| **ĐẠI HỌC ĐÀ NẴNG**  **TRƯỜNG ĐẠI HỌC BÁCH KHOA**  **KHOA CÔNG NGHỆ THÔNG TIN**    **PBL6: ĐỒ ÁN AN TOÀN THÔNG TIN**  **Đề tài: PHÁT HIỆN TẤN CÔNG CHÈN MÃ SQL DỰA TRÊN PHÂN TÍCH CÚ PHÁP CÂU TRUY VẤN**  **SINH VIÊN THỰC HIỆN:**  **Hồ Như Phong LỚP: 20TCL\_DT3 NHÓM: 20Nh14B**  **Trần Gia Huy LỚP: 20TCL\_DT1 NHÓM: 20Nh14B**  **GIẢNG VIÊN HƯỚNG DẪN: Ths.Nguyễn Văn Nguyên**  **Đà Nẵng 11/ 2023** |
| --- |

# MỤC LỤC

[MỤC LỤC 2](#_30j0zll)

[DANH SÁCH HÌNH VẼ 2](#_1fob9te)

[DANH SÁCH CÁC TỪ VIẾT TẮT 4](#_tyjcwt)

[GIỚI THIỆU ĐỀ TÀI 5](#_3dy6vkm)

[CHƯƠNG 1. CƠ SỞ LÝ THUYẾT 6](#_1t3h5sf)

[1.1. Tổng quan về tấn công chèn mã SQL và các biện pháp phòng chống 6](#_4d34og8)

[1.1.1. Khái quát về tấn công chèn mã SQL 6](#_8kirexb8c85s)

[1.1.2. Cơ chế tấn công chèn mã SQL 7](#_2l7dv7ydaim3)

[1.1.3. Các dạng tấn công chèn mã SQL 8](#_od07e26grrs9)

[1.2. Các biện pháp phòng chống tấn công chèn mã SQL 9](#_aqw4pwi5r9va)

[1.2.1. Các biện pháp phòng chống ở mức lập trình 9](#_i7vx26hq69wn)

[1.2.2. Các biện pháp phòng chống ở mức độ nền tảng 11](#_jatpfz7ibvgg)

[CHƯƠNG 2. ỨNG DỤNG CÁC THUẬT TOÁN TRONG AN TOÀN THÔNG TIN 13](#_3rdcrjn)

[2.1. Khái quát về ngôn ngữ SQL 13](#_26in1rg)

[2.2. Phát hiện tấn công chèn mã SQL dựa trên phân tích cú pháp câu lệnh 13](#_nxpkz5tk8chr)

[2.2.1. Xây dựng các đặc tả câu lệnh SQL hợp lệ 13](#_yhpxaq5a5s25)

[2.2.2. Bắt câu lệnh SQL từ ứng dụng web 14](#_472i2dhc9sih)

[2.2.3. Phân tích cú pháp câu lệnh SQL 14](#_wlqfswx6b0t2)

[CHƯƠNG 3. TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ 19](#_35nkun2)

[3.1. Xây dựng mô hình thử nghiệm 19](#_1ksv4uv)

[3.1.1. Kiến trúc bộ lọc câu lệnh SQL - Database Filter 19](#_ze29po71eo9w)

[3.1.2. Cài đặt bộ lọc cơ sở dữ liệu 20](#_g7spps63uzdl)

[3.1.3. Giới thiệu mô đun proxy sử dụng cho chương trình 20](#_nq9u2i66ofh4)

[3.2. Kết quả triển khai 21](#_eg1u60qsv7oj)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 24](#_2jxsxqh)

[1. Kết quả đạt được 24](#_xod54kcd6bgn)

[2. Hạn chế 24](#_vuqc19e54ije)

[3. Hướng phát triển 24](#_k2dephpd7mur)

[TÀI LIỆU THAM KHẢO 25](#_mci47ucirivg)

# DANH SÁCH HÌNH VẼ

[Hình 2.1: Minh họa đặc tả câu lệnh 14](#_z337ya)

[Hình 2.2: Sơ đồ bố trí bộ lọc CSDL 14](#_1y810tw)

[Hình 2.3: Ví dụ phân tích câu truy vấn 15](#_rcr3dzbtk6kj)

[Hình 2.4: Mô hình hoạt động phân tích cấu trúc truy vấn 15](#_lepjukppd3dh)

[Hình 3.1: Mô hình bộ lọc cơ sở dữ liệu 19](#_ls15jj8lozfw)

[Hình 3.2: Giao diện đăng nhập 21](#_v4gyma2ztu4a)

[Hình 3.3: Giao diện quản lý 21](#_qlu5anzdrome)

[Hình 3.4: Mô phỏng tấn công chèn mã 22](#_ss1j4ium3srq)

[Hình 3.5: Phát hiện từ khóa nguy hiểm khi đăng nhập 22](#_b30ynm7i8p91)

[Hình 3.6:Danh sách địa chỉ ip và thời gian đăng nhập 23](#_kdmpprl3bls5)

[Hình 3.7: Danh sách địa chỉ ip bị chặn truy cập 23](#_oc07vtlb7kg5)

# 

# DANH SÁCH CÁC TỪ VIẾT TẮT

| **STT** | **Từ viết tắt** | **Từ đầy đủ** |
| --- | --- | --- |
| 1 | SQL | Structured Query Language |
| 2 | HTTP | The Hypertext Transfer Protocol |
| 3 | IDS | Intrusion Detection System |
| 4 | OWASP | The Open Web Application Security Project |
| 5 | WAF | Web Application Firewall |

# GIỚI THIỆU ĐỀ TÀI

Trong thời đại số hóa ngày càng phát triển, việc quản lý và bảo vệ dữ liệu trở nên quan trọng hơn bao giờ hết. Một trong những mối đe dọa lớn đối với an toàn dữ liệu là tấn công chèn mã SQL (SQL injection attack). Đây là một trong những phương pháp tấn công phổ biến mà kẻ xâm nhập sử dụng để tận dụng lỗ hổng trong ứng dụng web và xâm nhập vào cơ sở dữ liệu của hệ thống.

