**10.3 An toàn truy vấn cơ sở dữ liệu**

**10.3.1 Truy cập dữ liệu an toàn**

Databases(Cơ sở dữ liệu) là nơi lưu trữ thông tin quan trọng. Trong ứng dụng web, không thường xuyên sử dụng Databases cho việc lưu trữ các ứng dụng. Databases có thể tác động tới các thành phần. Chúng bao gồm thông tin người dùng, thông tin giao dịch, chủ thông tin( trong trường hợp của tài khoản và chủ kho), và một smorgasbord của các kiểu khác nhau của những thông tin khác nhau truy vấn đến ứng dụng CRUD của web để thêm, đọc, cập nhật và xóa.

Attackers có thể hoàn thành việc khai thác ứng dụng thông qua truy vấn dữ liệu thông tin. Attackers có thể xác định triển khai tổn hại đến hệ thống thông qua truy vấn SQL và do đó có thể có được quyền truy cập vào nội dung của cơ sở dữ liệu. Đây là kiểu tấn công phổ biến, nó được gọi là kiểu tấn công SQL injection.

SQL injection là 1 kiểu tấn công của attacker, là một cách để người thực hiện có thể thâm nhập vào hệ thống và lấy được thông tin trong cơ sở dữ liệu. Trong một số trường hợp, SQL injection thành công giúp cho attacker có thể truy cập vào một số vùng hạn chế của hệ thống. Attackers có thể thêm hoặc xóa các bảng trong cơ sở dữ liệu, từ đó truy cập được vào một số dữ liệu chính, và dần chiếm được quyền kiểm soát toàn bộ cơ sở dữ liệu.

SQL injection có thế bị tấn công vì những lý do sau:

* Dynamic use of data to construct SQL query
* Lack of input validation
* Error handling flaws
* Improper authorization rules on database

**10.3.1.1** **Dynamic Use of Data to Construct SQL Query**

Thông thường các báo cáo SQL được sử dụng bởi các ứng dụng để truy vấn cơ sở dữ liệu cho các hoạt động CRUD. Báo cáo có dạng như này:

“SELECT \* FROM USERS where USER =” + request.getParameter(txtUsername) + “AND PASSWORD =” + request.getParameter(txtPassword)

Việc sử dụng này là không an toàn vì nó cho phép attackers thực hiện các cuộc tấn công. Các dữ liệu được người sử dụng nhập động vào có thể tạo ra các trạng thái và truy vấn tới cơ sở dữ liệu. Ví dụ, attackers có thế sử dụng việc bypass dữ liệu để bỏ việc xác thực hệ thống và từ đó truy cập được vào hệ thống. attackers có thể nhập “anything” OR “1” = “1” vào một ô nhập của người dùng và nhập ký tự bất kỳ vô ô mật khẩu.  
The statement generated because of this is as follows:

SELECT \* FROM USERS where USERNAME = ‘anything’ OR ‘1’=‘1’ AND PASSWORD = ‘nonsenseValue’

Các truy vấn được tạo ra ở trên có nghĩa là nếu người dùng là anything OR if 1=1, đó là một truy vấn thực sự, truy vấn thành công và người dùng được truy cập vào ứng dụng. Điều là là vì 1=1 luôn đúng, truy vấn thành công và giúp người tấn công truy cập được vào hệ thống. Bây giờ chúng ta sẽ xem một ví dụ nguy hiểm hơn. Ví dụ, Các ứng dụng thường chứa một lĩnh vực mà người dùng nhập vào địa chỉ email Anh/chị đã đăng ký với các ứng dụng để xác minh và gửi mật khẩu của người dùng khi quên mật khẩu. Truy vấn được sử dụng bởi các ứng dụng dễ bị tổn thương để truy cập vào cơ sở dữ liệu như sau:

“SELECT username, password, full\_name, email from USERS where email = ‘” + userEmail

Attackers thấy thiệt hại tối đa qua câu truy vấn sau đây:

anon@anon.com’; DROP TABLE USERS; --

Điều này xây dựng nên câu truy vấn sau đây:

SELECT username, password, full\_name, email FROM USERS where email = ‘anon@anon.com’;

DROP TABLE USERS; --’;

Nếu cơ sở dữ liệu không chỉ đọc và đầu vào cơ sở dữ liệu là không sanitized, sau đó toàn bộ bảng, người sử dụng, sẽ được giảm xuống và các ứng dụng sẽ mất bảng người dùng của nó.

