Хмельницький національний університет
Факультет інформаційних технологій
Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр

Система захисту від Injection і XSS атак веб застосунків

КРКБ. 190107.19.01.08 ПЗ

Галузь знань 12 – Інформаційні технології

Спеціальність 125 – Кібербезпека

Виконав студент 4 курсу, група КБ-19-1 — Підпис — Матвеєв М.В. (Ініціали, прізвище)

Керівник д-р філософії (Науковий ступінь, вчене звання) — Підпис, дата — Стецюк М.В (Ініціали, прізвище)

Нормконтролер (Науковий ступінь, вчене звання) — Підпис, дата — Мостовий С.В. (Ініціали, прізвище)

До захисту допускаю:
Зав. кафедри кібербезпеки — Підпис, дата — Ковбой Ю.П. (Ініціали, прізвище)

15 06 2023 р.

Хмельницький, 2023

Формат	Зона	Позиц	Позначення	Найменування	Кільк.	Прим.
А4		1	КРКБ.190107.19.01.08 ПЗ	Система захисту від	55	
				Injection і XSS атак		
				веб застосунків		
				Пояснювальна записка		
А2		2	КРКБ.190107.19.01.08 Е8	Блок-схеми		
				захисту від		
				XSS-атак і SQLIA		
				Схема структурна		
А2		3	КРКБ.190107.19.01.08 Е8	Результати перевірки		
				XSS-атак і SQLIA		
				Схема структурна		
А2		4	КРКБ.190107.19.01.08 Е8	Схема MVC		
				Схема структурна		

Зм.	Арк.	№ Докум.	Підп.	Дата	КРКБ.190107.19.01.08 ВП		
Розробив		Матвеєв М.В		15.06	Система захисту від Injection і XSS атак веб застосунків Відомість проєкту	Літера	Аркуш 1 / Аркушів
Перев.		Стецюк М.В		15.06.		н	
Н. контр.		Мостовий С.В.		15.06.23	ХНУ, КБ-19-1		
Затв.		Кльоц Ю.П.		15.06.23			

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КІБЕРБЕЗПЕКИ

Освітній рівень БАКАЛАВР

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 125 КІБЕРБЕЗПЕКА

Освітня програма ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА ПІДГОТОВКИ БАКАЛАВРІВ

ЗАТВЕРДЖУЮ

Зав. кафедри Ю.П. Кльоц

"1" 03 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Матвеєв М.В

Прізвище, ім'я, по батькові студента

1. Тема роботи Система захисту від Injection і XSS атак веб застосунків

Керівник роботи д-р філософії Стецюк М.В

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджено наказом ректора університету від 01 березня 2023р. №5

2. Строк подання студентом роботи на кафедру 1 червня 2023р.

3. Вихідні дані до проекту (роботи) проаналізувати предметну область; визначити основні напрямки та методи атак; розробити алгоритм для виявлення і знешкодження загроз.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) Вступ. Аналіз об'єкта захисту. Огляд існуючих рішень та розробка системи захисту. Робота захисту. Висновки.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень). «Блок-схеми захисту від XSS-атак і SQLIA», «Схема MVC».

6. Консультанти розділів курсового проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання __1 березня___2023р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) кваліфікаційної роботи	Термін виконання етапів проекту (роботи)	Примітка
1	Ознайомлення з предметною областю	12.03.2023	–
2	Пошук теоретичної інформації про системи виявлення аномального трафіку на основі сигнатур	20.03.2023	–
3	Дослідження існуючих рішень	04.04.2023	–
4	Постановка задачі	18.04.2023	–
5	Пошук теоретичної інформації про найкращі рішення для виявлення аномального трафіку на основі сигнатур	26.04.2023	–
6	Початок впровадження системи виявлення аномального трафіку на основі сигнатур	01.05.2023	–
7	Завершення реалізації виявлення аномального трафіку на основі сигнатур	26.05.2023	–
8	Оформлення пояснювальної записки згідно вимог	28.05.2023	–
9	Оформлення графічної частини	31.05.2023	–
10	Захист КП	15.06.2023	

Студент _____ Матвеєв М.В
Підпис Ініціали, прізвище

Керівник проекту (роботи) _____ Стецюк М.В
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Система захисту від Injection і XSS атак веб застосунків»

Автор роботи: Матвеєв Максим Вячеславович

Керівник роботи: Стецюк Микола Васильович.

Пояснювальна записка: 55 с., 3 додатки, 27 рис., 40 джерел. Графічна частина: 10 презентаційних слайдів.

Метою роботи є розробка системи захисту від XSS-атак і SQL injection attack.

Було проведе дослідження процесу виявлення загроз у веб-застосунку та було розглянуто існуючі та актуальні рішення. Визначено, що заміння символів та екранування є досить ефективним і надійним способом захисту веб-сайту.

Також у результаті був розроблений алгоритм, який виявляє підозрілий символи і замінює їх на більш безпечні символи.

15.06.2023

Матв

ANNOTATION

Course project: "Protection system against Injection and XSS attacks of web applications"

Author of the work: Matveiev Maksym Vyacheslavovich

Supervisor: Stetsyuk Mykola Vasyliovych.

Explanatory note: 55 pp., 3 appendices, 27 figures, 40 sources. Graphic part: 10 presentation slides.

The method of work is to develop a protection system against XSS attacks and SQL injection attacks.

A study of the web application threat detection process was conducted and existing and current solutions were reviewed. Character substitution and shielding have been found to be a fairly effective and reliable way to protect a website.

Also, as a result, an algorithm was developed that detects the suspicious characters and replaces them with safer characters.

15.06.2023

ЗМІСТ

ВСТУП

З повним розквітом веб-застосунків та інтернет-технологій безпека веб-додатків стала надзвичайно важливою. Крім того, Injection та XSS-атаки є серйозними загрозами для безпеки веб-застосунків, тому зловмисникам не вдалося отримати доступ до конфіденційної інформації та зловживати ними.

У цій дипломній роботі ми досліджуємо проблему Injection-та XSS-атаки та розробляємо захист системи від цих атак веб-застосунків. У роботі розглядається теоретичний аспект проблеми, включаючи аналіз типів Injection та XSS-атак та їхніх наслідків для безпеки веб-додатків. Також будуть представлені існуючі методи захисту від цих атак та їхні переваги та недоліки. Основна мета роботи виконується в розробці нової системи захисту від Injection-та XSS-атаки, яка буде ефективною, надійною та простою у використанні.

Для досягнення цієї мети в дипломній роботі будуть різні типи Injection-та XSS-атаки, такі як SQL Injection, LDAP Injection, XSS на основі DOM та інші. Для кожного з цих типів також будуть описані їх особливості та наслідки для безпеки веб-додатків.Далі будуть присутні існуючі методи захисту від Injection-та XSS-атак, такі як перевірка введення, фільтрація введення та екранізація виводу. Для кожного з цих методів будуть описані їх переваги та недоліки, а також наведені приклади їх використання.

Після аналізу існуючих методів захисту від Injection-та XSS-атаки буде запропонована нова система захисту від цих атак. Ця система базується на комбінації різних методів захисту, які будуть ефективними проти різних типів Injection-та XSS-атак.

Усі результати дослідження та розробки описані в дипломній роботі, включаючи опис розробленої системи захисту, результати тестування та порівняння з існуючими методами захисту від Injection-та XSS-атак.

1 АНАЛІЗ ОБ'ЄКТА ЗАХИСТУ

1.1 Характеристика предметної області

Injection-та XSS-атаки є однією з найбільш розширених методів атаки на веб-застосунки. Їх особливість полягає в тому, що зловмисники забирають невразливості в коді веб-застосунків для введення шкідливого коду на сервер або для залучення даних користувачів. Ці атаки можуть призвести до крадіжки конфіденційної інформації, віддаленого керування серверами або навіть повного знищення даних.

Для захисту веб-застосунків від Injection-та XSS-атак необхідно розуміти різні типи захисту цих атак, їх особливості та методи. Крім того, необхідно використовувати сучасні технології та інструменти, щоб забезпечити максимальний рівень безпеки веб-застосунків.

Отже, предметною областю є кібербезпека веб-застосунків, яка включає в себе розуміння типів атак, їхніх наслідків, методів захисту та сучасних технологій для забезпечення безпеки веб-застосунків.

З одного боку, в області кібербезпеки веб-застосунків є захист від Injection і XSS-атак. Для цього необхідно розуміти особливості цих атак і способи їх запобігання.

Injection-атаки полягають у використанні зловмисників деяких вразливостей у вхідних даних, які передаються на сервер. Зловмисники можуть внести шкідливий код у вхідні дані, які після обробки сервером можуть призвести до втрати конфіденційної інформації або ж до недоступності веб-застосунку. Одним із найбільш розширених типів Injection-атак є SQL Injection, коли зловмисник використовує вразливості в SQL-запитах, щоб отримати недозволені дії.

XSS-атаки, з іншого боку, призначені для введення шкідливого коду у вихідну веб-сторінку, яка після обробки браузером може призвести до крадіжки конфіденційної інформації або змінити вміст веб-сторінки. Існують два типи

XSS-атак: збережені і відображені. У збережених XSS-атаках шкідливий код зберігається на сервері і виконується кожним разом, коли користувач отримує доступ до веб-сторінок, що містять вразливість. У відображених XSS-атаках шкідливий код передається на сервер через запити від користувачів.

Однією з основних характеристик предметної області XSSA і SQLIA є їх взаємозв'язок з веб-додатками і базами даних. XSSA атака використовує вразливості веб-додатка, які дозволяють впровадження шкідливого скриптового коду на сторінки, які відображаються у браузері користувача. Це може стати наслідком недостатньої валідації та екранування вхідних даних або недостатньої фільтрації виведених даних.

SQLIA, з свого боку, використовує вразливості баз даних, коли шкідливі SQL-запити можуть бути виконані безпосередньо або модифікувати структуру бази даних. Це може стати наслідком некоректного формування SQL-запитів, використання недостатньо параметризованих або підготовлених запитів, а також недостатньої перевірки прав доступу до бази даних.

Щоб запобігти XSSA і SQLIA атакам, необхідно вжити ряд заходів безпеки. Наприклад, для запобігання XSSA можна валідувати та екранувати вхідні дані, використовувати безпечні методи виведення даних, такі як HTML екранування, а також встановлювати заголовки безпеки, які запобігають виконанню скриптів у браузері.

Для запобігання SQLIA рекомендується використовувати параметризовані або підготовлені запити, що дозволяють вбудовувати параметри у SQL-запити, замість конкатенації рядків. Також слід обмежити привілеї користувачів бази даних, встановити надійні правила аутентифікації та авторизації, а також регулярно оновлювати веб-додаток та використовув

Забезпечення захисту від атак типу XSS (Cross-Site Scripting) і SQLIA (SQL Injection Attack) є надзвичайно важливим аспектом розробки веб-додатків. Для захисту від XSSA атак рекомендується використовувати валідацію та екранування вхідних даних, використання контент-фільтрів для блокування

небезпечного коду, використання безпечних кукі та застосування CAPTCHA для уникнення автоматизованих атак. Щодо захисту від SQLIA атак, рекомендується використовувати параметризовані запити або ORM для взаємодії з базою даних, правильно екранувати вхідні дані та виконувати валідацію, а також регулярно проводити тестування безпеки. Важливо враховувати ці методи в розробці та експлуатації додатків, щоб забезпечити надійний рівень захисту та запобігти можливим атакам на безпеку додатку.

У сучасному світі де кібербезпека стає все важливішою та кількість атак зростає, захист від Injection і XSS-атак є критичним для збереження бізнес-процесів та даних. Розуміння характеристик предметної області, методів та інструментів захисту може допомогти у створенні більш безпечних та надійних веб-застосунків.

1.2 Аналіз відомих методів забезпечення захисту від Injection-та XSS-атак

Існує багато методів та інструментів для забезпечення захисту від Injection і XSS-атак. Деякі з них є стандартними для різних веб-застосунків, тоді як інші можуть бути специфічними для конкретних технологій чи платформи.

Основні методи та інструменти захисту від Injection-та XSS-атак:

– використання параметризованих запитів: цей метод використовується для запитів до бази даних, коли параметри запиту виділяються командою SQL. Використання параметризованих запитів дозволяє уникнути вразливостей, пов'язаних з SQL Injection;

– екранування вхідних даних: цей метод використання для заміни символів, які можуть бути використані для Injection атак, на еквівалентні безпечні символи. Це забезпечує захист від XSS-атак;

– перевірка вхідних даних: перевірка правильності та дозволеності введених даних, але уникнути ін’єкції та XSS-атак;

– content Security Policy (CSP): це механізм, який дозволяє завантажити джерела, з яких можна завантажувати ресурси (скрипти, зображення, стилі), що запобігає виконанню шкідливих скриптів, в тому числі XSS-атак;

– cross-Site Request Forgery (CSRF) protection: захист від CSRF-атаки, яка виникає при створенні токенів, які перевіряються на кожен запит, тим самим запобігаючи недозволеним запитам;

– використання вогневих стін та брандмауерів веб-застосунків (WAF): ці інструменти допомагають виявляти та блокувати шкідливі запити та дії.

Варто відзначити, що захист від Injection-та XSS-атаки є постійним процесом, після чого з'явилися нові вразливості та методи атаки. Тому важливо не забезпечити тільки захист від відомих атак, а й виявляти та запобігати новим.

Крім того, інші спеціалізовані платформи та інструменти, які можуть автоматично виявляти та виправляти вразливості, пов’язані з Injection та XSS-атаками. Такі інструменти можуть сканувати веб-застосунки, виявляти деякі вразливості та давати рекомендації щодо їх виправлення. Однак варто відзначити, що такі інструменти не можуть забезпечити 100% захист від нападу.

Також важливо виконати те, що захист від Injection та XSS-атак є не тільки технічними, а й організаційними питаннями. Можливо відповідально підходити до розробки веб-застосунків та використовувати практики безпеки програмного забезпечення, такі як регулярні аудити безпеки, захист даних та обмеження прав доступу користувачів до системи.

Одним із ефективних методів захисту від Injection та XSS-атак є використання параметризованих запитів. Параметризований запит - це запит до бази даних, в яку передається користувачем, замість підключення до запиту через конкатенацію, передається окремо як параметр запиту. Це дозволяє уникнути вразливості, пов'язаної з ін'єкційними атаками.

Щодо захисту від XSS-атаки, важливо коректно валідувати та екранувати дані, які наводяться користувачем.

Також до ефективних методів захисту від XSS-атак належать:

– використовуються HTTPOnly-куки, які не можуть бути доступні для скриптів на веб-сторінках, а тим самим запобігають крадіжку сесії;

– використання вбудованих фільтрів для запобігання введенню шкідливого коду, таких як HTMLPurifier, JSoup та ін.

Наприкінці можна вказати, що захист від Injection та XSS-атак є важливою складовою безпекою веб-застосунків. Для ефективного захисту необхідно використовувати різноманітні технічні та організаційні заходи, а також постійно вдосконалювати та контролювати безпеку програмного забезпечення.

Додаткові методи для захисту від ін'єкцій та XSS-атак включають використання параметризованих запитів та підготовлених виразів при роботі з базою даних. Це дозволяє відокремити дані від коду запиту, запобігаючи ін'єкції SQL-запитів. Також, екранування спеціальних символів при виведенні даних на сторінку може запобігти XSS-атакам шляхом перетворення потенційно небезпечних символів у безпечні. Використання коректних заголовків Content Security Policy (CSP) може додатково зменшити ризик XSS-атак. Регулярне оновлення всіх компонентів, включаючи фреймворк, бібліотеки та інструменти, є також важливим, оскільки це допомагає уникнути відомих вразливостей та отримати оновлення з виправленнями безпеки. Загалом, комбінація цих методів та постійна увага до безпеки можуть допомогти забезпечити веб-додаток від ін'єкцій та XSS-атак.

Отже, при розробці веб-застосунків необхідно лише використовувати відомі методи захисту від Injection та XSS-atak, а також підтримувати постійний контроль за безпекою програмного забезпечення та використовувати сучасні інструменти для виявлення та запобігання новим вразливостям.

1.3 Injection attack

Ін'єкційні атаки - це вид атаки на веб-застосунки, які зазвичай використовують зловмисники для отримання несанкціонованого доступу до даних або системи. Injection-атаки базуються на тому, що зловмисник може внести зловмисний код у веб-застосунок, який буде працювати на сервері без знання власника веб-застосунку. Якщо зловмисник може внести код у систему, він може використовувати його для виконання різноманітних дій, таких як:

– збереження доступу до конфіденційної інформації, яка зберігається на сервері;

– зміна даних у базі даних або виконання інших дій, які можуть бути включені на роботу веб-застосунку;

– виконання інших видів атак на веб-застосунок, таких як XSS-атаки.

