КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 4(20), 2023
ISSN 2663 - 4023
DOI 10.28925/2663-4023.2023.20.6271
УДК 004.056

**Трофименко Олена Григорівна**
кандидат технічних наук, доцент, доцент кафедри інформаційних технологій,
Національний університет «Одеська юридична академія», м. Одеса, Україна,
ORCID ID 0000-0001-7626-0886
*trofymenko@onua.edu.ua*

**Дика Анастасія Іванівна**
асистент кафедри інформаційних технологій,
Національний університет «Одеська юридична академія», м. Одеса, Україна,
ORCID ID 0000-0002-4196-8734
*dyka.anastasiia@gmail.com*

**Лобода Юлія Геннадіївна**
кандидат технічних наук, доцент, доцент кафедри інформаційних технологій,
Національний університет «Одеська юридична академія», м. Одеса, Україна,
ORCID ID 0000-0001-7083-552X
*jul.loboda@gmail.com*

# АНАЛІЗ ІНСТРУМЕНТІВ ТЕСТУВАННЯ ВЕБЗАСТОСУНКІВ

**Анотація**. У статті проаналізовані сучасні методи та інструменти, які використовуються для тестування безпеки вебзастосунків. Поширеність порушень безпеки вебзастосунків та важливість їх запобігання зробило тестування безпеки невіддільною складовою життєвого циклу розробки відповідного ПЗ, яка має виявляти вразливості, пов'язані із забезпеченням цілісного підходу до захисту програми від хакерських атак, вірусів, несанкціонованого доступу до конфіденційних даних. Для виявлення уразливостей безпеки є різні інструменти тестування безпеки, серед яких популярними є: статичне та динамічне тестування безпеки (SAST та DAST), інтерактивне тестування (IAST), аналіз складу ПЗ (SCA), самозахист програми під час виконання (RASP), брандмауери (WAF), керування станом захисту хмарних середовищ (CSPM). Аналіз сучасних інструментів тестування безпеки показав, що всі вони мають свої переваги і недоліки через специфіку своєї організації. Комбінування та використання переваг кожного з них може забезпечити високий рівень безпеки програмного вебпродукту. Можливими проблемами, пов’язаними з аспектом вебтестування безпеки, є: зламані або ненадійні паролі, переповнення буфера, маніпулювання прихованими полями, ненадійне використання криптографії, перехоплення файлів cookie, неправильні конфігурації сервера, слабке керування сеансами, розкриття конфіденційних даних, маніпуляції з параметрами, соціальне хакерство, неадекватна перевірка введених даних тощо. Зосередження на різних питаннях і проблемах, пов’язаних із тестуванням безпеки вебзастосунків, надає значні дивіденди у виявленні та усуненні різноманітних ризиків, уразливостей, атак, загроз, вірусів тощо. Щоб адаптуватися до динамічної та неоднорідної природи Інтернету та якнайкраще забезпечити захист вебзастосунків, ефективним є комплексний і збалансований підхід до тестування їх безпеки та вибору відповідних засобів.

**Ключові слова**: тестування безпеки; вебзастосунок; безпека вебзастосунків; уразливості безпеки; інструменти тестування.

## ВСТУП

Інформатизація усіх сфер нашого життя дозволила підтримувати комунікацію, стабілізувати та координувати процеси суспільної діяльності спочатку за умов пандемії COVID-19, а тепер за умов повномасштабної війни в Україні. Але разом зі зростанням попиту на програмні вебресурси зростають ризики та небезпеки через зацікавленість зловмисників у зламі та доступі до баз даних клієнтських систем. Тому на команду

розробників програмного забезпечення (ПЗ) покладено відповідальність за захист коду програмних продуктів.

**Постановка проблеми.** Спектр ризиків та вразливостей безпеки сайтів і вебзастосунків доволі широкий, оскільки вони стосуються різних складових розробки: баз даних та обробки запитів, ідентифікації та автентифікації, доступності та пропускної здатності серверів, дизайну та сумісності з браузерами, кросплатформеності та портабельності тощо. У таких багатомодульних системах закладається багато взаємодій поміж складовими, а тому критично важливо для функції безпеки виявити, ретельно проаналізувати, ідентифікувати та усунути усі помилки й уразливі місця. Метою тестування безпеки є підтримка на низькому рівні кількості критичних помилок задля уникнення станів непрацездатності системи. Саме тому атрибути безпеки варто розглядати як частину всіх рівнів тестування ПЗ.

