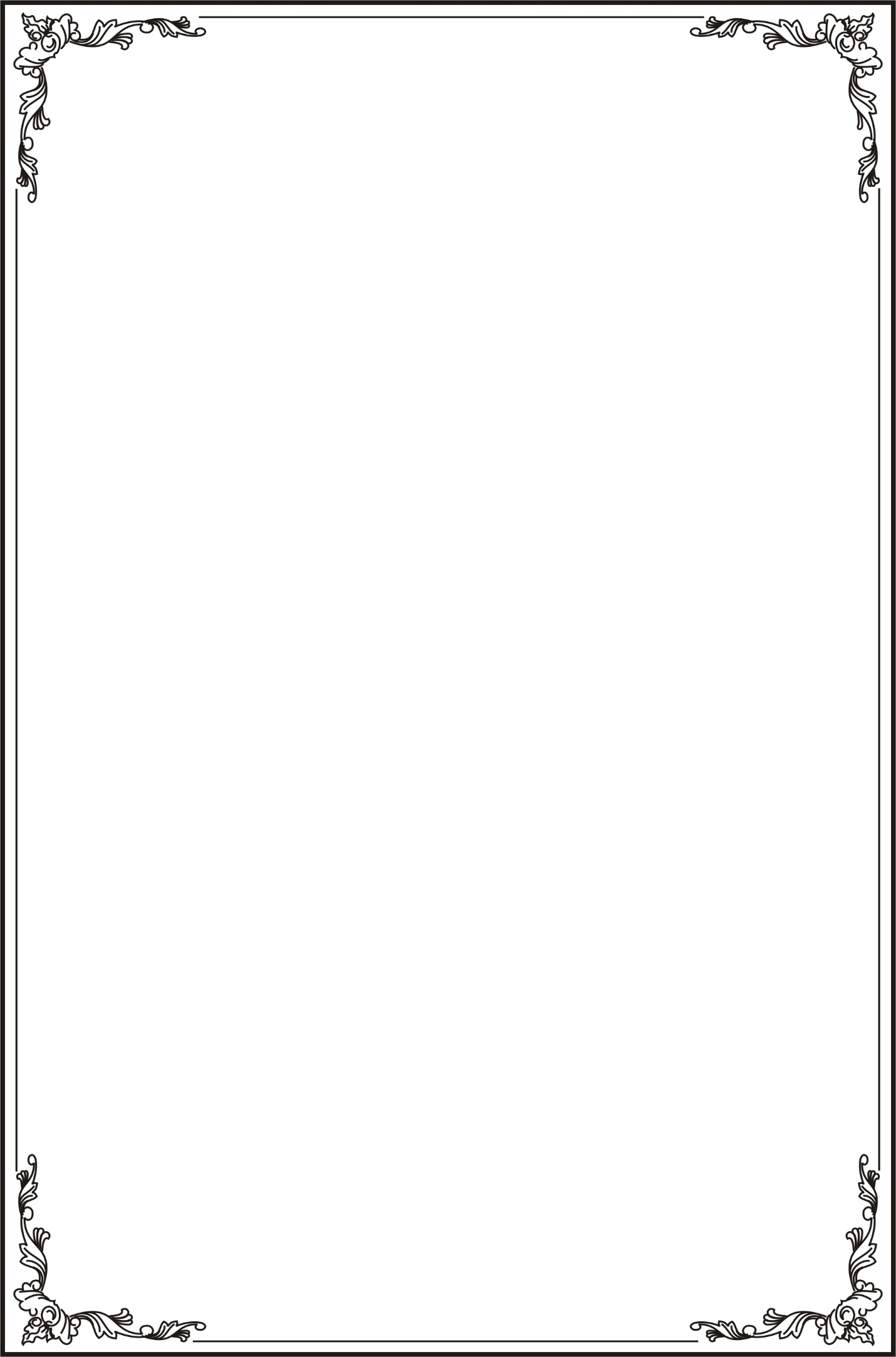
****

TRƯỜNG ĐẠI HỌC KINH TẾ - ĐẠI HỌC ĐÀ NẴNG

KHOA THỐNG KÊ – TIN HỌC





***Môn học*: An toàn và bảo mật hệ thống thông tin**

**BÁO CÁO BÀI TẬP NHÓM**

***Chủ đề:* TÌM HIỂU CÁC DẠNG TẤN CÔNG SQL INJECTION.**

**XÂY DỰNG VÀ DEMO CÁC KỊCH BẢN TẤN CÔNG SQL INJECTION.**

***GVHD*: Th.S Trần Thị Thu Thảo**

***Nhóm 5:*** Nguyễn Văn Hoàng Nhã

Chế Thị Nhã Quyên

Trần Quang Sang

Trần Thị Minh Uyên

Ngô Thị Hồng Vân

***Lớp:*** *45K14*

*Tháng 3 năm 2022*

**MỤC LỤC**

[**DANH MỤC HÌNH ẢNH** 2](#_Toc98578548)

[**PHÂN CHIA CÔNG VIỆC** 3](#_Toc98578549)

[**I.** **TỔNG QUAN** 4](#_Toc98578550)

[**II.** **CÁC DẠNG TẤN CÔNG** 4](#_Toc98578551)

[**1.** **Tấn công vượt qua kiểm tra đăng nhập (username/password)** 4](#_Toc98578552)

[**2.** **Tấn công sử dụng câu lệnh SELECT** 8](#_Toc98578553)

[**3.** **Tấn công dựa vào lệnh INSERT** 10](#_Toc98578554)

[**4.** **Tấn công sử dụng stored-procedures** 11](#_Toc98578555)

[**III.** **Cách phòng chống tấn công SQL Injection** 13](#_Toc98578556)

[**IV.** **Kết luận** 15](#_Toc98578557)

[**1.** **Kết luận** 15](#_Toc98578558)

[**2.** **Ưu điểm** 15](#_Toc98578559)

[**3.** **Nhược điểm** 15](#_Toc98578560)

### **DANH MỤC HÌNH ẢNH**

[Hình 1: Code đăng nhập 4](#_Toc98578217)

[Hình 2: Form đăng nhập 5](#_Toc98578218)

[Hình 3: Cơ sở dữ liệu của phần đăng nhập 5](#_Toc98578219)

[Hình 4: Mã nguồn truy vấn theo ID 8](#_Toc98578220)

[Hình 5: Mã nguồn tìm kiếm 9](#_Toc98578221)

[Hình 6: Mã xây dựng câu lệnh SQL 11](#_Toc98578222)

