ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

NGUYỄN THANH LIÊM

CHỐNG TẤN CÔNG TIÊM NHIỄM SQL SỬ DỤNG CÁC KHUÔN MẪU HỢP LỆ THEO BỐI CẢNH

LUẬN VĂN THẠC SĨ CÔNG NGHỆ THÔNG TIN

Hà Nội – 2017

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

NGUYỄN THANH LIÊM

CHỐNG TẤN CÔNG TIÊM NHIỄM SQL SỬ DỤNG CÁC KHUÔN MẪU HỢP LỆ THEO BỐI CẢNH

Ngành: Công nghệ thông tin
Chuyên ngành: Truyền dữ liệu và Mạng máy tính
Mã số: Chuyên ngành đào tạo thí điểm

LUẬN VĂN THẠC SĨ CÔNG NGHỆ THÔNG TIN

NGƯỜI HƯỚNG DẪN KHOA HỌC: TS. NGUYỄN ĐẠI THỌ

Hà Nội – 2017

Lời cam đoan:

Những kết quả nghiên cứu được trình bày trong luận văn là hoàn toàn trung thực, của tôi, không vi phạm bất cứ điều gì trong luật sở hữu trí tuệ và pháp luật Việt Nam. Nếu sai, tôi xin chịu trách nhiệm hoàn toàn trước pháp luật.

TÁC GIẢ LUẬN VĂN

Nguyễn Thanh Liêm

Danh mục các ký hiệu và từ viết tắt

STT	Ký hiệu	Ý nghĩa
1	AMNESIA	Kỹ thuật chống tấn công tiêm nhiễm AMNESIA (Analysis and Monitoring for Neutralizing SQL-Injection Attacks)
2	API	Giao diện lập trình ứng dụng (Application Programming Interface)
3	CSDL	Cơ sở dữ liệu
4	SQL	Ngôn ngữ truy vấn có cấu trúc (Structured Query Language)

Danh mục các hình vẽ

Danh mục bảng, biểu

MỤC LỤC

MỞ ĐẦU

Ngày nay, ứng dụng web trở nên thiết yếu trong cuộc sống hàng ngày của con người. Giao dịch ngân hàng online, mua bán online, mạng xã hội, blog… đều là những ứng dụng web khá phổ biến và chúng cũng là những mục tiêu ưa thích của tin tặc. Trong các cuộc tấn công mạng thì tấn công mà mục tiêu là ứng dụng web là phổ biến nhất và gây ra nhiều thiệt hại cho các tổ chức và cá nhân. Đặc biệt là dạng tấn công tiêm nhiễm SQL. Tiêm nhiễm SQL là một kĩ thuật cho phép những kẻ tấn công lợi dụng lỗ hổng trong việc kiểm tra dữ liệu nhập trong các ứng dụng web và các thông báo lỗi của hệ quản trị cơ sở dữ liệu, để tiêm nhiễm (inject) và thi hành các câu lệnh SQL trái phép (không được người phát triển ứng dụng lường trước). Hậu quả của nó rất tai hại vì nó cho phép những kẻ tấn công có thể thực hiện các thao tác xóa, hiệu chỉnh, … do có toàn quyền trên cơ sở dữ liệu của ứng dụng, thậm chí là server mà ứng dụng đó đang chạy.

Các nhà nghiên cứu đã đề xuất nhiều phương pháp phòng chống tấn công tiêm nhiễm SQL, nhưng nhìn chung có thể chia thành hai loại là phương pháp thực hành mã phòng thủ (defensive coding pratices) và phương pháp phát hiện và ngăn ngừa (dection and prevention techniques). Thực hành mã phòng thủ là dựa vào người phát triển ứng dụng cố gắng loại bỏ nguy cơ từ mã nguồn. Phương pháp này phụ thuộc nhiều vào sự chủ quan của người phát triển ứng dụng, cũng khó có thể đảm bảo là ứng dụng web không có lỗ hổng và sự thay đổi của ứng dụng web. Trong khi đó phương pháp phát hiện và ngăn ngừa lại tập trung vào việc tự động phát hiện tấn công tiêm nhiễm SQL và ngăn chặn chúng. Các nhà nghiên cứu đã đề xuất một loạt các kỹ thuật chống tấn công tiêm nhiễm SQL theo phương pháp này. Nhìn chung các kỹ thuật này đều dựa vào việc xây dựng tập hợp những tình huống tấn công có thể có hoặc xây dựng tập hợp những câu truy vấn hợp lệ. Với sự đa dạng và sự phát triển nhanh chóng của các dạng tấn công tiêm nhiễm SQL thì rất khó có thể xác định được tập hợp các cuộc tấn công tiêm nhiễm SQL. Trong khi đó để xác định được những truy vấn hợp lệ lại dễ dàng hơn nhiều.

