УДК 004.49

ДОСЛІДЖЕННЯ ТА РОЗРОБКА МЕТОДИКИ АВТОМАТИЗАЦІЇ ПОШУКУ ВРАЗЛИВОСТЕЙ SQL INJECTION У ВЕБ-ДОДАТКУ

Вознюк Дмитро Вівторович
к.т.н, доцент каф. ІС, Червоненко Петро Петрович
Національний університет «Одеська політехніка», Україна

АНОТАЦІЯ. Проаналізовано можливості тестування веб-додатків Black Box та показано що *SQL Injection* є найпоширенішим вразливим місцем**.** Наведено приклад ідентифікації *SQL* вразливості для веб-вітрини. Запропоновано методику автоматизації пошуку *SQL Injection* вразливості у веб-додатку, яка складається із модулів *Crawler, Attack, Analysis* та *Report.* Розглянуто можливості що до удосконалення пошуку графа в ширину для модуля Crawling (парсинг GET, POST запитів та їх параметрів) , що посприяло зменшенню часу сканування.

Вступ. Зростаюча кількість корпоративних веб-додатків обробляє конфіденційні фінансові та медичні дані, які в разі зламу можуть коштувати мільйони доларів. Надзвичайно важливо захистити ці програми від хакерів. Згідно з опитуванням *OWASP, SQL Injection* є найпоширенішим вразливим місцем у веб-додатках [1]. Якщо веб-додаток має вразливості, то введені користувачем дані можуть бути використані для кадрування виписок SQL які потім виконуються додатком у базі даних. Наслідки таких дій можуть бути тривожними.

Загальний підхід до тестування безпеки веб-додатків полягає у використанні *Black Box* сканерів. Це інструменти, які сканують веб-додаток, щоб перерахувати всі доступні *GET* та *POST* запити, генерувати спеціально створені вхідні значення, які надсилаються до сервера веб-додатку, і спостерігати за поведінкою програми, щоб визначити, чи була запущена вразливість. Якщо веб-додаток має більше мільйону сторінок, то час сканування розтягується на тижні. Тому зменшення часу сканування є обов'язковою мірою для вдосконалення підходу *Black Box.*

Тестування методом «*Black Box*» - це стратегія, в якій тестування засноване виключно на вимогах і специфікаціях, при цьому не знаючи, як влаштована всередині система, що тестується, і працювати виключно із зовнішніми інтерфейсами тестованої системи або компонента. Тестування чорної скриньки може бути застосоване на всіх рівнях - модульному, інтеграційному, системному та приймальному [2].

Мета роботи. Дослідження та розробка методики автоматизації пошуку вразливостей *SQL Injection* для зменшення часу сканування у веб-додатку.

Основна частина роботи. Атака *SQL Injection* полягає у вставці або ін'єкції *SQL*-запиту через вхідні дані від клієнта до веб сервера додатку. Успішний експлойт ін'єкції *SQL* може читати конфіденційні дані з бази даних, змінювати дані бази даних, виконати операції адміністрування бази даних.

Для ідентифікації *SQL* вразливості у веб-додатку розглянемо торгову програму, яка демонструє товари з різних категорій. Коли користувач вибирає підкатегорію «*Watch*», його браузер запитує URL-адресу: «*https://site.com/products?category=Watch».* Це змушує програму виконувати SQL запит для отримання відомостей про відповідні продукти з бази даних: «*SELECT * FROM products WHERE category = 'Watch' AND released = 1»* . Якщо веб-додаток, який не реалізує жодного захисту від атак *SQL Injection*, тому зловмисник може створити атаку так: «*https://site.com/products?category=Watch' +OR+1=1--»* . Це призводить до наступного SQL-запиту: «*SELECT * FROM products WHERE category = 'Watch' OR 1=1--' AND released = 1».* Символ «--» змушує закоментувати решту запиту, змінений запит поверне усі елементи, у яких або категорія «*Watch*», або 1 дорівнює 1. Оскільки 1=1 завжди вірно, запит поверне всі елементи.

Методика автоматизації пошуку вразливостей, яка показана на рисунку 1, складається з чотирьох модулів:

Crawler. Спочатку потрібно отримати всі існуючі *POST* та *GET* запити в веб-додатку. *Crawler* насправді є алгоритмом пошуку графа в ширину. Можна вважати інтернет великим

графом зі сторінками в якості вершин, а його запити та гіперпосилання як його ребра. *Crawler* запускається з кількох початкових вузлів, а потім слідує за ребрами для прибуття інших вузлів. Процес вилучення сторінок та посилань усередині нього аналогічний розширенню вузла у графічному пошуку. Найкраще для вилучення POST та GET запитів підходить метод пошук у ширину. Пошук у ширину обходить вершини графа в порядку віддаленості початкової сторінки. Для цього алгоритм використовує структуру даних "черга" - в черзі можна додавати елементи в кінці і витягувати з початку. [3]

Attack. Після отримання списку всіх запитів та їх параметрів, проводиться атака відправляючи ін'єкції на параметри методом перебору шаблонних атак із бази даних.

Analysis. Після відправки ін'єкції, потрібно визначити чи проходить дана ін'єкція. Для ідентифікації SQL-ін'єкцій підхід зіставлення шаблонів помилок полягає в надсиланні спеціально створених запитів до програми та пошуку конкретних шаблонів у відповідях, наприклад, повідомлення про помилки бази даних. Основна ідея полягає в тому, що наявність повідомлення про помилку SQL на сторінці відповіді HTML означає, що відповідний запит не був оброблений програмою. Таким чином, той факт, що цей запит було надіслано на сервер SQL без змін, свідчить про наявність уразливості.[4]

Report. Якщо вразливість знайдена, то виводиться інформація про неї.