Tấn công chèn mã SQL xảy ra khi kẻ tấn công thêm hoặc thay đổi các câu truy vấn SQL trong ứng dụng web để truy cập hoặc kiểm soát dữ liệu cơ sở dữ liệu. Điều này có thể dẫn đến tiết lộ thông tin quan trọng, mất dữ liệu, hoặc thậm chí là kiểm soát hệ thống hoàn toàn.

Mục tiêu của đề tài này là xây dựng một hệ thống phát hiện tấn công chèn mã SQL dựa trên phân tích cú pháp câu truy vấn. Hệ thống sẽ giám sát các câu truy vấn SQL được gửi từ ứng dụng web và sẽ kiểm tra tính hợp lệ của cú pháp của mỗi câu truy vấn. Nếu phát hiện bất kỳ dấu hiệu nào của chèn mã SQL, hệ thống sẽ ngăn chặn câu truy vấn đó và thông báo cho quản trị viên.

Ý nghĩa của đề tài này nằm ở việc cung cấp một cách hiệu quả và tự động để bảo vệ các ứng dụng web khỏi tấn công chèn mã SQL. Điều này giúp tăng cường an ninh thông tin, bảo vệ dữ liệu quan trọng và đảm bảo tính ổn định của hệ thống.

# CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

## Tổng quan về tấn công chèn mã SQL và các biện pháp phòng chống

### Khái quát về tấn công chèn mã SQL

#### *SQL Injection là gì*

SQL Injection là một kỹ thuật tấn công phổ biến trong lĩnh vực bảo mật web. Khi thực hiện SQL Injection, kẻ tấn công cố gắng chèn các đoạn mã SQL độc hại vào các truy vấn SQL được thiết kế để tương tác với cơ sở dữ liệu của ứng dụng web. Mục tiêu của tấn công này là lợi dụng lỗ hổng bảo mật trong việc xử lý đầu vào người dùng trong ứng dụng web để thực hiện các hành động không được phép.

Đối với một trang web hoặc ứng dụng web không đủ bảo vệ, khi người tấn công chèn các đoạn mã SQL vào các ô nhập liệu hoặc các tham số truyền vào URL, cơ sở dữ liệu có thể thực hiện các truy vấn mà người tấn công mong muốn. Điều này có thể dẫn đến những hậu quả nghiêm trọng, bao gồm:

* Truy cập không được phép: Người tấn công có thể thu thập, đọc, hoặc thậm chí xóa thông tin khỏi cơ sở dữ liệu.
* Thay đổi dữ liệu: Kẻ tấn công có thể thực hiện các truy vấn SQL để thay đổi dữ liệu trong cơ sở dữ liệu, có thể làm hỏng tính toàn vẹn của dữ liệu.
* Thực hiện các thao tác không hợp pháp: Người tấn công có thể thực hiện các thao tác như tạo, sửa đổi hoặc xóa bảng cơ sở dữ liệu.

1. *Mục đích tấn công SQL Injection*

Mục đích chính của tấn công SQL Injection là lợi dụng các lỗ hổng bảo mật trong ứng dụng web hoặc trang web để thực hiện các hành động không được phép đối với cơ sở dữ liệu. Dưới đây là một số mục đích phổ biến của kẻ tấn công khi thực hiện SQL Injection:

* **Thu thập thông tin về cơ sở dữ liệu**: Kẻ tấn công muốn khám phá các loại và phiên bản của cơ sở dữ liệu được ứng dụng web sử dụng. Một số loại cơ sở dữ liệu có những phản ứng lỗi khác nhau khi thực truy vấn và tấn công, những thông tin này có thể sử dụng để thu thập thông tin về cơ sở dữ liệu.
* **Thay đổi dữ liệu**: Kẻ tấn công có thể thực hiện các truy vấn SQL để thay đổi dữ liệu trong cơ sở dữ liệu. Điều này có thể bao gồm việc sửa đổi thông tin người dùng, thay đổi quyền truy cập, hoặc thậm chí xóa dữ liệu.
* **Thực hiện các thao tác không hợp pháp**: SQL Injection cũng có thể được sử dụng để thực hiện các thao tác không hợp pháp đối với cơ sở dữ liệu, bao gồm tạo bảng mới, thêm người dùng có quyền đặc biệt, hoặc thậm chí thay đổi cấu trúc của cơ sở dữ liệu.
* **Kiểm thử bảo mật**: Một số người cố gắng thực hiện SQL Injection như một phương tiện để kiểm thử bảo mật của hệ thống. Những người này thực hiện tấn công để xác định xem ứng dụng web có bị lỗ hổng SQL Injection hay không, giúp tổ chức cải thiện tính bảo mật của hệ thống.

1. *Hậu quả của tấn công SQL Injection*

Tấn công SQL Injection có thể gây ra nhiều hậu quả nghiêm trọng đối với cả ứng dụng web và dữ liệu trong cơ sở dữ liệu. Dưới đây là một số hậu quả phổ biến của tấn công SQL Injection:

* **Truy cập không được phép**: Kẻ tấn công có thể đọc thông tin nhạy cảm từ cơ sở dữ liệu mà họ không có quyền truy cập. Điều này có thể bao gồm tên người dùng, mật khẩu, thông tin tài khoản ngân hàng, và các thông tin cá nhân khác.
* **Thay đổi dữ liệu**: Khi tấn công thành công, kẻ tấn công có thể thực hiện các thay đổi không hợp lý trong cơ sở dữ liệu. Điều này có thể làm hỏng tính toàn vẹn của dữ liệu bằng cách thay đổi thông tin người dùng, thêm hoặc xóa dữ liệu.
* **Lợi dụng quyền hạn**: Nếu ứng dụng web chạy trên cơ sở dữ liệu với quyền hạn cao, tấn công SQL Injection có thể dẫn đến việc lợi dụng quyền hạn này. Kẻ tấn công có thể thực hiện các hành động như tạo, sửa đổi hoặc xóa các bảng cơ sở dữ liệu.
* **Mất dữ liệu**: Thao tác không đúng từ kẻ tấn công có thể dẫn đến mất mát dữ liệu quan trọng. Điều này có thể ảnh hưởng nghiêm trọng đến hoạt động kinh doanh và dẫn đến rủi ro phá sản.
* **Mất uy tín**: Nếu thông tin nhạy cảm của người dùng bị rò rỉ hoặc bị thay đổi, ứng dụng web và tổ chức có thể mất uy tín. Người dùng có thể mất niềm tin vào sự bảo mật của hệ thống.
* **Tấn công tiếp theo**: Kẻ tấn công có thể sử dụng thông tin thu thập được từ tấn công SQL Injection để thực hiện các tấn công tiếp theo, chẳng hạn như tấn công phishing hoặc tấn công mạng xã hội.