Dr. Dimitris Mitropoulos và Prof. Diomidis Spinellis đã đề xuất kỹ thuật chống tấn công tiêm nhiễm SQL bằng khuôn mẫu hợp lệ theo bối cảnh (location-specific signatures prevent SQL injection attack). Một trình điều khiển được gọi là SDriver sẽ được thêm vào giữa ứng dụng web và trình điều khiển kết nối tới cơ sở dữ liệu. SDriver sẽ chịu trách nhiệm phát hiện và ngăn chặn tấn công tiêm nhiễm SQL thông qua một cơ sở dữ liệu những câu truy vấn hợp lệ được gắn với bối cảnh. Do đó SDriver là một kỹ thuật đơn giản và hiệu quả

trong việc chống tấn công tiêm nhiễm SQL. Hơn nữa khi triển khai kỹ thuật này chỉ việc thay một mã rất ít ở phần kết nối tới cơ sở dữ liệu và không ảnh hưởng nhiều tới hiệu năng của hệ thống. Tuy vậy phương pháp này vẫn tồn tại vấn đề khiến kẻ tấn công có thể tiêm nhiễm thành công.

Xuất phát từ thực tế đó, luận văn tập trung nghiên cứu*: "Chống tấn công tiêm nhiễm SQL sử dụng các khuôn mẫu hợp lệ theo bối cảnh".*

Nội dung của luận văn gồm năm chương:

Chương 1: Tổng quan về tấn công tiêm nhiễm SQL và các phương pháp phòng chống. Trong chương này sẽ đề cập đến lý thuyết về tấn công tiêm nhiễm SQL, các phương pháp phòng chống một cách tổng quát.

Chương 2: Phương pháp chống tấn công tiêm nhiễm SQL sử dụng các khuôn mẫu hợp lệ theo bối cảnh - SDriver. Chương này sẽ trình bày về hiện trạng hiện nay của phương pháp chống tấn công tiêm nhiễm SQL sử dụng các khuôn mẫu hợp lệ theo bối cảnh. Phân tích chỉ ra vấn đề còn tồn tại.

Chương 3: Đề xuất cải tiến chống tấn công tiêm nhiễm SQL sử dụng các khuôn mẫu hợp lệ theo bối cảnh. Đưa ra được phương pháp cải tiến chống tấn công tiêm nhiễm SQL sử dụng các khuôn mẫu hợp lệ theo bối cảnh.

