УДК 004.56

**Д.Ю. Желнитський**
**В.В. Лукічов**

# ПРИНЦИП РОБОТИ ТА МЕТОДИ ІНСТРУМЕНТІВ ДИНАМІЧНОГО ТЕСТУВАННЯ БЕЗПЕКИ

Вінницький національний технічний університет

***Анотація***

*Розглянуто методи та принципи динамічного тестування безпеки програмних застосунків (DAST). Проведено аналіз основних принципів функціонування та можливостей DAST. Визначено основні типи вразливостей, які виявляються за допомогою DAST, та класифіковано методи тестування. Результати дослідження підкреслюють важливість використання DAST для підвищення безпеки програмного забезпечення.*

**Ключові слова:** динамічне тестування безпеки, DAST, програмне забезпечення, вразливості.

***Abstract***

*The paper examines methods and principles for dynamic application security testing (DAST). It analyzes the basic principles and capabilities of DAST, identifying key vulnerabilities detected using DAST and classifying testing methods. The results highlight the importance of DAST for enhancing software security..*

**Keywords**: dynamic application security testing, DAST, software, vulnerabilities.

## Вступ

У сучасному світі, де кількість кібератак зростає з кожним днем, забезпечення безпеки програмного забезпечення стає критично важливим завданням. Вразливості в програмних застосунках можуть призвести до значних фінансових втрат, витоку конфіденційної інформації та втрати репутації компаній. Тому питання тестування безпеки програмного забезпечення набуває особливої важливості. Динамічне тестування безпеки (DAST) є одним із ключових методів виявлення вразливостей, оскільки дозволяє оцінювати безпеку застосунків у режимі реального часу, аналізуючи їх поведінку під час виконання [1].

## Результати дослідження

DAST (Dynamic Application Security Testing) — це метод тестування безпеки, відомий як "чорна скринька". Він дозволяє знаходити вразливості та слабкі місця в працюючих додатках шляхом ін'єкції шкідливих навантажень для виявлення потенційних недоліків, що можуть спричинити атаки, такі як SQL-ін'єкції або міжсайтовий скриптинг (XSS). Інструменти DAST особливо корисні для виявлення таких проблем, як перевірка вхідних або вихідних даних, проблеми з автентифікацією та помилки в конфігурації сервера. DAST дозволяє проводити динамічне тестування безпеки додатків у режимі реального часу, коли вони працюють. Такий підхід дозволяє виявити вразливості, які можуть бути пропущені під час статичного аналізу вихідного коду [2].

Методи DAST можна класифікувати за типом вразливостей, методами тестування, рівнем інтеграції з розробницьким процесом та цільовою платформою. За типом вразливостей розрізняють:

- SQL-ін'єкції: зловмисник вставляє шкідливі SQL-коди в поля вводу додатка, що дозволяє виконувати небезпечні операції на базі даних.
- XSS атаки: зловмисники вставляють шкідливі скрипти на веб-сторінки, що може призвести до крадіжки сесійних файлів cookie та інших небажаних дій.
- Проблеми з перевіркою вхідних даних: належна перевірка може запобігти ін'єкціям коду, переповненню буфера тощо.

– Автентифікація та авторизація: помилки можуть дозволити несанкціонованим користувачам доступ до захищених ресурсів.

– Конфігураційні помилки серверу: відсутність шифрування, відкриті порти, некоректні права доступу.

За методами тестування виділяють аналіз на основі підписів, евристичний аналіз та фаззинг. Метод аналізу на основі підписів використовує базу даних відомих вразливостей і відповідних шаблонів атак. Інструменти DAST порівнюють поведінку тестованого додатка з цими шаблонами для виявлення відомих вразливостей. Перевага цього методу в його швидкості та ефективності для виявлення вже задокументованих проблем, однак він може не виявити нові, ще не відомі загрози. Евристичний аналіз використовує набори правил та евристик для виявлення потенційно нових або невідомих вразливостей. Цей метод може включати аналіз аномальної поведінки додатка, підозрілих моделей вводу та виходу даних тощо. Він ефективний для виявлення нових загроз, проте може бути менш точним і вимагати більше ресурсів. Фаззинг полягає у введенні великої кількості випадково згенерованих даних у додаток з метою виявлення помилок у його обробці. Цей метод може виявити неочікувані вразливості, спричинені нетиповими або неправильними вхідними даними. Фаззери можуть виявити різноманітні проблеми, від простих помилок у перевірці вводу до складних вразливостей, що спричиняють крах додатка.

За рівнем інтеграції з розробницьким процесом методи DAST поділяються на автономні та інтегровані інструменти. Автономні інструменти працюють незалежно від процесу розробки і можуть запускатися на вимогу. Вони ідеально підходять для періодичних перевірок безпеки або для використання зовнішніми командами аудиту. Недолік таких інструментів у тому, що вони можуть не бути інтегрованими в автоматизовані процеси розробки, що може знижувати їх ефективність у виявленні вразливостей на ранніх етапах. Інтегровані інструменти DAST працюють у середовищах розробки (IDE) або в системах безперервної інтеграції та доставки (CI/CD). Вони автоматично запускають сканування під час написання коду або під час збірки додатка, що дозволяє виявити вразливості на ранніх етапах розробки. Ці інструменти забезпечують безперервну безпеку та полегшують виправлення вразливостей до того, як вони потраплять у виробниче середовище.