Для захисту від Injection-атаки необхідно використовувати спеціальні засоби та інструменти, які можуть перевірити вхідні дані на валідність і запобігти внесенню зловмисного коду у веб-застосунок. Деякі з таких технік включають в себе валідацію вхідних даних, використання параметризованих запитів, а також відсіювання спецсимволів, які використовують в Injection-атаках.

Крім того, до інших технік захисту від Injection-атак можна віднести:

– використання пакетів захисту від Injection-атак. Такі пакети можуть бути встановлені в системі та автоматично перевіряти вхідні дані на валідність. Вони також можуть блокувати спроби внесення зловмисного коду в систему;

– використання пакетів захисту від Injection-атак на рівні сервера. Такі пакети можуть бути встановлені на сервері та перевіряти всі вхідні запити перед тим, як вони будуть оброблені веб-застосунком. Це дозволяє запобігти внесенню зловмисного коду в систему до того, як він буде оброблений веб-застосунком;

– використання технологій, таких як stored procedures та prepared statements. Stored procedures дозволяють виконувати попередньо скомпільований код на сервері, що може запобігти внесенню зловмисного коду в систему;

Prepared statements дозволяють підготувати запит до бази даних з відсіюванням зловмисного коду.

Ін’єкційна атака, наприклад SQL Injection (SQLIA), є вразливістю системи безпеки, коли зловмисник маніпулює вхідними даними для виконання ненавмисних команд або запитів. Це може призвести до несанкціонованого доступу до бази даних, витоку даних, маніпуляцій або навіть зламу системи. Щоб запобігти ін’єкційним атакам, вкрай важливо впроваджувати такі заходи, як використання параметризованих запитів, перевірка та дезінфекція вхідних даних, посилення контролю доступу, проведення регулярних оцінок безпеки та просування методів безпечного кодування. Поєднуючи ці засоби захисту, програми можна краще захистити від ін’єкційних атак і забезпечити безпеку та цілісність даних.

SQLIA (атаки впровадження SQL) є серйозною загрозою безпеці веб-додатків, які покладаються на системи баз даних. Ці атаки використовують уразливості в обробці програмою введених користувачем даних для виконання зловмисних запитів SQL до бази даних. Для ефективного захисту від SQLIA важливо впровадити профілактичні заходи та передові методи.

Одним із важливих профілактичних заходів є перевірка вхідних даних і параметризовані запити. Усі введені користувачем дані мають бути ретельно перевірені, щоб переконатися, що вони відповідають очікуваному формату та типу. Крім того, слід використовувати параметризовані запити або підготовлені оператори, щоб відокремити код SQL від введення користувача. Цей підхід гарантує, що введені користувачем дані розглядаються як дані, а не як виконуваний код, що ефективно зменшує ризик SQLIA. Іншим важливим механізмом захисту є конфігурація безпечної бази даних. Реалізуючи принцип найменших привілеїв, слід суворо контролювати права доступу до бази даних. Замість облікових записів адміністратора за замовчуванням слід використовувати спеціальні облікові записи бази даних програми з обмеженими

привілеями. Це обмежує масштаб потенційної атаки та мінімізує шкоду, яку може завдати зловмисник.

Регулярні оновлення програмного забезпечення та керування виправленнями є вирішальними для запобігання SQLIA. Системи баз даних і базові фреймворки або бібліотеки повинні постійно оновлюватися за допомогою останніх виправлень безпеки. Це допомагає усунути відомі вразливості та зменшує ймовірність успішних атак. Тестування безпеки, наприклад сканування вразливостей і тестування на проникнення, слід проводити регулярно, щоб виявити та усунути вразливості SQLIA. Інструменти автоматичного сканування можуть допомогти виявити загальні вразливості, тоді як ручне тестування дозволяє провести глибокий аналіз і виявити складні проблеми. Важливо перевірити програму з точки зору зловмисника, щоб виявити та пом’якшити будь-які потенційні точки впровадження SQL. Брандмауери веб-додатків (WAF) можуть забезпечити додатковий рівень захисту від SQLIA. Ці пристрої безпеки знаходяться між сервером додатків і клієнтом, відстежуючи та фільтруючи вхідні запити. WAF можуть виявляти та блокувати спроби зловмисного впровадження SQL, захищаючи програму від потенційних атак.

Реєстрація бази даних і моніторинг необхідні для виявлення та відповіді на SQLIA. За допомогою моніторингу та аналізу журналів активності бази даних можна виявити підозрілі або нестандартні запити. Системи виявлення вторгнень (IDS) і системи запобігання вторгненням (IPS) можуть бути використані для автоматичного виявлення та блокування спроб впровадження SQL на основі попередньо визначених шаблонів або евристик. Навчальні та просвітницькі програми мають життєво важливе значення для забезпечення того, щоб розробники та адміністратори розуміли ризики та наслідки SQLIA. Навчаючи персонал методам безпечного кодування, наголошуючи на важливості перевірки вхідних даних і сприяючи використанню параметризованих запитів, організації можуть посилити свій захист від атак SQL-ін’єкцій.

Підсумовуючи, захист від SQLIA вимагає поєднання профілактичних заходів, безпечної конфігурації, регулярних оновлень і керування виправленнями, тестування безпеки та моніторингу. Впроваджуючи ці методи та розвиваючи культуру безпеки, організації можуть значно знизити ризик атак із впровадженням SQL і захистити конфіденційність і цілісність своїх баз даних.

Перед лицем нових кіберзагроз організаціям і розробникам важливо залишатися пильними та постійно вдосконалювати заходи безпеки. Це включає в себе оновлення останніх методів атак, розуміння вразливостей, характерних для їхніх програм, і впровадження належного захисту. Регулярні перевірки безпеки, тестування на проникнення та перевірки коду можуть допомогти виявити й усунути потенційні вектори ін’єкційних атак. Крім того, впровадження безпечних методів кодування, таких як перевірка вхідних даних, кодування вихідних даних і належна дезінфекція введених даних користувачами, може значно зменшити ризик ін’єкційних атак. Використання брандмауерів веб-додатків і систем виявлення вторгнень може забезпечити додатковий рівень захисту шляхом активного моніторингу та фільтрації зловмисних запитів. Регулярні оновлення системи безпеки та виправлення для інфраструктури веб-додатків і основного стека програмного забезпечення є критично важливими для усунення будь-яких відомих уразливостей. Крім того, сприяння культурі обізнаності про безпеку серед розробників і користувачів, пропаганда важливості методів безпечного кодування та проведення навчання з виявлення та пом’якшення ін’єкційних атак можуть ще більше посилити загальну безпеку. Застосовуючи проактивний і комплексний підхід до безпеки, організації можуть ефективно захищатися від ін’єкційних атак і гарантувати цілісність і конфіденційність своїх веб-додатків і конфіденційних даних. сприяння важливості безпечних методів кодування та навчання з виявлення та пом’якшення ін’єкційних атак може ще більше посилити загальну безпеку. Застосовуючи проактивний і комплексний підхід до безпеки, організації можуть ефективно захищатися від ін’єкційних атак і гарантувати цілісність і

конфіденційність своїх веб-додатків і конфіденційних даних. сприяння важливості безпечних методів кодування та навчання з виявлення та пом'якшення ін'єкційних атак може ще більше посилити загальну безпеку. Застосовуючи проактивний і комплексний підхід до безпеки, організації можуть ефективно захищатися від ін'єкційних атак і гарантувати цілісність і конфіденційність своїх веб-додатків і конфіденційних даних.

Крім того, важливо залишатися пильним і стежити за останніми найкращими методами безпеки та оновленнями. Регулярний моніторинг і виправлення вразливостей програмного забезпечення, впровадження надійних механізмів автентифікації та використання брандмауерів веб-додатків (WAF) також можуть допомогти зменшити ризик ін'єкційних атак. Крім того, навчання розробників і користувачів про небезпеку ін'єкційних атак, просування методів безпечного кодування та проведення ретельного тестування безпеки може сприяти створенню надійного захисту від цих типів атак. Загалом, багаторівневий підхід до безпеки, що включає як превентивний, так і детективний контроль, є важливим для захисту від ін'єкційних атак і збереження цілісності та конфіденційності конфіденційних даних.

Крім того, існує кілька найкращих практик і методів, які можна застосувати для захисту від ін'єкційних атак. Вони включають впровадження безпечного життєвого циклу розробки, який передбачає проведення ретельних перевірок коду та тестування безпеки, а також використання методів безпечного кодування, таких як перевірка вхідних даних і вихідне кодування. Брандмауери веб-додатків можуть бути розгорнуті для виявлення та блокування спроб зловмисних ін'єкцій, тоді як регулярні виправлення безпеки та оновлення повинні застосовуватися до основного програмного забезпечення та фреймворків. Навчання розробників і користувачів щодо ризиків і методів ін'єкційних атак також є важливим для створення середовища, яке турбується про безпеку. Крім того, впровадження належного контролю доступу та принципів найменших привілеїв може обмежити потенційний вплив ін'єкційних

атак, гарантуючи, що користувачі та процеси мають лише необхідні дозволи для виконання своїх завдань. Регулярний моніторинг і журнал активності програми може допомогти виявити будь-яку підозрілу чи зловмисну поведінку та реагувати на неї. Поєднуючи ці профілактичні заходи та дотримуючись проактивного підходу до безпеки, організації можуть значно знизити ризик ін’єкційних атак і захистити свої веб-додатки та дані від використання.

Ін’єкційні атаки становлять значну загрозу для веб-додатків і можуть призвести до несанкціонованого доступу, витоку даних та інших проблем безпеки. SQL-ін’єкція та міжсайтовий сценарій (XSS) є двома поширеними типами ін’єкційних атак, націлених на запити бази даних і введення користувачами відповідно. Ці атаки використовують уразливості в механізмах перевірки та обробки вхідних даних програми, що дозволяє впроваджувати та виконувати шкідливий код або команди. Щоб зменшити ризик ін’єкційних атак, розробники та організації повинні дотримуватися методів безпечного кодування, таких як параметризовані запити та підготовлені оператори, щоб запобігти ін’єкції SQL. Вони також повинні запровадити перевірку вхідних даних і методи кодування вихідних даних, щоб запобігти атакам XSS. Крім того, використання брандмауерів веб-додатків, регулярних оновлень безпеки, контролю доступу, і моніторинг може підвищити рівень безпеки та стійкість проти ін'єкційних атак. Постійне навчання та підвищення обізнаності серед розробників і користувачів мають вирішальне значення для виховання усвідомлення безпеки та зниження ймовірності успішних ін’єкційних атак. Застосовуючи комплексний і проактивний підхід до безпеки, організації можуть ефективно захистити свої веб-додатки та захистити конфіденційні дані від руйнівних наслідків ін’єкційних атак.

В цілому, для захисту від Injection-атак необхідно використовувати комплексні техніки та інструменти, які дозволять запобігти внесенню зловмисного коду в веб-застосунок. Важливо також пам'ятати про потенційні наслідки вразливостей та забезпечувати своєчасне оновлення систем та

програмного забезпечення для запобігання використання вразливостей зловмисниками.

1.4 XSS атаки

XSS (Cross-Site Scripting) є однією з найбільш розширених атак на веб-застосунки. Ця атака відбувається у внесенні зловмисного скрипту на веб-сторінку, яка може бути виконана в браузері користувача. Зловмисник може використовувати цей скрипт для отримання доступу до конфіденційних даних користувача, таких як паролі, кредитні картки тощо.

Залежно від типу XSS-атаки, вона може бути категоризована як збережений XSS, відображений XSS і XSS на основі DOM.

Збережена атака XSS відбувається у внесенні зловмисного скрипту в базу даних або файл, який виконується при відображених веб-сторінках. Відображена атака XSS вимагає внесення зловмисного скрипту в параметр URL, який виконується при першому запиті до сервера.

XSS-атака на основі DOM відбувається на стороні клієнта, коли зловмисник використовує вразливість у коді JavaScript на стороні клієнта.

Щоб запобігти вразливостям XSS, розробники веб-додатків повинні дотримуватися безпечних методів кодування, таких як перевірка введення та кодування виводу. Вони повинні бути обережними при поводженні з користувачем

Слід проводити регулярне тестування безпеки та оцінку вразливостей, щоб виявити та усунути будь-які вразливості XSS. Це включає обидва засоби автоматичного сканування безпеки

XSS (Cross-Site Scripting) атаки продовжують бути поширеною проблемою безпеки веб-додатків. Для ефективної боротьби з XSS-атаками вкрай важливо

застосовувати багаторівневий підхід, який стосується як запобігання, так і пом’якшення.

Що стосується профілактики, перевірка вхідних даних є важливою. Усі введені користувачем дані мають бути ретельно перевірені на стороні сервера, щоб переконатися, що вони відповідають очікуваному формату та обмеженням. Це включає перевірку на наявність потенційно шкідливих символів і шаблонів. Крім того, вихідне кодування слід застосовувати під час відтворення створеного користувачем вмісту на веб-сторінках, щоб запобігти його інтерпретації як коду.

Контекстне кодування виводу є особливо важливим, оскільки різні контексти (наприклад, HTML, JavaScript, CSS) мають різні правила екранування символів. Використовуючи відповідні методи кодування, специфічні для контексту, у якому відтворюються дані, можна значно зменшити ризик атак XSS.

Фреймворки веб-додатків часто забезпечують вбудовані механізми для перевірки вхідних даних і кодування вихідних даних. Використання цих функцій може спростити впровадження методів безпечного кодування та зменшити ймовірність появи вразливостей.

Політика безпеки вмісту (CSP) — ще одна ефективна техніка захисту від атак XSS. Це дозволяє веб-розробникам визначати набір політик, які обмежують типи вмісту, який можна завантажити та виконати на веб-сторінці. Вказуючи надійні джерела для сценаріїв, таблиць стилів та інших ресурсів, CSP може запобігти виконанню зловмисних сценаріїв, введених через уразливості XSS.

Регулярне тестування безпеки, включаючи сканування вразливостей і тестування на проникнення, має вирішальне значення для виявлення та усунення вразливостей XSS. Інструменти автоматичного сканування можуть допомогти виявити типові вразливості, тоді як ручне тестування дозволяє більш глибоко аналізувати та виявляти складні проблеми. Важливо проводити ці тести регулярно, особливо під час внесення змін у програму чи її середовище.

Програми підвищення рівня безпеки та навчальні програми є важливими для навчання розробників, адміністраторів і кінцевих користувачів про ризики

та наслідки атак XSS. Підвищуючи обізнаність щодо методів безпечного кодування, важливості перевірки вхідних і вихідних даних, а також важливості оновлення програмного забезпечення, люди можуть стати більш пильними у запобіганні вразливостям XSS.

Підсумовуючи, боротьба з атаками XSS вимагає поєднання превентивних заходів, таких як перевірка вхідних даних і кодування вихідних даних, а також методи пом’якшення, як-от політика безпеки вмісту. Регулярне тестування безпеки, а також навчання та обізнаність користувачів є ключовими елементами підтримки безпечного середовища веб-додатків. Використовуючи ці методи, організації можуть значно знизити ризик атак XSS і захистити цілісність і конфіденційність своїх даних.

Обізнаність і освіта користувачів відіграють вирішальну роль у запобіганні атакам XSS. Користувачів слід поінформувати про ризики, пов’язані з натисканням невідомих посилань, завантаженням підозрілих файлів і обміном конфіденційною інформацією на ненадійних веб-сайтах. Регулярний

Нарешті, підтримання надійної безпеки вимагає постійного моніторингу та реагування на інциденти. Організації повинні мати механізми для своєчасного виявлення XSS-атак і відповіді на них. Це включає впровадження систем виявлення та запобігання вторгненням, ведення журналів і моніторинг дій веб-додатків, а також наявність

Постійно адаптуючи заходи безпеки, отримуючи інформацію про нові загрози та розвиваючи культуру обізнаності про безпеку, організації можуть краще захистити себе від атак XSS і забезпечити безпеку та цілісність своїх веб-додатків і даних користувачів.

Щоб захиститися від XSS-атак, необхідно використовувати валідацію вхідних даних і екранування спеціальних символів, які можуть бути використані для внесення зловмисного коду в систему. Також можна використовувати політику безпеки вмісту (CSP) та файли cookie лише HTTP для додаткового захисту від XSS-атак.

Загалом, захист від XSS-атак є кількістю елементів безпеки веб-застосунків. Використання найкращих практичних заходів безпеки, таких як перевірка вхідних даних і захист від спеціальних символів, допомагає зменшити ризик XSS-атак і зберегти конфіденційність даних користувачів.

