МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ ЗАХИСТУ ІНФОРМАЦІЇ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри Комп'ютеризованих систем захисту інформації

_____ Михайло СТЕПАНОВ
«____»_____ 2023 р.

На правах рукопису
УДК 004.056.53

КВАЛІФІКАЦІЙНА РОБОТА

ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
ОСВІТНЬОГО СТУПЕНЯ «МАГІСТР»

Тема: Програмний модуль захисту інформації для вебзастосунків

Виконавець: Микола ЛОСЬ
Керівник: к.т.н., доцент Наталія ГУЛАК
Консультант розділу «Охорона навколишнього середовища»: к.т.н., доцент Тетяна ДМИТРУХА
Нормоконтролер: к.т.н., доцент Наталія ГУЛАК

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет: Кібербезпекиї та програмної інженерії

Кафедра: Комп'ютеризованих систем захисту інформації

Освітній ступінь: Магістр

Спеціальність: 125 «Кібербезпека»

Освітньо-професійна програма: «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри Комп'ютеризованих систем захисту інформації
_____ Михайло СТЕПАНОВ
«__» _____ 20__ р.

ЗАВДАННЯ

на виконання дипломної роботи

здобувача вищої освіти Лося Миколи Володимировича

1. Тема: *Програмний модуль захисту інформації для вебзастосунків* затверджена наказом ректора від «15 »вересня 2023 № 1814/ст
2. Термін виконання: з 16.10.2023 р. по 31.12.2023 р.

3. Вихідні дані: вебзастосунки, стеганографічні методи захисту інформації, криптографічні методи захисту інформації, види атак на вебзастосунки, методи захисту від атак на вебзастосунки, SQL-ін'єкції, кросс-сайтовий скриптинг, мова програмування Python

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці): аналіз методів захисту інформації в вебзастосунках на основі нормативно-правової бази України; аналіз видів атак: кросс-сайтовий скриптинг, SQL-ін'єкції, фіксація сесії для обрання методів захисту від них; розробка та тестування програмного модулю для захисту вебзастосунків від атак мовою програмування Python.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

№ з/п	Етапи виконання кваліфікаційної роботи	Термін виконання етапів	Примітка
1.	Уточнення постановки задачі	16.10.2023	*Виконано*
2.	Аналіз літературних джерел	18.10.2023	*Виконано*
3.	Обґрунтування вибору рішення	21.10.2023	*Виконано*
4.	Збір інформації	25.10.2023	*Виконано*
5.	Аналіз правових документів в сфері кібербезпеки інформації для захисту вебзастосунків	29.10.2023	*Виконано*
6.	Аналіз атак кросс-сайтового скриптингу і SQL-ін'єкцій та методи захисту від них	20.11.2023	*Виконано*
7.	Розробка програмного модулю для захисту вебзастосунків від атак на основі формальної логіки	10.12.2023	*Виконано*
8.	Екологічна безпека літосфери.	15.12.2023	*Виконано*
9.	Апробація роботи на V Міжнародній науково-практичну конференції «Tends in science regarding the creation of new teaching methods»	16.12.2023	*Виконано*
10.	Перевірка на антиплагіат	17.12.2023	*Виконано*
11.	Оформлення і друк пояснювальної записки	19.12.2023	*Виконано*
12.	Оформлення презентації	21.12.2023	*Виконано*
13.	Отримання рецензій від рецензента	22.12.2023	*Виконано*

Консультанти з окремих розділів

Розділ	Консультант (посада, П.І.Б.)	Дата, підпис	
		Завдання видав	Завдання прийняв
Охорона навколишнього середовища	Дмитруха Т.І.		

Дата видачі завдання: «16» жовтня 2023 р.

Здобувач вищої освіти Микола ЛОСЬ

(підпис, дата)

Керівник кваліфікаційної роботи Наталія ГУЛАК

(підпис, дата)

РЕФЕРАТ

Дипломна робота складається зі вступу, чотирьох розділів, загальних висновків, списку використаних джерел, додатків, загальним обсягом робота складає 75 сторінок, має 34 рисунки, 1 таблицю. Список використаних джерел містить 30 найменувань і займає 4 сторінки.

Метою дипломної роботи є розробка та тестування програмного модулю для виявлення атак на вебзастосунки з використанням формальної логіки

В дипломній роботі розглянуті питання захисту вебзастосунків за допомогою криптограії та стегонографії. Проаналізовано та запропоновано методи безпеки вебзастосунку. Розроблено програмний модуль, який здатний підвищити рівень безпеки вебзастосунку за рахунком зменшення часу для виявлення їх.

Запропонований модуль може використовуватися у реальних вебзастосунках та підвищити рівень захищеності даних. Крім того, сформульовані практичні рекомендації будуть корисними експертам з інформаційної безпеки при прийнятті рішень щодо доцільності різних підходів до захисту вебзастосунків від інших можливих видів атак.

В роботі також надано основні відомості про визначення вебзастосунків, їх використання, та їх вразливості. Описано загрози кібератак, проаналізовано їх види та наведено їх класифікацію. Описано методи захисту, на основі їх аналізу обрано формальну логіку та захист за допомогою криптографії для того, щоб створити модуль який захищає від декількох видів атак одночасно.

Ключові слова: вебзастосунок, стегонографічні методи, криптографічні методи, база даних, інформаційна безпека, кросс-сайтовий скриптинг, фіксація сесії, SQL-ін’єкція

3MICT

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

DOM – об’єктна модель документа

XSS – кросс-сайтовий скриптинг

ШІ – штучний інтелект

PKI – інфраструктура ключів

SQL – структурована мова записів

SPA – односторінковий вебзастосунок

PWA – поступовий вебзастосунок

ІКС – інформаційной-комунікаційна система

HTML – мова розмітки гіпертексту

LSB – молодший значущий біт

CSS – каскадні таблиці стилів

DES – стандарт шифрування даних

AES – розширений стандарт шифрування

IDEA – міжнародний алгоритм шифрування даних

RSA – криптографічний алгоритм з відкритим ключем

ECC – еліптична криптографія

MD5 – дайджест повідомлення 5

SHA-1 – безпечний хеш-алгоритм 1

SHA-256 – безпечний хеш-алгоритм версії 2

SSL – рівень захищених сокетів

TLS – захист на транспортному рівні

IPsec – безпека інформаційної адреси

VPN – віртуальна приватна мережа

API – інтерфейс прикладного програмування

MFA – багатофакторна аутентифікація

HTTPS – безпечний протокол передачі гіпертексту

GDPR – загальний регламент про захист даних

HIPAA – акт перенесення та звітності медичного страхування

PCI – взаємозв’язок периферійних пристроїв

DSS – система підтримки рішень

ЗІ – захист інформації

RCE – відалене виконання коду

DOS – відмова в обслуговувані

ВСТУП

Актуальність. Актуальність захисту інформації у вебзастосунках набула найвищого пріоритету в наш час, оскільки цифрова трансформація значно збільшила залежність суспільства від вебзастосунків та онлайн-сервісів. Зростаюча кількість інтернет-користувачів і обсяги онлайн-інформації створюють ідеальні умови для кіберзлочинців та зловмисників, які намагаються незаконно отримати доступ до конфіденційних даних та вразливих ресурсів. Ось деякі ключові аспекти цієї актуальності:

- Зростання важливості вебзастосунків: Вебзастосунки стали невід'ємною частиною нашого повсякденного життя. Вони використовуються для виконання широкого спектру завдань, включаючи електронну комерцію, онлайн-банкінг, медичний облік, освіту, розваги та багато інших сфер. Ця залежність від вебзастосунків робить їх важливими для бізнесу, громадян та уряду.
- Загрози для інформаційної безпеки: З ростом важливості вебзастосунків зростає і кількість загроз, які вони відчувають. Кіберзлочинці намагаються незаконно отримати доступ до конфіденційних даних, використовуючи різноманітні техніки атак, такі як SQL-ін'єкції, кросс-сайтові скрипти, фішинг, DDoS-атаки та інші. Ці загрози можуть призвести до втрати даних, фінансових втрат, порушення конфіденційності та навіть пошкодження репутації. [1]
- Порушення конфіденційності та приватності: Важливим аспектом вебзастосунків є зберігання та обробка конфіденційної інформації, включаючи особисті дані користувачів. Порушення конфіденційності може призвести до серйозних проблем із захистом приватності, і це стає особливо актуальним у світлі регуляцій, таких як Загальний регламент з захисту даних (GDPR) в Європейському Союзі. [2]
- Економічні втрати: Кібератаки на вебзастосунки можуть призвести до серйозних економічних втрат для компаній та організацій. Відновлення від

атаки може вимагати великих витрат на відновлення, а також може пошкодити репутацію компанії, що призведе до втрати клієнтів та партнерів.

Відсутність належного захисту інформації може призвести до серйозних наслідків, включаючи втрату даних, порушення конфіденційності, фінансові втрати та пошкодження репутації. Відомі випадки масштабних кібератак на вебзастосунки свідчать про необхідність розробки та впровадження ефективних програмних модулів захисту інформації для вебзастосунків.

З урахуванням цих факторів, забезпечення інформаційної безпеки вебзастосунків стає великим викликом для організацій та розробників програмного забезпечення. Важливо враховувати ці загрози та вживати належних заходів для їх запобігання та виявлення, щоб забезпечити безпеку вебзастосунків та захист інформації користувачів.

Ця тема має за мету дослідити актуальні питання захисту інформації у вебзастосунках, враховуючи сучасні тенденції в інформаційній безпеці та вимоги законодавства. Вона розгляне програмні методи та рішення, які можна використовувати для забезпечення безпеки вебзастосунків, і надасть практичні рекомендації щодо їх імплементації.

Тому, тема кваліфікаційної роботи є досить актуальною на сьогоднішній день.

Метою дипломної роботи є розробка та тестування програмного модулю для виявлення атак на вебзастосунки з використанням формальної логіки

Досягнення мети потребує розв’язання таких **задач**:

- аналіз методів захисту інформації в вебзастосунках на основі нормативно-правової бази України;
- аналіз видів атак: кросс-сайтовий скриптинг, SQL-ін’єкції, фіксація сесії для обрання методів захисту від них;
- розробка та тестування програмного модулю для захисту вебзастосунків від атак мовою програмування Python.

Об’єкт дослідження: процес захисту інформації у вебзастосунках при виявленні атак на них з застосуванням формальної логіки.

Предмет дослідження: криптографічні методи захисту від атак на вебзастосунки.

Методи дослідження: методи захисту інформації у вебзатосунках, теорія обробки статистичних даних.

Галузь застосування: даний модуль можна використовувати в реальних вебзастосунках для підвищення безпеки управління.

Наукова новизна: одержаних результатів полягає в наступному: розроблено програмний модуль для виявлення атак, таких, як фіксація сесії, SQL-ін'єкції, кросс-сайтовий скриптинг, на основі формальної логіки мовою програмування Python, що дало можливість поєднання програмних продуктів для виявлення атак у одному програмному модулі.

Практична цінність: полягає у тому, що запропонований модуль захисту може використовуватися у реальних практичних вебзастосунках для більш швидкого виявлення фіксації сесії, SQL-ін'єкції, кросс-сайтового скриптингу. Одночасне тестування дозволяє автоматизувати процес виявлення атак.

Апробація. Основні положення роботи доповідалися та обговорювалися на таких конференціях:

Лось М.В. Захист інформації у вебзастосунках методами криптографії /Н.К. Гулак, М.В. Лось //V International Scientific and Practical Conference «Trends in science regarding the creation of new teaching methods», October 16-18, 2023, Madrid, Spain. - С. 184 - 186.

Розділ 1. ЗАКОНОДАВЧА БАЗА УКРАЇНИ З ІНФОРМАЦІЙНОЇ БЕЗПЕКИ ДЛЯ ЗАХИСТУ ІНФОРМАЦІЙНИХ ДОДАТКІВ

1.1 Нормативно-правова база України

Вимоги законодавства щодо захисту інформації у вебзастосунках є надзвичайно важливим аспектом інформаційної безпеки, оскільки вони визначають правила та стандарти, які повинні дотримуватися організації та розробники для забезпечення конфіденційності, цілісності та доступності даних у вебзастосунках.

Законодавча база: Багато країн мають законодавство, яке регулює питання захисту інформації в інтернеті та вебзастосунках. В Україні, наприклад, це включає Закон України "Про захист інформації в інформаційно-телекомунікаційних системах" та Закон України "Про авторське право та суміжні права"

Закон України "Про захист інформації в інформаційно-телекомунікаційних системах" Цей закон визначає основні положення щодо захисту інформації та вимоги до операторів інформаційно-телекомунікаційних систем забезпечувати відповідний рівень захисту. [3]

Закон був прийнятий для захисту конфіденційності, цілісності та доступності інформації у державних, комерційних та громадських організаціях.

Визначення ключових термінів закону**:**

- Закон містить визначення таких термінів, як інформація, інформаційна система, інформаційно-комунікаційна система, засіб захисту інформації, об'єкт інформаційної діяльності тощо, що допомагає уточнити інтерпретацію положень закону. [3]

Засоби захисту інформації:

- Закон визначає обов'язок користувачів ІКС встановлювати засоби захисту інформації, які відповідають рівню її конфіденційності та цілісності. [3]

Вимоги до організацій**:**

- Закон встановлює вимоги щодо організаційного забезпечення захисту інформації, включаючи створення систем управління інформаційною безпекою, надання доступу лише уповноваженим особам та встановлення процедур реагування на порушення безпеки. [3]

Класифікація інформації:

- Закон передбачає можливість класифікації інформації на рівні конфіденційності та надає право на визначення ступеня захисту для різних категорій інформації. [3]

Вимоги до операторів ІКС**:**

- Закон визначає вимоги до операторів інформаційно-комунікаційних систем, включаючи обов'язок надання інформації про інциденти безпеки та забезпечення контролю за засобами захисту інформації. [3]

Відповідальність за порушення:

- Закон встановлює відповідальність за порушення вимог щодо захисту інформації та передбачає штрафи та інші санкції для осіб, які порушують цей закон. [3]

Міжнародне співробітництво:

- Закон передбачає можливість співробітництва України з іншими країнами у сфері захисту інформації та обміну досвідом. [3]

Регулярна аудиторська перевірка:

- Закон вимагає проведення регулярних аудиторських перевірок захисту інформації для визначення ступеня відповідності вимогам закону. [3]

Захист важливої інформації:

- Закон передбачає окремі вимоги та заходи для захисту важливої інформації, яка має стратегічне значення для держави. [3]

Положення про кримінальну відповідальність:

- Закон встановлює положення щодо кримінальної відповідальності за порушення правил захисту інформації. [3]

Закон України "Про авторське право та суміжні права" є одним із основних правових актів, що регулюють сферу авторського права в Україні. Він

був прийнятий з метою захисту інтелектуальної власності, сприяння розвитку культури, науки, освіти і технічного прогресу. [4]

Визначення ключових термінів:

Об'єкти авторського права: Закон визначає велику кількість об'єктів, які підлягають авторському праву, включаючи літературні твори, музику, живопис, фотографії, комп'ютерні програми, винаходи, дизайн і багато інших видів творчості. [4]

Права автора: Закон надає авторам ряд прав, включаючи право на використання, відтворення, поширення, адаптацію та інші. Ці права є ексклюзивними і дозволяють автору контролювати використання своєї творчості. [4]

Тривалість авторського права: Закон встановлює тривалість авторського права на протязі життя автора і ще 70 років після його смерті. Після закінчення цього терміну твір переходить у суспільне надбання. [4]

Захист інтересів правовласників: Закон містить положення щодо захисту правовласників від порушень авторських прав, включаючи правила для розгляду судових справ і визначення відшкодування за завдані збитки. [4]

Суміжні права: Закон також регулює суміжні права, які стосуються правовласників, таких як виконавці, фонограми та ведення кабельного мовлення. [4]

Комп'ютерні програми: Закон визначає особливі правила для захисту комп'ютерних програм і встановлює, що вони є об'єктами авторського права. [4]

Відкриті ліцензії: Закон дозволяє авторам використовувати відкриті ліцензії, такі як Creative Commons, для регулювання використання своїх творів. [4]

Захист прав споживачів: Закон містить положення, що стосуються прав споживачів і захисту їхніх інтересів у випадку придбання продуктів із застосуванням авторських прав. [4]

Забезпечення відповідності цим вимогам законодавства є важливим завданням для будь-якої організації чи розробника вебзастосунків. Це

допомагає не лише уникнути юридичних проблем, але й забезпечити захист інформації користувачів та зберегти довіру споживачів.

