ОСОБЛИВОСТІ УСУНЕННЯ ВРАЗЛИВОСТЕЙ REST API ДО АТАК XSS ТА SQL- ІН'ЄКЦІЙ

Горяінова К.О., Капуста Р.Д.

Харківський національний університет радіоелектроніки,
кафедра інфокомунікаційної інженерії ім. В.В. Поповського,
Україна

E-mail: karyna.horiainova@nure.ua,
roman.kapusta@nure.ua

Abstract

The main purpose of this paper is to study methods of eliminating REST API vulnerabilities aimed at protecting against XSS and SQL injection attacks. In particular, we will consider the basic principles of protection, such as input data validation, parameterized queries, output data encoding, the use of security policies, as well as other approaches to protecting REST APIs from these threats.

З розвитком сучасних веб-технологій інтерфейси прикладного програмування (API) стали основою для зв'язку між різними програмними компонентами, і можуть працювати як на стороні сервера, так і на стороні клієнта. REST API (Representational State Transfer Application Programming Interface) – це архітектурний стиль для взаємодії програмних компонентів через мережу, зазвичай у форматі HTTP [1].

Завдяки зростаючій ролі REST API, у забезпеченні зв'язку між різними системами, виникає нагальна потреба в безпеці таких інтерфейсів. Оскільки REST API працює у відкритому середовищі, він потенційно вразливий до багатьох атак, зокрема міжсайтового скриптингу (XSS) та SQL-ін'єкцій. Вразливості API можуть призвести до витоку конфіденційних даних, втрати контролю над користувацькими акаунтами, порушення функціонування сервісу, або навіть компрометації всієї системи. Тому забезпечення безпеки API стає критично важливим завданням у розробці сучасного програмного забезпечення [2].

Для дослідження безпеки API було розроблено спеціальне віртуально-тренувальне средовище із двома вразливостями, а саме SQL-ін'єкції та міжсайтового скриптингу (XSS). Першим кроком продемонструємо роботу розробленого тренувального серодовища, який приведений на рисунку 1. Як ми бачимо, це повноцінно працюючий API, який має дві вразливі сторінки. Ці сторінки мають свій особистий файл доступ до якого можна отримати через відповідні посилання на цей файл, а також свій порт. На головній сторінці тренувального середовища ми маємо відслідкувати їх працездатність та логи, які вони видають. Наші сторінки для експерименту мають відповідну розмітку та архітектурну ієрархію в залежності від пріоритету виконання.

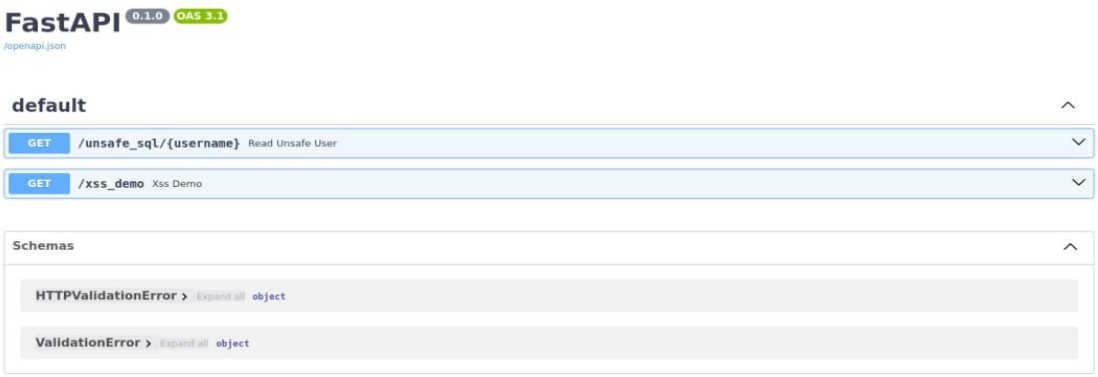

Рис. 1. Головна сторінка API

SQL-ін'єкція – це вид атаки, при якому зловмисник вводить SQL-код у запит, що відправляється до бази даних. Якщо сервер неправильно обробляє введені дані, це дозволяє зловмиснику отримати несанкціонований доступ до даних, модифікувати або видалити їх. SQL-ін'єкція особливо небезпечна для REST API, оскільки API часто працює з даними безпосередньо з бази, і вразливості у запитах можуть призвести до серйозних порушень конфіденційності та цілісності інформації [3].

