Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(освітній рівень)

на тему: *"Інформаційна система для автоматизованого виявлення вразливостей у вебдодатках"*

Виконав: студент VI курсу, групи СБм-62

Спеціальності:
125 «Кібербезпека»
(шифр і назва напряму підготовки, спеціальності)

Гарматій Р. В.
підпис (прізвище та ініціали)

Керівник *Оробчук О. Р.*
підпис (прізвище та ініціали)

Нормоконтроль *Лечаченко Т. А.*
підпис (прізвище та ініціали)

Завідувач кафедри *Загородна Н.В.*
підпис (прізвище та ініціали)

Рецензент *Боднарчук І.О.*
підпис (прізвище та ініціали)

м. Тернопіль – 2023

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В.
(підпис) (прізвище та ініціали)
«__» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека
(шифр і назва спеціальності)

Студенту Гарматій Роман Васильович
(прізвище, ім'я, по батькові)

1. Тема роботи Інформаційна система для автоматизованого виявлення вразливостей у вебдодатках

Керівник роботи Оробчук О. Р., доктор філософії, старший викладач кафедри КБ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 16 » листопада 2023 року № 4/7-1061

2. Термін подання студентом завершеної роботи 12.12.2023

3. Вихідні дані до роботи Інформаційна система

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналітична частина 1.1 Поточний стан вебдодатків 1.2 Тестування як спосіб пошуку вразливостей у веб-додатках 1.3 Сучасний стан кібербезпеки у вебдодатках. 1.4 Аналіз відомих засобів вирішення проблеми 1.5. Функціональність програмної системи 2 Теоретична частина 2.1. Огляд вразливостей у сучасних веб-додатках 2.2 Порушення безпеки додатку атакою SQL injection 2.3 Вразливість розкриття конфіденційних даних 2.5. Теоретичні відомості про системний аналіз 2.6 Дерево цілей 2.7 Діаграма потоків даних 2.8 Побудова ієрархії задач 2.9 Модель «сутність-зв'язок» 3 Практична частина 3.1. Вибір та обґрунтування засобів розв'язання задачі 3.2. Загальні відомості про програму 3.3. Інструкція користувача 4. Охорона праці та безпека в надзвичайних ситуаціях. 4.1 Охорона праці. 4.2 Організація оповіщення і зв'язку у надзвичайних ситуаціях техногенного та природного характеру. Висновки. Список використаних джерел. Додатки.

5. Перелік графічного матеріалу. 1. Титулка. 2. Мета та задачі дослідження 3. Дерево цілей 4. Контекстна діаграма інформаційної системи 5. Діаграма потоків даних 6. ER-діаграма інформаційної системи

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Осухівська Г.М., к.т.н., доцент		
Безпека в надзвичайних ситуаціях	Клепчик В.М., проректор з адміністративно-господарської роботи та будівництва		

7. Дата видачі завдання 16.11.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	14.11.2023-15.11.2023	*Виконано*
2.	Підбір наукових джерел	16.11.2023-20.11.2023	*Виконано*
3.	Переклад та опрацювання наукових джерел	21.11.2023-23.11.2023	*Виконано*
4.	Розробка інформаційної системи	24.11.2023-27.11.2023	*Виконано*
5.	Оформлення розділу “Аналітична частина ”	28.11.2023-30.11.2023	*Виконано*
6.	Оформлення розділу “Теоретична частина”	01.12.2023-04.12.2023	*Виконано*
7.	Оформлення розділу “Практична частина”	05.12.2023-07.12.2023	*Виконано*
8.	Виконання завдання до підрозділу «Охорона праці»	08.12.2023-09.12.2023	*Виконано*
9.	Виконання завдання до підрозділу «Безпека в надзвичайних ситуаціях»	10.12.2023-11.12.2023	*Виконано*
10.	Оформлення кваліфікаційної роботи	12.12.2023-13.12.2023	*Виконано*
11.	Нормоконтроль	14.12.2023-15.12.2023	*Виконано*
12.	Перевірка на плагіат	9.12.2023	*Виконано*
13.	Попередній захист кваліфікаційної роботи	16.12.2023	*Виконано*
14.	Захист кваліфікаційної роботи	27.12.2023	

Студент _____ *Гарматій Р. В.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Оробчук О. Р.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Інформаційна системи для автоматизованого виявлення вразливостей у вебдодатках // Кваліфікаційна робота освітнього рівня «Магістр» // Гарматій Роман Васильович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБм-62 // Тернопіль, 2023 // С. – 84, рис. – 20, кресл. – 6 табл. – 9, додат. – 2.

Ключові слова: ВРАЗЛИВІСТЬ, ТЕСТУВАННЯ, ВЕБ-ДОДАТОК, ІНФОРМАЦІЙНА СИСТЕМА, ТЕСТОВИЙ ВИПАДОК, АВТОМАТИЗАЦІЯ ТЕСТУВАННЯ.

У даній кваліфікаційній роботі досліджується проблема ефективного виявлення вразливостей у веб-додатках з використанням інформаційної системи. Основний акцент роботи спрямований на розробку та імплементацію системи, яка автоматизовано виявляє потенційні загрози безпеки в контексті сучасних веб-технологій. У роботі аналізуються ключові переваги використання інформаційних систем для виявлення вразливостей у вебдодатках та проводиться детальне вивчення методів виявлення та аудиту безпеки на основі трафіку.

Реалізована інформаційна система піддається порівняльному аналізу з використанням критеріїв ефективності, доступності та вартості, зокрема, для наведення обґрунтованого вибору в контексті кібербезпеки вебдодатків.

ANNOTATION

Information System for Automated Vulnerability Detection in Web Applications // Harmatii Roman Vasylovich // Ternopil Ivan Pulyuy National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Cybersecurity, СБм-62 group // Ternopil, 2023 // P. – 83, fig. – 20, drawing – 6 table – 9, apendix – 2.

Key words: VULNORABILITY, TESTING, WEB APPLICATION, INFORMATION SYSTEM, TEST CASE, TEST AUTOMATION.

This qualification paper examines the problem of effective detection of vulnerabilities in web applications using an information system. The main emphasis of the work is aimed at the development and implementation of a system that automatically detects existing security threats in the context of modern web technologies. The paper analyzes the key benefits of using an information system to detect vulnerabilities in web applications and provides a detailed study of traffic-based security detection and auditing methods.

The implemented information system is subjected to a comparative analysis using the criteria of efficiency, availability and cost, in particular, to guide a justified choice in the context of cyber security of web applications.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API	–	Application Programming Interface
B2B	–	Business to Business
CI/CD	–	Continuous integration / Continuous delivery
CRUD	–	Create, read, update, delete
HTTPS	–	Hyper Text Transfer Protocol Secure
IT	–	Information Technology
OWASP	–	Open Web Application Security Project
REST	–	Representational State Transfer
RPC	–	Remote procedure call
SDK	–	Software development kit
SQS	–	Simple Queue Service
TLS	–	Transport Layer Security
ОС	–	Операційна система

ВСТУП

Актуальність теми

Інтернет є основоположним відкриттям у сучасному суспільстві, яке спонукає до трансформаційних змін у різних аспектах повсякденного життя, політики, стосунків, поширення новин, науки, навчання та розваг. Він здійснив революцію в системах зв'язку, міцно утвердившись як переважаючий і широко поширений засіб взаємодії.

Однак, як і будь-яке нововведення, воно має свої недоліки та складності, які вимагають ретельного розгляду. З широким розповсюдженням щоденного використання Інтернету через мережу постійно протікає значна кількість даних, у тому числі конфіденційної інформації. Незважаючи на те, що Інтернет є переважно безпечним каналом для передачі інформації, Інтернет створює потенційні ризики. Постійний розвиток технологій сприяє зростанню загрози кіберзлочинності. Примітно, що кібератаки стали значним фактором, який спричинив значні збитки для багатьох компаній. Лише за минулий рік 53 відсотки кібератак призвели до фінансових збитків, що перевищують 500 000 доларів США.

Самі кібератаки можуть приймати різні форми, наприклад, зловмисне відключення комп'ютера, викрадення даних або використання скомпрометованого комп'ютера як стартової точки для подальших атак. Кіберзлочинці використовують різні методи для здійснення цих атак, зокрема зловмисне програмне забезпечення, фішинг, програми-вимагачі, відмова в обслуговуванні та інші. На жаль, кіберзлочинність продовжує зростати щороку, і люди постійно шукають нові способи використання вразливостей у бізнес-системах.

Мета і задачі дослідження

Метою магістерської кваліфікаційної роботи є проектування та розробка інформаційної для автоматизованого виявлення вразливостей у вебдодатках.

Засоби досліджень:

– метод дерево цілей

– структурна методологія у вигляді ієрархії DFD

– метод аналізу ієрархій

Для досягнення поставленої мети необхідно виконати наступні задачі:

1) Визначити мету та актуальність заданої теми.

2) Провести аналіз предметної області та порівняння з відповідними аналогами.

3) Сформулювати вимоги до створюваної системи.

4) Виконати проектування системи.

5) Здійснити вибір засобів реалізації системи та виконати програмну реалізацію.

Об’єкт дослідження – процеси побудови інформаційної системи для для автоматизованого виявлення вразливостей у вебдодатках.

Предмет дослідження – методи і засоби проектування та побудови інформаційної системи для автоматизованого виявлення вразливостей у вебдодатках.

Наукова новизна одержаних результатів кваліфікаційної роботи полягає у впровадженні передових методів та технологій в області кібербезпеки для створення ефективного та автоматизованого інструменту виявлення потенційних загроз безпеці вебдодатків.

Практичне значення одержаних результатів. Інформаційна система підвищенню безпеки веб-додатків, економії ресурсів та часу, а також просуває принцип проактивного підходу до кібербезпеки, сприяючи загальному покращенню захисту від кіберзагроз у цифровому середовищі.

Структура й обсяг кваліфікаційної роботи. Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку літератури із 24 найменувань та 2 додатків. Загальний обсяг кваліфікаційної роботи складає 89 сторінки, з них 83 сторінок основного тексту, який містить 20 рисунки та 9 таблиць.

1 АНАЛІТИЧНА ЧАСТИНА

1.1 Поточний стан вебдодатків

На початкових етапах розвитку Інтернету поняття веб-додатків не існувало; замість цього люди зосереджувалися на створенні статичних веб-сайтів, які не потребували веб-програмування чи дизайну баз даних. Хоча колись статичні веб-сайти вважалися стандартом, розвиток Інтернету зробив їх непрактичними для широкого використання. Керування безліччю статичних сторінок стало громіздким завданням, що спонукало розробників шукати ефективніші підходи. Це призвело до появи веб-додатків.

Веб-програми, по суті комп'ютерні програми, використовують такі онлайн-технології, як браузери, для виконання широкого спектру завдань. Нині вони зазвичай використовуються для різних цілей, починаючи від онлайн-роздрібних транзакцій до замовлень їжі та фінансової діяльності. Складність веб-додатків може варіюватися від простих контактних форм або онлайн-калькуляторів до складніших систем, які зберігають і отримують інформацію за допомогою серверних мов програмування та баз даних. Інтерфейс користувача може приймати різні форми.

Переваги веб-додатків численні. Вони сприяють зниженню витрат як для компаній, так і для окремих користувачів завдяки нижчим вимогам до технічного обслуговування та зменшенню вимог до обчислювальної потужності пристроїв користувачів. Веб-додатки можуть працювати в будь-якому веб-браузері, покращуючи доступність для всіх користувачів. Моделі на основі передплати, такі як програмне забезпечення як послуга (SAAS), борються з піратством програмного забезпечення, дозволяючи доступ лише через хмару, усуваючи необхідність встановлення та оновлення на локальних жорстких дисках.

У сучасному ландшафті веб-розробки веб-програми здобули величезну популярність, і підприємства більше не можуть досягти оптимального зростання без них. Веб-додатки відіграють ключову роль у брендингу, створюючи ефективні канали

зв’язку між компаніями та потенційними клієнтами. Вони покращують взаємодію з користувачами, сприяють залученню, підвищують коефіцієнт конверсії, утримання користувачів і отримання доходу.

У той час як статичні сайти зберігаються через свою економічну ефективність і швидкий процес розробки, веб-додатки швидко витісняють їх як переважне рішення для демонстрації бізнес-пропозицій. Для великих компаній і брендів веб-додатки стали незамінними, символізуючи зміну технологічної парадигми та пропонуючи динамічні, захоплюючі платформи для зв’язку з користувачами та клієнтами.

1.2 Тестування як спосіб пошуку вразливостей у веб-додатках

Тест – це методична спроба визначити, чи відповідають властивості чи характеристики речі, особи чи гіпотези очікуванням. Тест відрізняється від експерименту тим, що в тесті є очікування, яке потрібно довести або спростувати, тоді як в експерименті результат відкритий або про нього можна лише здогадуватися.

Тестування – це перевірка, дослідження об’єкта чи ситуації, наприклад, на правильність чи функціональність [1].

Тестування програмного забезпечення існує доти, доки не досягнута повна впевненість в відсутності помилок у програмному продукті. Загальний підхід до створення програмних продуктів визначається у тому, щоб визначити, коли і як проводиться тестування. Наприклад, у поетапних процесах більшість подій тестування відбувається після того, як вимоги до системи були визначені та реалізовані в програмах, які можна перевірити. На відміну від Agile підходу, вимоги, тестування під час програмування часто відбуваються одночасно.

У різних сферах технологій і бізнесу використовуються різні визначення. Одне з найбільш поширених трактувань процесу тестування – це перевірка наскільки об’єкт тестування відповідає вимогам.

Інколи під тестом розуміють також будь-яку перевірку, на приклад, чи поводиться об’єкт тестування, як очікувалося, за умов, які є максимально

реалістичними. Чим реалістичніші умови тестування, тим змістовнішим є тест. Не завжди можливо провести перевірку в реальних умовах, тому умови необхідно моделювати якомога реалістичніше. Хоча негативний тест є доказом наявності помилок, повна відсутність помилок часто ще не може бути доведена позитивним тестом.

Інколи під час процесу тестування проводять статистичний тест. Достовірність гіпотез перевіряється за допомогою випадкових вибірок.

При діагностиці люди, тварини і рослини можуть бути перевірені на наявність хвороб або їх збудників, наприклад за допомогою аналізу крові. Позитивний результат у сенсі тесту означає, що підозрювана причина була підтверджена, наприклад, шляхом виявлення збудника хвороби.

Тестування програмного забезпечення включає виконання системних компонентів або програмних компонентів для оцінки однієї чи кількох пов’язаних функцій. Як правило, ці характеристики вказують на систему або внутрішні компоненти, що перевіряються:

- Відповідність вимогам, що стосуються розробки та проектування
- Правильна реакція на всі пов’язані введення
- Виконання функцій у прийнятний час
- Достатня зручність використання.
- Завантажте та встановіть на платформах для запуску
- Досягнення загальних результатів, яких бажають зацікавлені сторони.

Існує практично нескінченна кількість можливих тестів для кожного компонента програмного забезпечення, всі методи тестування програмного забезпечення використовують певну стратегію для вибору відповідних тестів для доступного часу та ресурсів. У результаті тестування програмного забезпечення зазвичай (але не виключно) намагається виконати програму або програму, яка намагається знайти помилки програмного забезпечення. Тестування програмного забезпечення надає об’єктивно незалежну інформацію про якість програмного забезпечення та ризик збою його користувачів або спонсорів.

Якість програмного забезпечення – це рівень його відповідності до попередньо зазначених вимог, здатність задовольняти потреби користувача.

Верифікація – це процес ознайомлення з станом системи, чи співпадає стан програмного забезпечення з попередньо визначеними критеріями, які формуються перед початком розробки системи. Під час перевірки враховуються цілі кожного етапу розробки та терміни, в які вони мають бути виконаними.

Тест план – це частина документації програмного забезпечення, в якому вказаний порядок тестових сценаріїв, які мають бути виконаними під час тестування. Тест план часто включає опису предметної галузі тестованого продукту, стратегії його розвитку, умови для початку та завершення тестування, та терміни виконання тестового плану [1].

Тест дизайн – це перший етап у процесі тестування програмного забезпечення. Під час цього етапу проектуються та створюються тестові випадки, визначається тест план, зважаючи на необхідний рівень якості програмного продукту.

Еквівалентний поділ – нехай, існує діапазон значень від 10 до 100, спеціаліст із тестування програмного забезпечення повинен вибрати одне коректне значення із доступних чисел цього діапазону, нехай, 60, і одне неправильне значення, яке не входить до діапазону – 5. Після чого повинен перевірити реакцію тестованого програмного забезпечення на введення таких даних.

Аналіз межових значень. Із діапазону із 10 до 100 необхідно вибрати найменше можливе значення – 10 та найбільше можливе – 100, і значення, які виходять за межі діапазону – 9 і 101. Після чого передати ці дані до інформаційної системи, що тестується. Аналіз межових значень також застосовується під час тестування веб-додатків для аналізу полів введення даних.

Причинно-наслідковий спосіб – це спосіб тестування програмного забезпечення, під час якого до інформаційно системи передається множина умов(причин) та досліджується відповідь системи на цей запит, яка є наслідком передачі умов. Якщо спеціаліст із тестування тестує функціонал інформаційної системи, який полягає у додаванні запису на сторінку, використовуючи деяку

сторінкову форму. Для цього потрібно надати коректні дані для полів «Назва запису» та «Вміст запису» а потім, натиснути кнопку «Надіслати» – це натискання кнопки у даному випадку є причиною. Після натискання кнопки «Надіслати» програмне забезпечення перевіряє введені дані на коректність, додає їх до бази даних та відображає повідомлення про додавання запису. Спеціаліст із тестування програмного забезпечення у даному випадку повинен перевірити, чи отриманий наслідок відповідає причині [2].

Спосіб передбачення помилки – це коли спеціаліст із тестування програмного забезпечення використовує свої знання системи, яка тестується та свою можливість «передбачити» можливі помилки, ґрунтуючись на особистому досвіді тестування подібних систем, оскільки часто при розробці програмного забезпечення допускаються однакові помилки, не допустити які може досвідчений тестувальник. Наприклад, на етапі проектування система було передбачено дію: "користувач повинен ввести пароль". Для передбачення помилки під час виконання цієї дії, спеціаліст із тестування програмного забезпечення повинен провести дослідження наступних питань: «Результат роботи системи, якщо не ввести пароль», «Результат роботи системи, якщо ввести неправильний пароль», тощо. Результатом цього дослідження повинен бути список можливих помилок, тобто їх передбачення.

Вичерпне тестування – це застарілий та складний спосіб тестування програмного забезпечення. Він означає, що будуть перевірені усі дані, які можуть бути введені для надсилання запиту у інформаційну систему, що знайде усі можливі помилки у програмному забезпеченні, що тестується. Цей метод застосовується рідко, оскільки є надзвичайно неефективним, порівняно з іншими технологіями.

Попарне тестування – це техніка формування наборів тестових даних. Під час використання цього методу тестування необхідно створити пари тестових даних, обов’язковою умовою при цьому є те, що кожне з значень, які тестуються мінімум один раз поєднується з параметрами інших тестових даних [2].

