DOI: 10.64302/joshusc.v31n1e.1341

KHAI THÁC LỖ HỔNG BẢO MẬT SQL INJECTION TRONG ỨNG DỤNG WEB

Phan Văn Cường, Hồ Đức Tâm Linh, Vương Quang Phước*

Khoa Điện, Điện tử và Công nghệ vật liệu, Trường Đại học Khoa học, Đại học Huế

*Email: vqphuoc@husc.edu.vn

Ngày nhận bài: 30/9/2025; ngày hoàn thành phản biện: 8/10/2025; ngày duyệt đăng: 16/10/2025

TÓM TẮT

SQL Injection (SQLi) là kỹ thuật tấn công phổ biến trong bảo mật ứng dụng web, đây là kỹ thuật cho phép người tấn công xâm nhập, truy xuất cơ sở dữ liệu bằng cách khai thác lỗ hổng đầu vào không được xử lý đúng cách. Kỹ thuật tấn công này được sử dụng để đánh cắp dữ liệu hoặc thậm chí chiếm quyền kiểm soát hệ thống. Việc thiếu triển khai các biện pháp bảo mật trong quá trình phát triển ứng dụng web là nguyên nhân chính dẫn đến sự phổ biến của SQLi. Bài báo thực hiện phân tích và đánh giá các kỹ thuật tấn công SQL Injection thường gặp gồm: Blind SQLi, Union-based SQLi và Error-based SQLi. Từ đó, đề xuất một số giải pháp phòng chống để xây dựng hệ thống an toàn hơn như: Sử dụng truy vấn tham số hóa và quyền tối thiểu, triển khai tường lửa ứng dụng web (WAF) hay mã hóa bảo mật.

Từ khóa: SQL Injection, Blind SQLi, Error-based SQLi, Union-based SQLi, WAF.

1. GIỚI THIỆU

Hiện nay, theo OWASP (Open Web Application Security Project) - tổ chức quốc tế hàng đầu chuyên về việc bảo mật ứng dụng web, SQL Injection (hay SQLi) liên tục được xếp vào top 10 lỗ hổng bảo mật nguy hiểm nhất được ghi nhận trong những năm gần đây, với hàng nghìn vụ vi phạm dữ liệu nghiêm trọng trên toàn cầu [1]. Bên cạnh đó, theo thống kê từ các báo cáo về an ninh mạng, hơn 50% các cuộc tấn công vào hệ thống cơ sở dữ liệu bắt nguồn từ SQLi [2].

SQLi là kỹ thuật tấn công nhằm mục đích khai thác lỗ hổng bảo mật trong các ứng dụng web cho phép kẻ tấn công gửi các truy vấn SQL (Structured Query Language) không hợp lệ hoặc độc hại đến cơ sở dữ liệu. Những truy vấn này có thể bị thay đổi để bỏ qua xác thực truy vấn đầu vào, dẫn đến rò rỉ dữ liệu hoặc bị mất quyền kiểm soát hệ thống. SQLi thường xảy ra khi ứng dụng web xử lý các tham số đầu vào từ người dùng

không đúng cách, từ đó dẫn đến việc thực thi các câu lệnh SQL độc hại. Lợi dụng điều đó, các kẻ tấn công có thể gây ra các rủi ro như sau:

Mất các thông tin bảo mật: Người tấn công có thể truy cập vào các thông tin quan trọng như mật khẩu, thông tin cá nhân, hoặc dữ liệu tài chính gây rò rỉ thông tin và ảnh hưởng đến bảo mật của các cá nhân hoặc tổ chức.

Cơ sở dữ liệu chứa mã độc: Người tấn công có thể chèn mã độc (ví dụ như các đoạn mã JavaScript) vào cơ sở dữ liệu khiến ứng dụng web thực thi mã độc khi dữ liệu được truy xuất, từ đó có thể ảnh hưởng đến người dùng khác hoặc hệ thống.

Mất mát hoặc sai lệch dữ liệu: Một trong những tác động nghiêm trọng của SQLi là khả năng thay đổi hoặc xóa dữ liệu quan trọng trong cơ sở dữ liệu. Điều này dẫn đến việc mất mát dữ liệu hoặc làm gián đoạn hoạt động của hệ thống.