[Hình 7: Lấy thông tin trong bảng TAIKHOAN 11](#_Toc98578223)

[Hình 8: Ràng buộc kiểu dữ liệu 13](#_Toc98578224)

[Hình 9: Regular Expression 14](#_Toc98578225)

[Hình 10: Hàm mysqli\_real\_escape\_string 14](#_Toc98578226)

[Hình 11: Hàm mysqli\_real\_escape\_string 14](#_Toc98578227)

### **PHÂN CHIA CÔNG VIỆC**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **Nhiệm vụ** | **Hoàn thành** |
| 1 | Ngô Thị Hồng Vân | Tìm hiểu về phần tổng quan  Tổng hợp phần word tấn công | 100% |
| 2 | Trần Thị Minh Uyên | Tìm hiểu tấn công vượt qua kiểm tra đăng nhập (username/password) | 100% |
| 3 | Trần Quang Sang | Tìm hiểu tấn công sử dụng câu lệnh SELECT | 100% |
| 4 | Chế Thị Nhã Quyên | Tìm hiểu tấn công dựa vào lệnh INSERT | 100% |
| 5 | Nguyễn Văn Hoàng Nhã | Tìm hiểu tấn công sử dụng stored-procedures | 100% |
| 6 | Trần Quang Sang | Quay demo các cách tấn công | 100% |
| 7 | Chế Thị Nhã Quyên  Ngô Thị Hồng Vân | Tổng hợp word  Tìm hiểu về cách phòng chống | 100% |
| 8 | Nguyễn Văn Hoàng Nhã  Trần Thị Minh Uyên | Làm slide | 100% |
| 9 | Trần Quang Sang  Chế Thị Nhã Quyên  Ngô Thị Hồng Vân  Nguyễn Văn Hoàng Nhã  Trần Thị Minh Uyên | Chỉnh sửa word | 100% |

# **TỔNG QUAN**

SQL injection là kĩ thuật cho phép kẻ tấn công:

* Lợi dụng lỗ hổng trong kiểm tra dữ liệu nhập vào các ứng dụng
* Dựa vào các thông báo lỗi của hệ quản trị cơ sở dữ liệu
* Để "tiêm vào” (inject) và thi hành các câu lệnh SQL bất hợp pháp

SQL injection là một kỹ thuật điền vào những đoạn mã SQL cho phép khai thác thông tin trong CSDL của một ứng dụng.

