**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN 1**



**Báo Cáo Bài Tập Lớn**

**An Toàn Bảo Mật Hệ Thống Thông Tin**

**Nhóm 02 – Lớp 03**

**Đề Tài: SQL Injection**

**Giảng Viên: Ninh Thị Thu Trang**

|  |  |
| --- | --- |
| **Họ và Tên** | **Mã Sinh Viên** |
| Lê Đình Phúc (Nhóm trưởng) | B21DCCN593 |
| Nguyễn Thị Lan | B21DCCN818 |
| Ngô Tuấn Anh | B21DCCN149 |
| Mạc Văn Thành | B21DCCN677 |
| Hoàng Tiến Hưng | B21DCCN421 |

***Tháng 3 / 2024***

**Phân Chia Công Việc**

|  |  |
| --- | --- |
| **Họ Tên** | **Nhiệm Vụ** |
| Lê Đình Phúc | Vai hacker mũ cối |
| Ngô Tuấn Anh | Thuyết trình, Sếp |
| Nguyễn Thị Lan | Nhân viên |
| Mạc Văn Thành | Nhân Viên |
| Hoàng Tiến Hưng | Nhân Viên |

**SQL Injection**

[MỤC LỤC 3](#_Toc162020624)

[NỘI DUNG I: GIỚI THIỆU CHUNG 4](#_Toc162020625)

[NỘI DUNG II: KIẾN TRÚC, CÁC DẠNG TẤN CÔNG, ỨNG DỤNG 4](#_Toc162020626)

[A. KIẾN TRÚC 4](#_Toc162020627)

[B. CÁC DẠNG TẤN CÔNG VÀ CÁCH PHÒNG CHỐNG 5](#_Toc162020628)

[NỘI DUNG III: KÉT LUẬN 16](#_Toc162020629)

[NỘI DUNG IV: TÀI LIỆU THAM KHẢO 16](#_Toc162020630)

# NỘI DUNG I: GIỚI THIỆU CHUNG

Tấn công chèn mã **SQL** (**SQL Injection**) là một kỹ thuật cho phép kẻ tấn công chèn mã SQL vào dữ liệu gửi đến máy chủ và cuối cùng được thực hiện trên máy chủ cơ sở dữ liệu. Tùy vào mức độ tinh vi, tấn công chèn mã SQL có thể cho phép kẻ tấn công

(1) Vượt qua các khâu xác thực người dùng

(2) Chèn, sửa đổi, hoặc xóa dữ liệu

(3) Đánh cắp các thông tin trong cơ sở dữ liệu

(4) Chiếm quyền điều khiển hệ thống máy chủ cơ sở dữ liệu.

Tấn công chèn mã **SQL** là dạng tấn công thường gặp ở các ứng dụng web, các trang web có kết nối đến cơ sở dữ liệu.

Có ***2 nguyên nhân*** của lỗ hổng trong ứng dụng cho phép thực hiện tấn công chèn mã SQL:

* Dữ liệu đầu vào từ người dùng hoặc từ các nguồn khác không được kiểm tra hoặc kiểm tra không kỹ lưỡng;
* Sử dụng các câu lệnh SQL động trong ứng dụng, trong đó có thao tác nối dữ liệu người dùng với mã lệnh SQL gốc.

# NỘI DUNG II:

# KIẾN TRÚC, CÁC DẠNG TẤN CÔNG, ỨNG DỤNG

## KIẾN TRÚC

1. **SQL(Structured Query Language)**

Là ngôn ngữ truy vấn mang tính cấu trúc, là một loại [ngôn ngữ máy tính](https://vi.wikipedia.org/wiki/Ng%C3%B4n_ng%E1%BB%AF_m%C3%A1y) phổ biến để tạo, sửa, và lấy [dữ liệu](https://vi.wikipedia.org/wiki/D%E1%BB%AF_li%E1%BB%87u) từ một [hệ quản trị cơ sở dữ liệu quan hệ](https://vi.wikipedia.org/wiki/H%E1%BB%87_qu%E1%BA%A3n_tr%E1%BB%8B_c%C6%A1_s%E1%BB%9F_d%E1%BB%AF_li%E1%BB%87u_quan_h%E1%BB%87). Ngôn ngữ này phát triển vượt xa so với mục đích ban đầu là để phục vụ các [hệ quản trị cơ sở dữ liệu đối](https://vi.wikipedia.org/w/index.php?title=H%E1%BB%87_qu%E1%BA%A3n_tr%E1%BB%8B_c%C6%A1_s%E1%BB%9F_d%E1%BB%AF_li%E1%BB%87u_%C4%91%E1%BB%91i_t%C6%B0%E1%BB%A3ng-quan_h%E1%BB%87&action=edit&redlink=1) [tượng-quan hệ](https://vi.wikipedia.org/w/index.php?title=H%E1%BB%87_qu%E1%BA%A3n_tr%E1%BB%8B_c%C6%A1_s%E1%BB%9F_d%E1%BB%AF_li%E1%BB%87u_%C4%91%E1%BB%91i_t%C6%B0%E1%BB%A3ng-quan_h%E1%BB%87&action=edit&redlink=1). Nó là một tiêu chuẩn [ANSI](https://vi.wikipedia.org/wiki/ANSI)/[ISO](https://vi.wikipedia.org/wiki/ISO).