Тестовий випадок – це частина тестової документації, яка описує список кроків, умов та параметрів, які потрібно виконати для перевірки коректності роботи функціоналу, який тестується.

Кожен тест кейс повинен мати 3 частини:

1) Перелік дій, що після яких система переходить у стан, який є придатним для виконання тестування. Або перелік умов, виконання яких свідчить, що система перебуває у придатному щодо основного тесту стану.

2) Дії, яка змінюють стан системи в процесі досягнення результату, на якому можна стверджувати, що функціонал, який тестується повністю відповідає вимогам, які були визначені на початковому етапі розробки програмного забезпечення.

Чек-лист – це частина тестової документації, у які вказано набір функціоналу програмного забезпечення, який повинен бути протестовний. Чек-лист складається у довільній формі, залежно від вимог і потреб конкретної інформаційної системи. Об’єм та детальність чек-листа визначається окремо на етапі проектування інформаційної системи.

Зазвичай чек-лист включає в себе тільки список кроків, які необхідно виконати, без очікуваного результату після проведення цих дій. Чек-лист має менше формальних вимог, ніж тестовий випадок. Найдоречніше використовувати чек-листи під час розробки невеликих інформаційних систем. Коли звичайні тестові сценарії будуть занадто ресурсозатратними.

Дефект – це не співпадіння очікуваної реакції системи після проведення тестування і реального результату. Вони визначаються на фінальному етапі тестування програмного забезпечення, коли спеціаліст із тестування проводить порівняння результатів проведеного тестування із очікуваними.

Баг репорт – це частина тестової документації, яка містить опис ситуації або послідовність кроків, після виконання яких система повела себе не так, як цього очікувалось.

Серйозність – це атрибут, який визначає ступінь негативного впливу дефекту на загальну функціональність системи.

Пріоритет – це атрибут, який вказує на черговість виконання задач, які пов’язані зі виправленням дефекту. Його використовують менеджери, як інструмент оптимізації роботи розробників.

Ручне тестування – це процес ручного програмного забезпечення для виявлення недоліків, яку проводиться безпосередньо спеціалістом із тестування. Це вимагає, щоб тестувальник виконував роль кінцевого користувача, при цьому він або вона застосовує увесь набір функцій програмного продукту. Щоб забезпечити адекватність тесту, спеціалісти із тестування зазвичай дотримуються письмового плану тестування, який веде їх до ключових тестових випадків [3].

Таким чином, ручне тестування є найпримітивнішою формою тестування програмного забезпечення. Воно використовується для пошуку помилок у програмах та програмних системах.

Перевага в тому, що для цих тестів не потрібне спеціальне тестове програмне забезпечення, оскільки ми ним не користуємося.

Фактично, ми можемо виконати будь-який тип тестування програмного забезпечення як вручну, так і за допомогою засобів автоматизації. Наприклад, подумайте про тестування модулів, інтеграційне тестування, тестування системи та приймальне тестування. Зацікавлені сторони програми, такі як розробники, розробники, менеджери додатків, функціональні менеджери та кінцеві користувачі, часто також є тестувальниками.

Один із принципів тестування полягає в тому, що 100% автоматичне тестування неможливе. Це означає, що (частково) ручне тестування залишається необхідним для всіх типів тестування програмного забезпечення.

У програмній інженерії тестовий випадок – це набір умов, що використовуються для тестування програмного забезпечення. Він може бути розроблений для виявлення дефектів у внутрішній структурі програмного забезпечення за допомогою ситуацій, які адекватно реалізують усі структури, що використовуються в кодуванні; або навіть переконатися, що вимоги створеного програмного забезпечення повністю відповідають. Ми можемо використовувати

інструмент варіантів використання для створення та відстеження тестового випадку, полегшуючи таким чином ідентифікацію можливих збоїв.

Тестовий випадок повинен вказати вхідні значення та очікувані результати обробки. Найкраще, якщо при розробці тестових випадків можна очікувати знайти підмножину можливих тестових випадків з найбільшою ймовірністю знайти найбільшу кількість помилок.

Веб-елемент – це індивідуальна сутність, що генерується на веб-сторінці. Елементи – це все те, що користувач бачить (а іноді й не бачить) на сторінці – заголовки, кнопки "ОК", поля введення, тощо. Елементи HTML визначаються через ім'я тега та атрибути. Вони можуть бути дочірніми елементами – наприклад, таблиці. CSS може застосовуватися до елементів і змінювати їх кольори, розміри та розташування. Мови програмування зазвичай отримують доступ до веб-елементів як частина в об'єктній моделі документа (DOM) [4].

Локатор веб-елемента – це об'єкт, який знаходить та повертає веб-елементи на сторінці за запитом. Інакше кажучи, локатори знаходять елементи. Локатори необхідні для того, щоб інформаційна система могла взаємодіяти з елементами під час автоматизованого тестування.

Веб-додаток – група взаємопов'язаних документів, які доступні в мережі Інтернет з однієї доменной адреси через www-сервіс. Часто документи пов'язані один з одним з метою підвищення функціональності. Веб-сайти, окрім статичного вмісту, часто мають розділ новин і пропонують можливість увійти та запам’ятати вподобання одержувачів, щоб налаштувати вміст відповідно до індивідуальних уподобань. Веб-сайт може містити інтерактивні об'єкти, наприклад, форми, програми.

Веб-сайти можуть бути тематичними, тобто присвяченими одній галузі чи проблемі, або загальними [6].

Термін веб-сайт походить від англійського виразу web site і відноситься до групи мультимедійних веб-сторінок (містять текст, нерухомі зображення, анімацію тощо), доступних в Інтернеті в принципі будь-кому, зазвичай на певну тему, і які

пов'язані між собою за допомогою так званих гіперпосилань. Різні веб-сайти можуть створювати організація, особа, державні установи тощо.

На початку Інтернету доступ до кожного веб-сайту був шляхом вказівки його конкретної цифрової адреси. Пізніше для веб-сайтів були введені доменні імена, які дозволяють зручніше вказувати адресу словами або іменами, які легко запам’ятати. Адреси веб-сайтів мають бути чітко встановленими та унікальними [5].

Сайт складається з:

– Доменне ім’я – це ім’я, яке однозначно описує певну частину Інтернету. Доменні імена призначаються та інтерпретуються ієрархічно відповідно до правил системи доменних імен (DNS).

– Сервер або веб-хостинг – це місце, яке зберігає файли сайту. Вони можуть знаходитись фізично як на локальному пристрої, який називається сервером або на віддаленому пристрої, що називається хостингом.

– CMS – система управління контентом веб-сайту. Як центральне сховище, система керування вмістом може централізовано зберігати пов’язаний вміст і має такі функції, як контроль версій.

– Контент – це будь-яка інформація, спрямована на кінцевого користувача чи аудиторію, яка розміщена на веб-сайті.

1.3 Сучасний стан кібербезпеки у вебдодатках

Незважаючи на широку популярність сучасних веб-додатків і їх значні переваги для різних підприємств, є помітний недолік, очевидний у якості їх розробки, особливо з точки зору безпеки. Ця негативна тенденція зумовлена насамперед невпинною гонитвою за швидкими та значними прибутками компаній будь-якого розміру, що змушує їх не звертати уваги на питання безпеки у своїх веб-додатках. Іншим фактором, що сприяє цій проблемі безпеки, є неадекватно реалізований код у програмах, авторами яких часто є недосвідчені розробники. З нинішнім високим попитом на веб-розробників багато новачків приходять на роботу з обмеженим

досвідом або без нього. Навіть досвідчені розробники можуть помилятися через ненавмисні помилки або недбалість.

Ця тенденція створює ризик не лише для конфіденційності даних користувачів, оскільки помилки можуть поставити під загрозу зберігання даних у веб-додатках, але також ставить під загрозу репутацію компаній, відповідальних за безпеку довірених даних. Щоб підкреслити масштаб проблеми, нижче наведено статистику атак на компанії за останні три роки. Ці тривожні цифри підкреслюють недостатню увагу багатьох компаній до безпеки. Навіть ті організації, які щиро піклуються про своїх клієнтів і безпеку даних, не захищені від пасток погано написаного коду та помилкових рішень у своїх системах захисту розробників.

У середньому кожна веб-програма містить приблизно 22 уразливості, причому близько чотирьох із них класифікуються як критичні. Хоча ці статистичні дані є приблизними, вони, безсумнівно, висвітлюють суттєву проблему з поточним станом безпеки веб-додатків, підкреслюючи вкрай важливу потребу в розробці високоякісного та, перш за все, безпечного коду.

1.4. Аналіз відомих засобів вирішення проблеми

На даний момент однією з найвідоміших систем для автоматизації перевірки коректної роботи веб-додатків є Cypress. Він спрямований на подолання перешкод, з якими стикаються інженери та розробники під час тестування веб-додатків на основі React і AngularJS. Це швидкий, простий і надійний інструмент для тестування будь-яких програм, які працюють у браузерах. Таким чином, Cypress використовується для тестування широкого кола додатків, які працюють у браузері. Однією з переваг цієї системи є те, що вона є безкоштовною. Це дозволяє створювати тестові випадки, поки веб-додаток розробляється [6].

Важливі особливості Cypress:

– Cypress надає функціональність захоплення знімків під час проведення автоматизованого тестування веб-додатків.

– Cypress забезпечує легку підтримку коду розробленої на його основі інформаційної системи.

– Cypress гарантує, що нам не потрібно додавати методи синхронізації, такі як сон і очікування. За замовчуванням інформаційна система чекає наступних дій або перевірок, перш ніж перейти до наступного кроку.

– Cypress перевіряє характеристики функцій, таймерів і відповідей на сервер. Це важливо з точки зору модульного тестування.

– Cypress за замовчуванням має можливість робити знімки екрана в разі відмови. Він також може знімати відеозаписи виконання всього набору тестів під час запуску з інтерфейсу командного рядка.

– Cypress дозволяє перехресне тестування через браузер. Він підтримує такі браузери, як Firefox та Chrome.

– Завдяки своєму архітектурному дизайну Cypress забезпечує швидке, стабільне та надійне виконання тестів у порівнянні з іншими інструментами в автоматизації.

– У Cypress є меню статусу тесту, яке відображає кількість тестів, які пройшли чи не пройшли.

– Cypress має дуже хорошу документацію, яка надає необхідну підтримку для створення інформаційних систем.

– Cypress має належне повідомлення про помилку, яке описує причину збою тестового випадку.

1.5. Функціональність програмної системи

У таблиці зображено список функцій інформаційної системи, які повинні бути доданими до неї у процесі розробки (табл. 1.1).

Таблиця 1.1 - Функціональність програмної системи

Функція	Статус	Пріоритет	Трудоєм-ність	Ризик	Стабіль-ність	Цільова версія
2	3	4	5	6	7	8
Тестування авторизація	Включена	Критична	Середня	Високий	Низька	1
Тестування функціоналу додавання зображень	Включена	Важлива	Висока	Середній	Середня	1
Тестування функціоналу зміни мови веб-сторінки	Включена	Важлива	Висока	Низький	Висока	1
Тестування можливості додавання новини	Пропонов-ано	Корисна	Низька	Низький	Висока	2

Продовження таблиці 1.1

2	3	4	5	6	7	8
Тестування можливості попереднього перегляду новини	Включена	Важлива	Середня	Середні	Середня	1
Тестування можливості додавання тегів до створеної новини	Включена	Важлива	Середня	Низький	Висока	1
Тестування коректної роботи поля джерело	Включена	Середня	Низька	Середній	Середня	1
Тестування коректної роботи поля контент	Включена	Низька	Середня	Середня	Висока	1
Тестування можливості додавання новини	Включена	Критична	Висока	Високий	Висока	1

Продовження таблиці 1.1

Тестування повідомлень, які отримує користувач	Пропонов-ано	Середній	Середня	Низький	Середня	3

Опис функцій:

– Тестування авторизація – інформаційна система перевіряє, чи коректно працює авторизація користувачів на веб сторінці.

– Тестування функціоналу додавання зображень – інформаційна система перевіряє, чи коректно працює функціонал додавання зображень на веб сторінці

– Тестування функціоналу зміни мови веб сторінки – інформаційна система перевіряє, чи коректно працює функціонал зміни мови на веб сторінці.

– Тестування можливості додавання новини – інформаційна система перевіряє, чи коректно працює функціонал додавання новин на веб сторінці.

– Тестування можливості попереднього перегляду новини – інформаційна система перевіряє, чи коректно працює функціонал попереднього перегляду новин на веб сторінці.

– Тестування можливості додавання тегів до створеної новини – інформаційна система перевіряє, чи коректно працює функціонал додавання тегів до створених новин на веб сторінці.

– Тестування коректної роботи поля джерело – інформаційна система перевіряє, чи відповідає поле джерело вимогам щодо обмеження кількості символів у даному полі.

– Тестування коректної роботи поля контент – інформаційна система перевіряє, чи відповідає поле контент вимогам щодо обмеження кількості символів у даному полі.

– Тестування можливості додавання новини – інформаційна система перевіряє, чи коректно працює функціонал додавання новини на веб сторінці.

– Тестування повідомлень, які отримує користувач – інформаційна система перевіряє, чи коректно працює функціонал відправки повідомлень користувачу.

2 ТЕОРЕТИЧНА ЧАСТИНА

2.1 Огляд вразливостей у сучасних веб-додатках

Для забезпечення розробки безпечної веб-програми вкрай важливо визначити ключові вразливості, які розробники часто не помічають під час кодування та створення програми. Ці вразливості охоплюють:

- атака structured query language (SQL) injection
- розкриття конфіденційних даних
- порушена автентифікація
- порушення контроль доступу
- cross-site scripting (XSS)
- cross-site request forgery (CSRF)
- недостатнє логування та моніторинг

У подальшому обговоренні буде розглянуто кожну з цих вразливостей, надано приклади того, як їх можна пом’якшити або уникнути в рамках програми.

2.2 Порушення безпеки додатку атакою SQL injection

SQL, або мова структурованих запитів, – це мова, призначена для керування даними, що зберігаються в реляційних базах даних. Він використовується для доступу, зміни та видалення даних у цих базах даних, що є звичайною практикою для багатьох веб-додатків. Однак ін’єкція SQL створює значну вразливість веб-безпеки, що дозволяє зловмисникам використовувати запити, зроблені програмою до бази даних. Ця вразливість може призвести до несанкціонованого доступу до конфіденційних даних, що потенційно може призвести до серйозних наслідків.

Під час атаки SQL-ін’єкції зловмисник отримує доступ до запитів додатків, дозволяючи їм переглядати дані поза межами своїх типових дозволів. Це може включати дані інших користувачів або будь-яку інформацію, доступну програмі. Крім

того, зловмисники можуть змінити або маніпулювати наданими даними, спричиняючи тривалі зміни у вмісті чи поведінці програми. У більш складних сценаріях зловмисник може впровадити код SQL, який скомпрометує базовий сервер або внутрішню інфраструктуру, або навіть ініціювати атаку типу «відмова в обслуговуванні».

Наслідки успішного впровадження SQL є далекосяжними. Будь-яка веб-програма, яка покладається на базу даних SQL, наприклад MySQL, Oracle або SQL Server, чутлива до цих атак. Будучи однією з найстаріших і найнебезпечніших уразливостей, атаки SQL-ін’єкцій призвели до помітних порушень даних, що призвело до шкоди репутації, регуляторних штрафів і довгострокових компромісів, які можуть залишитися непоміченими.

Запуск SQL-атаки передбачає виявлення вразливих полів введення для користувача на веб-сторінці чи в додатку, куди зловмисник може ввести вміст, наповнений шкідливими корисними навантаженнями. Дії зловмисника можуть відрізнятися залежно від їхніх цілей, але зазвичай передбачають маніпулювання базою даних за допомогою шкідливих команд SQL. Існує три основні категорії впровадження SQL-ін’єкції:

In-band SQL Injection: найпоширеніший і простий метод, коли зловмисник використовує той самий канал зв’язку для запуску атаки та збору результатів. SQL-ін’єкції на основі помилок і об’єднань належать до цієї категорії.

Інференційне впровадження SQL: хоча це може зайняти більше часу, цей тип є однаково небезпечним. Під час інференційної атаки дані безпосередньо не передаються через веб-програму, а зловмисник реконструює структуру бази даних, спостерігаючи за відповідями програми. Прикладами цієї категорії є SQL-ін’єкції, засновані на логіці та часі.

Важливість усунення вразливостей SQL-ін’єкції є очевидною для захисту конфіденційних даних, підтримки цілісності програми та запобігання потенційно серйозним наслідкам як для користувачів, так і для організацій.

Різні фактори сприяють розкриттю даних, зокрема зберігання даних у базах даних, чутливих до атак, як-от впровадження SQL, використання слабких криптографічних алгоритмів або ключів, нехтування методами хешування паролів або використання неадекватних сховищ даних. Крім того, людський фактор відіграє значну роль у розкритті даних, включаючи фішинг, соціальну інженерію або випадкові інциденти.

2.3 Вразливість розкриття конфіденційних даних

Розголошення конфіденційних даних відбувається, коли програма, компанія чи особа ненавмисно розкриває особисту інформацію. Це може статися різними способами, наприклад, коли розробник помилково завантажує дані в неправильну базу даних. Важливо розрізняти розголошення та порушення даних, коли несанкціонований доступ до інформації відбувається як інцидент безпеки. Хакери можуть використати витік даних для отримання особистої інформації, що призведе до компрометації особистих даних, фінансової крадіжки або незаконного продажу інформації в глибокій мережі. Випадковий або помилковий характер розкриття даних суттєво відрізняється від навмисного витоку.

Випадкове розкриття даних може бути наслідком недостатнього захисту бази даних, слабкого шифрування, помилок шифрування або вразливості програмного забезпечення. Компанії неналежним чином обробляють інформацію, наприклад, зберігають конфіденційні дані в незахищених текстових документах або відсутність захисту Secure Sockets Layer (SSL) і протоколу HTTPS (Hypertext Transfer Protocol Secure) на веб-сторінках, підвищують ризик викриття даних.

Конфіденційні дані, які підлягають розголошенню, включають номери банківських рахунків, дані кредитних карток, медичні записи, токени сеансів, номери соціального страхування, домашні адреси, номери телефонів, дату народження та інформацію про облікові записи користувачів, як-от імена користувачів і паролі.

Захист від розголошення даних передбачає впровадження надійних заходів безпеки, протоколів шифрування та усунення вразливостей, пов’язаних із людиною.

2.4 Вразливість порушення автентифікації

Порушення даних автентифікації відбувається, коли програма, компанія чи особа випадково розкривають особисту інформацію. Це може статися різними способами, наприклад розробник помилково завантажує дані в неправильну базу даних. Дуже важливо розрізняти розголошення та порушення даних, коли несанкціонований доступ до інформації є інцидентом безпеки. Хакери можуть скористатися витоком даних для отримання особистої інформації, що призведе до компрометації особистих даних, фінансової крадіжки або незаконного продажу інформації в глибокій мережі. Ненавмисний або помилковий характер розкриття даних помітно відрізняється від навмисного витоку.

