Вісник ЛДУБЖД
Bulletin of Lviv State University of Life Safety
https://journal.ldubgd.edu.ua/index.php/Visnuk
ISSN 2078-4643 (print), ISSN 2708-1389 (online)

UDC 004.056.5:004.415.2
DOI: 10.32447/20784643.30.2024.11

А. І. Івануса, Р. Л. Ткачук, Т. Б. Брич,
В. С. Балацька, А. М. Ткаченко
Львівський державний університет безпеки життєдіяльності, м. Львів, Україна
ORCID: https://orcid.org/0000-0001-9141-8039 – А. І. Івануса
https://orcid.org/0000-0001-9137-1891 – Р. Л. Ткачук
https://orcid.org/0000-0001-6853-1981 – Т. Б. Брич
https://orcid.org/0000-0002-6262-6792 – В. С. Балацька
https://orcid.org/0009-0009-6830-4741 – А. М. Ткаченко
ivaaanusa@gmail.com

МЕТОДИ ТА МОДЕЛІ ПРОЄКТУВАННЯ СИСТЕМИ АВТОМАТИЗОВАНОГО ПОШУКУ ВРАЗЛИВОСТЕЙ У WEB-ДОДАТКАХ

Постановка проблеми. Проведено інформаційний аналіз наукових досліджень за темою безпеки WEB-додатків, і встановлено, що на ринку не вистачає якісних програмних продуктів для сканування вразливостей WEB-додатків. Виявлено, що існуючі інструменти не завжди ефективно виявляють усі види вразливостей, що підвищує ризик успішних атак та потребує вдосконалення підходів до забезпечення безпеки. Вразливості WEB-додатків можуть включати SQL-ін'єкції, міжсайтовий скриптинг (XSS), помилки автентифікації та управління сесіями, використання вразливих компонентів, ненадійні конфігурації безпеки тощо. Тому розробка автоматизованих інструментів для виявлення таких вразливостей є критично важливою для зменшення ризиків.

Мета. Метою дослідження є розробка нових або удосконалення існуючих методів та інструментів для автоматизованого пошуку вразливостей у WEB-додатках з метою підвищення рівня захисту. Зокрема, було визначено, що поєднання методів статичного та динамічного аналізу може підвищити ефективність процесу виявлення потенційних загроз. Дослідження спрямоване на створення надійної основи для розроблення комплексного інструменту, що охоплює широкий спектр вразливостей.

Методи досліджень. Інформаційний аналіз використаний для визначення інструментів сканування вразливостей; системний аналіз – для визначення методів та способів пошуку вразливостей WEB-додатків; порівняльний аналіз – для визначення переваг та недоліків ручного й автоматичного тестування WEB-додатків на вразливості; системний синтез – для розроблення програмного продукту автоматизованого пошуку вразливостей WEB-додатків.

Результати. Для реалізації цієї мети застосовувалися методи інформаційного аналізу для оцінки інструментів, системний аналіз для вивчення методів пошуку вразливостей, порівняльний аналіз для виявлення переваг і недоліків ручного та автоматизованого тестування, а також системний синтез для розроблення програмного продукту. Порівняльний аналіз ручного і автоматизованого тестування показав, що хоча ручне тестування забезпечує більш детальний аналіз і здатне виявити складні логічні вразливості, автоматизоване тестування значно швидше і краще підходить для регулярного сканування великих WEB-додатків. Розроблені базові алгоритми для автоматизованого тестування, включаючи алгоритми для виявлення SQL-ін'єкцій та XSS, забезпечують ефективне сканування та виявлення потенційних загроз. Ці алгоритми здатні проводити глибокий аналіз вхідних даних та відповіді серверів для виявлення ознак атак.

Розроблена концептуальна модель роботи програми для автоматизованого сканування забезпечує надійність і точність процесу. У моделі передбачено використання кількох етапів перевірки: сканування коду, тестування поведінки додатка та аналіз відповідей серверів. Система використовує методи статичного аналізу для оцінки коду без виконання додатка та динамічного аналізу для імітації реальних атак під час виконання додатка. Зокрема, для підвищення точності та зменшення кількості хибних позитивних результатів передбачено застосування алгоритмів машинного навчання.

Висновки. Висновки дослідження підкреслюють важливість застосування автоматизованих систем для підвищення рівня безпеки WEB-додатків. Поєднання ручного та автоматизованого тестування рекомендовано для забезпечення комплексного підходу до захисту. Запропоновані методи та перевірка функціональності розробленого інструменту гарантують його відповідність вимогам сучасної кібербезпеки. Використання такого інструменту допомагає швидко виявляти вразливості та вживати заходів для їх усунення до того, як вони можуть бути використані зловмисниками. Застосування комплексних підходів до безпеки WEB-додатків забезпечує захист конфіденційних даних користувачів і знижує ймовірність успішних атак.

Ключові слова: WEB-додаток, системи автоматизованого пошуку, сканування вразливостей, алгоритм SQL-ін’єкцій, алгоритм XSS, кібербезпека.

A. I. Ivanusa, R. L. Tkachuk, T. B. Brych,
V. S. Balatska, A. M. Tkachenko
Lviv State University of Life Safety, Lviv, Ukraine

METHODS AND MODELS FOR THE DESIGN OF AUTOMATED VULNERABILITY DETECTION SYSTEMS IN WEB APPLICATIONS

Introduction. An analysis of scientific research on web application security highlights a significant shortage of high-quality software tools for detecting vulnerabilities in web applications. Existing tools often fail to identify all potential vulnerabilities, which increases the risk of successful attacks and underscores the need for improved security approaches. Common vulnerabilities in web applications include SQL injections, cross-site scripting (XSS), authentication and session management flaws, usage of vulnerable components, and weak security configurations. Developing automated tools to detect such vulnerabilities is therefore critical for mitigating these risks.

Objective. This study aims to develop new or improve existing methods and tools for automated vulnerability detection in web applications to enhance their security. The research specifically focuses on the integration of static and dynamic analysis methods to improve the effectiveness of detecting potential threats. It seeks to establish a robust foundation for a comprehensive tool capable of addressing a wide range of vulnerabilities.

Methods. This research employs information analysis to evaluate existing vulnerability scanning tools, systematic analysis to identify methods and approaches for detecting web application vulnerabilities, comparative analysis to assess the strengths and weaknesses of manual and automated testing, and systematic synthesis for the development of an automated vulnerability detection system.

Results. To achieve the outlined objectives, information analysis was utilized to assess current tools, systematic analysis was conducted to explore detection methods, and comparative analysis was performed to identify the benefits and limitations of manual versus automated testing approaches. Automated testing emerged as significantly faster and more suitable for regular scans of large-scale web applications, while manual testing demonstrated an advantage in identifying complex logical vulnerabilities. The study developed foundational algorithms for automated testing, including those targeting SQL injections and XSS vulnerabilities, ensuring effective scanning and threat identification. These algorithms perform in-depth analysis of input data and server responses to recognize attack patterns.

A conceptual model for the operation of an automated scanning tool was proposed, emphasizing reliability and accuracy. The model encompasses multiple verification stages, such as code scanning, application behavior analysis, and server response evaluation. Static analysis is employed to examine code without execution, while dynamic analysis simulates real-world attacks during runtime. The integration of machine learning algorithms is incorporated to enhance accuracy and minimize false positives.