**Аналіз останніх досліджень і публікацій.** Аналізом засобів тестування веббезпеки присвячено не так багато досліджень. Переважно дослідників цікавлять проблеми тестування та аналіз можливих уразливостей ПЗ. У дослідженні [1] проаналізовано сукупність знань 80-ти технічних статей, пов'язаних із тестуванням безпеки вебзастосунків, опублікованих з 2005 по 2020 рік. У роботі [2] виявлено 69 проблем, з якими стикаються розробники та користувачі вебпрограм. У статтях [3,4] проаналізовано методологію та критерії, які використовуються для кількісної оцінки якості програмних сканерів безпеки вебзастосунків, зазначено, що недоліки й обмеження таких сканерів пов'язані з низьким охопленням тестування і неточними результатами. У дослідженні [5] порівнюються результати тестування різних інструментів автоматизованого і ручного тестування, зазначено, що ручне тестування лишається важливим, оскільки є такі типи вразливостей, які можна виявити лише засобами ручного тестування і досвіду тестувальника. Проведений аналіз публікацій виявив актуальність пошуку ефективних інструментів щодо мінімізації ризиків та вразливостей безпеки сайтів.

**Мета статті**: проаналізувати сучасні інструменти, які використовуються для тестування безпеки сайтів та вебзастосунків.

## РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Поширеність порушень безпеки вебзастосунків та важливість їх запобігання зробило тестування безпеки невіддільною складовою життєвого циклу розробки відповідного ПЗ, яка має виявляти вразливості, пов'язані із забезпеченням цілісного підходу до захисту програми від хакерських атак, вірусів, несанкціонованого доступу до конфіденційних даних. Тестування веббезпеки перевіряє бізнес-логіку, проблеми автентифікації та авторизації, кодування вхідних і вихідних даних, а також інші можливі ризики, щоб виявити уразливі місця в безпеці і переконатися, що всі функції вебзастосунків безпечні.

Для тестування безпеки важливо враховувати різницю між ризиком та ефективністю розроблених тестів безпеки щодо пом'якшення відповідного ризику безпеки для аналізу рентабельності інвестицій у безпеку [6].

Незалежно від використовуваної методології та практик розробки ПЗ (Agile, DevOps / DevSecOps, швидкої розробки застосунків (RAD) тощо), варто інтегрувати тести безпеки у робочі процеси неперервної інтеграції та розгортання (Continuous Integration / Continuous Delivery, CI/CD), щоб підтримувати безпеку на належному рівні і виявляти наявні уразливості. Методологія Agile та практики DevOps змінили способи розробки та доставки програмного забезпечення, і тим самим прискорили життєвий цикл

розробки від написання коду до релізу. Зараз групи розробників випускають ПЗ неперервно і значно швидше, ніж раніше, адже вони автономно приймають рішення про використовувані технології та їх впровадження. DevSecOps надає різноманітні інструменти і технології для усунення недоліків та забезпечення швидкого автоматизованого оцінювання безпеки в рамках конвеєра CI/CD. Проте і традиційні, і новітні наскрізні підходи до забезпечення безпеки вебзастосунків, включаючи DevSecOps, матимуть проблеми, якщо інструменти та процеси тестування безпеки залишати незмінними без оновлення та вдосконалення.

Для виявлення уразливостей безпеки є різні інструменти, підходи і практики тестування безпеки щодо їх застосування [7] – [12] (рис. 1):

- статичне тестування безпеки (Static Application Security Testing, SAST);
- динамічне тестування безпеки застосунків (Dynamic Application Security Testing, DAST);
- аналіз складу ПЗ (Software Composition Analysis, SCA);
- інтерактивне тестування (Interactive Application Security Testing, IAST);
- самозахист програми під час виконання (Runtime Application Self-Protection, RASP);
- брандмауери вебпрограм (Web Application Firewalls, WAF);
- керування станом захисту хмарних середовищ (Cloud Security Posture Management, CSPM) тощо.

