Міністерство освіти і науки України
Харківський національний університет імені В.Н. Каразіна
Факультет комп'ютерних наук
Спеціальність 125 «Кібербезпека»
Освітня програма «Кібербезпека»

«Допущено до захисту»
В.о. завідувача кафедрою БІСТ
Мелкозьорова О.М.
_____
«___» _____ 2024 р

Пояснювальна записка
до кваліфікаційної роботи бакалавра

на тему:_«Аналіз інструментальних засобів тестування вразливостей веб-додатків»

Оцінка « »
Голова ЕК
Лемешко О.В._____

Керівник: к.т.н., Колованова Є. П.

Рецензент:
PhD, Мороз О.Ю.

Виконавець: студент групи КБ41
_____Авдєєнко М. А.

Харків – 2024

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра: 55 с., 7 рис., 4 табл., 46 джерел.

Об'єкт дослідження – веб-додатки.

Предмет дослідження – методи та засоби тестування вразливостей веб-додатків.

Мета роботи – провести аналіз інструментальних засобів тестування вразливостей веб-додатків з метою використання в реальних системах.

Методи дослідження – використовуючи різні методи тестування програмного забезпечення здійснюється їх порівняння та вибір найоптимальніших, що доповнюють один одного. Здійснюється аналіз інструментальних засобів тестування вразливості веб-додатків.

У роботі проведено аналіз захищеності веб-додатків. Наведено статистику найбільш розповсюджених вразливостей зі списку OWASP.

Здійснено аналіз існуючих вразливостей та представлено узагальнені дані стосовно їх розповсюдженості.

Проаналізовано нормативне забезпечення та розглянуто міжнародні стандарти в галузі інформаційної безпеки, з урахуванням існуючих методів оцінювання та управління ризиками інформаційної системи.

Застосування комплексного рішення щодо тестування вразливостей веб-додатків можна мінімізувати ризики, пов'язані з компрометацією роботи інформаційної системи.

Галузь використання – пошук вразливостей веб-додатків.

Ключові слова: ВРАЗЛИВОСТІ, ТЕСТУВАННЯ, ІНСТРУМЕНТАЛЬНЕ ТЕСТУВАННЯ, ВРАЗЛИВОСТІ ВЕБ-ДОДАТКІВ

ABSTRACT

Explanatory note to the bachelor's qualification thesis: 55 pp., 7 figures, 4 tables, 46 sources.

The object of research is web applications.

The subject of the research is methods and means of testing the vulnerabilities of web applications.

The purpose of the work is to conduct an analysis of web application vulnerability testing tools for use in real systems.

Research methods - using various software testing methods, they are compared and the most optimal ones that complement each other are selected. An analysis of tools for testing the vulnerability of web applications is carried out.

The paper analyzes the security of web applications. Statistics of the most common vulnerabilities from the OWASP list are provided.

An analysis of existing vulnerabilities was carried out and generalized data on their prevalence was presented.

Regulatory support was analyzed and international standards in the field of information security were considered, taking into account the existing methods of assessing and managing information system risks.

Applying a comprehensive solution for testing the vulnerabilities of web applications can minimize the risks associated with compromising the operation of the information system.

The field of use is the search for vulnerabilities in web applications.

Keywords: VULNERABILITIES, TESTING, INSTRUMENTAL TESTING, WEB APPLICATIONS VULNERABILITIES

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ПЗ – програмне забезпечення

АЗ – апаратне забезпечення

ПК – персональний комп’ютер

БД – база даних

ОС – операційна система

ОП – оперативна пам’ять

СІ – сервер інтеграції

API – прикладний програмний інтерфейс

ТЗ – технічне завдання

GUI – графічний інтерфейс користувача

OWASP (Open Web Application Security Project) – це відкритий проект забезпечення безпеки WEB-додатків.

XSS (Cross – site Scripting) – міжсайтове виконання сценаріїв ДСТУ – державний стандарт України НД ТЗІ – нормативний документ системи технічного захисту інформації.

QoS (Quality of service) – набір методів для управління ресурсами пакетних мереж.

Модель OSI — абстрактна мережева модель для комунікацій і розробки мережевих протоколів.

ВСТУП

Сьогодні інтернет є звичайною складовою життя кожного з нас. За його допомогою люди збирають та аналізують необхідну інформацію, проводять час переглядаючи фільми, граючи в комп'ютерні онлайн-ігри, читаючи книги та спілкуючись один з одним.

Веб-ресурсів стає все більше. На жаль, не всі веб-додатки є безпечними для використання. Відсутність основних налаштувань безпеки є джерелами розповсюдження різноманітних загроз. Веб-сайти можуть містити шкідливі програми, віруси або ж бути вразливими до хакерських атак. Внаслідок цього може бути скомпрометована або навіть знищена важлива інформація користувачів. Тому розробникам необхідно приділяти увагу безпеці розроблюваних додатків, а користувачам пам'ятати про це та вживати всіх необхідних заходів для запобігання доступу до своїх даних.

Сьогодні з розвитком програмного забезпечення, постає також питання його надійності. Необхідний рівень функціонування програмного забезпечення можна забезпечити шляхом тестування загроз та виявленням дефектів. Ненадійне програмне забезпечення може призвести до великих втрат для компанії або окремої людини. Тому виникає потреба у розвитку, аналізі та застосування методів тестування програмного забезпечення. Для того, щоб забезпечити високу якість вихідного продукту, слід користуватися не одним методом тестування, а комплексом таких інструментів.

Рівень захищеності сучасних веб-додатків продовжує постійно зростати, однак все ще залишається на доволі низькому рівні, у порівнянні із темпами зростання загроз безпеці та цілісності даних. Зловмисники можуть проводити атаки на користувачів у 9 з 10 веб-додатків. В тому числі перенаправляти клієнтів на інший (небезпечний або підконтрольний їм) ресурс, викрадати дані за допомогою фішингових атак та інфікувати комп'ютери шкідливим програмним забезпеченням. На сьогоднішній день несанкціонований доступ до веб-додатків

можливий на 39% сайтів. Загроза втрати важливих даних існує в 68% веб-додатків.

Компанії готові платити фахівцям за дослідження та виявлення вразливостей своїх продуктів з метою їх усунення та перекриття можливих атак. Також актуальною проблемою залишається нестача кваліфікованих спеціалістів у цій галузі, тому вартість їх послуг може бути досить високою, внаслідок чого лише невелика частка компаній (зазвичай великих) здатні виділяти грошові ресурси для підтримки безпеки програмного забезпечення у належному стані.

Обґрунтування вибору теми та її актуальність: З метою подальшого забезпечення безпеки інформаційних систем, оптимізації та спрощення проведення операцій виявлення вразливостей веб-додатків було прийнято рішення проаналізувати найбільш актуальні на розповсюджені вразливості веб-додатків та можливі загрози та атаки. Розуміння та оцінка можливих ризиків є важливою складовою побудови захищеної системи, адже від захисту інформації, процесів та діяльності в кіберпросторі, можливості швидко виявити, кваліфікувати та перекрити загрози, можливі витоки інформації в кіберпросторі залежить дуже багато.

Об’єкт дослідження – веб-додатки.

Предмет дослідження – методи та засоби тестування вразливостей веб-додатків.

Мета роботи – провести аналіз інструментальних засобів тестування вразливостей веб-додатків з метою використання в реальних системах.

Методи дослідження – використовуючи різні методи тестування програмного забезпечення здійснити їх порівняння та вибір оптимальних, що доповнюють один одного, виконати аналіз інструментальних засобів тестування вразливості веб-додатків.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати діючі міжнародні стандарти, рекомендовані протоколи та практики у галузі інформаційної безпеки;

- розглянути питання захисту веб-ресурсів, основних принципів безпеки веб-додатків, загальних загроз та вразливостей;

- проаналізувати актуальні загрози веб-ресурсів, надати характеристику методів виявлення вразливостей, які можуть бути застосовані зловмисниками;

- дослідити типи атак на веб-ресурси та їх наслідки;

- проаналізувати методи та засоби захисту веб-додатків;

- провести дослідження методів експлуатації вразливостей для оцінки рівня захищеності веб-ресурсів;

- сформулювати рекомендації щодо застосування інструментів тестування вразливостей веб-додатків.

1 МЕТОДИ СТВОРЕННЯ ВЕБ-ДОДАТКІВ ТА РІВНІ ЇХ ТЕСТУВАННЯ

1.1 Види веб-додатків та технології їх створення

Концепція веб- додатків та їх класифікація

Веб-додаток — це набір статичних і динамічних веб-сторінок, тобто попередньо створених сторінок і програм, які створюють веб-сторінки у відповідь на запити користувачів. З певним ступенем припущення можна сказати, що веб-сайт і веб-додаток є ідентичними поняттями [1].

При створенні веб-додатків використовується клієнт-серверна архітектура. Користувач створює запит і передає його веб-додатку, веб-програма обробляє запит і надсилає його через мережу на веб-сервер. Сервер у відповідь на запит створює веб-сторінку та надсилає її через мережу назад до веб-додатку. Веб-додаток отримує відповідь на запит і відображає інформацію, надану користувачеві сервером. Веб- додаток використовує браузер як інтерфейс користувача.

Веб-додатки може розподілити на такі види:

- рейтинги;
- форми відправлення повідомлень;
- форми реєстрації;
- гостьові книги;
- форуми;
- форми завантаження;
- системи голосування ;
- пошукові системи;
- движки веб-сайту;
- content managment system (CMS);
- банерні движки та системи;

- веб-пошта ;
- персоналізовані веб-системи, які надають своїм користувачам комплекс послуг [2].

Основні вразливості наведеної вище класифікації веб-додатків включають:

- відсутність перевірки введення даних або тільки часткова перевірка даних;
- некоректна обробка вхідних даних;
- переповнення буферу;
- недбала робота програми з файлами, якщо ім'я файлу передано програмі ззовні (GET або POST);
- неврахування особливостей GET та POST запитів;
- некоректна робота з паролями;
- некоректні права доступу;
- некоректні права програм на сервері;
- неврахування особливостей роботи програм для завантаження файлів на сервер;
- некоректна логіка веб-додатку, яка при деяких правильних вхідних даних призводить до непередбачуваних наслідків;
- відображення інформації при помилках доступу до програми або бази даних (коли відображається додаткова службова інформація);
- некоректна робота з базами даних (паролі, виведення службової інформації, велика кількість запитів до бази);
- уразливості недостатньої обробки вхідних даних при роботі з базою даних дані (SQL-ін'єкція);
- неоптимізований програмний код, що призводить до значних навантажень на веб-сервер;
- вразливість веб-додатків та систем до DoS Так DDoS атак [2].

За типом призначення веб-додатки можна розділити на такі типи: CRM, ERP, ITRP, OSS, CGI.

CRM (Customer Relationship Management) – веб-додатки для автоматизації та підвищення ефективності бізнес-процесів (обробка замовлень, маркетинг, обслуговування клієнтів). CRM використовуються в спеціалізованих операторських «контакт-центрах». Веб-додатки Microsoft CRM реалізовано за допомогою SQL-сервера, та передбачає створення основного сховища даних Microsoft CRM, бази метаданих, бази звітів і дистрибуційної бази даних, яка призначена для відстеження взаємодії офлайн-користувачів клієнта Outlook з основною базою даних Microsoft CRM. Використання XML дозволяє інтегрувати Microsoft CRM з програмами аналогічного призначення, незалежно від мови програмування та операційної системи. Система передбачає обмеження доступу та перевірку прав доступу клієнта.

ERP (Enterprise Resource Planning) - веб-додатки, призначені для автоматизації процесів управління внутрішньогосподарською діяльністю компанії, включаючи управління виробництвом, фінансами, постачанням і персоналом.

ITRP (IT Resources Planning) - це клас веб-додатків, розроблених для підтримки управління корпоративними IT-ресурсами та послугами.

OSS (Operation Support Systems) - різновид веб-додатків, які призначені для підтримки роботи операторів розподілених комп'ютерних мереж. OSS забезпечує керування мережею, продуктивністю, усуненням несправностей, створення та облік послуг, планування мережевих ресурсів, моніторинг активності, контроль безпеки, якості послуг і рівня обслуговування клієнтів, а також збір статистичних даних. Різновидом OSS є система підтримки бізнесу - BSS (Business Support Systems).

CGI (Common Gateway Interface, загальний шлюзовий інтерфейс) - програми пошуку у віддалени базах даних, перенаправлення посилань, використання графічних меню, взаємодія з базами даних (за допомогою програм для конвертації форматів баз даних у формат мови HTML) [3].