Використання бібліотек безпеки та фреймворків, які пропонують вбудований захист XSS, також може допомогти зменшити ризик. Наприклад, фреймворки веб-додатків, такі як Django та Express.js, забезпечують вбудовані механізми для обробки введених і виведених даних користувачами

Необхідно проводити регулярні аудити безпеки та тестування на проникнення, щоб виявити та усунути будь-які потенційні вразливості XSS. Крім того, можна реалізувати заголовки безпеки, такі як Content Security Policy (CSP) і X-XSS-Protection, щоб забезпечити додатковий рівень захисту від XSS-атак шляхом визначення дозволених джерел вмісту та застосування суворих політик для виконання сценаріїв.

Навчання та обізнаність користувачів також є життєво важливими для запобігання атакам XSS. Користувачів слід поінформувати про ризики натискання підозрілих посилань або відвідування ненадійних веб-сайтів. Веб-браузери та програмне забезпечення безпеки повинні постійно оновлюватися, щоб користуватися перевагами останніх виправлень безпеки та функцій.

Застосовуючи проактивний і багаторівневий підхід, який включає безпечні практики кодування, перевірку вхідних даних і санітарну обробку, вихідне кодування, тестування безпеки та навчання користувачів, організації можуть значно знизити ризик атак XSS і захистити свої веб-додатки та конфіденційну інформацію користувачів.

Один зі способів захисту від XSS-атак - використання Content Security Policy (CSP). Це механізм, який дозволяє веб-розробникам вказати, які ресурси можуть бути завантажені на сторінці та які типи коду можуть бути виконані. CSP дозволяє обмежити використання JavaScript, CSS та інших ресурсів, що допомагає зменшити ризик XSS-атак.

Незважаючи на зусилля щодо запобігання атакам XSS, важливо залишатися пильним і бути в курсі нових методів атак. Уразливості системи безпеки можуть виникати через різні канали, включаючи сторонні бібліотеки, вміст, створений користувачами, і ненадійні джерела. Таким чином, постійний моніторинг, регулярні аудити безпеки та оперативне усунення відомих вразливостей мають вирішальне значення для підтримки безпечної веб-програми.

Окрім впровадження заходів безпеки на стороні сервера, користувачі також відіграють певну роль у захисті себе від атак XSS. Користувачам важливо бути обережними, натискаючи посилання, особливо ті з невідомих або підозрілих джерел. Оновлення веб-браузерів і програмного забезпечення безпеки може допомогти зменшити ризик атак XSS, скориставшись перевагами останніх виправлень безпеки та функцій.

Загалом для запобігання атакам XSS потрібен комплексний і багаторівневий підхід, який передбачає безпечне кодування, перевірку вхідних даних і санітарну обробку, вихідне кодування, тестування безпеки, навчання користувачів і постійний моніторинг. Впроваджуючи ці заходи, організації можуть зменшити ймовірність уразливості XSS і забезпечити безпеку та цілісність своїх веб-додатків і даних користувачів.

Ще один спосіб захисту - використання HTTP-only cookies. Це практика, що полягає в обмеженні доступу до куків з JavaScript. Якщо куки налаштовані з атрибутом HTTP-only, вони можуть бути доступні лише для сервера, а не для JavaScript. Це знижує ризик XSS-атак, оскільки зловмисник не може отримати доступ до куків, які містять конфіденційні дані.

Також можна використовувати фільтрацію та валідацію вхідних даних, щоб запобігти внесенню зловмисного коду в систему. Це включає перевірку на вході на наявність спеціальних символів та форматування вхідних даних залежно від типу даних.

У загальному, захист від XSS-атак є важливим елементом безпеки веб-застосунків. Важливо використовувати найкращі практики безпеки, такі як валідація вхідних даних, захист від спеціальних символів та використання механізмів, таких як CSP та HTTP-only cookies, щоб зменшити ризик XSS-атак та зберегти конфіденційність даних користувачів.

2 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ТА РОЗРОБКА СИСТЕМИ ЗАХИСТУ

2.1 Реалізація захисту від XSS атак

Щоб зрозуміти, від чого нам потрібно захиститись, для початку зрозумієм основний сценарій дій зловмисника. Як правило, сценарій можна описати ось такими кроками:

– зловмисник виявляє вразливий веб-сайт, який допускає введення небезпечного коду на свої веб-сторінки. Наприклад, це може бути сайт, де розміщується фальшива реклама або відображається неправдивий вміст;

– зловмисник впроваджує шкідливий код JavaScript (чи інший) у веб-додаток на стороні клієнта. Цей код передається або на веб-браузер потенційної жертви, або на веб-сервер, в залежності від типу XSS-атаки;

– користувач натискає на шкідливе посилання під час перегляду веб-сайту або отримання доступу до веб-сервера;

– зловмисник отримує доступ до конфіденційних облікових даних або особистої інформації жертви через вразливий веб-сайт, обходячи механізм Same Origin Policy (SOP).

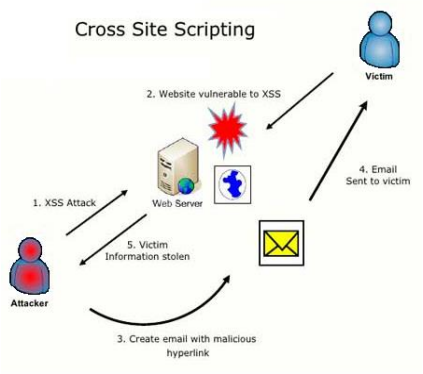


Рисунок 2.1 – Cross Site Scripting

Існує велика кількість робіт, де описуються різноманітні схеми захисту, деякі з них обговорюються нижче.

У роботі [17] пропонується інструмент під назвою QualysGuard, спрямований на зменшення вразливостей і шкоди, які можуть виникати від певних веб-додатків. Цей інструмент виконує функцію веб-сканера, який аналізує вхідні форми у веб-додатку, щоб переконатися, що дані коректно вводяться у відповідних полях. Наприклад, якщо зловмисник змінить URL-адресу форми, QualysGuard виявить цю зміну, просканує URL-адресу і визначить, до якої категорії вразливостей вона належить. Цей інструмент задовольняє потреби як серверної сторони, так і клієнта, забезпечуючи безпеку на обох рівнях.

У роботі [18] пропонується більш автоматизований підхід до виявлення даної вразливості, яка виникає внаслідок некоректного кодування ненадійних даних. Цей метод може виявляти XSS-вразливості на самому початку їх появи, коли інші методи статичного аналізу ще не виявили цих проблем. Основною перевагою цього підходу є його орієнтація на розробників, оскільки вони можуть виявляти та виправляти проблеми без потреби в додатковій експертизі з питань безпеки, що сприяє економії ресурсів. Проте, недоліком цього підходу є його спрямованість лише на один тип атаки XSS.

У дослідженні [19] було проаналізовано продуктивність та можливості сканерів безпеки веб-додатків типу Black Box щодо виявлення збережених SQLI та XSS. Попередні дослідження показали, що такі сканери мають відносно низьку ефективність у виявленні цих двох типів вразливостей. З метою порівняння можливостей різних сканерів був розроблений індивідуальний тестовий стенд. Одна з основних проблем, з якими зіштовхнулися, полягала у виборі відповідних векторів атак для виявлення та експлуатації збережених XSS-вразливостей у сканерах типу Black Box. Окрім цього, існують методи динамічного виявлення, які базуються на імітації поведінки браузерів для більш точного виявлення цих вразливостей.

У роботі [20] пропонується розширений підхід для точного виявлення прихованих точок ін'єкції XSS, що сприяє поліпшенню покриття виявлення. Ця пропозиція включає в себе більш портативну систему, яка спрощує розробку систем. Головною особливістю цієї системи є її орієнтація на інтерпретацію коду JavaScript та відновлення вмісту Ajax для виявлення прихованих точок введення на сторінках через браузер, який діє як веб-сканер без заголовків. Хоча цей підхід використовує фреймворк для виконання атак на сторінки з попередньо встановленими вразливостями, він може бути розглянутий як ще один динамічний метод, який представлений у роботі [21] Було запропоновано репертуар оптимальних векторів атаки, які дозволяють автоматично та динамічно виявляти вразливості XSS у веб-додатках. Шляхом створення цих векторів атак, вони можуть бути виконані автоматично, алгоритми машинного навчання використовуються для підвищення ефективності виявлення вразливостей XSS. Основною перевагою цього підходу є автоматичне формування векторів атак XSS. Однак, варто відзначити, що цей метод був протестований лише на 24 веб-сайтах, тобто його застосування обмежене кількістю сторінок, на яких він був перевірений.

В роботі [22] була запропонована комбінація еволюційного розмиття та моделей висновків для виявлення вразливостей ін'єкції XSS шляхом генерації тестових входів. Згідно їх моделі, ці записи генеруються за допомогою генетичних алгоритмів, які використовують формальну вивчену модель, що дозволяє автоматично створювати вхідні дані для активації екземплярів вразливості. Таким чином, вони запропонували інтелектуальний або розумний нечіткий підхід для виявлення глибоко вбудованих вразливих місць ін'єкції, що сприяє автоматизованому пошуку вразливостей типу 1 XSS і створенню тестових випадків.

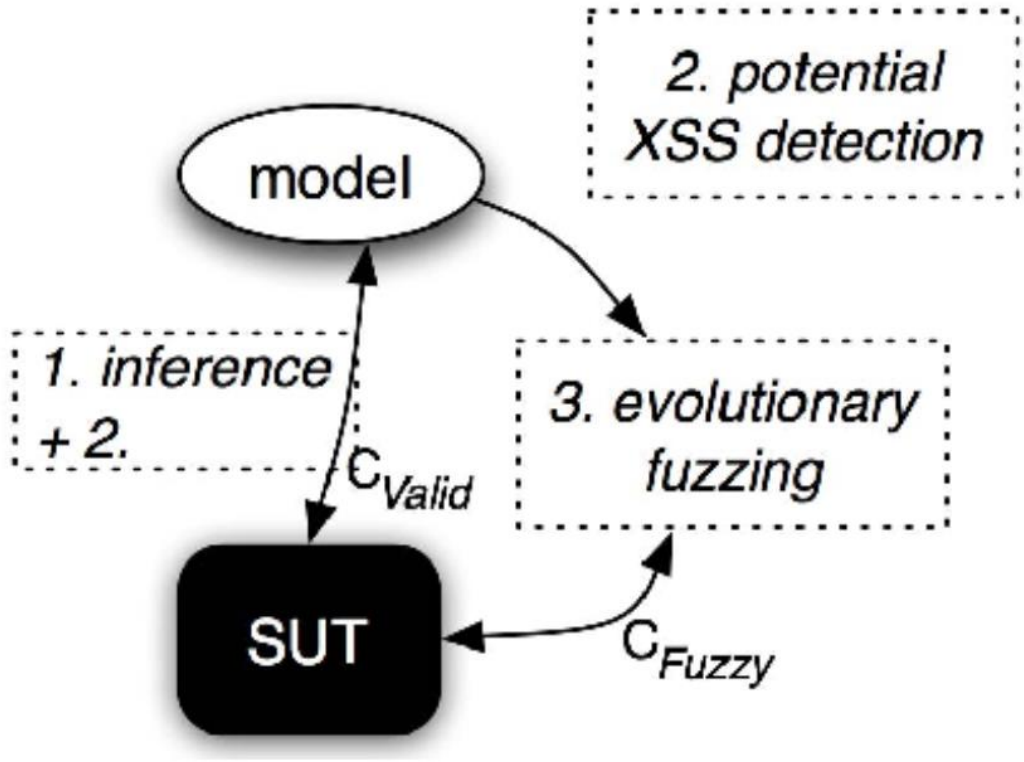


Рисунок 2.2 – Підхід схеми [22]

У роботі [23] було представлено ще один метод виявлення та запобігання вразливості XSS. На першому етапі веб-додаток був перетворений на мову за допомогою конкретного інструменту, що дозволило ідентифікувати вхідні та вихідні змінні, що використовуються в програмі. Це створило тестові випадки і виявило вхідні/вихідні залежності програми, які можуть свідчити про наявність уразливостей. На другому етапі були включені монітори, які автоматично виконували код і перевіряли експлуатацію в режимі реального часу. Це дозволяло виявляти шкідливі послідовності, що виникали під час виконання, шляхом поєднання безпечних послідовностей або за допомогою умовних копій.

Якщо використовувати новий стандарт HTML5, вразливості можуть бути розширені, оскільки веб-сторінки стають більш інтерактивними. У статті [24] була представлена робота, в якій було виявлено 14 векторів атак XSS, пов'язаних з HTML5, за допомогою систематичного аналізу нових тегів та атрибутів. Зібравши ці дані, було створено набір тестових векторів XSS для реалізації

динамічного інструменту виявлення вразливостей XSS, зосередженого на системах веб-пошти. Шляхом застосування цього інструменту до деяких популярних систем веб-пошти вдалося знайти вразливі місця XSS, які можуть бути використані.

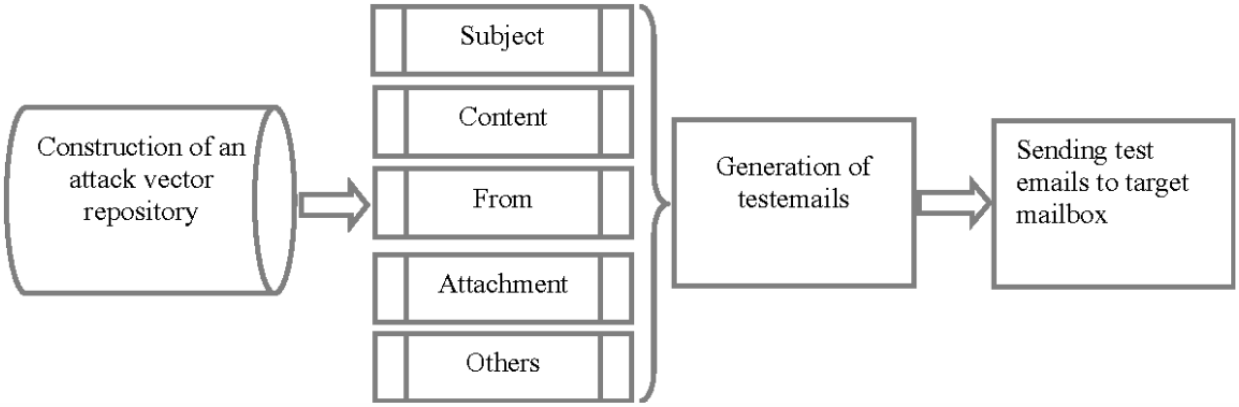


Рисунок 2.3 – Архітектура інструменту з статті [24]

У роботі [25] був представлений інструмент сканування, який відрізняється від [24], широко використовуваного і відстежуваного в сучасних веб-програмах, оскільки він виявляє і перевіряє лише вразливості типу DOM XSS. За словами авторів, їх пропозиція не призводить до помилкових спрацювань, а також є масштабованою. Це забезпечує базу даних векторів атак, де користувачі можуть шукати на своєму веб-сайті та виправляти вразливі місця, що сприяє створенню безпечнішого Інтернету. Цей інструмент є функціональним, оскільки він здійснює ретельний аналіз даних, використовуючи метод підключення на етапах відстеження та сканування.

Інший з запропонованих інструментів, відомий як DomXssMicro [26] є мікротестом, заснованим на шаблоні, який виникає з представницьких вразливостей. Ця пропозиція складається з шести ортогональних компонентів, відомих як джерело, розповсюдження, перетворення, поглинання, тригер і контекст. З використанням цього інструменту вони перевіряють конкретну властивість XSS на основі DOM за допомогою 175 загальних тестів. Однак це

емпіричне дослідження має на меті оцінити засоби виявлення XSS типу DOM, включаючи відкриті і комерційні. Недоліком цього інструменту є його поточна неповнота.

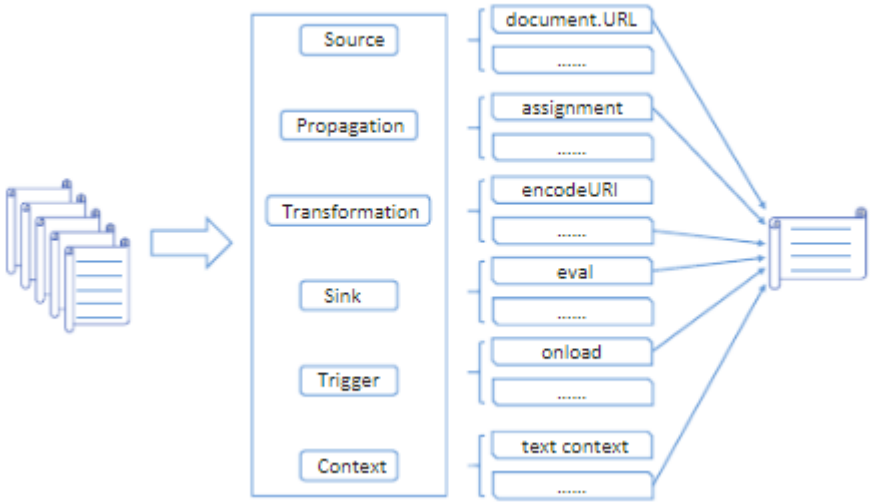


Рисунок 2.4 – Процес побудови мікротестів

У роботі [27] пропонується метод виявлення вразливих до XSS файлів коду у веб-додатках шляхом аналізу тексту. Цей метод використовує процес лексичного аналізу, спеціально налаштований для вилучення текстових характеристик, які в даному випадку представляють собою вихідний код веб-додатків. Кожен файл коду перетворюється на набір унікальних текстових функцій з відповідними частотами, що дозволяє створювати моделі для передбачення вразливостей.

