Technology, I2CT 2024. Scopus. Doi: https://doi.org/10.1109/I2CT61223.2024.10543484.*
10. *Pan Y. (2019). Interactive Application Security Testing. 2019 International Conference on Smart Grid and Electrical Automation (ICSGEA), 558–561. Doi: https://doi.org/10.1109/ICSGEA.2019.00131.*
11. *Top 11 DevOps Security Tools. Jit. 2024, August 8. URL: https://www.jit.io/resources/appsec-tools/top-11-devops-security-tools.*
12. *Alazmi S., Leon D. Customizing OWASP ZAP: A Proven Method for Detecting SQL Injection Vulnerabilities. 2023 IEEE 9th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS). 2023. 102–106. Doi: https://doi.org/10.1109/BigDataSecurity-HPSC-IDS58521.2023.00028.*
13. *Singh A. Microservices Security Vulnerability Remediation approach using Veracode and Checkmarx. Journal of Artificial Intelligence General Science (JAIGS). 2024. ISSN:3006-4023. Doi: https://doi.org/10.60087/jaigs.v4i1.128.*

14. Top 9 Open-Source SAST Tools | Wiz. Wiz.Io. 2025, February 14. URL: https://www.wiz.io/academy/top-open-source-sast-tools.

15. Fredj O. B., Cheikhrouhou O., Krichen M., Hamam H., Derhab A. An OWASP Top Ten Driven Survey on Web Application Protection Methods. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 12528 LNCS. 2021. 235–252. Scopus. Doi: https://doi.org/10.1007/978-3-030-68887-5_14.

16. Tudela F. M., Higuera J.-R. B., Higuera J. B., Montalvo J.-A. S., Argyros M. I. On combining static, dynamic and interactive analysis security testing tools to improve owasp top ten security vulnerability detection in web applications. Applied Sciences (Switzerland). 2020.10(24). 1–26. Scopus. Doi: https://doi.org/ 10.3390/app10249119.

REVIEW OF MODERN AUTOMATED SOFTWARE SECURITY TESTING TOOLS

M. V. Maksymovych, L. Z. Mychuda

Lviv Polytechnic National University,
Department of Information Technologies Security

Nowadays, with the rapid development of modern technologies in software engineering, active digitalization, and the migration of many services online, ensuring the security of these services in terms of integrity, confidentiality, and availability of information has become more important than ever. The level of application security directly depends on investments made in security during software development. Thus, it is crucial for developers to prioritize not only the implementation of functional requirements but also actively address non-functional requirements related to application security. Ensuring application security involves utilizing various tools designed to minimize the probability of successful cyber-attacks. Today, numerous such tools exist, including both commercial and open-source solutions. The purpose of this article is to analyze modern tools for automated software security testing, comparing their capabilities, implementation costs, and integration complexity into the software development lifecycle. Additionally, the article aims to identify the advantages and disadvantages of commercial and open-source solutions, as well as to outline prospective directions for further research aimed at simplifying the integration of free tools into the application security assurance practice.

Keywords: automation, cybersecurity, containerization, protection, software, testing.