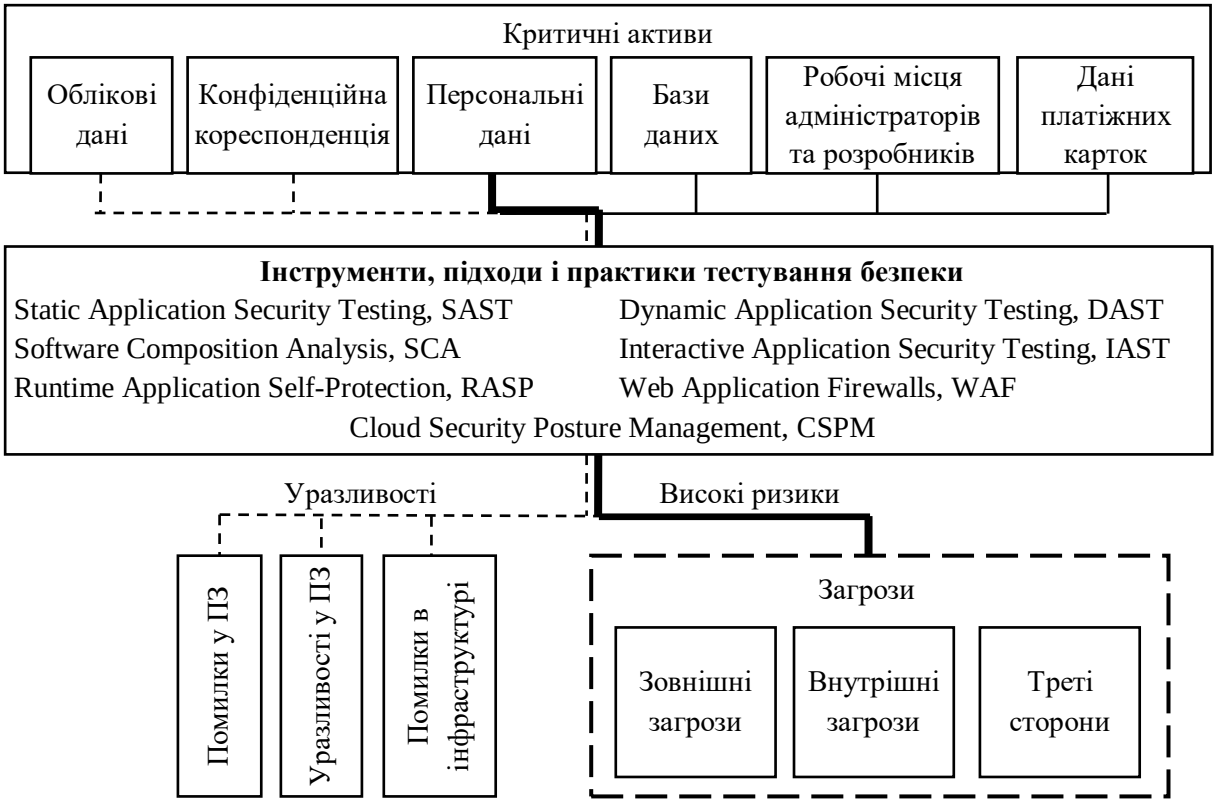


*Рис. 1. Виявлення уразливостей безпеки сайтів та вебзастосунків*

Всі інструменти тестування безпеки не є рівнозначними і взаємозамінними, оскільки вони мають різну специфіку оцінки, виявлення та реагування на ті чи інші різновиди вразливостей:

1) *інструмент безпеки DAST* є, по суті, методом тестування чорної скриньки, який можна інтегрувати на ранньому етапі життєвого циклу розробки ПЗ. Він проводить оцінку вразливостей ззовні, оскільки не має доступу до вихідного коду програми, і спрямований на те, щоб зменшити ризики та виявити вразливості, якими може скористатися зловмисник. DAST слід виконувати до запущеної програми перед її розгортанням в середовищі, подібному до робочого. Це дозволяє використовувати підхід «ззовні всередину» для тестування застосунків на наявність умов експлуатації, які неможливо виявити у статичному стані;

2) *SAST*, також відоме як тестування білої скриньки, сканує та аналізує програму ще до компіляції коду. Цей підхід використовують для перевірки внутрішньої структури ПЗ та коректності її інтеграції із зовнішніми системами. SAST досліджує код задля виявлення можливих SQL-ін'єкцій та інших уразливостей безпеки ПЗ. У DevSecOps це тестування зазвичай інтегрується в середовища розробки для оперативного зворотного зв’язку про ризики безпеки.

Методи SAST і DAST використовуються по-різному та доповнюють один одного. І те й інше тестування є доцільними і корисними для комплексного тестування безпеки. Вони знаходять різні типи уразливостей та ефективні на різних фазах життєвого циклу розробки ПЗ;

3) *інструменти SCA* зазвичай аналізують програми в процесі розробки і потрібні для ідентифікації вбудованих open source компонентів і бібліотек та виявлення у них уразливостей. SCA доповнює SAST, адже знаходить уразливості, які неможливо виявити шляхом перевірки вихідного коду;