1.2 Визначення та принцип роботи вебзастосунку

Вебзастосунок - це програмне забезпечення, призначене для виконання конкретних функцій або завдань через веб-браузер. Вебзастосунки можуть бути статичними або динамічними та можуть надавати різноманітні сервіси, від інтерактивних веб-сайтів до складних систем управління даними. [5]

Принцип роботи веб-застосунку:

Принцип роботи веб-застосунку полягає в обробці запитів від користувача через веб-браузер і наданні відповідей у вигляді HTML-сторінок. Вебзастосунок може взаємодіяти з базою даних, обробляти логіку за допомогою серверних скриптів і надавати відповіді на клієнтські запити.

Статичні сторінки:

Статичні сторінки - це HTML-документи, які не змінюються під час завантаження користувачем. Вони можуть містити текст, зображення і гіперпосилання, але не мають інтерактивної функціональності. Статичні сторінки швидко завантажуються і мають мінімальний обсяг обробки на сервері.

Динамічні сторінки:

Динамічні сторінки генеруються на сервері на основі даних або параметрів, наданих користувачем. Вони можуть включати інтерактивний контент, форми для введення даних, обробку даних і відображення різних сторінок в залежності від умов. Динамічні сторінки часто використовуються в онлайн-магазинах, соціальних мережах, банківських системах і т. д. [6]

Переваги веб-застосунків:

- Доступність: Веб-застосунки доступні з будь-якого пристрою з веб-браузером та підключенням до Інтернету.

- Спрощена розгортка та оновлення: Веб-застосунки можна оновлювати централізовано на сервері, що робить процес розгортання та оновлення зручним та ефективним.
- Масштабованість: Веб-застосунки можна легко масштабувати для обробки великої кількості користувачів і даних.
- Кросплатформенність: Веб-застосунки працюють на різних операційних системах і пристроях без необхідності розробки окремих версій для кожної платформи.
- Оновлення в реальному часі: Динамічні веб-застосунки можуть оновлювати вміст без необхідності перезавантаження сторінки, надаючи користувачам актуальну інформацію в реальному часі. [5]

Ці аспекти є ключовими для розуміння функціональності та цінності веб-застосунків, а також для розробки та захисту їхньої безпеки.

1.3 Стеганографічні методи захисту інформації

Стеганографія - це наука про те, як приховувати інформацію в інших даних таким чином, щоб незрозуміло було, що взагалі є прихована інформація. Вона відрізняється від криптографії, де метою є шифрування інформації для забезпечення конфіденційності, а стеганографія спрямована на те, щоб зробити наявність інформації невидимою. [6]

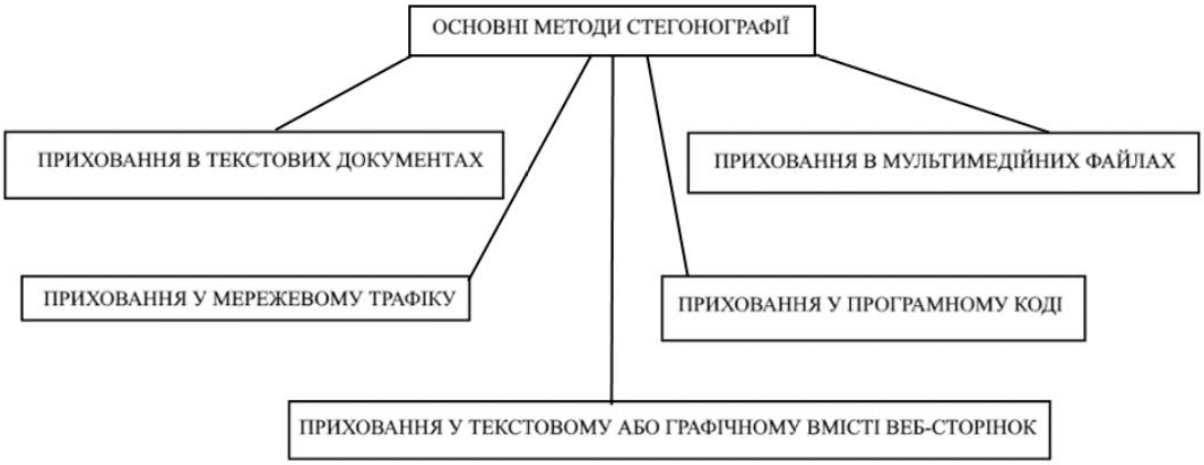

Рис. 1.1. Основні методи стеганографії

1.3.1 Приховання у текстових документах

У цьому методі захисту прихована інформація вбудовується у текстовий документ, наприклад, в пропусках між словами або символами. Можуть використовуватися різні методи кодування, які дозволяють приховати інформацію у тексті, такі як метод LEAST SIGNIFICANT BIT (LSB), де найменший значущий біт пікселів зображення може бути використаний для зберігання інформації. [7]

1.3.2 Приховання у мультимедійних файлів

Схожий до попереднього метод, але інформація вбудовується у зображення, відео або аудіофайли. Це може бути використано для приховування тексту, зображень [8] або навіть інших файлів у мультимедійних контентах. [6]

1.3.3 Приховання у мережевому трафіку

У цьому методі прихована інформація вбудовується у мережевий трафік, наприклад, шляхом встановлення призначених для цього позначок в заголовках пакетів даних. Це може бути використано для приховування інформації в комунікації між комп'ютерами.

1.3.4 Приховання у програмному коді

Інформація може бути прихована у програмному коді, де використовуються спеціальні коментарі або символи, які виглядають як нормальний програмний код, але містять приховану інформацію.

1.3.5 Приховання у текстовому або графічному вмісті веб-сторінок

Інформація може бути вбудована у текстові або графічні елементи веб-сторінок, такі як HTML-код, зображення або CSS-стилі.

Важливо зауважити, що стеганографія може бути використана як для законних, так і для незаконних цілей. Вона може бути корисною для захисту конфіденційної інформації, але також може бути використана для приховання шкідливих дій, таких як виток конфіденційних даних або атаки на мережі. Тому використання стеганографії повинно бути обговорене і санкціоноване відповідними політиками та правилами безпеки.

1.4 Методи використання стегонографії

Основні методи використання стегонографії представлені на рисунку 1.2

Рис. 1.2. Використання методів стегонографії

Захист конфіденційності інформації

Стеганографія може бути використана для захисту конфіденційних даних, коли важлива інформація може бути прихована у звичайних даних і передана без привернення уваги.

Контроль доступу

Вбудована інформація може бути використана для визначення доступу до ресурсів. Наприклад, веб-сайти можуть використовувати стеганографію для приховування ключів аутентифікації або прав доступу.

Антиплагіат

Стеганографія може бути використана для захисту авторських прав, додавання прихованих маркерів або підписів до медіафайлів, які дозволяють визначити їхніх авторів або джерело. [9]

Контроль цілісності даних

Вбудована інформація може бути використана для визначення цілісності даних під час передачі або зберігання. Це дозволяє виявити будь-які зміни в даних.

Захист від перехоплення

Стеганографія може бути використана для унікального шифрування даних, що робить їх менш вразливими до перехоплення атаками.

Спостереження

Стеганографія може бути використана для надання прихованої інформації агентам розвідки або правоохоронним органам для спостереження та збору розвідувальних даних.

Важливо враховувати, що стеганографія не замінює криптографію, і її ефективність залежить від специфічного застосування та заходів безпеки, вжитих для захисту прихованої інформації.[9] Також важливо, щоб використання стеганографії було в рамках закону і відповідало правилам конфіденційності та безпеки відповідних організацій.

1.5 Криптографічні методи захисту інформації

Криптографія - це наука про забезпечення конфіденційності, цілісності та автентичності інформації шляхом застосування математичних та обчислювальних методів. [10] Криптографічні методи захисту інформації

включають в себе різні техніки та алгоритми для шифрування та розшифрування даних.

Рис. 1.3. Основні криптографічні методи

1.5.1 Симетричне шифрування

У цьому методі швифрування один і той же ключ використовується як для шифрування, так і для розшифрування даних. Популярні симетричні алгоритми включають DES, AES, і IDEA. Симетричне шифрування використовується для шифрування великих обсягів даних, оскільки воно є швидким і ефективним. [11]

1.5.2 Асиметричне шифрування

У цьому методі використовуються два різні ключі - публічний і приватний. Публічний ключ використовується для шифрування даних, а приватний ключ - для їх розшифрування. Популярні асиметричні алгоритми включають RSA та ECC. Асиметричне шифрування використовується для забезпечення конфіденційності та автентичності даних. [12]

1.5.3 Хеш-функції

Хеш-функції призначені для перетворення великого обсягу даних у фіксований вихідний код (геш-код), який є унікальним для кожного набору вхідних даних. Популярні хеш-функції включають MD5, SHA-1 і SHA-256. Хеш-функції використовуються для перевірки цілісності даних і створення цифрових підписів. [13]

1.5.4 Цифрові підписи

Цифровий підпис - це спеціальна форма криптографічного підпису, який дозволяє підтвердити автентичність та цілісність документа або повідомлення. Він створюється за допомогою приватного ключа, і перевіряється за допомогою відповідного публічного ключа.

1.5.5 Протоколи безпеки

Протоколи безпеки, такі як SSL/TLS для захищеної передачі даних в мережі Інтернет, використовують криптографічні методи для забезпечення безпеки комунікації між клієнтом і сервером. [14]

1.5.6 Ключі для доступу

Ключі для доступу використовуються для автентифікації користувачів і контролю доступу до ресурсів. Вони можуть бути використані в парах - публічний і приватний ключі - для забезпечення безпеки вхідних даних.

1.5.7 Управління ключами

Управління ключами включає в себе створення, зберігання і розподіл ключів шифрування. Це важливий аспект криптографічної безпеки, оскільки від нього залежить захищеність ключів і, отже, конфіденційність інформації.

1.6 Методи використання криптографії

Криптографічні методи захисту інформації використовуються в різних галузях, включаючи інформаційну безпеку, електронну комерцію, фінанси, медицину та інші сфери. Вони допомагають забезпечити конфіденційність та цілісність даних, а також забезпечують захист від несанкціонованого доступу та зламу систем безпеки. [10]

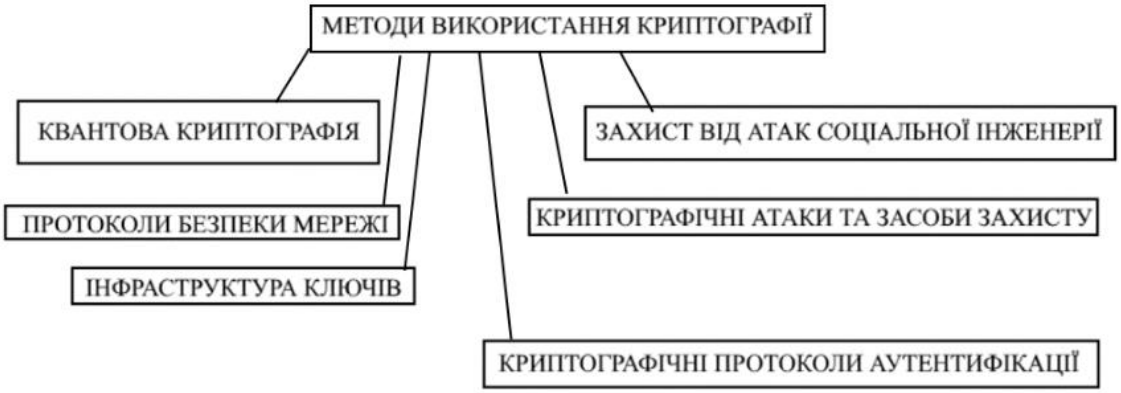

Рис.1.4. Методи використання криптографії

Квантова криптографія

Цей різновид криптографії використовує властивості квантової механіки для забезпечення безпеки передачі даних. Квантова криптографія базується на принципах, що використовують фотони та їхні квантові властивості для створення безпечних криптографічних ключів і захисту інформації від атак з використанням квантових комп'ютерів.

Криптографічні протоколи аутентифікації

Ці протоколи використовуються для підтвердження ідентичності користувачів або систем. Протоколи такого типу забезпечують взаємну аутентифікацію сторін і можуть використовувати симетричне або асиметричне шифрування. [15]

Протоколи безпеки мережі

Протоколи, такі як IPsec і VPN, використовуються для захисту комунікацій в мережах. Вони забезпечують шифрування даних, що передаються через мережу, і забезпечують конфіденційність і цілісність даних. [16]

Криптографічні атаки та засоби захисту

Розробники криптографічних систем також розглядають різні види атак, включаючи атаки методом перебору, атаки з використанням підставних ключів, атаки на імплементацію, і багато інших. Захист від цих атак включає в себе ретельний вибір криптографічних алгоритмів, безпечне зберігання ключів і використання найкращих практик безпеки.

Засоби для аутентифікації та управління доступом

До цих засобів входять системи багаторівневої аутентифікації, біометричні методи аутентифікації, механізми контролю доступу і системи управління ідентифікацією користувачів. [17]

Інфраструктура ключів

Інфраструктура ключів (PKI) використовується для створення, зберігання та розподілу криптографічних ключів і цифрових сертифікатів. Вона грає важливу роль у забезпеченні безпеки комунікацій та ідентифікації користувачів.

Захист від атак з використанням соціальної інженерії

Криптографія також використовується для захисту від атак, які використовують соціальну інженерію, такі як фішинг і обман користувачів. Шифрування даних та відповідна освіта користувачів є важливими аспектами захисту від цих атак.

Аналіз і аудит криптографічної безпеки

Важливим аспектом криптографічної безпеки є аналіз і аудит існуючих рішень для виявлення можливих вразливостей та удосконалення заходів безпеки [17]

Криптографічні методи захисту інформації використовуються для забезпечення конфіденційності, цілісності та автентичності даних у різних сферах, і вони є невід'ємною частиною сучасних систем безпеки. Криптографія постійно розвивається, і нові методи та алгоритми виникають для відповіді на зростаючі загрози і вимоги безпеки (табл 1).

1.7 Порівняльний аналіз

Стеганографія надає можливість приховувати інформацію у вигляді інших даних, зробивши її незамітною для потенційних зловмисників. З іншого боку, криптографія надає більшу стійкість до криптоаналітичних атак та гарантує конфіденційність даних.

Порівняльний аналіз методів стегонографії і криптографії надано у таблиці 1.1.

Таблиця 1.1

Порівняльний аналі стегонографії та криптографії

1	2	3
Критерій	**Стеганографія**	**Криптографія**
Захист даних	Приховання даних в іншому типічно незв'язаному з вмістом, наприклад, у зображеннях, звуку тощо.	Зашифрування даних так, що їх можна розшифрувати лише з допомогою ключа.
Використання в вебзастосунках	- Для прихованого обміну даними в текстових повідомленнях, фотографіях, аудіо та відео. [6] - Зазвичай використовується для створення "непомітних засобів водяного знака для авторизації вмісту."	- Для конфіденційного обміну даними в мережі та збереження даних у базах даних.
Переваги	- Невидимість застосування в данних методі захисту. - Позаяк на перший погляд нічого не приховано, важко виявити наявність прихованих даних.	- Висока стійкість до криптоаналітичних атак. - Ефективний захист інформації від несанкціонованого доступу. - Можливість встановлення різних рівнів шифрування. - Відносно велика швидкість зашифрування та розшифрування.
Недоліки	- Низька стійкість до криптоаналітичних атак. - Можливість виявлення прихованих даних при використанні спеціального програмного забезпечення.	- Вимагає обміну ключами між комунікуючими сторонами. - Вимагає додаткових обчислень для шифрування та розшифрування пов'язаних з ключем.

1.8 Висновки до першого розділу

У розділі було проаналізовано наступні закони, було проаналізовано криптографічні та стегонографічні методи. Було зроблено порівняльний аналіз

цих методів.

Вибір між стеганографією та криптографією для захисту веб-застосунків залежить від конкретних потреб та вимог проекту.

На основі проведеного аналізу для розробки модулю захисту вебзастосунків було обране криптографічне рішення, тому що в порівнянні з стегонографією криптографія дозволяє вебзастосунку мати вищий рівень захисту від прямих атак, що дозволить максимізувати рівень захищеності сервісу.