Mất quyền kiểm soát hệ thống: Khi SQL Injection kết hợp với Remote Code Execution (RCE), người tấn công có thể chiếm quyền điều khiển máy chủ hoàn toàn, cài đặt thêm các phần mềm độc hại hoặc thậm chí thực hiện tấn công các hệ thống khác trong mạng nội bộ.

Trong bài báo này, chúng tôi sẽ tập trung vào việc phân tích chi tiết các kỹ thuật tấn công SQL Injection nhằm làm sáng tỏ cách thức hoạt động và đánh giá mức độ rủi ro của chúng. Theo đó, chúng tôi trình bày tổng quan về kỹ thuật tấn công SQL Injection và đưa ra các phân tích chi tiết về các kỹ thuật tấn công SQLi phổ biến, bao gồm: Blind SQLi, Error-based SQLi và Union-based SQLi. Đồng thời, chúng tôi đề xuất một số giải pháp bảo mật hỗ trợ bảo vệ cơ sở dữ liệu trong các ứng dụng web như: triển khai các kỹ thuật truy vấn tham số hóa, tường lửa ứng dụng web (WAF) hay mã hóa bảo mật. Quá trình khảo sát và đánh giá các trường hợp tấn công cụ thể được thực hiện dựa trên môi trường của TryHackMe.

2. CÁC HÌNH THỨC TẤN CÔNG SQL INJECTION

Như đã đề cập ở phần 1, kỹ thuật SQL Injection (SQLi) là một trong những hình thức tấn công nguy hiểm và phổ biến nhất, tận dụng các lỗ hổng bảo mật trong ứng dụng web để thực thi các truy vấn SQL độc hại. Dù chỉ là những chuỗi lệnh đơn giản nhưng mỗi truy vấn này lại có thể gây ảnh hưởng đến toàn bộ hệ thống, từ việc đánh cắp dữ liệu bảo mật cho đến chiếm quyền điều khiển hệ thống. Việc hiểu được phương thức hoạt động và các loại hình tấn công SQLi không chỉ giúp người đọc thấy được mức độ nguy hiểm của nó mà còn đóng vai trò quan trọng trong việc xây dựng các giải pháp bảo mật hiệu quả hơn.

Trong nội dung này, chúng tôi sẽ phân loại và phân tích về các hình thức tấn công SQL Injection, giúp người đọc có thể hiểu rõ hơn những nguy cơ tiềm ẩn từ kỹ thuật tấn công này. Mỗi kỹ thuật tấn công có một cách tiếp cận khác nhau, tuy nhiên có thể phân SQLi thành 03 dạng phổ biến sau:

Đầu tiên, là kỹ thuật Blind SQL Injection (Boolean-Based, Time-Based), xảy ra trong các trường hợp mà hệ thống không hiển thị lỗi rõ ràng hoặc không cung cấp thông báo chi tiết về trạng thái truy vấn SQL. Tuy nhiên, người tấn công vẫn có thể lợi dụng các phản hồi gián tiếp từ hệ thống để thu thập thêm thông tin và xác định lỗ hổng. Cụ thể, sử dụng kỹ thuật Boolean-based để xác định dữ liệu thông qua sự khác biệt về kết quả trong các truy vấn SQL có điều kiện (true/false) hoặc sử dụng kỹ thuật Time-based xác định thông tin dựa vào khoảng thời gian mà hệ thống mất để trả lời truy vấn thông qua việc sử dụng các lệnh làm chậm thời gian phản hồi của hệ thống.

Kế đến, kỹ thuật Union-Based SQL Injection có thể được xem là kỹ thuật nhanh và hiệu quả nhất và nếu không có biện pháp bảo mật, dữ liệu có thể bị rò rỉ mà không để lại dấu hiệu rõ ràng. Kỹ thuật SQLi này cho phép người tấn công sử dụng câu lệnh UNION để kết hợp kết quả của hai hoặc nhiều truy vấn SELECT thành một tập kết quả duy nhất, loại bỏ các bản ghi trùng lặp từ đó lấy được thông tin của cơ sở dữ liệu.