Випадкове розголошення даних може виникнути через недостатню безпеку бази даних, слабке шифрування, помилки шифрування або вразливість програмного забезпечення. Компанії, які неправильно обробляють інформацію, наприклад, зберігають конфіденційні дані в незахищених текстових документах або нехтують Secure Sockets Layer (SSL) і HTTPS (Hypertext Transfer Protocol Secure) на веб-сторінках, підвищують ризик викриття даних.

Різні фактори сприяють розкриттю даних, як-от зберігання даних у вразливих базах даних, чутливих до атак, як-от впровадження SQL, використання слабких криптографічних алгоритмів або ключів, ігнорування методів хешування паролів або використання невідповідних рішень для зберігання даних. Крім того, людський фактор відіграє значну роль у витоку даних, включаючи фішинг, соціальну інженерію або випадкові інциденти.

Конфіденційна інформація, яка може бути розголошена, включає номери банківських рахунків, дані кредитних карток, медичні записи, токени сеансів, номери соціального страхування, домашні адреси, номери телефонів, дати народження та

дані облікових записів користувачів, як-от імена користувачів і паролі. Зменшення ризику розголошення даних передбачає впровадження надійних заходів безпеки, протоколів шифрування та усунення вразливостей, пов’язаних із поведінкою людей.

2.5. Теоретичні відомості про системний аналіз

Системний аналіз – дослідницький підхід у соціальних науках, який визнає поняття системи та її аналіз вирішальними для розуміння соціальних явищ [7].

Він особливо корисний для виконання складних завдань у типовому для нашого часу середовищі, яка швидко змінюється. Це офіційне та відкрите дослідження, яке підтримує дії осіб, відповідальних за рішення або курс дій у конкретній (складній) ситуації, що характеризується невизначеністю. Його мета – визначити бажану дію чи лінію поведінки шляхом розпізнавання й розгляду доступних варіантів та порівняння їх очікуваних наслідків [8].

Це вимагає чіткого визначення меж тестованої системи та її складових. Тому критикують, що це призводить до обмеженості мислення (наприклад, коли ми уявляємо організацію як систему, ми штучно створюємо стереотип її середовища, що згодом негативно впливає на наше сприйняття) [9].

Системний аналіз (модерністський, англійська школа) – в теорії міжнародних відносин він дозволяє вивчати зміни, що відбуваються в міжнародних відносинах, зміни впливу цих відносин і міжнародних систем в усі часи – минуле, сьогодення і майбутнє – його найважливіше Особливістю є те, що воно відносить до досліджуваного явища і всього процесу до широкого контексту подій і процесів, розглядає його як складне ціле. Важливою особливістю системного аналізу є те, що він представляє картину події шляхом аналізу взаємозв'язків між її компонентами. Поняття системи прирівнюється до цілого, в якому компоненти взаємопов’язані і взаємозалежні. Найважливіше дослідження в системному аналізі засноване на вивченні структури системи, а також на вивченні взаємодії між компонентами системи [10].

2.6 Дерево цілей

Дерево цілей – це орієнтований на ціль ітеративний метод зображення, який можна використовувати для визначення обсягу проектованої інформаційної системи. Це здійснюється шляхом встановлення цілей, які можна оновлювати в міру розширення знань. Метод складається з ієрархії з головною метою розробки програмного забезпечення зверху, за якою слідують підцілі, цілі проекту, результати та критерії успіху. Цілі розбиваються на різних рівнях деталей, запитуючи «чому?» і «як?». Задається питання: «Чому ми це робимо?. Це уточнюється, щоб переконатися, що цілі мають сенс, і «Як ми це робимо?», щоб зробити цілі конкретними. Оскільки метод є ітераційним процесом, питання задаються знову і знову, щоб покращити цілі проектованої інформаційної системи. Його можна використовувати як для управління проектами, програмами та портфоліо, однак увага буде зосереджена на використанні методу в контексті управління проектами [11].

Побудоване дерево цілей для створення інформаційної системи автоматизованого виявлення вразливостей у веб-додатках зображено на рис. 2.1.

Дане дерево цілей складається з трьох рівнів ієрархії цілей. Головною ціллю дерева цілей є «Створення інформаційної системи автоматизованого виявлення вразливостей у веб-додатках». Головна ціль в процесі переходу на нижчі рівні розбивається на три підцілі:

- аналіз предметної області та наявних аналогів;
- проектування системи;
- практична реалізація системи;

Рисунок 2.1 – Дерево цілей інформаційної системи автоматизованого виявлення вразливостей у вебдодатках

Виконання першої підцілі «Аналіз предметної області та наявних аналогів», яка означає ознайомлення з предметною областю, літературою про предметну область та станом вирішення проблеми на даний момент. Пошук існуючого програмного забезпечення для тестування веб-додатків та їх аналіз, порівняння переваг та недоліків існуючих рішень. Створення функціональної складової інформаційної системи проводиться базуючись на попередньому аналізі. Для досягнення цієї цілі є важливими є критерії якості та актуальності [13].

Ціль «Аналіз предметної області та наявних аналогів» можна розбити на такі три підцілі:

– «Ознайомлення з основними методами побудови подібних систем» – проведення ознайомлення з раніше створеними інформаційними системами, зі способами їх створення та підходів до їх реалізації.

– «Ознайомлення з наявними аналогами» – пошук інформаційних систем, які схожі за своєю функціональністю до інформаційної системи для автоматизованого виявлення вразливостей у вебдодатках, вивчення їх роботи, ознайомлення з наявним функціоналом. Аналіз їх переваг та недоліків, розгляд варіантів покращення цих систем.

– «Порівняння наявних аналогів та визначення функціональності проектованої системи» – збір інформації про кращі та гірші якості схожих систем, вибір оптимальних задач, які має виконувати інформаційна система, визначення функціональності системи.

Ціль «Проектування системи» означає побудову структури майбутньої інформаційної системи. Визначення головних процесів в інформаційній системі, побудова логіки для них, визначення функціоналу, конкретизація потоків даних які проходять в цій системі. Для виконання цієї цілі є важливими два критерії: швидкість та зручність проектованої системи.

Ціль «Проектування системи» можна розбити на чотири підцілі:

– «Побудова ER-діаграми» – визначення основних сутностей та побудова зв’язків між ними для відображення логіки обміну інформацією в проектованій системі.

– «Побудова діаграми потоків даних» – опис основного функціоналу системи, визначення та відображення потоків даних у системі.

– «Побудова діаграми декомпозиції» – детальніше зображення діаграми потоків даних шляхом її доповнення та розбиття процесів діаграми вищого рівня на підпроцеси.

– «Розробка функціональної схеми» – відображення взаємодії різних елементів інформаційної системи.

Ціль «Практична реалізація системи» означає повну розробку інформаційної системи. Що включає в себе вибір засобів та інструментів, використовуючи які система буде реалізована, процес розробки системи з використання обраних засобів, тестування системи на виявлення проблем.

Ціль «Практична реалізація системи» можна розбити на чотири підцілі:

– «Вибір засобів реалізації системи» – аналіз наявних засобів для реалізації системи, ознайомлення з їхніх перевагами і недоліками та вибір найбільш оптимального засобу.

– «Реалізація інформаційної системи» – розробка проектованої системи за допомогою обраних засобів реалізації.

– «Тестування системи та пошук несправностей» – пошук дефектів в роботі системи, їх аналіз та подальше виправлення. Забезпечення надійної роботи системи.

У методі аналізу ієрархій проблема добре структурована на ієрархічних рівнях, що сприяє розумінню та оцінці проблеми. Методі аналізу ієрархій – це математичний інструмент вирішення проблем. Він був створений після розуміння структури проблеми та реальних перешкод, з якими стикаються менеджери під час її вирішення.

Метод аналізу ієрархій розглядає проблему в трьох частинах. Перша частина – це проблема, яку потрібно вирішити, друга частина – альтернативні рішення, доступні для вирішення проблеми. Третя і найважливіша частина, що стосується методу аналізу ієрархій, – це критерії, які використовуються для оцінки альтернативних рішень [14].

Визначають три види інформаційних систем в залежності від завдань, які має вирішує інформаційна система, її функціоналу та рівня автоматизації процесів, визначать такі види інформаційних систем: інформаційно-пошукові, інформаційно-довідкові, інформаційно-керуючі, та інтелектуальні інформаційні системи.

Для визначення типу проектованої системи застосуємо метод аналізу ієрархій. Альтернативами оберемо вещеперелічені типи інформаційних систем:

– інформаційно-пошукова – А1;

– інформаційно-довідкова – А2;

– інформаційно-керуючі – А3;

– інтелектуальна інформаційна система – А4.

Для оцінки та вибору альтернатив визначимо 6 критеріїв:

– якість – К1;
– актуальність – К2;
– швидкість – К3;
– зручність – К4;
– надійність – К5;
– результативність – К6.

Після чого у методі аналізу ієрархій будуються матриці попарних порівнянь. На цьому етапі встановлюється відносна важливість кожного критерію до іншого шляхом їх порівняння між собою. Ці порівняння виконуються на основі експертної оцінки. Матриця попарних порівнянь наведена в таблиці 2.1.

Таблиця 2.1 - Матриця попарних порівнянь критеріїв

	К1	К2	К3	К4	К5	К6	Власні числа	Власні вектори
К1	1,00	0,17	0,33	0,20	0,25	0,50	0,33	0,05
К2	6,00	1,00	0,50	2,00	1,00	0,33	1,12	0,16
К3	3,00	2,00	1,00	2,00	0,50	0,33	1,12	0,16
К4	5,00	0,50	0,50	1,00	0,33	0,50	0,77	0,11
К5	4,00	1,00	2,00	3,00	1,00	2,00	1,91	0,28
К6	2,00	3,00	3,00	2,00	0,50	1,00	1,62	0,24

Далі було виконано парні порівняння альтернатив для кожного з критеріїв, результат наведено у табл. 2.2 – табл. 2.7.

Таблиця 2.2 - Матриця парних порівнянь альтернатив відносно критерію «Якість»

	А1	А2	А3	А4	Власні числа	Власні вектори
А1	1,00	0,50	0,33	0,25	0,45	0,09
А2	2,00	1,00	0,50	0,20	0,67	0,14

Продовження таблиці 2.2

А3	3,00	2,00	1,00	2,00	1,86	0,39
А4	4,00	5,00	0,50	1,00	1,78	0,37

Таблиця 2.3 - Матриця парних порівнянь альтернатив відносно критерію «Актуальність»

	А1	А2	А3	А4	Власні числа	Власні вектори
А1	1,00	0,50	0,25	0,33	1,00	0,10
А2	2,00	1,00	0,33	0,50	2,00	0,16
А3	4,00	3,00	1,00	2,00	4,00	0,47
А4	3,00	2,00	0,50	1,00	3,00	0,28

Таблиця 2.4 - Матрицят парних порівнянь альтернатив відносно критерію «Швидкість»

	А1	А2	А3	А4	Власні числа	Власні вектори
А1	1,00	2,00	0,50	0,33	0,76	0,17
А2	0,50	1,00	0,50	0,25	0,50	0,11
А3	2,00	2,00	1,00	2,00	1,68	0,37
А4	3,00	4,00	0,50	1,00	1,57	0,35

Таблиця 2.5 - Матриця парних порівнянь альтернатив відносно критерію «Зручність»

	А1	А2	А3	А4	Власні числа	Власні вектори
1	2	3	4	5	6	7
А1	1,00	0,50	0,33	0,20	1,00	0,17
А2	2,00	1,00	0,14	0,17	2,00	0,05
А3	3,00	7,00	1,00	3,00	3,00	0,25
А4	5,00	6,00	2,00	1,00	5,00	0,53

Таблиця 2.6 - Матриця парних порівнянь альтернатив відносно критерію «Надійність»

	A1	A2	A3	A4	Власні числа	Власні вектори
A1	1,00	0,50	0,33	0,20	0,43	0,07
A2	2,00	1,00	0,14	0,17	0,47	0,07
A3	3,00	7,00	1,00	3,00	2,82	0,43
A4	5,00	6,00	2,00	1,00	2,78	0,43

Таблиця 2.7 - Матриця парних порівнянь альтернатив відносно критерію «Результативність»

	A1	A2	A3	A4	Власні числа	Власні вектори
A1	1,00	0,50	0,33	2,00	0,76	0,16
A2	2,00	1,00	0,50	3,00	1,32	0,28
A3	3,00	2,00	1,00	0,50	1,32	0,28
A4	0,50	3,00	2,00	1,00	1,32	0,28

На основі попередніх матриць було створено узагальнену матрицю порівняння альтернатив (табл. 2.8).

Таблиця 2.8 - Матриця порівняння альтернатив

Критерії	K1	K2	K3	K4	K5	K6	Узагальнені пріоритети
	0,05	0,16	0,16	0,11	0,28	0,24	
1	2	3	4	5	6	7	8
A1	0,09	0,10	0,17	0,17	0,07	0,16	0,13
A2	0,14	0,16	0,11	0,05	0,07	0,28	0,14
A3	0,39	0,47	0,37	0,25	0,43	0,28	0,36
A4	0,37	0,28	0,35	0,53	0,43	0,28	0,37

Після проведеного аналізу узагальнену матрицю порівняння альтернатив було визначено альтернативу з найбільшим узагальненим пріоритетом, а саме альтернативу А4 що відповідає інтелектуальній інформаційній система. Отже, за

допомогою методу аналізу ієрархій було визначено, що тип розроблюваної системи за рівнем автоматизації управління є інтелектуальна інформаційна система.

2.7 Діаграма потоків даних

Діаграма потоків даних (DFD) відображає потік інформації для будь-якого процесу або інформаційної системи. Вона використовує визначені символи, такі як прямокутники, кола та стрілки, а також короткі текстові мітки, щоб відобразити дані введення, виведення, точки зберігання та маршрути між кожним пунктом призначення.

Діаграма потоку даних – це тип діаграми, яка показує переміщення інформації з одного місця в інше як частину певного процесора в цілому. В інших випадках – DFD може показати, як різні відділи організації співпрацюють – це робить речі зрозумілими та узгодженими.

Основні елементами діаграми потоків даних: зовнішня сутність, процес, сховище даних та потік даних.

Детальніше опишемо кожен елемент:

– Зовнішня сутність – це зовнішня система, яка надсилає або отримує дані, спілкуючись із системою, яка зображена на діаграмі. Вони є джерелами та місцями призначення інформації, що входить або виходить із системи. Це може бути стороння організація чи особа, комп’ютерна система чи бізнес-система. Вони також відомі як термінатори, джерела та поглиначі чи актори. Зазвичай їх малюють по краях діаграми

– Процес – будь-який процес, який змінює дані, виробляючи вихід. Він може виконувати обчислення, сортувати дані на основі логіки або спрямовувати потік даних на основі бізнес-правил. Для опису процесу використовується короткий ярлик.

– Сховище даних – файли або репозиторії, які містять інформацію для подальшого використання, наприклад, таблицю бази даних або форму членства. Кожне сховище даних отримує просту мітку. На приклад «Тестові дані»

– Потік даних – маршрут, який проходять дані між зовнішніми об’єктами, процесами та сховищами даних. Він зображує інтерфейс між іншими компонентами та відображається стрілками, зазвичай позначеними коротким ім’ям даних.

З метою зображення діаграми потоків даних було визначено основні входи та виходи в інформаційній системі. Більшість процесів починаються з введення від зовнішнього об’єкта і закінчуються виводом даних, в результаті проведення цього процесу, до іншого об’єкта або сховища даних. Розпізнавання цих точок входу і виходу дозволяє ширше розглянути інформаційну систему та показує головний функціонал програмного забезпечення.

Після того, як були визначені основні входи та виходи даних, було побудовано контекстну діаграму інформаційної системи. Було проведено визначення основних процесів інформаційної системи та з’єднання їх потоками даних зі сховищами даних та зовнішніми сутностями.

Модель data flow diagram є одним з видів ієрархічної моделі. У таких діаграмах є можливість провести декомпозицію процесів, тобто розділити їх на структурні складові і більш детально представити в необхідній для конкретної задачі нотації на новій окремій діаграмі. Після того, як було досягнуто необхідного рівня глибини декомпозиції було проведено текстовий опис всіх рівнів діаграми.

Для побудови діаграми потоків даних та її декомпозиції було використано середовище, у якому можливо створювати концептуальні та логічні моделі – All Fusion Process Modeller.

На рисунку 2.2. зображено інформаційну систему для автоматизованого виявлення вразливостей у вебдодатках та дві зовнішні сутності: користувач та тестувальник. Система та сутності пов’язані, оскільки від кожної сутності до системи та від системи до сутностей йдуть потоки даних.

Сутність користувач з’єднується з інформаційною системою трьома потоками даних:

– На вхід до системи:

1) тестові дані.

– На вихід з системи:

1) запит на надання тестових даних;

2) результат тестування.

Сутність тестувальник з’єднується з інформаційною системою трьома потоками даних:

– На вхід до системи:

1) список тестових випадків.

– На вихід з системи:

1) Запит на надання списку тестових випадків;

2) результат тестування.

Далі була побудовані діаграми потоків даних (DFD) та було зображенно основні процеси інформаційно системи (рис. 2.3).

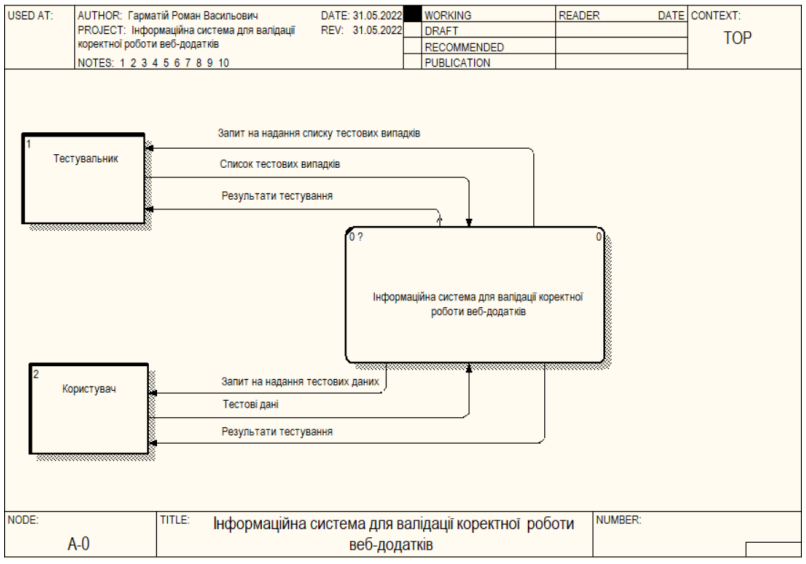

Рисунок 2.2 – Контекстна діаграма інформаційної системи для автоматизованого виявлення вразливостей у веб-додатках (зображення зовнішніх сутностей та їх зв’язків із інформаційною системою)

Сутність користувач з’єднується з інформаційною системою трьома потоками даних:

– На вхід до системи:

1) тестові дані.