Lỗ hổng bảo mật này xuất hiện khi ứng dụng không có đoạn mã kiểm tra chuỗi ký tự thoát nhúng trong câu truy vấn SQL, hoặc do sự định kiểu đầu vào không rõ ràng hay do lỗi cú pháp SQL của lập trình viên khiến cho một đoạn mã được xử lý ngoài ý muốn.

# **CÁC DẠNG TẤN CÔNG**

## **Tấn công vượt qua kiểm tra đăng nhập (username/password)**

Với dạng tấn công này, kẻ tấn công 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. 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 mật khẩu có hợp lệ hay không để quyết định cho phép hay từ chối thực hiện tiếp. Nếu thành công, thì có thể login vào mà không cần phải biết username và password.

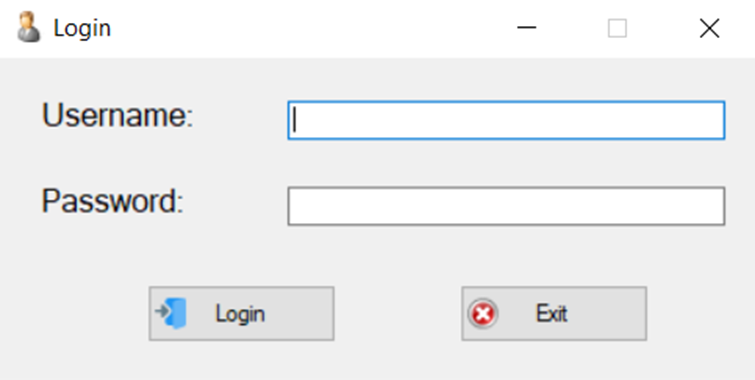
Ví dụ:

* Ta có đoạn code chương trình như sau:



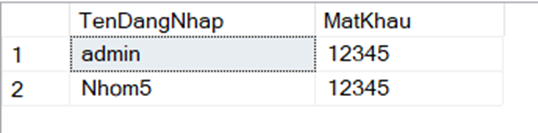
Hình 1: Code đăng nhập

* Với form login như dưới đây:



Hình 2: Form đăng nhập

* Có cơ sở dữ liệu như sau:



Hình 3: Cơ sở dữ liệu của phần đăng nhập

* **Trường hợp 1:**
* Nhập thông tin vào ô username hoặc password là 'or 1=1--.
* Nhập vào ô còn lại với nội dung bất kỳ, sau đó click vào nút Login.
* Chúng ta có thể đăng nhập thành công. Nguyên nhân chúng ta có thể dễ dàng đăng nhập vào hệ thống mà thậm chí không có thông tin đăng nhập là vì khi đăng nhập hệ thống, chương trình sẽ tự động tạo ra câu lệnh truy vấn dữ liệu với tham số được truyền vào từ form login.

Với dữ liệu chúng ta vừa nhập, câu lệnh truy vấn sẽ là:

**Select\*from TAIKHOAN where TenDangNhap=' 'or 1=1-- 'and MatKhau=''**

Sau ký tự “--“, hệ thống sẽ hiểu đó là comment, chương trình sẽ ẩn chuỗi ký tự phía sau nó trên cùng 1 dòng và không thực thi những gì phía sau nó. Câu truy vấn có thể rút gọn lại:

**Select\*from TAIKHOAN where TenDangNhap=' 'or 1=1**

Điều kiện trong mệnh đề WHERE luôn đúng và kết quả truy vấn sẽ trả về tất cả các bản ghi của bảng TAIKHOAN, đó là lý do vì sao chúng ta có thể vượt qua màn hình đăng nhập một cách dễ dàng.