Технології створення веб-додатків

Інструменти для створення веб-додатків включають: ISAPI, CGI, ASP, JSP, WAP. За час існування інтернету зміст веб-додатків, їх функції, принципи та архітектура їх побудови зазнали значних змін. Від простих засобів зберігання HTML-сторінок вони перетворилися на готові рішення, спрямовані на підтримку роботи корпоративних інформаційних систем [3].

AJAX (Asynchronous JavaScript and XML). Підхід до створення інтерфейсу користувача для веб-додатків, у якому для отримання відповіді на кожен запит користувач не повинен перезавантажувати сторінку браузера, а має лише оновити або додати нові дані. [3]. Тобто запити до сервера генеруються та надсилаються у фоновому режимі, жодним чином не відображаючи це користувачеві, а відповідь вставляється на готову веб-сторінку, оновлюючи лише певну її частину. Такий підхід дозволяє створювати більш зручні веб-інтерфейси для користувачів для активної взаємодії. Асинхронність у цьому випадку дозволяє користувачеві продовжувати перегляд вмісту веб-сторінки під час запиту до сервера, що дуже зручно. Класична модель AJA: користувач відкриває веб-сторінку, клацає на деякий елемент, браузер надсилає відповідний запит на сервер, у відповідь сервер генерує лише змінену частину веб-сторінки. Результатом є те, що певна частина веб-сторінки змінюється без її перезавантаження [4]. AJAX — це концепція використання пов’язаних специфічних технологій:

– головною складовою підходу є JavaScript – динамічна, об'єктно-орієнтована мова програмування, яка дозволяє динамічно завантажувати код за допомогою DOM, що виконується за допомогою формату JSON [5].

– DHTML (Dynamic HTML) використовується для динамічного змінення вмісту сторінки. Ця технологія створення веб-сайтів розглядає HTML-документ як об’єктну структуру, використовує комбінацію статичної мови розмітки HTML, скриптової мови JavaScript (виконання на стороні клієнта), CSS

(каскадні таблиці стилів) та DOM (об’єктна модель документа). Цю технологію також можна використовувати для навігації або для додавання інтерактивності формам веб-додатків [6].

– Важливим є використання XMLHttpRequest – браузер форми API - запит для зв'язоку із сервером у фоновому режимі за допомогою протоколу HTTP , що дозволяє уникнути повного перезавантаження сторінки. Запит такого формату можна використовувати як для синхронного, так і для асинхронного обміну інформацією у довільному текстовому форматі (XML, HTML тощо). Використання XMLHttpRequest створює враження «миттєвої» відповіді сервера порівняно з класичним методом перезавантаження всієї сторінки для оновлення представленої на ній інформації [7].

– Динамічне створення дочірніх фреймів [4].

ASP (Active Server Pages) — це технологія, яка дозволяє створювати динамічні веб-сторінки HTML за допомогою об’єктної моделі інтерфейсу, створеної на основі ISAPI (Internet Server Application Programming Interface) розширень та фільтрів. Принцип, який лежить в основі інтерфейсу, полягає в тому, що веб-сторінка містить фрагменти коду, які інтерпретуються веб-сервером, на кому встановлено розширення ISAPI, а він забезпечує користувача вже готовим результатом виконання вибраного фрагменту коду. Спеціальний «динамічний» тег вказує на наявність необхідного коду. Можна використовувати практично будь-яку мову програмування. JavaScript і VBScript підтримуються за замовчуванням.

ISAPI (Internet Server Application Programming Interface) – інтерфейс до Інтернет-сервера Microsoft, призначений для програмного керування сервером. ISAPI підтримується більшістю постачальників програмного забезпечення. Програми ISAPI — це особливий тип програми, яка обробляє запити користувачів і відображає їхні результати як потік HTML, який надходить безпосередньо до браузера клієнта.

JSP (Java Сервер Pages) — технологія створення веб-додатків на основі одноразову компіляції коду Java (сервлет) при першому його виклику з подальшим виконанням методів цього сервлет та розміщенням отриманих результатів до набору даних, котрий надсилаються до браузера [3].

1.2 Аналіз розробки програмного забезпечення

Структура веб -додатків

Найчастіше для створення програмного забезпечення за основу беруть як мінімум 3 компоненти:

1.Клієнтська частина являє собою графічний інтерфейс. Користувач взаємодіє з веб-додатком через браузер, який відображає графічний інтерфейс. Ця частина може використовувати такі технології: JavaScript, Flash, Java / JavaFX, ActiveX, Silverlight.

2.Серверна частина представлена програмою або сценарієм на сервері, що обробляє запити користувачів через браузер. Ця частина відповідає за централізовану обробку та зберігання даних, підтримку веб-інтерфейсу користувача, інформування користувачів у вигляді електронних повідомлень про події в системі та взаємодіє з обліковою системою клієнта. Найчастіше серверна частина веб-додатку програмується за допомогою PHP. Щоразу, коли користувач натискає посилання, браузер надсилає запит на сервер. Сервер обробляє цей запит, викликаючи певний скрипт PHP, який генерує веб-сторінку, описану мовою розмітки HTML, і надсилає її користувачеві через мережу. Наприкінці браузер відображає результат у вигляді іншої веб-сторінки.

База даних — це програмне забезпечення на сервері, яке відповідає за зберігання даних і видачу їх у потрібний час. Сама база даних знаходиться безпосередньо на сервері. Сервер відповідає за зв'язок з базою даних. Серверна частина веб-додатку, тобто скрипт PHP, звертається до бази даних, отримуючи дані, необхідні для створення сторінки, яку запитує користувач [8].

Класифікація методик розвитку ПЗ

Загально відомі такі методології розвитку програмного забезпечення:

- каскадна модель розробки (waterfall);
- гнучка модель розробки (agile);
- екстремальне програмування (XP).

Перевагами каскадної моделі розробки є повний комплект проектної документації після кожного етапу і чітка послідовність етапів. Недоліки - слабкий зв'язок з реальністю, жорсткі обмеження на формулювання вимог до проєкту, результати доступні замовнику тільки в кінці проєкту.

У гнучкій моделі розробки програмного забезпечення тестування не є одним і єдиним етапом, а повторюється залежно від ітерації та розробки. Перевагами є те, що вимоги не встигають ставати неактуальними, фіксована тривалість ітерації, команда самостійно оцінює завдання, самокерована команда, зворотний зв'язок, крос-функціональна команда. Але великим недоліком є ефективність. Ця модель виправдана лише для проєктів інфраструктурно складних проєктів, проектів з довгим строком циклу узгодження технічного завдання, проєктів без зворотного зв'язку, недостатньо кваліфікованої команди.

Екстремальне програмування - це спрощена методологія організації розробка програмного забезпечення для малих і середніх команд розробників, які займаються створенням продукту в невизначених або швидко змінюваних умовах. Основними цілями є підвищення довіри клієнтів до програмного продукту шляхом надання реальних доказів успішності процесу розробки та різкого скорочення часу розробки продукту. У той же час екстремальне програмування орієнтоване на мінімізацію помилок на ранніх стадіях розробки, що скорочує час випуску продукту. Принципами екстремального програмування є інтерактивність, простота рішень, інтенсивна розробка малими групами. зворотний зв'язок з клієнтом, представник якого насправді залучений у процес розробки, достатній рівень сміливості і бажання йти на ризики.

1.3 Рівні і види тестування програмного забезпечення у веб-додатках

Концепція тестування програмного забезпечення

Тестування – це процес перевірки відповідності між фактичною та очікуваною поведінкою програми, що здійснюється на скінченному наборі певним чином обраних тестів [9].

Оскільки якість ніколи не може бути абсолютною, а лише суб’єктивною, тестування з метою своєчасного виявлення помилок і дефектів не гарантує, що тестоване програмне забезпечення є повністю правильним. Він лише порівнює результати програми з вимогами, визначеними в специфікаціях. Специфікація - це повний перелік вимог до поведінки системи, яку буде розроблено. Складається специфікація зі сценаріїв користувача (use cases), які описують варіанти взаємодії між користувачем і програмним забезпеченням.

Отже, тестування є одним із прийомів контролю якості, який включає заходи з планування роботи (Test Management), проектуванню тестів (Test Design), виконання тестування (Test Execution) і аналізу отриманих результатів (Test Analysis). Після завершення тестування виявляється інформація про якість програмного забезпечення.

Якість програмного забезпечення – це сукупність характеристик програмного забезпечення, пов’язаних із його здатністю задовольняти заявлені та запропоновані потреби [10].

Верифікація – це оцінка програмного забезпечення для визначення відповідності отриманих результатів цього етапу розробки умовам, які були сформульовані напочатку цього етапу. Тобто це перевірка виконання цілей, завдань та термінів під час певного етапу розробки програмного забезпечення та узгодження їх з початково запланованими [9].

Валідація – це визначення того, чи відповідає програмне забезпечення, що розробляється, потребам і очікуванням користувача та вимогам системи [9].

«Баг» - це відхилення фактичного результату від очікуваного результату [11].

Відмова - це відхилення програми від функціонування та нездатність програми виконувати функції, визначені вимогами та обмеженнями.

Виділяють такі етапи тестування:

1) аналіз;

2) розвиток стратегії тестування і планування процедури контролю якості;

3) робота з вимогами;

4) створення тестової документації;

5) тестування прототипу;

6) основне тестування;

7) стабілізація;

8) експлуатація.

План тестування (Test Plan) — це документація, яка описує повний обсяг усієї роботи з тестування: опис об’єкта, стратегії, часу, критеріїв для початку та завершення тестування, обладнання, оцінки ризиків, варіанти вирішення проблем тощо. План тестування відповідає на запитання: Що потрібно тестувати? Що будуть тестувати? Як це буде тестуватися? Коли це буде тестуватися? Які критерії для початку тестування? Які критерії завершення тестування? [9].

В стандарті IEEE 829 перелічені пункти, з яких може складатися план тестування:

– ідентифікатор плану випробувань;

– введення;

– тестові завдання;

– характеристики для тестування;

– ознаки для не тестування;

– підхід;

– пункт критерій проходження/провалу;

– критерії тимчасової зупинки і вимоги відновлення;

– тест очікуваних результатів;

– завдання тестування;

– потреби навколишнього середовища;

– відповідальності;

– потреби у навчанні;

– розклад;

– ризики і непередбачені обставини;

– допуски[9].

Тест дизайн (Test Design) – це етап тестування програмного забезпечення, під час якого формуються і створюються тестові випадки (test cases), відповідно до сформульованих цілей тестування і критеріїв якості.

Відомі такі техніки тест дизайну:

1) Еквівалентний поділ (Equivalence Partitioning - EP) ;
2) Аналіз граничних значень (Boundary Value Analysis - BVA) ;
3) Причина / наслідок (Cause/Effect - CE) ;
4) Передбачення помилки (Error Guessing - EG) ;
5) Вичерпне тестування (Exhaustive Testing - ET).

Матриця відповідності вимогам (Traceability matrix) — двовимірна таблиця, що містить відповідність функціональним вимогам (functional requirements) продукту та підготовлених сценаріїв тестування (test cases). У заголовках стовпців таблиці містяться вимоги, а в заголовках рядків – тестові сценарії. На перетині позначка означає, що вимога поточного стовпця покривається тестом поточного рядка. Матриця відповідності вимогам використовується інженерами із забезпечення якості для перевірки охоплення продукту тестами. Це невід’ємна частина плану тестування [12].

Тестова ситуація (test case) – це набір кроків, конкретних умов і параметрів, необхідних для перевірки реалізації тестованої функції або певної її частини.

Тестові дані (test data) — це дані, які існують на початку виконання тесту та впливають на роботу або зазнають впливу з боку системи або компонента, що тестується. Тестові дані можуть бути створені тестером, модифікованими реальними даними, цілком реальними даними.

Звіт з тестування (Test report) – документ, основним призначенням якого є відображення результатів виконання плану тестування.

Тестове покриття (Test coverage) — одна з метрик для оцінки якості тестування, яка представляє повне покриття вимог або виконуваного коду тестами.

Баг/дефект репорт (Bug report) — це документ, який описує ситуацію або послідовність дій, що призводять до певної роботи об’єкта тестування, із зазначенням причини та очікуваних результатів.

Класифікація за рівнем тестування ПЗ:

- Модульне тестування (Unit Testing);
- Інтеграційне тестування (Integration Testing;
- Системне тестування (System Testing);
- Операційне тестування (Release Testing);
- Приймальне тестування (Acceptance Testing).

Класифікація видів тестування представлені на рисунку 1.1 [13].