Розділ 2. АНАЛІЗ МЕТОДІВ ЗАХИСТУ ВЕБЗАСТОСУНКІВ

2.1 Методи атаки на вебзастосунки

Існує багато видів атак на веб-додатки, які можуть використовуватися зловмисниками для отримання несанкціонованого доступу, викрадення даних або впливу на нормальну роботу вебзастосунків

Основні види атак на вебзастосунки надано на рисунку 2.1

Рис. 2.1. Види атак на вебзастосунки

2.1.1 Кросс-сайтовий скриптинг (XSS)

XSS - це атака, при якій зловмисник вставляє злоякісний код (зазвичай JavaScript) у веб-сторінку, яка потім виконується в браузері іншого користувача. основними видами XSS є:

- Stored XSS (Persisted XSS):

Атака відбувається, коли зловмисник вводить в шкідливий код на сервері, і цей код потім зберігається на сервері. Коли інший користувач відвідує сторінку, на якій розміщений зловмисний код, він виконується в браузері цього користувача.

Наприклад зловмисник може вводити шкідливий скрипт у поле коментарів на форумі або в область введення для користувачів, і кожен, хто переглядає цю сторінку, виконує шкідливий код.

- Reflected XSS:

При атакі Reflected XSS, зловмисник вмурує шкідливий код у URL-адресу або параметри запиту, і цей код відображається на сторінці і виконується, якщо користувач переходить за цим посиланням.

Наприклад зловмисник може надіслати користувачеві посилання з шкідливим кодом, який виконується при відкритті посилання.

- DOM-based XSS:

У цій атаці XSS відбувається на рівні Document Object Model (DOM) браузера. Зловмисник впливає на DOM-структуру сторінки, яка вже завантажена в браузері користувача.

Наприклад шкідливий код може змінювати DOM-елементи на сторінці, призводячи до виконання шкідливого коду в браузері користувача.

- Blind XSS (Second Order XSS):

Для виконання цієї атаки зловмисник вводить шкідливий код, але його виконання відбувається на іншій сторінці або для іншого користувача, ніж той, який ввів дані.

Прикладом виконання такої атаки може бути наступна ситуація: Зловмисник вводить шкідливий скрипт, який зберігається на сервері. Коли адміністратор або інший користувач з вищими привілеями переглядає сторінку, на яку зберігається шкідливий код, він виконується в браузері адміністратора.

- Self-XSS:

В цій атаці зловмисник намагається обдурити самого користувача та переконати його в виконанні шкідливого коду у власному браузері.

Наприклад зловмисник може надіслати фішинговий електронний лист або здійснити атаку через соціальні мережі, запропонувавши користувачеві ввести шкідливий код у власній адресній строкі браузера.

2.1.2 Ін'єкції (SQL і інші)

Ін'єкція - це атака, при якій зловмисник вставляє злоякісний код або команди в дані, які передаються веб-додатку через введення або параметри URL. Основні види ін'єкцій включають:

- SQL-ін'єкція**:** Зловмисний SQL-код вставляється в запит до бази даних, що може призвести до витоку, видалення або модифікації даних у базі даних.
- Ін'єкція команд**:** Злоякісні команди вставляються в системні команди сервера, що дозволяє зловмиснику виконувати дії на сервері, такі як створення, зміна або видалення файлів.
- Ін'єкція JavaScript (DOM-ін'єкція): Злоякісний код вставляється в DOM-структуру сторінки і виконується в браузері користувача, подібно до XSS

Атаки ін'єкцій можуть викликати різні наслідки, включаючи витік конфіденційних даних, втрату контролю над системою, відмову в обслуговуванні (DoS), і багато іншого. [7]

2.1.3 Переповнення буфера

Переповнення буфера- це ситуація, коли зловмисник вводить більше даних у буфер (наприклад, масив або буфер пам'яті) програми, ніж цей буфер може обмістити. Це може викликати переписування важливих даних, виконання злоякісного коду або навіть виведення програми з ладу.

Види атак на переповнення буфера:

- Переповнення стеку (Stack Overflow): Викликається, коли зловмисник переповнює стек програми, змінюючи адреси повернення і може виводити виконання коду на свій користь.
- Переповнення кучі (Heap Overflow): Відбувається, коли зловмисник змінює дані, розміщені у купі (heap) і може викликати неконтрольоване виконання коду.

2.2 Потенційні збитки від атак на вебзастосунки

2.2.1 Можливі збитки від атаки кросс-сайтвого скриптингу

Кросс-сайтовий скриптинг (XSS) є серйозною загрозою для веб-додатків і може призвести до різноманітних потенційних збитків, включаючи:

1. Крадіжка інформації користувача: Атаки XSS можуть використовуватися для крадіжки кредитних карток, паролів, сесійних токенів та інших особистих даних користувачів. Атаки можуть спрямовуватися на викрадення конфіденційних даних, які вводяться користувачем на заражених сторінках.

2. Виконання небезпечного коду: Зловмисники можуть впроваджувати та виконувати свій власний JavaScript-код на сторінці користувача. Це може включати в себе перехоплення сесій, відправку небажаних запитів, зміну вмісту сторінки та інші дії, що можуть вплинути на користувача.

3. Руйнування репутації: Зловмисники можуть використовувати XSS для внесення змін у відображення веб-сайту або додавання образливого чи шкідливого вмісту. Це може пошкодити репутацію власника веб-сайту чи організації.

4. Розповсюдження шкідливих програм: Атаки XSS можуть використовуватися для поширення шкідливих програм серед відвідувачів веб-сайту. Це може включати в себе впровадження вредоносного коду або посилань на шкідливі ресурси.

5. Нанесення шкоди корпоративним системам: Якщо атака XSS використовується в корпоративному середовищі, це може призвести до порушення безпеки корпоративних систем, витоку конфіденційної інформації, а також до втрати доступу до важливих функцій.

6. Втрата відвідувачів та клієнтів: Якщо веб-сайт стає жертвою атаки XSS, це може вплинути на довіру користувачів і призвести до втрати відвідувачів та клієнтів.

7. Перехоплення сесій: Злоякісний код може змінювати ідентифікатори сесій або переносити їх до атакуючого. Це дозволяє атакуючому представити себе як іншу особу.

8. Зміна сторінки веб-сайту**:** Злоякісний код може модифікувати вміст веб-сторінки, що призводить до відображення фальшивої інформації, спаму, або навіть фішингових атак [18]

2.2.2 Можливі збитки від атаки SQL-ін’єкції

Атака SQL ін'єкції є серйозною загрозою для веб-додатків, які використовують бази даних. Ця атака полягає у введенні зловмисником шкідливого SQL-коду в введені дані, що може призвести до різноманітних потенційних збитків, таких як:

1. Доступ до конфіденційної інформації: Зловмисники можуть використовувати атаку SQL ін'єкції для отримання несанкціонованого доступу до конфіденційної інформації в базі даних. Це може включати в себе дані про користувачів, платіжні дані, паролі та інші конфіденційні дані.

2. Модифікація або вилучення даних: Зловмисники можуть використовувати атаку SQL ін'єкції для внесення змін у вміст бази даних. Це може призвести до видалення, модифікації або втрати важливих даних, що може суттєво вплинути на функціонування веб-додатка та організації в цілому.

3. Використання атаки для поширення шкідливого коду: Зловмисники можуть використовувати атаку SQL ін'єкції для впровадження та виконання шкідливого коду в середовищі бази даних. Це може бути використано для розповсюдження шкідливих програм серед користувачів та серверів.

4. Отримання несанкціонованого доступу до системи: Вразливість, викликана атакою SQL ін'єкції, може використовуватися для отримання несанкціонованого доступу до сервера, на якому працює веб-додаток. Це може включати в себе доступ до файлової системи та інших ресурсів сервера.

5. Втрата доступності: Атаки SQL ін'єкції можуть призвести до відмови в обслуговуванні або інших проблем, які можуть спричинити втрату доступності веб-додатка для користувачів.

2.2.3 Можливі збитки від атаки переповнення буферу

Атака переповнення буфера є серйозною загрозою для веб-додатків, особливо для тих, які написані мовами програмування, такими як C або C++. Ця атака може призвести до різноманітних потенційних збитків, включаючи:

1. Виконання віддаленого коду (Remote Code Execution, RCE**):** Зловмисники можуть використовувати атаку переповнення буфера для впровадження свого власного виконуваного коду в пам'ять веб-додатка. Це може призвести до віддаленого виконання шкідливого коду на сервері, забезпечуючи зловмиснику контроль над системою.

2. Крадіжка конфіденційних даних: Атака переповнення буфера може бути використана для отримання доступу до конфіденційних даних, які зберігаються в пам'яті веб-додатка. Це може включати в себе паролі, ключі доступу, сесійні токени та іншу конфіденційну інформацію.

3. Відмова в обслуговуванні (Denial of Service, DoS): Атака переповнення буфера може призвести до переповнення пам'яті веб-додатка, що може призвести до відмови в обслуговуванні. Зловмисники можуть спробувати спеціально створені запити для перевищення ресурсів, що може вплинути на доступність веб-сайту.

4. Зміна потоку виконання програми: У разі успішного виконання атаки переповнення буфера зловмисник може змінити нормальний потік виконання програми. Це може призвести до непередбачених наслідків, таких як виконання шкідливого коду, некоректної обробки даних та інших проблем.

5. Компрометація безпеки в системі: Успішна атака може використовувати уразливість в веб-додатку для отримання несанкціонованого доступу до інших систем, на які веб-додаток має права доступу. Це може включати в себе бази даних, файлові системи та інші ресурси

2.3 Методи захисту від атак на вебзастосунки

Безпека веб-додатків - це постійний процес і вимагає уваги до деталей і врахування різних загроз.

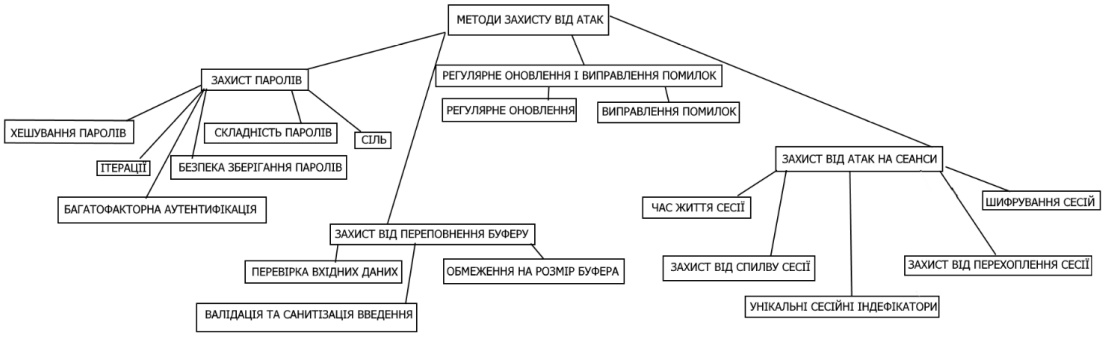

Рис. 2.2 Методи захисту від атак

Аналіз програмних методів захисту інформації в вебзастосунках від несанкціонованого доступу є важливою частиною процесу забезпечення кібербезпеки в онлайн-середовищі. Ось кілька ключових аспектів, які варто враховувати при аналізі програмних методів

Захист від XSS та ін'єкцій включає в себе перевірку та екранування даних перед виведенням на сторінку, використання безпечних API, які запобігають ін'єкціям (наприклад, параметризовані запити SQL), та регулярні оновлення безпеки системи.

2.3.1 Захист паролів

Захист паролів є критично важливим аспектом безпеки вебзастосунків, оскільки паролі використовуються для аутентифікації користувачів і надання їм доступу до системи. Дотримання найкращих практик у цій області допомагає уникнути небезпеки, пов'язаної з незаконним доступом і витоком паролів. Ось детальний огляд заходів для захисту паролів:

Складність паролів**:**

- Довжина**:** Довгі паролі більш надійні. Рекомендується встановлювати паролі довжиною не менше 12 символів.
- Різноманітність символів**:** Паролі повинні містити комбінацію великих і малих літер, цифр та спеціальних символів (наприклад, !, @, #, $).
- Уникнення простих шаблонів: Користувачам не слід використовувати очевидні паролі, такі як "password", "123456", або "qwerty". Система повинна вимагати складних і непередбачуваних паролів

Хешування паролів:

- Паролі повинні бути збережені у формі хешів, а не у відкритому тексті. Хеш - це велика криптографічна величина, створена на основі пароля, і він набагато складніший для розкриття

Сіль (Salt):

- Додавання солі до паролів підвищує безпеку. Сіль - це випадкова послідовність символів, яка додається до паролю перед хешуванням. Це запобігає атакам з використанням таблиць радужних таблиць.

Ітерації (Iterations):

При створенні хеша пароля важливо проводити багаторазове хешування (ітерації). Це зробить атаку на хеш більш витратною за часом і ресурсами.

Безпека зберігання паролів**:**

- Паролі повинні зберігатися в безпечному середовищі з обмеженим доступом. Краще всього - використовувати безпечну базу даних з правильними правами доступу. [19]

2.3.2 Захист від атаки "брутфорс" (Brute Force):

- Введення обмежень на кількість спроб невдалих входів (наприклад, блокування акаунта після кількох невдалих спроб). [20]

Оновлення паролів**:**

- Заохочення користувачів регулярно змінювати свої паролі і не використовувати один і той же пароль для багатьох ресурсів.

Багатофакторна аутентифікація (MFA)**:**

- Заохочення використання багатофакторної аутентифікації, яка включає в себе ще один рівень безпеки, крім пароля, наприклад, одноразовий код, біометричну ідентифікацію і т.д.

Захист паролів - це невід'ємна частина кібербезпеки вебзастосунків і даних користувачів. Важливо стежити за найновішими рекомендаціями і технологіями, щоб забезпечити надійний захист паролів і уникнути можливих порушень безпеки.

2.3.3 Захист від атак на сеанси

Захист від атак на сеанси- це важливий аспект безпеки вебзастосунків, оскільки атаки на сеанси можуть призвести до незаконного доступу до аккаунтів користувачів і витоку конфіденційної інформації. Ось детальний огляд заходів для захисту сеансів:

Унікальні сесійні ідентифікатори:

- Кожному користувачеві повинен бути призначений унікальний ідентифікатор сесії, який використовується для ідентифікації його сеансу. Цей ідентифікатор повинен бути достатньо складним і випадковим, щоб ускладнити спроби вгадати його.

Шифрування сесій**:**

- Усі дані, які передаються між браузером користувача і сервером, повинні бути шифрованими за допомогою протоколу HTTPS. Це захищає дані сесії від перехоплення або прослуховування.

Час життя сесій**:**

- Сесійні ідентифікатори повинні мати обмежений час життя. Після закінчення часу сесія повинна автоматично завершуватися і користувач повинен знову аутентифікуватися.

Захист від атаки "перехоплення сесії" (Session Hijacking) [21]**:**

- Захист від цього типу атак включає в себе заходи для ускладнення перехоплення і використання сесійних ідентифікаторів. Це може бути досягнуто шляхом використання HTTPS, відправки сесійних ідентифікаторів у закодованому вигляді і використання HTTP-only cookies.

Захист від атаки "підмена сесії" (Session Fixation)**:**

- Ця атака полягає в тому, що зловмисник встановлює свій сесійний ідентифікатор на комп'ютері жертви. Захист передбачає генерацію нового сесійного ідентифікатора після аутентифікації користувача.

Захист від атаки "фіксація сесії" (Session Fixation)**:**

- Ця атака відбувається, коли зловмисник змушує користувача використовувати сесійний ідентифікатор, який він контролює. Захист полягає в генерації нового сесійного ідентифікатора після аутентифікації.

Спливи сесій:

- Сесійні ідентифікатори повинні бути унікальними для кожного користувача і змінюватися при зміні авторизованого користувача або при зміні привілеїв.

Аутентифікація для кожного запиту**:**

- Кожен запит, який потребує авторизації, повинен бути перевірений для валідності сесійного ідентифікатора та прав доступу користувача. [21]

Захист сеансів - це не менш важливий аспект безпеки, і він вимагає комплексного підходу, який включає в себе технічні заходи та навчання користувачів щодо безпечного використання сесій.

2.3.4 Захист від переповнення буфера

Захист від переповнення буфера є важливою частиною безпеки програмного забезпечення, оскільки атаки на переповнення буфера можуть призвести до виконання злоякісного коду або виведення програми з ладу. Давайте розглянемо деталі цього виду захисту:

- Використання безпечних функції: У багатьох мовах програмування існують безпечні версії функцій, які автоматично встановлюють обмеження на розмір буфера або виконують перевірку на входження в діапазон.