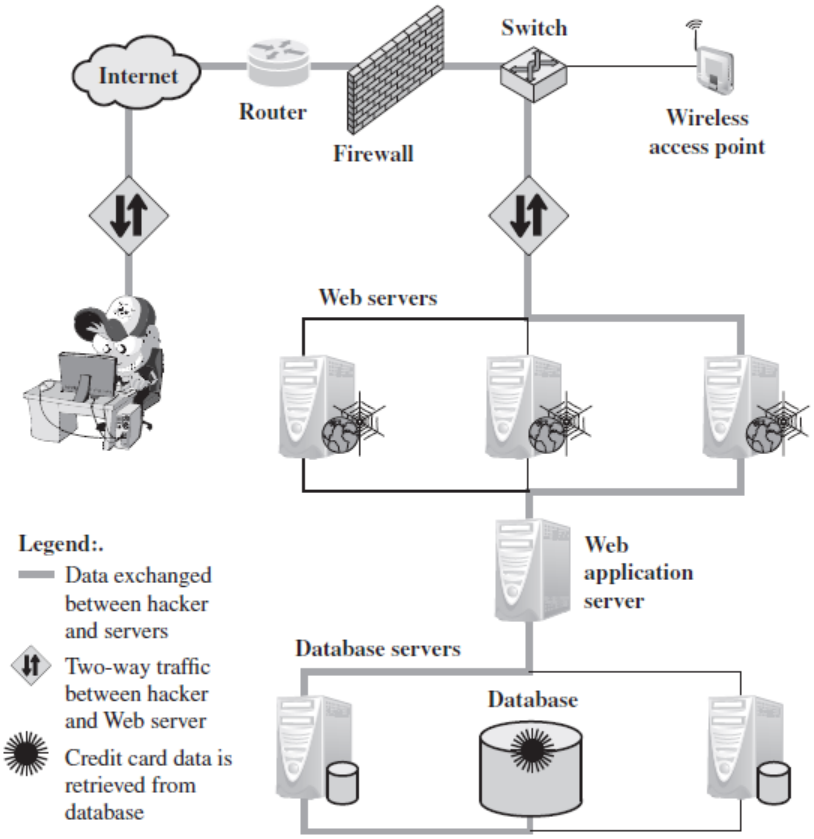
Chương 4: Kết quả thực nghiệm đánh giá. Chương này sẽ trình bày về kết quả thực nghiệm phương pháp cải tiến và đưa ra được đánh giá về phương pháp cải tiến.

CHƯƠNG 1 TỔNG QUAN VỀ TẤN CÔNG TIÊM NHIỄM SQL VÀ CÁC PHƯƠNG PHÁP PHÒNG CHỐNG

1.1. Khái niệm tấn công tiêm nhiễm SQL.

Theo OWASP (Open Web Application security Project), tấn công tiêm nhiễm SQL là một mối đe dọa an ninh phổ biến và nguy hiểm nhất [2]. Tấn công tiêm nhiễm SQL là một dạng tấn công tiêm nhiễm mã độc mà kẻ tấn công sẽ cố gắng khai thác lỗ hổng của chính ứng dụng web để tiến hành tiêm nhiễm mã độc vào câu truy vấn SQL của ứng dụng web nhằm truy cập trái phép vào cơ sở dữ liệu đằng sau ứng dụng web. Khi đó kẻ tấn công có thể lấy cắp được thông tin của khách hàng, người dùng, trong đó có không ít thông tin nhạy cảm như tên, tuổi, địa chỉ, số tài khoản… Tùy vào môi trường, kẻ tấn công thậm chí còn có thể thực hiện các thao tác sửa, xóa dữ liệu, thực thi các câu lệnh của hệ điều hành hay thực hiện một cuộc tấn công từ chối dịch vụ. Tấn công tiêm nhiễm SQL thường được sử dụng để khai thác lỗ hổng của các ứng dụng web có hệ quản trị cơ sở dữ liệu quan hệ như MySQL, MS SQL, DB2, Oracle…

Hình 1.1 từ [4, pp. 164] dưới đây là một ví dụ điển hình cho một cuộc Tấn công tiêm nhiễm SQL. Các bước thực hiện như sau:



Hình 1. 1 Một cuộc Tấn công tiêm nhiễm SQL điển hình.

Bước 1: Kẻ tấn công sẽ dò quét lỗ hổng của ứng dụng web. Khi phát hiện lỗ hổng, hắn sẽ tiêm nhiễm câu lệnh SQL vào cơ sở dữ liệu (CSDL) bằng cách gửi câu lệnh đến máy chủ web (Web server). Câu lệnh trái phép này sẽ được tiêm nhiễm vào luồng thông tin của ứng dụng web và nó sẽ vượt qua được tường lửa (Firewall).

Bước 2: Máy chủ web tiếp nhận mã độc và gửi nó đến máy chủ ứng dụng web (Web application server).

Bước 3: Tại máy chủ ứng dụng web, vì kẻ tấn công khai thác lỗ hổng của chính ứng dụng web nên mã độc sẽ tiếp tục được chấp nhận và gửi đến máy chủ CSDL (database).

Bước 4: Máy chủ CSDL sẽ thực thi đoạn mã độc và trả về dữ liệu chứa thông tin nhạy cảm như thông tin thẻ tín dụng.

Bước 5: Máy chủ ứng dụng web sẽ tạo ra các nội dung động bao gồm thông tin nhạy cảm về thẻ tín dụng.

