**ÔN TẬP**

**TUẦN 5 – SQLi**

**Câu 1: SQL Injection là:**

**a. Hình thức tấn công bằng cách chèn các câu lệnh SQL vào đầu vào của ứng dụng**

b. Tấn công qua mạng bằng cách sử dụng các script độc hại

c. Lỗ hổng trong việc xác thực người dùng

d. Kỹ thuật mã hóa dữ liệu

**Câu 2: Ghép các biện pháp phòng chống SQL Injection với giải thích tương ứng.**

**a. Stored Procedures → Thực hiện các câu truy vấn SQL trên cơ sở dữ liệu từ server mà không phụ thuộc vào dữ liệu người dùng, Input Validation → Kiểm tra dữ liệu từ người dùng để đảm bảo rằng dữ liệu an toàn và đúng định dạng, Escaping User Input → Xử lý các ký tự đặc biệt trong dữ liệu đầu vào để ngăn chặn sự can thiệp của SQL Injection, Parameterized Queries → Sử dụng các tham số để đảm bảo rằng dữ liệu người dùng không làm thay đổi cấu trúc của câu truy vấn**

b. Stored Procedures → Thực hiện các câu truy vấn SQL trên cơ sở dữ liệu từ client mà không phụ thuộc vào dữ liệu người dùng, Input Validation → Kiểm tra dữ liệu từ người dùng để đảm bảo rằng dữ liệu an toàn và đúng định dạng, Escaping User Input → Xử lý các ký tự đặc biệt trong dữ liệu đầu vào để ngăn chặn sự can thiệp của SQL Injection, Parameterized Queries → Sử dụng các tham số để đảm bảo rằng dữ liệu người dùng không làm thay đổi cấu trúc của câu truy vấn

c. Stored Procedures → Thực hiện các câu truy vấn SQL trên cơ sở dữ liệu từ server mà không phụ thuộc vào dữ liệu người dùng, Input Validation → Kiểm tra dữ liệu từ người dùng để đảm bảo rằng dữ liệu an toàn và đúng định dạng, Escaping User Input → Xử lý các ký tự đặc biệt trong dữ liệu đầu vào để hỗ trợ cho sự can thiệp của SQL Injection, Parameterized Queries → Sử dụng các tham số để đảm bảo rằng dữ liệu người dùng không làm thay đổi cấu trúc của câu truy vấn

d. Stored Procedures → Thực hiện các câu truy vấn SQL trên cơ sở dữ liệu từ server phụ thuộc vào dữ liệu người dùng, Input Validation → Kiểm tra dữ liệu từ người dùng để đảm bảo rằng dữ liệu an toàn và đúng định dạng, Escaping User Input → Xử lý các ký tự đặc biệt trong dữ liệu đầu vào để ngăn chặn sự can thiệp của SQL Injection, Parameterized Queries → Sử dụng các tham số để đảm bảo rằng dữ liệu người dùng không làm thay đổi cấu trúc của câu truy vấn

**Câu 3: Ví dụ nào mô tả điều kiện luôn đúng trong SQL Injection?**

**a. ' OR '1' = '1'**

b. ' OR 'a' = 'b'

c. admin' --

d. ' AND '1' = '1'

**Câu 4: Ghép các biện pháp ngăn chặn tấn công SQL Injection với lợi ích tương ứng.**

**a. Kiểm tra dữ liệu đầu vào → Ngăn chặn dữ liệu không hợp lệ hoặc nguy hiểm, Escape ký tự đặc biệt → Xử lý các ký tự như dấu nháy đơn để tránh bị tấn công, Sử dụng Prepared Statements → Truy vấn SQL an toàn và không phụ thuộc vào dữ liệu người dùng**

b. Kiểm tra dữ liệu đầu vào → Ngăn chặn dữ liệu không hợp lệ hoặc nguy hiểm, Escape ký tự đặc biệt → Xử lý các ký tự như dấu nháy đơn để tránh bị tấn công, Sử dụng Prepared Statements → Truy vấn SQL an toàn và phụ thuộc vào dữ liệu người dùng

c. Kiểm tra dữ liệu đầu vào → Cho phép dữ liệu không hợp lệ hoặc nguy hiểm, Escape ký tự đặc biệt → Xử lý các ký tự như dấu nháy đơn để tránh bị tấn công, Sử dụng Prepared Statements → Truy vấn SQL an toàn và không phụ thuộc vào dữ liệu người dùng

d. Kiểm tra dữ liệu đầu vào → Ngăn chặn dữ liệu không hợp lệ hoặc nguy hiểm, Encrypt ký tự đặc biệt bằng AES → Xử lý các ký tự như dấu nháy đơn để tránh bị tấn công, Sử dụng Prepared Statements → Truy vấn SQL an toàn và không phụ thuộc vào dữ liệu người dùng