Наприклад, якщо REST API приймає параметр для пошуку користувача за іменем і будує SQL-запит на основі цього параметра без належної обробки, зловмисник може передати рядок, що містить SQL-команди. Це дозволить йому змінити запит і, наприклад, отримати доступ до конфіденційних даних, видалити таблиці або змінити інформацію у базі. У результаті така уразливість може призвести до витоку даних, їхнього знищення чи навіть повного компрометування бази даних.

Для проведення експеритменту ми проводимо атаку на нашу базу даних. Для цього зловмисник використовує пошук користувача за стандартним логіном, а саме «testuser». Результат проведення атаки на нашу базу даних приведено на рисунку 2.

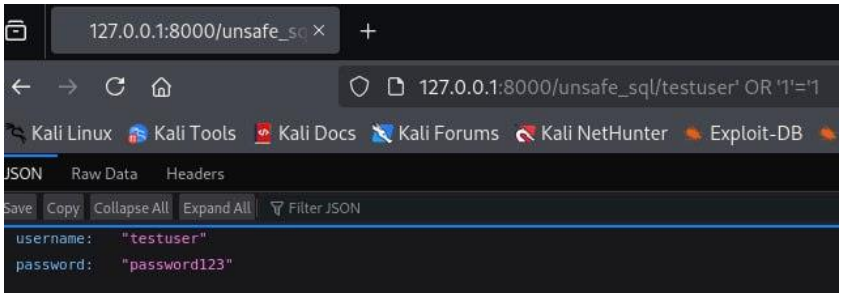

Рис. 2. Демонстрація інекції та її роботи до виправлення

Як ми бачимо, потенційний зловмисник може вкористати цю просту SQL-ін'єкці для отримання логіну та паролю від облікового запису користувача. Що свідчить про те, що дана сторінка є вразливою до подібного роду атаки. Для вирішення цієї проблеми було розроблено та використано спеціальний скрипт, мова про який буде йти далі.

XSS (Cross-Site Scripting) – це вид атаки, при якому зловмисник вводить шкідливий скрипт у веб-сторінку або додаток, що потім виконується у браузері інших користувачів. Вразливість XSS небезпечна для REST API, оскільки може призвести до викрадення даних користувачів, перенаправлення на зловмисні сайти, а також модифікації вмісту сторінок [4].

Наприклад, уявімо REST API, який приймає та зберігає коментарі користувачів, а потім повертає їх для відображення іншим користувачам на сторінці. Якщо API не здійснює належного очищення або кодування введених даних, зловмисник може впровадити шкідливий JavaScript у свій коментар. Коли інші користувачі завантажують сторінку з коментарями, цей скрипт виконується у їхніх браузерах, дозволяючи зловмиснику викрасти дані сесії, перенаправити користувачів на шкідливі сайти або змінити вміст сторінки.

Для проведення експеритменту ми проводимо атаку на нашу веб-сторінку. Для цього зловмисник використовує атаку міжсайтового скриптингу (XSS), що дозволяє йому отримати доступ до прихованої веб-сторінки, а також запросити відповідну відповідь на цей запит. Результат проведення атаки на нашу базу даних приведено на рисунку 3.