### Cơ chế tấn công chèn mã SQL

Tấn công chèn mã SQL (SQL Injection) thường được thực hiện bằng cách chèn các đoạn mã SQL độc hại vào các trường nhập liệu hoặc tham số của ứng dụng web. Dưới đây là một số cơ chế phổ biến mà kẻ tấn công sử dụng để thực hiện SQL Injection:

* **Chèn vào trường nhập liệu**: Kẻ tấn công thường chèn các đoạn mã SQL vào các trường nhập liệu của biểu mẫu trên trang web, như ô tìm kiếm, biểu mẫu đăng nhập, hoặc các trường khác.

**Ví dụ**: Nếu ứng dụng web chưa kiểm tra đầu vào người dùng đúng cách, người tấn công có thể nhập **“' OR '1'='1' –“** vào ô tìm kiếm để đảm bảo câu truy vấn luôn đúng và trả về kết quả.

* **Chèn mã thông qua Cookie**: Cookie là một tập tin chứa các thông tin trạng thái được tạo ra bởi các ứng dụng web và lưu trữ trên trình duyệt của máy khách. Các thông tin trong cookie có thể là tên đăng nhập, mật khẩu, các lựa chọn của người dùng đi kèm. Nếu ứng dụng web sử dụng nội dung của cookie để xây dựng các câu truy vấn SQL, kẻ tấn công có thể dễ dàng thực hiện một cuộc tấn công chèn mã bằng cách nhúng các câu truy vấn độc hại vào trong cookie.
* **Chèn vào câu lệnh SQL trực tiếp:** Nếu ứng dụng web không kiểm tra và xử lý đầu vào người dùng một cách an toàn, kẻ tấn công có thể chèn đoạn mã SQL trực tiếp vào câu lệnh SQL được thực thi bởi ứng dụng.

**Ví dụ**: Nếu một trang web sử dụng mã SQL như **SELECT \* FROM users WHERE username = '$input';**, kẻ tấn công có thể nhập **' OR '1'='1'; --** để làm suy đoán câu lệnh và thực hiện tấn công.

### Các dạng tấn công chèn mã SQL

Có nhiều dạng tấn công chèn mã SQL (SQL Injection) phức tạp mà kẻ tấn công có thể sử dụng để lợi dụng các lỗ hổng bảo mật trong ứng dụng web. Dưới đây là một số dạng phổ biến của tấn công này:

* *Classic SQL Injection*: Kẻ tấn công chèn đoạn mã SQL vào các trường nhập liệu của ứng dụng web, chẳng hạn như ô tìm kiếm, biểu mẫu đăng nhập, hoặc trường khác.

*Ví dụ*: **' OR '1'='1'; --**

* *Tấn công sử dụng câu lệnh SELECT:* Kiểu tấn công này cho phép kẻ tấn công thu thập các thông tin quan trọng về kiểu và cấu trúc của hệ thống cơ sở dữ liệu đầu cuối của ứng dụng web.

*Ví dụ*: Trang đăng nhập có 3 trường đầu vào là username, password và mã pin với đoạn truy vấn sau để thực hiện việc đăng nhập:

**SELECT accounts FROM users WHERE login=’’ AND pass=’’ AND pin=**

Mục đích gây ra lỗi có thể tiết lộ thông tin dữ liệu. Để làm việc này kẻ tấn công sẽ chèn nội dung sau vào trong trường đầu vào chứa mã pin: *“convert(int,(select top 1 name from sys.objects where xtype=’u’))”*. Câu truy vấn trở thành:

**SELECT accounts FROM users WHERE login=’’ AND pass=’’ AND pin= convert (int,(select top 1 name from sys.objects where xtype=’u’))**

* *Tấn công khai thác dữ liệu thông qua toán tử UNION:* Trong kiểu tấn công này, kẻ tấn công khai thác một tham số dễ bị tổn thương để thay đổi các thiết lập dữ liệu trả về với một câu truy vấn, từ đó lừa ứng dụng trong việc trả về dữ liệu từ bảng dữ liệu khác.

*Ví dụ*: Với bảng *Credit Cards*, kẻ tấn công sẽ chèn đoạn mã “ **‘UNION SELECT cardNo FROM CreditCards WHERE acctNo=10032--** ” vào trong trường login. Khi đó câu query sẽ trở thành:

**SELECT accounts FROM users WHERE login=’’ UNION SELECT cardNo from CreditCards WHERE acctNo=10032-- AND pass=’’ AND pin=**

## Các biện pháp phòng chống tấn công chèn mã SQL

### Các biện pháp phòng chống ở mức lập trình

#### *Làm sạch dữ liệu đầu vào*

Được coi là công việc quan trọng đầu tiên cần xử lý trong chuỗi các thao tác. Có hai mô hình có thể được áp dụng cho việc lọc dữ liệu đầu vào, đó là sử dụng danh sách cho phép – whitelist, hoặc danh sách cấm – blacklist.

1. Mô hình danh sách cho phép - Whitelist

Mô hình whitelist liệt kê danh sách những giá trị input nào được cho phép, một số đặc điểm của input mà mô hình này chú ý tới như kiểu dữ liệu, độ dài, miền dữ liệu (đối với input kiểu số) hoặc một số định dạng chuẩn khác

Phương pháp đơn giản và hiệu quả nhất để xây dựng các mẫu (pattern) hợp lệ là sử dụng biểu thức chính quy (regular expression). Xét một số mẫu biểu thức chính quy áp dụng cho username, password sau đây:

* Username: chỉ chứa các ký tự chữ cái, chữ số và dấu gạch dưới, độ dài tối đa 30 ký tự, tối thiểu 3 ký tự:

**“^([a-zA-Z0-9]|\_){3,30}$”**

* Password: chỉ chứa ký tự chữ cái, chữ số, dấu gạch dưới, độ dài tối thiểu 4, tối đa 50

**“^([a-zA-Z0-9]|\_){4,50}$”**

1. Mô hình danh sách cấm - Blacklist

Mô hình này xây dựng nên các mẫu input được cho là nguy hiểm và sẽ không chấp nhận những mẫu này. Mô hình blacklist kém hiệu quả hơn mô hình whitelist.