**Câu 5: Ghép cặp các loại tấn công SQL Injection với mô tả tương ứng.**

**a. Blind SQL Injection → Tấn công SQL Injection mà không hiển thị lỗi, Union SQL Injection → Kết hợp kết quả của nhiều truy vấn SELECT, PiggyBacked Query → Chèn thêm truy vấn SQL vào câu truy vấn chính, Tautology → Sử dụng điều kiện luôn đúng như '1=1' trong câu truy vấn**

b. Blind SQL Injection → Tấn công SQL Injection mà không hiển thị lỗi, Union SQL Injection → Giao kết quả của nhiều truy vấn SELECT, PiggyBacked Query → Chèn thêm truy vấn SQL vào câu truy vấn chính, Tautology → Sử dụng điều kiện luôn đúng như '1=1' trong câu truy vấn

c. Blind SQL Injection → Tấn công SQL Injection mà không hiển thị lỗi, Union SQL Injection → Kết hợp kết quả của nhiều truy vấn SELECT, PiggyBacked Query → Chèn thêm truy vấn SQL vào câu truy vấn chính, Tautology → Sử dụng điều kiện luôn đúng như '1!=1' trong câu truy vấn

d. Blind SQL Injection → Tấn công SQL Injection mà không hiển thị lỗi, Union SQL Injection → Kết hợp kết quả của nhiều truy vấn SELECT, PiggyBacked Query → Chèn thêm truy vấn SQL vào câu truy vấn chính, Tautology → Sử dụng điều kiện luôn đúng như '1=1' trong câu truy vấn

**Câu 6: Biện pháp nào sau đây không giúp ngăn chặn SQL Injection?**

**a. Sử dụng các truy vấn SQL động với dữ liệu người dùng không kiểm tra**

b. Sử dụng các Prepared Statements

c. Escape dữ liệu đầu vào

d. Kiểm tra và làm sạch dữ liệu đầu vào

**Câu 7: Union SQL Injection hoạt động như thế nào?**

**a. Kết hợp kết quả của nhiều truy vấn SELECT để lấy thêm dữ liệu**

b. Chèn thêm các câu truy vấn SQL sau truy vấn chính

c. Sử dụng điều kiện luôn đúng trong câu lệnh WHERE

d. Thay đổi cấu trúc của câu lệnh SQL mà không hiển thị lỗi

**Câu 8: Điều gì sẽ xảy ra nếu số cột của câu truy vấn Union không khớp?**

**a. Câu truy vấn sẽ trả về lỗi**

b. Truy vấn sẽ hoàn thành nhưng không có dữ liệu được trả về

c. Truy vấn sẽ xóa tất cả các bảng trong cơ sở dữ liệu

d. Truy vấn sẽ tự động điều chỉnh số cột

**Câu 9: Blind SQL Injection là:**

**a. Khi tấn công SQL Injection mà không nhận được thông báo lỗi từ server**

b. Khi kết quả của tấn công SQL Injection được hiển thị ngay lập tức

c. Khi attacker sử dụng điều kiện luôn đúng để tấn công

d. Khi attacker kết hợp nhiều truy vấn SQL cùng một lúc

**Câu 10: Ghép các phương thức HTTP với khả năng khai thác SQL Injection tương ứng.**

**a. GET → SQL Injection có thể xảy ra nếu dữ liệu được gửi qua URL, POST → SQL Injection có thể xảy ra nếu dữ liệu từ form không được kiểm tra, DELETE → SQL Injection có thể xảy ra nếu truy vấn xóa không được bảo vệ, PUT → SQL Injection có thể xảy ra khi cập nhật dữ liệu vào cơ sở dữ liệu**