- Перевірка вхідних даних: Потрібно переевіряти вхідні дані ззовні, щоб переконатися, що вони не перевищують призначений розмір буфера. Всі дані, які приходять з незнайомого джерела, повинні бути обмежені та валідовані.

- Використання обмеження на розмір буфера: Встановлення обмеження на максимальний розмір буфера та перевірка, чи не перевищено ці обмеження під час операцій із збереженням даних. [20]

- Використання безпечних мов програмування: Деякі мови програмування, такі як Java та C#, мають вбудовані заходи безпеки для запобігання переповненню буфера.
- Регулярне оновлення програмного забезпечення: Оновлення програмних бібліотек та компіляторів може включати в себе заходи безпеки для захисту від відомих уразливостей.

Валідація та санітизація введення:

Перевірка та санітизація вхідних даних може зменшити ризик переповнення буфера. Валідація передбачає перевірку даних на відповідність очікуваному формату, а санітизація - видалення потенційно небезпечних частин даних.

Загальною метою заходів захисту від переповнення буфера є забезпечення валідності та обмеження розмірів введених даних, щоб уникнути вразливостей, пов'язаних із переповненням буфера, і захистити програми від атак. [20]

2.3.5 Захист від атак, пов'язаних з індексацією

Захист від атак, пов'язаних з індексацією (зазвичай це атаки на основі масивів або списків), є важливою частиною безпеки програмного забезпечення. Ці атаки можуть викликати неконтрольований доступ до даних, виток конфіденційної інформації та виведення програм з ладу. Давайте розглянемо деталі захисту від цих атак:

Перевірка меж індексу: Одним із основних заходів для захисту від атак на індексацію є перевірка коректності індексів, які використовуються для доступу до масивів або списків. Перевіряйте, чи індекс не виходить за межі дозволеного діапазону і чи не від'ємний.

Валідація вхідних даних: Перевірка вхідних дані перед використанням їх як індекси для масивів або списків. Вхідні дані, що використовуються як індекси, повинні бути перевірені на валідність та обмежені до дозволеного діапазону.

Використання безпечних інтерфейсів: В мовах програмування існують безпечні функції та інтерфейси, які дозволяють безпечно взаємодіяти з масивами і списками. Наприклад, у Python є методи для списків, які автоматично виконують перевірку індексів.

Використання статичного аналізу коду та інструментів для виявлення уразливостей: Ці інструменти можуть бути використанні для виявлення уразливостей, які можуть виникнути внаслідок некоректної індексації. Це допоможе знайти потенційні проблеми ще до того, як програма буде запущена.

Використання моніторингу і журналювання: Моніторинг додатків для виявлення надзвичайних подій та журналювання важливих операцій допоможе вчасно виявити атаки та події, пов'язані з індексацією.

Загалом, захист від атак на індексацію - це важлива частина безпеки програмного забезпечення, яка допомагає уникнути витоку конфіденційної інформації та зберегти цілісність програм.

2.3.6 Регулярне оновлення і виправлення помилок

Регулярне оновлення і виправлення помилок (патчінг) є важливою частиною стратегії безпеки в інформаційних системах і програмному забезпеченні. [22] Це процес, за допомогою якого виробники програм видають оновлення і коригують виявлені уразливості та помилки в програмах і операційних системах. Нижче подано детальну інформацію про цей процес:

Регулярне оновлення:

- Постійні оновлення програм і операційних систем: Виробники програм і операційних систем постійно вдосконалюють свої продукти, включаючи в них нові функції та виправлення безпеки. Ці оновлення регулярно видаються у вигляді патчів або оновлень, які можна завантажити і встановити.
- Виправлення виявлених уразливостей: Один із головних приводів для випуску оновлень - це виявлені уразливості, які можуть бути використані зловмисниками для атак на систему. Виробники програм стараються реагувати на ці уразливості якомога швидше і надають патчі для їх усунення.

- Захист від відомих атак: В регулярних оновленнях також можуть бути включені заходи для захисту від відомих видів атак, таких як атаки на вразливості програмного забезпечення (exploits). Виробники можуть вдосконалювати механізми захисту і додавати нові функції для запобігання атакам.

Виправлення помилок:

- Виправлення функціональних і технічних помилок: Помилки в програмах можуть впливати на їхню коректну роботу і навіть призводити до аварійного завершення роботи. Регулярне виправлення таких помилок допомагає забезпечити стабільну та надійну роботу програм.
- Підвищення якості продукту: Виробники програм і операційних систем враховують повідомлення від користувачів і виправляють виявлені помилки, що допомагає підвищити якість продукту та забезпечити кращий досвід користувачів.
- Зменшення ризику втрати даних: Деякі помилки можуть призводити до втрати даних або навіть до їхнього пошкодження. Виправлення цих помилок допомагає запобігти втраті важливої інформації.

Важливо наголосити, що регулярне оновлення і виправлення помилок є критично важливими для забезпечення безпеки і надійності інформаційних систем. Користувачам і адміністраторам систем важливо слідкувати за випуском оновлень і вчасно встановлювати їх, щоб запобігти можлим загрозам і проблемам.

Аналіз програмних методів захисту інформації в вебзастосунках від несанкціонованого доступу є постійним ітеративним процесом, оскільки загрози змінюються і еволюціонують з часом. Забезпечення високого рівня безпеки вимагає постійного вдосконалення та оновлення заходів захисту.

2.4 Необхідність розробки програмного модулю для захисту вебзастосунків

Розробка програмних модулів для захисту веб-додатків є критично важливою в сучасному цифровому світі з огляду на різноманітні загрози, з якими стикаються організації та користувачі. Нижче наведено кілька ключових аргументів, які обгрунтовують необхідність розробки програмного модулю для захисту веб-застосунків:

1. Збільшення кількості та складності атак: Загрози в сфері кібербезпеки стають все більш вдосконаленими та різноманітними. Хакерські атаки поширюються [23] не тільки у розмірі, але і у виразності, використовуючи нові методи, такі як атаки штучного інтелекту, щоб обходити традиційні заходи захисту. Розробка програмного модулю дозволяє адаптуватися до сучасних загроз та захищати веб-додатки від новітніх видів атак.

2. Конфіденційність та інтегрітет даних: Веб-додатки обробляють великий обсяг конфіденційної інформації, такої як особисті дані користувачів, фінансові дані, комерційні інформації тощо. Збитки у конфіденційності та інтегрітеті даних можуть призвести до серйозних наслідків, таких як втрата репутації, фінансові збитки та порушення законодавства. Розробка програмного модулю дозволяє ефективно захищати ці дані від несанкціонованого доступу та модифікацій.

3. Відповідність стандартам та регуляціям: Багато організацій зобов'язані відповідати різноманітним стандартам безпеки та регуляціям, таким як GDPR, HIPAA, PCI DSS [24] та іншим. Розробка програмного модулю дозволяє ефективно впроваджувати заходи захисту, необхідні для дотримання цих стандартів, і уникнення можливих штрафів та санкцій.

4. Захист від внутрішніх загроз: На додаток до зовнішніх загроз, внутрішні загрози, такі як зловмисний персонал чи несанкціонований доступ до систем, можуть викликати серйозні проблеми безпеки. Розробка програмного модулю допомагає встановити ефективні механізми внутрішнього контролю та моніторингу для виявлення та запобігання таким загрозам.

5. Захист від нових вразливостей: З розвитком технологій і постійним оновленням веб-додатків, нові вразливості можуть виникати. Розробка програмного модулю дозволяє активно виявляти та виправляти нові вразливості, забезпечуючи стійкий рівень безпеки.

Узагальнюючи, розробка програмного модулю для захисту веб-додатків є важливою складовою стратегії кібербезпеки в сучасному світі, де збільшується кількість та складність кіберзагроз.

2.5 Висновки до другого розділу

Розглянуто узагальнені вимоги щодо безпеки вебзастосунків. Перераховані основні загрози та потенційні збитки для власників застосунків та підкреслені основні способи захисту від кожного з них.

Проаналізовано існуючі вразливості та описані основні методи протидії загрозам, що їх використовують проти вебзастосунків.

Маючи дані про можливі загрози та атаки можна зрозуміти, що необхідно створити методику чи певний алгоритм згідно з яким можна захищати вебзастосунки будь якого розміру.

Розділ 3. РОЗРОБКА МОДУЛЮ ДЛЯ ЗАХИСТУ ВЕБЗАСТОСУНКІВ ВІД ОСНОВНИХ ТИПІВ АТАК

На основі аналізу було вирішено розробити модуль захисту вебзастосунків який може поєднати в собі захист від декількох загроз одночасно. Було вирішено використовувати формальну логіку для того, щоб поєднати захист від SQL-ін'єкції, кросс-сайтового скриптингу та фіксації сесії одночасно.

Застосування формальної логіки в програмуванні є важливим елементом для забезпечення правильності та надійності програмних систем. Формальна логіка дозволяє розробникам математично виражати та аналізувати властивості програм, що допомагає у виявленні та усуненні помилок, а також забезпечує стабільність та ефективність коду. Основні аспекти застосування формальної логіки в програмуванні наступні:

1. Визначення специфікацій:

Формальна логіка використовується для визначення специфікацій програм, включаючи вимоги до функціональності, властивості безпеки, обмеження та інші аспекти. Це допомагає розробникам чітко розуміти, як повинна працювати програма.

2. Досягнення коректності:

Застосування математичних концепцій формальної логіки дозволяє довести коректність програм. Математичні докази можуть визначити, що програма виконує певні властивості безпеки та стабільності.

3. Формальна верифікація:

Використання формальних методів для перевірки правильності програмного коду включає в себе використання спеціальних інструментів та мов програмування для формальної верифікації коду, зокрема перевірку його відповідності визначеним специфікаціям.

4. Математичне моделювання:

Використання формальної логіки для математичного моделювання різних частин програм. Це дозволяє аналізувати та прогнозувати поведінку системи в різних сценаріях.

5. Валідація властивостей коду:

Формальна логіка використовується для визначення та перевірки властивостей коду, таких як відсутність дільників на нуль, відсутність безперервних циклів, та інші аспекти, що можуть впливати на його стабільність.

6. Управління ризиками:

Використання формальної логіки допомагає розробникам виявляти та усувати потенційні проблеми та помилки на ранніх стадіях розробки, що допомагає уникнути небезпек та покращити загальну якість програмного продукту.

7. Забезпечення безпеки:

Використання формальної логіки для визначення та аналізу потенційних безпекових уразливостей у програмному коді та встановлення відповідних заходів захисту.

Використання формальної логіки для створення програмного модуля захисту веб-застосунків від атак дозволяє систематизувати та формалізувати правила, які контролюють і забезпечують безпеку системи. Це дозволяє автоматизовано виявляти підозрілу активність або атаки та сповіщати про них.

Формальна логіка модулю дозволила поєднати захист від атак кросс-сайтового скриптингу, sql-ін’єкції та викрадання кукі-файлів і ці методи захисту можуть працювати одночасно і не заважати один одному. Використання формальної логіки представлено на рисунку 3.1.

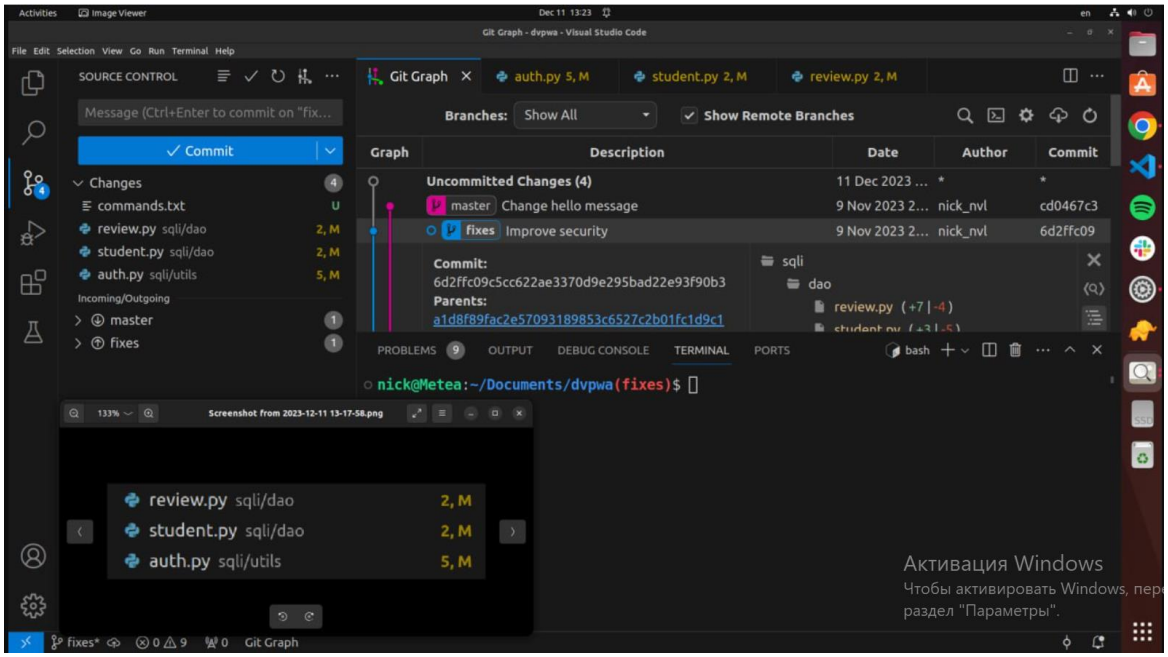

Рис. 3.1. Використання формальної логіки для того , щоб використовувати різні методи захисту одночасно

3.1 Розробка модулю для захисту вебзастосунків

Для того, щоб розробити модуль захисту було проаналізовано алгоритми атак. На основі аналізу алгоритмів атак, було створено алгоритми захисту на основі яких було прийнято рішення, про реалізацію методів захисту від атак.

3.1.1 Розробка алгоритму для захисту від XSS

Для того, щоб правильно захиститися від атаки XSS спочатку потрібно зрозуміти як працює ця атака

Алгоритм атаки кросс-сайтового скриптингу представлено на рисунку 3.2

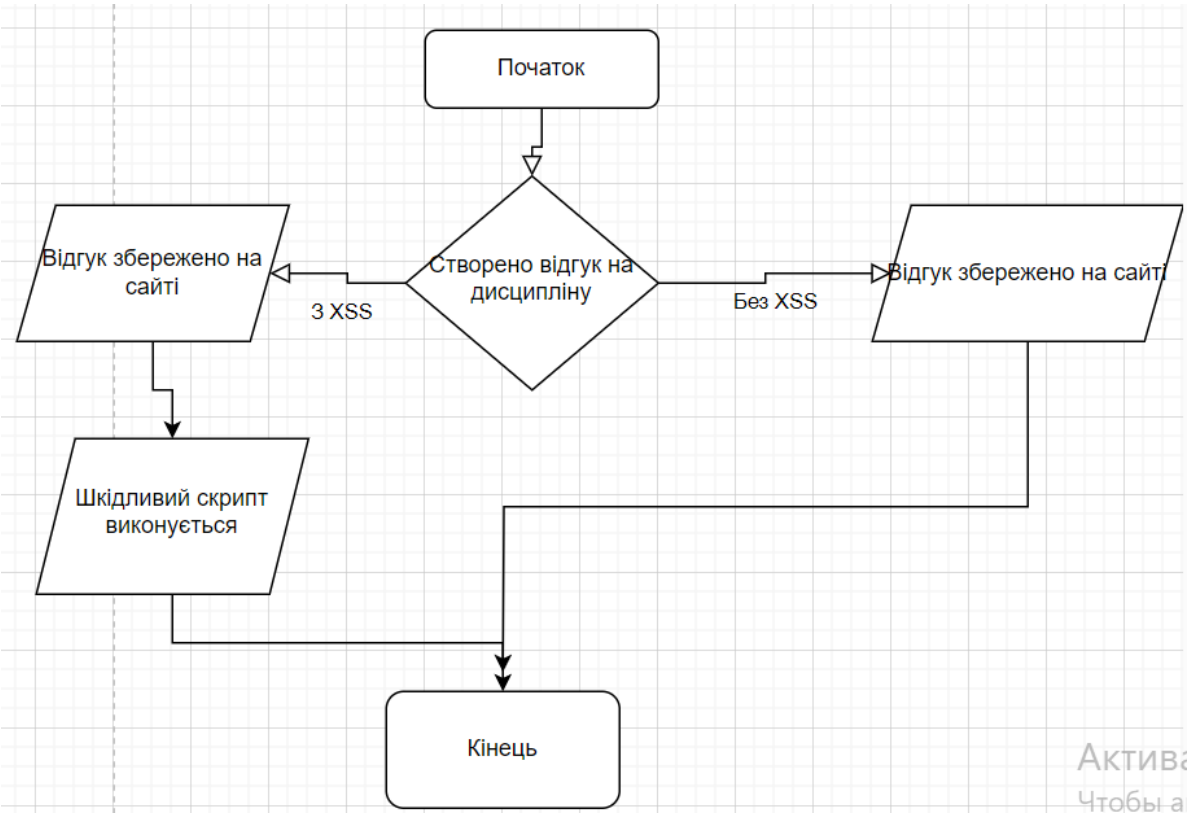

Рис 3.2 Схема дії кросс-сайтового скриптингу на веб-сайти

Для того, щоб захиститися від виконання небажаних скриптів було запропоновано наступний алгоритм, який перевіряє кожен створений відгук на наявність атаки кросс-сайтового скриптингу. (рис. 3.3)

Рис 3.3. Алгоритм програми для захисту вебзастосунків від кросс-сайтового скриптингу

3.1.2 Розробка алгоритму для захисту від SQL-ін’єкції

Для того, що зрозуміти як створити алгоритм захисту від атаки ін’єкцією спочатку потрібно зрозуміти алгоритм атаки.