Cuối cùng là kỹ thuật Error-Based SQL Injection. Kỹ thuật tấn công này thực hiện khai thác dựa trên các thông báo lỗi từ hệ thống trả về khi gặp các truy vấn SQL không hợp lệ. Thông thường, những thông báo này thường chứa thông tin về cấu trúc cơ sở dữ liệu như: tên các bảng và các trường dữ liệu; kiểu dữ liệu của từng trường hay thậm chí cách thức tổ chức /liên kết trong cơ sở dữ liệu. Từ đó, giúp người tấn công dần dần thu thập được thông tin đầy đủ về hệ thống cơ sở dữ liệu và có thể đưa ra các kế hoạch tấn công sâu hơn vào hệ thống.

3. PHÂN TÍCH VÀ KHAI THÁC SQL INJECTION

3.1. Công cụ và môi trường thử nghiệm

Bài báo này tập trung tìm hiểu, nghiên cứu và thử nghiệm về kỹ thuật khai thác lỗ hổng SQL Injection. Trong các quá trình trên, việc thiết lập một môi trường thử nghiệm là điều kiện quan trọng nhằm đảm bảo có thể tạo ra được một môi trường cho phép người dùng mô phỏng tấn công trong khi vẫn đảm bảo tính an toàn về mặt bảo mật cho hệ thống đang sử dụng. Hiện tại, cộng đồng an ninh mạng trên thế giới có nhiều môi trường thử nghiệm từ cơ bản đến nâng cao, ví dụ: TryHackMe, Hackthebox, HackmyVM ... Trong đó, TryHackMe là nền tảng phục vụ cho mọi đối tượng người dùng từ những người mới bắt đầu cho đến các chuyên gia trong ngành. TryHackMe hỗ

trợ người học qua các bài thực hành, điều này giúp họ cải thiện kỹ năng khai thác và phòng thủ lỗ hổng bảo mật trong một môi trường an toàn. Ngoài ra, điểm mạnh của TryHackMe là sự hỗ trợ từ cộng đồng người dùng rộng lớn và các chuyên gia trong lĩnh vực an ninh mạng.

Bên cạnh môi trường mô phỏng thực nghiệm, chúng ta còn có thể sử dụng các công cụ phổ biến khác để khai thác SQL Injection, bao gồm các bộ công cụ phổ biến như: SQLmap [3], Burp Suite[4].

3.2. Quy trình khai thác

Sau khi thực hiện triển khai môi trường và các bộ công cụ, có thể thực hiện mô phỏng và đánh giá các cuộc tấn công SQL Injection để có thể hiểu thêm về kỹ thuật tấn công trên. Và về cơ bản, việc khai thác SQL Injection có thể được thực hiện theo quy trình được trình bày trong bảng 1.

Bảng 1. Quy trình khai thác lỗ hổng SQL Injection

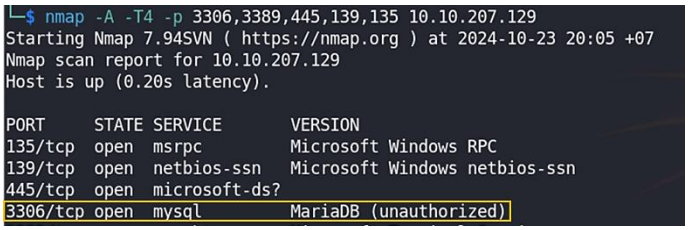
Giai đoạn i. Tìm kiếm điểm đầu vào không được kiểm tra hoặc không được xử lý đúng cách
Bước đầu tiên trong quá trình khai thác SQL Injection là xác định các điểm đầu vào trong ứng dụng web chẳng hạn như: các biểu mẫu nhập liệu, URL hoặc tham số GET/POST không được kiểm tra hoặc không được xử lý an toàn.
Giai đoạn ii. Chèn payload (trọng tải) để kiểm tra các yếu điểm
Sau khi xác định được các điểm đầu vào không an toàn hoặc độc hại, chèn vào các tham số và các payload SQL để kiểm tra khả năng bị tấn công của hệ thống. Các payload này gồm các câu lệnh SQL độc hại nhằm xác định các yếu điểm của hệ thống.
Giai đoạn iii. Phân tích phản hồi từ máy chủ
Sau khi chèn payload, cần phân tích phản hồi từ máy chủ để xác định ứng dụng hoặc hệ thống có bị ảnh hưởng bởi lỗ hổng SQL Injection hay không. Các phản hồi từ máy chủ đưa ra các thông tin về cấu trúc cơ sở dữ liệu, thông báo lỗi SQL và nhận dạng lỗ hổng SQL Injection.