* **Tương tự trường hợp 2:**
* Sử dụng kí tự đặc biệt “--".
* Sau ký tự “--“, hệ thống sẽ hiểu đó là comment, chương trình sẽ ẩn chuỗi ký tự phía sau nó trên cùng 1 dòng và không thực thi những gì phía sau nó.

Giá trị nhập vào: Username: admin'--

Password: “Nhập vào đây ký tự bất kỳ”

Với dữ liệu chúng ta vừa nhập, câu lệnh truy vấn sẽ là:

**Select\*from TAIKHOAN where TenDangNhap='admin'-- 'and MatKhau=''**

* Ta có thể đăng nhập thành công với quyền admin mà không cần password. Kết quả truy vấn sẽ trả về tất cả các bản ghi của bảng TAIKHOAN và dòng đầu tiên trong bảng TAIKHOAN sẽ được chọn.
* **Trường hợp 3:**

Giá trị nhập vào: Username: ' or ''='

Password: ' or ''='

Câu lệnh SQL sẽ như sau:

**Select\*from TAIKHOAN where TenDangNhap='' or ''='' and MatKhau='' or ''=''**

* Chúng ta có thể đăng nhập thành công vì điều kiện trong mệnh đề WHERE là luôn luôn đúng, kết quả truy vấn sẽ trả về tất cả các bản ghi của bảng TAIKHOAN và dòng đầu tiên trong bảng TAIKHOAN sẽ được chọn.
* **Trường hợp 4: :** Kết hợp với kí tự đặc biệt của SQL: Kí tự “ ; ”: đánh dấu kết thúc 1 câu truy vấn

Giá trị nhập vào: Username: ‘;drop table TAIKHOAN--

Password:

Câu lệnh SQL lúc này như sau:

**Select\*from TAIKHOAN WHERE TenDangNhap= ''; drop table TAIKHOAN -- ' and MatKhau= ''**

* Chúng ta có thể xóa một bảng trong cơ sở dữ liệu nếu biết tên của bảng mà không cần phải đăng nhập vì sau ký tự “;” hệ thống sẽ hiểu đó là đánh dấu để kết thúc 1 câu truy vấn và chương trình sẽ thực hiện câu truy vấn sau đó.

**Một số trường hợp khác tương tự vẫn đăng nhập được thành công:**

|  |  |  |
| --- | --- | --- |
| **Username** | **Password** | **Câu lệnh SQL** |
| Nhập ký tự bất kỳ | ' or '1' = '1 | Select \* from TAIKHOAN where TenDangNhap = ''and MatKhau = '' or '1' = '1' |
| Nhập ký tự bất kỳ | ' or 1='1 | Select \* from TAIKHOAN where TenDangNhap = ''and MatKhau = '' or 1 = '1' |
| 1' or 1 = 1 -- - | Nhập ký tự bất kỳ | Select \* from TAIKHOAN where TenDangNhap = '1' or 1 = 1 -- -'and MatKhau = '' |
| Nhập ký tự bất kỳ | 1' or 1 = 1 -- - | Select \* from TAIKHOAN where TenDangNhap = '' and MatKhau = '1' or 1 = 1 -- -' |

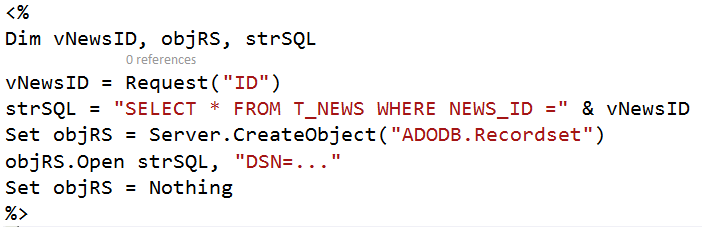
Bảng 1: Một số trường hợp khác tương tự vẫn đăng nhập

## **Tấn công sử dụng câu lệnh SELECT**

Dạng tấn công này phức tạp hơn. Để thực hiện được kiểu tấn công này, kẻ tấn công phải có khả năng hiểu và lợi dụng các sơ hở trong các thông báo lỗi từ hệ thống để dò tìm các điểm yếu khởi đầu cho việc tấn công.