Conclusion. The findings of this study underline the importance of automated systems in enhancing web application security. Combining manual and automated testing is recommended to achieve a comprehensive security framework. The proposed methods and the validated functionality of the developed tool ensure alignment with modern cybersecurity standards. The adoption of such a tool enables swift identification of vulnerabilities and the proactive implementation of mitigation measures, reducing the risk of exploitation. A holistic approach to web application security is essential to safeguard user confidentiality and reduce the likelihood of successful cyberattacks.

Keywords: Web application, automated detection systems, vulnerability scanning, SQL injection algorithm, XSS algorithm, cybersecurity.

Вступ. Зі зростанням використання додатків також зросла потреба в заходах безпеки для захисту конфіденційних даних. Вразливості WEB-додатків можуть призвести до порушень безпеки, що у свою чергу може призвести до втрати особистої та фінансової інформації користувачів, заподіяння шкоди для репутації організацій і юридичних наслідків. Тому розробка автоматизованого пошуку вразливостей для додатків необхідна з метою виявлення та усунення проблем безпеки до того, як ними зможуть скористатися хакери. Таким чином виникає необхідність проведення дослідження, спрямованого на забезпечення всебічного розуміння методів, робочих інструментів та методології, які можуть використовуватись для розробки механізму автоматизованого пошуку вразливостей у WEB-додатках.

Метою роботи є розроблення методів та робочого інструменту для автоматизованого пошуку вразливостей у WEB-додатках. Відповідно мета роботи потребує здійснення реалізації таких завдань: проведення аналізу інструментів для сканування вразливостей, які доступні на ринку, з метою їх оцінки ефективності. Також необхідно визначити найпоширеніші типи вразливостей, що виявляються у WEB-додатках, а також методи, способи та методології, що використовуються для їх виявлення.

Методи дослідження: інформаційний аналіз використаний для визначення інструментів сканування вразливостей; системний аналіз – для визначення методів та способів пошуку вразливостей WEB-додатків; порівняльний аналіз – для визначення переваг та недоліків ручного й автоматичного тестування WEB-додатків на вразливості; системний синтез – для розроблення програмного продукту автоматизованого пошуку вразливостей WEB-додатків.

Наукова новизна роботи полягає у розробленні методів та моделей автоматизованого пошуку

вразливостей WEB-додатків з метою підвищення рівня їх захисту від потенційних кібератак.

Результати досліджень. Вразливості WEB-додатків – це слабкі місця в системі безпеки, які зловмисники можуть використати для отримання несанкціонованого доступу до конфіденційних даних або порушення цілісності та доступності додатка. Ці вразливості можуть з'явитися під час проєктування та розроблення застосунку або через неадекватні методи забезпечення безпеки під час розгортання та обслуговування.

Слід зазначити, що вразливості WEB-додатків становлять значний ризик для безпеки додатків і конфіденційних даних, з якими вони працюють. Розробники та фахівці з безпеки повинні працювати разом, щоб виявити й усунути ці вразливості та забезпечити безпеку застосунку і його користувачів. Впроваджуючи передові методи безпечної розробки та тестування, організації можуть знизити ризик успішних атак і захистити свої активи від шкоди. У таблиці 1 представлено найпоширеніші вразливості WEB-додатків. У цьому рейтингу представлено тільки шкідливі програми, виключено з нього рекламні програми, які діють вельми настирливо і завдають неприємностей користувачеві, але не завдають шкоди комп'ютеру. Для складання таблиці 1 найбільш поширених вразливостей додатків використано останні доступні дані з OWASP (Open Web Application Security Project), яка регулярно оновлює рейтинг топових вразливостей [1]. Зазначені дані можуть відрізнятися залежно від року та географічного регіону, але загалом вигляді відображають тенденції у сфері кібербезпеки.

Таблиця 1

Найбільш поширені вразливості WEB-додатків

Вразливість	**Опис вразливості**	**Кількість атакованих користувачів, %**
SQL-ін'єкція	Вразливість, при якій зловмисники вводять SQL-код для доступу до баз даних	20-25
Неправильна аутентифікація	Помилки в аутентифікації, що дають змогу зловмисникам обходити перевірку особи	15-20
Міжсайтовий скриптинг (XSS)	Дає можливість вводити шкідливий код на сторонніх сайтах, щоб атакувати користувачів	15-20
Вразливі компоненти	Використання застарілих або вразливих компонентів (бібліотек, плагінів)	10-15
Ненадійне управління сесіями	Вразливість, при якій зловмисники можуть захоплювати сесії користувачів	10-12
Витік конфіденційних даних	Недостатній захист чутливих даних, які можуть бути викрадені	8-10
Ненадійна конфігурація безпеки	Використання слабких налаштувань безпеки або налаштувань за замовчуванням	7-10
Десеріалізація без перевірки	Атаки, які використовують серіалізовані дані для отримання доступу до системи	5-7
Багаторівнева ін'єкція	Комбінація ін'єкційних атак для проникнення в систему	5
Недостатній контроль доступу	Відсутність контролю прав доступу користувачів до певних даних чи функцій	5

Усі представники рейтингу – це, здебільшого, саморозповсюджувані програми та їхні компоненти. Дані показують, що інструментарій атак на бізнес відрізняється від того, що застосовують в атаках на домашніх користувачів. В атаках на корпоративних користувачів значно частіше використовуються експлойти до офісних додатків, шкідливі файли часто виявляються підписаними валідними цифровими сертифікатами, плюс до цього зловмисники намагаються використовувати у своїх цілях доступні легальні програми, щоб довше залишатися непоміченими.

Провівши інформаційний аналіз було встановлено, що найпоширенішими вразливостями WEB-додатків є SQL-ін'єкції, міжсайтовий скриптинг (XSS), підробка міжсайтових запитів (CSRF), зламана автентифікація та управління сесіями, небезпечні прямі посилання на об'єкти.

SQL-ін'єкція – це тип атаки, під час якої зловмисник впроваджує шкідливий код SQL у поля введення програми, щоб отримати несанкціонований доступ до бази даних [2]. Ця вразливість дає змогу зловмиснику переглядати, змінювати або видаляти конфіденційні дані з бази даних. Наслідки успішної атаки шляхом впровадження SQL-коду можуть бути серйозними: від фінансових втрат до компрометації конфіденційної інформації. Щоб запобігти

впровадженню SQL, розробники повинні використовувати запити, що містять конкретні параметри і перевірку їх введення, з метою переконання того факту, що введені дані користувачем очищаються додатком перед їх обробкою.

Міжсайтовий скриптинг (XSS) – це тип атаки, під час якої зловмисник впроваджує шкідливі скрипти або код на WEB-сторінку, яку переглядають інші користувачі [3]. Ці скрипти можуть використовуватися для крадіжки конфіденційної інформації або виконання несанкціонованих дій від імені користувача. XSS-атаки можна розділити на дві категорії: збережені та відбиті. Збережені XSS-атаки дають змогу зловмиснику впровадити шкідливий код, який постійно зберігається на сервері, а відбиті XSS-атаки впроваджують код, який негайно відбивається назад користувачеві. Щоб запобігти XSS-атакам, розробники повинні дезінфікувати користувацьке введення і кодувати виведення, щоб запобігти впровадженню скриптів. Крім того, використання Content Security Policy і файлів cookie тільки для HTTP може ще більше знизити ризик XSS.

Підробка міжсайтових запитів (CSRF) – це тип атаки, під час якої зловмисник примушує жертву виконати дію у WEB-додатку без її відома або згоди [4]. Атаки CSRF можуть використовуватися для виконання несанкціонованих дій, таких як зміна пароля жертви або здійснення несанкціонованих покупок. Наслідки успішної атаки CSRF можуть варіюватися від незначних незручностей до серйозних фінансових втрат. Щоб запобігти CSRF, розробники повинні впровадити токени CSRF і управління сеансами, щоб гарантувати, що запити можуть виконуватися тільки авторизованими користувачами.