Mẫu này tiến hành tìm sự xuất hiện của các ký tự như dấu nháy đơn, %, --, dấu chấm phảy, \\*, \*/, \_, [, @, xp\_, đương nhiên mẫu này không phải là một mẫu đủ tốt để có thể đảm bảo một đầu vào là “sạch”.

#### *Xây dựng truy vấn theo mô hình tham số hóa*

Xây dựng truy vấn theo mô hình tham số hóa là một biện pháp quan trọng để ngăn chặn tấn công chèn mã SQL. Thay vì sử dụng chuỗi để tạo câu lệnh SQL, mô hình tham số hóa sử dụng các tham số trong câu lệnh truy vấn, giảm nguy cơ bị tấn công SQL Injection.

1. *Sử dụng các câu lệnh thực thi tham số hóa*

* Tham số hóa trong Java

*String sql = "SELECT \* FROM users WHERE username = ? AND password = ?";*

*PreparedStatement statement = connection.prepareStatement(sql);*

*statement.setString(1, userInputtedUsername);*

*statement.setString(2, userInputtedPassword);*

*ResultSet resultSet = statement.executeQuery();*

* Tham số hóa trong C#

*string sql = "SELECT \* FROM users WHERE username = @username AND password = @password";*

*using (SqlCommand command = new SqlCommand(sql, connection))*

*{*

*command.Parameters.AddWithValue("@username", userInputtedUsername);*

*command.Parameters.AddWithValue("@password", userInputtedPassword);*

*SqlDataReader reader = command.ExecuteReader();*

*}*

* Tham số hóa trong PHP

*$sql = "SELECT \* FROM users WHERE username = :username AND password = :password";*

*$statement = $pdo->prepare($sql);*

*$statement->bindParam(':username', $userInputtedUsername);*

*$statement->bindParam(':password', $userInputtedPassword);*

*$statement->execute();*

#### *Chuẩn hóa dữ liệu*

Để thuận lợi cho quá trình kiểm tra dữ liệu đầu vào và đầu ra, chúng ta cần xây dựng các mô hình chuẩn hóa dữ liệu dưới một dạng đơn giản. Một mô hình có thể xem xét như, ban đầu giải mã dưới dạng URL, sau đó giải mã dưới dạng HTML, có thể thực hiện vài lần. Tuy nhiên có thể sẽ tin cậy hơn nếu chúng ta chỉ thực hiện giải mã theo định dạng phổ biến nhất nào đó đúng 1 lần, nếu phát hiện dấu hiệu nghi vấn, lập tức từ chối dữ liệu đó.

#### *Sử dụng ORM (Object-Relational Mapping)*

Sử dụng các framework ORM như Hibernate (Java), Entity Framework (C#), SQLAlchemy (Python) giúp tự động thực hiện tham số hóa và giảm khả năng xảy ra SQL Injection.

#### *Kiểm tra và xử lý an toàn đầu vào*

Kiểm tra và xử lý an toàn đầu vào người dùng để đảm bảo rằng dữ liệu nhập liệu không chứa các ký tự đặc biệt có thể làm suy đoán câu lệnh SQL.

### Các biện pháp phòng chống ở mức độ nền tảng

#### *Các biện pháp bảo vệ tức thời*

Bảo vệ trong quá trình chạy gồm các giải pháp sử dụng để phát hiện, giảm thiểu, hoặc ngăn chặn tấn công chèn mã SQL mà không cần sửa đổi mã nguồn ứng dụng để khắc phục lỗi chèn mã SQL. Bảo vệ trong quá trình chạy có thể được xem là giải pháp hữu ích cho các ứng dụng đang hoạt động có chứa lỗ hổng tấn công chèn mã SQL mặc dù việc chỉnh sửa mã nguồn để khắc phục lỗ hổng đó vẫn là biện pháp tốt nhất. Bảo vệ trong thời gian chạy là một cơ chế bảo vệ phản ứng, nhưng cũng là một bước chủ động trong việc hướng tới sự bảo vệ toàn diện cho ứng dụng khỏi tấn công chèn mã SQL.

#### *Các ứng dụng tường lửa Web (WAF)*

Giải pháp WAF (Web Application Firewall) là một giải pháp toàn diện hơn so với giải pháp bảo vệ thời gian chạy. WAF mà một thiết bị mạng hoặc phần mềm có các chức năng bảo mật cho một ứng dụng web tránh khỏi các cuộc tấn công trong đó có chèn mã SQL. Lợi ích của việc sử dụng WAF dựa trên phần mềm là nó không làm thay đổi hạ tầng triển khai máy chủ và các giao thức HTTP/HTTPS được sử dụng để truyền thông từ phía người dùng. WAF dựa trên phần mềm không tiêu tốn tài nguyên của máy chủ mà, nhưng có khả năng giúp nhiều dịch vụ của máy chủ web được bảo vệ bởi nhiều công nghệ khác nhau. Sau đây là một số WAF thường được sử dụng:

**ModSecurity**: Là một WAF mã nguồn mở được cài đặt như một mô đun của máy chủ web Apache. ModSecurity có thể sử dụng để giám sát, phát hiện tấn công, phát hiện xâm nhập vào các ứng dụng web chạy trên máy chủ Apache.

**Intercepting Filter**: Đây là dạng WAF thực hiện mô hình lọc chặn hoặc bao gồm nhiều mô hình lọc chặn khác nhau cùng triển khai trên một kiến trúc tổng thể. Bộ lọc chặn có thể gồm một số các mô đun độc lập có thể kết hợp lại với nhau để thực hiện các khâu xử lý một tài nguyên được yêu cầu (Web page, url...). Bộ lọc phù hợp cho việc xác nhận đầu vào, ghi nhật ký các yêu cầu/phản ứng và chuyển đổi của ứng dụng web.