У роботі [28] представлена інша програма, яка використовує аналіз шаблонів для виявлення та пом'якшення вразливостей XSS. Крім того, ця програма також використовує статичний аналіз забруднення. Відповідно до їх дослідження, більшість наявних підходів мають обмеження, пов'язані з хибнопозитивними і негативними результатами. Їхня пропозиція є прототипом, який оцінює набір загальнодоступних даних, проте їхні дослідження сфокусовані лише на веб-програмах, розроблених у мові PHP, оскільки ця мова становить

найбільший відсоток мов програмування для веб-додатків на серверному боці. Метод аналізу реалізовано на мові програмування C#.

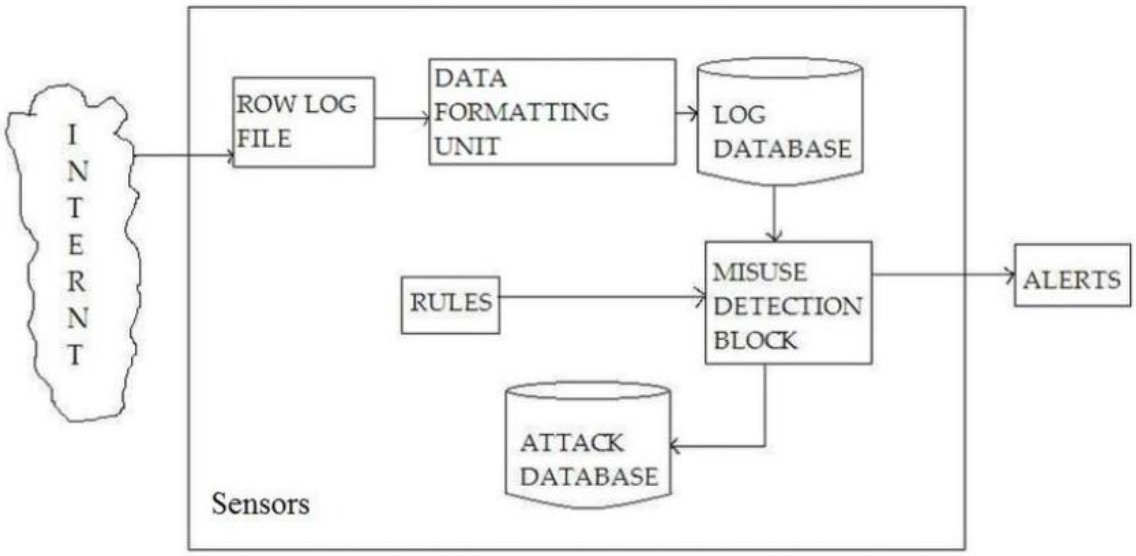


Рисунок 2.5 – Система виявлення вторгнень

Подібно до пропозиції, описаної в [29], існують підходи, які спрямовані на зменшення помилкових позитивних та негативних результатів. Наприклад, детальний аналіз з використанням захисного програмування [30] ґрунтується на чутливому контексті, порівняно з HTML-контекстом, для точного виявлення вразливостей XSS. Застосовуючи цей метод, вдалося досягти точного виявлення уразливостей у вихідному коді програм, що використовують мову програмування PHP, а також може надати вдосконалення для вразливого вихідного коду.

2.2 SQL Injection attack

Розробники можуть запобігти вразливості SQL Injection у веб-додатках, використовуючи параметризовані запити до бази даних із прив’язаними

введеними параметрами та обережне використання параметризованих збережених процедур у базі даних. Існують інструменти для виявлення вразливостей, як приклад, AMNESIA.

AMNESIA — це інструмент, який виявляє та запобігає атакам SQL-ін'єкцій шляхом поєднання статичного аналізу та моніторингу часу виконання. Емпіричне оцінювання показало, що AMNESIA ефективна проти SQL-ін'єкції.

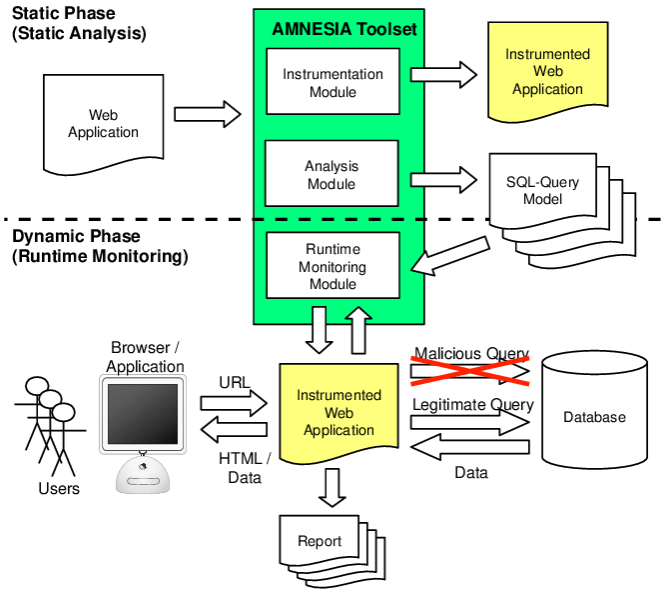


Рисунок 2.6 – Загальний огляд AMNESIA.

У техніці [31] використовується підхід на основі моделі для виявлення незаконних запитів до їх виконання в базі даних. У своїй статичній частині ця техніка використовує програмний аналіз для автоматичної побудови моделі законних запитів, які може генерувати додаток. У своїй динамічній частині метод використовує моніторинг під час виконання для перевірки динамічно створених запитів і перевірки їх на статично побудовану модель. Інструмент AMNESIA було перевірено сім додатків, під час оцінки основною ціллю були досліджувані програми з великою кількістю як легітимних, так і зловмисних вхідних даних і виміряли, скільки атак техніка виявила та запобігла. Результати дослідження показують, що AMNESIA змогла зупинити всі спроби атак без

генерації помилкових спрацьовувань. метод використовує моніторинг під час виконання для перевірки динамічно створених запитів і перевірки їх на статично побудовану модель.

Запобігання атакам SQL Injection за допомогою криптографії та зіставлення шаблонів, ще один досить ефективний спосіб запобігти SQLIA.

Моделі на основі шифрування довели свою ефективність проти SQLIA, перешкоджаючи зловмисникам доступу до автентифікації. Але така модель підриває цілісність таблиць, якщо її використовувати в місцях, відмінних від форми автентифікації. Таким чином, ми використовуємо додатковий рівень безпеки на основі методів зіставлення шаблонів. Така ідея полягає у тому, що вона порівнює тимчасову структуру, згенеровану із запиту користувача, з усіма визначеними доброякісними структурами, створеними з доброякісних запитів, які зазвичай очікуються веб-програмою.

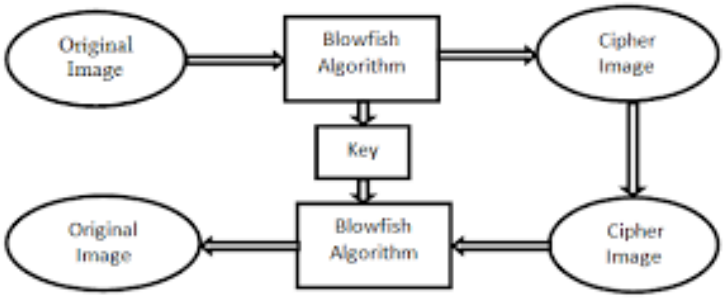


Рисунок 2.6 – Використання алгоритму Blowfish

Запропонована модель використовує алгоритм Blowfish у формі автентифікації, який за результатами моделювання запобігає доступу до автентифікації для всіх типів SQLIA, а після впровадження техніки зіставлення шаблонів Кнута-Морріса-Пратта модель забезпечить запобігання будь-яким новим і існуючим видам SQLIA.

2.3 Розробка системи захисту від XSS-атак

Розробка системи захисту передбачає собою предмет, який вона буде захищати. Тому для початку розробимо веб застосунок. Мною була обрана мова програмування Python та веб-фреймворк Flask, як зручний інструмент для швидкого створення веб-додатку із вбудованими функціями безпеки.

,

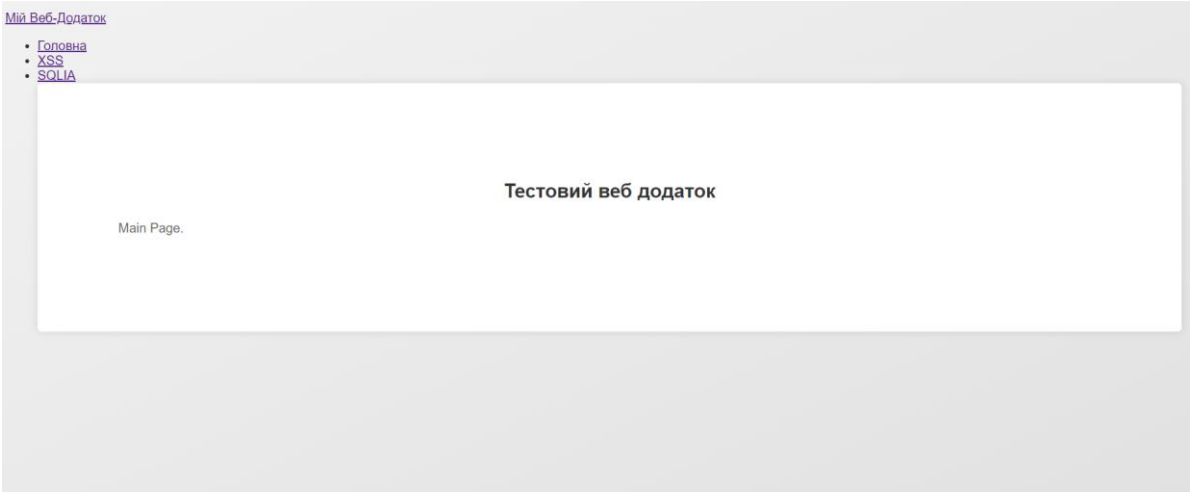


Рисунок 2.7 – Головна сторінка веб-додатку

Далі створимо сторінку, на якій будемо і проводити XSS атаку

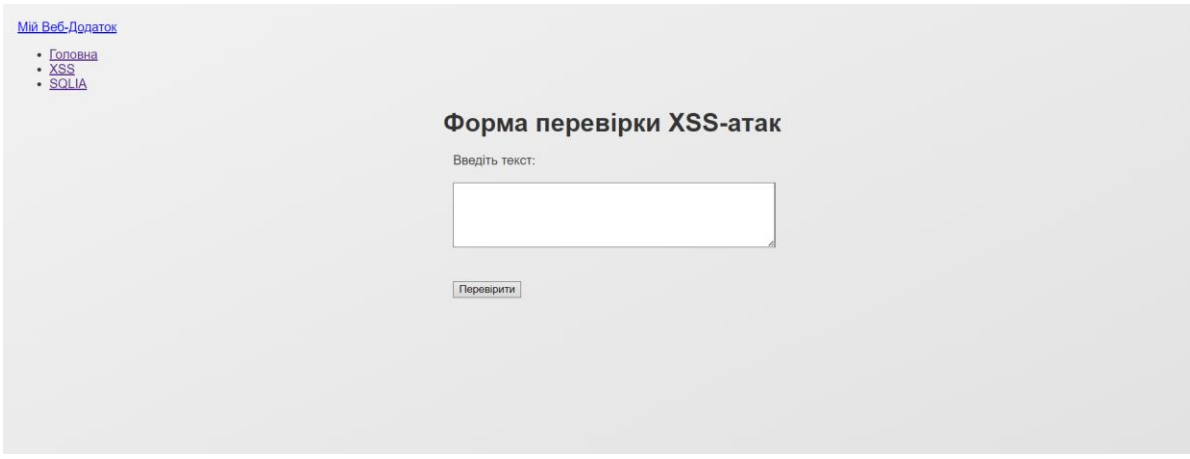


Рисунок 2.8 – Сторінка перевірки XSS-атак

На основі екранування та валідації введених даних розробив систему захисту:

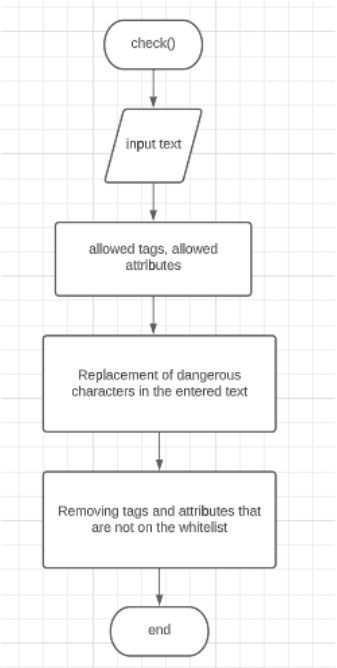


Рисунок 2.9 – Блок-схема захисту від XSS

Алгоритм забезпечує захист від XSS-атак шляхом обробки введеного тексту з форми перед його відображенням на веб-сторінці.

У першому кроці, введений текст отримується з форми. Далі, формується список дозволених тегів і атрибутів, які можуть бути використані безпечно. Наступною дією є заміна небезпечних символів < і > у введеному тексті на безпечні еквіваленти < і >, що запобігає можливості вбудовання HTML-коду.

Далі застосовується регулярний вираз для вилучення тегів і атрибутів, які не знаходяться у списку дозволених. Це забезпечує видалення небезпечних тегів та атрибутів з введеного тексту. На останньому кроці, безпечний текст передається на веб-сторінку для відображення за допомогою шаблону result_XSS.html, де він може бути безпечно відображений користувачеві.

Цей алгоритм дозволяє запобігти XSS-атакам шляхом очищення та фільтрації користувацького вводу перед його відображенням на веб-сторінці. Він забезпечує безпечне відображення введеного тексту, запобігаючи можливість вбудовання шкідливого HTML-коду, і таким чином забезпечує захист від XSS-атак.

2.4 Розробка системи захисту від SQLI-атак

Для початку створимо сторінку, на якій будемо проводити SQLI-атаки. В цьому прикладі мною було вибрано сторінку з формою входу для користувача.

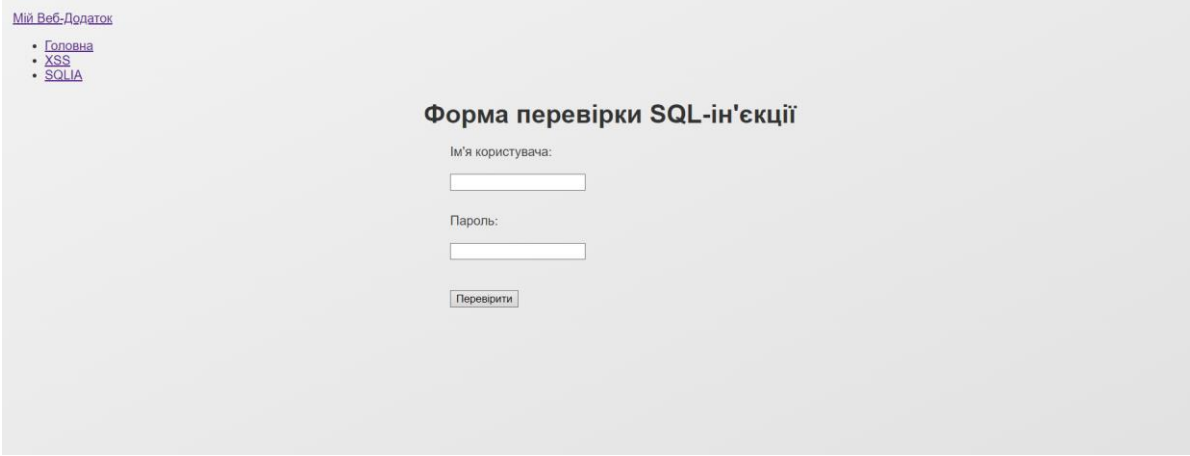


Рисунок 2.10 – Сторінка перевірки SQLI-атак

На цій сторінці ми маємо форму для введення імені користувача (username) та пароля (password). Форма передає дані на сервер за допомогою методу POST і URL-шляху /login.