SQL được thừa nhận là tiêu chuẩn của [ANSI](https://vi.wikipedia.org/wiki/ANSI) (American National Standards Institute) vào năm 1986 và [ISO](https://vi.wikipedia.org/wiki/ISO) (International Organization for Standardization) năm 1987. ANSI đã công bố cách phát âm chính thức của SQL là "ess kyoo ell", nhưng rất nhiều các chuyên gia cơ sở dữ liệu nói tiếng Anh vẫn gọi nó là sequel. Một quan niệm sai khác cũng được phổ biến rộng rãi đó là "SQL" là chữ viết tắt của "Structured Query Language" (Ngôn ngữ truy vấn có cấu trúc).

**SQL Chia thành nhiều nhóm**

***Lấy dữ liệu***

Thao tác sử dụng nhiều nhất trong một cơ sở dữ liệu dựa trên giao dịch là thao tác lấy dữ liệu.

[**SELECT**](https://vi.wikipedia.org/wiki/Select_(SQL)) được sử dụng để lấy dữ liệu từ một hoặc nhiều bảng trong cơ sở dữ liệu, SELECT là lệnh thường dùng nhất của [ngôn ngữ sửa đổi dữ liệu](https://vi.wikipedia.org/w/index.php?title=Ng%C3%B4n_ng%E1%BB%AF_s%E1%BB%ADa_%C4%91%E1%BB%95i_d%E1%BB%AF_li%E1%BB%87u&action=edit&redlink=1) (tạm dịch) ([tiếng](https://vi.wikipedia.org/wiki/Ti%E1%BA%BFng_Anh) [Anh](https://vi.wikipedia.org/wiki/Ti%E1%BA%BFng_Anh): Data Manipulation Language - DML). Trong việc tạo ra câu truy vấn SELECT, người sử dụng phải đưa ra mô tả cho những dữ liệu mình muốn lấy ra chứ *không* chỉ ra những hành động vật lý nào bắt buộc phải thực hiện để lấy ra kết quả đó. Hệ thống cơ sở dữ liệu, hay chính xác hơn là [bộ tối ưu hóa câu truy vấn](https://vi.wikipedia.org/w/index.php?title=B%E1%BB%99_t%E1%BB%91i_%C6%B0u_h%C3%B3a_c%C3%A2u_truy_v%E1%BA%A5n&action=edit&redlink=1) sẽ dịch từ câu truy vấn sang [kế hoạch truy vấn](https://vi.wikipedia.org/w/index.php?title=K%E1%BA%BF_ho%E1%BA%A1ch_truy_v%E1%BA%A5n&action=edit&redlink=1) tối ưu.

Những từ khóa liên quan tới **SELECT** bao gồm:

* FROM dùng để chỉ định dữ liệu sẽ được lấy ra từ những bảng nào, và các bảng đó [quan hệ](https://vi.wikipedia.org/w/index.php?title=Quan_h%E1%BB%87_(SQL)&action=edit&redlink=1) với nhau như thế nào.
* WHERE dùng để xác định những bản ghi nào sẽ được lấy ra, hoặc áp dụng với GROUP BY.
* GROUP BY dùng để kết hợp các bản ghi có những giá trị liên quan với nhau thành các phần tử của một tập hợp nhỏ hơn các bản ghi.
* HAVING dùng để xác định những bản ghi nào, là kết quả từ từ khóa GROUP BY, sẽ được lấy ra.
* ORDER BY dùng để xác định dữ liệu lấy ra sẽ được sắp xếp theo những cột nào.

## NGUYÊN NHÂN, CÁC DẠNG TẤN CÔNG VÀ CÁCH PHÒNG CHỐNG

## I – Nguyên Nhân

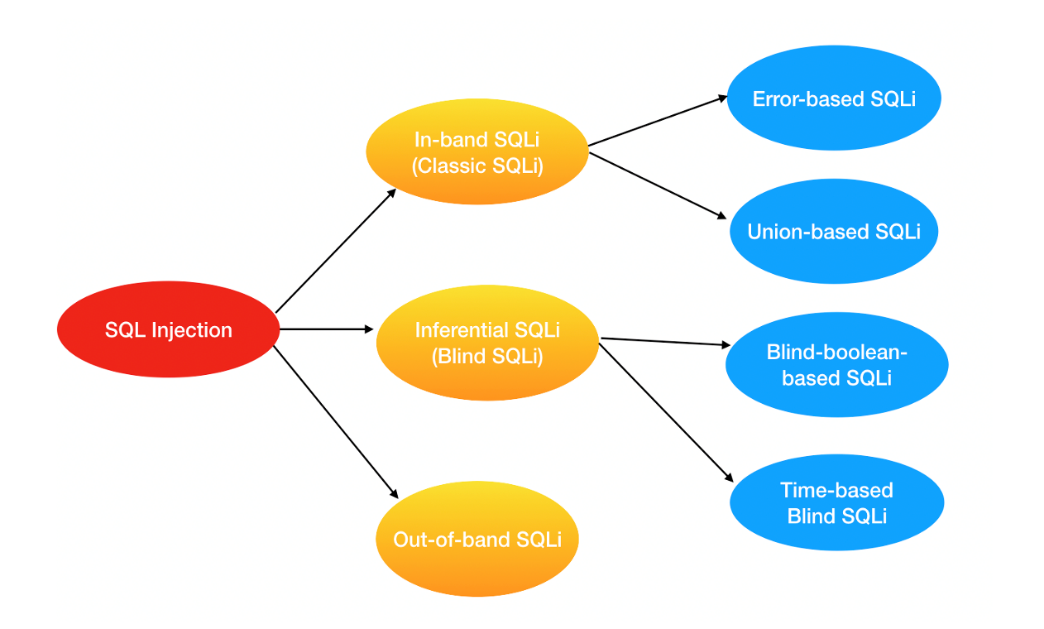
 Thiếu hiểu biết về bảo mật: Những nhà phát triển không có đủ kiến thức về bảo mật và không hiểu được nguy cơ của SQL Injection, do đó họ không áp dụng các biện pháp phòng ngừa thích hợp.