#### *Sử dụng hệ thống phát hiện xâm nhập*

Là hệ thống phát hiện xâm nhập dựa trên lưu lượng trên mạng để phát hiện ra tấn công chèn mã SQL. Tuy nhiên, các IDS thường không được tối ưu riêng cho ứng dụng web. Nó chỉ giúp phát hiện ra các cuộc tấn công từ bên ngoài mạng nội bộ, tuy nhiên vẫn có thể sử dụng nó như một phòng tuyến bổ sung chống lại các cuộc tấn công.

# CHƯƠNG 2. ỨNG DỤNG CÁC THUẬT TOÁN TRONG AN TOÀN THÔNG TIN

## Khái quát về ngôn ngữ SQL

SQL (Structured Query Language) là một ngôn ngữ lập trình sử dụng để quản lý và tương tác với cơ sở dữ liệu quan hệ. SQL được thiết kế để thực hiện các thao tác như truy vấn dữ liệu, cập nhật dữ liệu, chèn dữ liệu, xóa dữ liệu, và quản lý cấu trúc của cơ sở dữ liệu.

SQL được sử dụng để điều khiển tất cả các chức năng mà một hệ quản trị cơ sở dữ liệu cung cấp cho người dùng bao gồm:

* **Định nghĩa dữ liệu**: SQL cung cấp khả năng định nghĩa các cơ sở dữ liệu, các cấu trúc lưu trữ và tổ chức dữ liệu cũng như mối quan hệ giữa các thành phần dữ liệu.
* **Truy xuất và thao tác dữ liệu**: Với SQL, người dùng có thể dễ dàng thực hiện các thao tác truy xuất, bổ sung, cập nhật và loại bỏ dữ liệu trong các cơ sở dữ liệu.
* **Điều khiển truy cập**: SQL có thể được sử dụng để cấp phát và kiểm soát các thao tác của người sử dụng trên dữ liệu, đảm bảo sự an toàn cho cơ sở dữ liệu.
* **Đảm bảo toàn vẹn dữ liệu**: SQL định nghĩa các ràng buộc toàn vẹn trong cơ sở dữ liệu nhờ đó đảm bảo tính hợp lệ và chính xác của dữ liệu trước các thao tác cập nhật cũng như các lỗi của hệ thống.

## Phát hiện tấn công chèn mã SQL dựa trên phân tích cú pháp câu lệnh

### Xây dựng các đặc tả câu lệnh SQL hợp lệ

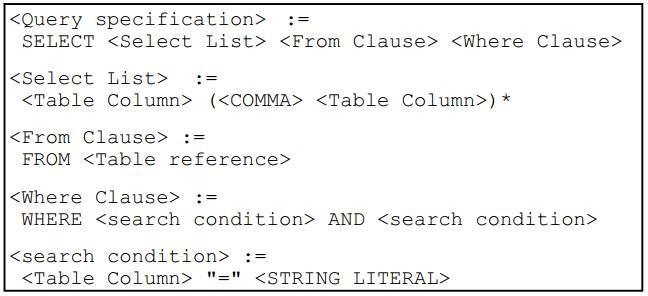
Đặc tả các câu lệnh SQL hợp lệ là một trong các thành phần quan trọng trong việc đánh giá một câu lệnh gửi từ ứng dụng web đến cơ sở dữ liệu là hợp lệ hay không. Một trong các yêu cầu chính của các đặc tả là tính hoàn thiện, tức là chúng phải bao gồm các luật cho từng lệnh gửi bởi ứng dụng web. Nếu các đặc tả không bao phủ hết các câu lệnh có thể có của ứng dụng web, các lỗi phát hiện (cảnh báo sai dương, hoặc sai âm) có thể xảy ra. Các đặc tả được xây dựng dựa trên ký hiệu Extended Backus-Naur Form (EBNF).

Xem xét một ví dụ xây dựng đặc tả một câu lệnh:

SELECT user\_id, full\_name, email FROM tbl\_users

WHERE username = ‘cuong’ AND password = ‘abc12345’

Hình 2.1 minh họa đặc tả tạo cho câu lệnh trên. Các luật của đặc tả này chỉ ra các trình tự hợp lệ của các thành phần của câu lệnh không có tấn công.



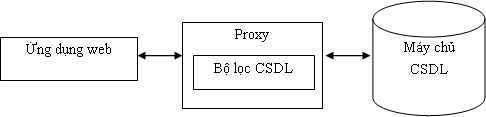
##### Hình 2.1: Minh họa đặc tả câu lệnh

### Bắt câu lệnh SQL từ ứng dụng web

Thông thường, ứng dụng web được cấu hình để gửi thẳng các câu lệnh SQL sang máy chủ cơ sở dữ liệu để thực hiện.

Với máy chủ cơ sở dữ liệu Microsoft SQL Server, ứng dụng web được cấu hình để kết nối với máy chủ cơ sở dữ liệu sử dụng một chuỗi kết nối (connection string) thông qua cổng TCP 1433. Để bộ lọc cơ sở dữ liệu có thể hoạt động, ứng dụng web cần được cấu hình để kết nối và gửi các câu lệnh SQL đến bộ lọc cơ sở dữ liệu, bộ lọc tiến hành đánh giá câu lệnh nhận được. Nếu câu lệnh SQL được đánh giá là hợp lệ thì nó được chuyển tiếp đến máy chủ cơ sở dữ liệu để thực hiện. Ngược lại, nếu câu lệnh SQL được đánh giá là không hợp lệ thì nó được ghi log và một thông báo lỗi thực hiện được gửi lại cho ứng dụng web.

Hình 2.2 dưới đây minh họa giao tiếp giữa Ứng dụng web, Proxy và Máy chủ CSDL, trong đó Proxy là thành phần trung gian có nhiệm vụ bắt các gói tin TCP gửi từ máy chủ web, chuyển thành các câu lệnh SQL, sau đó chuyển cho Bộ lọc CSDL kiểm tra, đánh giá các câu lệnh SQL.