Bước 6: Máy chủ web sẽ gửi toàn bộ thông tin trên cho kẻ tấn công.

Trên đây là các bước tấn công của một cuộc Tấn công tiêm nhiễm SQL điển hình, kẻ tấn công chỉ nhằm mục đích ăn cắp thông tin nhạy cảm của khách hàng, người dùng. Thay vì chỉ ăn cắp thông tin, kẻ tấn công có thể phá hủy CSDL, hay đánh sập cả ứng dụng web bằng cách tiêm nhiễm các đoạn mã độc nguy hiểm.

1.2. Phân loại tấn công tiêm nhiễm SQL.

Có nhiều loại Tấn công tiêm nhiễm SQL khác nhau, Tấn công tiêm nhiễm SQL có thể phân loại theo các tiêu chí như cơ chế tiêm nhiễm, mục đích tấn công và kỹ thuật tấn công [5].

1.2.1. Cơ chế tiêm nhiễm.

Câu lệnh SQL độc hại có thể được tiêm nhiễm theo nhiều cơ chế khác nhau, sau đây là một số cơ chế thông dụng nhất:

Tiêm nhiễm thông qua nhập liệu của người dùng: Kẻ tấn công tiêm nhiễm mã độc vào trường nhập liệu của người dùng. Ứng dụng web có thể đọc nhập liệu người dùng bằng nhiều cách khác nhau dựa trên môi trường mà nó được triển khai. Trong hầu hết các cuộc Tấn công tiêm nhiễm SQL có mục tiêu là ứng dụng web, mã độc tiêm nhiễm vào nhập liệu người dùng qua form xác nhận (form submissions) và được gửi về ứng dụng web qua yêu cầu HTTP GET và HTTP POST.

Tiêm nhiễm thông qua cookies: Khi một máy khách trả về ứng dụng web, cookies có thể được sử dụng để lưu trữ thông tin trạng thái của máy khách. Vì máy khách có thể kiểm soát được hoàn toàn cookies, kẻ tấn công có thể thay đổi được cookies. Khi ứng dụng web sử dụng nội dung của cookies trên máy khách để xây dựng câu truy vấn thì với cookies đã bị thay đổi, câu truy vấn cũng có thể bị thay đổi theo.

Tiêm nhiễm Second-order: Cơ chế tiêm nhiễm này hết sức tinh vi. Với tiêm nhiễm Second-order, kẻ tấn công sẽ dựa vào dữ liệu đã có trong hệ thống hoặc CSDL để kích hoạt Tấn công tiêm nhiễm SQL. Nên khi cuộc tấn công xảy ra, câu truy vấn bị thay đổi không phải từ người dùng mà từ chính bản thân hệ thống.

1.2.2. Mục đích tấn công.

Một cuộc Tấn công tiêm nhiễm SQL có thể có nhiều mục đích tấn công cùng lúc. Nhìn chung, mục đích của Tấn công tiêm nhiễm SQL có thể phân loại như sau:

Xác định các tham số có thể tiêm nhiễm: Kẻ tấn công sẽ thăm dò ứng dụng web nhằm khám phá ra các tham số hay trường nhập liệu người dùng có thể là lỗ hổng cho Tấn công tiêm nhiễm SQL.

Thực hiện tìm vết CSDL: Kẻ tấn công muốn tìm kiếm thông tin về kiểu và phiên bản CSDL mà ứng dụng web đang sử dụng. Tùy theo kiểu CSDL sẽ có phản ứng khác nhau đối với các dạng câu truy vấn và tấn công khác nhau. Và từ những thông tin phản hồi trên, kẻ tấn công có thể tìm ra được kiểu và phiên bản của CSDL. Kiểu và phiên bản của CSDL có thể cho phép kẻ tấn công tìm ra được cách tấn công cụ thể chuyển nhằm vào CSDL.

Xác định lược đồ CSDL: Để trích xuất dữ liệu chính xác từ CSDL, kẻ tấn công cần biết được thông tin về lược đồ CSDL như tên bảng, tên cột và kiểu dữ liệu. Kẻ tấn công có thể thu thập hoặc suy luận những thông tin về lược đồ CSDL.