Các quy trình khai thác trên giúp xác định và khai thác các lỗ hổng bảo mật trong hệ thống cơ sở dữ liệu từ đó truy xuất các dữ liệu bảo mật, chèn mã độc hoặc tấn công vào hệ thống.

3.3. Minh họa thực nghiệm

Để minh họa quá trình khai thác SQL Injection, nghiên cứu này khai thác nội dung: Advanced SQL Injection của nền tảng TryHackMe. Mô phỏng này cung cấp một môi trường thực tế cho người học về các lỗ hổng SQL Injection.

Đầu tiên, thông qua sử dụng Nmap có thể giúp phát hiện và kiểm tra được máy chủ/hệ thống có sử dụng dịch vụ cơ sở dữ liệu, chẳng hạn như MySQL hay MariaDB. Như trong trường hợp đang xét ở Hình 1, có thể nhận thấy hệ thống đang sử dụng MariaDB (port 3306, trạng thái Open) và do đó nó hoàn toàn có thể bị tấn công bằng kỹ thuật SQL Injection.



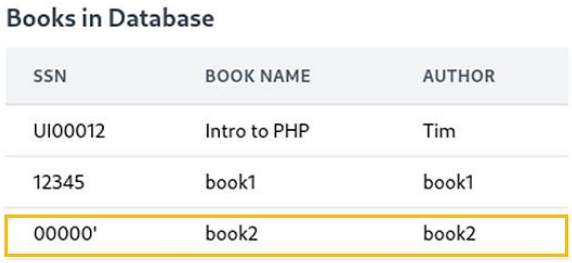
Hình 1. Xác định được cổng 3306 đang mở trên máy chủ mục tiêu bằng Nmap

Vì quá trình tấn công khá phức tạp và chi tiết, nên trong phạm vi bài báo này, chúng tôi xin được trình bày một cách súc tích và ngắn gọn về một số kiểu tấn công SQLi thường gặp trong các nội dung tiếp theo

3.3.1. Khai thác SQL Injection: Second-Order SQL Injection

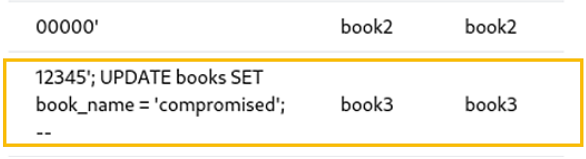
Bước 1: Tìm kiếm điểm đầu vào không được kiểm tra hoặc không được xử lý đúng cách.

Trong trường hợp này, khi chèn payload (payload: `00000'`) thì máy chủ/hệ thống vẫn phản hồi: "New book added successfully" cho thấy rằng đây có thể là điểm đầu vào không được kiểm tra hoặc không được xử lý đúng cách (Hình 2). Từ đây tiến hành chèn payload độc hại để khai thác dữ liệu có trong cơ sở dữ liệu của máy chủ/hệ thống.



Hình 2. Xác định điểm đầu vào không được kiểm tra hoặc không được xử lý đúng cách.

Bước 2: Chèn payload (trọng tải) để kiểm tra các yếu điểm



Hình 3. Chèn payload (trọng tải) để khai thác dữ liệu.

Tiến hành chèn payload độc hại để khai thác dữ liệu (payload: `12345' UPDATE books SET book_name='compromised';--`). Kết quả là dữ liệu đã bị khai thác thành công.