Цю сторінку можна використовувати для тестування наявності SQLI-атак. Наприклад, можна спробувати ввести наступні значення в полях вводу:

Ім'я користувача: admin' OR '1'='1' -- Пароль: 123456

Якщо система недостатньо захищена, то при спробі ввійти з такими значеннями, можливо отримати несанкціонований доступ до облікових даних, оскільки SQL-запит буде підмінений і виконаний з небезпечною логікою.

Опис алгоритму функції та її допоміжної функції:

– отримання даних з форми. Зчитування значень поля "username" та "password" з форми;

– ініціалізація змінних. result_username та result_password ініціалізуються значеннями змінних "username" та "password" відповідно. message_username та message_password ініціалізуються рядками "SQLIA не виявлено";

– перевірка наявності SQL-ін'єкції: Викликається функція is_sql_injection() для перевірки наявності SQL-ін'єкції в значеннях "username" та "password";

– якщо SQL-ін'єкція виявлена у "username": застосовуються певні заміни символів у змінній result_username, наприклад, заміна одинарної лапки на символ '$', подвійної лапки на символ '@', крапки з комою на символ '%' та подвійного дефісу на символ '*'. Змінна message_username оновлюється рядком "SQLIA Виявлено" разом зі значенням "username";

– якщо SQL-ін'єкція виявлена у "password": застосовуються певні заміни символів у змінній result_password, аналогічно до кроку для "username". Змінна message_password оновлюється рядком "SQLIA Виявлено" разом зі значенням "password";

– повернення шаблону: викликається функція render_template() для відображення шаблону "result_SQLIA.html". Змінні result_username, result_password, message_username та message_password передаються у шаблон для відображення результатів;

– допоміжна функція is_sql_injection(data): перевіряє наявність символів, що вказують на можливу SQL-ін'єкцію (одинарна лапка, подвійна лапка, крапка з комою, подвійний дефіс) у вхідних даних. Повертає значення True, якщо

знайдено хоча б один з небезпечних символів, або False, якщо жодного з них не виявлено.

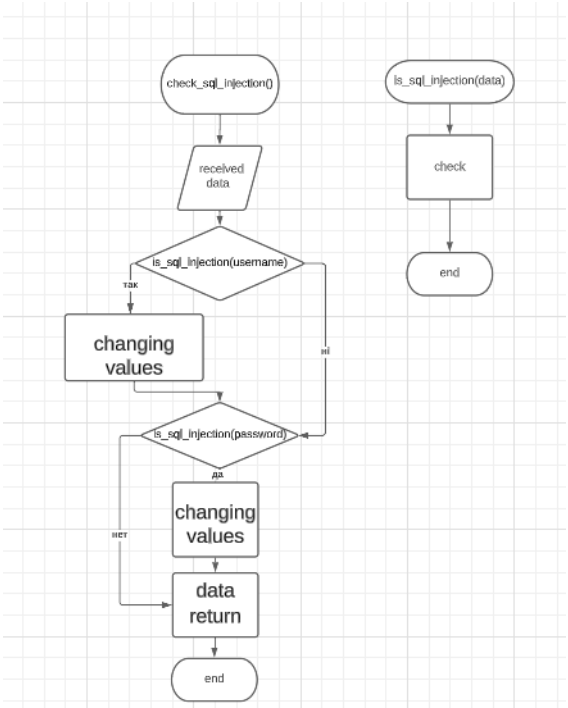


Рисунок 2.11 – Блок-схема захисту від SQLIA

2.5 Висновок

В цілому, розділ "Огляд існуючих рішень та розробка системи захисту" розкриває важливі аспекти забезпечення безпеки систем від XSS- та SQL Injection атак. Розглянуті методи та інструменти надають цінну підсумкову інформацію для інженерів, що допомагає розробити ефективну систему захисту, використовуючи передові техніки та стратегії. Крім того, була розроблена власна система безпеки, яка забезпечує захист шляхом екранування та заміни потенційно шкідливих символів. Ця система виявляє вразливості та виконує автоматичну заміну небезпечних символів на безпечні еквіваленти, що дозволяє запобігти можливим XSS- та SQL Injection атакам. Такий підхід забезпечує додатковий рівень безпеки і знижує ризик компрометації системи через вразливість у введених даних.

3. РОБОТА ЗАХИСТУ

3.1 Структура web додатку

Веб додаток складається з бази даних, клієнтської частини, та серверної частини.

Серверна частина включає налаштування та конфігурацію веб-додатку, встановлює шляхи до різних сторінок та визначає функції-обробники для кожного шляху. Веб-додаток запускається на сервері і слухає вхідні запити від клієнтів.

Функції-обробники виконуються, коли сервер отримує запит від клієнта за відповідним шляхом. У цих функціях виконується логіка обробки запиту, доступ до бази даних та генерація відповіді, яка надсилається клієнту. Функції-обробники можуть отримувати дані з форм, параметрів шляху або запитів AJAX.

У проекті для захисту від XSS-та SQL-ін'єкцій були розроблені функції, які перевіряють введені дані на потенційно небезпечні символи та застосовують необхідні екранування перед використанням у SQL-запитах або відображенням на сторінках.

Загальна структура серверної частини включає модулі, які відповідають за різні функціональні аспекти додатку, такі як роутинг, логіка бізнес-процесів, взаємодія з базою даних та безпека. Крім того, можна використати валідатори для перевірки вхідних даних, логгери для запису подій та інші модулі, які допомагають в розробці та підтримці серверної частини веб-додатку. Важливо було забезпечити безпеку та ефективність серверної частини, ретельно обробляти вхідні дані, виконувати оптимізацію запитів до бази даних та використовувати кешування для покращення продуктивності додатку.

У додатку було використано фрейм-ворк Flask і його стандартні бібліотеки, тому що вони забезпечують надійність і безпеку веб-застосунку.

Одна з використаних бібліотек – Flask-login.

@login_manager.user_loader - декоратор, який визначає функцію завантаження користувача. У проекті функція load_user повертає об'єкт користувача User на основі ідентифікатора, переданого з сесійних куків.

@login_required - декоратор, який забезпечує обмеження доступу до маршруту тільки для автентифікованих користувачів. Якщо користувач не увійшов у систему, він буде перенаправлений на сторінку входу.

У клієнтській частині було розроблено стиль сайту, меню, та форми для перевірки загроз. Основна роль клієнтської частини полягає в тому, щоб надати користувачам зручний і інтуїтивно зрозумілий інтерфейс для взаємодії з веб-додатком. Вона відповідає за відображення даних, взаємодію з користувачем і передачу запитів до сервера для обробки.

Основні функції клієнтської частини включають:

– відображення даних: Клієнтська частина відповідає за відображення даних, отриманих з сервера, у веб-сторінках. Це може включати відображення списків, таблиць, графіків, форм інтерфейсу та іншого контенту;

– взаємодія з користувачем: Клієнтська частина дозволяє користувачам взаємодіяти з додатком. Це може включати заповнення форм, вибір параметрів, натискання кнопок, перегляд деталей елементів тощо. Клієнтська частина виконує перевірку введених користувачем даних на валідність та надсилати запити до сервера для обробки;

– валідація даних: Клієнтська частина виконує перевірку введених даних на коректність і валідність ще до їх передачі на сервер. Це допомагає запобігти надходженню неправильних даних на сервер і забезпечити введення даних в правильному форматі;

– дизайн і стилізація: Клієнтська частина також відповідає за візуальний дизайн і стилізацію вашого веб-додатку. Використання CSS і HTML для створення привабливого і зручного для користувача інтерфейсу.

У базі данних було створенно лише одну таблицю – Users, з полями : id, username, password:

– id - поле id в моделі бази даних використовується для унікальної ідентифікації кожного запису в таблиці. Зазвичай воно визначається як автоінкрементне поле, що означає, що значення id буде автоматично збільшуватись при додаванні нового запису до таблиці;

– username - поле username використовується для збереження ім'я користувача. Це текстове поле, яке має обмеження на довжину і може є унікальним для кожного користувача. Поле username використовується для ідентифікації користувача при вході в систему або взаємодії з його обліковим записом. Воно дозволяє користувачеві вибрати унікальне ім'я, за яким його можна ідентифікувати;

– password - поле password в моделі бази даних використовується для збереження захешованого паролю користувача. Паролі зберігаються у базі даних у захешованому вигляді з метою забезпечення безпеки. Це текстове поле, яке зберігає захешований представлення паролю. Поле password в моделі бази даних не зберігає сам пароль у відкритому вигляді. Замість цього, застосовується функція хешування, яка перетворює пароль на неперевершену послідовність символів, відому як хеш. Цей хеш зберігається у полі password.

3.2 Тестування web застосунку за допомогою інструментів пентестингу

Першим інструментом для пентестингу я обрав Burp Suite.

Burp Suite є потужним інструментом для пентестингу веб-застосунків. Основна його функціональність пов'язана з перехопленням і маніпулюванням мережевим трафіком. Давайте розглянемо кілька основних функцій та налаштувань Burp Suite які будемо використовувати для перевірки веб-застосунка:

Проксі-сервер: Burp Suite може працювати як проксі-сервер, який дозволяє перехоплювати і аналізувати HTTP-запити та відповіді, що надходять між клієнтом і сервером. Налаштувавши ваш браузер або інші програму, щоб вони використовували Burp Suite як проксі-сервер і перехоплювали їхні запити.

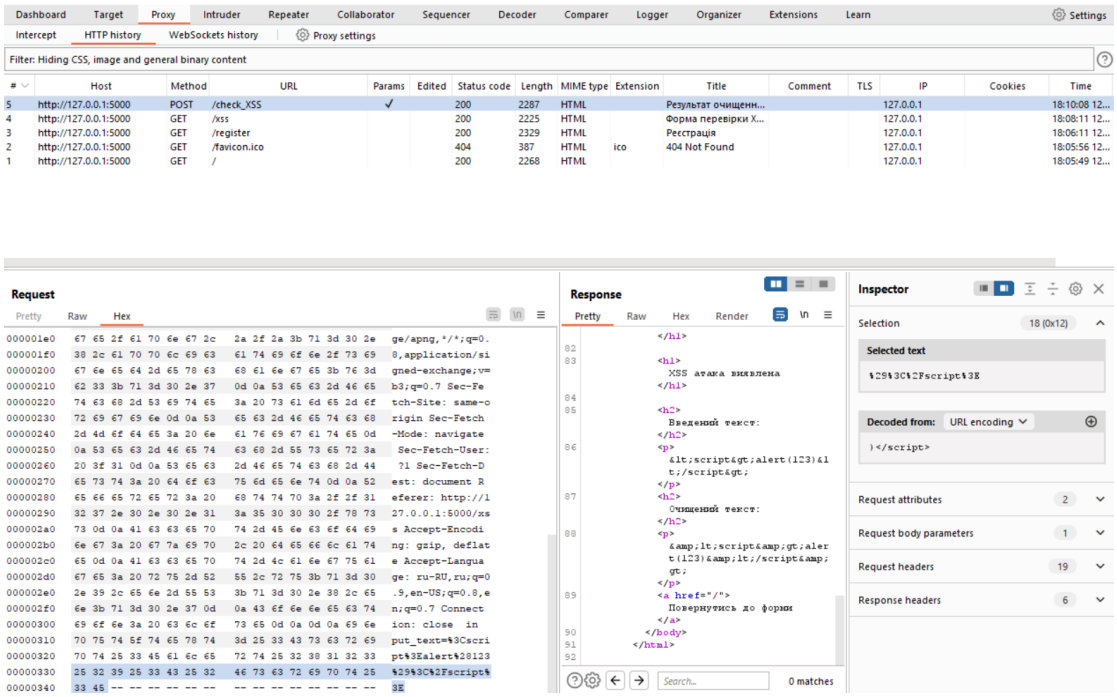


Рисунок 3.1 – Проксі-сервер

На рисунку 3.1 показано вкладку HTTP history, а імено Hex запиту, показує процес заміни спеціальних символів на їх еквіваленти або кодові представлення в контексті HTML-коду. У даному випадку, символи “<”, “>”, та “/”, які використовуються в тегах “<script>” та “</script>”, були замінені на їх кодове представлення, використовуючи сутності HTML. Конкретні кодові представлення, які використовуються, це “&lt”; для “<”, “>” для “>”, та “&” для “/”. Таким чином було забезпечено безпеку і виключено потенційну загрозу.

Intercept: Опція Intercept дозволяє зупиняти та перехоплювати клієнтські запити перед їхнім відправленням на сервер. Це дає можливість аналізувати та

модифікувати запити перед їхньою відправкою, що дозволяє виявляти потенційні вразливості і виконувати інші операції з мережевим трафіком.

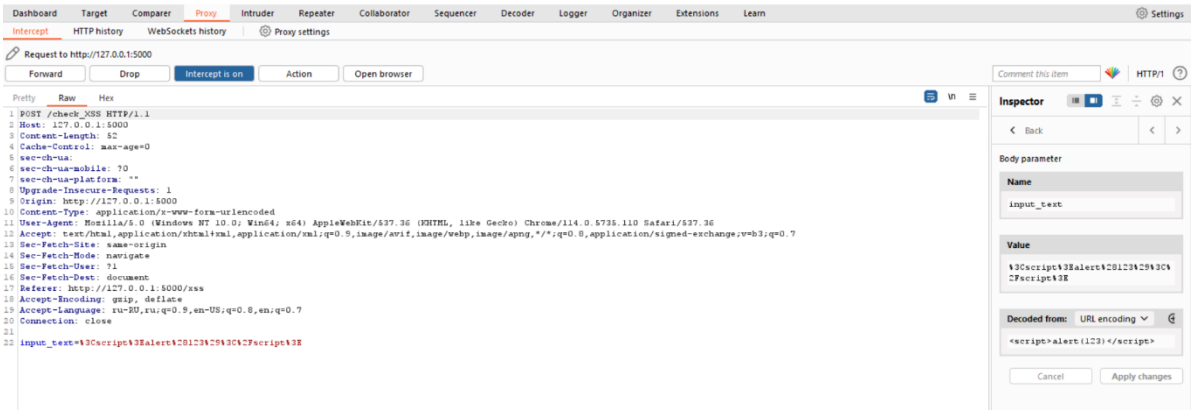


Рисунок 3.2 – Перевірка XSS вразливості (Intercept)

На рисунку 3.2 показано вкладку Intercept в програмі Burp Suite. Коли вкладка Intercept увімкнена, вона призупиняє всі HTTP запити та відповіді, що проходять через неї. Це дозволяє користувачеві переглядати та змінювати дані перед тим, як вони будуть відправлені на сервер.

В цьому випадку на вкладці Intercept показано змінене значення тексту перед відправленням на сервер. Замість початкового значення тексту, було внесено зміни і тепер відображається нове значення. Це може бути корисно для модифікації запитів, наприклад, зміна параметрів, додавання або видалення заголовків тощо, для перевірки реакції сервера на змінені дані або для виконання різних тестових сценаріїв.

Після внесення змін користувач може продовжити процес і відправити змінений запит на сервер.

Site Map: Burp Suite створює карту сайту (Site Map), яка відображає всі запити, які були здійснені під час тестування. Ви можете переглядати та аналізувати цю карту, виявляти потенційні проблеми та знайомитися з структурою веб-застосунку.