- Sử dụng các thư viện và framework không an toàn: Sử dụng các thư viện hoặc framework không an toàn có thể làm tăng nguy cơ bị tấn công SQL Injection. Các thư viện và framework này có thể không có các cơ chế bảo mật tích hợp hoặc không được cập nhật đều đặn để bảo vệ khỏi các lỗ hổng bảo mật mới.

- Sử dụng các thủ tục lưu trữ không an toàn: Các thủ tục lưu trữ (stored procedures) có thể cung cấp một cách để tấn công SQL Injection nếu chúng không được viết hoặc sử dụng một cách an toàn. Việc sử dụng các thủ tục lưu trữ không an toàn có thể tạo điều kiện cho việc tấn công SQL Injection thông qua việc truyền các tham số không an toàn.

- Thiếu kiểm tra dữ liệu đầu vào: Một số ứng dụng không thực hiện kiểm tra dữ liệu đầu vào từ người dùng hoặc các nguồn khác đối với các ký tự đặc biệt hoặc có thể là câu lệnh SQL độc hại. Điều này dẫn đến việc các ký tự này được chấp nhận mà không hề được xác thực, tạo điều kiện cho việc tấn công SQL Injection.

**II – Phân loại các kiểu tấn công SQL Injection**



**Hình 1. Phân loại các loại SQL Injection**

***SQL Injection có thể chia nhỏ thành các dạng sau***

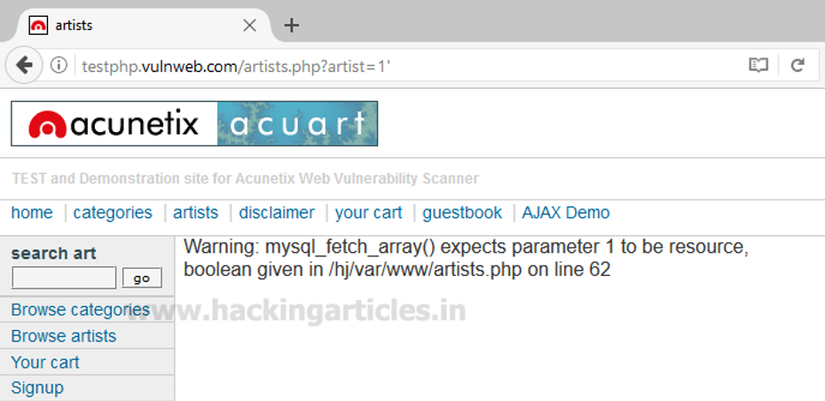
* In-band SQLi
  + Error-based SQLi
  + Union-based SQLi
* Inferential SQLi (Blind SQLi)
  + Blind-boolean-based SQLi
  + Time-based-blind SQLi
* Out-of-band SQLi

**2.1 In-band SQLi**

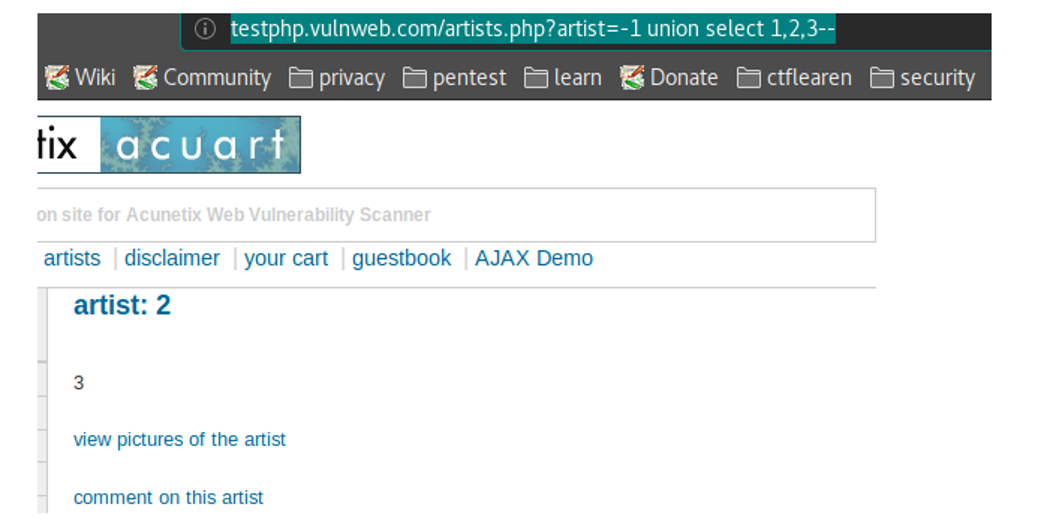
* Đây là dạng tấn công phổ biến nhất và cũng dễ để khai thác lỗ hổng SQL Injection nhất
* Xảy ra khi hacker có thể tổ chức tấn công và thu thập kết quả trực tiếp trên cùng một kênh liên lạc. Ví dụ
* In-Band SQLi chia làm 2 loại chính:
  + Error-based SQLi
  + Union-based SQLi

