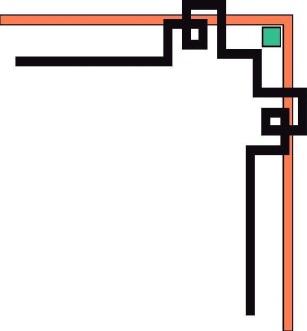
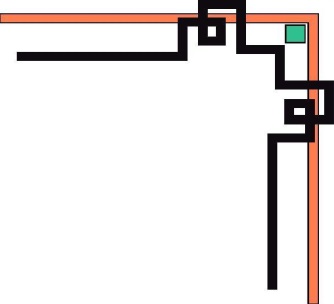
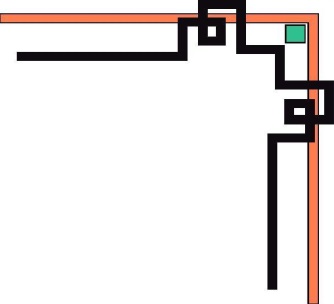
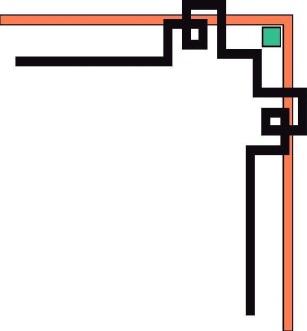
**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**



**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**



**BÁO CÁO MÔN HỌC**

**NHẬP MÔN AN TOÀN THÔNG TIN**

Tên đề tài: Tấn công SQL Injection và cách phòng chống

**Giáo viên hướng dẫn PGS Nguyễn Linh Giang**

**Nhóm sinh viên thực hiện:**

**Nguyễn Trần Hiếu KHMT05-K62 20173118**

**Nguyễn Minh Kiên CNTT11-K62 20173206**

**Phạm Lương Đức 20151056**

**Phan Bá Hoàng KHMT02 –K62 20173141**

**Hà Nội – Ngày 1 tháng 6 năm 2020**

Chương 1: Tấn công SQL Injection

* 1. Khái niệm tấn công SQL injection

SQL injection là một trong những kiểu tấn công mạng phổ biến. SQL injection là một kỹ thuật tấn công vào hệ thống cơ sở dữ liệu của ứng dụng (web, mobile hoặc desktop …) thông qua việc kẻ tấn công lợi dụng lỗ hổng để tiến hành đưa mã độc vào câu truy vấn SQL và thực thi các câu lệnh SQL bất hợp pháp trên cơ sở dữ liệu (CSDL) đằng sau ứng dụng web.

Thay vì nhập dữ liệu hợp lệ, kẻ tấn cống sẽ nhập một câu lệnh truy vấn độc hại bất kỳ vào, có khả năng gây ra một số thiệt hại nghiêm trọng đối với cơ sở dữ liệu và toàn bộ hệ thống. Kẻ tấn công có thể lấy cắp được thông tin nhạy cảm của khách hàng, người dùng như tên, tuổi, địa chỉ, số tài khoản, số điện thoại … Ngoài ra, tấn công SQL injection có thể cho phép những kẻ tấn công thực hiện các thao tác xóa, sửa, chèn, cập nhật trên CSDL của ứng dụng, thậm chí là máy chủ mà ứng dụng đó đang chạy. Lỗi này thường xãy ra trên các ứng dụng web có dữ liệu được quản lý bằng các hệ quản trị cơ sở dữ liệu như SQL Server, MySQL, Oracle, DB2, Sysbase...

* 1. Nguyên nhân gây ra các lỗi bảo mật SQL injection

Hai phương thức truyền dữ liệu cơ bản mà các ứng dụng web sử dụng là GET và POST.

GET : dữ liệu gửi từ trình duyệt lên qua phương thức GET là phần tử dữ liệu được nhập trực tiếp theo sau địa chỉ URL do trình duyệt gửi lên.

Ví dụ: http://127.0.0.1/index.php?trangweb=trangchu

Method POST: POST là phần gửi dữ liệu được gửi qua các form HTML có method = “POST”. Để lấy các biến theo kiểu dữ liệu POST, PHP sẽ tự động sinh ra mảng có tên là $\_POST. Mảng này có chỉ số là tên của các phần tử trong form và giá trị là nội dung giá trị do người sử dụng nhập vào các phần tử có tên tương ứng.