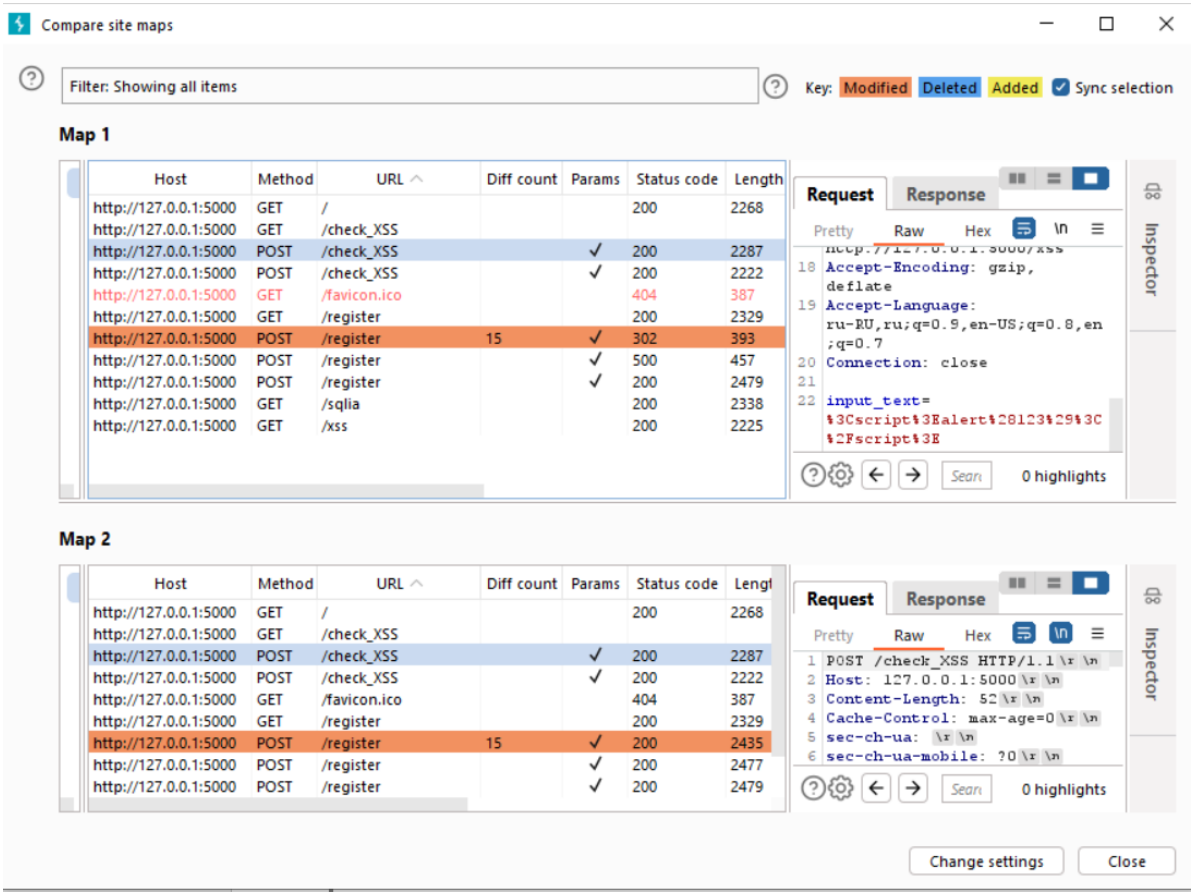


Рисунок 3.3 – Перевірка XSS вразливості (Site map)

На рисунку 3.3 показано мапу сайту. Мапа сайту є списком сторінок, які входять до складу веб-додатку. Вона надає структуроване представлення всіх доступних сторінок на сайті та їхній ієрархічний зв'язок. На рисунку можна побачити змінений текст, а також, що запит не може бути шкідливим.

Repeater: Модуль Repeater дозволяє вам повторювати запити, модифікувати їх та вивчати різні варіанти взаємодії з веб-застосунком. Ви можете вручну змінювати параметри запитів, додавати заголовки, модифікувати дані та багато іншого.

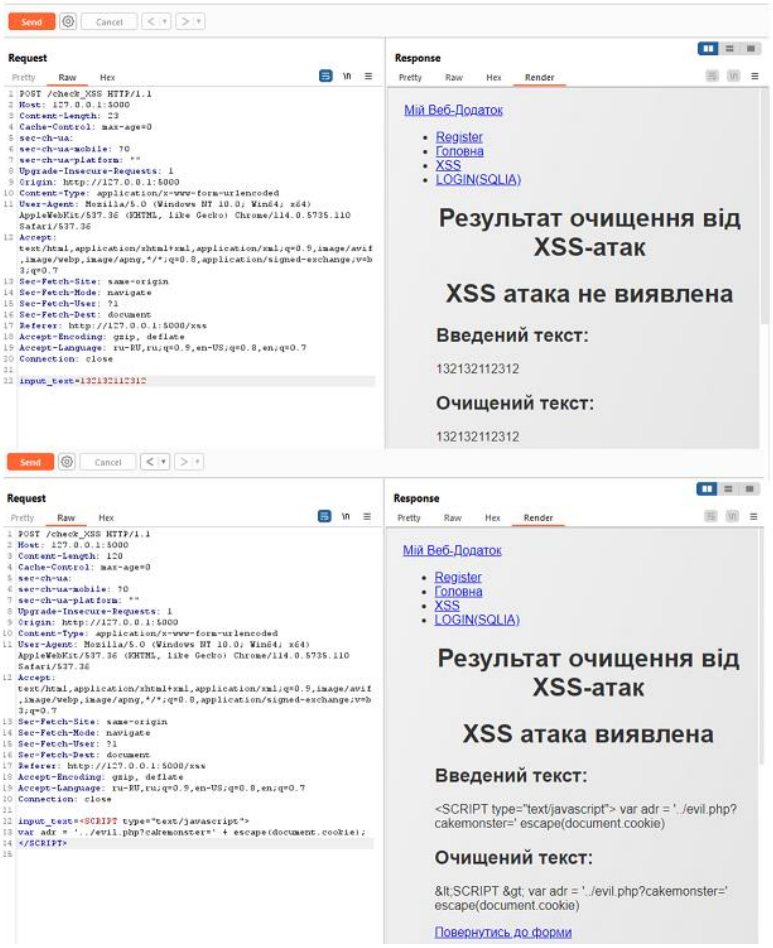


Рисунок 3.4 – Перевірка XSS вразливості (Repeater)

На рисунку 3.4 показано роботу модуля Repeat, який забезпечує повторюваність атак, для більш детального аналізу, а також дає змогу змінювати вхідхі дані при кожній наступній атаці. На рисунку зображено валідний текст, та змінений, який не є валідним. Модуль показує, що перший ввід – атака не виявлена, а другий – виявлена, а також очищений текст.

У Burp Suite Intruder для виконання атак XSS (Cross-Site Scripting) найчастіше використовуються типи атак Sniper і Battering Ram.

Sniper: Цей тип атаки виконує ітерацію по всіх значеннях в одному регіоні, перевіряючи кожне значення на наявність XSS-вразливостей. Він просто перебирає кожне значення зі списку або словника і вставляє його в різні місця в запитах для перевірки, чи відображаються вразливі скрипти.

Battering Ram: Цей тип атаки використовує кожне значення з одного регіону для виконання запиту, що містить потенційну XSS-вразливість. Кожен запит відправляється з різним значенням з регіону, щоб перевірити, чи вразливі скрипти відображаються на сторінках.

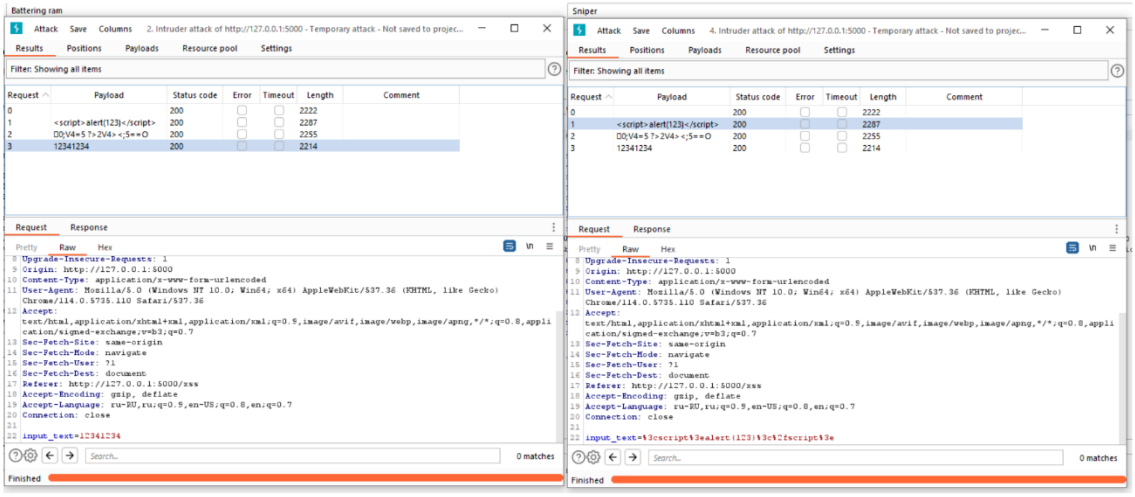


Рисунок 3.5 – Intruder (Sniper and Battering Ram)

На рисунку 3.5 показано роботу модуля Intruder в рамках двох типів атак: "Sniper" та "Battering Ram". Зокрема, тип атаки "Sniper" був використаний для проведення трьох атак, проте жодна з них не була успішною. В процесі атаки "Sniper" всі шкідливі символи були виявлені та замінені, завдяки чому не допущено передачу небезпечних даних на серверну частину. При використанні атаки "Sniper", яка є однією з функціональних можливостей модуля, система була успішно захищена від потенційних загроз.

В процесі використання типу атаки "Battering Ram" було здійснено три спроби, але жодна з них не мала успіху. Усі небезпечні символи, що могли бути використані для атаки, були виявлені та відповідно замінені.

Це свідчить про ефективність застосування модуля, оскільки він успішно виявляє потенційні загрози та запобігає їхній передачі на серверну частину. Використання типу атаки "Battering Ram" є одним з методів забезпечення

безпеки системи, оскільки він дозволяє перехоплювати та перевіряти вхідні дані на наявність шкідливих символів.

Ці результати підкреслюють важливість використання таких інструментів, як модуль Intruder, для забезпечення безпеки веб-додатків. Виявлення та блокування шкідливих символів допомагає запобігти можливим атакам і зберегти систему від вразливостей.

Рисунок 3.6 – Перевірка SQL вразливості (Intercept)

На рисунку 3.6 під час перехоплення запиту перевіряється наявність SQL ін'єкцій в полях username та password. Якщо виявляється потенційна загроза, ключові символи або символи, які можуть використовуватись для ін'єкцій, замінюються на інші символи. Це робиться з метою запобігання можливих атак і забезпечення безпеки системи.

Заміна ключових символів на інші допомагає уникнути вразливостей, які можуть бути використані для впровадження шкідливого SQL-коду. Цей підхід є одним з методів захисту від SQL-ін'єкцій, який дозволяє блокувати потенційно шкідливі запити і забезпечувати цілісність та безпеку бази даних.

Ці заміни символів відбуваються під час перехоплення запиту від клієнта до сервера. Коли запит перехоплюється, програма перевіряє поля username та

password на наявність можливих SQL-ін'єкцій шляхом аналізу символів у цих полях.

Наприклад, якщо поле містить символ одинарної лапки, який може використовуватись для впровадження SQL-коду, він замінюється «$», щоб унеможливити його використання для атаки.

Цей підхід допомагає уникнути вразливостей, пов'язаних з SQL-ін'єкціям

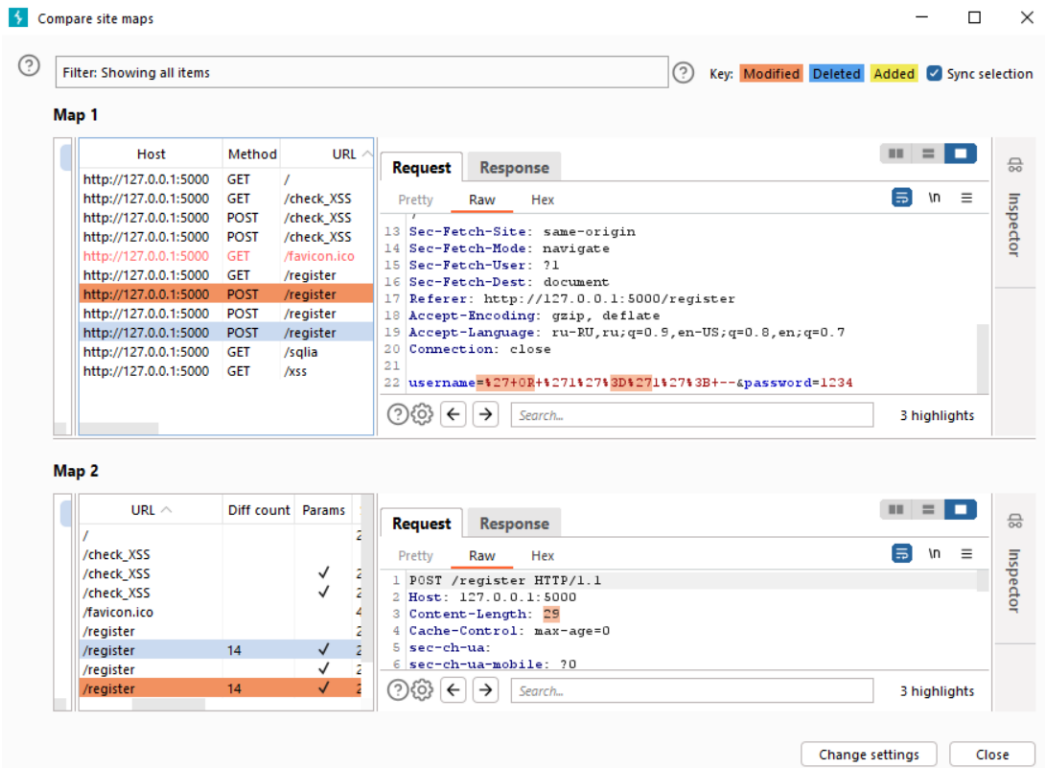


Рисунок 3.7 – Перевірка SQL вразливості (Site map)

На рисунку 3.7 показана робота модуля Site map, який дозволяє сканувати веб-додаток для виявлення різних полів і сторінок, а також визначати їх рівень потенційної загрози.

Під час роботи Site map веб-додаток аналізується з точки зору безпеки, і деякі полі та сторінки можуть бути віднесені до категорії "небезпечних". Проте, важливо зазначити, що не всі небезпечні поля або сторінки є критичними або мають значний потенціал для атак.

У даному випадку на рисунку 3.7 було виявлено кілька небезпечних полів, але вони не були визнані як велика загроза. Це може означати, що ці поля можуть мати деякі потенційні вразливості або можуть використовуватись для певних атак, але ризик їх використання є незначним або обмеженим.

Такий результат виявлення небезпечних полів дозволяє розпізнавати потенційні вразливості веб-додатку та проводити подальші дослідження та заходи для підвищення безпеки. Це важливий етап процесу тестування на проникнення та забезпечення безпеки веб-додатків.

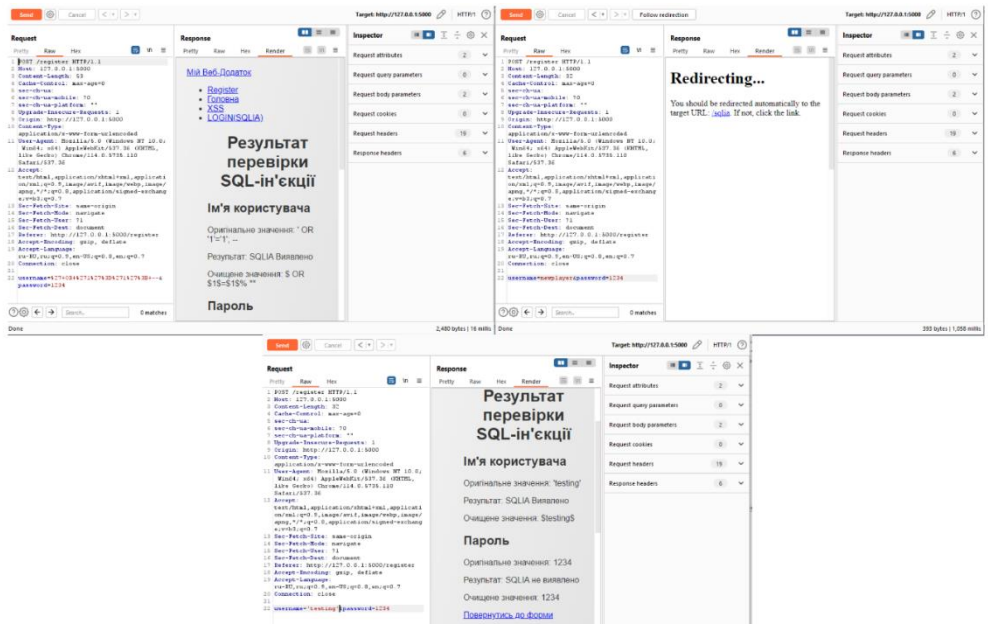


Рисунок 3.8 – Перевірка SQL вразливості (Repeater)

На рисунку 3.8 представлений модуль Repeat, який використовується для повторення атак з метою детального аналізу і дозволяє змінювати вхідні дані при кожній наступній атакі. У даному контексті розглядається атака типу SQLIA (SQL Injection Attack), яка спрямована на вразливість системи управління базами даних.