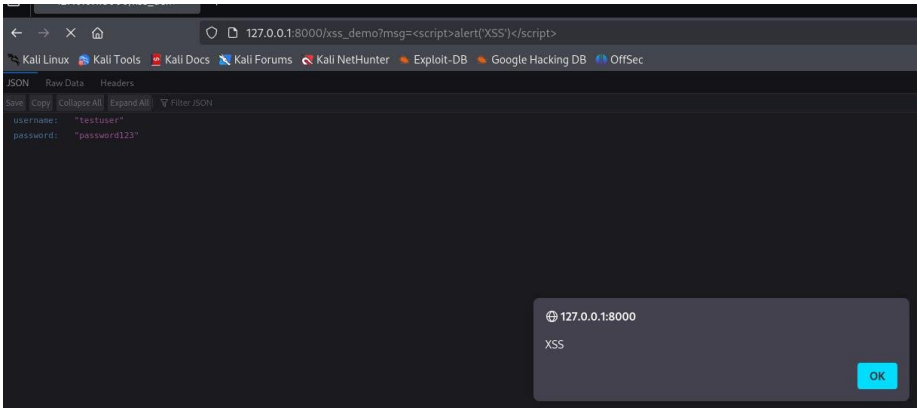

Рис. 3. Демонстрація XSS до виправлення

Для вирішення та усунення розглянутих вразливостей було запропоновано скрипт, представлений на рисунку 4. Цей скрипт автоматично сканує головний виконавчий файл API, визначає наявність уразливостей у коді, таких як SQL-ін'єкції та міжсайтовий скриптинг (XSS), і автоматично виправляє їх. Зокрема, скрипт замінює небезпечні SQL-запити на параметризовані та додає обробку HTML-коду для запобігання XSS.

На рисунку 5 показано відпрацювання скрипта, який виконує перевірку та фіксацію вразливостей у файлі `main.py`. У результаті роботи скрипта кожна виявлена вразливість позначається як виправлена, що підтверджується відповідними повідомленнями. Завдяки цьому автоматизованому підходу вдалося значно підвищити рівень безпеки API, мінімізуючи ризики SQL-ін'єкцій та XSS-атак.