**2.1.1 Error-based SQLi**

* Là một kỹ thuật tấn công SQL Injection dựa vào thông báo lỗi được trả về từ Database Server có chứa thông tin về cấu trúc của cơ sở dữ liệu.
* Trong một vài trường hợp, chỉ một mình Error-based là đủ cho hacker có thể liệt kê được các thuộc tính của cơ sở dữ liệu



* + 1. **Union-based SQLi**
* Là một kỹ thuật tấn công SQL Injection dựa vào sức mạnh của toán tử UNION trong ngôn ngữ SQL cho phép tổng hợp kết quả của 2 hay nhiều câu truy vấn SELECTION trong cùng 1 kết quả và được trả về như một phần của HTTP response



* 1. **Inferential SQLi (Blind SQLi)**
* Không giống như In-band SQLi, Inferential SQL Injection tốn nhiều thời gian hơn cho việc tấn công do không có bất kì dữ liệu nào được thực sự trả về thông qua web application và hacker thì không thể theo dõi kết quả trực tiếp như kiểu tấn công In-band
* Thay vào đó, kẻ tấn công sẽ cố gắng xây dựng lại cấu trúc cơ sở dữ liệu bằng việc gửi đi các payloads, dựa vào kết quả phản hồi của web application và kết quả hành vi của database server.
* Có 2 dạng tấn công chính
  + Blind-boolean-based
  + Blind-time-based SQLi

**2.2.1 Blind-boolean-based**

* Là kĩ thuật tấn công SQL Injection dựa vào việc gửi các truy vấn tới cơ sở dữ liệu bắt buộc ứng dụng trả về các kết quả khác nhau phụ thuộc vào câu truy vấn là True hay False.
* Tuỳ thuộc kết quả trả về của câu truy vấn mà HTTP reponse có thể thay đổi, hoặc giữ nguyên
* Kiểu tấn công này thường chậm (đặc biệt với cơ sở dữ liệu có kích thước lớn) do người tấn công cần phải liệt kê từng dữ liệu, hoặc mò từng kí tự

**2.2.2 Time-based Blind SQLi**

* Time-base Blind SQLi là kĩ thuật tấn công dựa vào việc gửi những câu truy vấn tới cơ sở dữ liệu và buộc cơ sở dữ liệu phải chờ một khoảng thời gian (thường tính bằng giây) trước khi phản hồi.
* Thời gian phản hồi (ngay lập tức hay trễ theo khoảng thời gian được set) cho phép kẻ tấn công suy đoán kết quả truy vấn là TRUE hay FALSE
* Kiểu tấn công này cũng tốn nhiều thời gian tương tự như Boolean-based SQLi

**2.3 Out-of-band SQLi**

* Out-of-band SQLi không phải dạng tấn công phổ biến, chủ yếu bởi vì nó phụ thuộc vào các tính năng được bật trên Database Server được sở dụng bởi Web Application.
* Kiểu tấn công này xảy ra khi hacker không thể trực tiếp tấn công và thu thập kết quả trực tiếp trên cùng một kênh (In-band SQLi), và đặc biệt là 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.
* Ví dụ như câu lệnh xp\_dirtree trên Microsoft SQL Server có thể sử dụng để thực hiện DNS request tới một server khác do kẻ tấn công kiểm soát, hoặc Oracle Database’s UTL HTTP Package có thể sử dụng để gửi HTTP request từ SQL và PL/SQL tới server do kẻ tấn công làm chủ

**III – Những khả năng mà kẻ tấn công có thể làm khi sử dụng SQL Injection**

**1. Vượt khâu xác thực người dùng**

Xem xét một form đăng nhập với thông tin username và password và đoạn mã xử lý xác thực người dùng lưu trong bảng cơ sở dữ liệu users (username, password)

<!— Form đăng nhập -->

<form method="post" action="/log\_in.asp">

Tên đăng nhập: <input type=text name="username"><br \> Mật khẩu: <input type=password name="password"><br \>

<input type=submit name="login" value="Log In">

</form>

<%

' Mã ASP xử lý đăng nhập trong file log\_in.asp:

' giả thiết đã kết nối với CSDL SQL qua đối tượng conn và bảng tbl\_accounts lưu thông tin người dùng

Dim username, password, sqlString, rsLogin ' lấy dữ liệu từ form

username = Request.Form("username") password = Request.Form("password") ' tạo và thực hiện câu truy vấn sql

sqlString = "SELECT \* FROM tbl\_accounts WHERE username='" & username & "' AND password = '" & password & "'"

set rsLogin = conn.execute(sqlString) if (NOT rsLogin.eof()) then

' cho phép đăng nhập, bắt đầu phiên làm việc else

' từ chối đăng nhập, báo lỗi end if

%>

Giả sử người dùng nhập 'admin' vào trường username và 'abc123' vào trường password của form, mã xử lý hoạt động đúng: Nếu tồn tại người dùng với username và password kể trên, hệ thống sẽ cho phép đăng nhập với thông báo đăng nhập thành công; Nếu không tồn tại người dùng với username và password đã cung cấp, hệ thống sẽ từ chối đăng nhập và trả lại thông báo lỗi. Tuy nhiên, nếu người dùng nhập ***aaaa' OR 1=1--*** vào trường username và một chuỗi bất kỳ, chẳng hạn 'aaaa' vào trường password của form, mã xử lý hoạt động sai và chuỗi chứa câu truy vấn SQL trở thành:

***SELECT \* FROM tbl\_accounts WHERE username='aaaa' OR 1=1--' AND password='aaaa'***

Câu truy vấn sẽ trả về mọi bản ghi trong bảng do thành phần OR 1=1 làm cho điều kiện trong mệnh đề WHERE trở lên luôn đúng và phần kiểm tra mật khẩu đã bị loại bỏ bởi ký hiệu (--). Phần lệnh sau ký hiệu (--) được coi là ghi chú và không được thực hiện. Nếu trong bảng tbl\_accounts có chứa ít nhất một bản ghi, kẻ tấn công sẽ luôn đăng nhập thành công vào hệ thống.

**2. Chèn sửa đổi hoặc xóa dữ liệu**

Xem xét một form tìm kiếm sản phẩm và đoạn mã xử lý tìm sản phẩm lưu trong bảng cơ sở dữ liệu tbl\_products(product\_id, product\_name, product\_desc, product\_cost)

<!— Form tìm kiếm sản phẩm -->

<form method="post" action="/search.asp">

Nhập tên sản phẩm: <input type=text name="keyword">

<input type=submit name="search" value="Search">

</form>

<%

' Mã ASP xử lý tìm sản phẩm trong file search.asp:

' giả thiết đã kết nối với CSDL SQL server qua connection ' conn và bảng tbl\_products lưu thông tin sản phẩm

Dim keyword, sqlString, rsSearch ' lấy dữ liệu từ form

keyword = Request.Form("keyword") ' tạo và thực hiện câu truy vấn SQL

sqlString = "SELECT \* FROM tbl\_products WHERE product\_name like '%" & keyword & "%'"

set rsSearch = conn.execute(sqlString) if (NOT rsSearch.eof()) then

' hiển thị danh sách các sản phẩm else

' thông báo không tìm thấy sản phẩm end if

%>

Nếu người dùng nhập chuỗi "Samsung Galaxy S8" vào trường *keyword* của form, mã xử lý hoạt động đúng: Nếu tìm thấy các sản phẩm có tên chứa từ khóa, hệ thống sẽ hiển thị danh sách các sản phẩm tìm thấy; Nếu không tìm thấy sản phẩm nào có tên chứa từ khóa, hệ thống thông báo không tìm thấy sản phẩm. Tuy nhiên, nếu người dùng nhập chuỗi "Samsung Galaxy S8';DELETE FROM tbl\_products;--" vào trường *keyword* của form, mã xử lý sẽ hoạt động sai và chuỗi chứa câu truy vấn SQL trở thành:

SELECT \* FROM tbl\_products WHERE keyword like '%Samsung Galaxy S8';DELETE FROM tbl\_products;--%'

Chuỗi lệnh SQL mới gồm 2 lệnh SQL: câu lệnh SELECT tìm kiếm các sản phẩm có tên chứa từ khóa "Samsung Galaxy S8" trong bảng tbl\_products và câu lệnh DELETE xóa tất cả các sản phẩm trong bảng tbl\_products. Sở dĩ kẻ tấn công có thể làm được điều này là do hệ quản trị cơ sở dữ liệu MS-SQL server nói riêng và hầu hết các hệ quản trị cơ sở dữ liệu nói chung cho phép thực hiện nhiều lệnh SQL theo lô và dùng dấu ; để ngăn cách các lệnh. Ký hiệu -- dùng để hủy tác dụng của phần lệnh còn lại nếu có.