##### Hình 2.2: Sơ đồ bố trí bộ lọc CSDL

### Phân tích cú pháp câu lệnh SQL

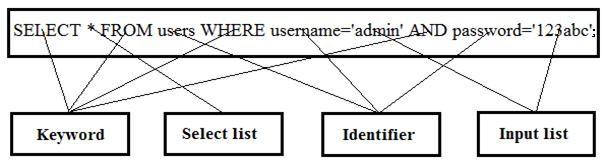
#### *Phân tích cú pháp câu truy vấn*

Phân tích cú pháp câu truy vấn là việc phân tích các thành phần đại diện của ngôn ngữ SQL được sử dụng trong câu truy vấn như cấu trúc ngữ pháp từ vựng của ngôn ngữ ( ví dụ như: MS-SQL, MySQL.. ). Khi câu truy vấn nguy hiểm được người dùng chèn vào thành công và gửi đến máy chủ cơ sở dữ liệu, hệ thống phân tích cú pháp sẽ so sánh và phát hiện câu truy vấn không hợp lệ.

Để đảm bảo việc phân tích câu truy vấn có thể phân loại đúng các thành phần của câu truy vấn như từ khóa, kiểu dữ liệu, dữ liệu từ người dùng, các trường khóa truy xuất dữ liệu và cần định nghĩa các loại câu lệnh của câu truy vấn một cách rõ ràng theo đúng quy chuẩn:

*SELECT \* FROM users WHERE username=’?’ AND password=’?’*

Các câu truy vấn được sử dụng đều theo quy chuẩn của ngôn ngữ SQL. Dựa trên đặc điểm đó, phân tích cấu trúc câu truy vấn dựa trên những từ khóa và cấu trúc ngữ pháp của ngôn ngữ SQL để phân tách các thành phần của một câu truy vấn khi được gửi đến hệ quản trị cơ sở dữ liệu.



##### Hình 2.3: Ví dụ phân tích câu truy vấn

#### 

##### Hình 2.4: Mô hình hoạt động phân tích cấu trúc truy vấn

#### Việc phân tích trên được thực hiện bằng cách phân tích câu truy vấn thành các từ. Các từ này sẽ được chuẩn hóa lại và so sánh chúng lần lượt với danh sách các từ khóa của ngôn ngữ truy vấn SQL được xây dựng sẵn. Danh sách các từ khóa ngôn ngữ SQL xây dựng từ bảng danh sách các từ khóa được định nghĩa cho mỗi hệ quản trị cơ sở dữ liệu mà máy chủ cơ sở dữ liệu sử dụng. Sau khi lọc tách được các từ khóa theo thứ tự ta sẽ được cấu trúc cơ bản của câu truy vấn. Các từ không thuộc danh sách từ khóa có thể là tên của cột, bảng và dữ liệu đầu vào của câu truy vấn. Việc phân tách các dữ liệu này dựa trên cấu trúc ngữ pháp của câu truy vấn SQL:

* *Đối với câu lệnh SELECT*: dữ liệu nằm sau từ khóa này và trước từ khóa FROM là thông tin về các cột dữ liệu được truy xuất trong các bảng nằm phía sau từ khóa FROM.
* *Đối với câu lệnh INSERT INTO*: dữ liệu nằm sau từ khóa INTO là bảng cần truy cập, tiếp theo đó là các trường nằm trong dấu ngoặc là các cột dữ liệu trong bảng, dữ liệu nằm trong dấu ngoặc sau từ khóa VALUES là dữ liệu nhạy cảm từ phía người dùng.
* *Đối với câu lệnh UPDATE*: dữ liệu sau từ khóa UPDATE là bảng dữ liệu cần truy cập, sau từ khóa SET là từng cặp tên cột = dữ liệu nhạy cảm của người dùng.
* *Đối với câu lệnh DELETE*: Sau FROM là bảng dữ liệu xét đến.

#### Trong ngôn ngữ truy vấn cơ sở dữ liệu SQL, các câu truy vấn rất phong phú và phức tạp tùy theo nhu cầu của người phát triển cũng như mục tiêu dữ liệu truy vấn, số lượng các loại dữ liệu cần truy xuất, độ phức tạp của điều kiện truy vấn.

#### Cơ sở của phân tích, phân tách cấu trúc dữ liệu cấu trúc câu truy vấn dựa trên các đặc điểm riêng biệt của các từ khóa và cú pháp được sử dụng trong ngôn ngữ truy vấn SQL. Dữ liệu thu được từ việc phân tích cấu trúc câu truy vấn được sử dụng trong mô hình phát hiện tấn công chèn mã SQL.

#### *Các bước tiến hành phân tích cú pháp câu lệnh*

***Câu truy vấn đầu vào:***

*SELECT \* FROM users WHERE username='admin' AND password='123abc'*

#### *Bước 1: Tách câu truy vấn*

Tách câu truy vấn thành 3 phần: cấu trúc, danh sách các thành phần truy vấn, dữ liệu đầu vào từ người dùng. Dữ liệu bao gồm cấu trúc của câu truy vấn và dữ liệu nhạy cảm trong câu truy vấn. Dữ liệu từ bước này là thành phần đầu vào của bước kiểm tra phát hiện tấn công chèn mã SQL.

#### *Bước 2: Kiểm tra cấu trúc và dữ liệu*

1. ***Kiểm tra cấu trúc***: Hỗ trợ 2 lựa chọn thực hiện

* *Lựa chọn 1*: Danh sách cấu trúc câu truy vấn SQL sử dụng trong quá trình so sánh cấu trúc câu truy vấn được tạo chủ quan theo hiểu biết và tham khảo về các câu truy vấn thông dụng trong các ứng dụng web.
* *Lựa chọn 2*: Danh sách cấu trúc câu truy vấn sử dụng trong ứng dụng được nhà phát triển ứng dụng web cung cấp và mô đun phân tích cấu trúc ở bước 1 phân tích thành cấu trúc chuẩn vào lưu vào file sử dụng trong quá trình kiểm tra dưới dạng các đặc tả như trình bày ở mục 2.2.1.