4) *IAST* поєднує підходи SAST і DAST, що дозволяє проявити переваги їх обох. Наприклад, що стосується покриття коду, то і статичне, і динамічне тестування пропускають великі частини програмного коду: SAST не перевіряє бібліотеки і фреймворки, що значно обмежує аналіз уразливостей, а DAST може перевіряти лише відкриту поверхню програми. IAST же перевіряє всю програму зсередини, включно з бібліотеками, фреймворками, потоками даних, конфігураціями, HTTP-запитами та відповідями на них, інформацією про внутрішні підключення тощо. Тож в IAST покриття всієї кодової бази значно краще [9]. Крім того, статичні й динамічні інструменти погано масштабуються і зазвичай потребують експертів для налаштування, запуску та інтерпретації результатів. При застосуванні IAST розмір і складність програми ніяк не впливають на продуктивність тестування, тобто можна безпроблемно тестувати великі за розміром програми. Оскільки інструменти інтерактивного тестування працюють всередині програми, то можна у режимі реального часу виявляти й усувати уразливості, практично без помилкових спрацьовувань і з миттєвим зворотним зв’язком. І при цьому програма тестується не періодично, як при SAST і DAST, а безперервно й автоматично. Важливо, що IAST легко інтегрується з наявними тестуваннями безпеки і дозволяє використовувати власні правила для персоналізації покриття загроз для конкретних підприємств;

5) *інструмент самозахисту застосунку під час виконання – RASP* – дозволяє на розгорнутих програмах безперервно відстежувати та аналізувати поведінку і структуру застосунку, повідомляти про атаки та блокувати неавторизовані дії зловмисника. Інструмент RASP інтегрується в програму та використовує підхід, подібний до штучного інтелекту, щоб постійно перехоплювати будь-які виклики до програми та перевіряти їхню безпеку. Він перевіряє та позначає трафік зловмисного програмного забезпечення, що надходить до програми, ще до того, як таке зловмисне ПЗ буде запущено всередині програми. Попри те, що RASP створює додаткове інфраструктурне навантаження на продуктивність програмного середовища, такого роду засоби можна використовувати проти атак на вебзастосунок без втручання людини. RASP зосереджено на захисті

окремої програми, а тому його не використовують для захисту всієї мережі. Це добре з точки зору пріоритету безпеки, оскільки йому потрібно лише контролювати кожен вхід, вихід і внутрішній процес у програмі, яку він захищає [10]. RASP працює всередині програми під час її виконання і є, по суті, інструментом безпеки, а не тестування [11]. Він інтегрується з програмою і може контролювати виконання цієї програми, захищаючи її, навіть якщо захист периметра мережі порушено, а програма містить уразливості безпеки;

6) *брандмауери вебпрограм (WAF)* відстежують трафік на рівні програм, виявляють потенційні атаки та спроби використання уразливостей шляхом зіставлення шаблонів за попередньо визначеними правилами. WAF пом'якшує атаки DDoS і добре справляється з блокуванням атак грубої сили, SQL-ін'єкцій та XSS. Він захищає програми, фільтруючи, відстежуючи та блокуючи зловмисні запити або трафік, що надходять у програму. При цьому, так само як і для RASP, усі дії фіксуються у журналі подій безпеки для можливості подальшого аудиту та виявлення відхилень у поведінці програмної системи. WAF є гнучким інструментом і підтримує гібридне розгортання. Зокрема, WAF може виявляти атаки з використанням автоматизованих засобів. Проте при всіх перевагах WAF не покриває автоматизовану перевірку усіх вразливостей застосунку. Існує багато способів обходу WAF, які дозволяють зловмиснику продовжувати атаки на систему. Застосування брандмауера недостатньо, якщо мережа захищена не належним чином. Для перевірки забезпечення постійного захисту та безпеки корпоративної мережі доцільно проводити тестування на проникнення через міжмережевий екран. Тести на проникнення в брандмауер є критично важливим і можуть допомогти групам безпеки виявити уразливі місця в архітектурі мережі;