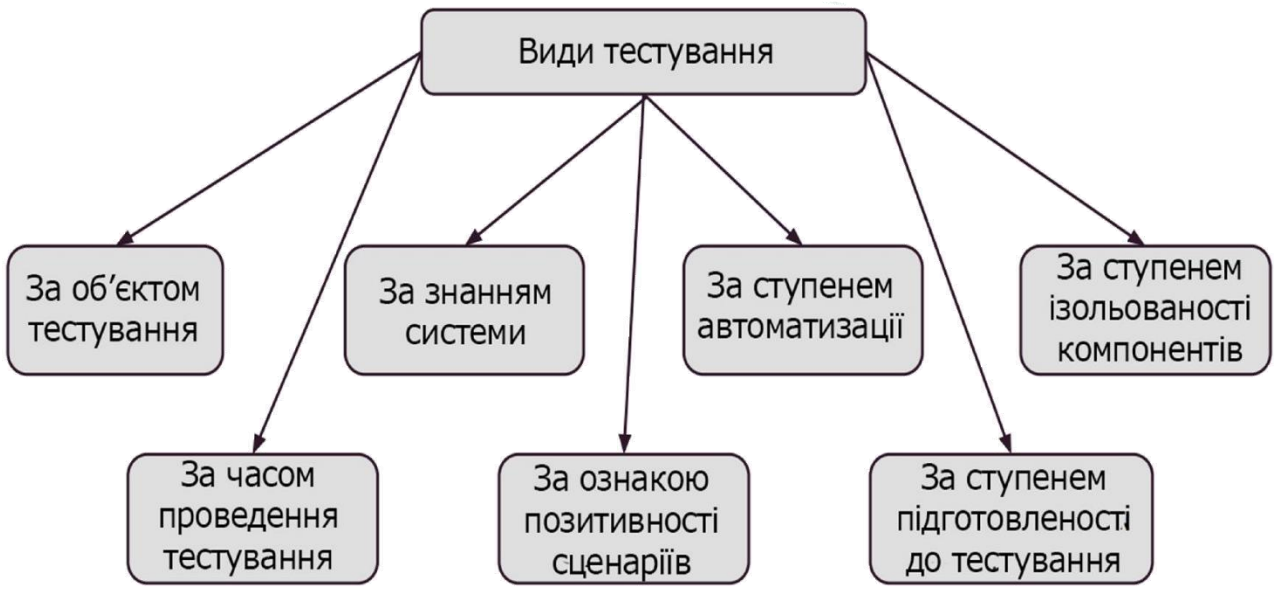

Рисунок 1.1 – Класифікація тестування ПЗ за ознаками

Види тестування за об’єктом тестування [13]:

– функціональне тестування перевіряє попередньо визначену поведінку та базується на аналізі специфікацій функціональності компонента або системи в цілому;

– тестування продуктивності;

– тестування безпеки використовується для перевірки безпеки системи;

– тестування інтерфейсу користувача;

– тестування зручності використання спрямоване на встановлення ступеня зручності використання, навчання, зрозумілості та привабливості для користувача в заданому контексті умов;

– тестування локалізації – процес тестування локалізованої версії програмного продукту;

– тестування на сумісність перевіряє здатність програмного забезпечення працювати в конкретному середовищі.

За знанням системи існують наступні види тестування[13]:

– тестування чорного ящика,

– тестування білого ящика,

– тестування сірого ящика.

За ступенем автоматизації розділяються тестування на[13];

– ручне тестування (manual testing);

– автоматизоване тестування (automated testing);

– напівавтоматизоване тестування (semiautomated testing).

За ступенем ізольованості компонентів [13]:

– модульне тестування складається з тестування кожного окремого модуля системи;

– інтеграційне тестування;

– системне тестування.

За часом проведення тестування розрізняють [13]:

– альфа-тестування,

– бета-тестування.

За ознаками позитивності сценаріїв визначають такі типи тестування [13]:

– позитивне тестування;

– негативне тестування.

За ступенем підготовленості до тестування розрізняють [13]:

– тестування по документації;

– інтуїтивне тестування;

– дослідницьке тестування.

Під час дослідження були розглянуті типи веб-додатків і технології їх створення. Також було виявлено, що в часи розвитку Інтернет-технологій надійність та захищеність веб-додатків є обов’язковою необхідністю. Було розглянуто основні методи тестування програмного забезпечення для задоволення висунутих вимог. У ході аналізу виникла необхідність проведення аналізу безпеки веб-додатків.

2. АНАЛІЗ БЕЗПЕКИ ВЕБ-ДОДАТКІВ

2.1Аналіз безпеки

За останні роки значно знизилася доля веб-додатків, які мають вразливості високого рівня ризику. Кількість критично небезпечних уразливостей має тенденцію до зменшення

Аналізуючи дані за останні п'ять років, можна побачити закономірний спад частки сайтів, що містять критичні уразливості та загальне підвищення рівня захищеності.

Рівень безпеки веб-додатків визначається експертами за результатами перевірок і залежить від потенційного впливу на систему [14]. Топ десять найбільш розповсюджених вразливостей представлена на рис. 2.1.

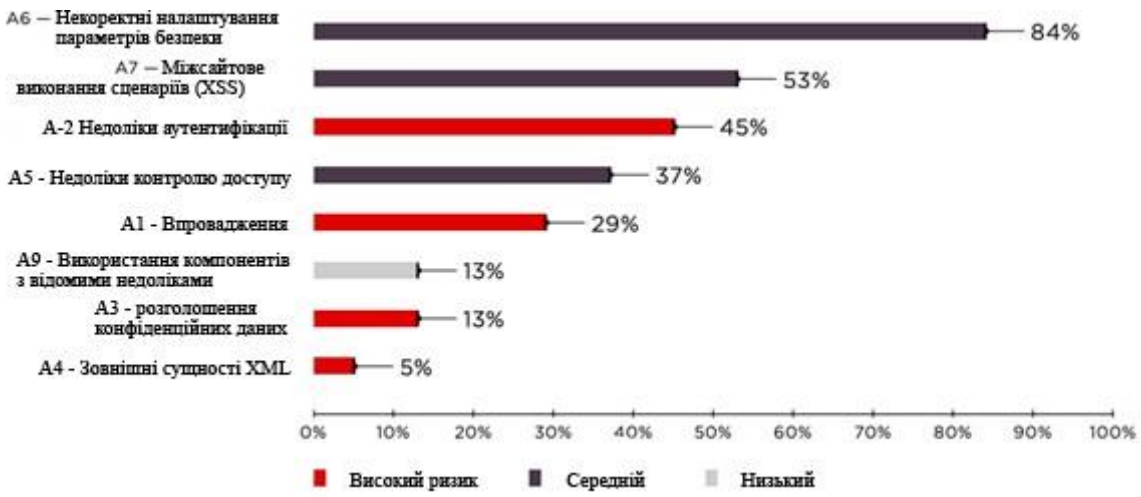

Рисунок 2.1 – Найбільш розповсюджені вразливості зі списку OWASP Top 10

Було виявлено уразливості пов'язані с невірними параметрами безпеки (Security Misconfiguration). В кожному п'ятому додатку були виявлено уразливості що дозволити атаку на сесію, включаючи відсутність HttpOnly і Secure в конфіденційних параметрах кукі. Використовуючи ці недоліки хакери можутьпроводити міжсайтові атаки (Cross-Site Scripting, XSS), щоб перехопити

ідентифікатор сесії користувача та від його імені виконувати різні дії у веб-додатку.

У 45% веб-додатків було виявлено недоліки автентифікації. Майже третина з виявлених вразливостей з цієї категорії є некоректним обмеженням кількості невдалих спроб автентифікації. Використовуючи цю вразливість, злочинці можуть збирати дані користувача та таким чином отримати доступ до веб-додатоку. Більшість атак на автентифікацію пов'язано з використанням паролів. Відповідно до останніх рекомендацій, необхідно використовувати багатофакторну автентифікацію. Рис. 2.2 показує вразливості, які пов'язані з недоліками автентифікації.

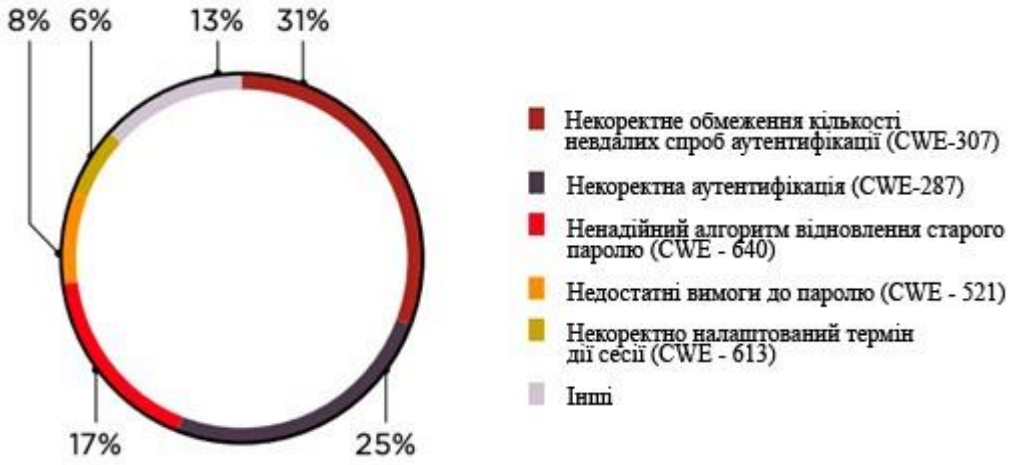

Рисунок 2.2 – Уразливості, які пов'язані з недоліками автентифікації

Недоліки контролю доступу виникають у кожному третьому веб-додатку. Обхід обмеження доступу зазвичай призводить до несанкціонованого розкриття, зміни або руйнуванню даних.

Кількість вразливостей, пов'язаних з авторизацією та автентифікацією можна мінімізувати, якщо в процесі розробки веб-додатку дотримуватися практик безпечного програмування.

Крім вразливостей зі списку ТОП 10, OWASP визначає ряд недоліків, на які рекомендовано проводити перевірку. Майже третина веб-додатків є вразливими до атак Clickjacking (містить уразливості, пов’язані з неправильним відображенням важливої інформації інтерфейсом користувача) і стільки ж до атак, пов’язаних з підробкою міжсайтових запитів. Під час атаки CSRF зловмисник за допомогою заздалегідь спеціально сформованих сценаріїв може виконувати дії від імені користувача, який авторизувався у веб-додатку.

На рисунку 2.3 представлено вразливості, які не ввійшли до OWASP ТОП 10

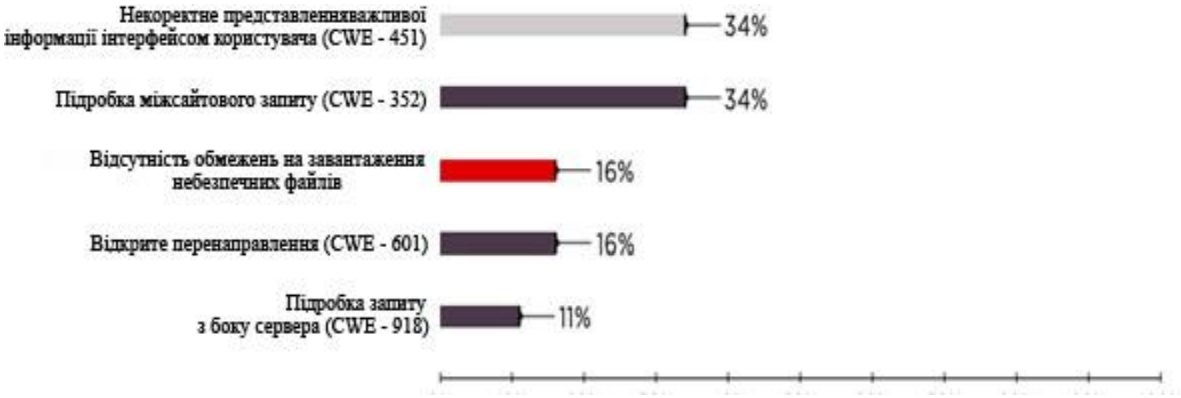

Рисунок 2.3 – Вразливості, які не ввійшли до OWASP ТОП 10

2.2Аналіз загроз

Найбільш розповсюдженою для веб-додатків є загроза атаки на клієнтів. Значну роль пр цьому відіграє «Міжсайтове виконання сценаріїв».

В результаті використання вразливостей нападник заразити комп'ютери користувачів шкідливим програмним забезпеченням, здійснювати фішингові атаки, а також виконувати дії від імені користувача. Під час формування загальних рекомендацій відносно захисту треба пам’ятати, що всі дані, котрі надходять від користувача та потім відображаються в браузері мають пройти попередню обробку. Потенційно небезпечні символи та слова, які можуть використовуватися при форматуванні сторінок HTML, мають бути заміненими на еквіваленти, які не є символами форматування. Також рекомендовано використовувати сучасниі мережеві екрани - брандмауери веб-додатків,тому що

вони мажуть заблокувати міжсайтове виконання скриптів. На рис. 2.4 представлено 5 найбільш розповсюджених загроз.