Алгоритм атак SQL-ін’єкції представлено на рисунку 3.4

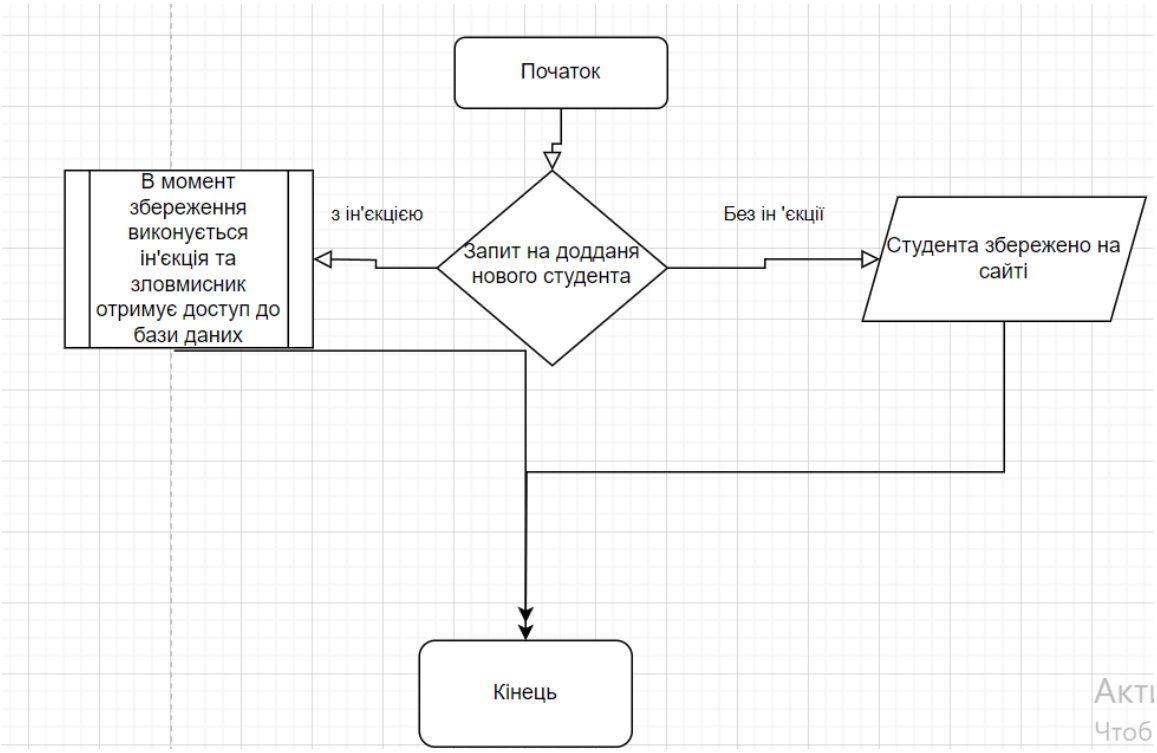

Рис 3.4. Алгоритм виконання SQL-ін’єкції

Для того, щоб захистити вебзастосунок від атаки SQL-ін’єкції було запропоновано наступний алгоритм (рис. 3.5), який фільтрує запит і не дозволяє робити незаплановані дії.

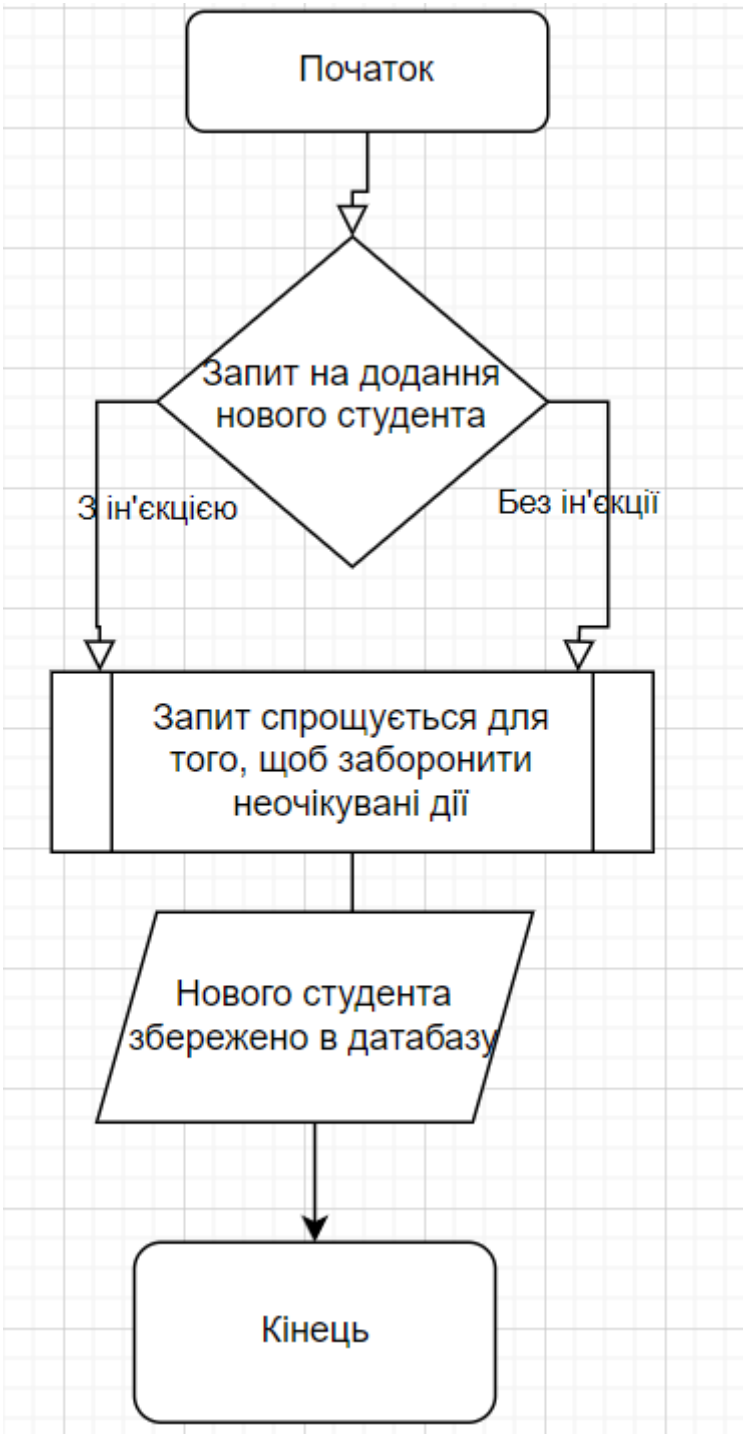

Рис 3.5. Алгоритм захисту вебзастосунку від ін’єкції

3.1.3 Розробка алгоритму для захисту від атаки фіксації сесії

Для того, що зрозуміти як створити алгоритм захисту від фіксації сесії спочатку потрібно зрозуміти алгоритм атаки.

Приклад алгоритму фіксації сесії через викрадання кукі-файлі представлено на рисунку 3.6

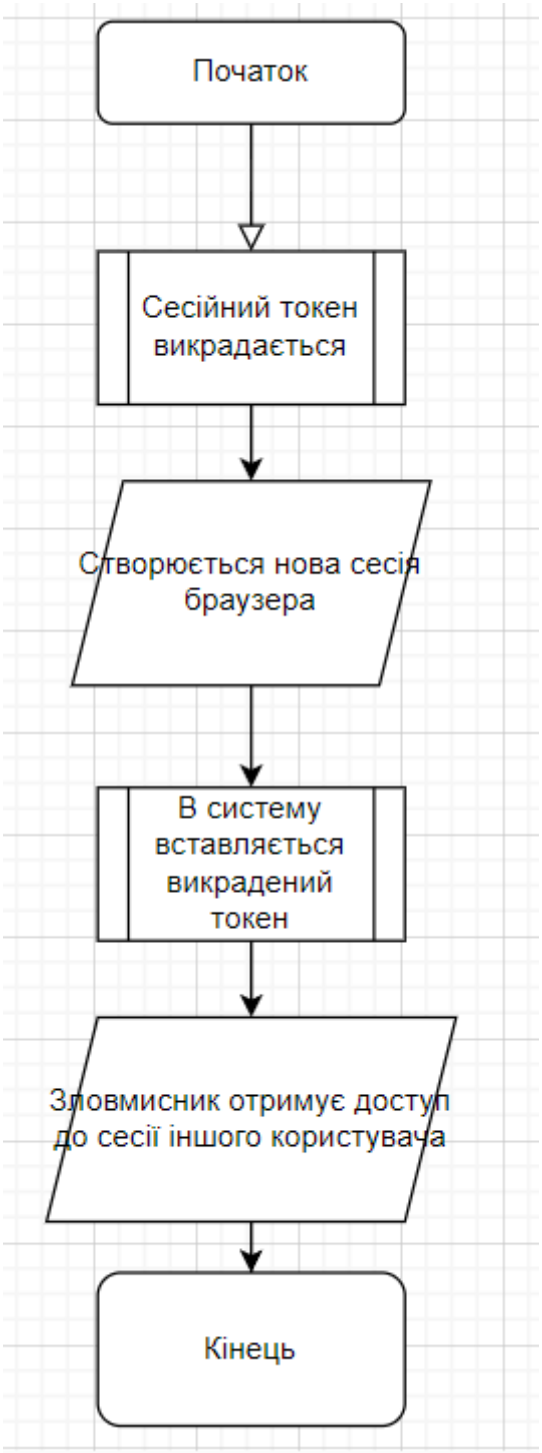

Рис 3.6. Приклад алгоритму атаки фіксації сесії

Для того, щоб захистити вебзастосунок від цієї атаки було розроблено алгоритм представлений на рисунку 3.7.

Захист від цієї атаки полягає в тому, щоб проводити регулярний оббіг токенів сесій. Наприклад можна оновлювати токен, кожен раз коли користувач закінчує роботу з сервісом.

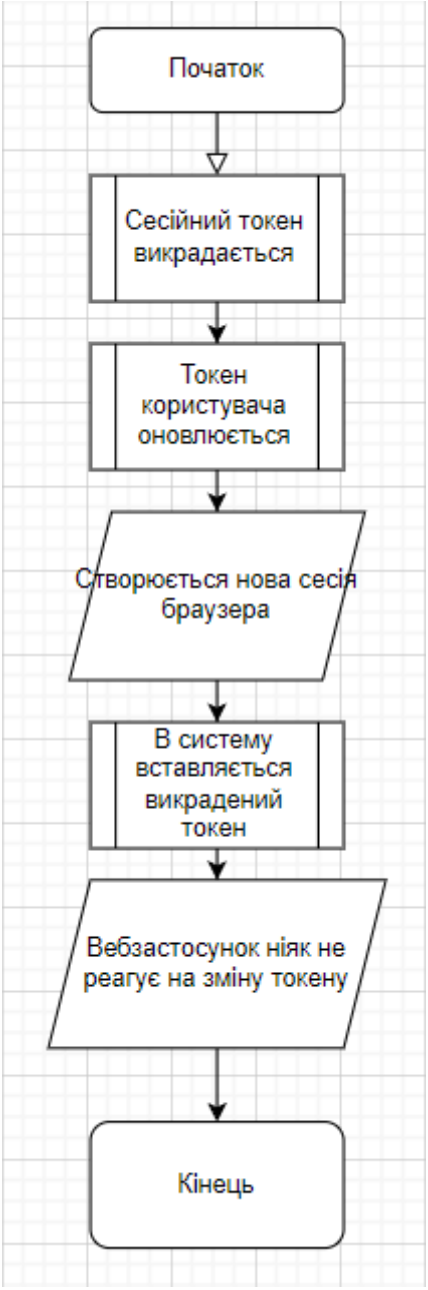

Рис 3.7. Алгоритм для захисту вебзастосунку від атак фіксації сесії

3.2 Захист від атак кросс-сайтового скриптингу

Приклад атаки кросс-сайтового скриптингу на веб-додаток наданий на рисунках 3.8 та 3.9. На цьому веб-сайті уявного університету є можливість додавати огляд до курсів. Зловмисник може спробувати використати цю функцію для того щоб внести шкідливий скрипт на вебсайт:

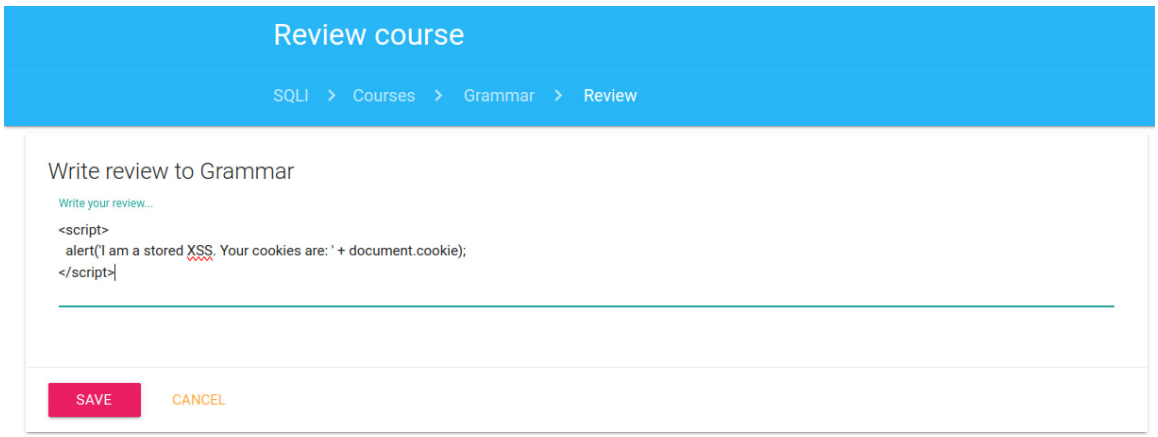

Рис 3.8. Приклад скрипту який може внести зловмисник.

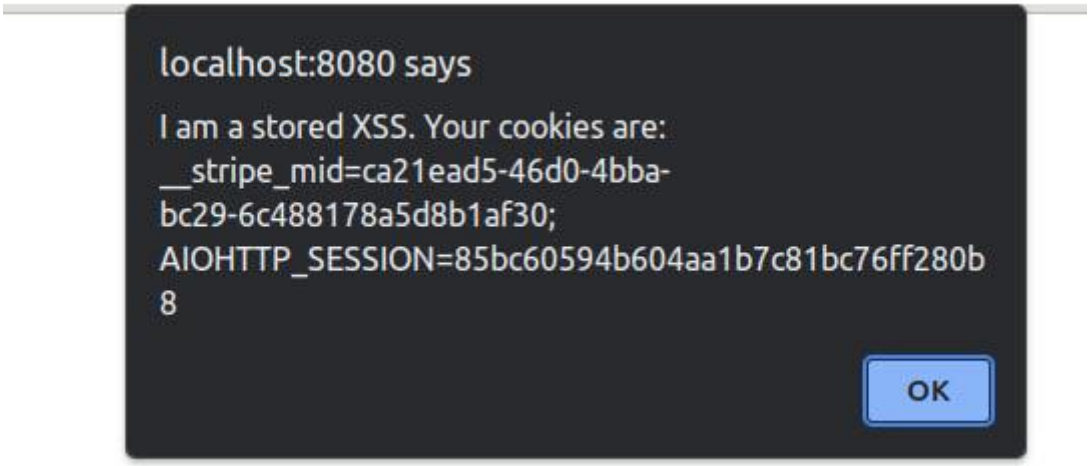

Рис. 3.9. Результат виконання шкідливого скрипта

Через незахищеність веб-сайту зловмисник може виконувати атаки кросс-сайтового скриптингу та отримувати дані до яких він не повинен мати доступу.