7) *CSPM* автоматизує керування ресурсами і сервісами хмарних середовищ, включаючи візуалізацію та оцінку стану захисту, виявлення помилкових конфігурацій для забезпечення безпеки корпоративних хмарних застосунків. Сучасні компанії та організації мають складні інформаційні системи, якими може бути важко керувати через численні ресурси та обов'язки, а тому важко відстежити втрату та нецільове використання основних активів. Рішення Cloud Security Posture Management можуть рекомендувати або автоматично застосовувати методи безпеки на основі внутрішньої політики організації. До функціонала CSPM входить: візуалізація та оцінка стану безпеки; виявлення помилкових конфігурацій у хмарній інфраструктурі, які можуть залишити неконтрольованими потенційні ризики та вектори атак; моделювання та активне застосування еталонних політик; захист від атак та внутрішніх загроз; розвідка загроз безпеки хмарного середовища для виявлення вторгнень у хмару; відповідність нормативним вимогам і практичним рекомендаціям. CSPM забезпечує для організацій швидкі й ефективні заходи хмарного захисту та автоматизовані передові методи DevSecOps. Серед популярних рішень CSPM, якими можна усунути уразливості хмарної служби, такі: CrowdStrike, Prisma Cloud, Continuity Software, CloudGuard Management, BMC Cloud Security, Lacework, Fugue тощо [12]. Оскільки інструменти CSPM працюють із хмарними ресурсами, то і самі вони є хмарними. Ці системи безпеки повинні мати можливість перетинати платформи та перевіряти роботу численних ресурсів, щоб усунути уразливі місця і забезпечити безпеку даних компаній і організацій у разі складних і гібридних хмарних систем.

Аналіз сучасних інструментів тестування безпеки показав, що всі вони мають свої переваги і недоліки через специфіку їх організації. Комбінування та використання переваг кожного чи то більшості з них може забезпечити високий рівень безпеки програмного вебпродукту.

Розробка автоматизованих інструментів завжди була проблемою тестування вебзастосунків на безпеку. Створити автоматизовані інструменти для тестування

безпеки важче, ніж для тестування функціональності вебзастосунку [13]. Традиційні інструменти не встигають за темпами екосистеми програмного забезпечення. Проблеми, з якими стикаються автоматизовані інструменти для тестування веббезпеки, полягають у тому, що вони повинні йти в ногу з технологіями, що стрімко розвиваються та змінюються (наприклад, RIA), і належним чином інтегруватися в наявні робочі процеси розробки.

Перевагу автоматизованого тестування на проникнення вебзастосунків важко переоцінити, оскільки сканер безпеки вебзастосунків не тільки скорочує час, вартість і ресурси, необхідні для тестування безпеки вебзастосунків, а й вивільняє час тестувальників безпеки. Проте сканери мають свої недоліки та обмеження, які пов'язані з низьким охопленням тестування і неточними результатами. Сканери безпеки не сканують вебпрограми повністю, а інколи пропускають уразливості і генерують неправильні результати тестування. Дослідження кількісного підрахунку результатів сканерів безпеки вебзастосунків [5] виявили відсутність чітко визначеного методу чи критеріїв для оцінки якості та ефективності їх результатів. Нині популярними автоматизованими інструментами тестування на проникнення є: OpenVAS, Wireshark, Burp Suite, Invicti (раніше Netsparker), Astra Pentest, Acunetix, Intruder, Indusface, AppKnox, Veracode, Detectify, ZAP, Wfuzz, Wapiti тощо [14]. Вибір сканера для автоматизованого тестування на проникнення залежить від багатьох чинників: бюджету, який може виділити організація на тестування, вартості та функцій сканера, вимог та потреб у такому тестуванні вебзастосунку, відповідності різноманітним важливим стандартам (PCI-DSS, HIPAA, GDPR, ISO 27001), наявності підтримки обслуговування, деталізації звітів, наявності допомоги у виправленні і відновленні після виявлення уразливостей тощо. До речі, є безплатні інструменти тестування безпеки застосунків, наприклад: ZAP, Wfuzz і Wapiti. Практики використовують різноманітні набори методологій, тестових стендів, сканерів безпеки вебзастосунків і вимірювальних показників для кількісної оцінки сканерів безпеки вебзастосунків. Проте показники вимірювання поверхневого покриття та кількості посилань надто неоднозначні, щоб через них визначати тестове покриття того чи іншого сканера безпеки, оскільки сучасні вебзастосунки складаються з численних вебелементів, які є критичними для оцінки уразливості.