Модуль Repeat дозволяє виконувати повторні атаки з різними варіаціями вхідних даних, щоб виявити наявність SQL-ін'єкцій у веб-додатку. Це забезпечує більш детальне тестування та аналіз безпеки. Під час проведення атаки типу

SQLIA, модуль Repeat може змінювати значення полів, передаваних у запиті, для перевірки реакції веб-додатку на різні вхідні дані.

Для виконання SQL Injection атак використовуються два типи атак - Sniper і Battering Ram.

Тип атаки Sniper включає ітерацію по всіх значеннях в одному регіоні, де кожне значення перевіряється на вразливість SQL Injection. Цей тип атаки просто перебирає всі значення зі списку або словника і виконує запити з цими значеннями.

Тип атаки Battering Ram використовує кожне значення з одного регіону для виконання запиту, який може містити вразливість SQL Injection. Кожен запит відправляється з різним значенням з регіону, що дозволяє перевірити, які значення спричиняють вразливість та як вона проявляється.

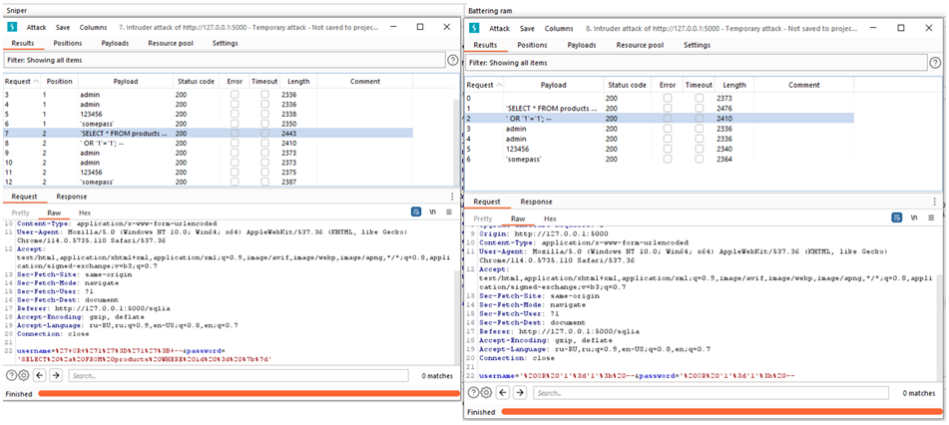


Рисунок 3.9 – Intruder (Sniper and Battering Ram)

На зображенні 3.9 можна побачити роботу модуля Intruder у рамках атак "Sniper" та "Battering Ram" на SQL-ін'єкції (SQL Injection Attack).

Атака "Sniper" використовується для виявлення SQL-ін'єкційних вразливостей у веб-додатку. Модуль Intruder перебирає різні значення параметрів запиту, вставляючи їх у потенційно вразливі місця запиту, де можлива SQL-ін'єкція. Після цього Burp Suite перевіряє, чи вдається виконати шкідливі SQL-запити та чи отримуємо результати, що несподівані для

нормального функціонування додатку. Якщо вразливість виявляється, Burp Suite реєструє інформацію про неї та зберігає результати атаки.

Атака "Battering Ram" також використовує модуль Intruder для виявлення SQL-ін'єкційних вразливостей. Кожне значення з визначеного діапазону вставляється у запит, що містить потенційну SQL-ін'єкцію. Таким чином, Burp Suite надсилає кілька запитів з різними значеннями, щоб перевірити, чи вдається виконати шкідливі SQL-запити та чи виявляються вразливості на цільовій сторінці.

Обидва типи атак, "Sniper" і "Battering Ram", дозволяють Burp Suite виявляти SQL-ін'єкційні вразливості шляхом вставки шкідливих SQL-запитів та аналізу реакції сторінки на такі запити.

3.3 Робота захисту від XSS-атак

Щоб перевірити чи працює захист, було вибрано декілька скриптів, та одне валідне повідомлення:

– <script>alert(123)</script> Один з найбільпопулярних скриптів для перевірки.

– Cookie Grabber <SCRIPT type="text/javascript"> var adr = '../evil.php?cakemonster=' + escape(document.cookie); </SCRIPT>

Валідне повідомлення:

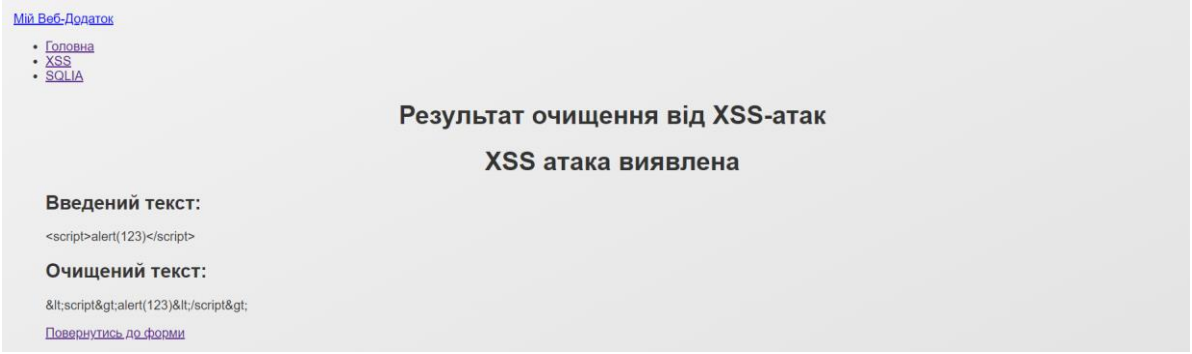


Рисунок 3.10 – Результат виявлення загрози 1

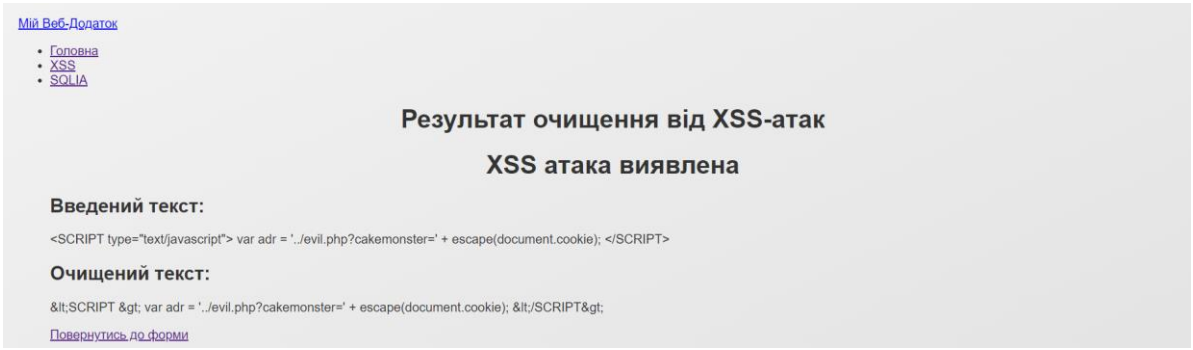


Рисунок 3.11 – Результат виявлення загрози 2

Рисунок 3.12 – Результат виявлення загрози

3.4 Робота захисту від SQLI-атак

Щоб перевірити чи працює захист, було вибрано декілька скриптів, та одне валідне повідомлення.

Для username використаємо наступні імена:

'SELECT * FROM products WHERE id = {}';

' OR '1'='1'; --;

admin.

Для password використаємо:

admin;

123456;

'somepass'.

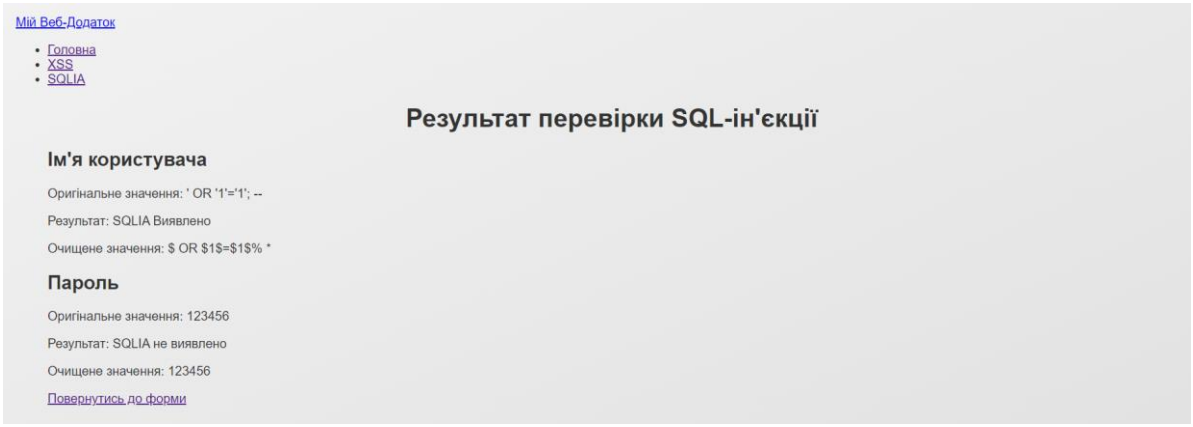


Рисунок 3.13 – Результат виявлення загрози у імені, але не у паролі

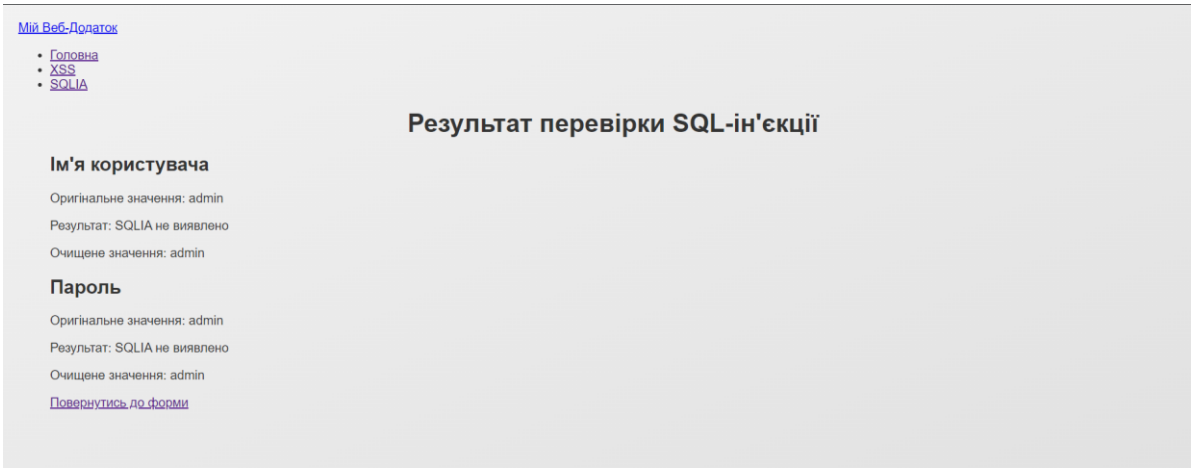


Рисунок 3.14 – Результат без загроз

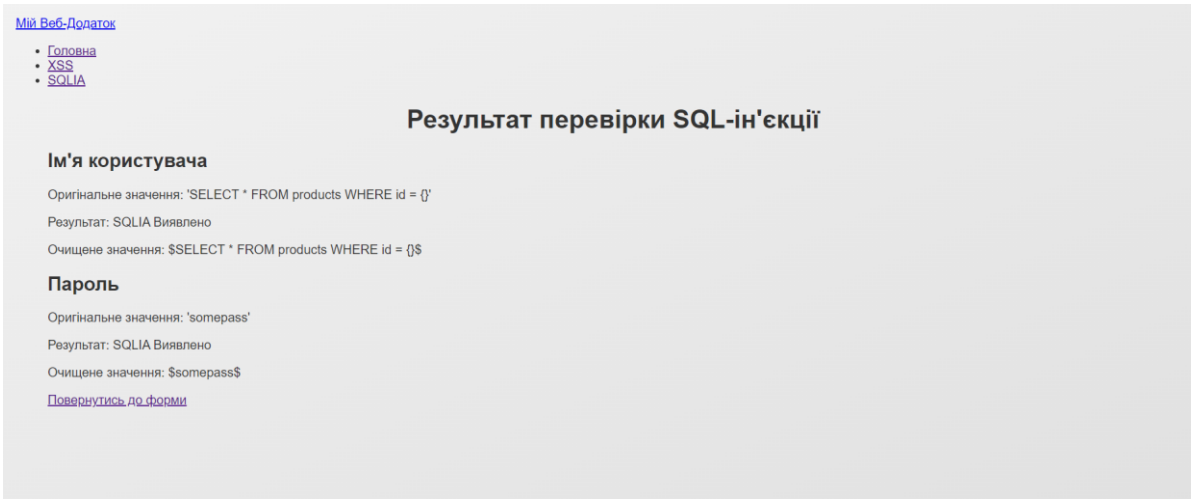


Рисунок 3.15 – Результат виявлення загрози у імені, та у паролі

3.5 Висновок

Отже, після проведення практичних перевірок ефективності захисту можна зробити наступний висновок. Розроблена система демонструє здатність виявляти та усувати потенційні загрози, такі як XSS-атаки та SQL Injection. Вона успішно впроваджує методи екранування та заміни потенційно шкідливих символів, що допомагає запобігти вразливостям та забезпечити безпеку веб-додатків. Проте, варто враховувати, що недостатня комплексність захисту може призвести до проблем. В деяких випадках може виникнути недостатній аналіз деяких вразливостей або пропуск певних типів атак. Також система може неадекватно реагувати на нові типи загроз, що вимагає постійного оновлення бази знань та додавання нових функціональних можливостей. З метою забезпечення максимального рівня безпеки, важливо продовжувати вдосконалювати систему захисту. Це включає в себе вдосконалення методів виявлення та усунення загроз, додавання нових захисних шарів та постійне оновлення бази знань про потенційні атаки. Такий підхід дозволить системі ефективно захищати веб-додатки та забезпечити надійний рівень безпеки.

ВИСНОВОК

Під час розробки проекту з фокусом на захист від XSS- та SQL-ін'єкцій було створено веб-застосунок на основі мови програмування Python та веб-фреймворку Flask. Він дозволяє користувачам вводити вразливі дані для перевірки, як система обробляє їх і як їх можна захистити. Застосунок включає функцію для захисту від XSS-атак, яка екранує небезпечні символи та вилучає небезпечні HTML-теги та атрибути. Також реалізована функція для захисту від SQL-ін'єкцій, яка перевіряє вхідні дані на наявність потенційно небезпечних символів та забезпечує безпеку бази даних. Результати проекту підкреслюють важливість безпеки програмного забезпечення та нагадують про необхідність враховувати потенційні вразливості, такі як XSS- та SQL-ін'єкції, під час розробки програмного забезпечення. Впровадження основних практик безпеки, таких як перевірка та екранування вхідних даних, використання параметризованих запитів та обмеження привілеїв користувачів, гарантує безпеку програмного забезпечення та захист від зловмисних атак.

Проект демонструє, як потенційні вразливості можуть бути використані зловмисниками для впровадження шкідливого коду або отримання несанкціонованого доступу до системи. Шляхом розробки веб-застосунку з урахуванням цих загроз та застосування відповідних заходів захисту, можна побачити, як система ефективно впорядається з потенційними атаками та забезпечує надійність та безпеку виконання коду.

Висновок проекту підкреслює, що безпека програмного забезпечення є невід'ємною частиною розробки та підтримки будь-якої системи. Розуміння загроз та методів їх запобігання дозволяє розробникам створювати безпечні, надійні та захищені додатки. Захист від XSS- та SQL-ін'єкцій є лише одним аспектом безпеки, і його успішна реалізація допомагає запобігати вразливостям та зберігати конфіденційні дані в безпеці.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. OWASP. Open Web Application Security Project. URL: https://owasp.org/
2. Flask. User's Guide. URL: https://flask.palletsprojects.com/en/2.3.x/
3. Python. Documentation. URL: https://docs.python.org/3/
4. StackOverflow. How to secure Angular JS from XSS atack?. URL: https://stackoverflow.com/questions/33554049/how-to-secure-angular-js-from-xss-atack
5. W3Schools. URL: https://www.w3schools.com/
6. PortSwigger Web Security Academy. Cross-site scripting. URL: https://portswigger.net/web-security/cross-site-scripting
7. PortSwigger Web Security Academy. SQL injection URL: https://portswigger.net/web-security/sql-injection
8. SQL Injection Cheat Sheet. SQL injection. URL: https://owasp.org/www-community/attacks/SQL_Injection
9. XSS Prevention Cheat Sheet. Cross Site Scripting Prevention Chet Sheet. URL: https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html
10. Mozilla Developer Network. Network Security. URL: https://developer.mozilla.org/en-US/docs/Web/Security
11. SQL Injection Prevention. SQL injection. URL: https://www.acunetix.com/websitesecurity/sql-injection/
12. Acunetix. Cross-site-scripting. URL: https://www.acunetix.com/websitesecurity/cross-site-scripting/
13. MySQL.Documentation. URL: https://dev.mysql.com/doc/
14. Mehta T.S. Model to prevent websites from xss vulnerabilities / T. S. Mehta, S. Jamwal // International Journal of Computer Science and Information Technologies. - 2015. - Т. 6, №2, С. 1059–1067. 81

15. Mohammadi M. Automatic web security unit testing: Xss vulnerability detection in 2016 / M. Mohammadi, B. Chu, H. R. Lipford, and E. MurphyHill // IEEE/ACM 11th International Workshop in Automation of Software Test (AST). - 2016. - №5, С. 78–84.

