BÀI TẬP NHÓM

Đề tài: A03:2021-Injection

Nhóm: 3

Tổng quan

Một cuộc tấn công chèn SQL bao gồm việc chèn hoặc "tiêm" truy vấn SQL thông qua dữ liệu đầu vào từ máy khách đến ứng dụng. Khai thác SQL injection thành công có thể đọc dữ liệu nhạy cảm từ cơ sở dữ liệu, sửa đổi dữ liệu cơ sở dữ liệu (Chèn / Cập nhật / Xóa), thực hiện các hoạt động quản trị trên cơ sở dữ liệu (chẳng hạn như tắt DBMS), khôi phục nội dung của một tệp nhất định có trên tệp DBMS hệ thống và trong một số trường hợp đưa ra các lệnh cho hệ điều hành. Các cuộc tấn công tiêm SQL là một loại tấn công tiêm, trong đó các lệnh SQL được đưa vào đầu vào mặt phẳng dữ liệu để ảnh hưởng đến việc thực thi các lệnh SQL được xác định trước.

Mô hình hóa mối đe dọa

- Các cuộc tấn công SQL injection cho phép những kẻ tấn công giả mạo danh tính, xáo trộn dữ liệu hiện có, gây ra các vấn đề từ chối như vô hiệu hóa các giao dịch hoặc thay đổi số dư, cho phép tiết lộ hoàn toàn tất cả dữ liệu trên hệ thống, phá hủy dữ liệu hoặc làm cho dữ liệu đó không khả dụng và trở thành quản trị viên của máy chủ cơ sở dữ liệu.
- SQL Injection rất phổ biến với các ứng dụng PHP và ASP do sự phổ biến của các giao diện chức năng cũ hơn. Do bản chất của các giao diện lập trình sẵn có, các ứng dụng J2EE và ASP.NET ít có khả năng dễ dàng khai thác SQL injection.
- Mức độ nghiêm trọng của các cuộc tấn công SQL Injection bị giới hạn bởi kỹ năng và trí tưởng tượng của kẻ tấn công, và ở mức độ thấp hơn là các biện pháp phòng thủ chuyên sâu, chẳng hạn như kết nối đặc quyền thấp với máy chủ cơ sở dữ liệu, v.v. Nói chung, hãy coi SQL Injection là một tác động nghiêm trọng cao.

Sự mô tả

Tấn công chèn SQL xảy ra khi:

1. Một dữ liệu không mong muốn đi vào chương trình từ một nguồn không đáng tin cậy.
2. Dữ liệu được sử dụng để tạo động một truy vấn SQL

Hậu quả chính là:

- **Tính bảo mật**: Vì cơ sở dữ liệu SQL thường lưu giữ dữ liệu nhạy cảm nên việc mất tính bảo mật là một vấn đề thường xuyên xảy ra với các lỗ hổng SQL Injection.
- **Xác thực**: Nếu các lệnh SQL kém được sử dụng để kiểm tra tên người dùng và mật khẩu, có thể kết nối với hệ thống với tư cách là người dùng khác mà trước đó không biết về mật khẩu.
- **Ủy quyền**: Nếu thông tin ủy quyền được lưu giữ trong cơ sở dữ liệu SQL, có thể thay đổi thông tin này thông qua việc khai thác thành công lỗ hổng SQL Injection.
- **Tính toàn vẹn**: Cũng như có thể đọc được thông tin nhạy cảm, cũng có thể thực hiện các thay đổi hoặc thậm chí xóa thông tin này bằng một cuộc tấn công SQL Injection.

Các yếu tố rủi ro

Nền tảng bị ảnh hưởng có thể là:

1. Ngôn ngữ: SQL
2. Nền tảng: Bất kỳ (yêu cầu tương tác với cơ sở dữ liệu SQL)

SQL Injection đã trở thành một vấn đề phổ biến với các trang web hướng cơ sở dữ liệu. Lỗ hổng này dễ dàng bị phát hiện và dễ dàng bị khai thác, và như vậy, bất kỳ trang web hoặc gói phần mềm nào với cơ sở người dùng tối thiểu đều có thể bị tấn công theo kiểu này.

Về cơ bản, cuộc tấn công được thực hiện bằng cách đặt một ký tự meta vào dữ liệu đầu vào để sau đó đặt các lệnh SQL trong mặt phẳng điều khiển, vốn không tồn tại ở đó trước đây. Lỗ hổng này phụ thuộc vào thực tế là SQL không phân biệt thực sự giữa mặt phẳng điều khiển và dữ liệu.

Example 1

In SQL: `select id, firstname, lastname from authors`

If one provided: `Firstname: evil'ex` and `Lastname: Newman`

the query string becomes:

`select id, firstname, lastname from authors where firstname = 'evil'ex' and lastname ='newman'`

which the database attempts to run as:

`Incorrect syntax near il' as the database tried to execute evil.`

A safe version of the above SQL statement could be coded in Java as:

```
String firstname = req.getParameter("firstname");
String lastname = req.getParameter("lastname");
// FIXME: do your own validation to detect attacks
String query = "SELECT id, firstname, lastname FROM authors WHERE firstname = ? and lastname = ?";
PreparedStatement pstmt = connection.prepareStatement( query );
pstmt.setString( 1, firstname );
pstmt.setString( 2, lastname );
try
{
    ResultSet results = pstmt.execute( );
}
```

Cách ngăn chặn

Ngăn chặn việc đưa vào yêu cầu giữ dữ liệu tách biệt với các lệnh và truy vấn:
Tùy chọn ưu tiên là sử dụng một API an toàn, tránh sử dụng hoàn toàn trình thông dịch, cung cấp giao diện được tham số hóa hoặc chuyển sang Công cụ ánh xạ quan hệ đối tượng (ORM).
Lưu ý: Ngay cả khi các thủ tục được lưu trữ, được tham số hóa vẫn có thể đưa vào SQL injection nếu PL / SQL hoặc T-SQL nối các truy vấn và dữ liệu hoặc thực thi dữ liệu thù địch với EXECUTE IMMEDIATE hoặc execute ().
Sử dụng xác thực đầu vào phía máy chủ tích cực. Đây không phải là biện pháp bảo vệ hoàn toàn vì nhiều ứng dụng yêu cầu các ký tự đặc biệt, chẳng hạn như vùng văn bản hoặc API cho các ứng dụng di động.
Đối với bất kỳ truy vấn động nào còn lại, hãy thoát các ký tự đặc biệt bằng cách sử dụng cú pháp thoát cụ thể cho trình thông dịch đó.
Lưu ý: Không thể thoát các cấu trúc SQL như tên bảng, tên cột, v.v., và do đó tên cấu trúc do người dùng cung cấp rất nguy hiểm. Đây là một vấn đề thường gặp trong phần mềm viết báo cáo.
Sử dụng LIMIT và các điều khiển SQL khác trong các truy vấn để ngăn tiết lộ hàng loạt các bản ghi trong trường hợp đưa vào SQL.