Xét một ví dụ rất thường gặp trong các website về tin tức. Thông thường, sẽ có một trang nhận ID của tin cần hiển thị rồi sau đó truy vấn nội dung của tin có ID này.

Ví dụ: http://www.myhost.com/shownews.asp?ID=123. Mã nguồn cho chức năng này thường được viết khá đơn giản theo dạng:



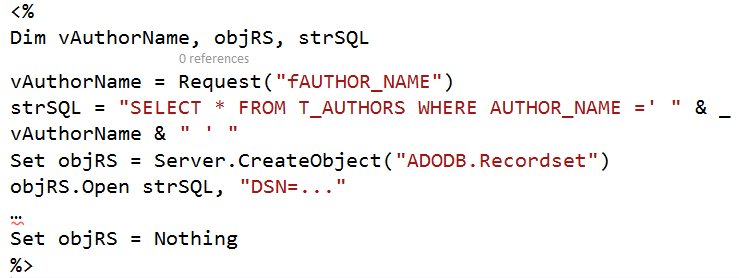
Hình 4: Mã nguồn truy vấn theo ID

Trong các tình huống thông thường, đoạn mã này hiển thị nội dung của tin có ID trùng với ID đã chỉ định và hầu như không thấy có lỗi. Tuy nhiên, giống như ví dụ đăng nhập ở trước, đoạn mã này để lộ sơ hở cho một lỗi SQL injection khác. Kẻ tấn công có thể thay thế một ID hợp lệ bằng cách gán ID cho một giá trị khác, và từ đó, khởi đầu cho một cuộc tấn công bất hợp pháp.

Ví dụ như: **0 OR 1=1** (nghĩa là http://www.myhost.com/shownews.asp?ID=0 or 1=1)

Câu truy vấn SQL lúc này sẽ trả về tất cả các article từ bảng dữ liệu vì nó sẽ thực hiện câu lệnh: **SELECT \* FROM T\_NEWS WHERE NEWS\_ID=0 or 1=1**

Một trường hợp khác, ví dụ như trang tìm kiếm. Trang này cho phép người dùng nhập vào các thông tin tìm kiếm như Họ, Tên, … Đoạn mã thường gặp là:



Hình 5: Mã nguồn tìm kiếm

Tương tự như trên, tin tặc có thể lợi dụng sơ hở trong câu truy vấn SQL để nhập vào trường tên tác giả bằng chuổi giá trị:

**' UNION SELECT ALL SELECT OtherField FROM OtherTable WHERE ''=' (\*)**

Lúc này câu truy vấn đầu không thành công, chương trình sẽ thực hiện thêm lệnh tiếp theo sau từ khóa UNION nữa.

Tất nhiên những ví dụ nói trên,dường như không có gì nguy hiểm,nhưng hãy thử tương tự kẻ tấn công có thể xóa toàn bộ cơ sở dữ liệu bằng cách chèn vào các đoạn lệnh nguy hiểm như lệnh DROP TABLE. Ví dụ như: **' DROP TABLE T\_AUTHORS –**

Chắc các bạn sẽ thắc mắc là làm sao biết được ứng dụng web bị lỗi dạng này được.

Rất đơn giản, hãy nhập vào chuỗi (\*) như trên, nếu hệ thống báo lỗi về cú pháp dạng: Invalid object name “OtherTable”; ta có thể biết chắc là hệ thống đã thực hiện câu SELECT sau từ khóa UNION, vì như vậy mới có thể trả về lỗi mà ta đã cố tình tạo ra trong câu lệnh SELECT.

Cũng sẽ có thắc mắc là làm thế nào có thể biết được tên của các bảng dữ liệu mà thực hiện các thao tác phá hoại khi ứng dụng web bị lỗi SQL injection. Cũng rất đơn giản, bởi vì trong SQL Server, có hai đối tượng là sysobjects và syscolumns cho phép liệt kê tất cả các tên bảng và cột có trong hệ thống. Ta chỉ cần chỉnh lại câu lệnh SELECT, ví dụ như:

**" UNION SELECT name FROM sysobjects WHERE xtype = 'U'** là có thể liệt kê được tên tất cả các bảng dữ liệu.

## **Tấn công dựa vào lệnh INSERT**

Thông thường các ứng dụng web cho phép người dùng đăng kí một tài khoản để tham gia. Khi muốn truy cập vào một trang web nào đó, sẽ có một mục đăng ký tài khoản cho phép người dùng ấn vào. Ngay sau khi đăng ký thành công, người dùng sẽ được tùy chỉnh thông tin cá nhân của mình. SQL injection có thể được dùng khi hệ thống không kiểm tra tính hợp lý của thông tin người dùng nhập vào.

