1. **SQL injection là gì:**



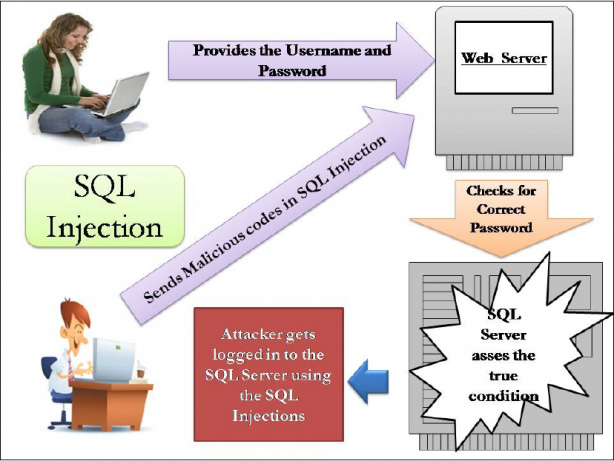
**- SQL injection** là một kỹ thuật cho phép những kẻ tấn công lợi dụng lỗ hổng của việc kiểm tra dữ liệu đầu vào trong các ứng dụng web và các thông báo lỗi của hệ quản trị cơ sở dữ liệu trả về để tiêm vào và thi hành các câu lệnh SQL bất hợp pháp.

- **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.

-**SQL injection** thường được biết đến như là một vật trung gian tấn công trên các ứng dụng web có dữ liệu được quản lý bằng các hệ quản trị cơ sở dữ liệu có cấu trúc như [SQL Server](http://toidicode.com/tu-hoc-mssql-20), [MySQL](http://toidicode.com/tu-hoc-mysql-database-19), [Oracle](http://toidicode.com/tu-hoc-oracle-database-21),...

-**Tuy nhiên** tại thời điểm mình viết bài này thì nó cũng đã giảm hẳn, nhưng vẫn còn. Do gần đây các khái niệm framework đã và đang được sử dụng rất nhiều (Các framework đều đã được test cẩn thận để phòng tránh các lỗi, trong đó có SQL Injection).

1. **Cách thức hoạt động:**



**"SELECT \* FROM `users` WHERE `username` = '$name'";**

Trường hợp bình thường nếu ta thay name = Duy thì dữ liệu sẽ trả về tất cả username có dữ liệu là Duy. Tuy nhiên nếu thay name= ' OR 1=1:

**"SELECT \* FROM `users` WHERE `username` = '' OR 1=1";**

Bởi vì 1=1 luôn đúng và kết hợp với mệnh đề OR, nên sẽ đúng trên mọi dòng dữ liệu và sẽ hiển thị tất cả các dòng dữ liệu của bảng users. Sử dụng kĩ thuật này có thể xem cả CSDL.

Nguy hiểm hơn nếu thay name= "'; DROP TABLE users";--,

"SELECT \* FROM `users` WHERE `username` = ''; DROP TABLE `users`";--";

Ở trên gồm 2 câu truy vấn, câu truy vấn đầu sẽ k làm gì cả, câu 2 sẽ xóa tất cả dữ liệu của bảng users.

Nếu thay đổi 2 thành câu truy vấn update, insert, … thì người viết có thể thay đổi cả CSDL

VD cụ thể:

Ta thường sử dụng câu lệnh SQL để truy cập dữ liệu. Giả sử, muốn tìm đăng nhập user, ta thường viết code như sau:

**var username = request.username; // Duy var password = request.password; // 123456 var sql = "SELECT \* FROM Users WHERE Username = '" + username + "' AND Password = '" + password + "'";// SELECT \* FROM Users WHERE Username = 'Duy' AND Password = '123456'**

Đoạn code trên đọc thông tin nhập vào từ user và cộng chuỗi để thành câu lệnh SQL.

Để thực hiện tấn công, Hacker có thể **thay đổi thông tin nhậpvào**, từ đó **thay đổi câu lệnh SQL** .

**var password = request.password; // ' OR '' = ''**

**var sql = "SELECT \* FROM Users WHERE Username = '" + username + "' AND Password = '" + password + "'";**

**// SELECT \* FROM Users WHERE Username = 'Duy' AND Password = '' OR '' = ''**

**// Câu SQL này luôn cho kết quả true**



Hacker có thể thông qua SQL Injection để **dò tìm cấu trúc dữ liệu** (Gồm những table nào, có những column gì), sau đó bắt đầu **khai thác dữ liệu** bằng cách sử dụng các câu lệnh như UNION, SELECT TOP 1…

1. **Cách ngăn chặn:**



* Hãy loại bỏ các kí tự meta như '"/\ và các kí tự extend như NULL, CR, LF, ... trong các string nhận được từ: input do người dùng nhập vào các tham số từ URL các giá trị từ cookie
* Đối với các giá trị numeric, hãy chuyển nó sang integer trước khi query SQL, hoặc dùng ISNUMERIC để chắc chắn nó là một số integer.
* Thay đổi "Startup and run SQL Server" dùng mức low privilege user trong tab SQL Server Security.
* Xóa các stored procedure trong database master mà không dùng như:
* xp\_cmdshell
* xp\_startmail
* xp\_sendmail
* sp\_makewebtask
* Lọc dữ liệu từ người dùng: Cách phòng chống này tương tự như [XSS](https://toidicodedao.com/2016/10/18/lo-hong-bao-mat-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](https://toidicodedao.com/2016/11/01/an-thong-tin-he-thong/) về lỗi, tránh hacker lợi dụng.
* **Phân quyền rõ ràng trong DB:** Nếu chỉ truy cập dữ liệu từ một số bảng, hãy tạo một account trong DB, 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.

**Ngăn chặn trong ASP.NET**

* Trong ASP.NET có cách ngăn chặn đơn giản là sử dụng các Parameters khi làm việc với object SqlCommand (hoặc OleDbCommand) chứ không sử dụng các câu lệnh SQL trực tiếp. Khi đó .NET sẽ tự động validate kiểu dữ liệu, nội dung dữ liệu trước khi thực hiện câu lệnh SQL.
* Ngoài ra, cũng cần kiểm soát tốt các thông báo lỗi. Và mặc định trong ASP.NET là thông báo lỗi sẽ không được thông báo chi tiết khi không chạy trên localhost.

1. **Tài liệu tham khảo**

How I hacked PacketStorm (Rain Forest Puppy) http://www.wiretrip.net/rfp/p/doc.asp?id=42&iface=6  
Great article on gathering information from ODBC error messages  
[http://www.blackhat.com/presentations/win-usa 01/Litchfield/BHWin01Litchfield.doc](http://www.blackhat.com/presentations/win-usa%2001/Litchfield/BHWin01Litchfield.doc)  
A good summary of SQL Injection on various SQL Server on <http://www.owasp.org/asac/input_validation/sql.shtml>  
Senseport's article on reading SQL Injection <http://www.sensepost.com/misc/SQLinsertion.htm>  
Khác:  
<http://www.digitaloffense.net/warga.../IOWargames.ppt>  
<http://www.wiretrip.net/rfp/p/doc.asp?id=7&iface=6>  
<http://www.wiretrip.net/rfp/p/doc.asp?id=60&iface=6>