Поруч із застосуванням різних автоматизованих інструментів не втрачає актуальність і ручне тестування безпеки вебзастосунків. Сама по собі автоматизація не в змозі гарантувати ретельне тестування програми з точки зору безпеки. Так, деякі типи вразливостей автоматизовані тести можуть пропустити, наприклад, помилки авторизації, атаки сліпих SQL-ін'єкцій, логічні недоліки, міжсайтові сценарії, вразливості контролю доступу [5]. Їх можна виявити лише за допомогою ручного тестування та спостережень, інтуїції, досвіду й інтелекту тестувальника безпеки. І хоча ручне тестування потребує багато часу, у порівнянні з автоматизованим, і не завжди є точним через людські помилки, а, отже, і менш надійним, саме ручне тестування є нині найпоширенішим методом тестування безпеки. Під час такого тестування Pentester керується не лише власним досвідом, а використовує відповідні інструменти, серед яких популярними нині є: Nmap, Burp Suite, Metasploit, Nessus, Astra Pentest, WireShark, Nikto, Intruder, W3AF, SQLmap, Zed Attack Proxy, Acunetix, Aircrack тощо [15, 16]. Наприклад, безплатний і доступний Nmap використовується для сканування великих мереж і допомагає перевіряти хости, служби та виявляти вторгнення. Проксі-сервер Burp Suite допомагає перехоплювати та змінювати запити, надіслані на сервер, що дозволяє імітувати атаки і збирати інформацію про їхню ціль. Платформу Metasploit тестувальники безпеки використовують для перевірки безпеки мережі або злому віддаленого комп'ютера шляхом розробки та виконання коду експлойту на віддаленій цільовій машині.

Не існує універсального засобу, методу чи інструмента тестування безпеки, який би підходив усім вебзастосункам. Ручне тестування безпеки вручну є трудомістким процесом і вимагає розуміння програми для виконання тесту. Pentester використовує свої особисті навички і досвід, щоб виявити вразливості програми. Автоматизовані інструменти можуть бути менш дорогими і їх можна використовувати для тестування більшої кількості цілей. У будь-якому разі, враховуючи динамічно зростаючі загрози кібербезпеці, тестування безпеки вебзастосунку є критично важливим для виявлення уразливостей як у програмному коді, так і в архітектурі мережі. Тому важливо використовувати різні ефективні інструменти автоматизованого тестування безпеки, а також проводити ретельне ручне тестування, щоб максимально забезпечити виявлення та усунення уразливостей, експлойтів і слабких місць у роботі вебзастосунку. Сканери уразливостей, аналізатори коду та аналізатори компонування ПЗ є автоматизованими інструментами, тоді як фреймворки атак і засоби зламування паролів є ручними інструменти тестувальника безпеки. При виборі методів і засобів тестування безпеки варто враховувати різні чинники: вартість інструменту, можливості автоматизації, функції звітування інструменту і надання доказів уразливостей, які допоможуть вжити правильних заходів і зведуть до мінімуму помилкові спрацьовування. Одні інструменти добре знаходять недоліки безпеки, інші мають гарні можливості звітування, є прості у використанні, а деякі пропонують багатий набір функцій. Не просто знайти найкращий інструмент тестування безпеки програм для того чи іншого середовища і того чи іншого вебзастосунку. Кожне ПЗ має свої унікальні функції. Отже, при такому пошуку і виборі варто дотримуватись правильного балансу між швидкістю, точністю, покриттям і вартістю.

## ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

Проведений аналіз сучасних методів та інструментів тестування вебзастосунків виявив наявність різних підходів і розмаїття засобів для захисту програмних продуктів. З такою кількістю підходів до тестування безпеки вебзастосунків доволі не просто зорієнтуватися і зрозуміти, які саме методи використовувати або коли і в якій послідовності застосовувати ті чи інші засоби тестування. Нерідко команди з тестування безпеки на практиці стикаються з високими вимогами, але застарілими інструментами і методами безпеки, розробленими для «дохмарної епохи». Досвід показує, що за умов зростання ризиків та небезпек через зацікавленість зловмисників у зламі клієнтських систем не існує єдиної методики, яка б могла ефективно розв’язати всі проблеми захисту. Щоб адаптуватися до динамічної та неоднорідної природи Інтернету та якнайкраще забезпечити захист вебзастосунків, ефективним є комплексний і збалансований підхід до тестування їх безпеки та вибору відповідних засобів. Доцільним є баланс у використанні декількох різних методів і підходів, які будуть доповнювати і посилювати один одного. Так, для автоматизованого тестування безпеки на різних етапах розробки варто залучати засоби IAST, RASP, WAF та інші, а область ручного тестування для економії часу та ресурсів варто звузити і зосередити на тих областях, в яких автоматизовані тести можуть пропустити якісь уразливості, наприклад, помилки авторизації чи то бізнес-логіки. З іншого боку автоматизовані процеси тестування дозволяють суттєво знизити кількість виробничих проблем і помилок через усунення людського фактора. При цьому можна проводити кілька тестів одночасно в режимі 24/7 і збільшити обсяги перевірок. Неперервне тестування сприяє швидшому виявленню уразливостей, а тому виправлення стають менші за масштабом і потребують менше часу. Це в решті решт скорочує час між релізами програмних продуктів та дозволяє командам DevOps і безпеки працювати

паралельно. Якість ПЗ при цьому також покращується, оскільки з'являється час для виявлення та розв'язання проблем ще в процесі розробки.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Aydos, M., Aldan, Ç., Coşkun, E., Soydan, A. (2022). Security testing of web applications: A systematic mapping of the literature. *Journal of King Saud University - Computer and Information Sciences*, 34(9), 6775-6792, https://doi.org/10.1016/j.jksuci.2021.09.018.