Зламана автентифікація та управління сесіями – це порушена перевірка автентичності та управління сеансом виникають, коли додаток не виконує належним чином перевірку автентичності користувачів або управління їхніми сеансами. Це може дати можливість зловмиснику обійти аутентифікацію або перехопити сеанс користувача, надавши йому доступ до конфіденційної інформації або несанкціонованих дій. Наслідки успішної атаки можуть варіюватися від незначних незручностей до серйозних фінансових втрат або репутаційного збитку. Щоб запобігти порушенню автентифікації та управління сеансами, розробники мають використовувати безпечні методи автентифікації, такі як багатофакторна автентифікація, і забезпечувати безпечне зберігання та належне анулювання токенів сеансу [5].

Небезпечні прямі посилання на об’єкти – це вид небезпеки, коли зловмисник може отримати доступ до конфіденційних даних або функцій безпосередньо, маніпулюючи параметрами програми. Це може дати можливість зловмиснику переглядати, змінювати або видаляти конфіденційні дані без належної авторизації. Наслідки успішної атаки можуть варіюватися від незначних незручностей до серйозних фінансових втрат або репутаційного збитку. Щоб запобігти небезпечним прямим посиланням на об'єкти, розробники повинні використовувати непрямі посилання на об’єкти та належним чином перевіряти користувацьке введення, щоб гарантувати, що тільки авторизовані користувачі можуть отримати доступ до конфіденційних даних або функцій [6].

Автоматичний пошук вразливостей може допомогти швидко й ефективно ідентифікувати всі ці вразливості, знижуючи ризик успішної атаки. Використовуючи автоматизовані інструменти, розробники можуть виявляти вразливості та вживати заходів щодо їх усунення до того, як їх можна буде використовувати.

Огляд наявних методів захисту WEB-додатків від вразливостей. Тестування вразливостей – важливий процес забезпечення безпеки WEB-додатків. Він включає в себе виявлення вразливостей у WEB-додатку, які можуть бути використані зловмисниками. Існує два основні методи тестування вразливостей: ручний і автоматичний. Ручне тестування потребує втручання людини, у той час як автоматизоване тестування використовує програмні засоби для сканування WEB-додатків на наявність вразливостей.

Існують також і різні методи захисту WEB-додатків, деякі з яких є ручними, а інші – автоматизованими. Ручні методи можуть бути ефективними, але вони забирають багато часу і потребують команди фахівців з безпеки. З іншого боку, автоматизовані методи швидші та ефективніші, але вони можуть бути не такими точними, як ручні. Поєднання ручних і автоматичних методів може бути найкращим підходом до забезпечення безпеки WEB-застосунків. Розробка автоматизованого пошуку вразливостей для WEB-додатків може стати перспективним підходом до підвищення ефективності автоматизованих методів.

Однією з суттєвих переваг ручного тестування вразливостей є те, що це ретельний процес, який дає точний результат. Тестувальники можуть використовувати свій досвід, креативність та інтуїцію для виявлення потенційних вразливостей, які можуть пропустити автоматизовані інструменти. Ручне тестування також дає змогу тестувальникам імітувати реальні атаки і розуміти, як зловмисники можуть використовувати

вразливості. Крім того, ручне тестування може виявити складні вразливості, які потребують глибокого розуміння архітектури застосунку і сукупності використовуваних технологій.

Ручне тестування вразливостей – це трудомісткий процес, для якого потрібна команда досвідчених тестувальників. Він також не захищений від людських помилок, і якщо тестувальники пропустять уразливість, це може призвести до серйозного порушення безпеки. Крім того, ручне тестування не масштабується, що ускладнює тестування великих WEB-додатків зі значною кількістю сторінок.

У свою чергу автоматичне тестування вразливостей включає використання програмних інструментів для виявлення потенційних вразливостей у WEB-додатку [7]. Автоматизовані інструменти використовують різні методи, як-от сканування Web-сторінок, перевірка полів введення та аналіз відповідей сервера для виявлення вразливостей. Автоматичне тестування швидше й ефективніше, ніж ручне, і дає змогу просканувати тисячі WEB-сторінок за лічені хвилини. І це є однією з суттєвих переваг автоматизованого тестування вразливостей. Крім того, автоматизоване тестування є більш точним, ніж ручне, оскільки воно не схильне до людських помилок. Автоматизоване тестування також є масштабованим, що дає змогу тестувати великі WEB-додатки зі складною архітектурою.

Автоматичне тестування вразливостей має певні обмеження. Автоматизовані інструменти можуть пропустити деякі вразливості, для виявлення яких потрібне втручання людини. Крім того, автоматизоване тестування не може імітувати реальні атаки, що ускладнює розуміння того, як зловмисники можуть використовувати вразливості. Автоматизоване тестування також обмежене вразливостями, які може виявити інструмент, що ускладнює виявлення складних вразливостей, які потребують глибокого розуміння архітектури застосунку і сукупність взаємозв'язаних технологій. Як ручне, так і автоматизоване тестування вразливостей мають свої переваги та недоліки.

Враховуючи вищезазначену інформацію, можна стверджувати, що поєднання ручного та автоматизованого тестування є найкращим підходом до комплексного тестування вразливостей у WEB-додатках. У таблиці 2 представлено результати порівняльного аналізу ручного й автоматичного тестування, також прописано, для чого кожне підходить і наведено приклади.

Таблиця 2

Порівняння ручного й автоматичного режимів тестування WEB-додатків на вразливості

Критерій	**Ручне тестування**	**Автоматичне тестування**
Точність виявлення вразливостей	Висока точність при правильному налаштуванні та досвіді тестувальника. Можна виявити складні, контекстуальні проблеми, які важко автоматизувати	Середня точність залежно від інструмента. Автоматичні сканери можуть пропустити складні вразливості або видати хибні спрацьовування (false positives)
Час тестування	Потребує більше часу, особливо для великих додатків. Не підходить для частої перевірки в масштабних проєктах	Швидке тестування, особливо на великих проєктах або з регулярним оновленням, підходить для частих сканувань
Складність виявлення специфічних вразливостей	Може ефективно виявляти логічні помилки, проблеми із бізнес-логікою і контекстуальні вразливості, які автоматичні сканери пропускають	Може мати обмеження на складні або нестандартні вразливості. Автоматизовані інструменти не завжди здатні аналізувати бізнес-логіку
Вартість	Висока, оскільки потребує залучення кваліфікованих тестувальників	Низька після первинної інвестиції у придбання та налаштування інструментів. Потребує мінімального втручання після налаштування
Оновлення баз вразливостей	Залежить від знань та досвіду тестувальника, тому потребує постійного оновлення кваліфікації	Більшість інструментів оновлюється автоматично, що дає змогу виявляти нові типи вразливостей
Можливість інтеграції в CI/CD	Обмежена, бо ручне тестування не підходить для повної автоматизації у DevOps середовищах	Легко інтегрується в CI/CD пайплайни для регулярного тестування при кожному оновленні коду
Людський фактор	Можливі помилки через людський фактор або недостатню уважність	Мінімальний, оскільки процес автоматизований, хоча помилки можуть бути через налаштування або обмеження інструменту
Підходить для	Поглиблене тестування, фінальний аудит безпеки, ручне дослідження специфічних або нових вразливостей	Постійний моніторинг, регресійне тестування, швидке тестування великих обсягів коду
Приклади інструментів	Burp Suite, ручне тестування з використанням чек-листів	OWASP ZAP, Acunetix, Nessus, Netsparker