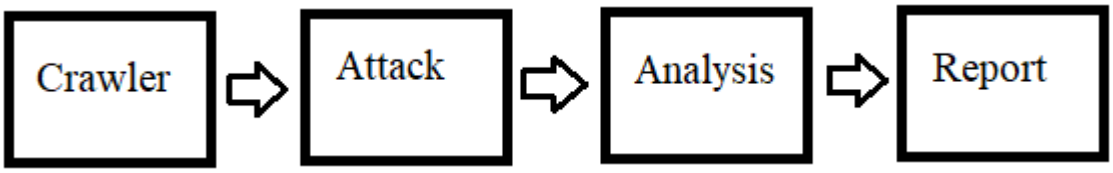

Рисунок 1 – Методика автоматизації пошуку вразливостей

Для збільшення швидкості сканування вразливостей, потрібно адаптувати метод пошуку в ширину для модуля *Crawler*. Щоб зменшити час, головним завданням є різноманітність функцій, а не схожість вмісту. Це пояснюється тим, що різні функціональні можливості зазвичай відображаються в різних сегментах коду цілі. Наприклад, якщо збирається посилання «*https://site.com/order.php?user=lila*», збирається подібне посилання «*https://site.com/order».php?user=jack*» також можна знайти. Оскільки функціональне призначення цих двох веб-сторінок ідентичне, вони використовують той самий сегмент коду. Якщо сканер виділяє ресурси для аналізу цієї нової веб-сторінки, це буде марною витратою. Тому, слід зосередитися на зборі якомога більшої кількості різних точок входу, а не на відстежуванні всіх посилань.

У методі пошуку в ширину потрібно після кожної ітерації, перед додаванням до стеку, перевіряти чи нема схожих за функціоналом посилань, якщо є то потрібно видаляти їх, щоб на наступній ітерації не проходити це посилання. Це заощадить багато часу – не потрібно парсити GET та POST запити, якщо уже на подібній по функціоналу посилання вони э. Також це заощадить час на подальшому тестуванні.

Формула 1 відображає час сканування та пошуку вразливостей.

$$t = N_r \times t_c \times t_a, \quad (1)$$

де N_r , t_c , t_a – відповідно кількість знайдених запитів, середній час парсингу однієї сторінки, середній час атаки на один запит.

Було знайдено веб-додаток «Форум автомобілістів» в якому 91% сторінок містять один функціонал – сторінка теми, сторінка користувача і так далі. При аналізі були отриманні данні, які занесено до таблиці 1.

Таблиця 1 – Дані зібрані при аналізі

	Кількість запитів	t_c	t_a	Час сканування
Оригінальний метод	3120	3.2с	5.1с	14.15 годин
Адаптований метод	280	3.2с	5.1с	1.27 годин

Судячи з таблиці 1, можна сказати, що метод пошуку в ширину для модуля Crawler, було адаптовано ефективно, час сканування скоротився на 90%, але якщо сканувати веб-додаток, де всі запити та сторінки с з параметрами є унікальними – час не буде скорочений. Тобто, можна сказати, що скорочення часу буде змінюватись, в залежності від веб-додатку, приблизно від 0 до 90%.

Висновки. В результаті роботи показані можливості щодо автоматизованого пошуку вразливостей *SQL Injection* у веб-додатках. Запропоновано модифікацію метода пошука графа в ширину, який реалізований в модулі *Crawler*. Це вже дало суттєве зменшення часу сканування такого веб-дотатку як «Форум автомобілістів». Для майбутньої роботи можна покращити модуль *Attack* та *Analysis*. В модулі *Analysis* є свої недоліки, так як цей метод не виявляє *SQL Injection*, які не виводять помилок.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. OWASP Top 10 – 2021. URL: https://owasp.org/Top10/ (дата звернення : 01.05.2023)
2. Тестування методом чорної скриньки (Black Box Testing). URL: https://vladislaveremeev.gitbook.io/qa_bible/vidy-metody-urovni-testirovaniya/testirovanie-metodom-chernogo-yashika-black-box-testing (дата звернення : 01.05.2023)
3. Nguyen Thanh Viet1 , Alla G. Kravets THE ALGORITHM OF WEB CRAWLER TO SOLVE THE PROBLEM OF COLLECTING DATA FROM OPEN INTERNET SOURCES, Published by St Petersburg State Institute of Technology (Technical University) – 2019.
4. Rim Akrout, Eric Alata, Mohamed Kaaniche. An automated black box approach for web vulnerability identification and attack scenario generation, // Journal of the Brazilian Computer Society – 2014.

RESEARCH AND DEVELOPMENT OF THE METHOD OF AUTOMATING THE SEARCH OF SQL INJECTION VULNERABILITIES IN A WEB APPLICATION

Dmytro Vozniuk

PhD, Associate Professor of IS department, Petr Chervonenko

Odesa Polytechnic National University, UKRAINE

ANNOTATION. Black Box web application testing capabilities were analyzed and SQL Injection was shown to be the most common vulnerability. An example of SQL vulnerability identification for a web store is provided. A technique for automating the search for SQL Injection vulnerabilities in a web application is proposed, which consists of Crawler, Attack, Analysis and Report modules. Considered the possibilities of improving the width graph search for the Crawling module (parsing GET, POST requests and their parameters), which contributed to reducing the scanning time.