Trích xuất dữ liệu: Với mục đích tấn công này, kẻ tấn công mong muốn đánh cắp được dữ liệu từ CSDL. Tùy vào loại ứng dụng web mà dữ liệu có thể chứa những thông tin nhạy cảm, bí mật của khách hàng, người dùng hay của tổ chức. Phần lớn kẻ tấn công vào ứng dụng web có mục đích này.

Thêm hoặc thay đổi dữ liệu: Mục tiêu của kẻ tấn công là chèn thêm thông tin hoặc thay đổi thông tin của CSDL.

Thực hiện từ chối dịch vụ: Những kẻ tấn công với mục đích phá hoại bằng cách thực hiện dừng CSDL của ứng dụng web, khiến người dùng không thể sử dụng được. Ngoài ra, kẻ tấn công thực hiện xóa bảng dữ liệu, khóa bảng trong CSDL, làm tê liệt khả năng hoạt động của CSDL cũng thuộc loại này.

Tránh né phát hiện: Kẻ tấn công sử dụng các kỹ thuật tấn công nhằm tránh né sự phát hiện của cơ chế bảo vệ hệ thống.

Vượt qua xác thực: Mục tiêu của dạng tấn công này là cho phép kẻ tấn công có thể vượt qua được cơ chế xác thực của CSDL và ứng dụng. Khi thực hiện thành công, kẻ tấn công sẽ được hưởng các quyền mà được gắn với người dùng khác.

Thực thi câu lệnh từ xa: Kẻ tấn công sẽ cố gắng thực thi các câu lệnh tùy ý trên CSDL. Những câu lệnh này có thể được lưu trữ các thủ tục hay các hàm cho CSDL người dùng.

Thực hiện leo thang đặc quyền: Kẻ tấn công lợi dụng những sai sót logic trong khi thực thi của CSDL để tiến hành chiếm những đặc quyền. Trái với tấn công vượt qua xác thực, kiểu tấn công này tập trung vào việc khai thác những đặc quyền của CSDL người dùng.

1.2.3. Kỹ thuật tấn công.

Có rất nhiều kỹ thuật Tấn công tiêm nhiễm SQL khác nhau nhưng chúng ít khi được sử dụng một cách đơn độc mà thường được sử dụng kết hợp hay tuần tự, tùy theo mục tiêu cụ thể của kẻ tấn công. Dưới đây là các kỹ thuật Tấn công tiêm nhiễm SQL tiêu biểu:

Tautologies: Với kỹ thuật này, kẻ tấn công sẽ tiêm nhiễm mã độc vào câu truy vấn có điều kiện và khiến điều kiện của câu truy vấn trở thành luôn đúng. Sau đây là một câu truy vấn hợp lệ, chức năng của nó là xác thực người dùng:

```
"SELECT accounts FROM users WHERE login=' + $login + ' AND pass=' + $pass + '";
```

Kẻ tấn công có thể nhập "`' or 1=1--`" vào trường login, khiến câu truy vấn hợp lệ trên trở thành:

```
"SELECT accounts FROM users WHERE login='' or 1=1-- AND pass='';
```

Khi thực thi câu truy vấn đã bị thay đổi trên, CSDL sẽ bỏ qua chuỗi ký tự đằng sau dấu chú thích "`--`", và với điều kiện "`or 1=1`" thì điều kiện của câu truy

vấn sẽ luôn đúng. Tức là CSDL dữ liệu sẽ luôn trả về kết quả đối với điều kiện luôn đúng như vậy, từ đó kẻ tấn công có thể vượt qua xác thực của CSDL.

Chú thích cuối dòng (End-of-Line Comment): Kẻ tấn công lợi dụng việc CSDL sẽ bỏ qua chuỗi đằng sau dấu chú thích dòng, ví dụ như "`--`", khi thực thi truy vấn. Bằng cách chèn dấu chú thích vào vị trí thích hợp của câu truy vấn, kẻ tấn công sẽ lừa được CSDL thực thi câu truy vấn đã bị thay đổi. Ví dụ trên cũng là một ví dụ điển hình cho kỹ thuật tấn công này khi kẻ tấn công đã thêm vào dấu chú thích "`--`", làm CSDL bỏ qua phần "`AND pass=''`" khi thực thi câu truy vấn.