Chúng ta có thể thấy rõ ràng sự nguy hiểm và thiệt hại mà truy vấn SQL hình thức mang lại cho người dùng

Chúng ta có thể thấy rõ từ các ví dụ trên, tạo ra các truy vấn SQL từ đầu vào người dùng tự động có thể rất nguy hiểm và là một trong những nguyên nhân hàng đầu của các cuộc tấn công SQL injection đang được gây ra. Tất cả các lời gọi của cơ sở dữ liệu phải thông qua tham số. Tham số trong SQL được coi là rất có lợi từ một số điểm xem. Bảo mật và Hiệu suất là hai lợi ích chính. From a security standpoint, Tham số SQL truy vấn giúp tăng an ninh bằng cách tách SQL từ dữ liệu logic đang được cung cấp. Tham số trả về các kết quả truy vấn SQL trong, thoát các nhân vật nguy hiểm như single quotes, double quotes, and backslash characters. Nếu truy vấn động đã được khai báo, thì chức năng thoát đã được viết riêng. Đây không phải là khuyến cáo nhưng chức năng thoát là rất quan trọng và viết không đúng cách các chức năng này có thể dẫn đến các cuộc tấn công SQL injection.

Hiệu suất là một lợi ích của parameterizing truy vấn SQL. Parameterizing SQL đòi hỏi mỗi truy vấn được phân tích một thời gian duy nhất. Một khi truy vấn được chạy nhiều lần preparsing hoạt động thực hiện bởi các truy vấn tham số sẽ đảm bảo rằng động cơ tải về cơ sở dữ liệu là giảm tại thực hiện và truy vấn như vậy là tối ưu hóa so với một chức năng xây dựng động của một truy vấn dựa trên đầu vào người dùng.

**10.3.1.2 Sử dụng PrepareStatement cho tham số câu hỏi SQL**

PrepareStatement là một đối tượng dùng để thực hiện thanh trạng thái tĩnh SQL và quay trở lại kết quả mà nó tạo ra. Trong khi một đối tượng PrepareStatement thường giống như đối tượng Statement, trong đó nó có thể được sử dụng để thực hiện trạng thái SQL. Điểm khác biệt quan trọng đáng được lưu ý lại đó là SQL dùng cho PrepareStatement được biên dịch bởi dữ liệu cho chạy trước đó. Một khi PrepareStatement đã được biên dịch, nó có thể vẫn được tiếp tục được điều chỉnh bởi tham số đã được điều chỉnh xác định trước. PrepareStatement hữu ích cho việc sử dụng trong ứng dụng mà phải chạy yêu cầu SQL chung thường xuyên.

**10.3.1.3 Thiếu đầu vào của biến**

Đầu vào của biến cũng là một yêu cầu quan trọng để chắc chắn rằng SQL injection truy cập lại vào trang web ứng dụng và dữ liệu không thành công của nó. Attackers phụ thuộc đầu vào không chính xác, xác thực ( hay không xác thực) để tiến hành SQL injection và một số loại khác của attack như XSS. Input validation có thể được biểu diễn dựa vào phần trình bày được thảo luận ở phần trước của chương này. Nó quan trọng để lưu lại rằng tính xác thực của máy chủ thì không phủ nhận sự cần thiết của tham số hóa SQL queries, hoặc ngược lại. Dựa vào phần chính của defense-in-depth (được đề cập ở chương 2) quan trong để hiểu rằng phải có nhiều sự đề phòng để chắc chắn rằng attacker không thể cố gắng tấn công bằng cách phá hoại một phần phòng thủ đơn lẻ.

**10.3.1.4 Xử lý lỗi thiếu sót**

Attacker dựa nhiều vào lỗi ứng dụng tin nhắn trong khi thực hiện SQL injection attacks. Ví dụ, attacker lần đầu xác nhận 1 dạng HTML với crafted SQL query để kiểm tra việc dọn sạch của dữ liệu đầu vào. Thỉnh thoảng, tất cả những gì attacker phải làm là một lời trích dẫn trong form kèm theo dữ liệu đầu vào và một HTTP 500 Error được hiển thị đến attacker, chỉ ra rằng đầu vào đã thực sự bị chuyển đổi. Ngoài lỗi, một vài ứng dụng cũng hiển thị tìm kiếm ngoại lệ và cung cấp cho attacker những thông tin không giá trị về SQL statement bị lỗi và đưa ra kết luận giản đồ về dữ liệu. Với việc tận dụng những thông tin xác thực, attacker có thể khai thác các ứng dụng với SQL injection. Nó quan trọng khi chắc chắn rằng lỗi mà tiết lộ những thông tin nhạy cảm thì không đc báo cáo cho ng sử dụng của ứng dụng. OWASP Top Ten 2007 cũng xử lý lỗi không chính xác việc đặt tên, như là một trong những phần quan trọng dễ bị tấn công nhất cho ứng dụng Web. Một trong những yêu cầu cơ bản để để ngăn cho thông tin bị rò rỉ vì thiếu sót trong xử lý lỗi là bằng cách tạo ra những trang lỗi điều chỉnh được mà không tiết lộ những thông tin nhạy cảm đến người sử dụng. Bằng cách này, attacker không thể lấy được thêm thông tin mà có thể được sử dụng để thỏa hiệp với ứng dụng.