Захистити веб-сайт від атак кросс-сайтового скриптингу можливо декількома способами, але я пропоную це зробити за допомогою очищення даних введених користувачем перед відтворенням їх на HTML сторінці веб-сайту.

Для того щоб захистити додаток від цього типу атаки було змінено оригінальний код (рис. 3.10) на більш захищену версію (рис 3.11) яка не дозволяє зловмисникам використовувати крос-сайтовий скриптинг у цьому веб додатку:

```
    @staticmethod
    async def create(conn: Connection, course_id: int,
                     review_text: str):


        q = ('INSERT INTO course_reviews (course_id, review_text) '
             'VALUES (%(course_id)s, %(review_text)s)')
        params = {'course_id': course_id,
                  'review_text': review_text}
        async with conn.cursor() as cur:
            await cur.execute(q, params)
```

Рис. 3.10. Оригінальний код який не захищав від кросс-сайтового скриптингу

```
    @staticmethod
    async def create(conn: Connection, course_id: int, review_text: str):
        # Escape the review_text before storing it in the database
        review_text_escaped = escape(review_text)

        q = ('INSERT INTO course_reviews (course_id, review_text) '
             'VALUES (%(course_id)s, %(review_text)s)')
        params = {'course_id': course_id, 'review_text': review_text_escaped}

        async with conn.cursor() as cur:
            await cur.execute(q, params)
```

Рис. 3.11. Змінений код який захищає від кросс-сайтового скриптингу

Тепер при спробі виконати той самий скрипт (рис 3.12) на оновленій версії веб-сайту ми отримуємо наступний результат (рис 3.13).

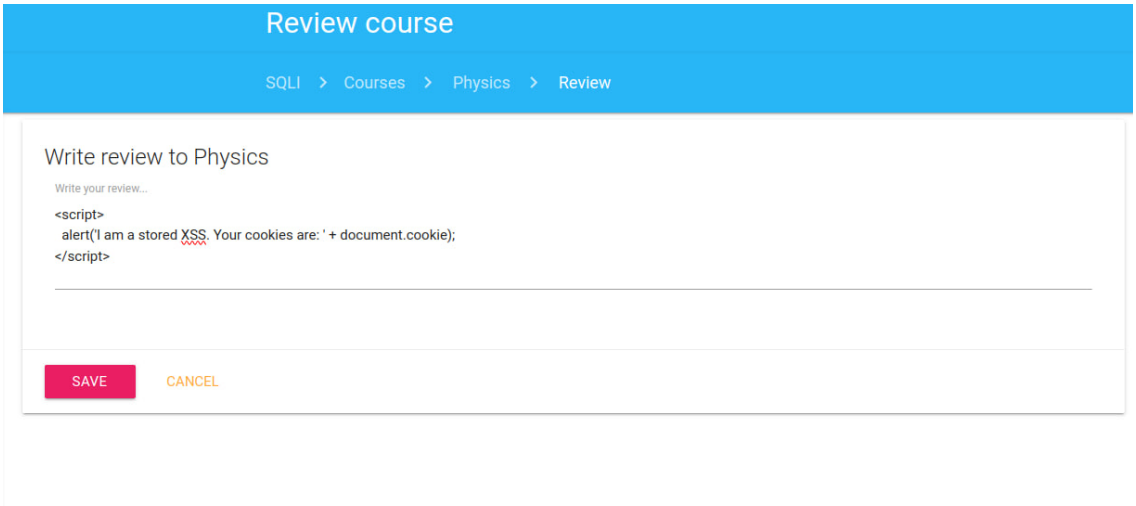

Рис. 3.12. Спроба виконати скрипт на оновленій версій веб-сайту

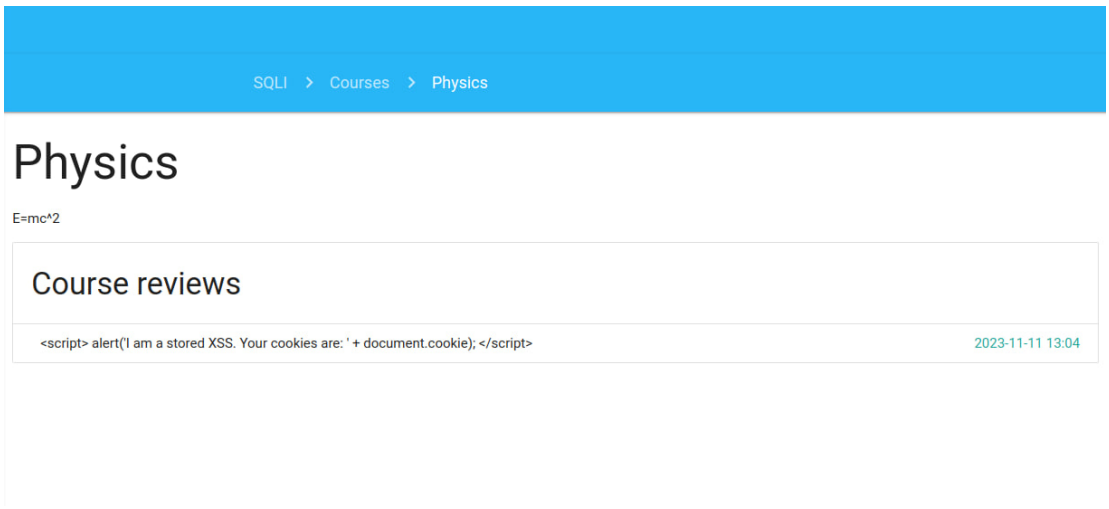

Рис. 3.13. Невдалий результат спроби атакувати веб-сайт

В оновленій версії веб-сайту скрипт зловмисника не виконується і тому можна вважати, що веб-сайт захищено від цього типу атаки.

3.3 Захист від атак SQL-Ін’єкції

Приклад атаки SQL-ін’єкції на веб-додаток наданий на рисунках 3.14 – 3.15.

На цьому веб-сайті уявного університету у адмінів є можливість додавати нових студентів до датабази. Але метод який додає нових студентів не захищений через нього дуже легко зробити SQL-ін’єкцію.

Рис. 3.14. Спроба виконання SQL-ін’єкції

500 Internal Server Error

Traceback:

```
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/site-packages/aiohttp/web_protocol.py", line 418, in start
    resp = await task
  File "/usr/local/lib/python3.7/site-packages/aiohttp/web_app.py", line 458, in _handle
    resp = await handler(request)
  File "/usr/local/lib/python3.7/site-packages/aiohttp/web_middlewares.py", line 119, in impl
    return await handler(request)
  File "/app/sqli/middlewares.py", line 22, in session_middleware
    return await middleware(request, handler)
  File "/usr/local/lib/python3.7/site-packages/aiohttp_session/__init__.py", line 152, in factory
    response = await handler(request)
  File "/app/sqli/middlewares.py", line 45, in middleware
    response = await handler(request)
  File "/usr/local/lib/python3.7/site-packages/aiohttp_jinja2/__init__.py", line 116, in context_processors_middleware
    return await handler(request)
  File "/usr/local/lib/python3.7/site-packages/aiohttp_jinja2/__init__.py", line 91, in wrapped
    context = await coro(*args)
  File "/app/sqli/views.py", line 59, in students
    students = await Student.get_many(conn)
  File "/app/sqli/dao/student.py", line 36, in get_many
    await cur.execute(q, params)
  File "/usr/local/lib/python3.7/site-packages/aiopg/cursor.py", line 114, in execute
    yield from self._conn._poll(waiter, timeout)
  File "/usr/local/lib/python3.7/site-packages/aiopg/connection.py", line 238, in _poll
    yield from asyncio.wait_for(self._waiter, timeout, loop=self._loop)
  File "/usr/local/lib/python3.7/asyncio/tasks.py", line 416, in wait_for
    return fut.result()
  File "/usr/local/lib/python3.7/site-packages/aiopg/connection.py", line 135, in _ready
    state = self._conn.poll()
psycopg2.ProgrammingError: relation "students" does not exist
LINE 1: SELECT id, name FROM students
                             ^
```

Рис 3.15. Результат успішної SQL-ін’єкції.

Як ми можем бачити після виконання команди сторінка веб-додатку не завантажується, а натомість видає помилку про те, що таблиця студентів відсутня – це результат атаки SQL-ін’єкції.

Для того щоб забезпечити захист від SQL-ін’єкції, було внесено наступні зміни до логіки веб-сайту (рис.3.16 – рис 3.17)

```
    @staticmethod
    async def create(conn: Connection, name: str):
        q = ("INSERT INTO students (name) "
             "VALUES ('%(name)s')" % {'name': name})
        async with conn.cursor() as cur:
            await cur.execute(q)
```

Рис 3.16. Оригінальний код який не має захисту від SQL-ін’єкцій.

```
@staticmethod
async def create(conn: Connection, name: str):
    q = "INSERT INTO students (name) VALUES (%s)"
    values = (name,)
    async with conn.cursor() as cur:
        await cur.execute(q, values)
```

Рис 3.17. Код зі змінами який захищає додаток від SQL-ін’єкцій.

Тепер при спробі виконати ін’єкцію (рис 3.18) шкідливий код не виконується і новий студент просто додається до датабази (рис 3.19)

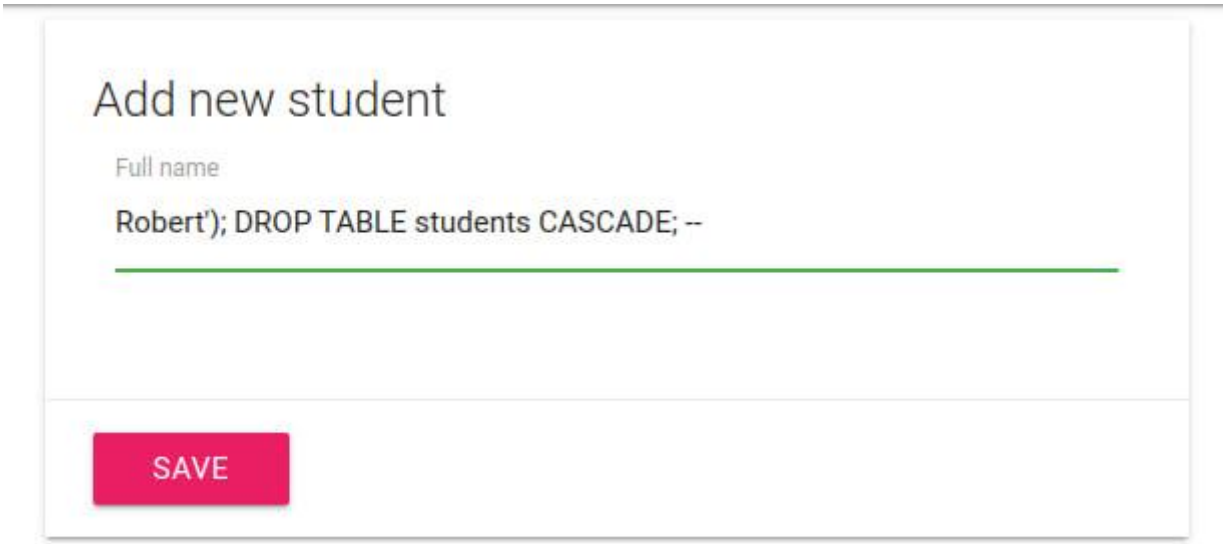

Рис 3.18. Спроба виконання ін’єкції на оновленій версії веб-додатку

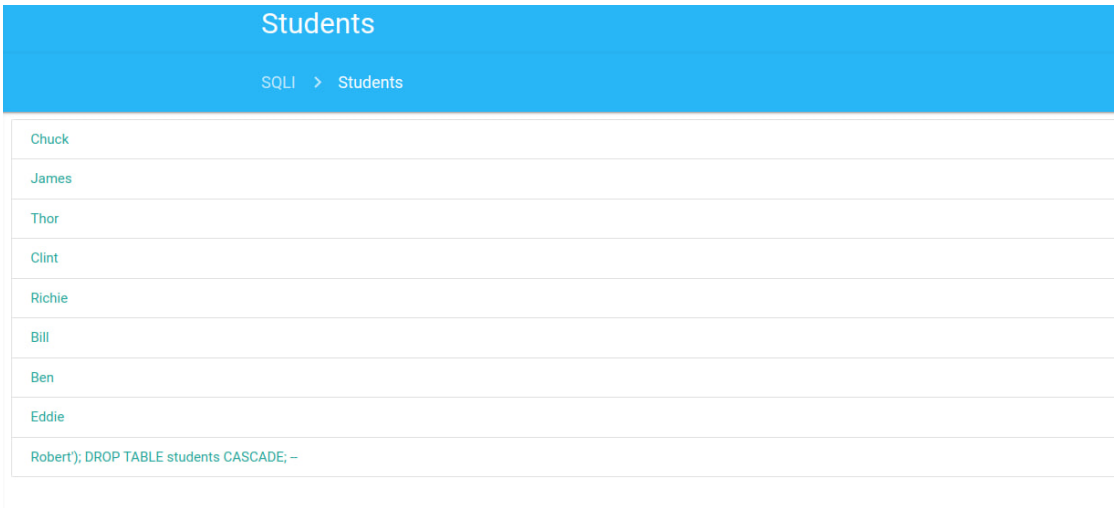

Рис 3.19. Результат спроби виконати ін’єкцію на оновленій версії веб-додатку

Як ми можемо бачити після внесення змін до логіки додатку видалення датабази студентів за допомогою SQL-ін’єкції не відбувається, отже ця вразливість вирішена.

3.4 Захист від атаки фіксації сеансу

Приклад атаки фіксації сеансу на веб-додаток надано на рисунках 3.20 – 3.24.

На цьому веб-сайті користувачі можуть заходити в свої акаунти і сайт зберігає кожну індивідуальну сесію в кукі-файлах. Якщо зловмисник зможе викрасти кукі-файли іншого користувача, то він зможе відтворити його сеанс і оперувати сайтом від чужого імені та потенційно отримати доступ до функцій чи інформації до яких він не повинен мати доступу.