Інструменти автоматизованого тестування вразливостей використовуються для автоматизації процесу виявлення вразливостей і можуть використовуватися для виявлення широкого кола проблем, включно з SQL-ін'єкціями, міжсайтовими сценаріями та іншими вразливостями [8]. Було проведено кілька досліджень з автоматизованого тестування вразливостей для WEB-додатків. Одне з таких досліджень було проведено дослідниками з Каліфорнійського університету в Санта-Барбарі, які розробили інструмент під назвою "Peach Fuzz" для автоматизованого тестування вразливостей. Інструмент був розроблений для виявлення вразливостей у WEB-додатках шляхом створення набору тестових випадків, які можна використовувати для виявлення слабких місць у додатку. Дослідники виявили, що Peach Fuzz зміг виявити кілька вразливостей у WEB-додатках, включно з SQL-ін'єкціями та міжсайтовими сценаріями [9].

Інше дослідження провели дослідники з Університету Меріленда, які розробили інструмент під назвою "WebSSARI" для автоматизованого тестування вразливостей. Інструмент був розроблений для визначення вразливостей у WEB-додатках шляхом аналізу коду та виявлення потенційних вразливостей. Дослідники виявили, що WebSSARI зміг виявити кілька вразливостей у WEB-додатках, включно з SQL-ін'єкціями та міжсайтовими сценаріями [10].

Дослідники з Вашингтонського університету, розробили інструмент під назвою "AppSealer" для автоматизованого тестування вразливостей. Інструмент був розроблений для знаходження вразливостей у мобільних додатках шляхом аналізу коду та виявлення потенційних вразливостей. Дослідники показали, що AppSealer зміг виявити кілька вразливостей у мобільних додатках, зокрема небезпечне зберігання даних, небезпечний зв'язок і небезпечну авторизацію [11].

Ще одне дослідження, провели вчені з Техаського університету в Остіні. Вони розробили інструмент під назвою "Sage" для автоматизованого тестування вразливостей. Інструмент був розроблений для виявлення вразливостей у WEB-додатках шляхом створення набору тестових випадків, які можна використовувати для знаходження слабких місць у додатку. Дослідники показали, що Sage змогла виявити кілька вразливостей у WEB-додатках , включно з SQL-ін'єкцією, міжсайтовим скриптингом та іншими вразливостями [12].

Загалом, автоматизований інструмент пошуку вразливостей повинен надавати комплексне і надійне рішення для виявлення і зниження ризиків безпеки у WEB-додатках.

Вибір оптимального методу розроблення автоматизованого інструменту пошуку вразливостей для WEB-додатків залежить від конкретних потреб і цілей проєкту. Однак комбінація статичного і динамічного аналізу коду часто ефективна при виявленні вразливостей. Крім того, машинне навчання можна використовувати для підвищення точності виявлення вразливостей [13]. Тому доцільно було провести порівняльний аналіз методів виявлення вразливостей WEB-додатків, який представлено у таблиці 3.

Таблиця 3

Порівняльна таблиця методів виявлення вразливостей WEB-додатків

Алгоритм / Метод	**Опис**	**Приклади вразливостей, що виявляються**
Сканування сигнатур	Використовує базу відомих сигнатур вразливостей для порівняння з кодом або запитами WEB-додатка. Ефективний для швидкого виявлення відомих вразливостей	SQL-ін'єкції, міжсайтовий скриптинг (XSS), вразливі компоненти
Аналіз поведінки	Аналізує поведінку додатка під час виконання для виявлення нетипових дій або аномалій	Витоки даних, підозрілі операції, зміна прав доступу
Fuzz-тестування	Введення випадкових або помилкових даних у додаток для провокування помилок або некоректної поведінки	SQL-ін'єкції, XSS, буферні переповнення
Статичний аналіз коду (SAST)	Аналіз вихідного коду без виконання додатка для виявлення вразливостей на ранніх етапах розробки	Недостатній контроль доступу, вбудовані паролі, XSS
Динамічний аналіз коду (DAST)	Аналіз додатка під час його виконання. Імітує атаки з метою виявлення вразливостей	SQL-ін'єкції, CSRF, XSS, витік конфіденційних даних
Аналіз потоків даних	Перевіряє, як дані переміщуються між різними компонентами додатка для виявлення небезпечних шляхів	SQL-ін'єкції, витоки конфіденційних даних
Метод "чорної скриньки"	Тестування без доступу до вихідного коду. Виявляє вразливості шляхом імітації атак із зовнішнього середовища	SQL-ін'єкції, XSS, відкриті порти, помилки авторизації
Метод "білої скриньки"	Тестування з повним доступом до вихідного коду та інфраструктури додатка	Логічні помилки, недостатній контроль доступу, бізнес-логіка
Аналіз за допомогою машинного навчання	Використання алгоритмів машинного навчання для виявлення нових і нетипових вразливостей на основі попередніх атак	Аномальні запити, DDoS-атаки, нові форми ін'єкцій

Аналіз на основі шаблонів (Pattern Matching)	Виявлення вразливостей шляхом пошуку шаблонів відомих атак або вразливостей у коді або мережевому трафіку	SQL-ін'єкції, XSS, підробка запитів між сайтами (CSRF)

Для розробки інструментів автоматизованого пошуку вразливостей для WEB-додатків потрібна відповідна системна архітектура і компоненти. Нижче наведено рекомендовану системну архітектуру та компоненти для проєктування системи автоматизованого пошуку вразливостей WEB-додатків:

– зовнішній інтерфейс: компонент зовнішнього інтерфейсу відповідатиме за надання користувачам користувацького інтерфейсу. Він надасть користувачам інтерфейс для введення інформації про WEB-додаток. Інтерфейс зовнішнього інтерфейсу має бути простим для навігації та розуміння, а також повинен мати можливість відображати результати сканування вразливостей у чіткій і лаконічній формі;

– бекенд-сервер відповідатиме за зберігання відсканованих даних WEB-додатків, запуск сканерів вразливостей і створення звітів. Бекенд-сервер також повинен мати можливість керувати робочим навантаженням процесу сканування вразливостей;

– сканери вразливостей скануватимуть WEB-додаток на наявність вразливостей. Сканери можуть бути як сторонніми, так і виготовленими за індивідуальним замовленням. Рекомендується використовувати кілька сканерів для підвищення точності результатів сканування;

– база даних буде потрібна для зберігання даних WEB-додатку, результатів сканування вразливостей та іншої відповідної інформації;

– звітність, компонент звітності відповідатиме за створення звітів про сканування вразливостей. Звіти мають бути чіткими та зрозумілими для користувачів;

– безпека має бути ключовим фактором під час розроблення архітектури системи для інструментів автоматичного пошуку вразливостей. Система має бути розроблена з урахуванням вимог безпеки, щоб запобігти несанкціонованому доступу до даних WEB-додатків і результатів сканування вразливостей;

– інтеграція. Архітектура системи має бути розроблена для інтеграції з іншими інструментами та системами безпеки, такими як SIEM, щоб забезпечити комплексне рішення для забезпечення безпеки.

Загалом, системна архітектура і компоненти мають бути розроблені таким чином, щоб забезпечити надійний і всебічний автоматизований засіб пошуку вразливостей для WEB-додатків.