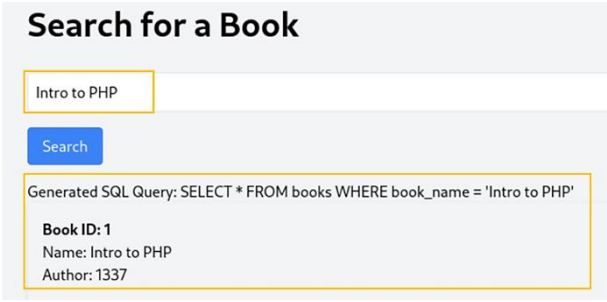
Bước 3: Khai thác và lấy dữ liệu

Hình 4. Dữ liệu đã bị khai thác thành công.

Các kết quả trên cho thấy, chúng ta đã khai thác được toàn bộ thông tin về cơ sở dữ liệu của thư viện sách, bao gồm các thông tin được ẩn giấu trong cơ sở dữ liệu.

3.3.2. Khai thác SQL Injection: Filter Evasion Techniques

Bước 1: Tìm kiếm điểm đầu vào không được kiểm tra hoặc không được xử lý đúng cách

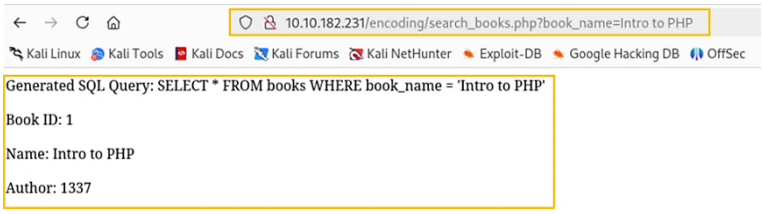


Hình 5. Xác định điểm đầu vào không được kiểm tra hoặc không được xử lý đúng cách.

Tìm kiếm tên sách ở ô tìm kiếm cho thấy phản hồi của máy chủ/hệ thống không chỉ đưa ra thông tin của quyển sách mà còn đưa ra câu lệnh truy xuất dữ liệu từ cơ sở dữ liệu. Từ đây có thể xác định điểm đầu vào không được kiểm tra hoặc không được xử lý đúng cách.

Bước 2: Chèn payload (trọng tải) để kiểm tra những yếu điểm để khai thác

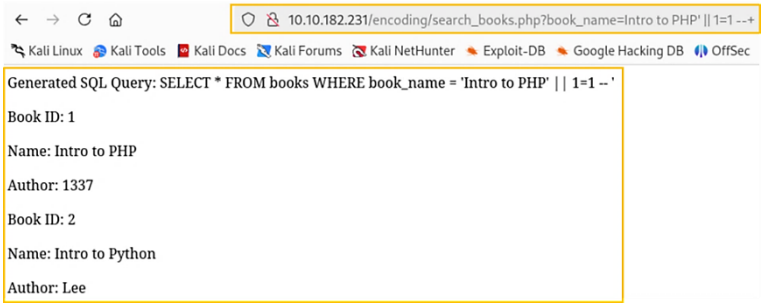
Tiến hành đưa payload (payload: `search_books.php?book_name=Intro to PHP`) lên địa chỉ truy cập web để dễ khai thác và chèn payload độc hại.



Hình 6. Chèn payload (trọng tải) để khai thác dữ liệu.

Bước 3: Khai thác và lấy dữ liệu

Chèn payload độc hại (payload: `search_books.php?book_name=Intro to PHP' || 1=1--+`) kết quả là toàn bộ dữ liệu đã trong cơ sở dữ liệu đã bị liệt kê đồng thời có thể thay đổi, xóa hoặc thêm dữ liệu vào cơ sở dữ liệu.

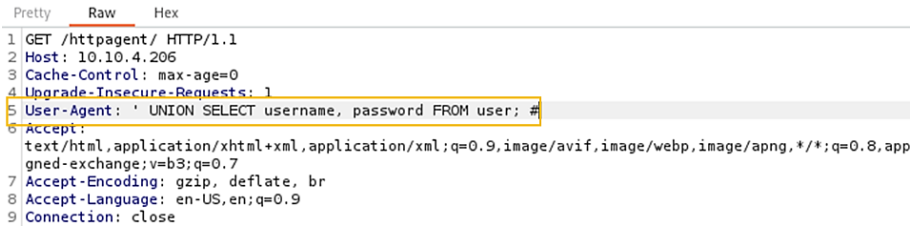


Hình 7. Dữ liệu đã bị khai thác thành công.