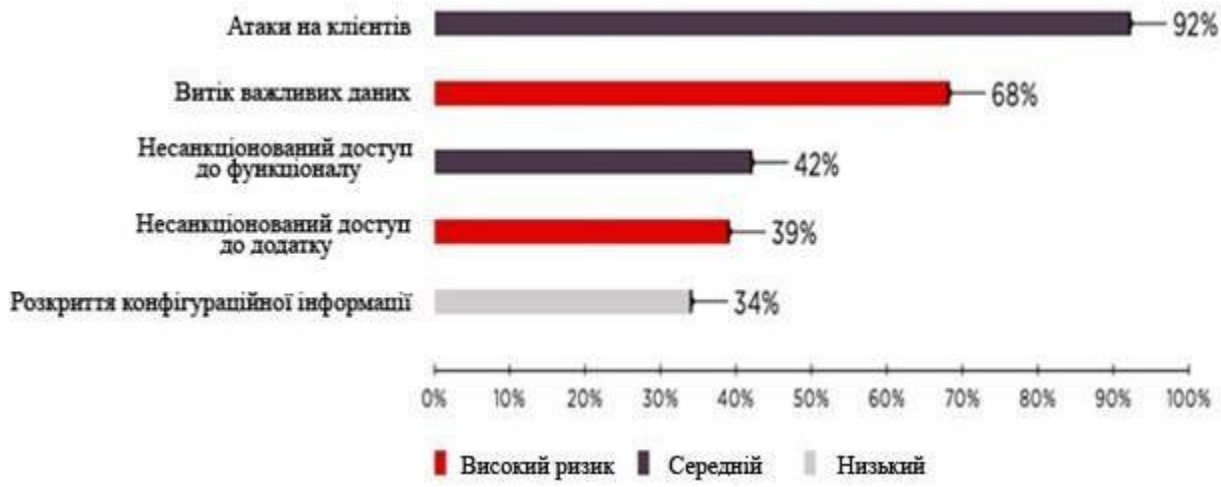

Рисунок 2.4 – 5 найбільш розповсюджених загроз

Витік важливої інформації є другою за актуальністю загрозою безпеці веб-додаткам. Майже половина витоків (47%) загрожує персональним даним, а 31% - даним користувача. Саме крадіжка інформації є основною метою зловмисників при атаці на компанії.

Результати вивчення свідчити, що на сьогоднішній день далеко не всі компанії можуть забезпечити надійний захист особистих даних.

У 16% веб-додатки знайдено критично небезпечні вразливості, які дозволяють отримати контроль як над веб-додатком так і над операційною системою сервера.

Зловмисник після отримання контролю над веб-додатком може вбудувати в нього код JavaScript і продовжити атака не лише на сам веб-додаток, а і на його користувачів. Щоб їх виявити такі вставки, необхідно провести аналіз захищеності методом білого ящика.

У разі спрямованої атаки на компанію вразливості веб-додатків можуть допомогти нападникам отримати дані про внутрішню мережу компанії (структуру мережі взагалі та її сегментів, вибір потрібного порту, про сервіси

тощо). У деяких випадках порушник навіть може отримати доступ до внутрішніх ресурсів компанії та до конфіденційної інформації, яка там зберігається.

Зловмисник може збирати та зберігати викрадену інформацію у спеціальних базах, а потім використовувати її для здійснення атак на веб-ресурси конкретної компанії або за допомогою отриманої інформації і на ресурси інших компаній-партнерів.

2.3 Інструментальний аналіз захищеності коду

Для проведення інструментальної перевірки веб-додатків використовуються спочатку сканери безпеки (у більшості випадків перевірка обмежується тільки саме використанням сканерів), які дозволяють при перевірці веб-додатка ідентифікувати потенційну схильність їх до різних уразливостей.

Інструментальне тестування вразливостей веб-додатків є найбільш простим і найбільш поширеним способом проведення аналізу безпеки. Однак саме простота такого способу спричиняє помилки «другого роду». Такі помилки трапляються, коли частина вразливостей, які є у досліджуваного веб-додатку, не виявлені в процесі виконання аналізу. Це може бути пов'язано з обмеженням функціональних можливостей автоматизованих засобів тестування.

Незважаючи на деякі обмеження у виконанні аналізу безпеки в певному сенсі використання сертифікованиих сканерів безпеки задовольняє вимоги стандарту PCI DSS (Payment Card Industry Data Security Standard). Цей стандарт описує вимоги до інформаційної безпеки для компаній, що працюють з міжнародними платіжними системами. З 2007 року цей стандарт є обов'язковим для виконання для міжнародних та українських компаній.

2.4 Неавтоматизований аналіз захищеності

У порівнянні з попереднім методом, ручний спосіб пошуку вразливостей веб-додатків дозволяє виявити більше вразливостей та провести перевірки, що неможливі при проведенні інструментального тестування. Проте слід зазначити,

що на його реалізацію може піти набагато більше часу, ніж при виконанні подібних робіт з використовуючи інструментальних засобів тестування.

Ручний метод часто використовується в тих випадках, коли неможливо або вкрай важко провести інструментальне сканування. Прикладом таких веб-додатків можуть бути ресурси, які використовують добре продуману модель захисту від уразливостей типу HTTP Request Forgery - «Підробка HTTP-запитів» (такий захист запобігає обробці запитів, що обходять логіку навігації програми). На практиці такі ресурси зустрічаються переважно в банківському секторі. У цьому випадку єдиний спосіб віддалено проаналізувати безпеку веб-додатку — виконати всі перевірки вручну. Консалтингові компанії в області інформаційної безпеки прагнуть надавати послуги з оцінки безпеки інформаційних систем на високому рівні та при проведенні аналізу безпеки намагаються поєднувати обидва методи - інструментальний і ручний. Завдяки цьому можна отримати максимально об'єктивну оцінку безпеки веб-додатку з мінімальними витратами часу на проведення такого аналізу.

2.5 Аналіз вихідниого коду

Такий аналіз дозволяє виявити всі вразливості, що загрожують веб-додатку. Однак через його складність це може займати багато часу, що залежить від складності стилю програмування розробника веб-додатку, а також обсягом коду для аналізу.

У реальних умовах аналіз вихідного коду в чистому вигляді та в повній мірі використовується переважно для окремих модулів або функції веб-додатків.

Розрізняють наступні методи аналізу захищеності веб-додатків.

Метод «чорного ящика». Полягає у виконанні робіт для оцінки безпеки веб-додатку без попереднього отримання будь-якої інформації з боку клієнта про досліджуваний додаток. Такий метод застосовується, коли необхідно оцінити безпеку веб-додатку з позиції зловмисника з мінімальним рівнем знання про досліджувану систему. В основному такі дослідження проводяться в межах

тестування на проникнення. При цьому відбувається моделювання дії справжнього зловмисника на інформаційну систему клієнта.

Такі дослідження можуть проводитися з попередженням персоналу компанії про планові роботи або без попередження (з імітуванням реальної ситуації нападу). У другому випадку з'являється можливість дослідити і реакцію персоналу при можливій атаці реального зловмисника та виникненні такого кіберінциденту.

Метод «Сірого ящика». Це найпоширеніша практика для проведення робіт з аналіз безпеки веб-додатків. Його принцип полягає в проведенні роботи з оцінки безпеки з наданням всієї необхідної інформації про веб-додаток виконавцю, крім прямого доступу до самого сервера, на якому працює веб-додаток. Зазвичай виконавцю надаються наступні дані: структура каталогів веб-додатку, необхідні дані для авторизації при підключенні до веб-додатку, вихідний код деяких файлів або функцій тощо.

Метод «Білого ящика». Це метод дозволяє досягти максимуму ефективності при проведенні аналізу захищеності. Такий принцип передбачає передачу всього веб-додатку виконавцю аналізу.

У цьому випадку спеціалісти з боку «зловмисника» мають можливість відслідковувати всі реакції веб-додатку на будь-які запити, що йдуть до його. Це найпродуктивніший метод проведення аналізу безпеки веб-додатків. Він дозволяє виявити найбільшу кількість загроз.

Найбільша кількість вразливостей веб-додатків виникає через помилки в коді веб-додатків. Для усунення таких помилок, а вдповідно і пов'язаних з цим загроз, є доцільним проводити аналіз вихідного коду (самостійно або із залученням фахівців з цього питання) в процесі безпечної робробки веб-додатку.

Аналіз безпеки методом «Білого ящика» виконують декілька фахівців одночасно, щоб не пропустити жодної деталі та ідентифікувати якомога більше недоліків. Також цей вид роботи включає як ручний аналіз коду, так і використання автоматичного тестування. Автоматизований пошук уразливостей допомагає прискорити процес тестування, однак при цьому потребує перевірки

вручну для уникнення випадкових тривог. Ручний спосіб тестування кому може забрати більше часу, однак при цьому гарантує актуальність виявлениих загроз.

В таблиці 2.1 представлена узагальнена інформація щодо виявлених вразливостей за допомогою детального (ручного) аналізу веб-додатків та автоматичному скануванні.

Таблиця 2.1 – Узагальнена статистика вразливостей веб-додатків

Тип уразливості	Автоматичне сканування		Детальний аналіз	
	% вразливостей	% вразливих сайтів	% вразливостей	% вразливих сайтів
Cross-Site Scripting	30,08	50,10	41,75	61,01
Information Leakage	29,82	97,19	12,50	16,94
SQL Injection	7,95	15,50	17,69	67,79
Brute Force	0,01	0,06	3,54	18,64
Path Traversal	0,23	0,70	4,95	11,86
HTTP Response Splitting	0,84	2,07	2,59	5,08
Predictable Resource Location	0	0	3,54	18,64
Insufficient Authentication	0	0	2,36	15,25
Abuse of functionality	0	0	1,65	6,77
Insufficient Process Validation	0	0	1,18	5,08
Insufficient Transport Layer Protection	11,18	53,25	0,94	3,38
Insufficient Session Expiration	0	0	0,71	5,08

Продовження таб. 2.1

Remote File Inclusion	0,22	0,44	0,71	3,38
Credential/Session Prediction	0	0	0,47	3,38
Insufficient Anti-automation	0	0	0,47	3,38
OS Commanding	0,08	0,06	0,47	3,38
Mail Command Injection	0	0	0,24	1,69
Session Fixation	0	0	0,24	1,69

2.6 Організація процесу тестування захищеності веб-додатків

Для проведення тестування захищеності веб-додатків необхідно, перш за все, визначити мету проведення такого аналізу, визначити область дослідження. Після цього з урахуванням стратегії управляння інформаційною безпекою в компанії необхідно сформувати перелік необхідних перевірок захищеності веб-додатків в межах виділеного на це бюджету.

Якщо метою аналізу безпеки веб-сайту є демонстрація проникнення, демонстрація порушення повний звичайного режиму його роботи або демонстрація компрометації інформації, тоді краще використовувати метод «Чоргого ящика». Результати цього дослідження наочно демонструють поточний стан захищеності керівництву компанії. Коли рівень безпеки досліджуваних об'єктів низький подібні роботи продемонструють загрози з боку зовнішнього порушника. Для усунення таких загроз та мінімізації ризиків може бути додаткове виділення бюджету компанії на роботу відділу захисту інформації.

Якщо метою аналізу безпеки веб-додатку є підвищення рівня безпеки цього ресурсу та водночас обмежений бюджетний тиск прийняття рішень, найкращим рішенням є організація процесу перевірки методом «Сірого ящика» із застосуванням інструментального аналізу та частковими ручними перевірками.

При виборі будь-якого варіанту проведення аналізу захищеності веб-додатку важливим є створення резервних копій ресурсу для мінімізації або запобігання втрат важливої інформації, а також мінімізації небажаних наслідків.

Таким чином, при організації процесу аналізу безпеки веб-додатків важливо визначити:

– мету дослідження веб-додатку;
– область дослідження;
– можливі обмеження при проведенні аналізу;
– необхідні методи та перевірки.

Після цього можна виконувати подальші кроки для аналізу захищеності веб-додатку, а саме:

– виконання резервного копіювання об'єктів вивчення;
– проведення аналізу захищеності веб-додатку та системи в цілому;
– усунення вразливостей, якщо вони були виявлені (змінити процеси системи управління безпекою, приділити більше уваги недолікам, які не були виявлені до цього, тощо);
– повторне проведення перевірки захищеності веб-додатку, підтвердження правильності усунення вразливостей, що були виявлені.

Дієвою практикою у системах управління інформаційною безпекою ситем в цілому та веб-додатків, як частини системи, є використання попереджувальних заходів – «превентивних» підходів. Аналіз захищеності веб-додатків має бути частиною глобальної стратегії захисту компанії та її інформаційних ресурсів, особливо у сферах критичної інфораструктури.