16. Parvez M. Analysis of effectiveness of black-box web application scanners in detection of stored sql injection and stored xss vulnerabilities. / M. Parvez, P. Zavarsky, and N. Khoury // 10th International Conference for Internet Technology and Secured Transactions (ICITST) - 2015. - С. 186–191.

17. Liu Y. A xss vulnerability detection approach based on simulating browser behavior / Y. Liu, W. Zhao, D. Wang, and L. Fu // 2nd International Conference on Information Science and Security (ICISS). - 2015. - С. 1–4.

18. 21 Guo X. Xss vulnerability detection using optimized attack vector repertory / X. Guo, S. Jin, and Y. Zhang // - 2015. - International Conference on CyberEnabled Distributed Computing and Knowledge Discovery. - 2015 - С. 29–36.

19. Duchene F. Xss vulnerability detection using model inference assisted evolutionary fuzzing. / F. Duchene, R. Groz, S. Rawat, and J. L. Richier // IEEE Fifth International Conference on Software Testing, Verification and Validation. - 2012. - С. 815–817.

20. Ruse M. E. Detecting cross-site scripting vulnerability using concolic testing. / M. E. Ruse and S. Basu // 10th International Conference on Information Technology: New Generations. - 2013. - С. 633–638.

21. Dong G. Detecting cross site scripting vulnerabilities introduced by html5 / G. Dong, Y. Zhang, X. Wang, P. Wang, and L. Liu // 11th International Joint Conference on Computer Science and Software Engineering (JCSSE). - 2014. - С. 319–323.

22. Nguyen T. K. Large-scale detection of dom-based xss based on publisher and subscriber model / T. K. Nguyen, S. O. Hwang // International Conference on Computational Science and Computational Intelligence (CSCI). - 2016. - С. 975–980.

23. Pan J. Domxssmicro: A micro benchmark for evaluating dom-based cross-site 82 scripting detection / J. Pan and X. Mao // IEEE Trustcom/BigDataSE/ISPA. - 2016. - С. 208–215.

24. Chaudhary P. Xss detection with automatic view isolation on online social network / P. Chaudhary, B. B. Gupta, S. Yamaguchi // IEEE 5th Global Conference on Consumer Electronics. - 2016. - С. 1–5.

25. . Gupta, M. C. Govil, G. Singh, and P. Sharma, “Xssdm: Towards detection and mitigation of cross-site scripting vulnerabilities in web applications,” in 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Aug. 2015, С. 2010–2015.

26. Gupta M. K. A context-sensitive approach for precise detection of cross-site scripting vulnerabilities / M. K. Gupta, M. C. Govil, G. Singh // 10th International Conference on Innovations in Information Technology (IIT). - 2014. - С. 7–12.

27. AMNESIA: analysis and monitoring for NEutralizing SQL-injection attacks. SQL injection. URL: https://dl.acm.org/doi/10.1145/1101908.1101935

28. Web Security Basics. Web Security. URL: https://www.tutorialspoint.com/web-application-penetration-testing-security/index.asp

29. OWASP Top Ten Project. Penetration testing. URL: https://owasp.org/www-project-top-ten/

30. Secure Development Lifecycle (SDL). Web development. URL: https://www.microsoft.com/en-us/securityengineering/sdl/practices

31. TechTarget. SQL injection. URL: https://www.techtarget.com/searchsoftwarequality/definition/SQL-injection

32. Full Stack Python. Web Security. URL: https://www.fullstackpython.com/web-application-security.html

33. Full Stack Python. Web Development URL: https://www.fullstackpython.com/web-development.html

34. SnykAdvisor. Python MVC security. URL: https://snyk.io/advisor/python/flet-mvc

35. VERACODE. Web Security. URL: https://www.veracode.com/blog/secure-development/how-secure-are-popular-web-frameworks-here-comparison

36. Akamai. Secure database. URL: https://www.linode.com/docs/guides/securing-mysql/

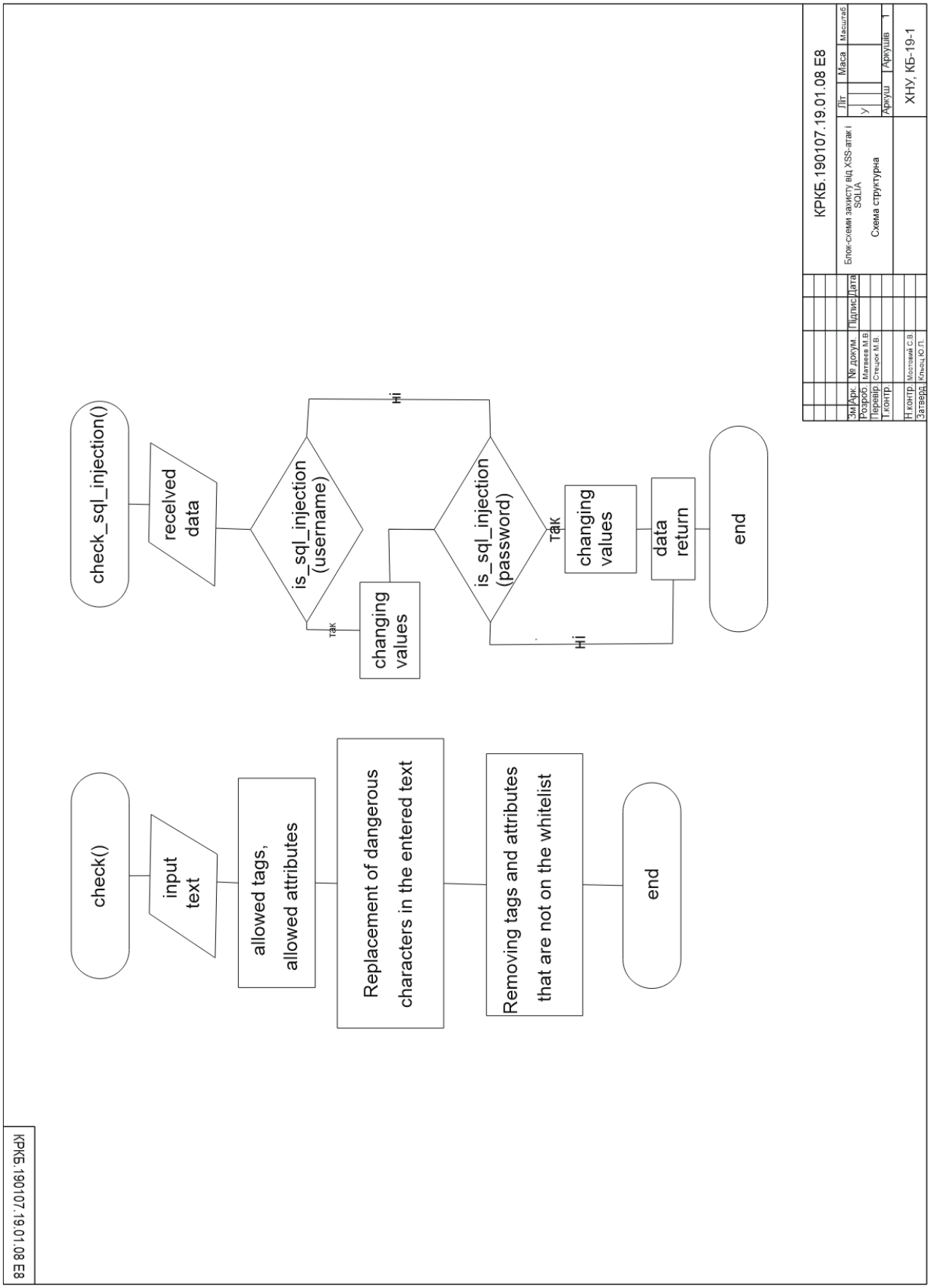
37. Checkio. Web security. URL: https://py.checkio.org/blog/how-to-write-secure-code-in-python/

38. eSecurity Planet. Secure Cross-site scripting. URL: https://www.esecurityplanet.com/endpoint/prevent-xss-attacks/

39. Bright. Secure Cross-site scripting URL: https://brightsec.com/blog/xss/

40. CrashTest Security. XSS attacks prevention. URL: https://crashtest-security.com/xss-attack-p

Додаток А


check_sql_injection()
recelved data
is_sql_injection (username)
так
changing values
ні
is_sql_injection (password)
так
changing values
ні
data return
end
check()
input text
allowed tags, allowed attributes
Replacement of dangerous characters in the entered text
Removing tags and attributes that are not on the whitelist
end
КРКБ.190107.19.01.08 Е8
Блок-схеми захисту від XSS-атак і SQLIA
Схема структурна
ХНУ, КБ-19-1
КРКБ.190107.19.01.08 Е8

Мій Веб-Додаток

- Register
- Головна
- XSS
- LOGIN(SQLIA)

Результат перевірки SQL-ін'єкції

Ім'я користувача

Оригінальне значення: ' OR '1'='1'; --

Результат: SQLIA Виявлено

Очищене значення: $ OR 1=1% **

Пароль

Оригінальне значення: 123456

Результат: SQLIA не виявлено

Очищене значення: 123456

Повернутись до форми

Мій Веб-Додаток

- Register
- Головна
- XSS
- LOGIN(SQLIA)

Результат очищення від XSS-атак

XSS атака виявлена

Введений текст:

<script>alert(123)</script>

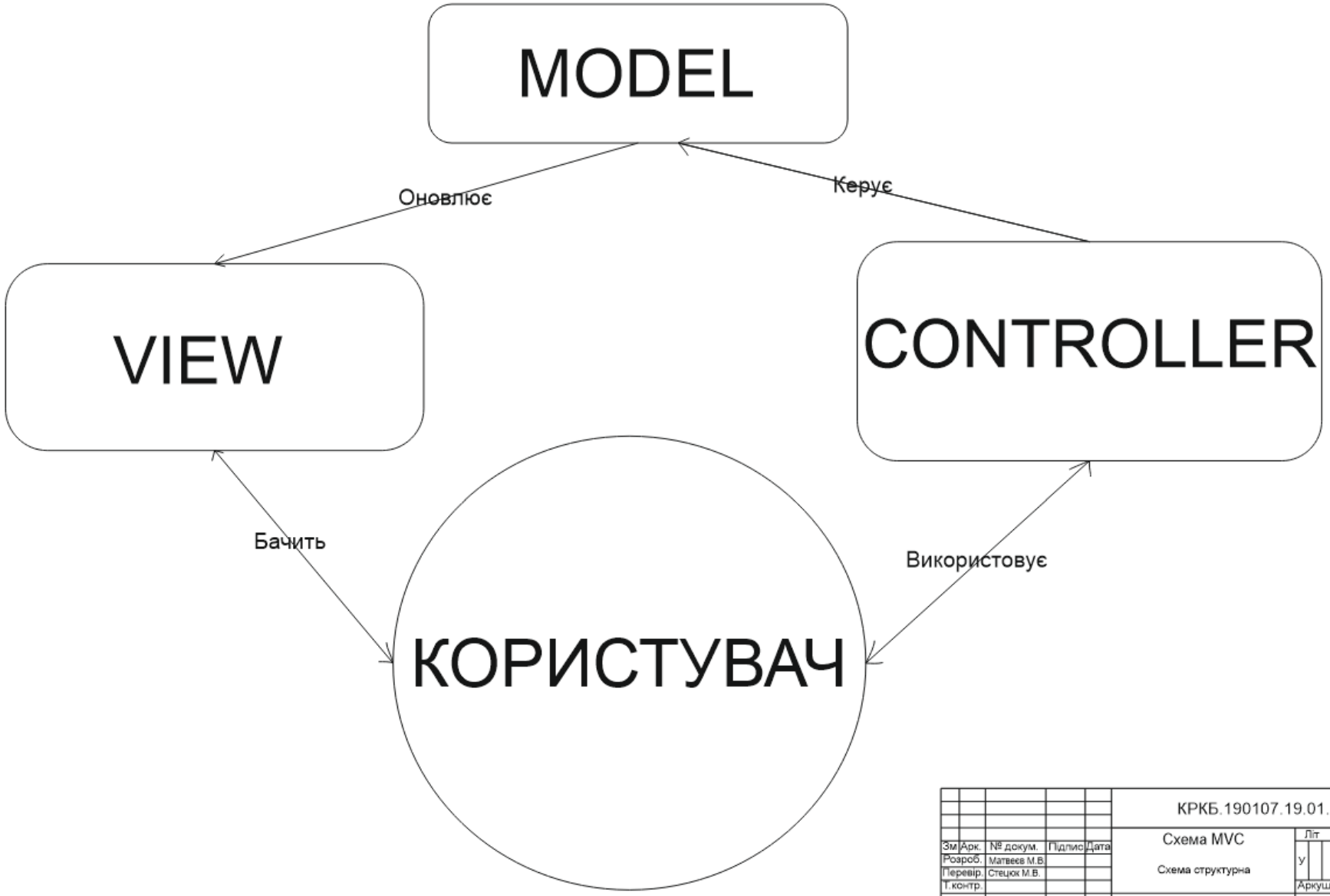
Очищений текст:

<script>alert(123)</script>

Повернутись до форми

					КРКБ.190107.19.01.08 Е8		
Зм.	Арк.	№ докум.	Підпис	Дата	Результати перевірки XSS-атак і SQLIA Схема структурна	Літ.	Маса	Масштаб
Розроб.		Матвеєв М.В.				У		
Перевір.		Стецюк М.В.				Аркуш	Аркушів 1	
Т.контр.								
Н.контр.		Мостовий С.В.				ХНУ, КБ-19-1		
Затверд.		Кльоц Ю.П.						

КРКБ.190107.19.01.08 Е8



					КРКБ.190107.19.01.08 Е8		
Зм.	Арк.	№ докум.	Підпис	Дата	Схема MVC	Літ	Маса / Масштаб
Розроб.		Матвєєв М.В.				У	
Перевір.		Стецюк М.В.			Схема структурна		
Т. контр.						Аркуш	Аркушів 1
Н.контр.		Мостовий с.в.				ХНУ, КБ-19-1	
Затверд.		Кльоц Ю.П.					

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Матвеєв Максим Вячеславович

Тема: Система захисту від Injection і XSS атак веб застосунків

Спеціальність: 125 «Кібербезпека»

Обсяг кваліфікаційної роботи:

Кількість листів креслень ___3____ Кількість сторінок записки ___55___

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є розробка системи захисту від Injection і XSS атак веб застосунків.

2. Висновок про відповідність роботи дипломному завданню: Дипломний проєкт відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: В першому розділі проведено аналіз атак, а також шкоди, яку вони можуть заподіяти. Другий розділ присвячено огляду існуючих рішень, та розробки власної системи захисту. У третьому розділі показано роботу системи та її ефективність.

4. Позитивні сторони роботи: У роботі було розроблено просту і ефективну систему, яка не тільки змінює небезпечні символи, а і зберігає початковий текст, який може бути використано для подальшого аналізу або відображення в шаблоні відповіді.

5. Негативні сторони роботи: недостатня комплексність захисту.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка та листи креслення оформлені коректно згідно діючих стандартів оформлення документації.

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА СИСТЕМНОГО ПРОГРАМУВАННЯ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Комп'ютерна мережа розподіленого офісу

Автор: Матвеєв Максим Вячеславович

Галузь знань *12 «Інформаційні технології»*

Спеціальність: 123 – *«Комп'ютерна інженерія»*

Освітня програма: *«Комп`ютерна інженерія»*

Науковий керівник: Стецюк Микола Васильович, д-р.,

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних). Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання роботи та ідентичності версії роботи.	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з 10-30 джерелами на один фрагмент речення;
4) в якості запозичень в окремих місцях системою зафіксовано послідовності кодів, які є вхідними даними до великої кількості задач і не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;
5) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 7.29 % і адресується до першоджерела, що, з урахуванням наведених обґрунтувань, відповідає характеру роботи і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ М.В. Стецюк

Завідувач кафедри кібербезпеки _____ Ю.П. Кльоц

Дата: 14.06.2023