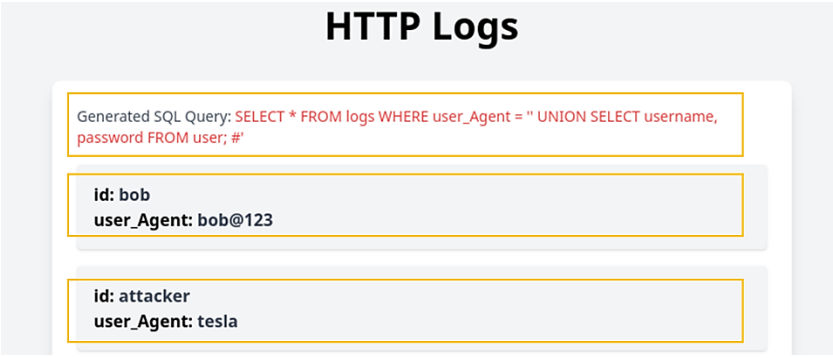
3.3.3. Khai thác SQL Injection: Other Techniques

Bước 1: Chèn payload (trọng tải) để kiểm tra yếu điểm của hệ thống.

Sử dụng phần mềm burpsuite, payload: `' UNION SELECT username, password FROM user; #` (payload này liệt kê tất cả tên người dùng và mật khẩu có trong cơ sở dữ liệu của máy chủ/hệ thống)



Hình 8. Chèn payload (trọng tải) để khai thác dữ liệu.

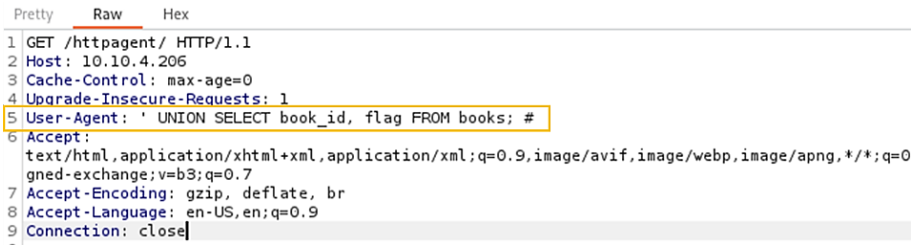


Hình 9. Kết quả phản hồi của máy chủ khi chèn payload độc hại

Phản hồi từ máy chủ/hệ thống cho thấy tất cả tên người dùng và mật khẩu được liệt kê đi kèm với đó là câu lệnh truy vấn dữ liệu SQL.

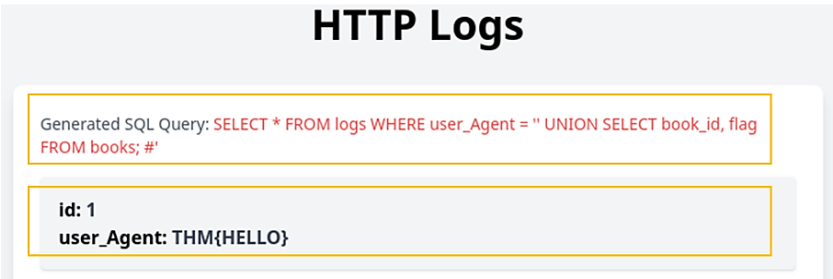
Bước 2: Khai thác và lấy dữ liệu

Tiến hành chèn payload độc hại: `' UNION SELECT book_id, flag FROM books; #` (payload này liệt kê tất cả sách và cờ được ẩn giấu trong cơ sở dữ liệu của hệ thống)



Hình 10. Chèn payload (trọng tải) để khai thác dữ liệu

Phản hồi từ máy chủ/hệ thống cho thấy tất cả tên người dùng và mật khẩu được liệt kê đi kèm với đó là câu lệnh truy vấn dữ liệu SQL.



Hình 11. Dữ liệu đã bị khai thác thành công.