2 Mubshra, Q., Shahid, F., Mohd, H., Nizam, B., Md, N., Atif, A. (2021). A Rigorous Approach to Prioritizing Challenges of Web-Based Application Systems. *Malaysian Journal of Computer Science*, 34, https://doi.org/10.22452/mjcs.vol34no2.1.

3 Lim, S., Norafida, I., Syed, S. (2018). The approaches to quantify web application security scanners quality: A review. *International Journal of Advanced Computer Research*, 8, 285-312, https://doi.org/10.19101/IJACR.2018.838012.

4 Shahid, J., Hameed, M., Javed, I., Qureshi, K., Ali, M., Crespi, N. (2022). A Comparative Study of Web Application Security Parameters: Current Trends and Future Directions. *Applied Sciences*, 12, 4077, https://doi.org/10.3390/app12084077.

5 Dukes, L., Yuan, X., Akowuah, F. (2013). A case study on web application security testing with tools and manual testing. *Proceedings of IEEE Southeastcon-2013*, 1-6. https://doi.org/10.1109/SECON.2013.6567420.

6 Web Security Testing Guide. https://owasp.org/www-project-web-security-testing-guide/stable/2-Introduction/

7 The complete guide to developer-first application security. GitHub. https://assets.ctfassets.net/wfutmusr1t3h/397ElOPOMY8H6wSwfFvf4z/06ed44457b6fb3a9bd77134c098749ea/GitHubAdvanced_SecurityEbook.pdf.

8 Software Testing Help. Differences between SAST, DAST, IAST, and RASP. https://www.softwaretestinghelp.com/differences-between-sast-dast-iast-and-rasp/.

9 Interactive Application Security Testing. https://www.contrastsecurity.com/glossary/interactive-application-security-testing

10 What Is RASP: Runtime Application Self Protection. https://www.softwaretestinghelp.com/rasp-tutorial/

11 Інструменти тестування безпеки: SAST / DAST / IAST / RAPS. https://qagroup.com.ua/publications/instrumenty-testuvannia-bezpeky-sast-dast-iast-raps/

12 Top 28 Cloud Security Posture Management (CSPM) Tools. https://startupstash.com/cloud-security-posture-management-tools/

13 Трофименко, О.Г., Пастернак, Ю.Ю., Манаков, С.Ю., Лобода, Ю.Г. (2021). Автоматизація тестування вебсайтів електронної комерції. *Сучасна спеціальна техніка*, 2(65), 46-59, https://doi.org/10.36486/mst2411–3816.2021.2(65).5.

14 Nivedita, J. 10 Best Automated Penetration Testing Tools of 2023. https://www.getastra.com/blog/security-audit/automated-penetration-testing-software/

15 Saumick, B. 17 Best Penetration Testing Tools/Software of 2023 [Reviewed]. https://www.getastra.com/blog/security-audit/best-penetration-testing-tools/

16 Keshav, M. Automated VS Manual Security Testing – Which One to Choose? https://www.getastra.com/blog/security-audit/manual-security-testing/

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 4 (20), 2023
ISSN 2663 - 4023

**Trofymenko Olena,**
PhD, Associate Professor, Associate Professor at the Department of Information Technologies of the National University "Odessa Law Academy", Odesa, Ukraine,
ORCID ID: 0000-0001-7626-0886
*trofymenko@onua.edu.ua*

**Dyka Anastasiia,**
Assistant of Department of Information Technologies of the National University "Odessa Academy of Law", Odesa, Ukraine,
ORCID ID 0000-0002-4196-8734
*dyka.anastasiia@gmail.com*

**Loboda Yuliia,**
PhD, Associate Professor, Associate Professor at the Department of Information Technologies of the National University "Odessa Law Academy", Odesa, Ukraine,
ORCID ID: 0000-0001-7083-552X
*jul.loboda@gmail.com*