Слід зазначити, що рівень безпеки більшості веб-додатків продовжує залишатися на досить низькому рівні [16]. В цілому можна сказати, що скожного року з постійною тенденцією зменшується частка веб-додатків, які містять критичні вразливості.

Підтримка високого рівня безпеки для веб-додатків є складним та коштовним процесом. Для ефективного налаштування рівня захищеності веб-додатків необхідно дотримуватися таких основних рекомендацій-правил:

– необхідно виправляти виявлені вразливості як можна раніше;

– проводити максимальну автоматизацію процесів там, де це тільки можливо.

Для виконання цих правил компанії також мають приділяти значну увагу підготовці програмістів щодо застосування методів безпечного програмування та розробки і використання інструментів автоматизованого аналізу вихідного коду. Такі заходи дозволять зменшити кількість помилок та вразливостей ще на етапі проєктування та розробки системи. Крім цього завжди необхідно виконувати попереджувальні засоби та заходи захисту веб-додатків (брандмауер веб-додатків, WAF). Це дозволить зменшити ризики. Однак WAF не повинен виявляти та запобігати відомим атакам на рівні програм і бізнес-логіки, виявляти використання вразливостей так званого «нульового дня», а також аналізувати та порівнювати багато подій для виявлення можлиих атак та запобігати атакам на користувачів.

3. МЕТОДИ ТА ЗАСОБИ ТЕСТУВАННЯ ВЕБ-ДОДАТКІВ

3.1 Системи управління конфігурацією програмним забезпеченням

Конфігураційне управління (software configuration management, SCM) – це це набір методів, спрямованих на систематичний облік змін, внесених розробниками в програмний продукт під час його розробки та супроводу, підтримку цілісності системи після змін, запобігання небажаним і непередбачуваним ефектам, формалізацію процесу внесення змін [17].

Найпопулярнішими програмними серверами для безперервної інтеграції програмного забезпечення є:

– Jenkins – написаний на Java, інструмент безперервної інтеграції.

Працює в контейнері сервлетів, наприклад Apache Tomcat або GlassFish. Виконує підтримку інструментів для роботи з різними системами контролю версій, включаючи CVS, Subversion, Mercurial , Git і Clearcase, може створювати проекти Apache Ant і Apache Maven , а також виконувати shell-сценарії та команди Windows [18].

– BuildBot написаний на Python і заснований на роботі Twisted фреймворку. Розпочато як легша альтернатива Mozilla Tinderbox, а зараз використовується в таких проєктах, як Mozilla, Chromium, Webkit та багатьох інших. Buildbot встановлюється як головний сервер (master) і підпорядкований (slave). З головного сервера відстежуються зміни в репозиторії, координується робота підлеглих серверів, надсилаються звіти користувачам і розробникам. Підлеглий сервер може працювати під різними операційними системами. Налаштовується за допомогою скриптів Python на головному сервері. Ці скрипти для налаштування компонентів збірки дуже прості для розуміння, але вони мають усі можливості Python[19].

– CruiseControl – на платформі Java інструмент безперервної інтеграції програмного забезпечення, спрямований на автоматизацію процесу компіляції.

Управління та перегляд за збіркою здійснюється через веб-інтерфейс. Інтегрується з Apache Ant, різними системами контролю версій. Це програмне забезпечення з відкритим кодом, що розповсюджується за BSD-подібною ліцензією. Окрім версії Java, існують версії інструменту для платформи Microsoft .Net (CruiseControl.NET, CCNet) та версія для Ruby (CruiseControl. rb)[19].

– GoCD дозволяє виконувати збірки в різних системах і керувати ними в одному місці. Регулярно виконувані дії можна додати до джерел, а потім їх викликають для виконання. GoCD має простий та інтуїтивно зрозумілий інтерфейс, а також детальну документацію[19].

– Strider написаний на JavaScript і його серверному фреймворку Node.JS. Використовує MongoDB для зберігання даних. Щоб розпочати роботу зі Strider, потрібно встановити MongoDB і Node.js . Тоді його легко встановити за допомогою однієї команди через інсталятор пакетів Node.JS. Strider можна налаштувати за допомогою плагінів, але часто доводиться вручну редагувати файли конфігурації, щоб зробити його придатним для використання у конкретному проєкті[19].

– TeamCity серверне програмне забезпечення від компанії JetBrains написано на Java , білд-сервер для забезпечення постійної інтеграції [18].

– Travis CI ймовірно найпростіший CI сервер для початківців. Він поширюється як відкритий вихідний код, є безкоштовним для встановлення на сервері. Travis CI має версію SaaS, яка є безкоштовною для проєктів з відкритим кодом. При реєстрації і налаштуванні потрібно захистити свій обліковий запис GitHub, отримати необхідні права та додати файл .travis.yaml до свого проєкту. Travis CI скомпілює нову збірку після додавання змін до GitHub [19].

У таблиці 3.1 наведено порівняння характеристик найбільш популярних програм безперервної інтеграції. Порівняння здійснюється на основі платформи, обчислювальної платформи, інструментів побудови та інтеграції та підтримки різних інтеграційних середовищ [20].

Таблиця 3.1 – Порівняння серверів інтеграція[20].

Назва	Платформа	Збірка, Windows	Збірка, Ява	Збірка, інші	Інтеграція, середовище розробки	Інтеграція інші
Jenkins	Web container	MSBuild, NAnt	Ant, Maven 2, Kundo	Cmake, Gant, Gradle, Grails, Phing, Rake, Ruby, SCons, Python, shell script, command-line	Eclipse, IntelliJ IDEA, NetBeans	Bugzilla, Google Code, Jira, Bitbucket, Redmine, FindBugs, Checkstyle, PMD and Mantis, Trac, HP ALM
BuildBot	Python	Command-line	Command-line	Command-line	-	-
Cruise Control	Cross-platform	NAnt, Rake, Xcode	Phing, Apache Ant, Maven	catch-all 'exec'	Eclipse	-
GoCD	Cross-platform	Command-line	Command-line	Command-line	Hi	GitHub

Продовження таб. 3.1

Strider	Node.js	Hi	Hi	C, C++, Clojure, Erlang, Go, Groovy, Haskell, Java, Node.js, Perl, PHP, Python, Ruby, Scala	Hi	GitHub, Bitbucket, Heroku, GitHub Enterprise, Git
Team City	Web container	MSBuild, NAnt, Visual Studio, Duplicates finder for .NET	Ant, Maven 2-3, Gradle, IntelliJ IDEA .ipr based and Inspections and Duplicates finder	Rake, FxCop, command-line	Eclipse, Visual Studio, IntelliJ IDEA, RubyMine, PyCharm, PhpStorm, WebStorm	Jetbrains Youtrack, Jira, Bugzilla, FishEye, FindBugs, PMD, dotCover, NCover
Travis CI	Hosted	Hi	Ant, Maven, Gradle	C, C++, Clojure, Elixir, Erlang, Go, Groovy, Haskell, Java, Node.js, Perl, PHP, Python, Ruby, Rust, Smalltalk	Hi	GitHub, Heroku

У наведеній нижче таблиці 3.2 порівнюються характеристики деяких найпопулярніших програм безперервної інтеграції на основі системи контролю версій, яка є невід'ємною частиною системи програмного забезпечення безперервної інтеграції [20].

Таблиця 3.2 - Підтримка найпопулярніших систем SCM серверами інтеграції

Назва	Darcs	Git	GNU Bazaar	Integrity	Mercurial	Perforce	PVCS	Subversion
Jenkins	Так	Так	Так	Так	Так	Так	Так	Так
BuildBot	Так	Так	Так	Ні	Так	Так	Ні	Так
Cruise Control	Так	Так	Ні	Так	Так	Так	Ні	Так
GoCD	Ні	Так	Ні	Ні	Так	Так	Ні	Так
Strider	Ні	Так	Ні	Ні	Ні	Ні	Ні	Ні
Team City	Ні	Так	Ні	Ні	Так	Так	Ні	Так
Travis CI	Ні	Так	Ні	Ні	Ні	Ні	Ні	Ні

3.2 Засоби модульного тестування веб-додатків

Фреймворк jUnit є досить хорошим рішенням задач, пов'язаних з тестуванням java-додатків. jUnit використовується для модульного тестування, що дозволяє перевірити правильність окремих модулів вихідного коду програми. Перевагою такого підходу є ізоляція одного модуль від інших. Мета цього методу дозволяє програмісту переконатися, що модуль сам по собі здатний працювати правильно. jUnit — це бібліотека класів. Альтернативним фреймворком є TestNG , який призначений для тестування та поєднує в собі JUnit і NUnit. TestNG оновлено новими інноваційними функціями, які роблять його потужнішим і простішим у використанні.

Можливості TestNG:

– анотація;

– використання XML для гнучкої конфігурації тестів;

– підтримка data-driven тестування;

– залежні методи для тестування серверних додатків;

– підтримується в Eclipse, IDEA, Ant, Maven, Netbean, Hudson;

– тестування коду проходить багатопотоково, це дає безпеку та швидкодію;

– легкий перехід від JUnit[21].

NUnit – відкрите середовище модульного тестування додатків для . NET. Бібліотека була перенесена з Java (JUnit).

PHPUnit – це спеціальний фреймворк, призначений для тестування PHP скриптів. Має переваги PHPUnit:

– інструменти для створення модульних тестів і організацій їх в ієрархічні набори;

– застосування інтерфейсу командного рядка для виконання тестів;

– у якості провайдерів даних є генератори наборів даних для тестування елементів скрипта з використанням різних вхідних параметрів;

– підтримка тестування коду, який працює з базою даних;

– тестування винятків;

– підтримка фіктивних об'єктів;

– генератори звітів[22].

3.3 Алгоритми та інструменти функціонального тестування веб-додатків

Забезпечення якості програмного забезпечення є невід’ємною частиною життєвого циклу розробки, і зі зростанням складності програм, що тестуються, виникла потреба автоматизувати і керувати процесом тестування. Великі компанії-розробники програмного забезпечення мають власні розробки в цій області. Яскравий приклад - Hewlett Packard. Компанія (HP) має великий вибір рішень для тестування програмного забезпечення[24]. З відокремленням процесу

забезпечення якості в окрему частину розвитку програмного забезпечення збільшився і попит на інструменти для тестування [25,26].

Сьогодні існує велика кількість інструментів для тестування ПЗ: ресурси, виділені для тестування або розробки ПЗ, містять описи кількох сотень різних інструментів [27,28]. Зазвичай при такій кількості різноманітних рішень постає питання про раціональний вибір інструментів для тестування, але у випадку систем тестування, які мають ряд специфічних особливостей, вибір повинен здійснюватися з урахуванням усіх особливостей архітектури системи, а також її підсистем. Перелік засобів тестування, які реалізують найпоширеніші тестові підходи, досить великий. Тому слід провести аналіз найбільш часто використовуваних інструментів тестування.

Розглянувши інструменти для функціонального тестування, проаналізувавши різні ресурси [29,30,31], а також ознайомившись зі статистикою пошукових запитів на цю тему, можна виділити десять інструментів: п’ять комерційний і п'ять з відкритим кодом.

Комерційні інструменти:

1) HP Quick Test Professional (HP QTP);
2) SilkTest;
3) Telerik TestStudio;
4) TestComplete;
5) Ranorex.

Інструменти з відкритим кодом:

1) Selenium;
2) AutoIt;
3) SoapUI;
4) Sahi;
5) Watir.

На основі зібраних даних було вирішено вибрати п’ять з них, а саме:

– Selenium[32];

– HP Quick Test Professional (HP QTP)[33];

– Telerik TestStudio[34];

– TestComplete[35];

– Ranorex[36].

Selenium — це набір інструментів, призначених для автоматизації веб-браузерів на різних платформах. Selenium може автоматизувати багато різних браузерів на різних платформах, використовуючи різні мови програмування та інтегруючись із різними платформами тестування. Найпопулярнішою сферою застосування Selenium є автоматизація тестування веб-додатків. Однак за допомогою Selenium ви можете автоматизувати будь-які інші рутинні дії, які виконуються через браузер. Розробка Selenium підтримується виробниками популярних браузерів. Вони адаптують браузери для більш тісної інтеграції з Selenium, а іноді навіть реалізують власну підтримку Selenium у браузері. Selenium є центральним компонентом ряду інших інструментів і фреймворків автоматизації. Selenium підтримує мобільні та настільні браузери. Selenium дозволяє розробляти сценарії автоматизації майже будь-якою програмованою мовою. Використовуючи Selenium, можна організувати розподілені тестові випробування, що складаються із сотень машин із різними операційними системами та браузерами, а також запускати скрипти в хмарах[23].