b. GET → SQL Injection có thể xảy ra nếu dữ liệu được gửi qua URI, POST → SQL Injection có thể xảy ra nếu dữ liệu từ form không được kiểm tra, DELETE → SQL Injection có thể xảy ra nếu truy vấn xóa không được bảo vệ, PUT → SQL Injection có thể xảy ra khi cập nhật dữ liệu vào cơ sở dữ liệu

c. GET → SQL Injection có thể xảy ra nếu dữ liệu được gửi qua URL, PUSH → SQL Injection có thể xảy ra nếu dữ liệu từ form không được kiểm tra, DELETE → SQL Injection có thể xảy ra nếu truy vấn xóa không được bảo vệ, PUT → SQL Injection có thể xảy ra khi cập nhật dữ liệu vào cơ sở dữ liệu

d. GET → SQL Injection có thể xảy ra nếu dữ liệu được gửi qua URL, POST → SQL Injection có thể xảy ra nếu dữ liệu từ form không được kiểm tra, DELETE → SQL Injection có thể xảy ra nếu truy vấn xóa không được bảo vệ, PULL → SQL Injection có thể xảy ra khi cập nhật dữ liệu vào cơ sở dữ liệu

**Câu 11: SQL Injection có thể gây ra những tác hại nào?**

**a. Lấy cắp thông tin, chèn thêm dữ liệu, hoặc thực hiện các lệnh của hệ điều hành**

b. Giảm tốc độ của hệ thống mạng

c. Ngăn chặn việc truy cập vào website

d. Tạo các kết nối trái phép đến mạng nội bộ

**Câu 12: Loại tấn công SQL Injection nào mà kẻ tấn công không thể nhìn thấy kết quả trực tiếp của truy vấn?**

**a. Blind SQL Injection**

b. Union SQL Injection

c. PiggyBacked Query

d. Tautology

**Câu 13: Loại SQL Injection nào cho phép kẻ tấn công chèn thêm các câu truy vấn SQL vào câu truy vấn chính?**

**a. PiggyBacked Query**

b. Union SQL Injection

c. Blind SQL Injection

d. Stored SQL Injection

**Câu 14: Tấn công SQL Injection có thể thực hiện trên cơ sở dữ liệu nào sau đây?**

**a. MySQL, Oracle, PostgreSQL và bất kỳ cơ sở dữ liệu nào hỗ trợ SQL**

b. Chỉ trên MySQL

c. Chỉ trên Oracle

d. Chỉ trên cơ sở dữ liệu mã nguồn mở

**Câu 15: Làm thế nào để Blind SQL Injection kiểm tra giá trị TRUE/FALSE?**

**a. Sử dụng các điều kiện "AND 1=1" (TRUE) và "AND 1=2" (FALSE) trong câu lệnh SQL**

b. Bằng cách gửi hàng loạt truy vấn SELECT khác nhau

c. Thực hiện lặp qua các bảng dữ liệu

d. Chèn lệnh DROP TABLE để kiểm tra phản hồi

**Câu 16: Loại SQL Injection nào sử dụng điều kiện luôn đúng để thay đổi kết quả của truy vấn SQL?**

**a. Tautology**

b. Union SQL Injection

c. PiggyBacked Query

d. Blind SQL Injection

**Câu 17: Ghép các loại tấn công SQL Injection với ví dụ tương ứng.**

**a. Union SQL Injection → SELECT \* FROM Users WHERE Name = 'admin' UNION SELECT NULL, Password FROM Users, Tautology → SELECT \* FROM Users WHERE Name = 'admin' OR '1' = '1', PiggyBacked Query → SELECT \* FROM Users; DROP TABLE Users --, Blind SQL Injection → SELECT \* FROM Users WHERE ID = 1 AND 1=1**

b. **Tautology** → SELECT \* FROM Users WHERE Name = 'admin' UNION SELECT NULL, Password FROM Users, **Union SQL Injection** → SELECT \* FROM Users WHERE Name = 'admin' OR '1' = '1', **PiggyBacked Query** → SELECT \* FROM Users; DROP TABLE Users --, **Blind SQL Injection** → SELECT \* FROM Users WHERE ID = 1 AND 1=1