Bằng thủ thuật tương tự, kẻ tấn công có thể thay lệnh DELETE bằng lệnh UPDATE hoặc INSERT để chỉnh sửa, hoặc chèn thêm dữ liệu. Chẳng hạn, kẻ tấn công chèn thêm lệnh UPDATE để cập nhật mật khẩu của người quản trị bằng cách nhập chuỗi sau làm từ khóa tìm kiếm (giả thiết bảng tbl\_administrators chứa thông tin người quản trị):

Galaxy S8';UPDATE tbl\_administrators SET password=abc123 WHERE username = 'admin';--

Hoặc kẻ tấn công có thể chèn thêm bản ghi vào bảng tbl\_administrators bằng cách nhập chuỗi sau làm từ khóa tìm kiếm:

Galaxy S8';INSERT INTO tbl\_administrators (username, password) VALUES ('attacker', 'abc12345');--

**3. Đánh cắp thông tin trong cơ sở dữ liệu**

Lỗ hổng chèn mã SQL có thể giúp kẻ tấn công đánh cắp dữ liệu trong cơ sở dữ liệu thông qua một số bước như sau:

* + Tìm lỗ hổng chèn mã SQL và thăm dò các thông tin về hệ quản trị cơ sở dữ liệu:

+ Nhập một số dữ liệu mẫu để kiểm tra một trang web có chứa lỗ hổng chèn mã SQL, như các dấu nháy đơn, dấu --,…

+ Tìm phiên bản máy chủ cơ sở dữ liệu: nhập các câu lệnh lỗi và kiểm tra thông báo lỗi, hoặc sử dụng @@version (với MS-SQL Server), hoặc version() (với MySQL) trong câu lệnh ghép với UNION SELECT.

* + Tìm thông tin về số lượng và kiểu dữ liệu các trường của câu truy vấn hiện tại của trang web.

+ Sử dụng mệnh đề ORDER BY <số thứ tự của trường>

+ Sử dụng UNION SELECT 1, 2, 3, …

* + Trích xuất thông tin về các bảng, các trường của cơ sở dữ liệu thông qua các bảng hệ thống (metadata).
  + Sử dụng lệnh UNION SELECT để ghép các thông tin định trích xuất vào câu truy vấn hiện tại của ứng dụng.

**Ví dụ:**

- form tìm kiếm sản phẩm có lỗi chèn mã SQL với câu lệnh tìm kiếm: SELECT \* FROM tbl\_products WHERE product\_name like '%' + keyword + '%' với keyword là từ khóa người dùng cung cấp từ form.

**- Tìm thông tin về máy chủ CSDL:**

+ Sử dụng lệnh UNION [ALL] SELECT để tìm số cột trong lệnh truy vấn hiện tại: gõ chuỗi tìm kiếm: samsung%' union all select '1', '2', '3', '4' -- Thay đổi (tăng, giảm số trường) cho đến khi thấy hiển thị giá trị 1, 2,... ◊ đã tìm đúng số cột trong lệnh truy vấn hiện tại.

+ Sử dụng ORDER BY để tìm số trường samsung%' ORDER BY 5 ASC | DESC Tăng giảm số thứ tự trường để tìm số trường. Khi kết quả hiển thị và được sắp xếp đúng ◊ số trường tìm đã đúng.

+ Sử dụng @@version hoặc version() tùy theo phiên bản máy chủ CSDL đưa vào union select để lấy thông tin về máy chủ CSDL: samsung%' union select @@version, '2' –

**- Lấy thông tin về các bảng trong CSDL:**

+ Nhập chuỗi tìm kiếm: samsung' union select name, object\_id from sys.objects where type='u' -- Bảng sys.objects chứa danh sách các bảng kèm thuộc tính; 'u' là kiểu bảng do người dùng tạo; name chứa tên đối tượng (bảng) và object\_id là mã số đối tượng.

**- Lấy thông tin về các trường trong một bảng:**

+ Nhập chuỗi tìm kiếm: samsung' union select name, 0 from sys.columns where object\_id = -- trong đó lấy từ cột object\_id ở trên

**- Lấy thông tin từ một bảng đã biết tên và các trường:**

+ Nhập chuỗi tìm kiếm: samsung' union select username+'-'+password, 0 from tbl\_users -- ◊ lấy danh sách tên truy nhập và mật khẩu của tất cả các users

+ Nhập chuỗi tìm kiếm: samsung' union select username+'-'+password, 0 from tbl\_administrators -- ◊ lấy danh sách tên truy nhập và mật khẩu của tất cả các admins.

+ Tin tặc có thể đánh cắp gần như mọi thông tin trong CSDL

**4. Chiếm quyền điều khiển hệ thống**

Khả năng máy chủ cơ sở dữ liệu bị chiếm quyền điều khiển xảy ra khi trang web tồn tại đồng thời 2 lỗ hổng:

(1) lỗ hổng cho phép tấn công chèn mã SQL

(2) lỗ hổng thiết lập quyền truy nhập cơ sở dữ liệu – sử dụng người dùng có quyền quản trị để truy nhập và thao tác dữ liệu của website.