Для перехоплення мережевого трафіку зазвичай використовують мережеві карти, переведені в режим прослуховування. Прослуховування мережі Інтернет потребує підключення комп’ютера із запущеним перехоплювачем до сегмента мережі, після чого хакеру стає доступним увесь мережевий трафік, який надсилають і отримують комп'ютери в цьому мережевому сегменті. Ще простіше виконати перехоплення трафіку радіомереж, що використовують бездротові мережеві посередники, у цьому разі не потрібно навіть шукати місце для підключення до кабелю. Або ж зловмисник може під’єднатися до телефонної лінії, що зв’язує комп’ютер із сервером Інтернету, знайшовши для цього зручне місце. При проходженні авторизації на сайті без шифрування особистих даних, зловмисник при успішному підключенні до мережі підприємства перехоплює особисті дані користувача.

Мережева модель OSI (systems interconnection basic reference model) – це модель взаємодії мережевих протоколів. А протоколи, своєю чергою, це стандарти, які визначають, яким чином обмінюватимуться даними різні програми.

Проєктована програма повинна вміти аналізувати пакети інформації та знайшовши в ньому пароль користувача, виводити повідомлення про загрозу і додавати сайт до списку небезпечних сайтів. Приклад зображення пакета інформації з відкритими даними в базі даних зображено на рисунку 1.

Рисунок 1 – Відображення пакета з даними користувача в базі даних

Також необхідно передбачити функціональну можливість проєктованої системи виводити звіт її роботи у формі таблиці, яка представлена на рисунку 2.

IP адреса	Дата виявлення	Тип вразливості	Підсумок
00.000.00.00	14.10. 2024	Передача даних під час авторизації у відкритому вигляді	Сайт небезпечний, потенційна загроза викрадення даних
…	…	…	…

Рисунок 2 – Звіт за результатами роботи програми

Використання як безпечних, так і небезпечних версій призначеного для користувача WEB-додатка може виявити помилкові спрацьовування та хибні негативні результати сканерів вразливостей. Це може бути пов'язано із методами, які вони використовують. Кореляція між використовуваними методами та виникненням хибних позитивних або хибних негативних результатів може допомогти у визначенні способів покращення методів сканування WEB-додатків.

Враховуючи результати інформаційного аналізу найбільш поширених вразливостей WEB-додатків, порівняльного аналізу методів та технологій виявлення вразливостей було розроблено конкретні алгоритми для виявлення певних вразливостей, таких як SQL-ін'єкція та XSS. Алгоритм проведення автоматизованого пошуку ін'єкцій SQL представлений на рисунку 3.

Для пошуку вразливостей у WEB-додатках, насамперед ініціалізуємо SQL-символи в масиві. Після цього необхідно створити два списки для зберігання повідомлень про помилки SQL:

– один для зберігання повідомлень про помилки конкретної бази даних, таких як повідомлення про помилки SQL тощо;

– інший для зберігання загальних повідомлень про помилки бази даних.

Далі визначаємо значення помилок. Після цього ініціалізуємо метод сканера – сканер приймає повідомлення типу «http», як введення від шукача. HTTP-повідомлення містить відомості про кожен запит або URL-адресу зі списком параметрів.

Для кожного параметра в повідомленні «http»:

– введемо SQL-символи з масиву;

– перевіримо відповідь, щоб дізнатися чи відповідає вона повідомленням про помилки з двох карт або списків;

– перевіряємо чи відбувається збіг «Flag» та SQL-вразливості;

– у іншому випадку повторюємо дії допоки не буде досягнуто кінця списку параметрів.

Розглянемо розроблений на рисунку 4 алгоритм для XSS. Для кожної URL-адреси в списку відвідуваних URL-адрес:

– визначаємо всі параметри;

– вводимо параметри в список параметрів;

– для кожного параметра в черзі параметрів;

– поставимо сценарій або тестовий приклад XSS як вхідний параметр і передбачаємо передачу запиту;

– перевіримо відповідь, щоб визначити чи наданий сценарій або тестовий випадок;

– повідомляємо про цю вразливість, якщо у відповіді є скрипт.