c. **Union SQL Injection** → SELECT \* FROM Users WHERE Name = 'admin' UNION SELECT NULL, Password FROM Users, **Tautology** → SELECT \* FROM Users WHERE Name = 'admin' OR '1' = '1', **Blind SQL Injection** → SELECT \* FROM Users; DROP TABLE Users --, **PiggyBacked Query** → SELECT \* FROM Users WHERE ID = 1 AND 1=1

d. **Blind SQL Injection** → SELECT \* FROM Users WHERE Name = 'admin' UNION SELECT NULL, Password FROM Users, **Tautology** → SELECT \* FROM Users WHERE Name = 'admin' OR '1' = '1', **PiggyBacked Query** → SELECT \* FROM Users; DROP TABLE Users --, **Union SQL Injection** → SELECT \* FROM Users WHERE ID = 1 AND 1=1

**Câu 18: Ghép các kiểu tấn công SQL Injection với mục tiêu tương ứng của chúng.**

**a. Boolean-based Blind SQL Injection → Xác định giá trị đúng/sai qua phản hồi của truy vấn, Error-based SQL Injection → Tạo ra thông báo lỗi để lấy thông tin từ cơ sở dữ liệu, Time-based Blind SQL Injection → Xác định sự tồn tại của dữ liệu thông qua độ trễ của phản hồi, Union-based SQL Injection → Truy xuất dữ liệu từ nhiều bảng khác nhau**

b. Boolean-based Blind SQL Injection → Xác định giá trị đúng/sai qua phản hồi của truy vấn, Error-based SQL Injection → Tạo ra thông báo lỗi để lấy thông tin từ cơ sở dữ liệu, Time-based Blind SQL Injection → Xác định sự tồn tại của dữ liệu thông qua độ trễ của phản hồi, Union-based SQL Injection → Truy xuất dữ liệu từ nhiều bảng giống nhau

c. Error-based Blind SQL Injection → Xác định giá trị đúng/sai qua phản hồi của truy vấn, Boolean-based SQL Injection → Tạo ra thông báo lỗi để lấy thông tin từ cơ sở dữ liệu, Time-based Blind SQL Injection → Xác định sự tồn tại của dữ liệu thông qua độ trễ của phản hồi, Union-based SQL Injection → Truy xuất dữ liệu từ nhiều bảng khác nhau

d. Union-based SQL Injection → Xác định giá trị đúng/sai qua phản hồi của truy vấn, Error-based SQL Injection → Tạo ra thông báo lỗi để lấy thông tin từ cơ sở dữ liệu, Time-based Blind SQL Injection → Xác định sự tồn tại của dữ liệu thông qua độ trễ của phản hồi, Boolean-based Blind SQL Injection → Truy xuất dữ liệu từ nhiều bảng giống nhau

**Câu 19: Ghép các lỗi thường gặp trong bảo mật ứng dụng với mô tả tương ứng.**

**a. Sử dụng truy vấn SQL động → Truy vấn được tạo bằng cách ghép chuỗi mà không có kiểm tra, Không sử dụng các kỹ thuật bảo mật → Ứng dụng không sử dụng các công cụ như Prepared Statements, Không kiểm tra đầu vào → Dữ liệu từ người dùng được đưa thẳng vào câu truy vấn SQL**

b. Sử dụng truy vấn SQL tĩnh → Truy vấn được tạo bằng cách ghép chuỗi mà không có kiểm tra, Không sử dụng các kỹ thuật bảo mật → Ứng dụng không sử dụng các công cụ như Prepared Statements, Không kiểm tra đầu vào → Dữ liệu từ người dùng được đưa thẳng vào câu truy vấn SQL

c. Sử dụng truy vấn SQL động → Truy vấn được tạo bằng cách ghép chuỗi mà không có kiểm tra, Không sử dụng các kỹ thuật bảo mật → Ứng dụng không sử dụng các công cụ như Prepared Statements, Kiểm tra đầu vào → Dữ liệu từ người dùng được đưa thẳng vào câu truy vấn SQL