Cơ chế: Kích thích cho hệ quản trị phát sinh thông báo lỗi mang theo cả thông tin về CSDL. Có thể sử dụng một số lỗi cú pháp thông thường như sau:

* Tìm tên bảng, tên cột: Điển hình với mệnh đề group by, tên các trường sau select phải có sau mệnh đề group by, có thể có having. Những không thể chấp nhận câu truy vấn có having đứng lẻ loi.
* Định kiểu của từng trường: Điển hình với thông báo từ lỗi tính tổng (sum) hoặc trung bình (avg) trường có kiểu không phải dạng số.

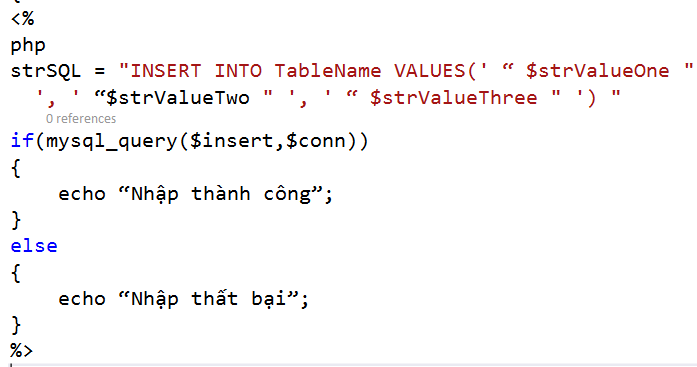
**Username: ’; insert into tblUser(tkUser, tkPass) values(‘hacker’, ‘123456’)**

**🡪** Hacker có thể đăng nhập vào hệ thống thông qua tài khoản vừa tạo

**Cú pháp lệnh INSERT:**

**INSERT INTO TableName VALUES('Value One‘,'Value Two', 'Value Three'**

* Nếu đoạn mã xây dựng câu lệnh SQL có dạng:



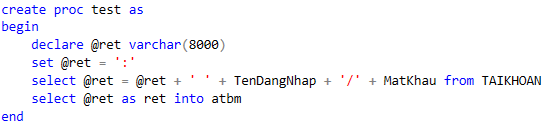
Hình 6: Mã xây dựng câu lệnh SQL

* Thì chắc chắn sẽ bị lỗi SQL Injection, bởi vì nếu ta nhập vào trường thứ nhất ví dụ như: **' + (SELECT TOP 1 FieldName FROM TableName) + '.**
* Lúc này câu truy vấn sẽ là: **INSERT INTO TableName VALUES(' ' + (SELECT TOP 1 FieldName FROM TableName) + ' ', 'abc', 'def')**
* Khi đó, lúc thực hiện lệnh xem thông tin, xem như ta đã yêu cầu thực hiện thêm một lệnh nữa đó là: **SELECT TOP 1 FieldName FROM TableName**

## **Tấn công sử dụng stored-procedures**

Trong SQL, máy chủ có các cơ sở dữ liệu tổng thể và trong các cơ sở dữ liệu này có các hệ thống thủ tục lưu trữ được lập trình sẵn để thực hiện các công việc cụ thể nào đó. Nếu như kẻ tấn công chiến được quyền điều khiển hệ quản trị và thực hiện truy vấn đến các lưu trữ này hoặc nguy hiểm hơn là xóa đi cơ sở dữ liệu tổng thể thì toàn bộ cơ sở dữ liệu sẽ bị ảnh hưởng và không hoạt động được.