– На вихід з системи:

1) запит на надання тестових даних;

2) результат тестування.

Сутність тестувальник з’єднується з інформаційною системою трьома потоками даних:

– На вхід до системи:

1) список тестових випадків.

– На вихід з системи:

1) Запит на надання списку тестових випадків;

2) результат тестування.

Далі була побудовані діаграми потоків даних (DFD) та було зображенно основні процеси інформаційно системи (рис. 2.3).

На діаграмі (рис. 2.3). можна спостерігати ті самі 2 зовнішні сутності:

– Користувач – особа яка хоче автоматизувати тестування власного веб-додатку. Для цього вона надає тестові дані.

– Тестувальник – основна зовнішня сутність системи, надає список тестових випадків, які будуть тестуватись.

Також на ній зображено 3 сховища даних:

– Тестові дані – це дані, які використовуються під час тестування, надаються користувачем.

– Список тестових випадків – це вибірка конкретний тестових випадків, які будуть тестуватися.

– Результати тестування – список з виконаних тестів, де вказано, чи пройшов тест, чи ні.

На діаграмі зображено 4 основних процеси:

– внесення списку тестових випадків;
– внесення тестових даних;
– тестування;
– обробка результатів.

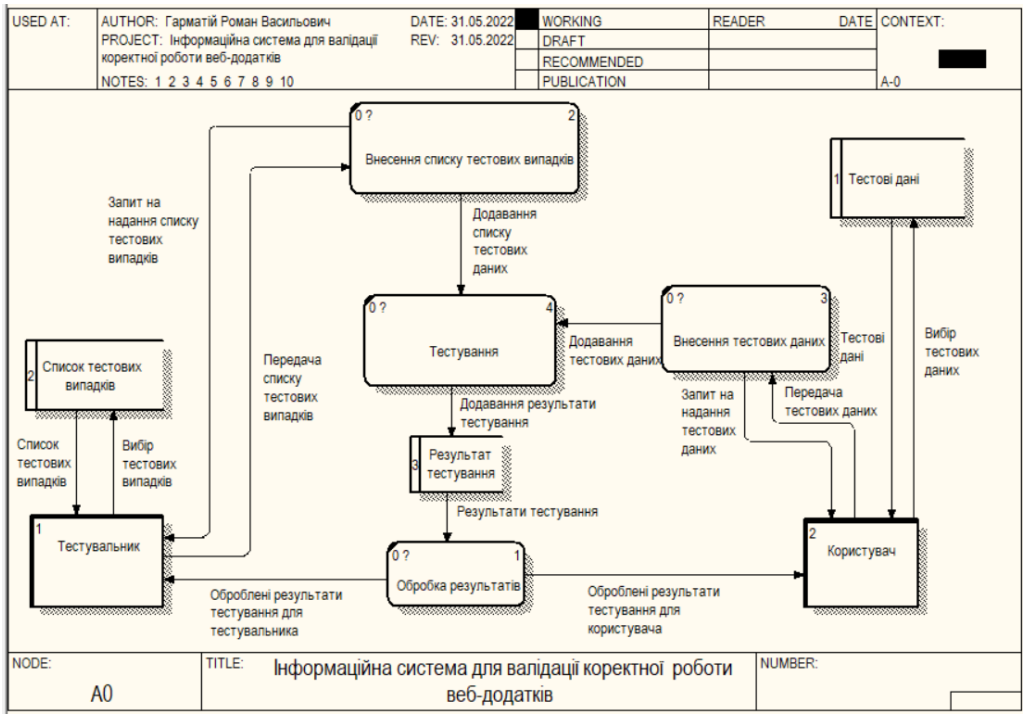

Рисунок 2.3 – Діаграма потоків даних інформаційної системи для автоматизованого виявлення вразливостей у вебдодатках

Процес «Внесення списку тестових випадків» означає, що тестувальник обирає із списку усіх тестових випадків ті, які будуть тестуватись, після цього йде з’єднання з сховищем даних «список тестових випадків». Після успішної отримання списку тестових випадків вони передаються для тестування. Процес «Внесення списку тестових випадків» має 3 потоки даних, 1 на вхід та 2 на вихід:

– Потоки даних на вхід:
  1) передача списку тестових випадків.
– Потоки даних на вихід:

1) запит на надання списку тестових випадків;
2) додавання списку тестових випадків.

Процес «Внесення тестових даних» означає, що клієнт звертається до сховища «Тестові дані» та обирає необхідні для тестування дані з нього. Після успішної отримання тестових даних вони передаються для тестування. Процес «Внесення тестових даних» має 3 потоки даних, 1 на вхід та 2 на вихід:

– Потоки даних на вхід:
  1) передача тестових даних.
– Потоки даних на вихід:
  1) запит на надання тестових даних;
  2) додавання тестових даних.

Процес «Тестування» – це основний процес який здійснюється інформаційною системою. Саме під час цього процесу здійснюються головні функції системи. Протягом процесу «Тестування» інформаційна система для автоматизованого виявлення вразливостей у вебдодатках здійснює автоматизоване тестування веб-сторінок веб-додатку. Для виконання цього процесу необхідно отримати дані з передаються під час попередніх процесів. Та після завершення процесу додаються у сховище «Результат тестування». Процес «Тестування» має 3 потоків даних, 2 на вхід та 1 на вихід:

– Потоки даних на вхід:
  1) додавання списку тестових даних;
  2) додавання списку тестових випадків.
– Потоки даних на вихід:
  1) додавання тестових даних.

Процес «Обробка результатів» означає приведення результатів тестування, які були отримані з попереднього процесу у вигляд доступний для користувача та тестувальника та передача цих даних. Для виконання цього процесу необхідно отримати дані зі сховища «Результати тестування». Процес «Обробка результатів» має 3 потоків даних, 1 на вхід, 2 на вихід:

– Потоки даних на вхід:

1) результати тестування.

– Потоки даних на вихід:

1) результати тестування для тестувальника;

2) результати тестування для користувача.

Наступним кроком є розбиття основних процесів діаграми DFD зображеної на рисунку 2.2.2. на підпроцеси, тобто їх декомпозиція.

Для початку виконаємо декомпозицію процесу «Внесення списку тестових випадків». Результат проведення декомпозиції даного процесу зображено на рисунку 2.4. Його можна декомпозувати на 2 підпроцеси:

– отримання тестових випадків;

– обробка списку тестових випадків.

Процес «Отримання тестових випадків» означає отримання інформації, яку ввів користувач, після того, як отримав запит на введення даних. Цей процес має 3 потоки даних, 1 на вхід та 2 на вихід:

– Потік даних на вхід:

1) передача списку тестових випадків – це дані які надає користувач

– Потік даних на вихід:

1) запит на надання тестових випадків – це запит, який має бути надісланим користувачу;

2) список тестових випадків – це дані, які передаються для подальшої обробки.

Процес «Обробка списку тестових випадків» означає обробку отриманих даних для подальшого їх використання. Даний процес має 2 потоки даних, 1 на вхід та 1 на вихід:

– Потоки даних на вхід:

1) список тестових випадків.

– Потоки даних на вихід:

1) додавання списку тестових випадків.

Наступним процесом для декомпозиції оберемо процес «Внесення тестових даних». Результат декомпозиції зображено на рисунку 2.5. Цей процес можна розбити на 2 підпроцеси:

– отримання тестових даних;
– обробка списку тестових даних.

Процес «Отримання тестових даних» означає отримання інформації, яку ввів користувач, після того, як отримав запит на введення даних. Цей процес має 3 потоки даних, 1 на вхід та 2 на вихід:

– Потік даних на вхід:

1) передача тестових даних – це дані, які надає користувач

– Потік даних на вихід:

1) запит на надання тестових даних – це запит, який має бути надісланим користувачу;

2) тестові дані – це дані, які передаються для подальшої обробки.

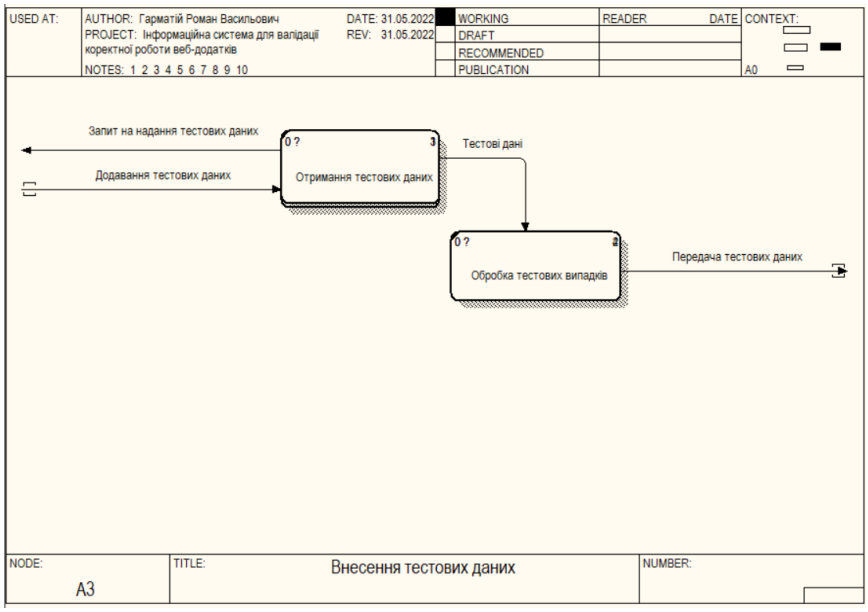

Рисунок 2.4 – Декомпозиція процесу «Внесення списку тестових випадків» інформаційної системи для автоматизованого виявлення вразливостей у вебдодатках

Процес «Обробка тестових даних» означає обробку отриманих даних для подальшого їх використання. Під час цього процесу інформаційна система обробляє отримані дані з попередніх етапів. Даний процес має 2 потоки даних, 1 на вхід та 1 на вихід:

– Потоки даних на вхід:

1) тестові дані.

– Потоки даних на вихід:

1) додавання тестових даних.

Виконаємо декомпозицію процесу «Тестування», результат декомпозиції зображений на рисунку 2.6. Даний процес можна розбити на 2 підпроцеси:

– Виконання тестування.

– Виведення результатів.

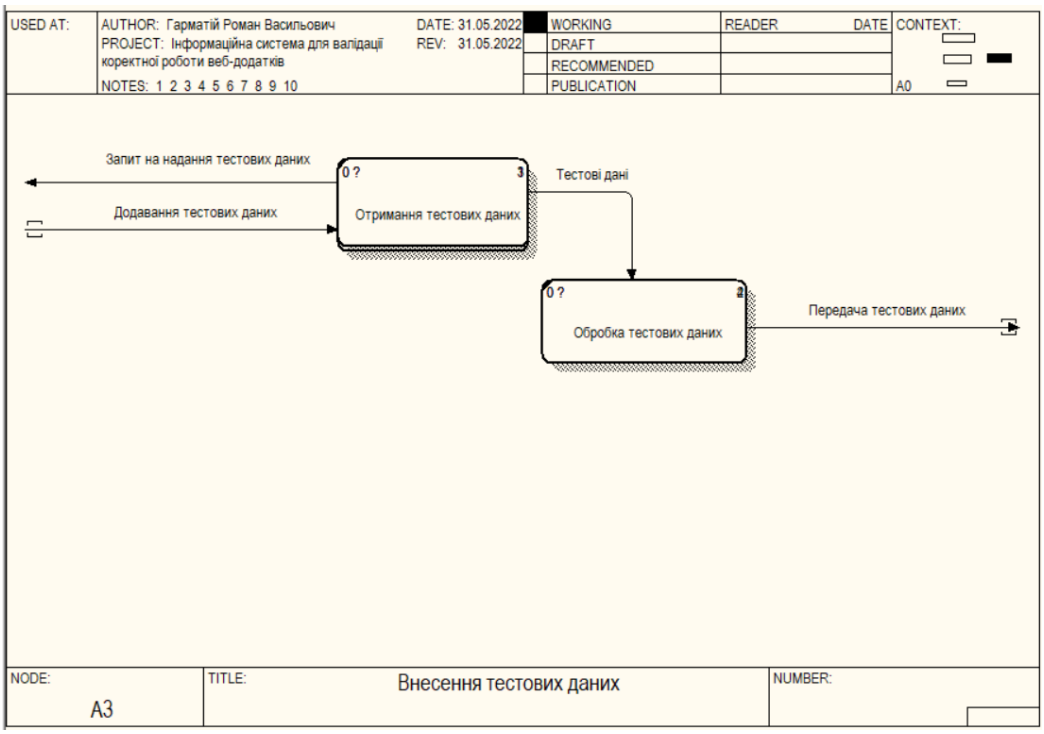

Рисунок 2.5 – Декомпозиція процесу «Внесення тестових даних» інформаційної системи для автоматизованого виявлення вразливостей у вебдодатках

Процес «Виконання тестування» основний у системі, він означає проведення автоматизованого тестування веб-додатку. Цей процес має 3 потоки даних, 2 на вхід та 1 на вихід:

– Потоки даних на вхід:

1) додавання списку тестових даних – отримані з від користувача.

2) додавання тестових даних – отримані від тестувальника.

– Потоки даних на вихід:

1) результати тестування.

Останнім процесом для виконання декомпозиції буде «Обробка результатів». Результат декомпозиції зображено на рисунку 2.7. Обраний процес можна розбити на 2 підпроцеси:

– Перевірка результатів тестування.

– Обробка результатів тестування.

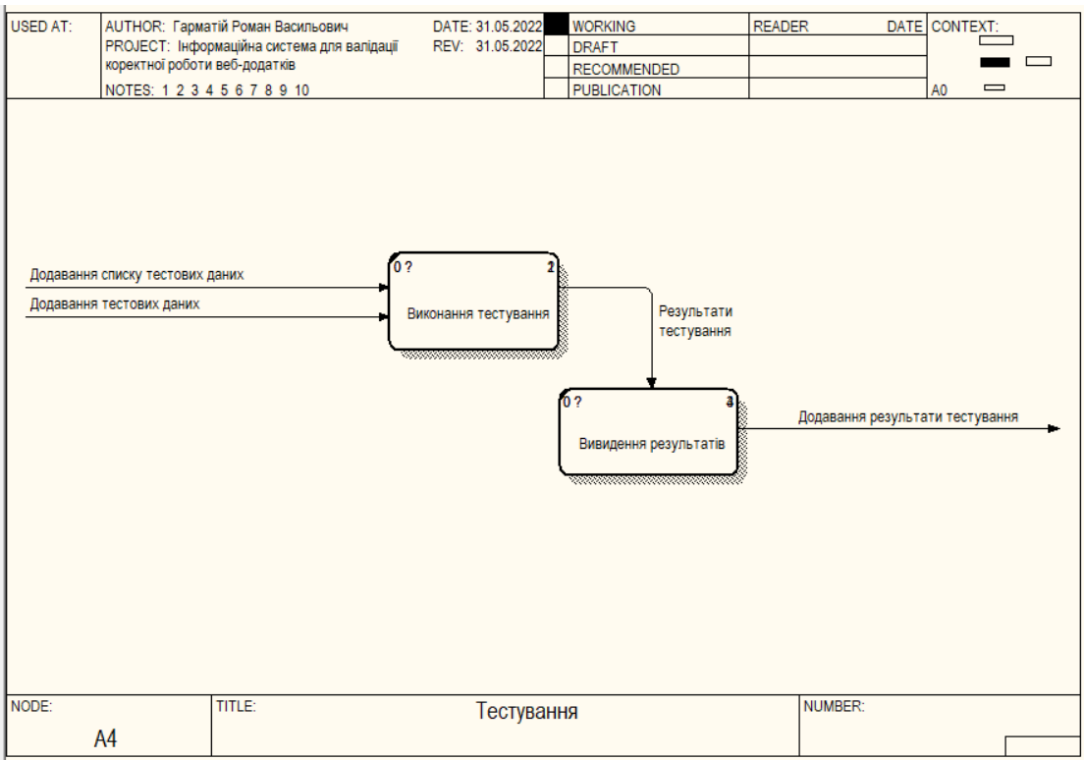

Рисунок 2.6 – Декомпозиція процесу «Тестування» інформаційної системи для автоматизованого виявлення вразливостей у вебдодатках

Процес «Перевірка результатів тестування» означає перевірку отриманих результатів та передача їх для подальшої обробки. Цей процес має 2 потоки даних:

– Потік даних на вхід:

1) результати тестування.

– Потік даних на вихід:

2) передача результатів тестування.

Процес «Обробка результатів тестування» означає безпосередню обробку отриманих результатів. Даний процес має 3 потоки даних:

– Потоки даних на вхід:

1) передача результатів тестування.

– Потоки даних на вихід:

1) оброблені результати тестування для тестувальника.

2) оброблені результати тестування для користувача.

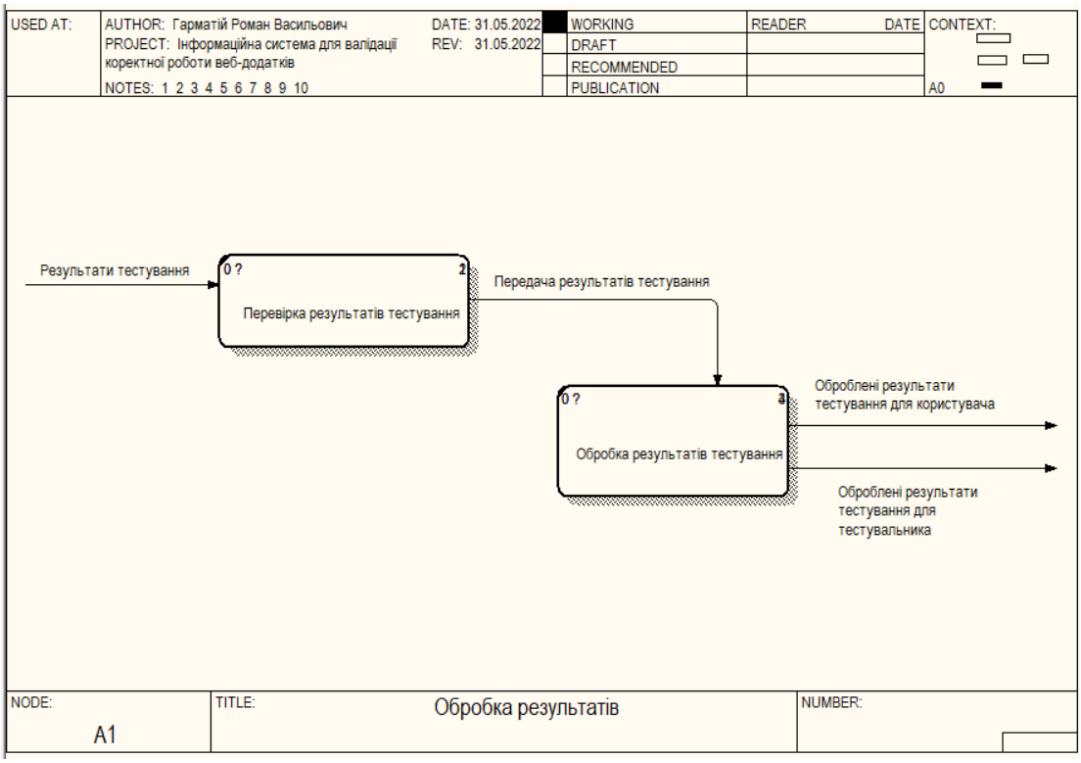

Рисунок 2.7– Декомпозиція процесу «Обробка результатів» інформаційної системи для автоматизованого виявлення вразливостей у вебдодатках

2.8 Побудова ієрархії задач

Ієрархія задач – це графічний спосіб відображення головних процесів у інформаційній системі та їх підпроцесів, які зображується у вигляді дерева. Ця діаграма дозволяє розширити розуміння функціоналу інформаційної та відстежити процес декомпозиції кожного наступного процесу.

