# **SQL Injection**

# **SQL Injection là gì:**

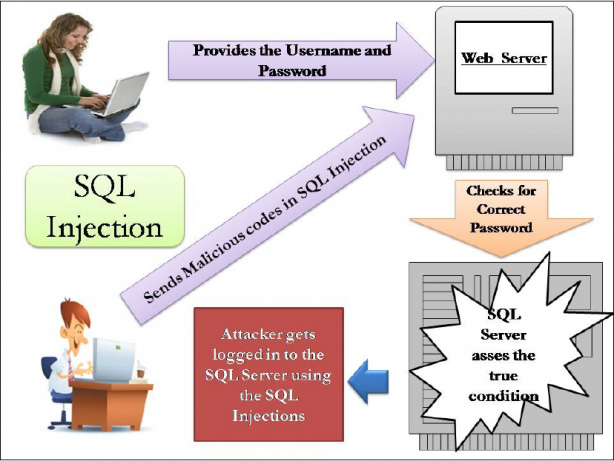


* **SQL injection** là một kỹ thuật cho phép những kẻ tấn công lợi dụng lỗ hổng của việc kiểm tra dữ liệu đầu vào trong các ứng dụng web và các thông báo lỗi của hệ quản trị cơ sở dữ liệu trả về để tiêm vào và thi hành các câu lệnh SQL bất hợp pháp.
* **SQL injection** có thể cho phép những kẻ tấn công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

### **Tại sao SQL Injection lại “thần thánh”?**

* **Cực kỳ nguy hiểm** – Có thể gây ra những thiệt hại khổng lồ. Với SQL Injection, hacker có thể truy cập một phần hoặc toàn bộ dữ liệu trong hệ thống.
* **Rất phổ biến và dễ thực hiện** – Lỗ hổng này rất nổi tiếng, từ developer đến hacker gần như ai cũng biết. Ngoài ra, còn có 1 số tool tấn công SQL Injection cho dân “ngoại đạo”, những người không biết gì về lập trình.
* **Rất nhiều ông lớn từng bị dính** – Sony, Microsoft UK. Mọi vụ lùm xùm liên quan tới “lộ dữ liệu người dùng” ít nhiều đều dính dáng tới SQL Injection.
* Dễ tấn công, phổ biến, gây ra hậu quả nghiêm trọng, đó là lý dó Inject (Không chỉ SQL mà OS và LDAP) nằm chễm chễ ở vị trí **đầu bảng trong top 10** lỗ hỗng bảo mật của OWASP

# **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 = 'Hoangcute' AND Password = '' OR '' = ''**

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



* Hacker có thể drop luôn table Users, xoá toàn bộ người dùng trong database
* 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…

**Bonus:**

* Ngoài ra, cũng có thể thử cách khác bằng **Submit ' or 'a’=‘a**

**SELECT \* FROM product WHERE PCategory='food' or 'a'=‘a’**

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

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

# **Cách ngăn chặn:**



* **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.
* **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.
* **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.
* **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 :** 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.

**Fun fact:**

* Tại thời điểm 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)

# **So sánh LinQ To SQL và Entity Framework**

**LinQ to SQL** cho phép ta query và modify data của SQL Server sử dụng cú pháp LinQ.

**Entity framework** là một ORM (Object-relational mapping), cho phép chúng ta query và modify RDBMS như SQL Server, Oracle, DB2, MySql... cũng bằng cách sử dụng cú pháp LinQ

EF được sử dụng thường xuyên trong ứng dụng .NET để query tới database

Và sau đây là một số điểm khác nhau giữa LinQ to SQL và EF :

|  |  |
| --- | --- |
| **LINQ to SQL** | **Entity Framework** |
| Chỉ làm việc đượv với SQL Server Database. | Có thể works trên nhưng db khác nhau như Oracle, DB2, MYSQL, SQL Server etc. |
| Nó tạo ra một file .dbml để duy trì mối quan hệ | Nó tạo ra một file .edmx lúc đầu. Các mối quan hệ được duy trì sử dụng 3 file khác nhau .csdl, .msl và .ssdl |
| Không hỗ trợ cho kiểu phức tạp. | Hỗ trợ các kiểu phức tạp. |
| Không thể tạo db từ model | Có thể tạo db từ model. |
| Chỉ cho phép 1-1 ánh xạ giữa các lớp thực thể và quan hệ các tables / views. | Cho phép một-một, một-nhiều và; nhiều-nhiều ánh xạ giữa các lớp thực thể và các quan hệ tables / views |
| Cho phép bạn truy vấn dữ liệu bằng DbContext. | Cho phép bạn truy vấn dữ liệu sử dụng Entity SQL, ObjectContext, DbContext. |
| Nó cung cấp một phương pháp tiếp cận kết chặt chẽ. | Code first cho phép sử dụng Dependency Injection pattern |
| Có thể được sử dụng để phát triển ứng dụng nhanh chóng chỉ với SQL Server. | Có thể được sử dụng để phát triển ứng dụng nhanh chóng với RDBMS như SQL Server, Oracle, DB2 và MySQL vv. |

**ADO.NET:**

* Công nghệ đầu tiên của mircosoft, cũ và lạc hậu.
* Cộng chuỗi từ câu lệnh trong sql thì rất mất thời gian.
* Newbie thì nên làm quen cái này

**LINQ:**

* Cải tiến hơn tí xíu.
* Chạy trực tiếp câu sql.

2 cái này đều có điểm yếu là thay đổi cấu trúc database và tên field thì phải sửa.  
**ADO.NET ENTITY FRAMEWORK:**

* Map database vào nên sẽ không cần viết query sql
* Không lo thay đổi cấu trúc, số field.

Chúng tương tự nhau và có thể được sử dụng theo cách giống nhau, nhưng cũng có 1 số sự khác biệt quan trọng.

**Lưu ý rằng "LINQ" không tương tự với "LINQ to SQL"**

EF cũng sử dụng LINQ.

Một số điểm khác biệt đáng chú ý là:

* LTS(LINQ to SQL) chủ yếu là SQL Server. EF được thiết kế để hỗ trợ và hỗ trợ nhiều DB, nếu như bạn có ADO.NET tương thích.
* Về cách thay đổi siêu dữ liệu. Bạn phải tạo lại từng phàn của mô hình bạn từ đầu và mất các tùy chỉnh.
* EF hỗ trợ các tính năng mô hình như quan hệ nhiều-nhiều và thừa kes. LTS thì không hỗ trợ trực tiếp.
* Trong .NET 3.5, LTS hỗ trợ nhiều hơn cho các chức năng SQL Server riêng biệt so với EF.
* Nhưng ở .NET 4, chúng tương đối giống nhau trong khía cạnh trên
* EF cho bạn chọn Model First, DB First, or Code First modeling. LTS chỉ hỗ trợ DB First.