Từ việc khai thác trên, chúng ta đã lấy được toàn bộ thông tin về cơ sở dữ liệu bao gồm người dùng, mật khẩu và nhiều thông tin khác ở các bảng khác nằm trong cơ

sở dữ liệu (bao gồm cả thông tin được ẩn giấu trong cơ sở dữ liệu) từ đây có thể chỉnh sửa, xóa hoặc thêm dữ liệu vào cơ sở dữ liệu để kiểm soát và khai thác hệ thống.

3.3.4. Đề xuất các giải pháp nâng cao bảo mật cho hệ thống ứng dụng Web

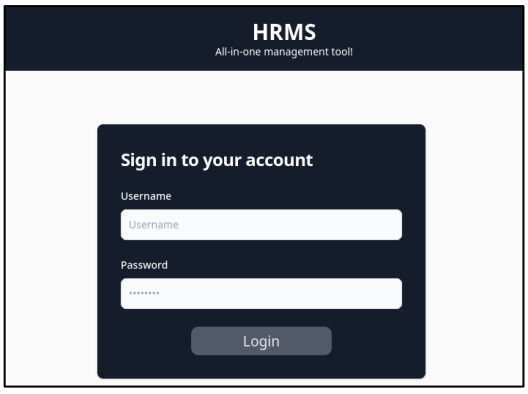
Từ các kết quả trên, chúng ta có thể nhận thấy được các nguy cơ và ảnh hưởng của SQL Injection đến cơ sở dữ liệu và hệ thống. Do đó, nhóm tác giả đề xuất một số giải pháp nhằm cải thiện tính bảo mật và an ninh cho hệ thống bao gồm:

***Giải pháp lập trình*:** Sử dụng truy vấn tham số hóa (Prepared Statements/Parameterized Queries) [5], Áp dụng nguyên tắc quyền tối thiểu - Least Privilege [6], Kiểm tra và làm sạch dữ liệu đầu vào (Input Validation) [7].

***Giải pháp hệ thống*:** Triển khai tường lửa cho ứng dụng Web (Web Application Firewall - WAF) [8], Sử dụng các công cụ quét bảo mật định kỳ [9], Cập nhật và vá lỗi hệ thống thường xuyên [10].

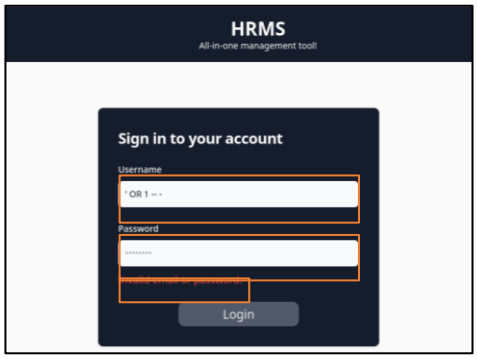
***Nâng cao nhận thức bảo mật*:** Tổ chức các khóa học lập trình an toàn cho nhà phát triển [11], Thực hiện kiểm tra bảo mật định kỳ [12].

Để kiểm tra tính khả thi của các giải pháp trên, tiến hành truy cập vào trang web: ***http://10.10.124.94/?url=localhost/copyright*** kết quả trả về như sau:



Hình 12. Trang web sau khi truy cập theo địa chỉ đã cung cấp.

Sau khi truy cập vào trang web trên, tiến hành truy cập vào trang web: ***https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/SQL%20Injection*** để sử dụng các payload SQL Injection độc hại từ trang web này. Sau khi có được các payload SQL Injection độc hại tiến hành thử tất cả trường hợp vào 2 khung Username và Password.



Hình 13. Kết quả sau khi đã thử toàn bộ trường hợp SQL Injection độc hại.

4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

SQL Injection là một trong những mối đe dọa bảo mật nghiêm trọng nhất đối với các ứng dụng/hệ thống web gây ra nhiều hậu quả nguy hiểm như rò rỉ, vi phạm dữ liệu, xâm nhập cơ sở dữ liệu trái phép hoặc kiểm soát hoàn toàn hệ thống. Qua bài báo này, nhóm tác giả đã phân tích chi tiết các kỹ thuật khai thác SQL Injection phổ biến. Từ đó, đưa ra các giải pháp phòng chống hiệu quả từ khía cạnh lập trình, hệ thống và nhận thức bảo mật.