# ANALYSIS OF WEB APPLICATION TESTING TOOLS

**Abstract.** The article analyzes modern methods and tools used for security testing of web applications. The prevalence of security violations of web applications and the importance of their prevention made security testing an integral part of the software development life cycle (SDLC), which should detect vulnerabilities associated with providing a holistic approach to protecting the program from hacker attacks, viruses, unauthorized access to confidential data. To identify security vulnerabilities, there are various security testing tools, among which the popular ones are: static and dynamic application security testing (SAST and DAST), interactive application security testing (IAST), software composition analysis (SCA), runtime application self-protection (RASP), web application firewalls (WAF), cloud security posture management (CSPM). Analysis of modern security testing tools showed that they all have their advantages and disadvantages due to the specifics of their organization. Combining and using the advantages of each of them can ensure a high level of security for a web software product. Possible issues related to the web testing aspect of security are cracked or untrusted passwords, buffer overflows, manipulation of hidden fields, insecure use of cryptography, interception of cookies, incorrect server configurations, weak session management, disclosure of sensitive data, manipulation of parameters, social hacking, inadequate verification of input data, etc. Focusing on various questions and issues related to web application security testing pays significant dividends in identifying and remediating various risks, vulnerabilities, attacks, threats, viruses, and more. To adapt to the dynamic and heterogeneous nature of the Internet and to ensure the best protection of web applications, a comprehensive and balanced approach to testing their security and selecting appropriate tools is effective.

**Keywords**: security testing, web application, web application security, security vulnerabilities, testing tools.

## REFERENCES (TRANSLATED AND TRANSLITERATED)

1 Aydos, M., Aldan, Ç., Coşkun, E., Soydan, A. (2022). Security testing of web applications: A systematic mapping of the literature. *Journal of King Saud University - Computer and Information Sciences*, 34(9), 6775-6792, https://doi.org/10.1016/j.jksuci.2021.09.018.

2 Mubshra, Q., Shahid, F., Mohd, H., Nizam, B., Md, N., Atif, A. (2021). A Rigorous Approach to Prioritizing Challenges of Web-Based Application Systems. *Malaysian Journal of Computer Science*, 34, https://doi.org/10.22452/mjcs.vol34no2.1.

3 Lim, S., Norafida, I., Syed, S. (2018). The approaches to quantify web application security scanners quality: A review. *International Journal of Advanced Computer Research*, 8, 285-312, https://doi.org/10.19101/IJACR.2018.838012.

4 Shahid, J., Hameed, M., Javed, I., Qureshi, K., Ali, M., Crespi, N. (2022). A Comparative Study of Web Application Security Parameters: Current Trends and Future Directions. *Applied Sciences*, 12, 4077, https://doi.org/10.3390/app12084077.

5 Dukes, L., Yuan, X., Akowuah, F. (2013). A case study on web application security testing with tools and manual testing. *Proceedings of IEEE Southeastcon-2013*, 1-6. https://doi.org/10.1109/SECON.2013.6567420.
6 Web Security Testing Guide. https://owasp.org/www-project-web-security-testing-guide/stable/2-Introduction/
7 The complete guide to developer-first application security. GitHub. https://assets.ctfassets.net/wfutmusr1t3h/397ElOPOMY8H6wSwfFvf4z/06ed44457b6fb3a9bd77134c098749ea/GitHubAdvanced_SecurityEbook.pdf.
8 Software Testing Help. Differences between SAST, DAST, IAST, and RASP. https://www.softwaretestinghelp.com/differences-between-sast-dast-iast-and-rasp/.
9 Interactive Application Security Testing. https://www.contrastsecurity.com/glossary/interactive-application-security-testing
10 What is RASP: Runtime Application Self Protection. https://www.softwaretestinghelp.com/rasp-tutorial/
11 Security testing tools: SAST / DAST / IAST / RAPS. https://qagroup.com.ua/publications/instrumenty-testuvannia-bezpeky-sast-dast-iast-raps/
12 Top 28 Cloud Security Posture Management (CSPM) Tools. https://startupstash.com/cloud-security-posture-management-tools/
13 Trofymenko, O., Pasternak, Yu., Manakov, S., Loboda, Yu. (2021). Automation of testing e-commerce websites. *Modern Special Technics*, 2(65), 46-59, https://doi.org/10.36486/mst2411–3816.2021.2(65).5.
14 Nivedita, J. 10 Best Automated Penetration Testing Tools of 2023. https://www.getastra.com/blog/security-audit/automated-penetration-testing-software/
15 Saumick, B. 17 Best Penetration Testing Tools/Software of 2023 [Reviewed]. https://www.getastra.com/blog/security-audit/best-penetration-testing-tools/
16 Keshav, M. Automated VS Manual Security Testing – Which One to Choose? https://www.getastra.com/blog/security-audit/manual-security-testing/