За цільовою платформою інструменти DAST можуть бути орієнтовані на веб-додатки та інфраструктурні сервіси. Інструменти DAST, орієнтовані на веб-додатки, спеціалізуються на виявленні вразливостей, властивих веб-технологіям. Вони можуть перевіряти безпеку HTTP/HTTPS трафіку, аналізувати веб-форми, кукі, сесії та інші аспекти веб-додатків. Такі інструменти враховують специфіку роботи з браузерами та веб-серверами. Інструменти, спрямовані на аналіз безпеки мережевих сервісів, серверів, баз даних та інших інфраструктурних компонентів, перевіряють налаштування серверів, мережеві протоколи, механізми аутентифікації та інші аспекти інфраструктури.

Основним принципом динамічного тестування є аналіз HTTP/HTTPS запитів та відповідей. Тестер або інструмент генерує запити до веб-додатку, і на основі отриманих відповідей робить висновки про наявність вразливостей. При аналізі HTTP/HTTPS запитів можна виявити різні типи вразливостей, такі як SQL-ін'єкції, шляхом відправлення запитів з різними SQL-перевірками і аналізу відповідей на наявність ознак уразливостей. Також можна тестувати безпеку аутентифікації, перевіряючи механізми входу на предмет наявності слабких місць, таких як підбір паролів або уразливості у механізмах відновлення пароля. Аналіз міжсайтового скриптингу (XSS) передбачає відправлення запитів із спеціальними скриптами та перевірку, чи виконується введений код на стороні клієнта. Ще одним важливим принципом є маніпуляція сесіями. Веб-додатки зазвичай використовують сесії для зберігання інформації про користувачів між запитами. Інструменти динамічного тестування можуть маніпулювати сесійними ідентифікаторами, змінювати їх, підставляти значення інших користувачів з метою виявлення вразливостей типу CSRF або проблем з керуванням сесіями. При маніпуляції сесіями можна тестувати захищеність сесій, змінюючи значення сесійних ідентифікаторів з метою доступу до інформації інших користувачів. Також можна перевіряти наявність уразливостей типу CSRF, відправляючи запити від імені користувача без його відома, щоб перевірити, чи можливо здійснити небажані дії. Аналіз механізмів відновлення сесій передбачає перевірку надійності механізмів, які використовуються для відновлення сесій після виходу або переривання. Інструменти динамічного

тестування використовують різні методи для виявлення вразливостей, включаючи автоматичне сканування та ручне тестування. Автоматичне сканування дозволяє швидко перевірити додаток на наявність загальновідомих вразливостей, тоді як ручне тестування дозволяє детально аналізувати специфічні аспекти безпеки. Методи виявлення вразливостей включають автоматичне сканування, яке використовує бази даних відомих вразливостей для швидкого виявлення потенційних проблем, і ручне тестування, що передбачає детальний аналіз певних аспектів безпеки шляхом проведення спеціалізованих атак або аналізу реакцій додатку на неочікувані дії. Комбінація автоматичного та ручного тестування дозволяє отримати більш комплексну картину безпеки. Після виявлення вразливостей вони документуються у вигляді детального звіту. Цей звіт містить опис кожної вразливості, її потенційний вплив, метод, використаний для її виявлення, та рекомендовані кроки з усунення.

## Висновки

Розглянуто методи та принципи динамічного тестування безпеки програмних застосунків. Проведено аналіз основних принципів функціонування DAST. Визначено основні типи вразливостей, які виявляються за допомогою DAST, та класифіковано методи тестування. Результати дослідження підкреслюють важливість використання DAST для підвищення безпеки програмного забезпечення.

## СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ


1. Williams, A., Wichers, D. OWASP Top 10: The Ten Most Critical Web Application Security Risks. 2017.
2. OWASP Foundation. Dynamic Application Security Testing (DAST). [Електронний ресурс]. – Режим доступу: https://owasp.org/www-project-devsecops-guideline/latest/02b-Dynamic-Application-Security-Testing



***Желнитський Дмитро Юрійович*** – студент групи 1БС-20б, факультет інформаційних технологій і комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, lasswerds@gmail.com.

***Лукічов Віталій Володимирович*** – к. т. н., доцент, доцент кафедри захисту інформації, Вінницький національний технічний університет, м. Вінниця, e-mail: lukichov.vitalyi@vntu.edu.ua.

***Dmytro Zhelnytskyi*** – student of group 1BS-20b, Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, lasswerds@gmail.com.

***Vitaly Lukichev*** – associate professor at the Department of Information Security, Vinnytsia National Technical University, Vinnytsia, email: lukichov.vitalyi@vntu.edu.ua .