**Câu 20: Ghép cặp các bước khai thác SQL Injection với mục tiêu tương ứng.**

**a. Bước 1: Xác định lỗ hổng → Kiểm tra lỗi bằng cách thêm dấu ' vào URL hoặc trường đầu vào, Bước 2: Xác định số cột → Sử dụng ORDER BY để xác định số cột trong bảng, Bước 3: Xác định cột hiển thị → Dùng UNION SELECT để kiểm tra cột hiển thị trên trình duyệt**

b. Bước 2: Xác định lỗ hổng → Kiểm tra lỗi bằng cách thêm dấu ' vào URL hoặc trường đầu vào, Bước 1: Xác định số cột → Sử dụng ORDER BY để xác định số cột trong bảng, Bước 3: Xác định cột hiển thị → Dùng UNION SELECT để kiểm tra cột hiển thị trên trình duyệt

c. Bước 3: Xác định lỗ hổng → Kiểm tra lỗi bằng cách thêm dấu ' vào URL hoặc trường đầu vào, Bước 2: Xác định số cột → Sử dụng ORDER BY để xác định số cột trong bảng, Bước 1: Xác định cột hiển thị → Dùng UNION SELECT để kiểm tra cột hiển thị trên trình duyệt

**Câu 21: Ghép các bước phát hiện tấn công SQL Injection với mô tả tương ứng.**

**a. Bước 2: Kiểm tra lỗi trong cơ sở dữ liệu → Kiểm tra các thông báo lỗi hoặc kết quả bất thường từ cơ sở dữ liệu, Bước 4: Sử dụng công cụ phát hiện tự động → Sử dụng các công cụ như SQLMap để phát hiện và thử nghiệm SQL Injection, Bước 1: Phân tích đầu vào → Kiểm tra các trường đầu vào để phát hiện các ký tự bất thường, Bước 3: Kiểm tra truy vấn SQL → Phân tích truy vấn SQL để phát hiện các câu truy vấn không mong muốn**

b. Bước 1: Kiểm tra lỗi trong cơ sở dữ liệu → Kiểm tra các thông báo lỗi hoặc kết quả bất thường từ cơ sở dữ liệu, Bước 4: Sử dụng công cụ phát hiện tự động → Sử dụng các công cụ như SQLMap để phát hiện và thử nghiệm SQL Injection, Bước 2: Phân tích đầu vào → Kiểm tra các trường đầu vào để phát hiện các ký tự bất thường, Bước 3: Kiểm tra truy vấn SQL → Phân tích truy vấn SQL để phát hiện các câu truy vấn không mong muốn

c. Bước 4: Kiểm tra lỗi trong cơ sở dữ liệu → Kiểm tra các thông báo lỗi hoặc kết quả bất thường từ cơ sở dữ liệu, Bước 2: Sử dụng công cụ phát hiện tự động → Sử dụng các công cụ như SQLMap để phát hiện và thử nghiệm SQL Injection, Bước 1: Phân tích đầu vào → Kiểm tra các trường đầu vào để phát hiện các ký tự bất thường, Bước 3: Kiểm tra truy vấn SQL → Phân tích truy vấn SQL để phát hiện các câu truy vấn không mong muốn

d. Bước 2: Kiểm tra lỗi trong cơ sở dữ liệu → Kiểm tra các thông báo lỗi hoặc kết quả bất thường từ cơ sở dữ liệu, Bước 4: Sử dụng công cụ phát hiện tự động → Sử dụng các công cụ như SQLMap để phát hiện và thử nghiệm SQL Injection, Bước 3: Phân tích đầu vào → Kiểm tra các trường đầu vào để phát hiện các ký tự bất thường, Bước 1: Kiểm tra truy vấn SQL → Phân tích truy vấn SQL để phát hiện các câu truy vấn không mong muốn

**Câu 22: Ghép các biện pháp bảo vệ với giải thích tương ứng.**

**a. Stored Procedures → Truy vấn được lưu trữ trên server và thực thi từ phía server, Input Validation → Kiểm tra dữ liệu từ người dùng để đảm bảo đúng định dạng và an toàn, Prepared Statements → Các tham số đầu vào được truyền riêng và không ảnh hưởng đến câu truy vấn**

b. Prepared Statements → Truy vấn được lưu trữ trên server và thực thi từ phía server, Input Validation → Kiểm tra dữ liệu từ người dùng để đảm bảo đúng định dạng và an toàn, Stored Procedures → Các tham số đầu vào được truyền riêng và không ảnh hưởng đến câu truy vấn

c. Input Validation → Truy vấn được lưu trữ trên server và thực thi từ phía server, Stored Procedures → Kiểm tra dữ liệu từ người dùng để đảm bảo đúng định dạng và an toàn, Prepared Statements → Các tham số đầu vào được truyền riêng và không ảnh hưởng đến câu truy vấn