##### 

1. ***Kiểm tra thành phần câu truy vấn****:*

Danh sách các cột, bảng trong câu truy vấn gửi đến máy chủ cơ sở dữ liệu tách ra ở bước 1 được so sánh với bộ danh sách các cột, bảng dữ liệu nhạy cảm không được phép truy cập. Danh sách này được lấy từ các cheat sheet do các nhà nghiên cứu và đảm bảo an toàn máy chủ ứng dụng web sử dụng cơ sở dữ liệu đưa ra như cho trên Bảng 2.1 - Danh sách một số từ khóa nguy hiểm.

| *"@@version", "version", "userO", "system\_userO",*  *"mysql.user", "alter", "begin", "cast", "create",*  *"cursor", "href", "declare", "delete", "drop", "exec",*  *"execute", "fetch", "insert", "kill", "open", "select",*  *"sys", "sysobjects", "syscolumns", "table", "update",*  *"CHARO", "dumpfile", "load\_fileO", "Grant ",*  *"PRIVILEGES", "schemata", "information\_schema",*  *"column\_name", "table\_name", tables", "schema\_name",*  *"Super\_priv", "table\_schema", "columns", "@@datadir",*  *"ASCIIO", "IF", "OUTFILE", "grantee", "privilege\_type",*  *"is\_grantable", "user\_privileges", "host", "user", "Select\_priv",*  *"Insert\_priv", "Update\_priv", "Delete\_priv",* |
| --- |

###### Bảng 2.1: Danh sách một số từ khóa nguy hiểm

1. ***Kiểm tra dữ liệu đầu vào***

Dữ liệu đầu vào của người dùng bình thường có thể chứa các đoạn mã nguy hiểm cần kiểm tra, lọc dựa trên các thư viện lọc dữ liệu chuẩn cung cấp bởi dự án OWASP.

***Bước 3: Kiểm tra kết thúc***

Nếu trong quá trình kiểm tra phát hiện ra tấn công chèn mã SQL thì mọi thông tin về câu truy vấn, thời gian thực hiện, địa chỉ IP của ứng dụng web đều được lưu trữ lại để phục vụ quá trình điều tra, nghiên cứu về sau. Đồng thời câu truy vấn bị loại bỏ. Ngược lại nếu câu truy vấn là hợp lệ nó được chuyển đến máy chủ cơ sở dữ liệu để thực hiện.

# 

# CHƯƠNG 3. TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ

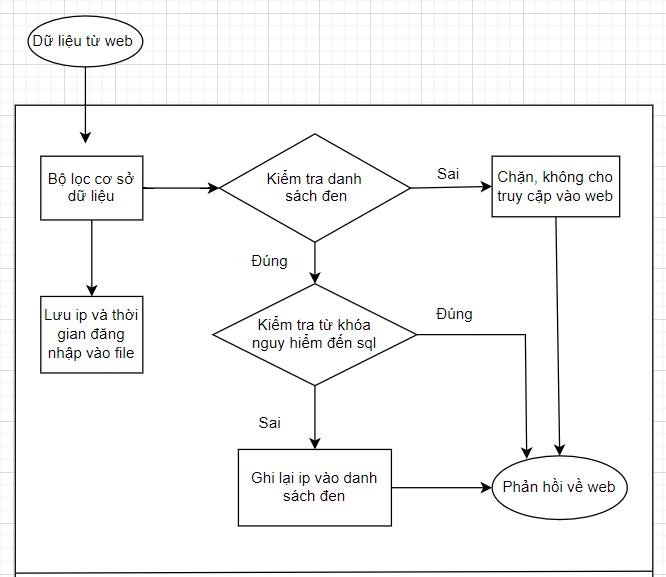
## Xây dựng mô hình thử nghiệm

### Kiến trúc bộ lọc câu lệnh SQL - Database Filter

Mô hình bộ lọc cơ sở dữ liệu được thực hiện theo 3 giai đoạn xử lý:

**Giai đoạn 1**: Xử lý dữ liệu đầu vào. Dữ liệu đầu vào ở đây là dữ liệu người dùng nhập vào để đăng nhập. Sẽ được gửi từ web qua bộ lọc

**Giai đoạn 2**: Kiểm tra địa chỉ ip của người dùng có nằm trong danh sách đen hay không sau đó, lọc dữ liệu so sánh dữ liệu người dùng với list từ khóa nguy hiểm. Mỗi gói tin được gửi đi giữa các máy tính khác nhau trong mạng đều được gán địa chỉ IP và kết nối đến một dịch vụ nhất định.



##### Hình 3.1: Mô hình bộ lọc cơ sở dữ liệu

Sử dụng một Proxy có chức năng bắt các gói tin gửi đến máy chủ cơ sở dữ liệu và xử lý trước khi nó được thực hiện. Dữ liệu gói tin bắt được sẽ được kiểm tra có chứa từ khóa nguy hiểm đến sql không.

**Giai đoạn 3**: Đây là quá trình kiểm tra tính đúng đắn của câu truy vấn được gửi đến máy chủ cơ sở dữ liệu. Nếu câu truy vấn là đúng đắn nó sẽ được gửi đến máy chủ cơ sở dữ liệu để thực hiện ngược lại nó sẽ bị loại bỏ và gây ra lỗi máy chủ không hồi đáp phía máy chủ ứng dụng web.

### Cài đặt bộ lọc cơ sở dữ liệu

Bộ lọc cơ sở dữ liệu có thể xem như một plugin được cài đặt trên cùng một máy chủ vật lý với máy chủ ứng dụng web hoặc máy chủ cơ sở dữ liệu, hoặc đặt trên một máy chủ riêng biệt nằm giữa máy chủ ứng dụng web và máy chủ cơ sở dữ liệu.

Trong mô hình thử nghiệm, bộ lọc được cài đặt trên máy chủ riêng biệt với máy chủ web và máy chủ cơ sở dữ liệu. Yêu cầu khi cài đặt bao gồm:

**Đầu vào**:

* Thông tin cổng, địa chỉ IP của máy chủ cơ sở dữ liệu: Thông tin về cổng, địa chỉ ip của máy chạy bộ lọc cơ sở dữ liệu.
* File chứa danh sách các từ khóa nguy hiểm gây ra tấn công chèn mã

**Đầu ra**:

* File chứa danh sách các ip bị chặn.

Khi nhận dữ liệu đầu vào ứng dụng bắt đầu quá trình xử lý dữ liệu đầu vào.