Ієрархію задач було побудовано у середовищі All Fusion Process Modeler за допомогою інструменту «Node tree diagram» (діаграма дерева вузлів). Ця діаграма є ієрархічною та відображає функції інформаційної системи і зв'язки між ними. Вона створюється на основі попередньо побудованих діаграм та їх декомпозицій в нотації DFD чи IDEF0. «Node tree diagram» забезпечує широкий загальний огляд моделі інформаційної системи. Верхній процес, який знаходиться в корені дерева є головним в системі і визначає основну її функцію. На наступному рівні йдуть процеси на які можна розбити (виконати декомпозицію) основний процес. Після чого згідно із цією логікою продовжуємо спускаємось за ієрархією вниз, до досягнення найнижчого рівня декомпозиції інформаційної системи.

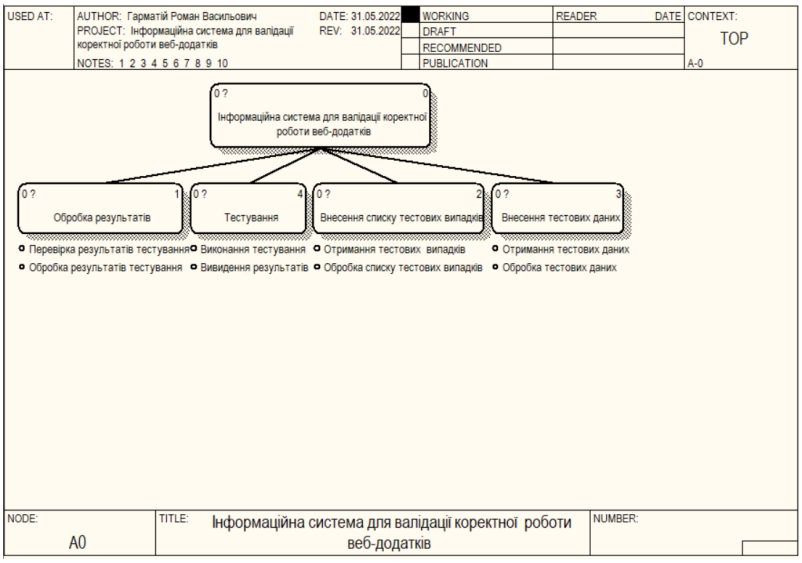

Рисунок 2.8 – Ієрархія задач інформаційної системи для автоматизованого виявлення вразливостей у вебдодатках

Побудована ієрархія задач для інформаційної системи для автоматизованого виявлення вразливостей у вебдодатках 2.8.

Коренем дерева є «Інформаційна система для автоматизованого виявлення вразливостей у вебдодатках», його можна розбити на 4 підзадачі:

– Внесення тестових даних – даний процес передбачає надання користувачем тестових даних для проведення тестування системою. Даний процес можна розбити на 2 підпроцеси:

1) отримання тестових даних.
2) обробка тестових даних.

– Внесення списку тестових випадків – додавання тестових випадків тестувальником для подальшого проведення тестування. Даний процес можна розбити на 2 підпроцеси:

1) обробка списку тестових випадків.
2) отримання списку тестових випадків.

– Тестування – основний процес системи. Проведення автоматизованого тестування згідно з наданими даними. Даний процес можна розбити на 2 підпроцеси:

1) виконання тестування.
2) виведення результатів.

– Обробка результатів – останні процес, під час якого отримані результати тестування обробляються і виводяться для користувача та тестувальника. Даний процес можна розбити на 2 підпроцеси:

1) перевірка результатів тестування.
2) обробка результатів тестування.

2.9 Модель «сутність-зв’язок»

Для побудови ER-діаграми використовуються такі елементи:

– Сутність – це об’єкт інформаційної системи, який має важливе значення для визначеної предметної області, а інформація про цей об'єкт може бути збережена. Набір об’єктів утворює один клас сукупні елементи якого мають однакові властивості. Кожен об’єкт цього класу має описуватись унікальним набором значень, який буде асоціюватись тільки із конкретною сутністю, який дозволяв би розрізняти окремі екземпляри класу. Сутності поділяються на звичайні, слабкі та проміжні.

– Зв’язок – це конкретна асоціація, яка поєднує дві чи більше сутності однієї предметної області та є важливою для даної області.

– Атрибут – це певна властивість сутності інформаційної системи, або зв’язку, які містить в собі дані про них. Атрибути існують прості та складені.

Побудована ER-діаграма для інформаційної системи для автоматизованого виявлення вразливостей у вебдодатках на рис. 2.9.

Дана ER-діаграма має 6 сутностей:

– Користувач – містить інформацію про користувача системи.

– Тестувальник – містить інформацію про тестувальника системи.

– Тестові дані – містить тестові дані.

– Список тестових випадків – містить список тестових випадків.

– Інформаційна система – містить інформацію про систему тестування.

– Отриманий результат – проміжна сутність, містить інформацію результат автоматизованого тестування.

Сутності між собою пов’язані такими зв’язками:

– Між сутностями Інформаційна система і Отриманий результат зв’язок «Тестує», 1:N – інформаційна система здійснює тестування і надає результат.

– Між сутностями Тестові дані і Інформаційна система зв’язок «Передає», 1:N – тестові дані передаються в інформаційну систему для тестування.

– Між сутностями Список тестових випадків і Інформаційна система зв’язок «Передає», 1:N – список тестових випадків передається в інформаційну систему.

– Між сутностями Тестувальник і Список тестових випадків зв’язок «Надає», 1:N – тестувальник надає інформацію про тестові випадки.

– Між сутностями Користувач і Тестові дані зв’язок «Надає», 1:N – користувач надає інформацію про тестові дані.

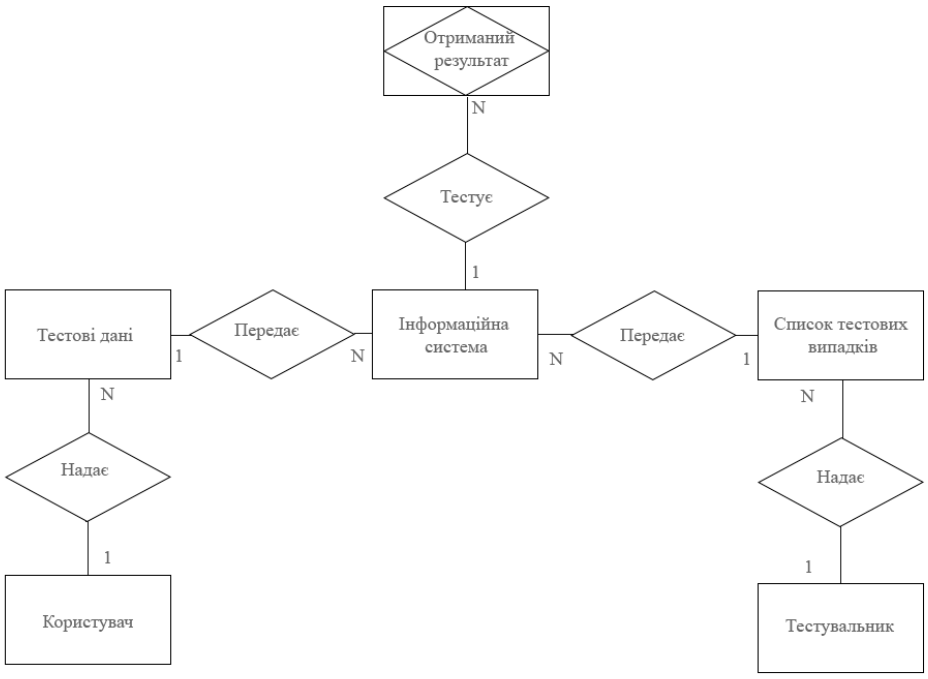

Рисунок 2.9 – ER-діаграма інформаційної системи для автоматизованого виявлення вразливостей у вебдодатках

3 ПРАКТИЧНА ЧАСТИНА

3.1. Вибір та обґрунтування засобів розв’язання задачі

3.1.1. JavaScrip і TypeScript

Незважаючи на схожість назви та синтаксису, зв’язку між JavaScript та Java немає. Як і Java, JavaScript є C-подібною мовою і наслідує деякий синтаксис в мови C.

Найчастіше JavaScript використовується в сценарії веб-сторінок. Веб-розробники можуть вбудовувати сценарії в сторінки HTML для виконання різноманітних завдань, таких як перевірка даних, введених користувачем, створення меню та інших анімованих ефектів.

Браузери запам’ятовують представлення веб-сторінки у вигляді дерева об’єктів і роблять ці об’єкти доступними для сценаріїв JavaScript, які можуть читати й маніпулювати ними. Дерево об’єктів називається моделлю об’єктів документа або DOM. Існує стандарт W3C для DOM, який повинен надаватися браузером, що забезпечує передумову для написання скриптів, які працюють у всіх браузерах. На практиці, однак, стандарт W3C для DOM реалізований не повністю. Хоча браузери, як правило, відповідають стандарту W3C, деякі з них все ще мають серйозні несумісності, наприклад Internet Explorer.

Однією з найпоширеніших технік створення веб-сторінок останніх років є AJAX, скорочення від «Asynchronous JavaScript and XML». Ця техніка передбачає виконання HTTP-запитів у фоновому режимі, без перезавантаження всієї веб-сторінки та оновлення лише певних частин сторінки, маніпулюючи сторінкою DOM. Техніка AJAX дозволяє створювати веб-інтерфейси з низьким рівнем реакції, оскільки (забирає багато часу) операція завантаження всієї HTML-сторінки значною мірою виключена. JavaScript є справді динамічною мовою. Для таких речей рідко

доводиться використовувати оператор оцінювання, тому що ви можете написати практично все, що завгодно, якщо синтаксис правильний і якщо те, що ви написали, не існує, ви отримаєте помилку під час виконання.

TypeScript – це мова програмування з відкритим вихідним кодом, розроблена та підтримувана компанією Microsoft, яка являє собою повний набір можливостей JavaScript і додає до мови необов'язкову статичну типізацію даних. Тобто якщо у JavaScript тип даних визначається динамічно, що у свою чергу може призвести до помилок, то у TypeScript тип даних визначається заздалегідь. Андерс Хельсберг, головний архітектор C# і творець ObjectPascal і TurboPascal, працював над розробкою TypeScript. TypeScript можна використовувати для виконання на стороні клієнта або сервера (node.js). Мова програмування призначена для розробки великих додатків. Її код у процесі компіляції перетворюється в JavaScript код. TypeScript – це наднабір JavaScript, тому потенційно будь-яка програма, написана на JavaScript, є дійсною програмою TypeScript [19].

Система типів Typescript виконує формалізацію типів Javascript через статичне представлення їх динамічних типів. Це дозволяє розробникам визначати змінні та типізовані функції, не втрачаючи суті Javascript. Можливість визначати типи під час розробки допомагає нам уникнути помилок під час виконання, таких як передача функції неправильного типу змінної.

На додаток до типів рядка і числа він підтримує такі основні типи:

– Boolean: логічний тип даних, що представляє істину або хибність.

– Масив: структурований тип даних, що дозволяє зберігати колекцію елементів.

– Кортеж: подібний до масиву, але з фіксованою кількістю записаних елементів.

– Будь-який: вказує, що змінна може бути будь-якого типу. Це дуже корисно при роботі із зовнішніми бібліотеками.

– Void: вказує, що функція не повертає жодного значення.

3.1.2. Node.js

Node.js – це програмна система, призначена для створення масштабованих веб-додатків. Програми можуть бути написані на JavaScript з керованим подіями асинхронним вводом-виводом, щоб мінімізувати перевантаження та максимізувати масштабованість.

Node.js використовує двигун JavaScript V8 від Google для виконання коду, і багато основних модулів написані на JavaScript. Node.js містить вбудовані бібліотеки, які дозволяють програмам працювати як сервери без програмного забезпечення, такого як Apache HTTP Server або IIS.

Node.js набуває популярності як серверна платформа і використовується різними компаніями.

Node.js працює з одним потоком, не блокуючи зв'язок введення-виводу, що дозволяє йому підтримувати десятки тисяч з'єднань одночасно, не турбуючись про затримки перемикання контексту між потоками. Конструкція однопотокового спільного використання між усіма запитами означає, що її можна використовувати для створення дуже програмного забезпечення з можливістю паралельної обробки запитів.

Недоліком цього підходу є те, що швидкість обробки запитів значно залежить від потужності ядра центрального процесора серверу, на якому буде працювати Node.js.

У поєднанні з браузером, базою даних документів (наприклад, MongoDB або CouchDB) і JSON, Node.js забезпечує уніфікований стек JavaScript для розробки. Завдяки підвищенню уваги до клієнтських фреймворків, а також адаптації переважно серверних шаблонів розробки, таких як MVC, MVP, MVVM тощо, Node.js дозволяє повторно використовувати ту саму модель та інтерфейс служби як для сервера, так і для клієнтів.

3.1.3. Mocha

Mocha – це фреймворк для валідації коректної роботи веб-додатків, розроблений мовою JavaScript, працює на основі node.js. Даний фреймворк підтримує асинхронне тестування, роботу з більшістю бібліотек припущень, які доступні мовою JavaScript.

Mocha-тести виконуються послідовно, забезпечуючи гнучкі та точні звіти, а також відображаючи невловлені винятки у правильні тестові випадки. Mocha – це досить проста і зрозуміла тестова система для JavaScript [20].

it

Дана анотація відповідає за позначення безпосередньо тестів. Саме за допомогою цієї позначки програмне забезпечення буде розуміти, що запущено тест, а не основний функціонал програми. Оскільки програма розумію, що запущено тест, відповідно буде надаватися більше інформації, яка корисна саме тестувальнику, а саме: покриття програми, у разі виникнення помилки буде вказано, у якому саме тесті вона виникла, у разі не відповідності очікуваного результату з реальним з’явиться порівнювальна таблиця. Саме за допомогою цієї анотації у тестувальників. Також використовується анотація describe для об’єднання тестів у групи. Найчастіше дане об’єднання відбувається через те, що тести стосуються однакового або суміжного функціоналу. Тести об’єднують для зручності роботи з ними, оскільки у разі, якщо тести будуть лише в одній великій групі – це значно ускладнить роботу з ними.

Before та after

За допомогою цих анотацій можна проводити дії до старту групи тестів. Дані дії виконуються кожного разу до та після роботи тестів. Ці анотації варто використовувати для налаштування тестового середовища, на приклад, для виконування дій, які повинні кожного разу бути виконані перед тестуванням. Також їх варто використовувати для повернення тестового середовища у початковий стан для подальшого тестування.

BeforeEach та afterEach

Ці анотації допомагають виконувати певні дії до та після кожного тесту. Як і попередні анотації before та after їх варто використовувати для виконання повторюваних дій, але у даному випадку ці дії виконуються не перед та після групи тестів, а саме до та після кожного окремого тесту. Ці анотації знадобляться для того, щоб запобігти проблемі залежності тестів, оскільки кожен тест буде починатися та закінчуватися однаковими діями. Залежні тести – це такі, де успішність виконання одного тесту залежить від результатів виконання попереднього тесту. Тобто якщо попередній тест закінчився помилкою, це означатиме, що і наступний теж закінчиться помилкою, хоч насправді її там може не бути.

3.1.4. Webdriver.io

Плюси WebdriverIO:

– WebdriverIO – це імплементація W3C WebDriver API. Не потрібно прив'язуватись до імплементації WebDriverJS.

– Підтримка синхронного коду. Можна забути про асинхронність JavaScript.

– Зручне стартове налаштування, яке здійснюється за допомогою вбудованого інтерфейсу для командного рядка wdio.

– Підтримує майже більшість тестових фреймворків, такі як BDD та TDD.

– Підтримує зручну бібліотеку 'webdrivercss', яка використовується для порівняння CSS-стилів елементів на сторінці.

Багатий функціями. Величезна різноманітність плагінів спільноти дозволяє легко інтегрувати та розширити налаштування відповідно до вимог. WebdriverIO дозволяє автоматизувати будь-які програми, написані за допомогою сучасних веб-фреймворків, а також власних мобільних додатків для Android та iOS.

WebdriverIO можна використовувати для автоматизації:

– сучасних веб-додатків, написаних в React, Vue, Angular, Svelte або інших фронтенд-фреймворках;

– гібридних або рідних мобільних програм, що працюють в емуляторі/симуляторі або на реальному пристрої;

– рідних настільних програм.

WebdriverIO використовує потужність протоколу WebDriver, розробленого та підтримуваного усіма постачальниками браузерів, і гарантує справжній досвід кросбраузерного тестування. Порівняно з багатьма засобами автоматизації, WebdriverIO – це інформаційна система з відкритим кодом, який працює з відкритим управлінням та належить некомерційній організації під назвою OpenJS Foundation.

3.1.5. Page Object pattern

Page Object – одне з найбільш корисних та використовуваних архітектурних рішень в автоматизації тестування веб-додатків. Даний шаблон проектування допомагає розбити роботу з окремими елементами сторінки на окремі класи, що дає змогу зменшити кількість коду та його підтримку. Якщо, наприклад, елементи змінили своє розташування на сторінці, то потрібно буде переписати лише відповідний клас, який стосується цієї сторінки.

Основні переваги:

– Поділ коду тестів на окремі класи та описи сторінок.

– Об'єднання всіх дій по роботі з веб-сторінкою та елементів веб-сторінки в одному класі.

Об’єкти сторінки найчастіше використовуються в тестуванні, але також можуть використовуватися для створення інтерфейсу сценаріїв поверх програми. Зазвичай краще розмістити інтерфейс сценаріїв під інтерфейсом користувача, це зазвичай менш складно і швидше.

3.1.6. Система контролю версій

Система контролю версій – це програмне забезпечення, яке дозволяє зберігати набір файлів, зберігаючи хронологію всіх внесених до нього змін. Зокрема, це дає змогу знайти різні версії пакету пов’язаних файлів.

Існують також децентралізоване (розповсюджене) програмне забезпечення та служби для контролю версій. Git і Mercurial1 є двома прикладами децентралізованого програмного забезпечення для контролю версій, які доступні в більшості систем Unix і Windows [22].

Для розробки інформаційної системи для автоматизованого виявлення вразливостей у вебдодатках було обрану Git.