Khai thác 2 lỗ hổng này, kẻ tấn công có thể gọi thực hiện các lệnh hệ thống của máy chủ cơ sở dữ liệu cho phép can thiệp sâu vào cơ sở dữ liệu, hệ quản trị cơ sở dữ liệu và cả hệ điều hành trên máy chủ. Chẳng hạn, hệ quản trị cơ sở dữ liệu MS-SQL Server cung cấp thủ tục *sp\_send\_dbmail* cho phép gửi email từ máy chủ cơ sở dữ liệu và thủ tục *xp\_cmdshell* cho phép chạy các lệnh và chương trình cài đặt trên hệ điều hành Microsoft Windows. Sau đây là một số ví dụ chạy các lệnh Microsoft Windows thông qua thủ tục *xp\_cmdshell*:

EXEC xp\_cmdshell 'dir \*.exe' : liệt kê nội dung thư mục hiện thời

EXEC xp\_cmdshell 'shutdown /s /t 00' : tắt máy chủ nền chạy hệ quản trị CSDL EXEC xp\_cmdshell 'net stop W3SVC' : dừng hoạt động máy chủ web

EXEC xp\_cmdshell 'net stop MSSQLSERVER' : dừng hoạt động máy chủ CSDL.

Ngoài ra, kẻ tấn công có thể thực hiện các thao tác nguy hiểm đến cơ sở dữ liệu nếu

có quyền của người quản trị cơ sở dữ liệu hoặc quản trị hệ thống, như:

* Xóa cả bảng (gồm cả cấu trúc): DROP TABLE <tên bảng> Xóa cả cơ sở dữ liệu: DROP DATABASE <tên CSDL>
* Tạo 1 tài khoản mới truy nhập CSDL: sp\_addlogin <username> <password>
* Đổi mật khẩu tài khoản truy nhập CSDL: sp\_password <password>

**5. Mức độ thiệt hại**

Thực chất, nguồn gốc của quá trình tấn công SQL Injection là sự cẩu thả trong quy trình mã hóa ứng dụng, mật khẩu ứng dụng nhưng vẫn có thể ngăn chặn được, tuy nhiên mức độ thiệt hại là không thể lường trước được và phụ thuộc vào quy mô của từng database. Trong quá trình một ứng dụng web bất kỳ có thể giao tiếp với backen database thì sẽ phải cung cấp thông tin đăng nhập tới một cơ sở dữ liệu (quy trình này khác hẳn so với lúc người dùng tiến hành đăng nhập vào website hay form login). Phụ thuộc vào mức phân quyền tương ứng mà ứng dụng web yêu cầu, tài khoản trong cơ sở dữ liệu có thể sử dụng được bất kỳ quyền chỉnh sửa nào, từ việc đơn giản như đọc, ghi đơn thuần trên những bảng có sẵn cho tới mức quyền đầy đủ.

Chúng ta thấy bằng việc nhập tên tài khoản (ví dụ “youruser”, “admin’--” hoặc bất kỳ) thì hoàn toàn có thể đăng nhập trực tiếp vào database mà không cần biết **password.** Khi ở bên trong hệ thống sẽ không thể biết được tài khoản đó bình thường hoặc có đầy đủ quyền tương ứng. Về mặt bản chất, việc phân quyền của database không cung cấp quyền chính xác tương ứng và an toàn trong quá trình này, bởi vì thông thường một website ít nhất phải có quyền đọc hoặc ghi tới cơ sở dữ liệu tương ứng.

Giải sử rằng hệ thống website của chúng ta có đầy đủ quyền truy cập, bao gồm việc xóa dữ liệu, thêm hoặc xóa bảng, tạo mới tài khoản… và trên thực tế, có khá nhiều chương trình nếu muốn cài đặt và sử dụng thì phải được phân quyền ở mức cao nhất, do vậy phải cẩn thận khi thực hiện việc gán quyền.

**6. Điểm mạnh/yếu của SQL Injection**

Mặc dù SQL Injection là một trong những lỗi bảo mật khá phổ biến và có thể nguy hiểm, tuy nhiên, việc khắc phục nó không hề đơn giản.

- Đa dạng các cách thức tấn công: SQL Injection có thể được thực hiện qua nhiều phương thức khác nhau, từ các trường nhập liệu trên giao diện người dùng đến các thủ thuật nâng cao như bypassing authentication hoặc thực thi lệnh từ xa. Việc phát hiện và ngăn chặn tất cả các hình thức tấn công này đòi hỏi kiến thức chuyên sâu và kỹ năng kỹ thuật cao.

- Phạm vi ảnh hưởng lớn: SQL Injection có thể ảnh hưởng đến hệ thống cơ sở dữ liệu và ứng dụng web một cách nghiêm trọng, từ việc lấy thông tin nhạy cảm đến việc thực thi các lệnh nguy hiểm như xóa, cập nhật hoặc thậm chí là kiểm soát toàn bộ hệ thống.

- Khó khăn trong việc phát hiện: Việc phát hiện SQL Injection không phải lúc nào cũng đơn giản, đặc biệt là đối với các ứng dụng lớn và phức tạp. Các tấn công có thể được thực hiện một cách rất tinh vi, thậm chí là không để lại dấu vết.

- Khả năng tương thích ngược: Một số ứng dụng có thể không tương thích hoặc không thể sửa đổi một cách dễ dàng để ngăn chặn SQL Injection, đặc biệt là các ứng dụng lưu trữ trên nền tảng cũ hoặc không được bảo trì thường xuyên.