* Ví dụ lấy thông tin trong bảng TAIKHOAN:



Hình 7: Lấy thông tin trong bảng TAIKHOAN

* Bước 1:Tạo một Stored procedure để chép vào tất cả thông tin của 2 trường *TenDangNhap* và *MatKhau* trong bảng TAIKHOAN thành một chuỗi vào một bảng mới là *atbm* có một trường là ret: Thực thi câu lệnh bằng cách nhập vào form

**Username: ‘; Create proc test as begin declare @ret varchar(8000) set @ret=‘:’ select @ret=@ret + '’ + TenDangNhap****+ ‘/’ + MatKhau from TAIKHOAN select @ret as ret into atbm**

* Bước 2: Gọi Stored procedure đó: **Username:’;** exec test. Kết quả tạo ra một table mới trên CSDL có tên atbm
* Bước 3: Dùng UNION để xem nội dung bảng atbm

**Username:’;select ret,1 from atbm union select 1,1 from atbm**

* Lỗi phát sinh:

Microsoft OLEDB Provider for ODBC Drivers error '80040e07'[Microsoft][ODBC SQL Server Driver][SQLServer]Syntax error convertingthe varchar value ':admin/danang2014 user01/123456 user02/123abc ' to a column of data type int.

* Hacker có nội dung gồm có tên TenDangNhap và mật khẩu MatKhau
* Bước 4: xoá bảng foo để xoá dấu vết: **Username: ‘; drop table atbm—**
* Tấn công bằng stored-procedures sẽ gây hậu quả lớn nếu ứng dụng được thực thi với quyền quản trị hệ thống 'sa'.
* Ví dụ, nếu ta thay đoạn mã tiêm vào dạng:

**Username: ' ; EXEC xp\_cmdshell ‘cmd.exe dir C: '.**

* Hệ thống sẽ thực hiện lệnh liệt kê thư mục trên ổ đĩa C:\ cài đặt server.
* Việc phá hoại kiểu nào tuỳ thuộc vào câu lệnh đằng sau cmd.exe. Nếu cài SQL ở chế độ mặc định thì SQL chạy trên nền SYSTEM, tương đương mức truy cập ở Windows. Có thể dùng master..xp\_cmdshell để thi hành lệnh từ xa.

# **Cách phòng chống tấn công SQL Injection**

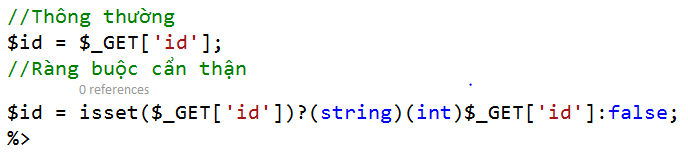
Như vậy, có thể thấy lỗi SQL Injection khai thác những bất cẩn của các lập trình viên phát triển ứng dụng web khi xử lí các dữ liệu nhập vào để xây dựng câu lệnh SQL. Để phòng chống có một số biện pháp sau:

* Kiểm soát chặt chẽ dữ liệu đầu vào: Điểm yếu SQL Injection bắt nguồn từ việc xử lý dữ liệu từ người dùng không tốt, do đó cần có cơ chế kiểm soát chặt chẽ và giới hạn quyền xử lí dữ liệu đến tài khoản người dùng mà ứng dụng web đáng sử dụng, xây dựng mã nguồn đảm bảo an ninh là cốt lõi của việc phòng chống SQL Injection
* Thiết lập các đối tượng giả làm mồi nhử: Chiến thuật này được đưa ra nhằm cảnh báo cho quản trị viên nguy cơ một cuộc tấn công khi một ai đó cố tình tìm cách khai thác những dữ liệu nhạy cảm như password, id. Phương pháp này nên phối hợp với việc đặt tên các đối tượng khó đoán.
* Thiết lập cấu hình an toàn cho hệ quản trị cơ sở dữ liệu: Cần có cơ chế kiểm soát chặt chẽ và giới hạn quyền xử lí dữ liệu đến tài khoản người dùng mà ứng dụng web đang sử dụng. Các ứng dụng thông thường nên tránh dùng đến các quyền như ‘dbo’ hay ‘sa’. Quyền càng bị hạn chế, thiệt hại càng ít.

Dưới đây là cách phòng chống đối với dạng tấn công vượt qua kiểm tra đăng nhập:

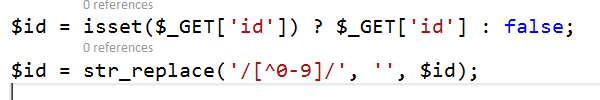
* Luôn ràng buộc kiểu dữ liệu:

Ví dụ: Như đối với phương thức get url: <http://domain.com?id=5>.