**Câu 23: Cách nào giúp xác định các cột có thể sử dụng trong Union SQL Injection?**

**a. Sử dụng lệnh UNION SELECT NULL,...NULL và tăng dần số lượng NULL**

b. Sử dụng câu lệnh DROP TABLE

c. Thực hiện các truy vấn INSERT nhiều lần

d. Sử dụng các tham số đầu vào động

**Câu 24: Biện pháp tốt nhất để chống lại SQL Injection là gì?**

**a. Sử dụng Prepared Statements với tham số hóa**

b. Sử dụng truy vấn SQL động với dữ liệu người dùng

c. Giới hạn truy cập vào cơ sở dữ liệu

d. Lưu trữ mật khẩu dưới dạng văn bản thuần túy

**Câu 25: Biện pháp nào giúp bảo vệ ứng dụng khỏi tấn công SQL Injection?**

**a. Kiểm tra và lọc dữ liệu đầu vào**

b. Giới hạn truy cập của người dùng

c. Mã hóa mật khẩu của người dùng

d. Sử dụng session cookies an toàn

**Câu 26: Biện pháp nào sau đây giúp ngăn chặn SQL Injection?**

**a. Sử dụng Prepared Statements**

b. Mã hóa toàn bộ cơ sở dữ liệu

c. Sử dụng cookie an toàn

d. Giới hạn số lần đăng nhập không thành công

**Câu 27: Nguyên nhân chính gây ra SQL Injection là:**

**a. Dữ liệu đầu vào không được kiểm tra hoặc làm sạch**

b. Sử dụng giao thức HTTP thay vì HTTPS

c. Định dạng sai dữ liệu trong cơ sở dữ liệu

d. Lỗi xác thực mật khẩu người dùng

**Câu 28: Ghép cặp các biện pháp ngăn chặn SQL Injection với mô tả tương ứng.**

**a. Escape dữ liệu → Xử lý các ký tự đặc biệt trong đầu vào người dùng, Sử dụng Prepared Statements → Các giá trị đầu vào được truyền dưới dạng tham số, không trực tiếp vào câu truy vấn, Kiểm tra dữ liệu đầu vào → Xác thực dữ liệu người dùng nhập vào để đảm bảo đúng kiểu dữ liệu**

b. Encrypt dữ liệu → Xử lý các ký tự đặc biệt trong đầu vào người dùng, Sử dụng Prepared Statements → Các giá trị đầu vào được truyền dưới dạng tham số, không trực tiếp vào câu truy vấn, Kiểm tra dữ liệu đầu vào → Xác thực dữ liệu người dùng nhập vào để đảm bảo đúng kiểu dữ liệu

c. Escape dữ liệu → Xử lý các ký tự đặc biệt trong đầu vào người dùng, Sử dụng Prepared Statements → Các giá trị đầu vào được truyền dưới dạng Base64, không trực tiếp vào câu truy vấn, Kiểm tra dữ liệu đầu vào → Xác thực dữ liệu người dùng nhập vào để đảm bảo đúng kiểu dữ liệu

d. Encrypt dữ liệu → Xử lý các ký tự đặc biệt trong đầu vào người dùng bằng AES-GCM-256, Sử dụng Prepared Statements → Các giá trị đầu vào được truyền dưới dạng tham số, không trực tiếp vào câu truy vấn, Kiểm tra dữ liệu đầu vào → Xác thực dữ liệu người dùng nhập vào để đảm bảo đúng kiểu dữ liệu

**Câu 29: SQL Injection dựa trên điều kiện luôn đúng được gọi là gì?**

**a. Tautology**

b. PiggyBacked Query

c. Blind SQL Injection

d. Union SQL Injection

**Câu 30: Ví dụ nào sau đây mô tả một cuộc tấn công SQL Injection thành công?**

**a. SELECT \* FROM Users WHERE Name = 'admin' OR 1=1 --**

b. INSERT INTO Users (Name, Password) VALUES ('admin', '12345')

c. UPDATE Users SET Password = 'newpass' WHERE Name = 'admin'

d. SELECT \* FROM Users WHERE Name = 'admin' AND Password = '12345'