Selenium WebDriver — це надійна система автоматизації, яка може працювати з будь-яким браузером. Він надає широкий набір тестів, включаючи тести з досить складною поведінкою та логікою перевірки.

Selenium WebDriver - це набір бібліотек для різних мов програмування, які дозволяють керувати браузером з програми, написаної на цій мові програмування.

Діаграма взаємодії між різними інструментами проекту Selenium показана на рисунку 3.1.

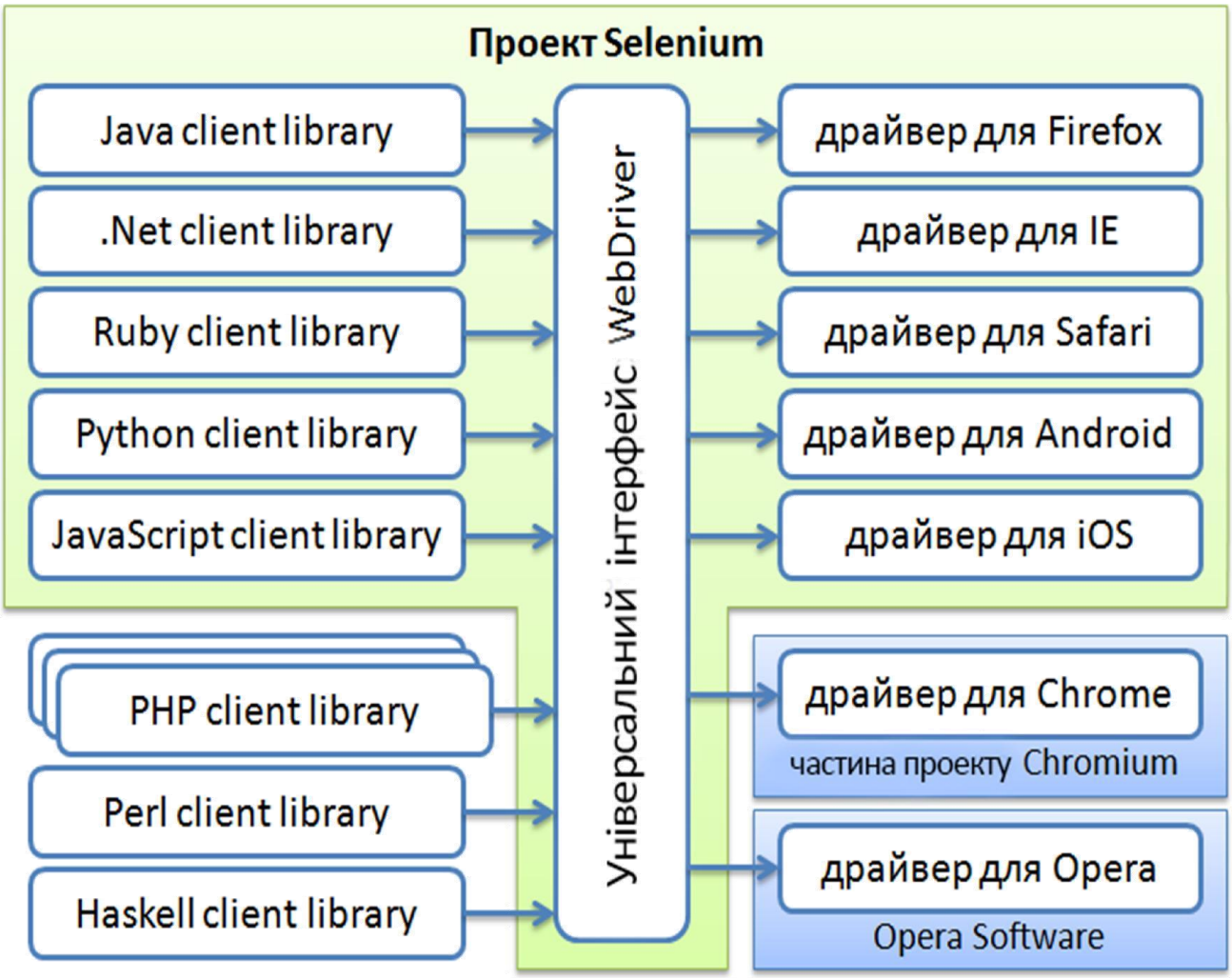

Рисунок 3.1 – Схема можливості Selenium WebDriver

Проект Selenium розробляє драйвери для браузерів Firefox, Internet Explorer і Safari, а також драйвери для мобільних браузерів Android і iOS. Драйвер для браузера Google Chrome розроблений в рамках проекту Chromium, а драйвер для браузера Opera (включаючи мобільні версії) розроблений компанією Opera Software. Тому формально вони не є частиною проєкту Selenium, а розповсюджуються та підтримуються незалежно. Аналогічна ситуація і з клієнтськими бібліотеками - в рамках проекту Selenium розробляються бібліотеки для Java , . Net (C #), Python , Ruby , JavaScript [23].

HP QuickTest Professional (QTP) є одним із провідних інструментів автоматизації функціонального тестування та його флагманським продуктом HP у своїй лінійці. Для розробки автоматизованих тестів QTP використовує мова

VBScript, яка підтримує такі технології: Presentation Foundation, Web services, Macromedia Flex, Ajax, Delphi, .NET, J2EEWeb, Visual Basic, ActiveX, Java, Oracle, SAP Solution, TE , PowerBuilder, Siebel, PeopleSoft, VisualAge, Stingray. Компанія HP рекомендує використовувати QTP для інтеграції HP Quality Center для налагодження зв'язку тестів з вимогами, зберігання тестів, керування їх запуском, формування звітів [26].

Його популярність багато в чому пояснюється наявністю в ньому рекордера для користувацької активності, що дозволяє записати дії користувача і перетворити їх на сценарій. Об'єкти, з якими взаємодіє користувач, автоматично ідентифікуються QTP і зберігаються в спеціальному сховищі репозиторію. Під час збереження в репозиторій QTP автоматично зберігає ідентифікаційні властивості об’єкта, але не завжди робить це правильно. Наприклад, якщо на веб-сторінці є кілька таблиць (навіть якщо кожна має власний ідентифікатор), QTP ідентифікує їх за порядковими номерами. Цей метод ідентифікації об'єктів викликає проблеми під час відтворення автотестів. Крім того, багато об'єктів взагалі не потрапляють у сховище під час запису. Це викликано багатьма причинами, найпоширенішою з яких є складне компонування або компонування з використанням DIV [38].

QTP — це програмне забезпечення для автоматизованого тестування, призначене для тестування різних програмних програм і середовищ. QTP виконує функціональне і регресійне тестування за допомогою інтерфейсу користувача (GUI або веб-інтерфейс). Працює, ідентифікуючи об’єкти в інтерфейсі програми чи веб-сторінки та виконуючи необхідні операції (наприклад, клацання мишею або події на клавіатурі); також може фіксувати властивості об’єкта (ім’я чи ID). QTP використовує мову сценаріїв VBScript для визначення процедури тестування та керування об’єктами та елементами керування тестової програми.

Хоча QTP зазвичай використовується для спрямованого тестування інтерфейсу автоматизації test case, він також може автоматизувати деякі тести на основі випадків , такі як операції файлової системи, тестування бази даних або

тестування веб-служб. Обробка винятків. HP QTP керує обробкою винятків за допомогою сценаріїв оновлення; мета полягає в тому, щоб продовжити виконання тестів, якщо станеться несподівана помилка. Оскільки QTP займає простір пам’яті програм, що тестуються, деякі винятки можуть призвести до припинення виконання QTP і неможливості відновлення.

Тестування базується на даних. HP QTP підтримує тестування на основі даних. Наприклад, дані можна вивести як таблицю для повторного використання в інших місцях. Реалізовано тестування на основі даних В книжкова форма Microsoft Excel , Що може отримати с QTP . HP QTP має два типи таблиць даних: глобальний записувач даних і діючий (локальний) лист даних. Тестові кроки можуть зчитувати дані з цих таблиць даних для контролю зміни даних в тестових програмах і для перевірки очікуваних результатів.

Автоматизація та складні об’єкти інтерфейсу користувача. HP QTP Може не розпізнати окремі об'єкти користувацького інтерфейсу та інші складні об'єкти. Користувачі можуть визначати ці типи об’єктів як віртуальні об’єкти. QTP не підтримує віртуальні об’єкти для аналогового або низькорівневого запису.

QTP можна розширити за допомогою окремих доповнень для ряду розробок, які не підтримуються за межами збірки. Додаткові компоненти включають підтримку Web , . NET , Java і Delphi [39].

Telerik WebUI Test Studio — це інструмент для автоматизації функціонального тестування веб-додатків (ASP.NET, Silverlight). Представляє розширення Microsoft Visual Studio. До основних переваг програми можна віднести:

– підтримку Silverlight;

– засіб автоматизації добре справляється із запитами AJAX будь-якої складності;

– немає проблеми в написання тестів з компонентами RadControls Telerik. Деякі елементи RadControls досить складно автоматизувати, наприклад, Selenium в попередніх версіях не міг працювати з випадаючими списками та

календарями;

– WebUI Test Studio має комфортний редактор тестів;

– інтеграція з іншими інструментами тестування. Всі можливості Visual Studio доступні;

– базовий функціонал будь-якого розумного сучасного інструменту автоматизації реалізований на хорошому рівні (Test Recorder, Elements Explorer, DOM Explorer, генерація коду, підтримка кількох браузерів тощо).

Недоліки WebUI Test Studio: по-перше, він може працювати лише з ASP . NET, програмами Silverlight . Також варто зазначити, що у Visual Studio 2010 виявилася незавершеними Coded UI Tests, які забезпечують приблизно той самий набір функціональних можливостей для автоматизації функціонального тестування, а з ліцензією можна тестувати програми Silverlight [26].

TestComplete — це автоматизований інструмент тестування, який дозволяє створювати тести для програм Windows , веб- серверів і веб- сторінок.

TestComplete використовується для автоматизації різних типів тестування програмного забезпечення для . NET , Java , Visual C ++, Visual Basic , Delphi , C ++ Builder , веб-сторінок та інших програм. Інструмент призначений для розробників програмного забезпечення та відділів тестування, щоб скоротити час, необхідний для ручного тестування.

TestComplete має спеціальний механізм, який полегшує створення сценаріїв для тестування програм. Суть механізму полягає в наступному: тестер виконує необхідні дії, які автоматично записуються у файл. Цей файл доступний для перегляду і редагування, а при запуску повторить всі операції, які виконувалися раніше. Важливою обставиною є те, що записані тести можуть бути пізніше змінені [24].

Базовий характеристики TestComplete:

– Тестування Windows і веб-додатків. Програма не залежить від типів програм і інструментів розробки. Тести роботи для програм, створених на C#, C++, Delphi , Java I будь-який інші мови.

– Кросбраузерне веб-тестування. Створені тести в одному браузері можна запускати в іншому підтримуваному браузері, жодним чином не змінюючи тест.

– Тестування додатків HTML5. Автоматизація тестування HTML5-контенту, включаючи нові теги, веб-форми та пов'язаний вміст JavaScript і CSS3.

– Вибір типу тестування. TestComplete дозволяє вибирати з різних типів тестування: тести GPU, розподілені тести та тести на основі даних.

– Тестування ключових слів. Незалежні від сценаріїв тести ключових слів і простий інтерфейс користувача дозволяють швидко почати створювати автоматичні перевірки.

– Функціональне тестування додатків RIA. Тестування надійності та функціональності програм HTML5, Flash, Flex, AIR та Silverlight.

– Інтегрована підтримка сторонніх рішень. Інтеграція з JIRA, Axosoft, Bugzilla тощо. Підтримує найпопулярніші елементи керування інтерфейсом, включаючи extJS 4, JavaFX2, Developer Express, Infragistics, Rogue Wave, Telerik, .NET.

Ranorex — це повноцінна середа розробки автоматизації тестування GUI . Ranorex — це також набір інструментів для написання тестів і бібліотек, призначених для комп'ютера, веб-орієнтованих і мобільних програм.

Це середовище забезпечує наступні можливості:

– Підтримка динамічно створюваних графічних частин керування (елементів керування) за допомогою GUI Object Repository. Для доступу до елементів використовується мова XPath, тому ви можете вибрати елемент, сформувавши відповідний запит за допомогою вбудованого реєстратора.

– гнучка і настроювана система пошуку елементів керування. У Ranorex XPath вирази називаються RanoreXPath. Якщо елемент не має унікального імені, то можна отримати до нього доступ однією дією, знаючи для нього вираз XPath.

– Проста підтримка тестів на основі даних (Data Driven Testing). Ranorex також має можливість розробляти тести на основі даних, але виключає формат XML для даних.