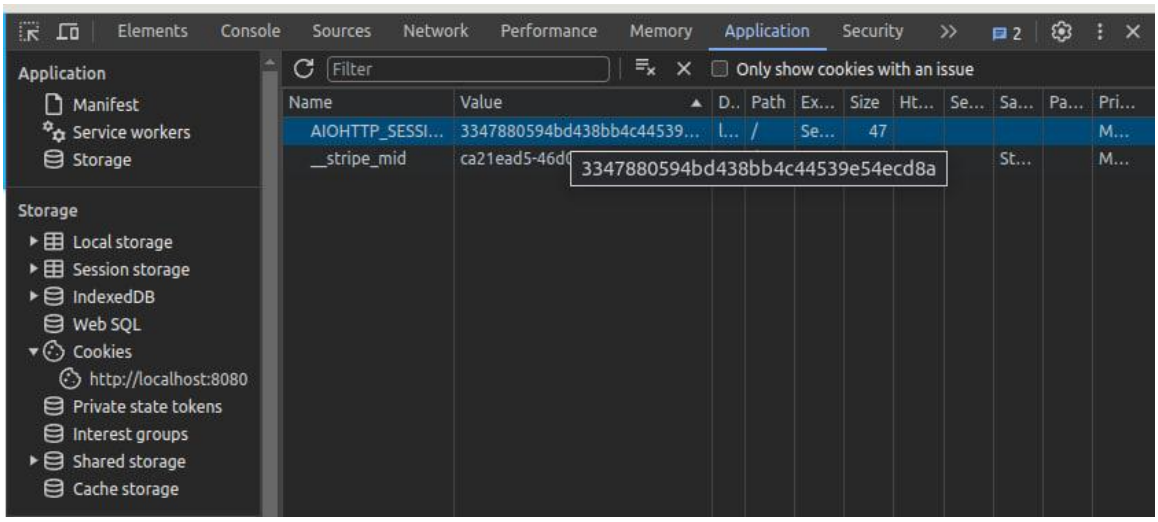

Рис 3.20. Приклад отримання кукі-файлу сеансу

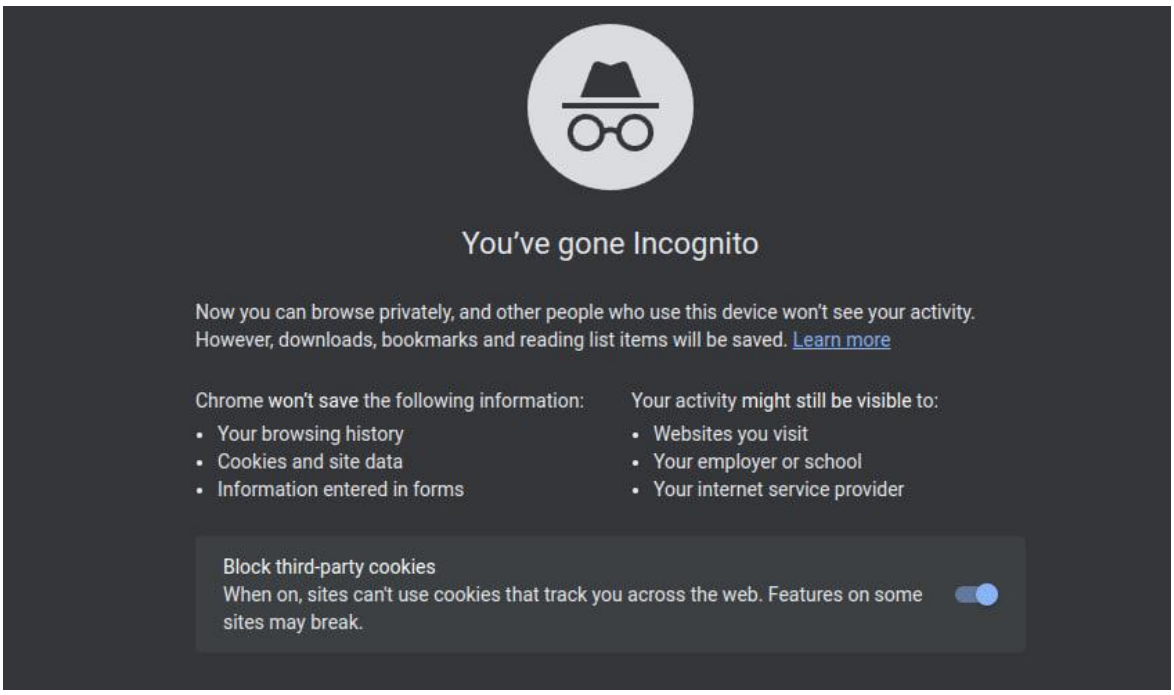

Рис 3.21. Перехід в режим "інкогніто" в браузері для того щоб втратити локальний сеанс

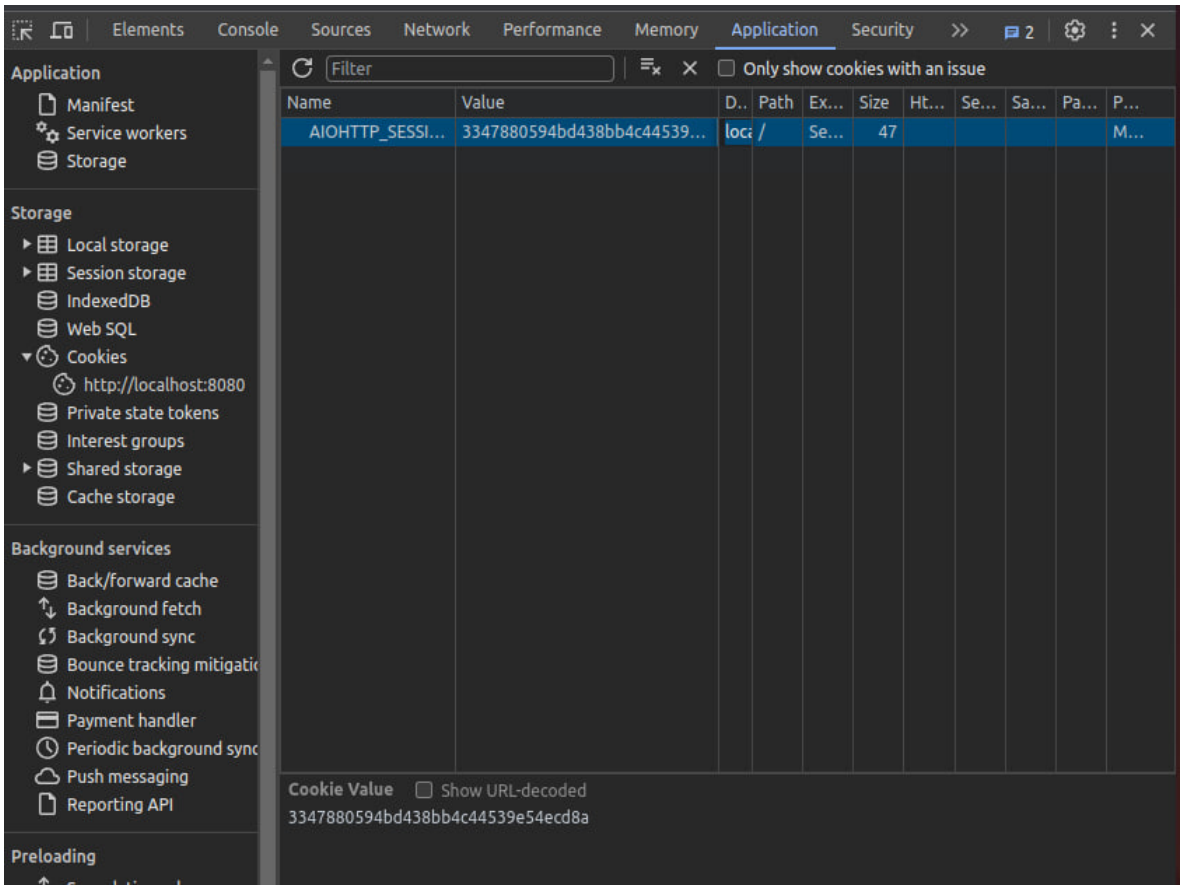

Рис 3.22. Перевірка кукі-файлу сеансу в режимі "інкогніто"

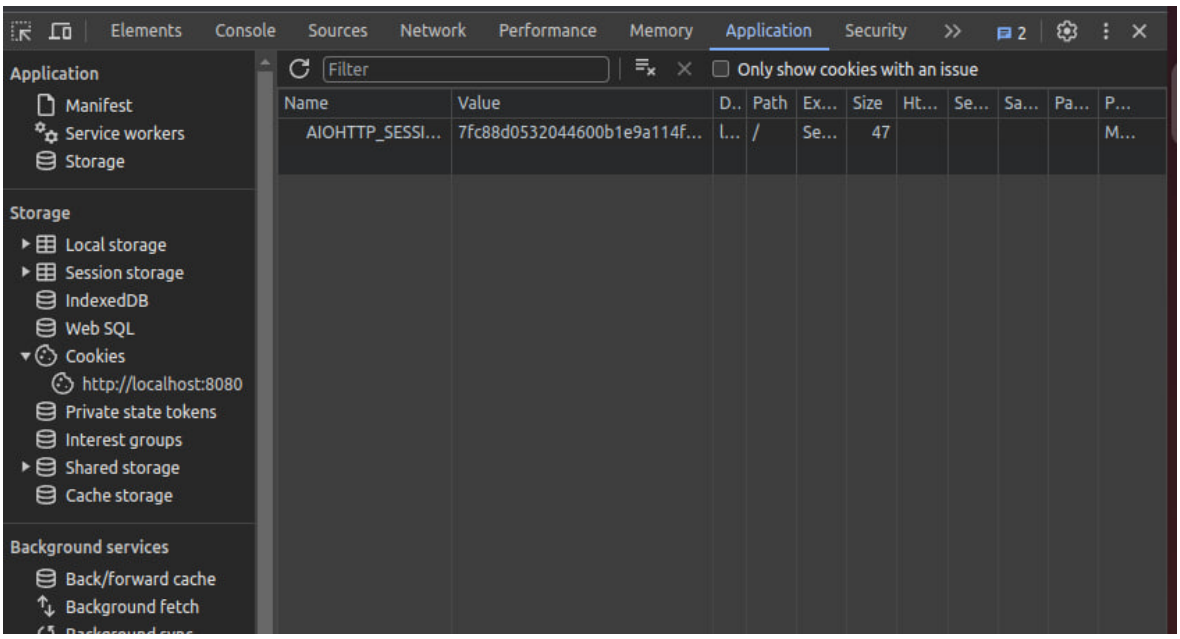

Рис 3.23. Вставлення отриманого раніше файлу кукі-сеансу в режим “інкогніто”

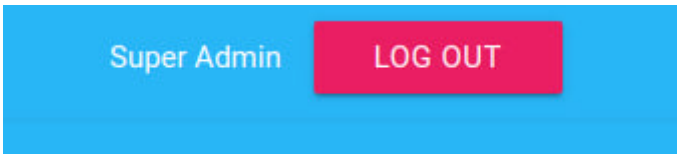

Рис 3.24. Результат атаки фіксації сеансу.

Як ми можемо бачити, через незахищеність веб-сайту зловмисник може виконувати атаки фіксації сесії та отримувати доступ до сесій інших користувачів.

Існують різні способи захистити веб-додаток від атаки такого типу, але я пропоную це зробити за допомогою реалізації ротації ідентифікатора сеансу під час кожного входу, виходу з системи, зміни user_id або дозволів, для цього я змінив логіку системи (рис 3.25), щоб повторно створювати ідентифікатор сеансу в цих конкретних сценаріях. Це було досягнуто завдяки оновленню ідентифікатор сеансу кожного разу, коли виконуються такі такі дії, як вхід, вихід із системи або змінення даних користувача.

```python
async def rotate_session(request: Request, user_id: Optional[int]):
    session = await get_session(request)
    session.invalidate()  # Invalidate the current session
    new_session(request)  # Create a new session
    session['user_id'] = user_id  # Set the new user_id in the session

async def logout(request: Request):
    await rotate_session(request, None)
```

Рис. 3.25. Логіка для інвалідації сеансу

Після цих змін при спробі виконати атаку фіксації сеансу (рис 3.26-3.27) зловмисник не отримує доступу до сеансу іншого користувача (рис 3.28)

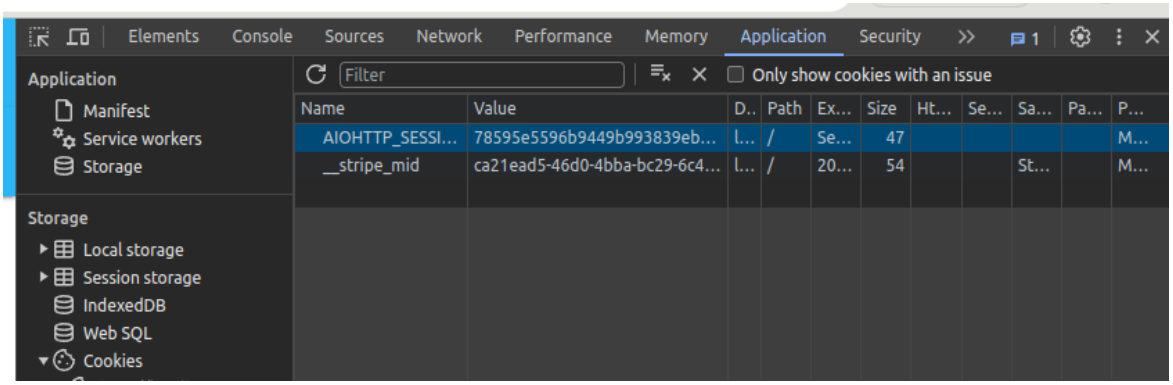

Рис. 3.26. Спроба копіювання токену сеансу

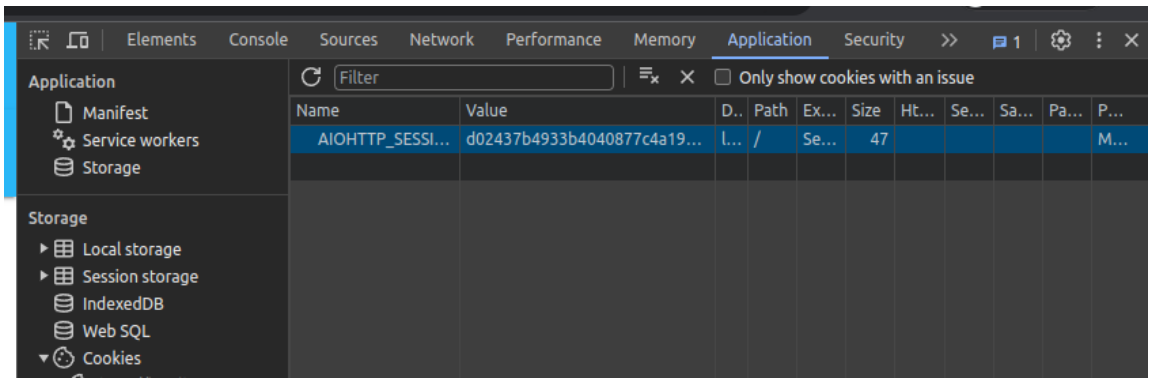

Рис. 3.27. Спроба підміни токену сеансу

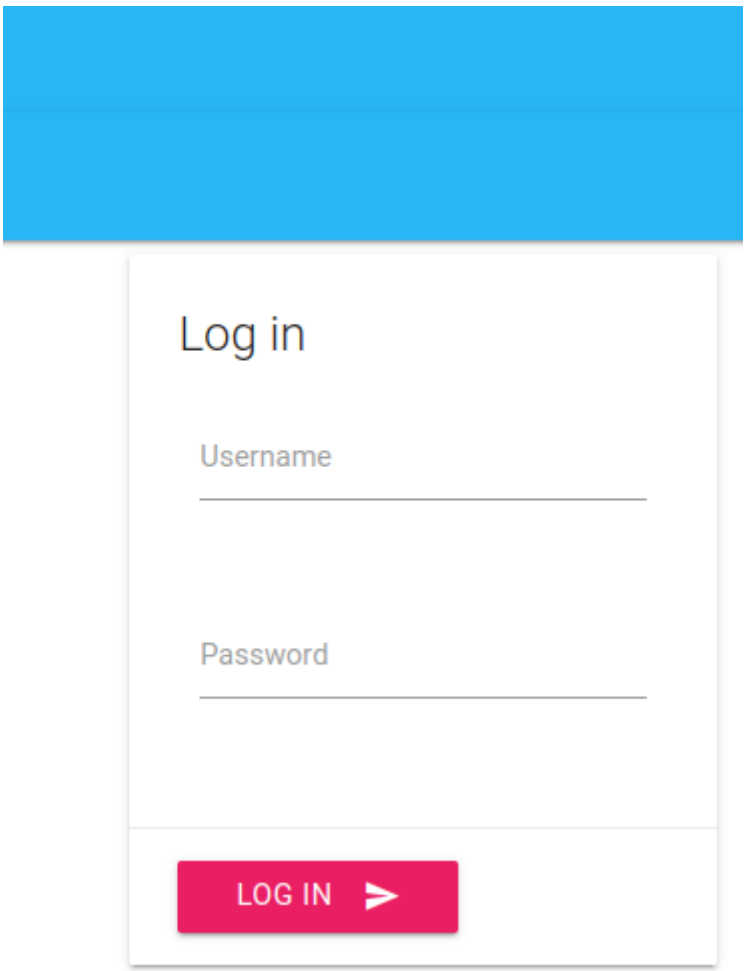

Рис. 3.28 Результат атаки фіксації сеансу

Як ми можемо бачити після внесення змін до логіки додатку зловмисник більше не може перехопити сеанс іншого користувача, отже можна вважати, що ця вразливість була вирішена.

3.5 Висновки третього розділу

Було розроблено модуль захисту веб-додатку на основі формальної логіки який може одночасно зупиняти такі атаки як кросс-сайтовий скриптинг, SQL-ін’єкція та фіксації сесії за допомогою викрадання кукі-файлів. Було продемонстровано на прикладі веб-сайту університету яким чином модуль може захистити вебзастосунок від атак.

Застосування формальної логіки у виявленні атак допомагає системі автоматизовано визначати, аналізувати та реагувати на потенційні загрози.