Git – це система контролю версій, яка відстежує зміни у файлах і дозволяє зберігати точно історію розробки програного забезпечення.

В основному він використовується для розробки програмного забезпечення, але використовується для керування роботою над файлами будь-якого типу. Це розподілена система контролю версій, яка має на меті бути швидкою, підтримувати цілісність даних і підтримувати розподілені, нелінійні робочі моделі. Програма є безкоштовною і випущена під ліцензією GNU GPL 2.

3.2. Загальні відомості про програму

Інформаційна система для автоматизованого виявлення вразливостей у вебдодатках є призначеною для автоматизованого тестування веб-сторінок. За допомогою її можна автоматизувати процес тестування, який виконують тестувальники, значно спрощуючи та здешевлюючи цей процес. Система розроблена за допомогою мови програмування TypeScript, бібліотеки Webdriver.io.

3.2.1 Функціональне призначення

Програма призначена для тестування веб додатків. В першу чергу користувач визначає тестові дані, які він вносить в систему. Для різних тестових випадків користувач вводить різні дані, які користувач самостійно визначає перед кожним запуском інформаційної системи. Оскільки від введених тестових даних часто залежить який саме функціонал веб-додатку перевіряється. Від введених тестових даних залежить який набір тестів буде виконуватись. Система окремо їх перевіряє на дублікати, для того, щоб оптимізувати кількість ресурсів системи, які буть використовуватись. Система також дозволяє вибрати які безпосередньо тестові випадки будуть виконані під час даного циклу роботи системи. Можливо ще видалити тестові випадки з поточного сеансу, які будуть системою проігноровані та не будуть виконані. Такі тестові випадки будуть позначені, як пропущені.

Основною функцією системи є сеанс проведення тестування. Саме під час цього процесу використовується увесь створений до того функціонал системи. Під час цього система крок за кроком виконує тестовий випадок. Часто процес починається х авторизації в системі, коли система автоматизовано вводить логін і пароль, після чого веб-додаток обробляє даний запит і виконує авторизацію в системі, яка тестується. Після чого часто провадяться інші дії. Які саме залежать від тестового випадку. На приклад, може перевірятися реакція веб-додатку на автоматизоване натискання на елементи. Інформаційна система може перевіряти колір елементу веб-додатку до та після взаємодії, на основі чого робити висновки про те, чи коректно працює веб-додаток.

Початково в інформаційній системи додано лише кілька ознайомчих тестових випадків, а користувач повинен самостійно або з допомогою спеціаліста збільшувати кількість тестових випадків для їхнього веб-додатку. Це зроблено тому, що кожен веб-додаток унікальний і розробити інформаційну систему, яка підходила б для багатьох веб-сайтів неможливо. Тому інформаційна система для автоматизованого

виявлення вразливостей у вебдодатках легко масштабована. І її можливо у короткий термін підлаштувати під будь-який веб-додаток.

3.2.2 Опис логічної структури

Розроблена інформаційна система не має інтерфейсу користувача. Після того, як користувач введе та виконає зазначену у файлі README.md команду, вона звернеться за подальшими вказівками у відповідний файл з розширенням .spec.ts, який знаходиться в папці specs, в якому описані усі подальші кроки. Безпосередню логіку кроків, за методологією page object, розміщено у файлах у папці pages.

3.2.3 Технічні засоби

Для коректної роботи інформаційної системи необхідно застосовувати комп’ютер, який працює на операційній системі Windows 10 або Linux.

3.2.4 Виклик та завантаження

Інформаційна система розміщена на GitHub, для її завантаження потрібен доступ до інтернету. Після завантаження користувачу необхідно виконати усі кроки з файлу README.md для запуску роботи інформаційної системи. Після розгортання, програмний засіб працює без доступу до інтернету.

3.2.5. Вхідні дані та вихідні дані

Вхідними даними до системи є:

– Дані отримані від власника веб-додатку: логін, пароль для авторизації у веб-додатку.

– Дані отримані від тестувальника: список тестових випадків.

Вихідними даними системи інформація про статус тестових випадкові – пройдено або не пройдено тестування.

3.3. Інструкція користувача

Розроблена інформаційна система для автоматизованого виявлення вразливостей у вебдодатках призначена для автоматизації тестування веб-сторінко, для спрощення цього процесу. За допомогою даної системи тестувальник може виконувати перевірку тестових випадків, які часто повторюються. Також система дозволяє одночасно проводити тестування одного або декількох тестових випадків. Після виконання тестування система сповіщає користувача або тестувальника про результат кожного тестового випадку.

3.3.1. Загальні відомості про програму

Початок тестування здійснюється з допомогою команди, яка вказана у файлі README.md.

Інформаційна система розроблена з допомогою мови програмування TypeScript та середовища розробки IntelliJ Idea Ultimate.

Дана система може бути використана для тестування будь-яких веб-додатків, якщо попередньо буде індивідуально підлаштована під них.

3.3.2. Класи вирішуваних завдань

Для системи основною задачею є автоматизоване тестування веб-додатків. Для цієї задачі система збирає дані від тестувальника та користувача. Та після виконання тестування повертає результат кожного тестового випадку, відповідно до його статусу – або пройшов тестовий випадок, або ні.

У системі можливо вибрати окремо які тестові випадки будуть виконуватись, а які будуть проігноровані.

Перед початком роботи системи користувачу необхідно надати дані про свій веб-додаток. На приклад, логін та пароль для авторизації у ньому.

Отже, основними можливостями інформаційної системи для автоматизованого виявлення вразливостей у вебдодатках є:

– тестування веб-додатку;
– надання звіту про тестування;
– включення та виключення деяких тестових випадків у перед процесом тестування;

3.3.3. Опис основних характеристик і особливостей програми

Інформаційна система для автоматизованого виявлення вразливостей у вебдодатках унікальна через те, що на вона включає у себе готові рішення, які можна одразу використовувати у більшості веб-додатків, так і є можливість розширення функціоналу система для задовільнення потреб будь-якого веб-додатку.

Для функціонування інформаційної системи підключення до мережі Інтернет не є необхідним, і може запускатись на будь-якому комп’ютері чи ноутбуку з операційною системою Windows або Linux та наявним програмним інструментами для запуску системи, які потрібно встановити окремо. Система здатна працювати в будь-який зручний для користувача час. Підключення до мережі Інтернету знадобится тільки для завантаження системи з репозиторію.

3.3.4. Відомості про функціональні обмеження на застосування

Код системи розміщений в репозиторії GitHub у відкритому доступі і є доступним для завантаження, для чого необхідне підключення до мережі Інтернет.

Після завантаження коду із репозиторію підключення до мережі Інтернет не є необхідним для продовження роботи із системою.

Для запуску проект необхідно встановити будь-яке середовище розробки, на приклад Visual Studio Code або IntelliJ Idea Ultimate та виконувати усі кроки з файлу README.md.

3.4. Аналіз контрольного прикладу

Виконаємо аналіз контрольного прикладу, щоб зобразити роботу інформаційної системи для автоматизованого виявлення вразливостей у вебдодатках, а також переконатися у правильності виконання програмної реалізації.

Після завантаження програми користувач повинен відкрити файл README.md (рис. 3.1). Для початку роботи необхідно виконати кроки з розділу «Початкові кроки» та команду з розділу «Запуск тестування».

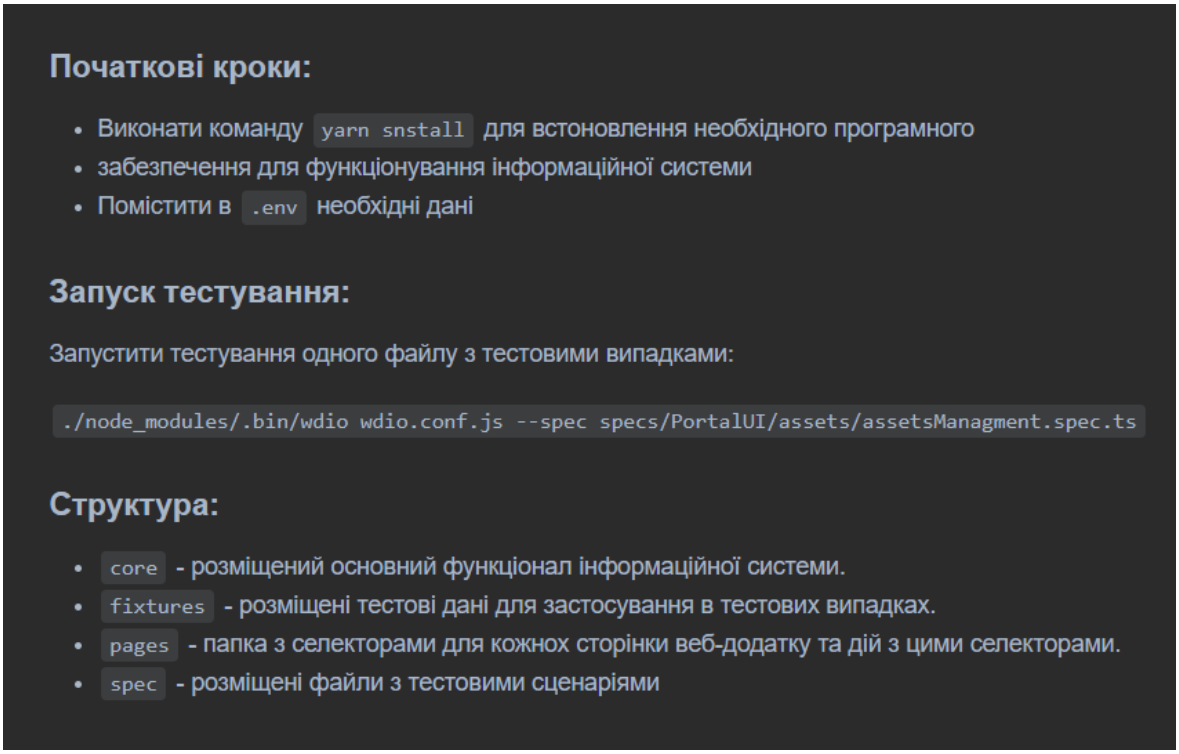

Рисунок 3.1 – Файл README.md

Якщо дані введено коректно, то система звертається до вказано файлу з тестовими випадками.

```
it( title: 'Viewing client assets details on the "Management/Assets" page [1797]', fn: () => {
  AssetsPage.searchPanel
    .chooseAvailableList(scopes.client)
    .chooseScopeName( scopeName: `[ Client ] ${clients.qa_client.name}`);
  AssetsPage.clickOnExpendButtonByName(
    deviceName: 'TEST-WIN10',
  ).detailTable.verifyDeviceName( expectedDeviceName: 'TEST-WIN10');
});

it( title: 'Disabling devices for the client on the "Management/Assets" page [1799, 1977]', fn: () => {
  AssetsPage.clickOnExpendMenuButton().filterMenu.clickOnFilterCheckbox(
    filters.monitorStatus.notMonitored,
  );
  AssetsPage.searchPanel
    .chooseAvailableList(scopes.client)
    .chooseScopeName( scopeName: `[ Client ] ${clients.qa_client.name}`);

  AssetsPage.clickOnExpendButtonByName( deviceName: 'TEST-WIN10')
    .detailTable.switchMonitoredDeviceState( isDeviceNotMonitored: false)
    .switchMonitoredDeviceState( isDeviceNotMonitored: true);
});
```

Рисунок 3.2 – Приклад файлу із тестовими випадками

В тестовому випадку детально та покроково описана усі взаємодія інформаційної системи з веб-додатком. Приклад такого файлу вказано на рисунку 3.2. Також можливе розподілення тестових випадків на ті, які будуть виконані та ті, які будуть пропущені (рис 3.3)

```
it.only( title: 'Viewing client assets details on the "Management/Assets" page [1797]', fn: () => {
  AssetsPage.searchPanel
    .chooseAvailableList(scopes.client)
    .chooseScopeName( scopeName: `[ Client ] ${clients.qa_client.name}`);
  AssetsPage.clickOnExpendButtonByName(
    deviceName: 'TEST-WIN10',
  ).detailTable.verifyDeviceName( expectedDeviceName: 'TEST-WIN10');
});

it.skip( title: 'Disabling devices for the client on the "Management/Assets" page [1799, 1977]', fn: () => {
  AssetsPage.clickOnExpendMenuButton().filterMenu.clickOnFilterCheckbox(
    filters.monitorStatus.notMonitored,
  );
  AssetsPage.searchPanel
    .chooseAvailableList(scopes.client)
    .chooseScopeName( scopeName: `[ Client ] ${clients.qa_client.name}`);

  AssetsPage.clickOnExpendButtonByName( deviceName: 'TEST-WIN10')
    .detailTable.switchMonitoredDeviceState( isDeviceNotMonitored: false)
    .switchMonitoredDeviceState( isDeviceNotMonitored: true);
});
```

Рисунок 3.3 – Приклад файлу із тестовими випадками, які розподілена на такі, які будуть виконуватись та ті, які будуть проігноровані

У наведеному прикладі інформаційна система звертається до файлу з тестовими даними.(рис. 3.4) Тестові дані зберігаються у такому форматі для зручності в користуванні тестувальником та заповнюються користувачем.

Рисунок 3.4 – Файл з тестовими даними

Також у прикладі тестового випадку ми можемо бачити методи, в яких реалізований функціонал взаємодії з сторінкою. Детально вони зображені на рисунку 3.5.

Рисунок 3.5 – Приклад методів для взаємодії з веб-додатком через локатор

На даному етапі свого функціонування система звертається безпосередньо до локатора (рис. 3.6) та до файлу, в якому зберігається основний функціонал системи – driver (рис 3.7)

```
private groupsByName(groupName: string) {
  return driver.getElement(
    selector: `//span[@class = 'parent-name' and contains(text(), "${groupName}")]`,
  );
}

private groupToggleButton(groupName: string) {
  return driver.getElement( selector: `${this.groupsByName(groupName).asString}//i`);
}

private expendMenuButton(): WebdriverIO.Element {
  return $( selector: `.menu-toggle .mdi`);
}

private loadingIndicator(): WebdriverIO.Element {
  return $( selector: '[class = "progress-wrapper"]');
}

private addToCollectionListButton(): WebdriverIO.Element {
  return $(
    selector: ' div.device-list-page > div.device-list > div.devices-table-view > div >
  );
}
```

Рисунок 3.6 – Приклад локатора, через який відбувається взаємодія з веб-додатком

```
waitForNotExist(
  element: WebdriverIO.Element,
  waitTime : number = this.defaultWaitTime,
): boolean {
  return element.waitForExist( options: { timeout: waitTime, reverse
}

waitPageToBeLoaded(): void {...}

waitUntilElementWillNotHaveAttributeContainsValue(
  selector: WebdriverIO.Element,
  attribute : string = 'class',
  attributeValue : string = 'loading',
  timeout : number = 20000,
) {...}

findElementByText(text: string): WebdriverIO.Element {
  return $( selector: `//*[text()=normalize-space('${text}')]`);
}

click(selector: WebdriverIO.Element): void {
  this.waitForVisible(selector);
  this.waitForEnabled(selector);
  selector.click();
}
```

Рисунок 3.7 – Приклад одного з основних методів системи

Після виконання тестового випадку інформаційна система повертає повідомлення про успішне проходження автоматизованого тестування (рис 3.8).

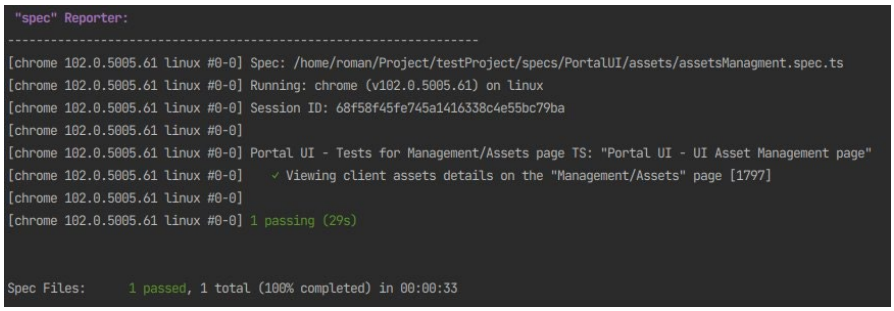

Рисунок 3.8 – Приклад повідомлення про успішне проходження тестування

Якщо у автоматизованому тестовому випадку виникла помилка, або якщо у веб-додатку помилка, тоді система вказує, на якому методі вона трапилась (рис. 4.9). У даному випадку необхідне додаткове дослідження спеціаліста з тестування.

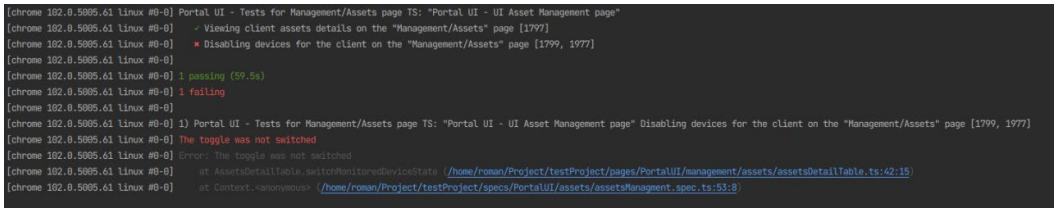

Рисунок 3.9 – Приклад повідомлення про помилку

Також після тестування доступний розширений огляд результатів, де детально описано кожен протестований тестовий випадок і його статус.

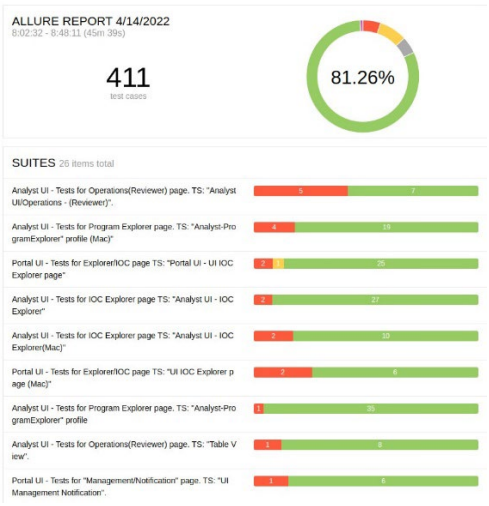

Рисунок 3.10 – Розширені результати тестування

Ще під час перегляду розширених результатів тестування можливо переглянути усі помилки, які виникали під час проведення тестування, що зображено на рисунку 4.11.