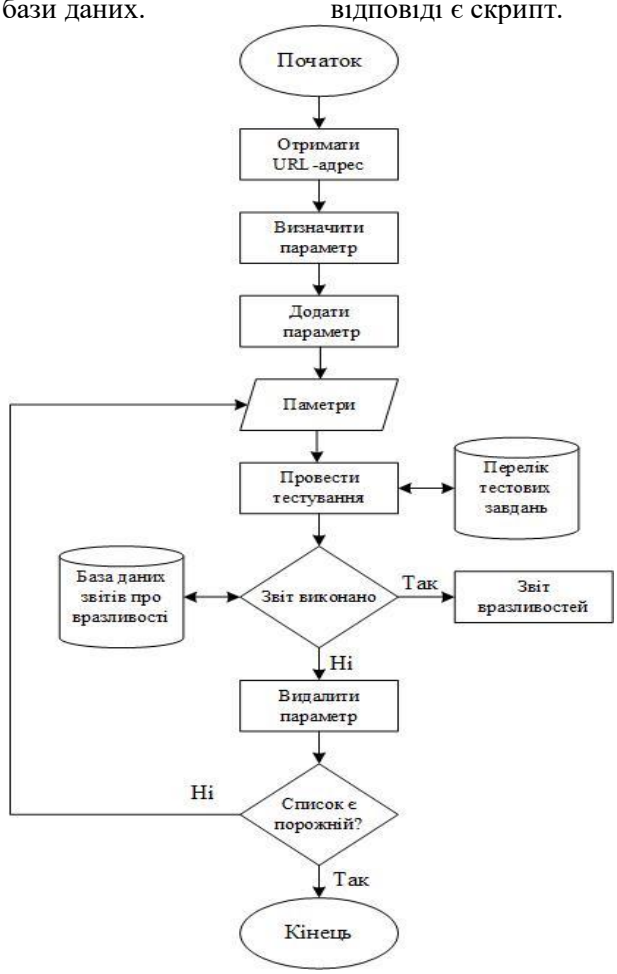

Рисунок 3 – Алгоритм автоматизованого пошуку SQL-ін'єкцій у проєктованій системі

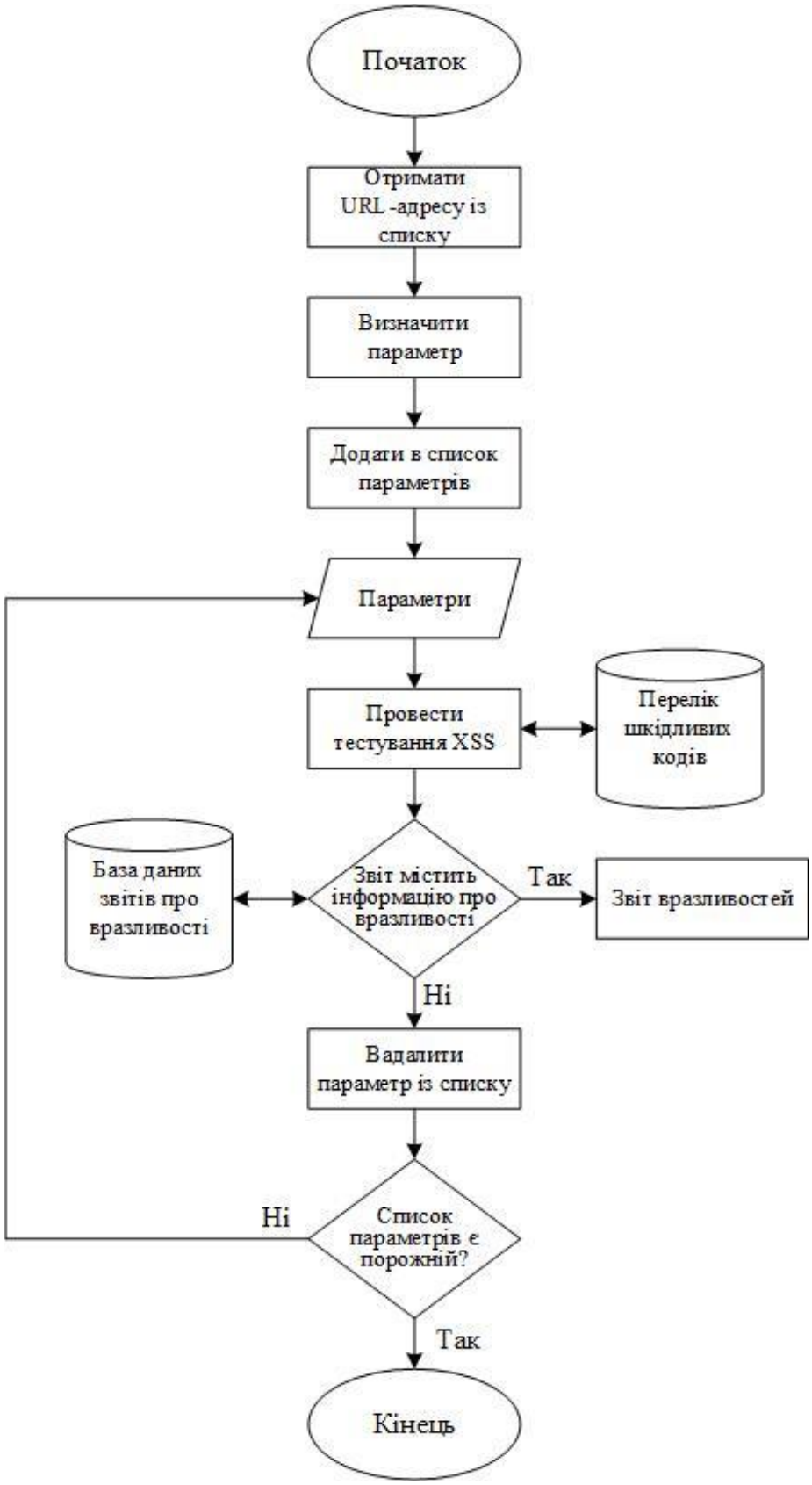

Рисунок 4 –Алгоритм автоматизованого пошуку XSS у проєктованій системі

Розробивши алгоритми роботи проєктованої системи, можна приступити уже до розроблення сканеру вразливостей WEB-додатків. Для цього можна використати мову програмування Python, середовище програмування Pycharm, WEB-сервер Apache, SQL-сервер MySQL (або щось інше залежно від компетенції розробників). Концептуальна модель роботи автоматизованої системи пошуку вразливостей WEB-додатків представлена на рисунку 5.

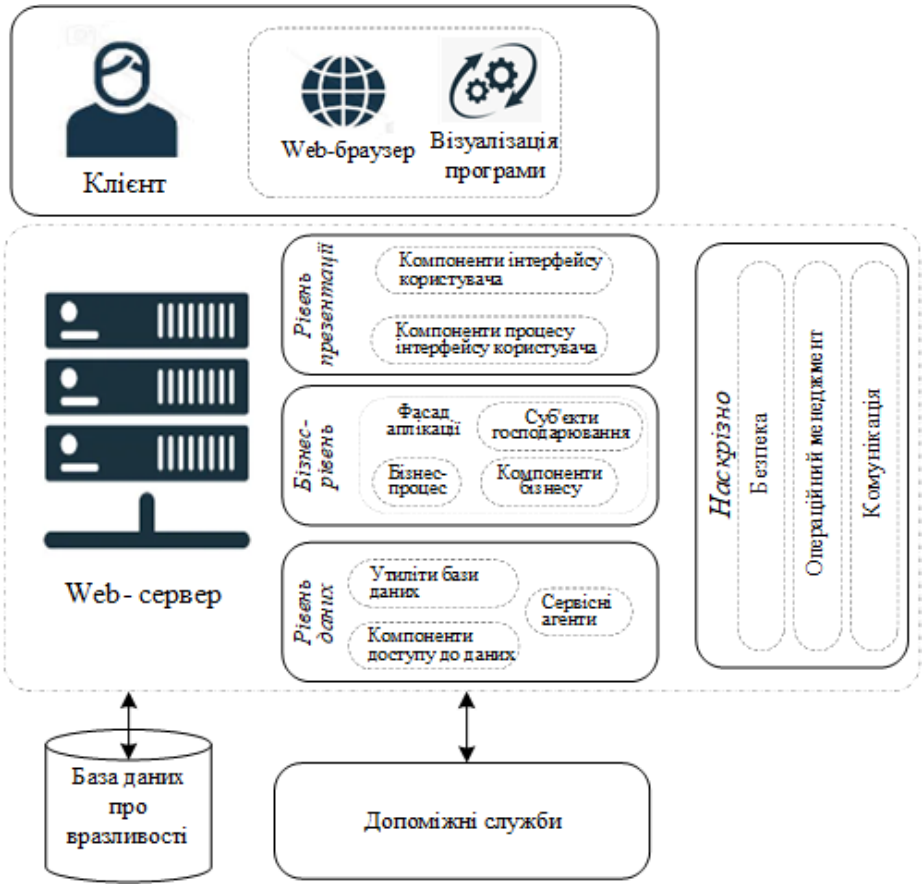

Рисунок 5 – Концептуальна модель автоматизованої програми для перевірки WEB-додатків на вразливість

Опис схеми процесу перевірки:

– користувач: використовує клієнтську програму, щоб перевірити наявність вразливостей на WEB-сайті, зв'язавши сайт із тестовим сайтом;

– клієнтська програма під'єднується до сервера, завантажує відповідну WEB-сторінку, аналізує, щоб отримати всі пов'язані посилання. Після цього вона виконує відповідну атаку на WEB-сайті й отримує результат. Знову аналізує, чи містить WEB-сайт дефект і повідомляє користувача;

– серверна програма: отримує з'єднання з клієнтом, відповідає певним вимогам. З відповідними налаштуваннями для тестування на сервері, який містить будь-який WEB-сайт, що може містити недоліки. Згідно з аналізом інтерактивної моделі, ми моделюємо продуктивність програми на основі цих взаємодій.

Механізм сканування встановлює з'єднання із сервером, використовуючи посилання, надане користувачем через параметр. Потім він аналізує вхідні параметри, щоб ідентифікувати відповідний механізм сканування або зловмисника. За відсутності будь-яких параметрів рушій виконує сканування всіх компонентів. На сервері WEB-додатків можуть бути вразливості в мережі, базі даних або інші пов'язані вразливості, які роблять його сприйнятливим до проблем перевірки вхідних даних. Якщо сканер здатний виявити такі вразливості та видати попередження, то програмісту необхідно обов'язково перевірити безпеку на WEB-сайті.