Tuy nhiên, khi các kỹ tấn công ngày càng tinh vi cần đưa ra các biện pháp phòng chống và giải pháp tiên tiến hơn. Trong tương lai, nhóm tác giả đề xuất nghiên cứu và phát triển các ứng dụng/hệ thống tích hợp trí tuệ nhân tạo (AI) giúp tự động phát hiện và ngăn chặn các cuộc tấn công SQL Injection. Các thuật toán học máy hỗ trợ phân tích các mẫu tấn công phức tạp từ đó đưa ra các giải pháp thích hợp để bảo vệ ứng dụng/hệ thống.

TÀI LIỆU THAM KHẢO

[1]. OWASP Foundation (2024). "OWASP Top Ten Web Application Security Risks", Website: https://owasp.org/www-project-top-ten.

[2]. Akamai (2023). "State of the Internet Security Report", Website: https://www.akamai.com/solutions/security.

[3]. Bernardo Damele A. G., et al. (2024). *SQLmap – Automatic SQL Injection and Database Takeover Tool*, Website: https://sqlmap.org.

[4]. PortSwigger (2024). *Burp Suite – Web Vulnerability Scanner and Penetration Testing Toolkit*, Website: https://portswigger.net/burp.

[5]. L. Grossman (2020). "SQL Injection Mitigation Techniques," *Database Security Essentials*, 2nd ed., O'Reilly Media, tr. 95–120.

[6]. C. B. Williams (2019). "Principles of Database Security," *Database Security Best Practices,* McGraw-Hill Education, tr. 60–89.

[7]. M. Halfond, J. Viegas, và A. Orso (2006). *A Classification of SQL Injection Attacks and Countermeasures,* Proceedings of IEEE International Symposium on Secure Software Engineering, Arlington, USA, tr. 13–23.

[8]. R. Smith và J. Carter (2021). "Web Application Firewalls in Modern Security," *Journal of Information Security* [Online], Vol. 38, No. 2, tr. 112–145, Website: https://journalofinfosec.com/waf2021.

[9]. K. Mitnick và W. L. Simon (2020). "The Art of Intrusion Techniques," *Secrets and Lies: Digital Security in a Networked World,* Wiley Publishing, tr. 135–160.

[10]. OWASP Foundation (2023). "OWASP Top Ten 2023: Mitigation Strategies," *OWASP Resources* [Online], Website: https://owasp.org.

[11]. B. Schneier (2015). "Practical Approaches to Digital Security," *Secrets and Lies: Digital Security in a Networked World,* Wiley Publishing, tr. 203–225.

[12]. J. Cooper và A. Jones (2022). *The Role of Penetration Testing in Web Security, International Journal of Cybersecurity* [Online], Vol. 12, No. 4, tr. 67–88, Website: https://ijcybersec.org/penetration-testing-2022.

EXPLOITING SQL INJECTION VULNERABILITIES IN WEB APPLICATIONS

Phan Van Cuong, Ho Duc Tam Linh, Vuong Quang Phuoc*

Faculty of Electronics, Electrical Engineering and Material Technology,
University of Sciences, Hue University

*Email: vqphuoc@husc.edu.vn

ABSTRACT

SQL Injection (SQLi) is a widely known attack technique in web application security, which allows attackers to infiltrate and retrieve data from databases by exploiting improperly handled input vulnerabilities. This attack technique is commonly used to steal data or obtain system control. The prevalence of SQLi is primarily due to the lack of security methods during web application development. This paper analyzes and evaluates common SQL Injection attack methods, including Blind SQLi, Union-based SQLi, and Error-based SQLi. The results indicate that SQLi has a significant impact on database and system security. The paper also provides an overview of web application security and practical solutions for developing more secure systems, proposing preventive solutions such as parameterized queries, Web Application Firewalls (WAF), and secure encryption.

Keywords: SQL Injection, Blind SQLi, Error-based SQLi, Union-based SQLi, WAF.