– Можливість розробляти власні модулі (фреймворки) і використовувати їх при розробці тестів на C#. Ranorex надає власне середовище розробки – Ranorex Studio, яке базується на SharpDevelop і тому має дуже схожий інтерфейс і основні операції з Visual Studio. Ranorex має зручний інспектор елементів і можливість створювати тести майже одним клацанням миші. Крім того, можлива розробка додаткових програмних модулів, які потім будуть використовуватися для створення більш складних тестових сценаріїв.

– Підтримка виконання тестів на сервері безперервної інтеграції (TeamCity). Ranorex, у свою чергу, може скомпілювати Test Suit у виконуваний файл, який потім можна просто запустити як окреме завдання збірки TeamCity.

– Формування інформативних звітів за результатами поточних тестів. Середовище розробки Ranorex створює чудові звіти з ілюстраціями та поясненнями. Однак для цього потрібно запускати тести з самої Ranorex Studio. Після виконання скомпільовані тести повертають XML-звіт Ranorex, який можна перетворити на xUnit.

– Можливість інтеграції тестів з тест-кесами системи управління тестами (TMS). У Ranorex Тестовий проект Studio компілюється в exe- файл. Тобто при поточній реалізації утиліти неможливо використовувати її як джерело тестів.

– Простота навчання Так використання тестери. Ranorex спеціально розроблено як тестове середовище. З його допомогою Зручно створювати, запускати та переглядати результати тестів. Крім того, після розробки кількох прикладів для тестового додатку не виникає проблем із пошуком динамічних елементів. Ranorex також успішно працював з . NET WinForms .

– Має можливість запускати тести за розкладом і з прив'язкою до системних подій.

Практика використання планувальників для організації розкладу виконання різних додатків не є новою і зустрічається в багатьох програмних продуктах. Усі проаналізовані засоби мають вміння працювати згідно з графіком. Наприклад, Telerik TestStudio , TestComplete і HP QTP дозволяє

виконувати тести у запланований час без додаткових плагінів або налаштувань. А планування в Selenium і Ranorex можна реалізувати за допомогою серверів безперервної інтеграції. Прив'язку до системних подій можна зробити за допомогою зовнішніх плагінів і утиліт. Наприклад, за допомогою Skybot Планувальник може організовувати запуск тести без жорсткий прив'язка до часу, а для будь-якої події [37].

Незважаючи на безліч способів реалізації прив'язки виконання тесту до часу або системних подій, всі методи мають свої недоліки. Вони мають складну конфігурацію додаткових утиліт і відносно невелику функціональність вбудованих засобів планування.

Усі розглянуті інструменти дозволяють запускати тестові сценарії паралельно в кількох тестових середовищах, що дає змогу розподілити тестове навантаження, а також скоротити час, необхідний для виконання незалежних тестів.

Наприклад, для вибраних комерційних інструментів ця функція включена за замовчуванням. У випадку з Selenuim необхідна інтеграція із зовнішніми модулями. Так, це рішення в поєднанні з бібліотекою TestNG дозволяє запускати кілька тестів паралельно, але з деякими обмеженнями:

– необхідна складна мережева архітектура типу клієнт -сервер;

– тести насправді будуть виконуватися в різних середовищах.

Однак з одночасним виконанням тестових скриптів все трохи складніше, тому що для всіх інструментів тест є найменшою неподільною одиницею автоматизації. Звичайно, можна організувати тестовий скрипт як псевдопаралельний, але це значно ускладнює роботу тестувальників, а у випадку тестування графічного інтерфейсу стає практично неможливим.

Також трек аналізувати або інструменти відносно загальний вимоги:

– можливість взаємодії с система ззаду за допомогою графічний інтерфейс;

– можливість взаємодії з системою через API;

– тестування Back- End;

– гнучкість і універсальність тест-сценаріїв;

– простота використання.

У Test Complete тести можна створювати в обох форматах за допомогою вбудованих редакторів. Цей інструмент підтримує такі мови, як: VBScript , JScript , DelphiScript та інші. Quick Test Professional дозволяє записувати дії користувача в тесті програми. Ці дії зберігаються в у вигляді сценаріїв, які можуть бути виконані пізніше і можуть бути представлені у двох форматах: як у кодова форма у VBScript, так в якість набору послідовних кроків із діями.

Selenium IDE також надає можливість записувати дії користувача, в даному випадку, як пари Command – Target , які можна експортувати в декілька форматів: HTML , Java , Groovy , C #, Python та інші. В Ranorex і Test Studio використовує Keyword-driven підхід.

Незважаючи на те, що всі розглянуті інструменти є потужними та комплексними рішеннями, вони розроблені з акцентом на тестуванні програм і додатків, які часто використовуються. Цей фокус, безумовно, важливий на ринку програмного забезпечення, але він значно ускладнює їх конфігурацію для більш специфічних розрахункових і клірингових систем, тим самим знижуючи ефективність процесу тестування та збільшуючи витрати на забезпечення якості програмного забезпечення (час, навчання, налаштування середовища та інструментів) [40].

Проведемо порівняльний аналіз досліджуваних засобів. Давайте напишемо функціональний тест для сторінки vidfall . com для кожного проаналізованого інструменту. заради більш точні дані час виконання тест Давайте зробимо це кожен тест 5 разів. З отриманої інформації складемо порівняльний графік часу виконання цього тесту.

Як видно з рисунку 3.2 Інструмент Selenium WebDriver є найшвидшим, тобто для його завершення потрібно дуже мало часу. Також Hewlett Packard QuickTest Professional має хорошу продуктивність, але вона нестабільна

(виконання тесту не показує однаковий час). TestComplete показав найгірший час виконання тестової функції.

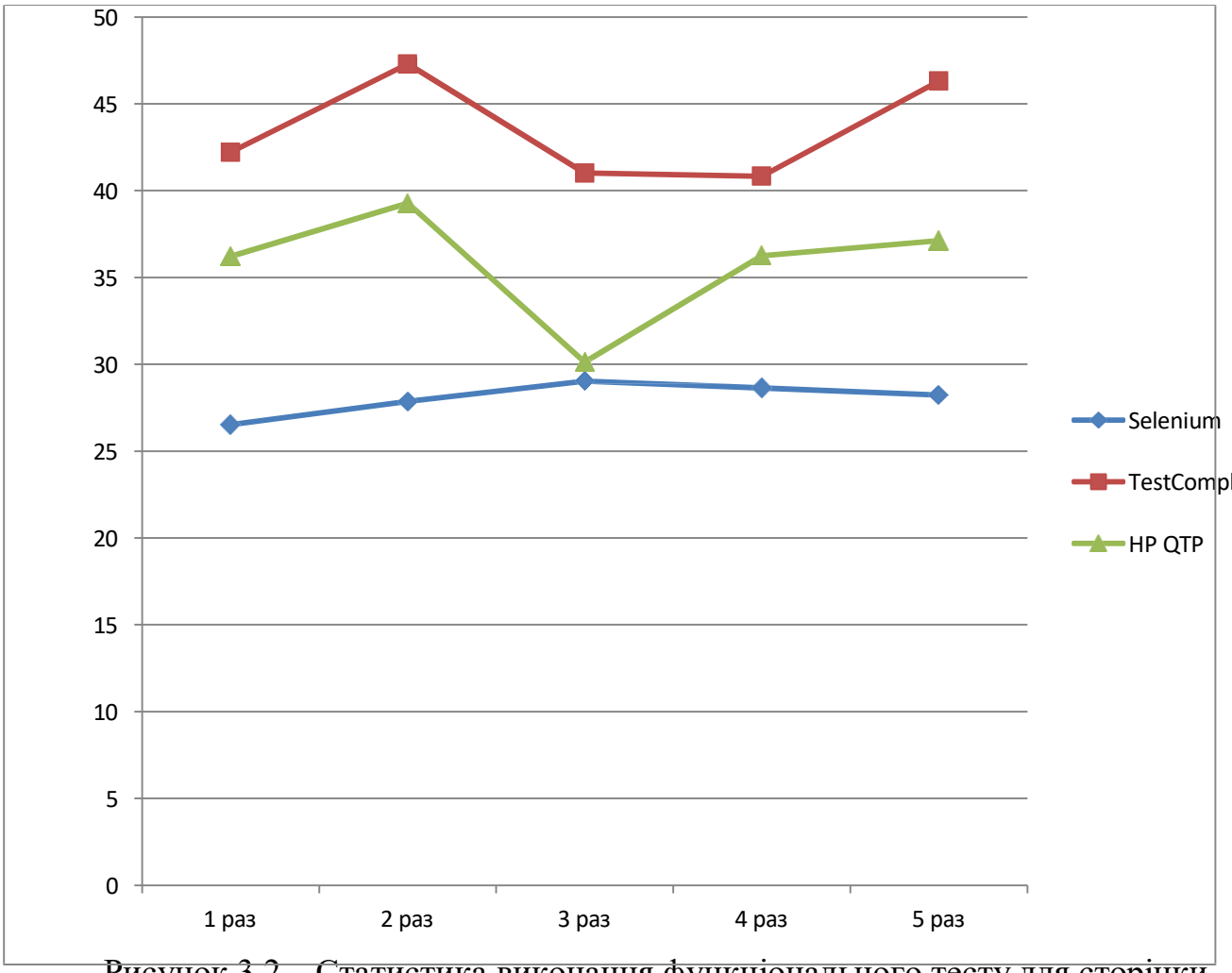

Рисунок 3.2 – Статистика виконання функціонального тесту для сторінки vidfall . com в залежності від часу

Отже, на основі аналізу ми виберемо найкращий інструмент для написання функціональних тестів – Selenium WebDriver .

3.4 Засоби тестування прикладного програмного інтерфейсу веб-додатку

Прикладний програмний інтерфейс (API – англ. Application Programming Interface) забезпечує зв’язок та обмін даними між двома окремими програмними системами. Програмна система, яка реалізує API, містить функції (підпрограми), які можуть бути виконані іншою програмною системою.

API повністю відрізняється від тестування графічного інтерфейсу і в основному зосереджується на рівні бізнес-логіки архітектури програмного забезпечення. Це тестування не буде зосереджено на зовнішньому у вигляді додатків.

Замість використання стандартного введення та виведення користувача (клавіатура) тестування API використовує програмне забезпечення для викликів API , отримання виводу та запису відповіді системи.

API вимагає взаємодії програми з API . Щоб перевірити API , вам потрібно:

– використовувати інструмент тестування для керування API .

– написати код для перевірки API.

Тестові випадки API тестування базуються на :

– Повернене значення базується на стані введення. Це порівняно легко перевірити, оскільки вхідні дані можна визначити, а результати можна перевірити.

– Результат не повернуто. Якщо повернення не має значення, можна перевірити поведінку API в системі.

– Виклик іншого API/події/переривання. Якщо вихід із API викликає будь-яку подію чи переривання, ці події та переривання відстежуватимуться.

– Оновлення структури даних. Оновлення структури даних матиме певний вплив або вплив на систему, і це має бути підтверджено.

– Зміна певних ресурсів. Якщо виклик API змінює деякі ресурси, це потрібно підтвердити шляхом доступу до відповідних ресурсів.

Основи підходу до тестування API :

– розуміння функціональність API програми і ясне визначення обсягу охоплення програми;

– застосування методів тестування (класи еквівалентності, аналіз крайового значення і помилки вгадування, написання тестів для API);

– вхідні параметри для API повинні бути правильно сплановані та визначені ; виконання тест приклади І порівняння очікуваний І фактичний

результати.

API та модульне тестування представлено в таблиці 3.3[41].

Таблиця 3.3 – Порівняльна таблиця модульного та API тестування

Модульне тестування	Тестування API
Зазвичай виконують розробники	Зазвичай виконують тестувальники
Тестуються окремі функції	Тестується вся функціональність системи
Розробник може звертатися до вихідного коду	Тестувальник не може звертатися до вихідного коду
Включене тестування графічного інтерфейсу	Тестуються лише API функції
Тестуються лише базові функції	Тестуються усі функціональні питання
Обмежене застосування	Широка сфера застосування
Зазвичай проходить до початку роботи	Проходить після створення побудови

Оскільки як API , так і модульне тестування націлені на вихідний код, такі інструменти тестування можна використовувати для тестування обох:

– SOAPUI;

– Runscope;

– Postman with jetpacks;

– Postman with Newman;

– PHPUnit+Curl;

– Cfix;

– Check;

– CTESK;

– dotTEST;

– Eclipse SDK tool.