- Sự phát triển liên tục của kỹ thuật tấn công: Các hacker liên tục phát triển các kỹ thuật tấn công mới để vượt qua các biện pháp bảo mật. Điều này đòi hỏi các nhà phát triển và quản trị hệ thống phải duy trì kiến thức và cập nhật kỹ năng để đối phó với các mối đe dọa mới.

**7. Phòng Chống**

Do tính chất nguy hiểm của tấn công chèn mã SQL, nhiều giải pháp đã được đề xuất nhằm hạn chế tác hại và ngăn chặn triệt để dạng tấn công này. Nhìn chung, cần áp dụng kết hợp các biện pháp phòng chống tấn công chèn mã SQL để đảm bảo an toàn cho hệ thống. Các biện pháp, kỹ thuật cụ thể có thể áp dụng gồm:

1. **Các biện pháp phòng chống dựa trên kiểm tra và lọc dữ liệu đầu vào:**

+ Kiểm tra tất cả các dữ liệu đầu vào, đặc biệt dữ liệu nhập từ người dùng và từ các nguồn không tin cậy;

+ Kiểm tra kích thước và định dạng dữ liệu đầu vào;

+ Tạo các bộ lọc để lọc bỏ các ký tự đặc biệt (như \*, ‘, =, --) và các từ khóa của ngôn ngữ SQL (SELECT, INSERT, UPDATE, DELETE, DROP,. ) mà kẻ tấn

công có thể sử dụng:

1. **Sử dụng thủ tục cơ sở dữ liệu (stored procedures) và cơ chế tham số hóa dữ liệu:**

+ Đưa tất cả các câu truy vấn (SELECT) và cập nhật, sửa, xóa dữ liệu (INSERT, UPDATE, DELETE) vào các thủ tục. Dữ liệu truyền vào thủ tục thông qua các tham số, giúp tách dữ liệu khỏi mã lệnh SQL, nhờ đó hạn ngăn chặn hiệu quả tấn công chèn mã SQL;

+ Hạn chế thực hiện các câu lệnh SQL động trong thủ tục;

+ Sử dụng cơ chế tham số hóa dữ liệu hỗ trợ bởi nhiều ngôn ngữ lập trình web như ASP.NET, PHP và JSP.

1. **Các biện pháp phòng chống dựa trên thiết lập quyền truy nhập người dùng cơ sở dữ liệu:**

+ Không sử dụng người dùng có quyền quản trị hệ thống hoặc quản trị cơ sở dữ liệu làm người dùng truy nhập dữ liệu. Ví dụ: không dùng người dùng *sa* (Microsoft SQL) hoặc *root* (MySQL) làm người dùng truy nhập dữ liệu. Chỉ dùng các người dùng này cho mục đích quản trị.

+ Chia nhóm người dùng, chỉ cấp quyền vừa đủ để truy nhập các bảng biểu, thực hiện câu truy vấn và chạy các thủ tục.

+ Tốt nhất, không cấp quyền thực hiện các câu truy vấn, cập nhật, sửa, xóa trực tiếp trên các bảng dữ liệu. Thủ tục hóa tất cả các câu lệnh và chỉ cấp quyền thực hiện thủ tục.

+ Cấm hoặc vô hiệu hóa (disable) việc thực hiện các thủ tục hệ thống (các thủ tục cơ sở dữ liệu có sẵn) cho phép can thiệp vào hệ quản trị cơ sở dữ liệu và hệ điều hành nền.

1. Sử dụng các công cụ rà quét lỗ hổng chèn mã SQL, như SQLMap, hoặc Acunetix Vulnerability Scanner để chủ động rà quét, tìm các lỗ hổng chèn mã SQL và có biện pháp khắc phục phù hợp.

* SQLmap là một công cụ mã mở miễn phí viết bằng Python:
  + Cho phép kiểm tra website tìm lỗi chèn mã SQL
  + Cho phép khai thác lỗi để điều khiển máy chủ CSDL
  + Hỗ trợ hầu hết các máy chủ quản trị CSDL hiện nay: MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase và SAP MaxDB.

# NỘI DUNG III: KÉT LUẬN

Báo Cáo trên đã tìm hiểu về cách hoạt động cũng như các kiến thức liên quan đến SQL Injection, mọi nôi dung được tóm tắt sau đây:

NỘI DUNG I: Giới thiệu sơ lược về SQL Injection

NỘI DUNG II:

1. Giới thiệu về ngôn ngữ SQL
2. Các dạng tấn công phổ biến trong SQL
3. Những khả năng kẻ tấn công có thể làm
4. Mức độ thiệt hại
5. Cách phòng chống

# NỘI DUNG IV: TÀI LIỆU THAM KHẢO

1. [Các Kiểu Tấn Công Thường Gặp](https://viblo.asia/p/sql-injection-MgNeWWbKeYx%23_ii-cac-kieu-tan-cong-thuong-gap-1)
2. [Các Hình Thức Tấn Công](https://comlink.vn/tan-cong-sql-injection/)
3. [Phân Loại Kiểu Tấn Công Trong SQL Injection Và cách phòng chống](https://vietnix.vn/sql-injection-la-gi/%23in-band-sqli)
4. Giáo Trình, Slide Bài Giảng