Програма може бути вбудована в різні модулі, виконуючи конкретні завдання:

– нижня частина – це модулі мови програмування Python з відповідними бібліотеками;

– модуль Core містить дрібніші модулі, які виконують різні завдання: модуль збирає посилання на WEB-сайт, перетворює їх на різні формати відповідно до звичайної структури WEB-сайту, як-от: форма, текст, посилання, текстове поле, модулі також надають інструменти для ведення журналу, параметри аналізу та деякі інші допоміжні функції;

– модуль містить атаки для виконання перевірок на сайті, включно з XSS, CSRF, Breach, Clickjack.

Розробка інструменту автоматизованого пошуку вразливостей для WEB-додатків потребує ретельного тестування для забезпечення його працездатності. Можливі такі методи тестування і перевірки працездатності інструменту:

– функціональне тестування, яке включає в себе тестування функцій інструменту, щоб переконатися, що він відповідає вимогам. Інструмент слід протестувати, щоб переконатися,

що він може автоматизувати пошук вразливостей у WEB-додатках;

– естування продуктивності. Сюди входить тестування продуктивності інструменту для перевірки його швидкості та ефективності. Інструмент слід протестувати, щоб переконатися, що він може швидко й ефективно сканувати WEB-додатки;

– тестування сумісності. Сюди входить перевірка сумісності інструмента з різними WEB-додатками та операційними системами. Інструмент слід протестувати, щоб переконатися, що він може працювати з різними типами WEB-додатків та операційних систем;

– тестування безпеки. Це передбачає тестування безпеки інструменту, щоб переконатися, що він не містить вразливостей, які можуть бути використані зловмисниками. Інструмент слід протестувати, щоб переконатися, що він не створює нових вразливостей у WEB-додатках, які скануються;

– юзабіліті-тестування. Сюди входить перевірка зручності використання інструменту, щоб переконатися, що він простий у використанні і не потребує великих технічних знань. Інструмент слід протестувати, щоб переконатися, що він зручний для користувача і може використовуватися нетехнічними користувачами.

Висновки. Провівши інформаційний аналіз предметної області безпечного користування WEB-додатками, було визначено проблематику нашого дослідження, яка полягала у розробленні методів та робочого інструменту для автоматизованого пошуку вразливостей у WEB-додатках

За допомогою використання порівняльного аналізу було визначено переваги та недоліки ручного й автоматичного методів і середовищ тестування вразливостей, та обгрунтовано вибір на користь автоматизованого тестування вразливостей. Ця інформація забезпечила основу для розуміння важливості автоматичного тестування вразливостей у безпеці WEB-додатків. Порівняльний аналіз алгоритмів та методів виявлення вразливостей підкреслив необхідність розробки ефективних, дієвих інструментів і методів тестування на вразливості для забезпечення безпеки WEB-додатків.

Розробка автоматизованого інструменту пошуку вразливостей для WEB-додатків має вирішальне значення для забезпечення безпеки online-систем. Тому розроблено концептуальну модель роботи автоматизованої програми, що забезпечує ефективне та дієве сканування WEB-додатків на наявність вразливостей. Розробка модулів і функціональних можливостей цієї програми створює комплексний інструмент, який охоплює широкий спектр потенційних вразливостей. Розроблені базові алгоритми для програми забезпечують міцну основу для виявлення вразливостей. Запропоновані методи тестування та перевірка функціональності інструменту гарантують, що він працюватиме належним чином і даватиме надійні результати. Загалом розробка автоматизованого інструменту пошуку вразливостей є важливим кроком на шляху підвищення безпеки WEB-додатків.

Список літератури:

1. Open Web Application Security Project [Електронний ресурс]. Дата доступу: 10.11.2024. URL: https://owasp.org/

2. Satapathy, S. C., Govardhan, A., Raju, K. S., Mandal, J. K. Виявлення та виправлення ін’єкцій SQL за допомогою методів машинного навчання // Advanced Intelligent Systems Computing. 2015. Вип. 337. С. 435–442.
DOI: 10.1007/978-3-319-11191-9_41.

3. Рагвендра Пратап Сінгх, Б. Р. Чандаваркар. Генерація динамічної політики безпеки вмісту на стороні клієнта для пом’якшення атак XSS // Матеріали 15-ї Міжнародної конференції з комп’ютерних комунікаційних та мережевих технологій (ICCCNT). 2024. С. 1–7.

4. Мартінс, Н., Круз, Дж. М., Круз, Т., Абреу, П. Х. Змагальне машинне навчання, застосоване до сценаріїв вторгнення та зловмисного програмного забезпечення: систематичний огляд // IEEE Access. 2020. Т. 8. С. 35403–35419. DOI: 10.1109/ACCESS.2020.2975204.

5. Цю С., Лю К., Чжоу С., Ву К. Огляд технологій протиборчої атаки та захисту штучного інтелекту // Applied Sciences. 2019. Т. 9. С. 909. DOI: 10.3390/app9050909.

6. Kim, E., Choi, H.-K. Security analysis and bypass user authentication bound to device of Windows Hello in the wild // Security and Communication Networks. 2021. С. 1–13. DOI: 10.1155/2021/6245306.

7. Salahdine, F., Kaabouch, N. Social engineering attacks: A survey // Future Internet. 2019. Т. 11. № 4. С. 89. DOI: 10.3390/fi11040089.

8. Dasmohapatra, S., Priyadarshini, S. B. B. A comprehensive study on SQL injection attacks, their mode, detection and prevention // Lecture Notes in Networks and Systems. 2021. С. 617–632. DOI: 10.1007/978-981-16-3346-1_50.

9. Son, S., McKinley, K. S., Shmatikov, V. Diglossia: detecting code injection attacks with precision and efficiency // Proceedings of ACM Conference on Computer and Communications Security. 2013. Т. 2. С. 1181–1191. DOI: 10.1145/2508859.2516696.

10. Jeffrey, S., Foster, M., Hicks, W., Pugh, W. Improving Software Quality with Static Analysis

[Електронний ресурс]. URL: cs.umd.edu/~mwh/papers/paste41gp-foster.pdf.

11. Klein, A. Cross Site Scripting Explained [Електронний ресурс]. URL: pages.cs.wisc.edu/~rist/642-fall-2011/CSS.pdf.

12. Top 10-2025 A05-Security Misconfiguration [Електронний ресурс]. Дата доступу: 2024. URL: https://owasp.org/www-project-top-ten/.

13. Цигой П., Степанчич Ж., Блажіч Б. Й. Широкомасштабний аналіз WEB-вразливості системи безпеки: висновки, проблеми та способи усунення // Gervasi O., et al. Обчислювальна наука та її застосування – ICCSA 2020. ICCSA 2020. Том 12253. Springer, Cham, 2020. DOI: 10.1007/978-3-030-58814-4_64.

14. Коваленко. І. В., Михайленко С. О. Моделі оцінки ефективності систем автоматизованого пошуку вразливостей у веб-додатках // Кібербезпека та захист інформації, 2023, № 2, С. 23–36.

15. Сушко А.В. І., Кравчук В. О. Підходи до побудови систем автоматизованого аналізу безпеки веб-додатків // Інформаційні технології управління, 2023, № 3, С. 15–27. 21.

16. Кулик О.В. А., Марченко І. Застосування гібридних методів аналізу для визначення вразливостей у веб-додатках // Захист інформації та кібербезпека, 2024, № 2, С. 34–49. 30.

17. Кравчук О.В. М., Білоус Н. С. Розробка тестових середовищ для автоматизованого пошуку вразливостей у веб-додатках // Системи забезпечення інформаційної безпеки, 2024, № 1, С. 14–28. 39.