Отже, під час проведеного аналізу були розглянуті найпопулярніші на сьогодні інструменти тестування веб-додатків. Досліджено їх продуктивність на прикладі роботи з реальними даними в існуючій системі, виміряно швидкодію і характеристики, побудовано графіки залежностей характеристик. Аналізуючи графіки, зроблені висновки про слабкі і сильні сторони методів і зроблено висновок про їх ефективність. Розглянуто перспективи та можливості використання технологій безперервної інтеграції. Вибрано конкретні інструменти та сервіси, необхідні для програмної системи для тестування веб-додатків.

ВИСНОВКИ

При виконанні кваліфікаційної робити було проаналізовано діючі міжнародні стандарти та рекомендовані протоколи для тестування веб-додатків в галузі інформаційної безпеки інформації. Розглянули питання важливості захисту веб-ресурсів. Проведено класифікацію вразливостей та загроз веб-додаткам. Також в роботі надано характеристику основним методам виявлення вразливостей веб-додатків від загроз, які можуть використовувати зловмисники.

В роботі було проведено дослідження типів атак на веб-ресурси, їх наслідки та можливі шлхи перекриття атак. Серед багатьох можливих засобів та методів перекриття загроз визначено важливість прведення тестування – автоматизованого та ручного.

З проведеного аналізу засобів автоматичного тестування захищеності веб-додатків визначено можливість та необхідність їх комбінування при застосування в реальних системах та веб-додатках. Визначено важливість розуміння масштабів та наслідків можливх загроз та необхідність їх перекриття.

В роботі розглянуто інструментальні засоби тестування захищеності веб-додатків та ефективність їх застосування при застосуванні в реальних інформаційних системах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Totallink. Терміни та поняття. [Електронний ресурс]: А. О. Береца. – 2012. – Режим доступу : http://totallink.cv.ua/%D1%82%D0%B5%D1%80%D0%BC%D1%96%D0%BD%D0%B8%D1%82%D0%B0%D0%BF%D0%BE%D0%BD%D1%8F%D1%82%D1%82%D1%8F/

2. Websecurity. Веб безпека. [Електронний ресурс]: Р. І. Ромашин. – 2015. – Режим доступу : http://websecurity.com.ua/security/chapter1/

3. Що таке вебдодатки та як вони функціонують? [Електронний ресурс]:– 2023. – Режим доступу : https://asabix.com.ua/what-are-web-applications-and-how-do-they-function/

4. AJAX: [Електронний ресурс] // Вікіпедія – вільна енциклопедія. – Режим доступу : https://uk.wikipedia.org/wiki/AJAX

5. JavaScript: [Електронний ресурс] // Вікіпедія – вільна енциклопедія. – Режим доступу : https://uk.wikipedia.org/wiki/JavaScript

6. DHTML: [Електронний ресурс] // Вікіпедія – вільна енциклопедія. – Режим доступу : https://uk.wikipedia.org/wiki/DHTML

7. XMLHttpRequest: [Електронний ресурс] // Вікіпедія – вільна енциклопедія. – Режим доступу : https://uk.wikipedia.org/wiki/XMLHttpRequest

8. Правильна структура веб сайту під SEO: приклади, види та 15+ рекомендації щодо розробки структури. [Електронний ресурс]: І. Кобилянський. – 2023. – Режим доступу : https://impulse-design.com.ua/ua/pravilnaya-struktura-veb-sajta-pod-seo.html

9. IEEE Guide to Software Engineering Body of Knowledge (SWEBOK) / IEEE Computer Society. – IEEE Computer Society Press, 2004. – 204p.

10. ISO 8402:1994 Quality management and quality assurance. – International Organization for Standardization, 1994. – 39p.

11. Савін, Р. Тестування Дот Ком, або Посібник із жорсткого поводження з багами в інтернет-стартапах. [Електронний ресурс]: – https://spacelab.ua/literature/testirovanie-dot-kom-ili-posobie-po-zhestkomu-obrasheniu-s-bagami-v-internet-startapah/.

12. DOU. Тестування. Фундаментальная теорія. [Електронний ресурс]: Gennadii Mishchevskii. – 2015. – Режим доступу : https://dou.ua/forums/topic/13389/

13. Тестування програмного забезпечення. [Електронний ресурс] // Вікіпедія – вільна енциклопедія. – Режим доступу : https://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F

14. OWASP Top Ten. [Електрон. ресурс]: – Режим доступу: https://owasp.org/www-project-top-ten/ — Топ 10 загроз.

15. Website Security Statistics Report: 2018. — WhiteHat Security, 2018. — 30 p. Режим доступу: https://info.whitehatsec.com/Website-Stats-Report-2018.html.

16. Керування конфігурацією. [Електронний ресурс] // Вікіпедія – вільна енциклопедія. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%9A%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%BA%D0%BE%D0%BD%D1%84%D1%96%D0%B3%D1%83%D1%80%D0%B0%D1%86%D1%96%D1%94%D1%8E

17. Website Security Statistics Report: 2015. — WhiteHat Security, 2015. — 30 p. Режим доступу в Інтернет: https://info.whitehatsec.com/Website-Stats-Report- 2015.html.

18. Управління конфігураціями в інформаційній безпеці. [Електронний ресурс]: Маркус Єгелька. – 2024. – Режим доступу : https://www.dqsglobal.com/uk-ua/navchajtesya/blog/upravlinnya-konfiguraciyami-v-informacijnij-bezpeci

19. 20+ найкращих інструментів і програмного забезпечення для CI/CD на 2024 рік. [Електронний ресурс]. – 2023. – Режим доступу : https://visuresolutions.com/uk/alm-guide/best-ci-cd-tools-for-2023/

20. Comparison of continuous integration software. [Електронний ресурс] // Вікіпедія – вільна енциклопедія. – Режим доступу : https://en.wikipedia.org/wiki/Comparison_of_continuous_integration_software

21. DevColibri. Тестування за допомогою TestNG у Java. [Електронний ресурс]: А.Р. Барчук. – 2013. – Режим доступу : http://devcolibri.com/1528

22. WebForMyself. Тестування коду з PHPUnit. [Електронний ресурс]: О.В. Тотко. – 2014. – Режим доступу : https://webformyself.com/testirovanie- koda-s-phpunit/

23. Selenium / WebDriver. [Електронний ресурс]. – Режим доступу : https://www.selenium.dev/documentation/webdriver/

24. Що таке Selenium? [Електронний ресурс]:. – 2024. – Режим доступу : http https://qagroup.com.ua/publications/shcho-take-selenium/

25. PVS-Studio. TestComplete. [Електронний ресурс]: Р.В. Воронов. – 2014. – Режим доступу : http://www.viva64.com/ru/t/0089/

26. HP QuickTest Professional. [Електронний ресурс] // Вікіпедія – вільна енциклопедія. – Режим доступу: https://en.wikipedia.org/wiki/HP_QuickTest_Professional

27. List of Testing Tools. [Електронний ресурс] // guru99 – professional courses. – Режим доступу : http://www.guru99.com/list-of-testing-tools.html

28. Test Tools [Електронний ресурс] // Ministry Of Testing – co-creating smarter testers. – Режим доступу: http://www.ministryoftesting.com/resources/softwaretesting-tools/

29. 10 REGRESSION/FUNCTIONAL WEB TESTING TOOLS [Електронний ресурс] // TECH BLOG. – Режим доступу: http://www.hurricanesoftwares.com/10-regressionfunctional- web-testing-tools/

30. 5 Best Test Automation Tools [Електронний ресурс] // automated-360 blog . – Режим доступу : http://automated- 360.com/automation-tools/5-best-test-

automation-tools

31. ISO/IEC 17799:2005. Information technology. Security techniques. Code of practice for information security management.

32. Category:Software testing tools [Електронний ресурс] // Вікіпедія – вільна енциклопедія. – Режим доступ : https://en.wikipedia.org/wiki/Category:Software_testing_tools

33. What is Selenium? [Електронний ресурс] // Selenium HQ. – Режим доступу: http://docs.seleniumhq.org/

34. HP Unified Functional Testing software [Електронний ресурс] // Web-site of Hewlett-Packard Company. – Режим доступу : http://www8.hp.com/h20195/V2/GetPDF.aspx/4AA4- 8360ENW.pdf

35. Test Studio [Електронний ресурс] // Web-site of Telerik. – Режим доступу : http://www.telerik.com/teststudio

36. TestComplete Platform: Testing for Desktop, Mobile, & Web Applications [Електронний ресурс] // Web-site of Smartbear. – Режим доступу : http://smartbear.com/product/testcomplete/overview/

37. Automated Testing of Desktop. Web. Mobile. [Електронний ресурс] // Ranorex Company. – Режим доступу : http://www.ranorex.com/

38. Skybot Job Scheduler [Електронний ресурс] // HelpSystems issues. – Режим доступу : http://www.helpsystems.com/skybot/products/skybot-scheduler

39. Хабрахабр . Descriptive Programming в QuickTest Pro. [Електронний ресурс]: – 2009. – Режим доступу : https://habr.com/en/articles/69138/

40. Bugs Catcher. Telerik WebUI Test Studio. [Електронний ресурс]: В.Н. Лувин. – 2011. – Режим доступу : http://bugscatcher.net/archives/817

41. Coded UI and Ranorex in Visual Studio. [Електронний ресурс]:– 2018. – Режим доступу : https://www.ranorex.info/coded-ui-and-ranorex-in-visual-studio-t11800.html

42. Wiki. Ranorex Studio [Електронний ресурс] – 2023. - Режим доступу : https://en.wikipedia.org/wiki/Ranorex_Studio

43. API testing. [Електронний ресурс] // guru99 – professional courses. –

Режим доступу : http://www.guru99.com/api-testing.html

44. ISO/IEC 27001:2005. Information technology. Security techniques. Information security management systems. Requirements.

45. ISO/IEC TR 27035:2011. Information technology − Security techniques − Information security incident management.

46. Defining Incident Management Processes for CSIRTs: A Work in Progress // CMU/SEI-2004-TR-015: ESC-TR-2004-015 Chris Alberts, Audrey Dorofee, Georgia Killcrece October 2004 Networked Systems Survivability Program.

ДОДАТОК А. ЗАВДАННЯ ДО КВАЛІФІКАЦІНОЇ РОБОТИ БАКАЛАВРА

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук
Кафедра безпеки інформаційних систем і технологій
Рівень вищої освіти (освітньо-кваліфікаційний рівень) бакалавр
Галузь знань: 12 Інформаційні технології
Спеціальність: 125 «Кібербезпека»

«Затверджую»
В.о. зав. кафедри безпеки інформаційної
систем і технологій, к.т.н., доцент
_____ Ольга МЕЛКОЗЬОРОВА
«___» _____ 2024 р

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

_____АВДЄЄНКА Миколи Андрійовича_____
(прізвище, ім'я, по батькові)

1. Тема роботи _«Аналіз інструментальних засобів тестування вразливостей веб-додатків»,_____
Керівник роботи Колованова Євгенія Павлівна, к.т.н., доцент зво кафедри безпеки інформаційної систем і технологій
затверджені наказом по університету від __ _____2024 р. № _____

2. Строк подання студентом роботи _31 травня_2024 р.

3. Перелік питань, які потрібно розробити

- проаналізувати діючі міжнародні стандарти, рекомендовані протоколи та практики у галузі інформаційної безпеки;

- розглянути питання захисту веб-ресурсів, основних принципів безпеки веб-додатків, загальних загроз та вразливостей;

- проаналізувати актуальні загрози веб-ресурсів, надати характеристику методів виявлення вразливостей, які можуть бути застосовані зловмисниками;

- дослідити типи атак на веб-ресурси та їх наслідки;

- проаналізувати методи та засоби захисту веб-додатків;

- провести дослідження методів експлуатації вразливостей для оцінки рівня захищеності веб-ресурсів;

- сформулювати рекомендації щодо застосування інструментів тестування вразливостей веб-додатків.

4. План роботи

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Проведення огляду наукових джерел та аналіз актуальних тенденцій у галузі тестування вразливостей веб-додатків	01.12.2023-25.01.2024	Виконано
2	Формулювання об’єкту, предмету, мети та завдань роботи	20.12.2023-10.01.2024	Виконано
3	Дослідження методів створення веб-додатків	10.01.2024-02.02.2024	Виконано
4	Аналіз безпеки веб-додатків	03.02.2024-15.03.2024	Виконано
5	Вивчення та аналіз методів та засобів тестування веб-додатків	16.03.2024-30.04.2024	Виконано
6	Оформлення пояснювальної записки	31.03.2024-27.05.2024	Виконано
7	Підготовка супровідних документів та презентації до захисту кваліфікаційної роботи	30.05.2024-27.05.2024	Виконано

5. Дата видачі завдання 01.12.2024 р.

Студент М. А. Авдєєнко _____
Ініціали, прізвище підпис

Керівник роботи Є.П. Колованова _____
Ініціали, прізвище підпис