Hình 8: Ràng buộc kiểu dữ liệu

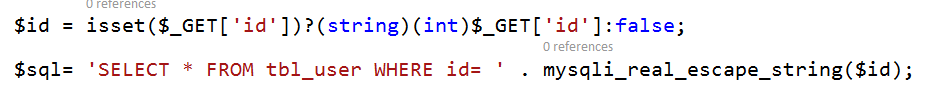
* Regular Expression: Hoặc có thể dùng Regular Expression để loại bỏ đi các ký tự lạ hoặc các ký tự không phải là số. Ví dụ:



Hình 9: Regular Expression

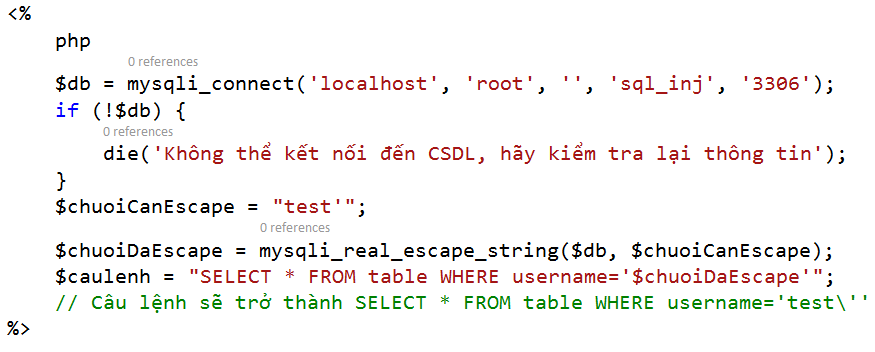
* Dùng các hàm có sẵn để giảm thiểu lỗi:

Sử dụng thêm hàm mysqli\_real\_escape\_string để chuyển đổi một chuỗi thành một query an toàn. Ví dụ:



Hình 10: Hàm mysqli\_real\_escape\_string

Đa số các ngôn ngữ lập trình hiện tại đã hỗ trợ hàm để có thể escape câu lệnh (thêm các dấu ( \ ) trước các ký tự đặc biệt) trước khi thực thi nó trên cơ sở dữ liệu, do đó sẽ hạn chế và có thể sẽ chặn lỗ hổng SQL injection một cách an toàn.Và ngôn ngữ lập trình PHP hiện tại có hỗ trợ nhiều hàm cho việc này, tiêu biểu nhất là hàm mysqli\_real\_escape\_string(), với hàm này, hệ thống sẽ dùng cú pháp của hệ thống CSDL để escape các chuỗi lệnh thành lệnh SQL tiêu chuẩn.



Hình 11: Hàm mysqli\_real\_escape\_string

# **Kết luận**

# **Kết luận**

Trên đây là bài báo cáo về tìm hiểu các dạng tấn công SQL Injection và xây dựng demo các kịch bản tấn công SQL Injection củanhóm 5. Qua đó giúp chúng ta hiểu hơn về thế nào là tấn công SQL Injection, cách thức tấn công của từng. Bên cạnh đó cung cấp thêm những kiến thức cơ bản và các giải pháp công nghệ giúp hạn chế vấn đề này. Hy vọng những thông tin này đã giúp ích cho chúng ta trong việc nâng cao độ bảo mật cho hệ thống của cá nhân cũng như doanh nghiệp một cách hiệu quả nhất.

# **Ưu điểm**

Các thành viên trong nhóm hoàn thành đầy đủ nhiệm vụ bài tập được phân chia và nộp lại đúng hạn. Mọi người trong nhóm tương tác hợp tác tốt.

# **Nhược điểm**

Việc mới tiếp xúc với các kiến thức về tấn công SQL Injection nên kiến thức còn hạn hẹp và thời gian hạn chế nên trong bài báo cáo này sẽ không tránh khỏi những thiếu sót, hạn chế nhất định. Vì vậy, chúng em rất mong được sự đóng góp ý kiến của cô và các bạn để chúng em và cả lớp có điều kiện nâng cao kiến thức của bản thân.