Do đây là hình thức truyền dữ liệu từ client lên server do đó việc không kiểm tra chính xác dữ liệu cung cấp bởi người sử dụng sẽ gây ra các lỗi bảo mật SQL Injection.

Ví dụ:

“payload:

username:‘admin’;

password:‘12345’;

”

* 1. Các bước thực hiện tấn công SQL injection của một cuộc tấn công cơ bản

Bước 1: Kẻ tấn công dò quét lỗ hổng của ứng dụng web sử dụng phương pháp backtracking. Khi phát hiện lỗ hổng, hắn tiến hành nhập vào chuỗi ký tự độc hại, và bắt đầu tấn công.

Bước 2: Máy chủ web tiếp nhận truy vấn và gửi nó đến máy chủ ứng dụng web.

Bước 3: Tại máy chủ ứng dụng web, truy vấn độc hại tiếp tục được chấp nhận mà không có phương án chặn lại.

Bước 4: Tại máy chủ CSDL, truy vấn độc hại được thực thi. Dữ liệu chứa thông tin nhạy cảm của người dung sẽ được trả về và hiển thị trên ứng dụng web.

Bước 5: Kẻ tấn công nhận được thông tin nhạy cảm. Sau đó, kẻ tấn công tiếp tục thực hiện các tấn công SQL khác.

1.4. Phân loại tấn công SQL injection

Tấn công SQL injection được phân chia thành ba loại như sau: Order Wise, Blind SQL Injection, Against Database

**1.4.1. Order Wise**

Tấn công kiểu dạng “yêu cầu khôn” là kiểu tấn công chủ yếu lợi dụng sự khôn khéo của những đoạn code để phục vụ mục đích tấn công

Tấn công order wise có 3 dạng:

+ First order: Kẻ tấn công sẽ lấy dữ liệu luôn và để lại vết tích trên máy chủ nạn nhân.

+ Second order: Kẻ tấn công sẽ để lại một đoạn mã độc trên ứng dụng để log lại dữ liệu. Mã độc này sẽ kích hoạt để gửi các thông tin về máy chủ của kẻ tấn công.

+ Lateral attack: Kẻ tấn công có thể lợi dụng một chút sáng tạo khi sửa một vài biến môi trường (Environment Variable) để thao túng cơ sở dữ liệu

**1.4.2. Blind SQL Injection**

Tấn công SQL injection ‘mù’ được sử dụng khi mà một ứng dụng web được cấu hình không hiển thị thông báo lỗi hay chỉ hiển thị những thông báo lỗi chung chung, không hiển thị ra lỗi của SQL. Hai biến thể chính thực hiện tấn công Blind SQL injection là: Tấn công dự vào nội dung phản hồi và tấn công dựa vào độ trễ thời gian phản hồi.

***Tấn công dựa vào nội dung phản hồi:*** Trong trường hợp tấn công Blind SQL injection dựa trên nội dung, kẻ tấn công thực hiện các truy vấn SQL khác nhau để hỏi các câu hỏi TRUE hoặc FALSE của cơ sở dữ liệu. Sau đó, họ phân tích sự khác biệt trong câu trả lòi giữa các câu lệnh TRUE và FALSE.

Ví dụ:

[www.shop-online.com/product\_detail.php?id=1](http://www.shop-online.com/product_detail.php?id=1)

Liên kết trên sẽ hiển thị thông tin chi tiết của sản phẩm có id=1 được lấy từ cơ sở dữ liệu. Câu lệnh SQL cho liên kết này khi được yêu cầu là:

SELECT \* FROM products

WHERE id = 1

Hoặc

SELECT column\_name, column\_name\_2 FROM table\_name

WHERE id = 1

Kẻ tấn công sẽ sửa lại truy vấn bằng các sửa url thành:

[www.shop-online.com/product\_detail.php?id=1](http://www.shop-online.com/product_detail.php?id=1) and 1=2

Câu lệnh SQL sẽ đổi thành:

SELECT \* FOR products

Where id = 1 and 1=2

Hoặc

SELECT column\_name, column\_name\_2

FROM table\_name

WHERE id = 1 and 1 = 2

Điều này sẽ khiến truy vấn trả về sai và không có mục nào được hiển thị trong thông tin sản phẩm. Kẻ tấn công sau đó tiến hành thay đổi yêu cầu thành

[www.shop-online.com/product\_detail.php?id=1](http://www.shop-online.com/product_detail.php?id=1) and 1=1

Và câu lệnh SQL lúc này lại đổi thành:

SELECT \* FOR products

Where id = 1 and 1=1

Hoặc

SELECT column\_name, column\_name\_2

FROM table\_name

WHERE id = 1 and 1 = 1

Điều này trả về TRUE và các chi tiết của sản phẩm có id=1 hiển thị. Đây là một dấu hiệu rõ ràng rằng ứng dụng này đang gặp lỗi SQL injection và chúng tôi đã dùng Blind SQL injection để tấn công.

***Tấn công dựa vào độ trễ thời gian phản hồi.***Trong trường hợp này, các cuộc tấn công dựa trên thời gian, kẻ tấn công làm cho cơ sở dữ liệu thực hiện một hành động tốn thời gian. Nếu ứng dụng không trả về phản hồi ngay lập tức, ứng dụng này dễ bị Blind SQL injection. Một hành động phổ biến dùng cho thời gian là sleep.

Dựa trên ví dụ trước, kẻ tấn công trước tiên sẽ đánh giá thời gian phản hồi của máy chủ web cho một truy vấn thông thường. Yêu cầu kẻ tấn công đưa ra là:

www.shop-online.com/product\_detail.php?id=1 and if(1=1, sleep(10), false)

Nếu ứng dụng gặp lỗi SQL injection thì phản hồi sẽ bị trễ 10 giây.

**1.4.3. Against Database**

Một số kĩ thuật tấn công Against Database:

***Inline injection:*** Mục đích của một cuộc tấn công là đưa mã vào một hoặc nhiều câu lệnh điều kiện luôn đúng. Trong loại tấn công SQL này, kẻ tấn công khai thác mệnh đề WHERE, tức là các truy vấn luôn trả về kết quả khi đánh giá tham số điều kiện tại mệnh đề WHERE

Ví dụ: Kẻ tấn công có thể nhập chuỗi ’ or ‘1’ = ‘1 . Truy vấn kết quả là:

“SELECT \*

FROM user\_account

WHERE user\_name = ‘adm’ AND password = ‘’ or ‘1’ = ‘1’;”

Khi thực thi truy vấn, mệnh đề ‘1’ = ’1’ là luôn đúng nên cơ sở dữ liệu sẽ trả về kết quả là toàn bộ dữ liệu trong bảng user\_account mà không cần quan tâm đến user\_name và password. Như vậy là kẻ tấn công có thể có dữ liệu truy cập của toàn bộ người dùng. Với trường hợp đăng nhập hệ thống, ở đây kẻ tấn công đã vượt qua được xác thực.

***Chú thích cuối dòng:*** Kẻ tấn công lợi dụng việc cơ sở dữ liệu sẽ bỏ qua chuỗi đằng sau chú thích dòng, ví dụ như: -- hay # khi truy vấn. Bằng cách chèn dấu chú thích vào vị trí thích hợp của truy vấn, kể tấn công sẽ lừa được cơ sở dữ liệu khi truy vấn.

Ví dụ:

User\_name : admin’ or 1=1--

Password : admin

Khi ta submit thì câu lệnh sql trên server sẽ là

“SELECT \*

FROM user\_account

WHERE user\_name = ‘admin’ or 1 = 1 --‘ AND password = ‘test’;”

Như vậy bằng việc bỏ qua phần chuỗi sau “- -” thì câu truy vấn trên luôn đúng đúng.

***Truy vấn UNION:*** UNION cho phép thực một hay nhiều câu lệnh truy vấn SELECT và bổ sung kết quả đến câu truy vấn gốc.