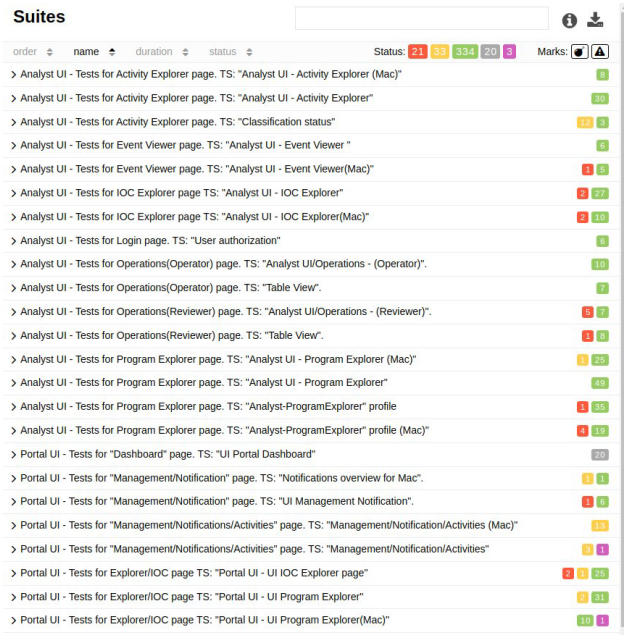

Рисунок 3.11 – Список помилок, які виникли під час тестування

На рисунку 3.10 зображені розширені результати тестування, де показана загальна кількість тестових сценарії та відсоток успішних тестових сценаріїв.

Зеленим кольором на зображені показано успішно виконані тестові сценарії. Червоним кольором відзначені такі тестові сценарії, які закінчились невдачею. Такий статус отримують тестові сценарії, у яких виникла помилка, або які закінчились знаходженням дефекту у веб-додатку. На рисунку 3.12 зображена візуальна репрезентація результатів тестування, де можливо оцінити скільки тестових випадків і з яким результатом були пройдені.

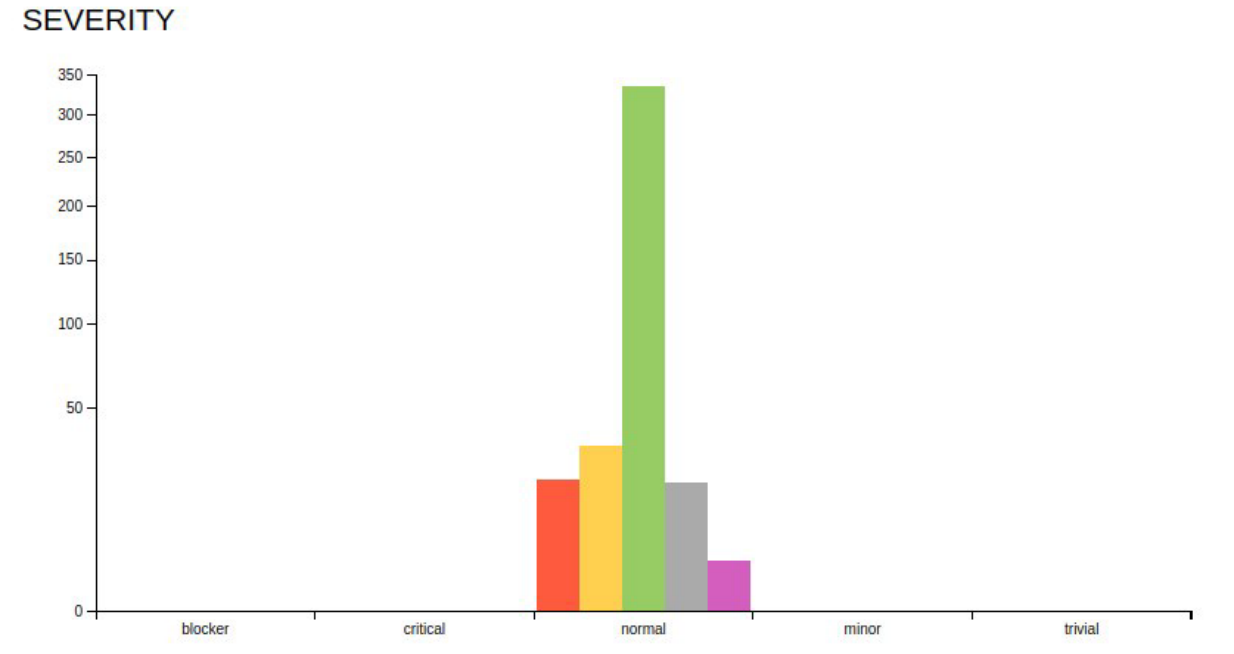

Рисунок 3.12 – Стовпцева діаграма результатів тестування

На рисунку 3.13 зображена кругова діаграма залежно від статусу тестового випадку.

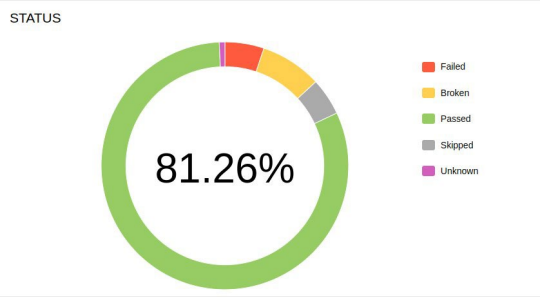

Рисунок 3.13 – Кругова діаграма результатів тестування

На рисунку 3.14 зображено графік проведення тестування відносно часу. На ньому можна оцінити час проведення тестування.

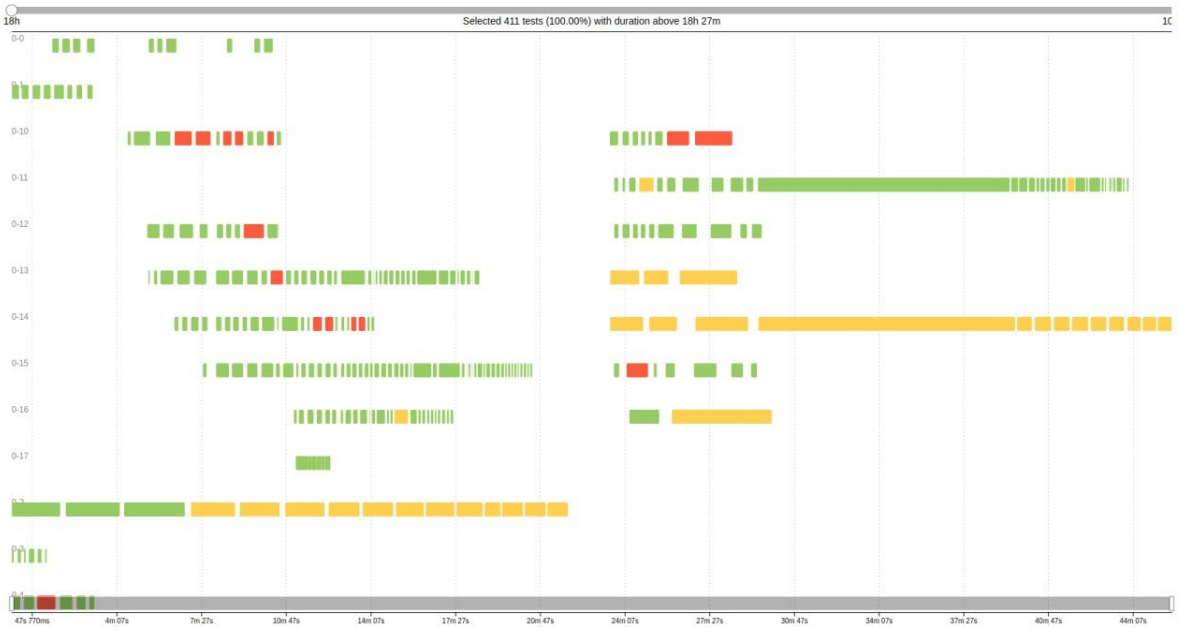

Рисунок 3.14 – Графік відносно часу проведення тестування

Після чого користувач може детально дослідити результат виконання кожного тестового випадку із різними статусами. На рисунку 3.15 зображено результат тестування тестового випадку із статусом «Пройдений».

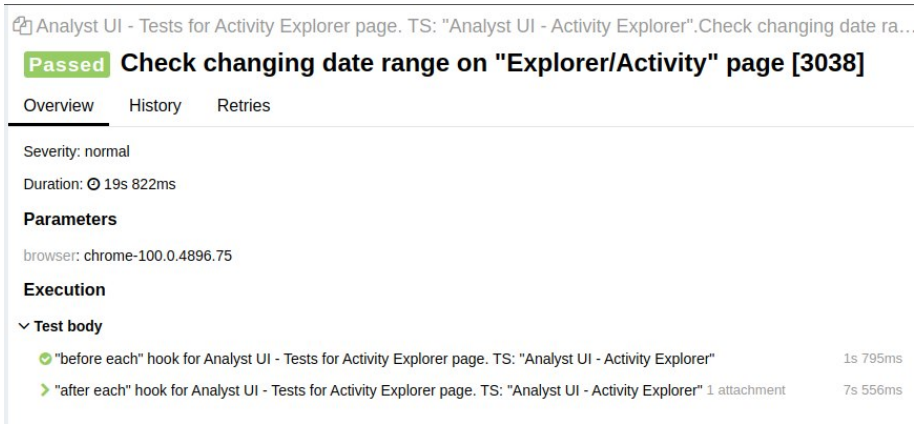

Рисунок 3.15 – Приклад результату тестування тестового випадку із статусом «Пройшов»

Якщо тестовий випадок закінчився із іншим статусом, його відображення після проходження тестування буде відображатись відповідно. На рисунку 3.16 зображено тестовий випадок із статусом «Не пройшов».

Також можливо побачити чому тестовий випадок отримав такий статус після тестування.

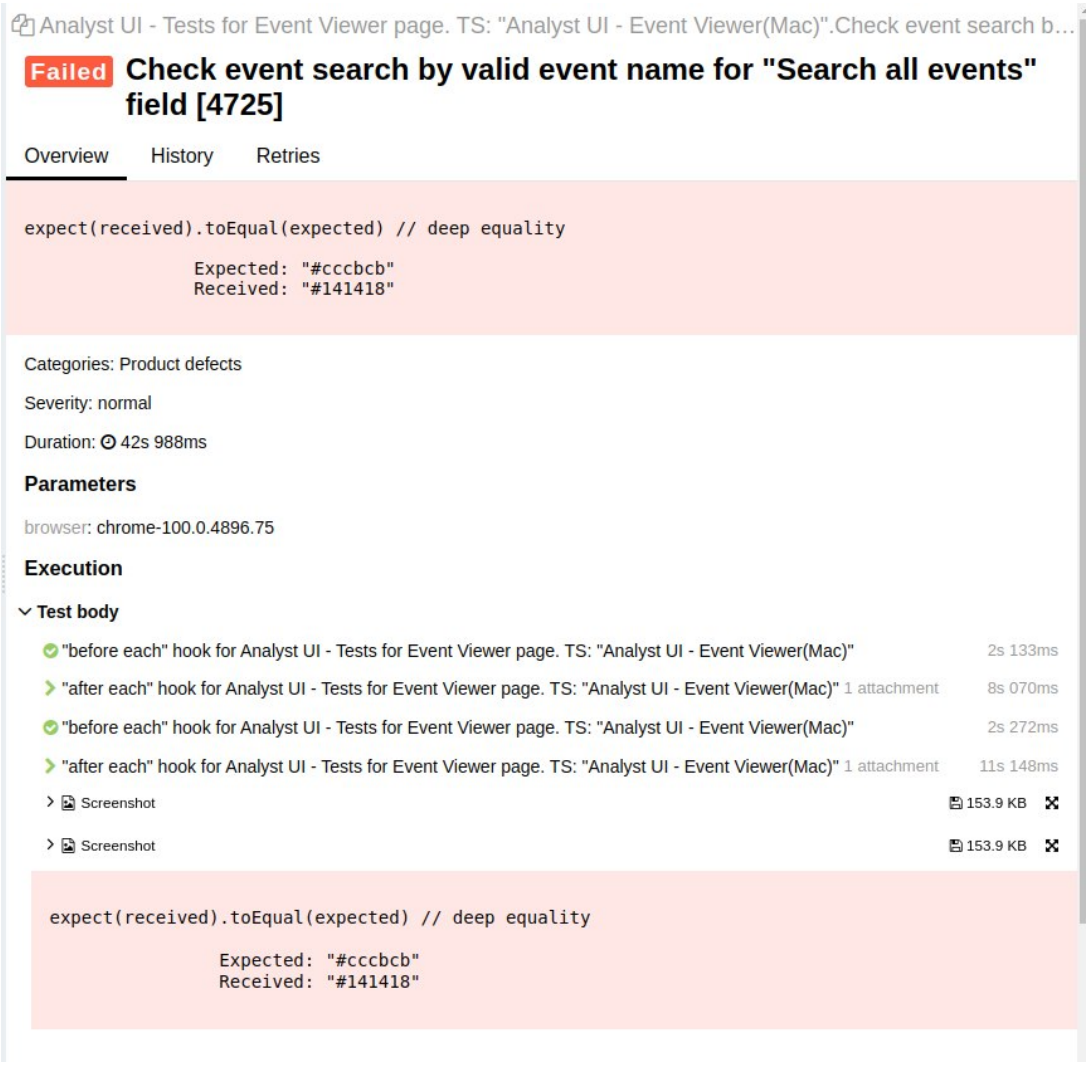

Рисунок 3.15 – Приклад результату тестування тестового випадку із статусом «Не пройшов»

Також тестові сценарії в деяких випадках можуть отримати статус «Зламаний», тобто такий, який потребує уваги зі сторони тестувальника, оскільки в логіці тесту з

високою ймовірністю помилка. Такий тестовий випадок може бути тимчасово пропущений, бо в деяких випадках може бути оцінений користувачем як такий, який отримав статус «Зламаний» через дефект у веб-додатку, а не через помилку в логіці тесту. На рисунку 3.16 зображено результат тестування тестового випадку із статусом «Зламаний».

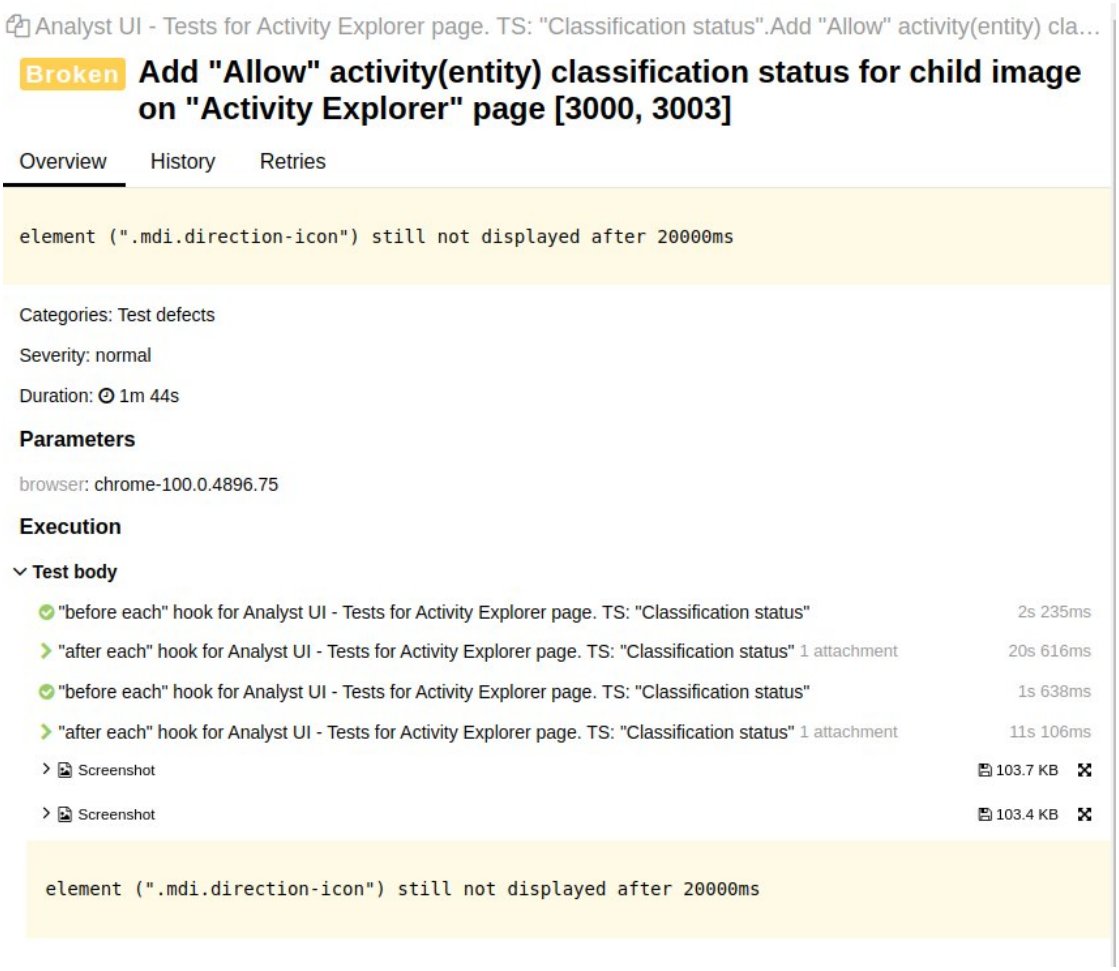

Рисунок 3.16 – Приклад результату тестування тестового випадку із статусом «Не пройшов»

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

Дослідження, проведені в рамках виконання магістерської кваліфікаційної роботи на тему "Інформаційної системи для автоматизованого виявлення вразливостей у вебдодатках" відповідають вимогам охорони праці, техніки безпеки та протипожежної безпеки.

Організація робочого місця та його технічне обладнання відповідають вимогам статті 43 Конституції України та статті 153 Кодексу законів про працю України, зокрема, створені безпечні та комфортні умови праці.

Впроваджені сучасні засоби техніки безпеки відповідають вимогам нормативних актів, зокрема, статті 49 Кодексу законів про працю України. Шаг регулювання елементів стільця, зусилля регулювання та висота поверхні сидіння відповідають рекомендаціям та стандартам щодо ергономіки робочого місця.

Заходи щодо протипожежної безпеки враховують вимоги статті 13 Закону України "Про охорону праці". Організація робочого простору та використання електрообладнання відповідає встановленим стандартам безпеки в умовах пожежі.

Забезпечення кібербезпеки в контексті розробки інформаційної системи відповідає вимогам стосовно захисту інформації та уникнення кіберзагроз, враховуючи нормативні документи, що регулюють цю сферу.

Всі ці заходи мають на меті не лише забезпечення ефективності дослідження, а й гарантування безпеки та здоров'я учасників, відповідно до вимог законодавства та стандартів охорони праці.

Під час розробки "Інформаційної системи для автоматизованого виявлення вразливостей у вебдодатках" основна увага приділялась не лише технічній ефективності системи, але і забезпеченню високого рівня охорони праці та кібербезпеки, що відповідає вимогам відомих стандартів та нормативів. Нижче представлена детальна інформація з цих аспектів.

Робоче середовище проекту було вивчено згідно із вимогами Закону України "Про охорону праці" та інших нормативних актів. Застосовані заходи включають в себе регулярні перевірки обладнання на відповідність, впровадження сучасних систем безпеки, а також організацію інструктажів для працівників.