18. Куценко І. Г., Поліщук В. С. Адаптивні методи тестування веб-додатків для виявлення вразливостей Zero-Day // Інформаційна безпека та інноваційні технології. – 2023. – № 2. – С. 14–27.

19. Савченко П. В., Литвин Т. В. Моделювання виявлення та запобігання вразливостям у веб-додатках із використанням методів аномального виявлення // Захист інформаційних систем. – 2024. – № 3. – С. 31–44.

20. Гончарук О. В., Іващенко Л. М. Автоматизовані методи пошуку логічних помилок у бізнес-логіці веб-додатків // Технології інформаційного аудиту. – 2023. – № 4. – С. 25–38.

21. Лисенко А. В., Шевченко Т. д. Використання підходів атак на основі моделей зловмісника для аналізу безпеки веб-додатків // Інформаційна безпека в кіберпросторі. – 2023. – № 6. – С. 39–52.

22. Романюк І. С., Гаврилюк В. Л. Системи аналізу безпеки веб-додатків із застосуванням технологій блокчейн // Новітні підходи до кібербезпеки. 2024. № 1. С. 28–42.

23. Шульга О. К., Іванченко Н. Г. Застосування методу рішення дерев для оцінки критичності вразливостей у веб-додатках // Аналіз даних та інформаційна безпека. 2023. № 5. С. 48-61.

24. Литвиненко М. О., Ткаченко О. І. Автоматизовані платформи для багатофак-торного аналізу вразливостей веб-додатків // Системи кіберзахисту, 2023, № 3, С. 33–46.

25. Білий В. І., Остапчук П. А. Використання розподілених обчислень для автоматизованого тестування масштабованих веб-додатків // Хмарні технології та безпека. 2024. № 2. С. 14–29.

26. Жуков С. П., Марченко Д. М. Інтеграція механізмів захисту у фреймворки розробки веб-додатків для мінімізації вразливостей // Програмна інженерія та кібербезпека. 2023. № 4. С. 22–36.

References:

1. Open Web Application Security Project [Electronic resource]. Accessed: 10.11.2024. Available at: https://owasp.org/

2. Satapathy, S. C., Govardhan, A., Raju, K. S., Mandal, J. K. Detection and remediation of SQL injections using machine learning methods // Advanced Intelligent Systems Computing. 2015. Vol. 337. P. 435–442. DOI: 10.1007/978-3-319-11191-9_41.

3. Raghvendra Pratap Singh, B. R. Chandavarkar. Client-side dynamic content security policy generation to mitigate XSS attacks // Proceedings of the 15th International Conference on Computer Communication and Networking Technologies (ICCCNT). 2024. P. 1–7.

4. Martins, N., Cruz, J. M., Cruz, T., Abreu, P. H. Adversarial machine learning applied to intrusion and malware scenarios: a systematic review // IEEE Access. 2020. Vol. 8. P. 35403–35419. DOI: 10.1109/ACCESS.2020.2975204.

5. Qiu, S., Liu, K., Zhou, S., Wu, C. Overview of adversarial attack and defense technologies in artificial intelligence // Applied Sciences. 2019. Vol. 9. P. 909. DOI: 10.3390/app9050909.

6. Kim, E., Choi, H.-K. Security analysis and bypass user authentication bound to device of Windows Hello in the wild // Security and Communication Networks. 2021. P. 1–13. DOI: 10.1155/2021/6245306.

7. Salahdine, F., Kaabouch, N. Social engineering attacks: A survey // Future Internet. 2019. Vol. 11. No. 4. P. 89. DOI: 10.3390/fi11040089.

8. Dasmohapatra, S., Priyadarshini, S. B. B. A comprehensive study on SQL injection attacks, their mode, detection and prevention // Lecture Notes in Networks and Systems. 2021. P. 617–632. DOI: 10.1007/978-981-16-3346-1_50.

9. Son, S., McKinley, K. S., Shmatikov, V. Diglossia: detecting code injection attacks with precision and efficiency // Proceedings of ACM Conference on Computer and Communications Security. 2013. Vol. 2. P. 1181–1191. DOI: 10.1145/2508859.2516696.

10.Jeffrey, S., Foster, M., Hicks, W., Pugh, W. Improving Software Quality with Static Analysis [Electronic resource]. Available at: cs.umd.edu/~mwh/papers/paste41gp-foster.pdf.

11.Klein, A. Cross Site Scripting Explained [Electronic resource]. Available at: pages.cs.wisc.edu/~rist/642-fall-2011/CSS.pdf.

12.Top 10-2025 A05-Security Misconfiguration [Electronic resource]. Accessed: 2024. Available at: https://owasp.org/www-project-top-ten/.

13.Tsigoi, P., Stepanchich, Z., Blazic, B. Y. Large-scale web vulnerability analysis of security systems: findings, issues, and remedies // Gervasi, O., et al. Computational Science and Its Applications – ICCSA 2020. ICCSA 2020. Vol. 12253. Springer, Cham, 2020. DOI: 10.1007/978-3-030-58814-4_64.

14. Kovalenko. I. V., Mykhailenko S. O. Models for assessing the effectiveness of automated vulnerability search systems in web applications // Cybersecurity and Information Protection, 2023, № 2, P. 23–36.

15. Sushko A.V. I., Kravchuk V. O. Approaches to building automated security analysis systems for web applications // Information Management Technologies, 2023, № 3, P. 15–27.

16. Kulyk O.V. A., Marchenko I. Application of hybrid analysis methods to identify vulnerabilities in web applications // Information Protection and Cybersecurity, 2024, №2, P. 34 –49.

17. Kravchuk O.V. M., Bilous N. S. Development of test environments for automated search for vulnerabilities in web applications // Information Security Systems, 2024, No. 1, P. 14 –28. 39.

18. Kutsenko I. G., Polishchuk V. S. Adaptive methods of testing web applications to detect Zero-Day vulnerabilities // Information Security and Innovative Technologies. – 2023. – №2/ P. 14 – 27.

19. Savchenko P. V., Lytvyn T. V. Modeling the detection and prevention of vulnerabilities in web applications using anomalous detection methods // Information Systems Protection. – 2024. – №3. – P. 31–44.

20. Goncharuk O. V., Ivashchenko L. M. Automated methods for searching for logical errors in the business logic of web applications // Information Audit Technologies. – 2023. – № 4. P. 25–38.

21. Lysenko A. V., Shevchenko T. d. Using attack approaches based on attacker models for analyzing the security of web applications // Information Security in Cyberspace. – 2023. – № 6. P. 39–52.

22. Romanyuk I. S., Gavrilyuk V. L. Web application security analysis systems using blockchain technologies // New approaches to cybersecurity. 2024. № 1. P. 28–42.

23. Shulga O. K., Ivanchenko N. G. Applying the decision tree method to assess the criticality of vulnerabilities in web applications // Data Analysis and Information Security. 2023. №5. P. 48-61.

24. Lytvynenko M. O., Tkachenko O. I. Automated platforms for multifactor analysis of web application vulnerabilities // Cybersecurity Systems, 2023, № 3, P. 33–46.

25. Bily V. I., Ostapchuk P. A. Using distributed computing for automated testing of scalable web applications // Cloud technologies and security. 2024. № 2. P. 14–29.

26.Zhukov S. P., Marchenko D. M. Integration of protection mechanisms into web application development frameworks to minimize vulnerabilities // Software engineering and cybersecurity. 2023. № 4. P. 22–36.

Оглядова стаття.
Надійшла до редакції 18.11.2024.
Прийнято до публікації 18.12.2024.