Ví dụ:

SELECT a, b FROM table1

UNION

SELECT c,d FROM table2

Câu lệnh truy vấn này sẽ trả về kết quả là hai cột chứa giá trị từ cột a, b của bảng table1 và côt c,d của bảng table2

Để thực thi thành công đòi hỏi hai yêu cầu: Các truy vấn tương ứng phải có cùng cố cột và Kiểu dữ liệu mỗi cột phải tương thích với nhau giữa các truy vấn tương ứng.

Để thực hiện SQL injection bằng UNION thì cần: Câu truy vấn gốc trả về bao nhiêu cột . Sau đó ta có thế xem được loại dữ liệu của cột hoặc version(), loại của database, hoặc có thế thực hiện các tấn công khác.

**Bước 1:** Xác định số cột thì ta sẽ thực hiện hai cách như sau:

*Cách thứ nhất* : sử dụng ORDER BY kết hợp với chỉ số

Ví dụ: Ta sẽ thực hiện lần lượt các truy vấn:

‘ ORDER BY 1 --

‘ ORDER BY 2 –

‘ORDER BY 3 –

…………

Dựa vào kết quả được trả về sau mỗi lần thực hiện thì kẻ tấn công sẽ xác định sự khác nhau giữa các lần thực hiện truy vấn từ đó xác định được số cột của database

Cách thứ hai : Sử dụng UNION SELECT kết hợp với giá trị NULL

Ví dụ:Ta sẽ thực hiện truy vấn

UNION SELECT NULL—

UNION SELECT NULL, NULL - -

UNION SELECT NULL , NULL, NULL - -

……….

Dựa vào kết quả trả về đến khi nào số cột của bảng bằng với số giá trị NULL thì kẻ tấn công sẽ xác định được số cột của bảng .

**Bước 2:** Có thế xác định được dữ liệu của cột có phải string không. Gỉa sử ta tìm được 3 cột

‘ UNION SELECT ‘a’, NULL, NULL- -

‘ UNION SELECT NULL, ‘a’,NULL- -

‘ UNION SELECT NULL, NULL, ‘a’- -

Dựa vào kết quả trả về thì kẻ tấn công sẽ xác định được kiểu dữ liệu của từng cột có phải là string hay không.

Tím được kiểu dữ liệu của các cột hoặc xác định được cột kẻ tấn công có thể thu thập được những thông tin quan trọng bằng cách dựa vào thông tin của database, tên các bảng và tên các cột của bảng.

**Chương 2: Phương pháp chống tấn công SQL Injection**

Như vậy, có thể thấy lỗi SQL injection khai thác bất cẩn của các lập trình viên phát triển ứng dụng web khi xử lý các dữ liệu nhập avof để xây dựng câu lệnh SQL. Tác hại SQL injection tùy thuộc vào môi trường và cách cấu hình hệ thống. Kẻ tấn công có thế xóa toàn bộ các bảng dữ liệu, tạo các bảng dữ liệu mới hoặc tạo các tài khoản người dùng bất hợp pháp để điều khiển hệ thống của bạn. Để phòng tránh thì ta có thể thực hiện những cách như sau:

**2.1. Kiểm soát chặt chẽ dữ liệu nhập vào**

Tất cả các form từ người dùng nhập lên cần phải được kiểm duyệt một cách nghiêm chỉnh. Nó không chỉ bao gồm những input boxes như kiểu textarea hay input text mà nó còn bao gồm rất nhiều các khác ví dụ như hidden input, query string parameters, cookies và file uploads. đảm bảo các thuộc tính dữ liệu phù hợp như kiểu/ loại, độ dài, định dạng,... Chỉ xử lý dữ liệu đầu vào đã qua xác thực cho CSDL. Với những kiểm soát đơn giản như kiểm soát về kiểu dữ liệu cũng có thể hạn chế đáng kể các cuộc tấn công

**Validate dữ liệu trên server side.** Validation là quá trình để đảm bảo dữ liệu người dùng gửi lên là hợp lệ. Ví dụ ở trong PHP, bạn có thể dùng hàm mysql\_real\_escape\_string() để loại bỏ những kí tự có thể gây ảnh hưởng đến câu lệnh SQL.