У сфері кібербезпеки використовувалися передові практики та технології для захисту інформації від несанкціонованого доступу. Забезпечено конфіденційність та цілісність даних, виявлено та виправлено потенційні вразливості. Система відповідає вимогам міжнародних стандартів ISO/IEC 27001 та ISO/IEC 27002 з питань інформаційної безпеки.

Розробка відповідає вимогам стандартів охорони праці, таких як "Правила охорони праці під час експлуатації електронно-обчислювальних машин" та "Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин".

Застосовані заходи включають в себе не лише перевірки обладнання та використання сучасних систем безпеки, але й систематичні навчання працівників щодо правил охорони праці та надання їм необхідного інструктажу. Забезпечено відповідність робочого середовища стандартам, таким як "Правила охорони праці під час експлуатації електронно-обчислювальних машин" (НПАОП 0.00-1.28-10).

В області кібербезпеки вжиті заходи направлені на гарантування конфіденційності, цілісності та доступності інформації. Враховані вимоги міжнародних стандартів ISO/IEC 27001 і ISO/IEC 27002. Технічні рішення включають шифрування даних, двофакторну аутентифікацію та системи моніторингу для виявлення потенційних загроз.

Весь процес розробки відповідає нормативам та вимогам, визначеним Державним комітетом України з промислової безпеки, охорони праці та гірничого нагляду. Здійснювались регулярні перевірки відповідності нормативам, а також оновлення заходів з охорони праці відповідно до змін у відповідних законодавчих актах.

4.2 Безпека в надзвичайних ситуаціях

Кожен індивідуум має право працювати в умовах, які відповідають нормам належності, безпеки та здоров'я. Це забезпечено Конституцією України (частина 4, стаття 43).

Згідно зі статтею 153 Кодексу законів про працю України, на всіх підприємствах та в установах створюються безпечні та нешкідливі умови праці. Відповідальність за це покладається на власника чи уповноважений ним орган. Умови праці, безпека технологічних процесів, машин, устаткування та інших засобів виробництва, а також засоби колективного та індивідуального захисту повинні відповідати нормативам охорони праці.

Стаття 158 Кодексу законів про працю зобов'язує власника чи уповноважений орган вживати заходів для полегшення та оздоровлення умов праці, використовуючи сучасні засоби техніки безпеки. Згідно з Законом України "Про охорону праці", роботодавець повинен забезпечити відповідні умови праці відповідно до нормативно-правових актів та додержуватися вимог законодавства щодо прав працівників у сфері охорони праці.

Умови праці офісних працівників, які використовують персональні комп'ютери, повинні відповідати вимогам правил охорони праці під час експлуатації електронно-обчислювальних машин та державних санітарних норм. Ці правила охоплюють всі суб'єкти господарювання, які працюють з комп'ютерами. Створення комфортного робочого середовища передбачає правильне розташування робочого місця, відстань між комп'ютерами, обмеження шуму та вібрацій, а також використання ергономічних засобів.

Конструкція робочого місця користувача комп'ютера повинна забезпечувати оптимальну робочу позу. Регульованість робочого столу, стільця та інших елементів дозволяє адаптувати їх до індивідуальних потреб працівника. Зокрема, регулювання висоти, кута нахилу та інших параметрів стільця сприяє підтримці комфортної пози під час роботи.

Крім того, важливим елементом забезпечення здоров'я та безпеки працівників є робоче середовище офісних робочих місць. Згідно з нормами Державного комітету України з промислової безпеки, охорони праці та гірничого нагляду та Державних санітарних правил, працівники, які користуються персональними комп'ютерами, повинні мати достатнє природне освітлення та відстань до екрана монітору, що відповідає стандартам для зручного читання.

Розміщення робочих столів з комп'ютерами повинно враховувати не лише ергономічні аспекти, але і можливості концентрації уваги працівників. Для цього може бути застосовано перегородки висотою 1,5-2 м між робочими місцями, особливо там, де потрібна збільшена концентрація на виконанні завдань.

Важливою частиною облаштування офісного робочого місця є меблі. Регульованість робочого столу та стільця дозволяє підтримувати оптимальні розміри та висоту для кожного працівника, щоб уникнути несприятливого впливу на фізичне здоров'я. Застосування стаціонарних або змінних підлокітників може сприяти зниженню статичного напруження м'язів верхніх кінцівок.

Доцільно також розташовувати робочі місця відносно світових прорізів так, щоб природне світло падало з лівого боку, що сприяє кращій видимості та психофізіологічному комфорту.

Інтеграція сучасних технологій безпеки та охорони праці, дотримання нормативів та вивчення позитивного досвіду у галузі охорони праці є ключовими для забезпечення належних умов праці, що позитивно впливає на ефективність та здоров'я працівників.

Крім того, важливо дбати про ергономічність робочого стільця, забезпечуючи його підйомність, можливість повороту, регулювання висоти, кута та нахилу сидіння та спинки. Поверхня сидіння має бути плоскою, а передній край – заокругленим для комфорту користувача. Регулювання кожного параметра повинно бути незалежним, легким та надійним, з шагом регулювання в межах 15-20 мм для лінійних розмірів та 2-5 градусів для кутових. Зусилля при регулюванні не повинно перевищувати 20Н.

Висота поверхні сидіння повинна регулюватися в межах 400-500 мм, ширина і глибина не менше 400 мм. Кут нахилу сидіння може коливатися від 15 градусів вперед до 5 градусів назад. Висота спинки стільця, її ширина та радіус кривизни горизонтальної площини також регулюються. Кут нахилу спинки має змінюватися в межах 1-30 градусів від вертикального положення, а відстань від спинки до переднього краю сидіння регулюється в межах 260-400 мм.

Для забезпечення комфорту рук і зменшення статичного напруження м'язів, слід використовувати підлокітники завдовжки не менше 250 мм, завширшки 50-70 мм, що регулюються за висотою та відстанню між ними. Поверхня сидіння та спинки повинна бути напівм'якою з нековзним, повітронепроникним покриттям, що легко чиститься і не електризується. Робоче місце повинно включати підставку для ніг, регульовану за висотою та кутом нахилу, з рифленою поверхнею та бортиком по передньому краю заввишки 10 мм.

Розташування робочого місця відносно світових прорізів важливо для оптимального природного освітлення. Монітор повинен розташовуватися на оптимальній відстані від очей користувача, не ближче 600 мм, але з урахуванням розміру символів. Важливо також дотримуватися правил розташування екрана монітору, щоб забезпечити зручність зорового спостереження у вертикальній площині під кутом +30 градусів.

ВИСНОВКИ

В результаті виконання магістерської кваліфікаційної роботи було спроектовано та розроблено інформаційну систему для автоматизованого виявлення вразливостей у вебдодатках.

Для досягнення поставленої мети було виконано наступні задачі:

– Визначено мету та актуальність заданої теми.

– Проведено аналіз предметної області та порівняння з відповідними аналогами.

– Сформульовано вимоги до створюваної системи.

– Виконано проектування системи.

– Здійснено вибір засобів реалізації системи.

– Виконано програмну реалізацію.

Отже, поставлену мету було досягнуто та було виконано всі поставлені задачі, як результат було створено інформаційну систему для автоматизованого виявлення вразливостей у вебдодатках. Працездатність системи та її можливість виконувати поставлені завдання було доведено під час аналізу контрольного прикладу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Канер Кем, Фолк Джек, Нгуен Енг Кек. Тестирование програмного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений. – Киев. ДиаСофт, 2001. 544 с.

2. Що таке тестування програмного забезпечення. Режим доступу до ресурсу: https://www.quality-assurance-group.com/shho-take-testuvannya-programnogo-zabezpechennya-ta-yake-jogo-znachennya/ (дата звернення 09.12.2023)

3. Ручне та автоматизоване тестування. Режим доступу до ресурсу: https://qalight.ua/baza-znaniy/ruchne-ta-avtomatizovane-testuvannya/ (дата звернення 09.12.2023)

4. Що таке веб-додаток. Режим доступу до ресурсу: https://ukr.4meahc.com/what-exactly-is-web-application-50384 (дата звернення 09.12.2023)

5. Що таке веб-сайт. Режим доступу до ресурсу: http://www.webtec.com.ua/uk/articles/index/view/2011-05-05/web-site (дата звернення 09.12.2023)

6. JavaScript end-to-end testing framework. Режим доступу до ресурсу: https://www.cypress.io/ (дата звернення 09.12.2023)

7. Сусіденко В. Т. Інформаційні системи і технології в обліку: навч. посіб. / В. Т. Сусіденко. – Київ: «Центр учбової літератури», 2016. – 224 с.

8. Системний аналіз інформаційних процесів: навч. посіб. / В. М. Варенко, І. В. Братусь, В. С. Дорошенко, Ю. Б. Смольніков, В. О. Юрченко. – Київ: Університет «Україна», 2013. – 203 с.

9. Теорія систем і системний аналіз: навчальний посібник / О. А. Балтовський, К. Ю.Ісмайлов, О. І. Сіфоров, Г .В. Форос, О. М. Заєць. – Одеса: РВВ ОДУВС, 2021. – 156 с. Сорока К. О. Основи теорії систем і системного аналізу: навч. посібник / К. О. Сорока. – Харків: ХНАМГ, 2004. – 291 с.

10. Дудник І. М. Вступ до загальної теорії систем: навчальний посібник / І. М. Дудник. – Київ: Кондор, 2009. – 205 с

11. Скопенко Н. С. Інформаційні системи і технології в управлінні: конспект лекцій для для здобувачів освітнього ступеня «Магістр» спеціальності 073 «Менеджмент» освітньо – професійної програми «Менеджмент організацій і адміністрування» денної та заочної форм навч. / Н. С. Скопенко, М. П. Турчина. – Київ: НУХТ, 2020. – 91 с.

12. Дивак М. П. Системний аналіз: методичний посібник розроблений у відповідності з навчальним планом спеціальності «Економічна кібернетика» / М. П. Дивак. – Тернопіль: ТАНГ, 2004. – 136 с.

13. Васильків Н. М. Ефективність інформаційних систем: опорний конспект лекцій з освітньо-кваліфікаційного рівня “Спеціаліст” для спеціальності «Економічна кібернетика» / Васильків Н. М. – Тернопіль: Економічна думка, 2005. – 98 с.

14. Дегтярьова Л. М. Системний аналіз: методичні вказіки для здобувачів вищої освіти за освітньо-професійною програмою «Інформаційні управляючі системи» / Л. М. Дегтярьова. – Полтава: ПДАУ, 2021. – 56 с.

15. Сорока К. О. Основи теорії систем і системного аналізу: навч. посібник / К. О. Сорока. – Харків: ХНАМГ, 2004. – 291 с.

16. Функциональное моделирование с AllFusion Process Modeler 4.1.4: всё о работе с диаграммой Node Tree. Часть 1. Режим доступу до ресурсу: http://www.interface.ru/home.asp?artId=9871 (дата звернення 09.12.2023).

17. Дудник І. М. Вступ до загальної теорії систем: навчальний посібник / І. М. Дудник. – Київ: Кондор, 2009. – 205 с

18. The modern JavaScript. Режим доступу до ресурсу: https://javascript.info/ (дата звернення 09.12.2023)

19. Документація Typescript. Режим доступу до ресурсу: https://www.typescriptlang.org/ (дата звернення 09.12.2023)

20. Mocha – simple JavaScript testing framework. Режим доступу до ресурсу: https://mochajs.org/ (дата звернення 09.12.2023)

21. What is Webdriver.io? Режим доступу до ресурсу: https://webdriver.io/ (дата звернення 09.12.2023)

22. Основний сайт системи контролю версій git. Ключ доступу: https://git-scm.com/ (дата звернення 09.12.2023)

23. Allure Framework. Режим доступу до ресурсу: https://docs.qameta.io/allure/ (дата звернення 09.12.2023)

24. Wdio spec reporter. Режим доступу до ресурсу: https://webdriver.io/docs/spec-reporter/ (дата звернення 09.12.2023)

ДОДАТКИ

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

XI НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

13-14 грудня 2023 року

ТЕРНОПІЛЬ
2023

УДК 004.45

О.Р. Оробчук, доктор філософії, Р. В. Гарматій
Тернопільський національний технічний університет імені Івана Пулюя

АКТУАЛЬНІСТЬ ІНФОРМАЦІЙНИХ СИСТЕМ ДЛЯ АВТОМАТИЗОВАНОГО ВИЯВЛЕННЯ ВРАЗЛИВОСТЕЙ У ВЕБДОДАТКАХ

O.R. Orobchuk, Dr, R. V. Harmatii

INFORMATION SYSTEM FOR AUTOMATED VULNORABILITY DETECTION IN WEB APPLICATION

Дослідження та впровадження автоматизованих систем виявлення вразливостей у веб-додатках наразі є критично важливими у сфері інформаційної безпеки. Веб-технології, які є основними компонентами сучасних інформаційних систем, знаходять широке застосування в різних секторах, у тому числі в тих, які мають велике значення щодо інформаційної безпеки[1]. І навпаки, базові атаки на ці системи не обов'язково вимагають від зловмисників високого технічного досвіду, оскільки інформація про типові вразливості та методи атак широко поширюється в загальнодоступних джерелах.

Важливість цього дослідження підкреслюється різноманітністю веб-додатків, які задовольняють різні функціональні потреби в таких сферах, як фінанси, охорона здоров'я, зв'язок тощо[1]. Поширення таких додатків сприяє збільшенню потенційної вразливості та ризиків для інформаційної безпеки, що вимагає систематичного та ефективного підходу до виявлення потенційних недоліків.

Вищезазначені обставини підкреслюють актуальність впровадження автоматизованих систем виявлення вразливостей у веб-додатках. Ці системи відіграють ключову роль у запобіганні потенційним атакам і усуненні вразливостей, перш ніж їх можна буде використати для компрометації інформації.

У цьому контексті необхідність постійного моніторингу та вдосконалення систем виявлення стає невід'ємною частиною стратегії кібербезпеки. Розробка та впровадження передових технологій у цій галузі сприяє не лише покращенню захисту веб-додатків від потенційних загроз, але й зміцненню довіри користувачів до цих систем. Такий підхід дозволяє активно адаптуватися до змін загроз, забезпечуючи постійну безпеку в цифровому ландшафті, що швидко розвивається.

Автоматизовані системи виявлення вразливостей не лише вирішують безпосередні проблеми безпеки веб-додатків, але й сприяють досягненню головної мети сприяння культурі проактивної безпеки. Постійно скануючи вразливі місця, організації можуть зміцнити свій захист від нових загроз і адаптуватися до нових тактик, які використовують зловмисники[2].

Тонкощі цих додатків, які часто взаємопов'язані з різними базами даних і зовнішніми інтерфейсами, створюють складні проблеми для забезпечення надійної безпеки. Оскільки організації все більше покладаються на веб-платформи для критично важливих операцій, потенційні наслідки порушень безпеки мають далекосяжні наслідки, включаючи фінансові втрати, шкоду репутації та правові наслідки.

Література

1. Bandr Siraj Fakiha. 2020. Effectiveness of Security Incident Event Management (SIEM) System for Cyber Security Situation Awareness. International Journal of Forensic Medical and Toxicological Sciences. [online] Available at: https://medicopublication.com/index.php/ijfmt/article/view/11587/10679
2. Safe Exam Browser. https://safeexambrowser.org/about_overview_en.html

Додаток Б

assetsManagment.spec.ts

```
import LoginPage from '../../../pages/PortalUI/loginPage';
import { users } from '../../../fixtures/users';
import { availablePages } from '../../../fixtures/availablePages';
import driver from '../../../core/webDriver';
import AssetsPage from '../../../pages/PortalUI/management/assets/assetsPage';
import { clients } from '../../../fixtures/clients';
import { scopes } from '../../../fixtures/scopes';

describe('Portal UI - Tests for Management/Assets page TS: "Portal UI - UI
Asset Management page"', () => {
  before('Login to Portal UI', () => {
    LoginPage.openPortalUI()
      .login(users.registered.email, users.registered.password)
      .verifyAlertErrorIsNotExist();
  });

  beforeEach('Open Management/Assets page on Portal UI', () => {
    AssetsPage.navigationBar.goToThePage(
      availablePages.portalPages.management.main,
      availablePages.portalPages.management.assets,
    );
  });

  it('Viewing client assets details on the "Management/Assets" page [1797]',
() => {
    AssetsPage.searchPanel
      .chooseAvailableList(scopes.client)
      .chooseScopeName(`[ Client ] ${clients.qa_client.name}`);
    AssetsPage.clickOnExpendButtonByName(
      'TEST-WIN10',
    ).detailTable.verifyDeviceName('TEST-WIN10');
```

```
  });

  afterEach('Refresh page', () => {
    driver.actionAfterEachTest();
  });
});
import { AssetsCardView } from './assetsCardView';
import { AssetsSearchPanel } from './assetsSearchPanel';
import { NavigationBar } from '../../navigationBar.item';
import driver from '../../../../core/webDriver';
import { AssetsDetailTable } from './assetsDetailTable';
import { MainTable } from '../../../BaseElements/baseViews/tableViews/mainTable';
import { FilterMenu } from './devices/filterMenu';
```

assetsPage.ts

```
export class AssetsPage extends MainTable {
  readonly navigationBar = new NavigationBar();
  readonly searchPanel = new AssetsSearchPanel();
  readonly cardView = new AssetsCardView();
  readonly detailTable = new AssetsDetailTable();
  readonly filterMenu = new FilterMenu();

  private informationButton(): WebdriverIO.Element {
    return $('[title="Toggle client info"]');
  }

  private groupsByName(groupName: string) {
    return driver.getElement(
      `//span[@class = 'parent-name' and contains(text(), "${groupName}")]`,
    );
  }
```

```
  private expendMenuButton(): WebdriverIO.Element {
    return $(`.menu-toggle .mdi`);
  }

  private loadingIndicator(): WebdriverIO.Element {
    return $('[class = "progress-wrapper"]');
  }

  private addToCollectionListButton(): WebdriverIO.Element {
    return $(
      ' div.device-list-page > div.device-list > div.devices-table-view > div > div.table-wrapper.has-mobile-cards.has-sticky-header > table > tbody > tr.detail > td > div > div > div.columns.is-mobile > div > div > div > div > p:nth-child(1) > button',
    );
  }

  /*--------------------------------------------------------------------------*/
  clickOnInfoButton(): AssetsPage {
    driver.click(this.informationButton());
    return this;
  }

  clickOnExpendButtonByName(deviceName: string): AssetsPage {
    driver.click(this.privateExpandButtonByName(deviceName));
    return this;
  }

  clickOnExpendMenuButton(): AssetsPage {
    driver.click(this.expendMenuButton());
    return this;
  }
```

```
  clickOnCopyButtonByDeviceName(deviceName: string): AssetsPage {
    driver.click(this.copyButtonByDeviceName(deviceName));
    return this;
  }

  clickOnAddToCollectionListButton(): AssetsPage {
    driver.click(this.addToCollectionListButton());
    return this;
  }

  verifyTableIsLoaded(): AssetsPage {
    driver.waitForNotExist(this.loadingIndicator());
    return this;
  }
}
export default new AssetsPage();
```