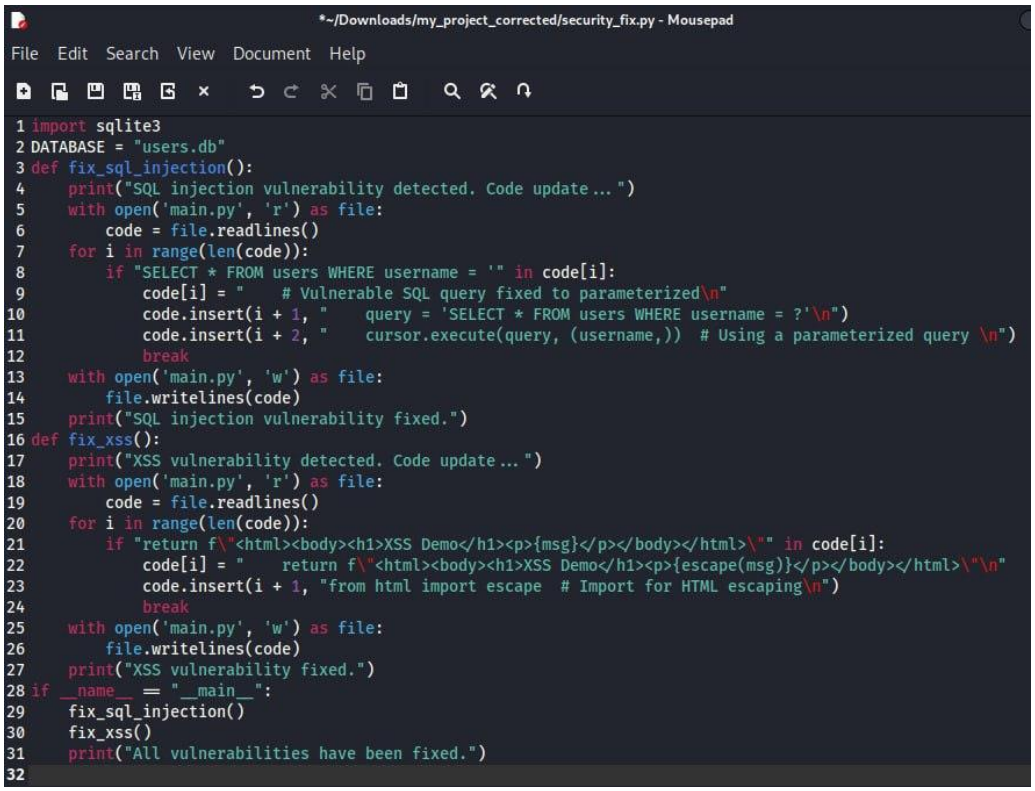

Рис. 4. Скрипт для усунення вразливостей коду

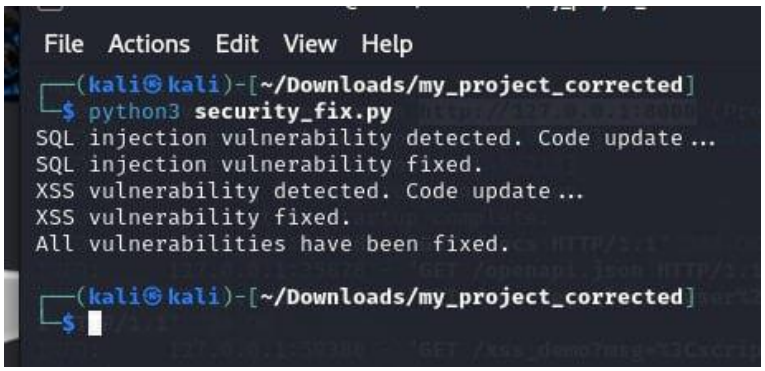

Рис. 5. Відпрацьовування скрипту захисту

Після того, як скрипт завершив свою роботу, ми повторили спроби провести атаки на базу даних через SQL-ін'єкцію та на веб-сторінку через XSS. Як показано на рисунку 6, при спробі SQL-ін'єкції сервер видає помилку "Internal Server Error", що свідчить про відсутність доступу до бази даних для зловмисника. Аналогічно, на рисунку 7 продемонстровано спробу XSS-атаки, де введений скрипт не виконався, а замість цього відобразився як текст. Це підтверджує, що зловмисник більше не може впровадити шкідливий код на веб-сторінку або отримати доступ до прихованих даних.

Таким чином, впроваджені захисні заходи успішно запобігли обом видам атак і забезпечили безпеку системи.

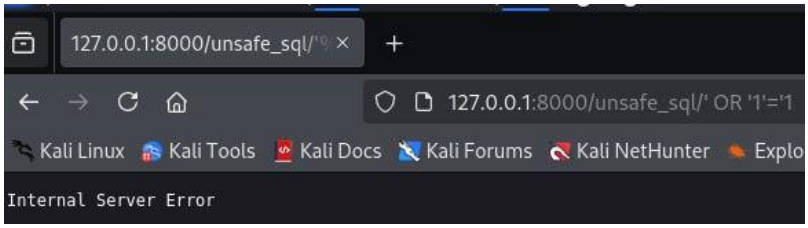

Рис. 6. Результат виправлення SQL

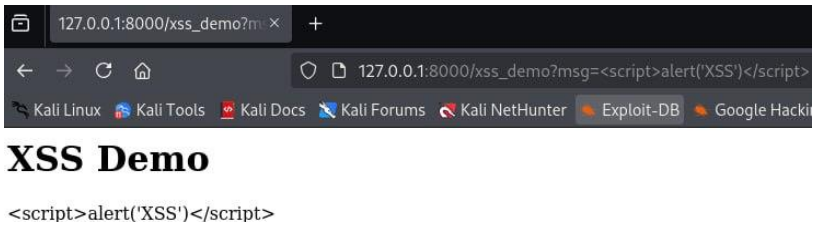

Рис. 7. Результат виправлення XSS

Результати дослідження демонструють важливість реалізації заходів безпеки для REST API на етапі розробки та підтримки програмного забезпечення. Правильна обробка вхідних даних та автоматизовані методи захисту є важливими для запобігання потенційним атакам і забезпечення надійного захисту даних користувачів.

Для забезпечення безпеки API важливо використовувати параметризовані запити для запобігання SQL-ін'єкціям, кодувати введення користувачів, щоб уникнути XSS-атак, регулярно перевіряти систему на вразливості за допомогою автоматизованих сканерів, налаштувати моніторинг та журналювання для фіксації аномальних дій, а також своєчасно оновлювати систему та бібліотеки для захисту від нових загроз і вразливостей [5].

Література

1. Astera Software. RESTф API: Definition, Types, and Examples. Astera. 2022. URL: https://www.astera.com/type/blog/rest-api-definition/.

2. Fastly. What is API Security? Fastly. 2023. URL: https://www.fastly.com/learning/what-is-api-security.

3. PortSwigger. SQL Injection. PortSwigger Web Security Academy. 2022. URL: https://portswigger.net/web-security/sql-injection.

4. PortSwigger. Cross-Site Scripting (XSS). PortSwigger Web Security Academy. 2022. URL: https://portswigger.net/web-security/cross-site-scripting.

5. ByteByteGo. API Security Best Practices. ByteByteGo Blog. 2022. URL: https://blog.bytebytego.com/p/api-security-best-practices