**2.2. Tham số hóa truy vấn**

Sử dụng tham số hóa truy vấn là một trong những cách tốt để chống lại SQL injection. Với phương pháp này cho phép nhà phát triển xác định chính xác cấu trúc của câu truy vấn và truyền các tham số giá trị một cách chính xác. Nếu người dùng nhập vào trường username là “tuan” và password là “or 1=1 - -” thì trong database sẽ tìm dữ liệu khớp hoàn toàn với username là “tuan” và password là “or 1=1 - -”, chính điều này làm cho cấu trúc câu truy vấn không bị thay đổi bởi bất kì đầu vào nào. Trong PHP đó là cơ chế prepared thực hiện theo nguyên tắc:

* Chúng ta đưa vào một câu truy vấn với các tham số là một ẩn danh
* Chúng ta truyền vào giá trị tương ứng cho các ẩn danh đó
* PHP sẽ dựa vào thư tự các tham số ẩn danh và các giá trị để repared sao bảo mật nhất.
* Cuối cùng sẽ thực thi câu truy vấn.
* Khi bạn đã khai báo các tham số lần đầu rồi và sau đó muốn sử dụng tiếp thì không cần phải khai báo nữa. Đây cũng chính là lợi thế của cơ chế prepared trong PHP.

Ví dụ:

$conn = new mysqli(‘localhost’, ‘root’, ‘ ‘, ‘dbase\_new’;

$sql = “INSERT INTO DBNEW (username, password) VALUES (?, ?)

$stmt = $conn->prepare($sql);

$stmt->blind\_param(“ss”, $username, $password);

$username = ‘tuan’;

$password = ‘123456’

$stmt->execute();

**2.3. Mã hóa dữ liệu nhạy cảm**

Hãy mà hóa những dữ liệu nhạy cảm trong cơ sở dữ liệu của bạn. Nó bao gồm mật khẩu, câu hỏi và câu trả lời bí mật, các dữ liệu về tài chính, thông tin sức khỏe và những thông tin có ích cho hacker. Nó đảm bảo rằng nếu mà thông tin mà có lộ ra ngoài thì tin tặc cũng không thể sử dụng nó nhằm mục đích gây hại cho khách hàng của chúng ta.

**2.4. Ẩn thông tin của các thông báo**

Thông báo lỗi rất hữu ích cho những kẻ tấn công tìm hiểu thêm về kiến trúc CSDL. Trong trường hợp tấn công suy đoán, kẻ tấn công có thể lợi dụng các thông báo lỗi để hoàn thiện kiến thức, phương án tấn công. Vì vậy thông báo lỗi nên chỉ hiển thị các thông tin cần thiết. Tốt hơn hết là hiển thị thông báo lỗi chung cho biết có lỗi xảy ra và khuyến khích người dùng liên hệ với nhóm hỗ trợ kỹ thuật trong trường hợp sự cố vẫn còn.

**2.5.Tránh quyền quản trị**

Không kết nối với cơ sở dữ liệu của bằng tài khoản có quyền truy cập root trừ khi được yêu cầu vì những kẻ tấn công có thể có quyền truy cập vào toàn bộ hệ thống. Do đó, tốt nhất là sử dụng một tài khoản có các đặc quyền hạn chế để giới hạn phạm vi thiệt hại trong trường hợp bị tấn công SQL injection.

**2.6. Phân tích mã**

Sử dụng kiểm thử để phát hiện ra lỗ hổng của ứng dụng. Bộ kiểm thử sẽ sinh ra một loạt các dạng tấn công SQL injection nhằm kiểm tra phản hồi của ứng dụng web. Dựa vào kết quả trả về của bộ kiểm thử, nhà phát triển có thể xác định các lỗ hổng trên ứng dụng và tìm cách khắc phục các lỗ hổng này. Một số bộ kiểm thử thông dụng với tấn công SQL injector như SQLMap, Acunetix, Burp suite, Netsparker, …

**2.7. Sử dụng các dịch vụ nhạy cảm bằng API**

Sử dụng các dịch vụ nhạy cảm bằng API trên những máy chủ khác ngoài máy chủ web, nhằm tránh cho máy chủ chính bị tấn công, khi