Також було розібрано потенційну архітектуру веб-додатку для якої можна використовувати розроблену логіку.

Запропонований метод захисту веб-додатків може повністю захистити веб-додаток від атак за допомогою атак крос-сайтового скриптингу, SQL-ін’єкцій та фіксації сеансів одночасно за допомогою формальної логіки.

Розділ 4. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

4.1 Екологічна безпека літосфери.

Екологічна безпека літосфери є важливим аспектом у забезпеченні сталого розвитку та збереження природних ресурсів. Літосфера - це зовнішній оболонковий шар Землі, який включає поверхневий шар кори, гірські породи, ґрунти та інші тверді компоненти земної поверхні. [25]

Охорона навколишнього середовища у сфері літосфери включає в себе ряд заходів та принципів:

1. Управління земельними ресурсами: Збереження родючості ґрунтів та використання їх урожайного потенціалу є ключовими завданнями. Запобігання ерозії ґрунтів, введення технологій сталого землекористування та врахування принципів агроекології сприяють підтримці екологічної стійкості літосфери.

2. Мінімізація індустріального впливу: Здійснення виробництва та видобутку ресурсів з урахуванням принципів екологічної безпеки дозволяє уникнути негативних наслідків для літосфери. Впровадження сучасних технологій та стандартів у сфері промисловості сприяє зменшенню забруднення та деградації літосфери. [26]

3. Охорона природних ландшафтів: Збереження природних ландшафтів та екосистем, які є частиною літосфери, важливо для підтримання біорізноманіття та стабільності екосистем. Створення природних заповідників, національних парків та інших охоронних територій сприяє збереженню унікальних природних утворень.

4. Впровадження систем управління відходами: Раціональне використання та утилізація відходів допомагає уникнути їх накопичення на поверхні літосфери. Сортування та переробка відходів, а також використання енергії відновлюваних джерел може значно зменшити негативний вплив на довкілля.

5. Моніторинг та наукові дослідження: Систематичний моніторинг стану літосфери та наукові дослідження її процесів дозволяють вчасно виявляти проблеми та розробляти ефективні заходи для їх вирішення.

6. Освіта та свідомість громадськості: Поширення екологічної освіти серед населення сприяє формуванню екологічної культури та відповідального ставлення до природних ресурсів. Громадська участь у розв'язанні екологічних проблем є ключовою для досягнення сталого розвитку літосфери.

Розуміння та запровадження основних заходів та принципів у забезпеченні екологічної безпеки літосфери є дуже важливим.

1. Стале землекористування:

Контурна обробка: Розробка систем контурної обробки землі, включаючи застосування терасування та зведення берегових насипів, для зменшення швидкості стікання води та ерозії.

Агрофорестри: Впровадження методів агрофорестрі, що поєднують сільське господарство з висадженням дерев, для збереження ґрунтової родючості та контролю ерозії.

2. Методи боротьби з ерозією та деградацією ґрунтів:

Контроль водостоку: Впровадження систем контролю водостоку для уникнення стікання води та втрати верхнього шару ґрунту.

Агротехнічні заходи: Використання агротехнічних прийомів, таких як мінімальна обробка ґрунту та покриви рослинності, для підтримки його структури та захисту від ерозії. [27]

3. Методи управління відходами:

Рециклінг: Впровадження систем рециклінгу для максимального використання та переробки відходів, щоб зменшити негативний вплив на ґрунт.

Компостування: Заохочення компостування як методу обробки органічних відходів для отримання природного добрива та покращення ґрунтової структури. [28]

4. Захист від забруднення:

Біоремедіація: Використання біоремедіації для очищення ґрунту від забруднюючих речовин та відновлення його природних властивостей.

Відсівання та ізоляція: Використання технологій для відсівання та ізоляції забруднюючих речовин для запобігання їхньому проникненню в ґрунт.

5. Охорона природних екосистем:

Екологічна реконструкція: Проведення проектів екологічної реконструкції для відновлення природних екосистем та їх функціональності.

Заповідництво: Заснування заповідників для збереження унікальних природних комплексів та забезпечення їх невтручання.

6. Нормативно-правове регулювання:

Ліцензування та моніторинг: Введення систем ліцензування для господарської діяльності, що може впливати на літосферу, та постійний моніторинг викидів та впливу. [29]

Штрафи та відповідальність: Застосування жорстких штрафів та відповідальності за порушення екологічних норм та стандартів.

7. Громадська участь та освіта:

Екологічні програми: Розробка та впровадження освітніх програм для громадськості щодо ефективного збереження літосфери.

Громадські групи: Залучення громадськості до моніторингу стану літосфери та участі в розробці та реалізації заходів її охорони. [26]

Ці заходи та принципи спрямовані на створення умов для збереження та відновлення екологічної стійкості літосфери, забезпечуючи екологічну безпеку та стійкий розвиток природних систем.

Загальною метою захисту літосфери є створення умов для того, щоб літосфера залишалася життєздатною та функціонувала в інтересах сучасного покоління і майбутніх поколінь, забезпечуючи екологічну безпеку та сталість природних систем. [30]

ВИСНОВКИ

В процесі переддипломної практики було: розглянуто законодавчу базу України з інформаційної безпеки та методи захисту інформаційних додатків. Нормативно-правова база в Україні регулює питання захисту інформаційних активів та визначає вимоги до веб-застосунків щодо їх безпеки.

Зроблено порівняльний аналіз методів захисту інформації вебзастосунків від різних типів атак, що дало можливість обрати методи захисту для подальшого використання у розробці програмного модуля.

Проведено аналіз атак та методів їх виконання, що дало можливість виявити найбільш поширенні види атак на вебзастосунки, яке дало розуміння для розробки алгоритму програмного продукту для програмного модулю мовою програмування Python.

Було розроблено та протестовано програмний модуль для захисту вебзастосунків з використанням формальної логіки, що дало можливість поєднання програмних продуктів для виявлення кросс-сайтового скриптингу, SQL-ін’єкції, фіксації сесії у одному програмному модулі.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Боротьба з кіберзлочинністю в умовах дії воєнного стану: Закон 2149-IX - 2022 - Режим доступу: World Wide Web.- https://jurliga.ligazakon.net/analitycs/210562_borotba-z-kberzlochinnstyu-v-umovakh-d-vonnogo-stanu-zakon-2149-ix

2. Загальний регламент про захист даних (GDPR) - Режим доступу: World Wide Web. - https://gdpr-text.com/uk/

3. Закон України [Електронний ресурс] : Про захист інформації в інформаційно-телекомунікаційних системах – 1994. – Режим доступу: World Wide Web. – URL: https://zakon.rada.gov.ua/laws/show/80/94-вр#Text.

4. Закон України [Електронний ресурс] Про авторське право і суміжні права — 2022 -– Режим доступу: World Wide Web. – URL: https://itd.rada.gov.ua/billInfo/Bills/pubFile/761781

5. Що таке веб додаток? Різниця між сайтом, веб-додатком, SPA і PWA - Режим доступу: World Wide Web.- https://webcase.com.ua/uk/blog/cho-takoe-web-prilozhenie-vse-vidy/

6. Using Steganography within WEB Security and Its Application in User Login System – Toghrul VALIBAYLI, Ahmet GÜRHANLI – 2020 - Режим доступу: World Wide Web. - https://www.academia.edu/42794094/Using_Steganography_within_WEB_Security_and_Its_Application_in_User_Login_System

7. What is the least significant bit (LSB)? - Режим доступу: World Wide Web. - https://www.lenovo.com/us/en/glossary/least-significant-bit

8. Хакери поширюють шкідливе програмне забезпечення у порожніх зображеннях - 2023 - Режим доступу: World Wide Web. - https://internetua.com/hakeri-poshiruauat-shkidlive-programne-zabezpecsennya-u-porojnih-zobrajennyah

9. What is Steganography? A Complete Guide – 2023- Режим доступу: World Wide Web. https://intellipaat.com/blog/what-is-steganography/#no15

10. Криптографічні методи захисту інформації. Контроль цілісності програмних та інформаційних ресурсів. - 2020 - Оксана Десятник - Режим доступу: World Wide Web. - https://classmill.com/659/112/m/xnb7A

11. ISSP \ Домен 06. Криптография. Часть 3 - 2010 - Режим доступу: World Wide Web. - http://dorlov.blogspot.com/2010/09/issp-06-3.html

12. ECC Vs RSA Certificate Difference Explained - Janki Mehta - Режим доступу: World Wide Web. - https://cheapsslweb.com/resources/ecc-vs-rsa-certificate

13. What are MD5m SHA-1 and SHA-256 Hashes, and how do I check them? - 2017 - Chris Hoffman - Режим доступу: World Wide Web. - https://www.howtogeek.com/67241/htg-explains-what-are-md5-sha-1-hashes-and-how-do-i-check-them/

14. What is an SSL/TLS Certificate - Режим доступу: World Wide Web. - https://aws.amazon.com/what-is/ssl-certificate/

15. What are Authentication protocols in Cryptography? - 2020 – Logsign - Режим доступу: World Wide Web. - https://www.logsign.com/blog/what-are-authentication-protocols-in-cryptography/

16. What is IPsec? - Режим доступу: World Wide Web. - https://aws.amazon.com/what-is/ipsec/

17. Characteristics, Types and Applications of Cryptography - 2021 - Soumyaa Rawat - Режим доступу: World Wide Web. - https://www.analyticssteps.com/blogs/characteristics-types-and-applications-cryptography

18. «Вразливості веб-застосунків та методи захисту від них» - Олецький О. В. – 2021 - Режим доступу: World Wide Web.- https://ekmair.ukma.edu.ua/server/api/core/bitstreams/b94fea39-69c0-4b92-9f23-3ab5eb1b04ee/content

19. Паролі: базовий захист усіх акаунтів – 2021 - Режим доступу: World Wide Web. - https://yak.dslua.org/articles/paroli-bazovyy-zakhyst-usikh-akauntiv/

20. Як захистити свій сайт від атак методом грубої сили (Brute-force) – 2020 - Режим доступу: World Wide Web.- https://sebweo.com/yak-zahistiti-svij-sajt-vid-atak-metodom-gruboyi-sili-brute-force/

21. Уразливості для фіксації сеансу – 2023 - Режим доступу: World Wide Web. - https://cqr.company/ua/web-vulnerabilities/session-fixation-vulnerabilities/

22. Why Use Git for Your Organization - Режим доступу: World Wide Web. - https://www.atlassian.com/git/tutorials/why-git\

23. Recent Cyber Attacks - 2023 - Nivedita James Palatty - 2023 - Режим доступу: World Wide Web. - https://www.getastra.com/blog/security-audit/recent-cyber-attacks/

24. A brief look at 4 major Data Compliance Standards: GDRP, HIPAA, PCI, DSS, CCPA - 2020 - Режим доступу: World Wide Web. - https://www.pentasecurity.com/blog/4-data-compliance-standards-gdpr-hipaa-pci-dss-ccpa

25. Що таке літосфера – 2023 - Режим доступу: World Wide Web. - https://translations.com.ua/litosfera.html

26. Закон України [Електронний ресурс] : Про Основні напрями державної політики України у галузі охорони довкілля, використання природних ресурсів та забезпечення екологічної безпеки — 1998 - Режим доступу: World Wide Web. – URL: https://ips.ligazakon.net/document/F980188

27. Мінімальна обробка грунту (Mini-till) - Режим доступу: World Wide Web. – URL: https://superagronom.com/slovnik-agronoma/minimalna-obrobka-gruntu-mini-till-id20491

28. Компостування: ефективно, екологічно, корисно для грунтів - 2017 - Режим доступу: World Wide Web. – URL: https://superagronom.com/blog/115-kompostuvannya-efektivno-ekologichno-korisno-dlya-gruntiv

29. Закон України [Електронний ресурс] : Про ліцензування видів господарської діяльності — 2015 - Режим доступу: World Wide Web. – URL: https://zakon.rada.gov.ua/laws/show/222-19#Text

30. захист літосфери - Режим доступу: World Wide Web. – URL: http://ni.biz.ua/14/14_8/14_86778_zashchita-litosferi.html

Додаток А

Код програми

```
import logging
from datetime import datetime
from itertools import groupby

from aiohttp.web import Request, HTTPFound
from aiohttp.web_exceptions import HTTPNotFound, HTTPForbidden
from aiohttp_jinja2 import template
from aiohttp_session import get_session
from trafaret import DataError

from sqli.dao.course import Course
from sqli.dao.mark import Mark
from sqli.dao.review import Review
from sqli.dao.student import Student
from sqli.dao.user import User
from sqli.schema.forms import EVALUATE_SCHEMA
from sqli.utils.auth import get_auth_user, authorize

log = logging.getLogger(__name__)

@template('index.jinja2')
async def index(request: Request):
    app: Application = request.app
    auth_user = await get_auth_user(request)

    session = await get_session(request)
    last_visited = session.get('last_visited', 'never')
    session['last_visited'] = datetime.now().isoformat()
```

```
    errors = []

    if request.method == 'POST':
        if auth_user:
            raise HTTPForbidden()
        data = await request.post()
        username = data['username']
        password = data['password']
        async with app['db'].acquire() as conn:
            user = await User.get_by_username(conn, username)
        if user and user.check_password(password):
            session['user_id'] = user.id
            auth_user = user
        else:
            errors.append('Invalid username or password')
    return {'last_visited': last_visited,
            'errors': errors,
            'auth_user': auth_user}


@template('students.jinja2')
async def students(request: Request):
    app: Application = request.app
    if request.method == 'POST':
        data = await request.post()
        async with app['db'].acquire() as conn:
            await Student.create(conn, data['name'])
    async with app['db'].acquire() as conn:
        students = await Student.get_many(conn)
    return {'students': students}
```

```
@template('student.jinja2')
async def student(request: Request):
    app: Application = request.app
    student_id = int(request.match_info['id'])
    async with app['db'].acquire() as conn:
        student = await Student.get(conn, student_id)
        if not student:
            raise HTTPNotFound()
        marks = await Mark.get_for_student(conn, student_id)
        courses = await Course.get_many(conn)
    courses_marks = {c: list(ms) for c, ms
                     in groupby(marks, lambda m: m.course_id)}
    results = [
        (course, courses_marks.get(course.id))
        for course in courses
        if course.id in courses_marks
    ]
    return {'student': student, 'results': results}


@template('courses.jinja2')
async def courses(request: Request):
    app: Application = request.app
    if request.method == 'POST':
        data = await request.post()
        async with app['db'].acquire() as conn:
            await Course.create(conn, data['title'],
                                data['description'])
    async with app['db'].acquire() as conn:
        courses = await Course.get_many(conn)
```

```
    return {'courses': courses}


@template('course.jinja2')
async def course(request: Request):
    app: Application = request.app
    course_id = int(request.match_info['id'])
    async with app['db'].acquire() as conn:
        course = await Course.get(conn, course_id)
        if not course:
            raise HTTPNotFound()
        reviews = await Review.get_for_course(conn, course_id)
        students = await Student.get_many(conn)
    return {'course': course,
            'reviews': reviews,
            'students': students}


@template('review.jinja2')
async def review(request: Request):
    app: Application = request.app
    course_id = int(request.match_info['course_id'])
    async with app['db'].acquire() as conn:
        course = await Course.get(conn, course_id)
        if not course:
            raise HTTPNotFound()
        if request.method == 'POST':
            data = await request.post()
            review_text = data.get('review_text')
            if not review_text:
                return {
                    'course': course,
```

```
                'errors': {
                    'review_text': 'this is required field',
                },
            }
            await Review.create(conn, course_id, review_text)
            raise HTTPFound(f'/courses/{course_id}')
        return {'course': course, 'errors': {}}

@template('evaluate.jinja2')
async def evaluate(request: Request):
    app: Application = request.app
    student_id = int(request.match_info['student_id'])
    course_id = int(request.match_info['course_id'])
    data = await request.post()
    async with app['db'].acquire() as conn:
        student = await Student.get(conn, student_id)
        course = await Course.get(conn, course_id)
        if not student or not course:
            raise HTTPNotFound()
        try:
            data = EVALUATE_SCHEMA.check_and_return(data)
        except DataError as e:
            return {'errors': e.as_dict(),
                    'course': course,
                    'student': student}
        await Mark.create(conn, student_id, course_id,
                          data['points'])
    raise HTTPFound(f'/courses/{course_id}')

@authorize()
```

```
async def logout(request: Request):
    session = await get_session(request)
    session.pop('user_id', None)
    raise HTTPFound('/')
```