Quá trình xử lý dữ liệu thành công, bộ lọc cơ sở dữ liệu bắt đầu hoạt động theo đúng chức năng của mình. Dữ liệu từ máy chủ ứng dụng web trong quá trình gửi sang máy chủ cơ sở dữ liệu đi qua bộ lọc cơ sở dữ liệu. Tại đây dữ liệu được kiểm tra các từ khóa gây ra tấn công XSS hoặc tấn công bước thứ hai trong tấn công chèn mã SQL. Nếu tồn tại hàm sẽ trả lại giá trị False

**Quá trình kiểm tra kết thúc trả lại giá trị True or False :**

* Nếu giá trị trả lại là False : thực hiện ghi lại thời gian, địa chỉ IP vào file blacklist.txt.
* Giá trị trả lại là True : gói tin ban đầu sẽ được gửi sang bên máy chủ cơ sở dữ liệu và thực hiện truy vấn bình thường.

### Giới thiệu mô đun proxy sử dụng cho chương trình

Mô đun proxy giữ chức năng chặn gói tin chuyển từ máy chủ ứng dụng web sang máy chủ cơ sở dữ liệu sau đó kiểm tra trước khi thực hiện truy vấn. Nó thường được sử dụng để cung cấp các dịch vụ bảo mật, kiểm soát truy cập, hoặc tạo điều kiện cho việc tương tác giữa các ứng dụng hoặc hệ thống khác nhau.

Mô đun thực hiện một số chức năng sau:

* Lưu trữ các từ khóa nguy hiểm mà nếu xuất hiện trong tên người dùng hoặc mật khẩu, sẽ được coi là đe dọa và ghi log.
* Kiểm tra xem tên người dùng hoặc mật khẩu có chứa từ khóa nguy hiểm nào không.
* Lưu trữ địa chỉ IP và thời gian hiện tại khi đăng nhập
* Kiểm tra từ khóa nguy hiểm, nếu có thì ghi vào file log.

## Kết quả triển khai

# 

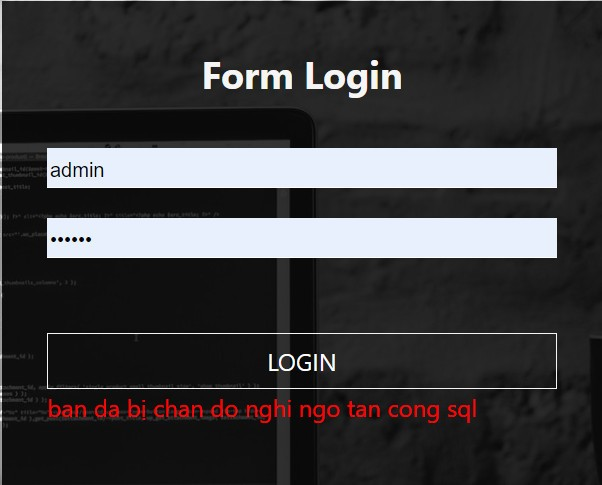
##### Hình 3.2: Giao diện đăng nhập

##### 

##### Hình 3.3: Giao diện quản lý

##### 

##### Hình 3.4: Mô phỏng tấn công chèn mã



##### Hình 3.5: Phát hiện từ khóa nguy hiểm khi đăng nhập

##### 

##### Hình 3.6: Danh sách địa chỉ ip và thời gian đăng nhập

##### 

##### 

##### Hình 3.7: Danh sách địa chỉ ip bị chặn truy cập

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết quả đạt được

* Đồ án trình bày được khái quát về tấn công chèn mã SQL, chi tiết về cơ chế và các dạng tấn công chèn mã SQL. Trong đó đã phân tích chi tiết về mục đích, phương thức và mức độ nghiêm trọng của một số kiểu tấn công chèn mã SQL gây ra cho ứng dụng web.
* Đưa ra các biện pháp phát hiện và ngăn chặn tấn công chèn mã SQL từ mức lập trình đến mức nền tảng. Tập trung nghiên cứu sâu phương pháp và mô hình phát hiện tấn công chèn mã SQL dựa trên câu trúc ngữ pháp câu truy vấn.
* Xây dựng và thử nghiệm thành công bộ lọc cơ sở dữ liệu sử dụng trong phát hiện tấn công chèn mã.

## Hạn chế

* Chỉ vừa triển khai được trên một mô hình nhỏ, chưa thể áp dụng để mở rộng mô hình.
* Chưa thể phân tích một cách chính xác được các câu truy vấn SQL.
* Chỉ mới ngăn chặn được một vài phương pháp tấn công chèn mã, chưa thể tối ưu hóa toàn bộ các biện pháp.
* Chương trình cần được tối ưu hóa để tận dụng tối đa hiệu năng của hệ thống phần cứng máy tính và mạng.

## Hướng phát triển

* Tiếp tục thử nghiệm bộ lọc trên với nhiều ứng dụng web cùng kết nối đến một máy chủ cơ sở dữ liệu. Từ đó, ta có thể tinh chỉnh, cài đặt lại ứng dụng sao cho đạt hiệu năng cao nhất.
* Sử dụng phương pháp học truy vấn cho ứng dụng để khả năng phát hiện tấn công chèn mã của ứng dụng đạt hiệu quả cao nhất.
* Nghiên cứu kết hợp ứng dụng với các phương pháp phát hiện tấn công khác nhằm nâng cao khả năng phòng thủ cho ứng dụng web.

# TÀI LIỆU THAM KHẢO

1. Atefeh Tajpour, Suhaimi Ibrahim and Maslin Masrom (2011), *SQL Injection Detection and Prevention Techniques*, International Journal of Advancements in Computing Technology Volume 3, Number 7.
2. C. Anley (2002), Advanced SQL Injection In SQL Server Applications, Next Generation Security Software.
3. J. Clarke (2009), SQL attack and Defense, Syngress Publishing, Inc.,
4. M. F. Esteban (2005), “Advanced SQL Injection in Oracle Databases,” **Black Hat Briefings**, pp. 1-30.
5. M. Howard và D. LeBlanc (2003), Writing Secure Code, tập 2, Microsoft Press,Redmond, Washington.
6. Indran Balasundaram and E. Ramaraj (2011), *An Approach to Detect and Prevent SQL Injection Attacks in Database Using Web Service*, International Journal of Computer Science and Network Security, VOL.11 No.1.