Truy vấn Union (Union query): Với kỹ thuật này, kẻ tấn công lợi dụng lỗ hổng tham số để thay đổi bộ dữ liệu trả về cho một câu truy vấn. Với câu truy vấn hợp lệ bên trên, kẻ tấn công có thể tiêm nhiễm vào trường login là "`'` UNION `SELECT cardNo FROM CreditCards WHERE accNo=20301--`". Lúc này câu truy vấn sẽ trở thành:

"`SELECT accounts FROM users WHERE login='' UNION SELECT cardNo FROM CreditCards WHERE accNo=20301-- AND pass='';`

Với câu truy vấn đã bị thay đổi trên, CSDL dù không tìm thấy bản ghi login nào phù hợp với "" nhưng vẫn sẽ tìm thấy bản ghi cardNo phù hợp với "accNo=20301". CSDL sẽ tiến hành hợp 2 bộ kết quả, và cardNo vẫn sẽ được trả về cho ứng dụng.

Truy vấn Piggy-Backed (Piggy-Backed queries): Kỹ thuật tấn công này dựa vào máy chủ CSDL được cấu hình để thực thi nhiều câu truy vấn khác nhau trên cùng 1 dòng mã và được ngăn cách bởi dấu ";". Kẻ tấn công sẽ chèn thêm các câu truy vấn trái phép vào câu truy vấn ban đầu với mục đích trích xuất dữ liệu, thay đổi dữ liệu, thực hiện từ chối dịch vụ hay thực thi lệnh từ xa. Ví dụ, kẻ tấn công có thể nhập mã độc "`'; DROP table users --`" vào trường password của câu truy vấn hợp lệ trên. Câu truy vấn lúc này sẽ là

"`SELECT accounts FROM users WHERE login='' AND pass=''; DROP table users --'";`

Khi thực thi câu truy vấn trên, CSDL sẽ thực thi luôn câu lệnh "`DROP table users`", bảng users sẽ bị xóa.

Suy luận (Inference): Với kỹ thuật này, kẻ tấn công không thực sự trích xuất dữ liệu, thay vào đó hắn sẽ tiến hành suy luận và tái cấu trúc lại thông tin

dữ liệu. Kỹ thuật tấn công này có hai loại là truy vấn trái phép hoặc không đúng logic (illegal/logically incorrect queries) và tiêm nhiễm SQL mù (Blind SQL injection). Với loại tấn công truy vấn trái phép hoặc không đúng logic, kẻ tấn công sẽ cố gắng thu thập thông tin về kiểu và cấu trúc của CSDL. Kẻ tấn công có thể đơn giản nhập vào trường login của câu truy vấn hợp lệ trên ký tự "`'`" hay "`' or 1=0--`" thì sẽ có lỗi khi thực thi. Dựa vào lỗi mà CSDL trả về, kẻ tấn công có thể thăm dò được thông tin của dữ liệu, từ đó xác định được lỗ hổng hay cách thức tấn công phù hợp với kiểu của CSDL.

1.3. Các phương pháp ngăn chặn tấn công tiêm nhiễm SQL.

Các nhà nghiên cứu đã đề xuất nhiều kỹ thuật phòng chống tấn công tiêm nhiễm SQL, nhưng nhìn chung có thể chia thành hai loại kỹ thuật là mã phòng thủ (defensive coding) và kỹ thuật phát hiện và ngăn chặn (detection and prevention techniques).

1.3.1. Phương pháp mã phòng thủ.

Rất nhiều cuộc Tấn công tiêm nhiễm SQL thành công bởi vì các nhà phát triển đã sử dụng mã nguồn không được bảo vệ. Do vậy, mã phòng thủ là cách thức vô cùng hiệu quả để làm giảm đáng kể các mối đe dọa từ Tấn công tiêm nhiễm SQL. Dưới đây là các ví dụ điển hình cho phương pháp mã phòng thủ:

Thực hành mã phòng thủ (Defensive coding practices): Các lỗ hổng thông thường được khai thác bởi Tấn công tiêm nhiễm SQL là do việc kiểm soát đầu vào không được hoàn thiện. Dưới đây là các kỹ thuật thực hành mã phòng thủ thông dụng và rất có hiệu quả:

- Kiểm tra kiểu đầu vào (Input type checking): Tấn công tiêm nhiễm SQL có thể tiêm nhiễm vào cả tham số kiểu chuỗi hay kiểu số. Ngay cả những cách kiểm soát đơn giản cũng có thể hạn chế đáng kể các cuộc tấn cống. Ví dụ như trường nhập liệu là kiểu số, nhà phát triển chỉ cần loại bỏ tất cả kiểu chuỗi hay kiểu ký tự (những ký tự không phải là ký tự số) được nhập vào trường này, thì đã có thể hạn chế đáng kể các cuộc tấn công.
- Mã hóa đầu vào (Endcoding of inputs): Tấn công tiêm nhiễm SQL thường tiêm nhiễm các tham số kiểu chuỗi bằng cách sử dụng các ký tự đặc biệt để đánh lừa trình phân tích cú pháp SQL hiểu nhầm các ký tự đó là mã SQL chứ không phải nhập liệu của người dùng. Dù nhà phát triển có thể cấm nhập các ký tự này, nhưng điều đó cũng làm giới hạn việc nhập liệu của người dùng thông thường. Một giải pháp tốt

hơn là sử dụng hàm mã hóa chuỗi theo cách mà tất cả ký tự đặc biệt sẽ được mã hóa và giải thích giống như các ký tự bình thường khác.

- Phù hợp với khuôn mẫu tích cực (Positive pattern matching): Nhà phát triển có thể thiết lập việc kiểm soát đầu vào để xác định đâu là đầu vào tốt (thông thường) và đâu là đầu vào xấu (độc hại). Cách tiếp cận này thường được gọi là xác nhận hợp lệ, tìm kiếm các mẫu bị cấm hay các mã SQL có trong đầu vào. Dù nhà phát triển khó có thể hình dung được tất cả các dạng tấn công, nhưng họ có thể chỉ định tất cả các mẫu đầu vào hợp lệ. Xác nhận hợp lệ là cách thức an toàn để kiểm tra đầu vào.
- Xác định tất cả nguồn đầu vào (Identification of all input sources): Nhà phát triển phải kiểm tra tất cả đầu vào. Mỗi ứng dụng web đều có rất nhiều nguồn đầu vào khác nhau. Nếu các đầu vào này được sử dụng để cấu trúc câu truy vấn thì rất có thể chúng sẽ trở thành lỗ hổng cho một cuộc Tấn công tiêm nhiễm SQL. Để đảm bảo an toàn, tất cả đầu vào từ các nguồn đều phải được kiểm tra.

Tham số hóa truy vấn (Parameterized query): Phương pháp này nhắm tới việc ngăn chặn Tấn công tiêm nhiễm SQL bằng cách cho phép nhà phát triển có thể xác định chính xác cấu trúc của câu truy vấn và truyền các tham số giá trị cho nó 1 cách tách biệt. Điều đó sẽ ngăn ngừa cấu trúc câu truy vấn bị thay đổi bởi bất kỳ đầu vào nào. Hai kỹ thuật chủ yếu của phương pháp này là sử dụng Prepared Statements và thủ tục lưu trữ (Stored Procedures).

- Prepared Statements: Kỹ thuật này đảm bảo kẻ tấn công không thể thay đổi được mục tiêu của một truy vấn. Trong ví dụ câu truy vấn hợp lệ trên, nếu kẻ tấn công tiêm nhiễm vào trường login chuỗi "`' or 1=1`" thì câu truy vấn sẽ tìm kiếm bản ghi login phù hợp với toàn bộ chuỗi "`' or 1=1`". Sau đây là một số ngôn ngữ lập trình có áp dụng kỹ thuật này:
  - Ngôn ngữ Java EE – sử dụng PreparedStatement() với các biến số bắt buộc.
  - Ngôn ngữ .NET – sử dụng các truy vấn tham số như SqlCommand() hoặc OleDbCommand() với các biến số bắt buộc.
  - Ngôn ngữ PHP – sử dụng PDO với các loại truy vấn tham số bền vững bindParam().
- Thủ tục lưu trữ: Các nhà phát triển có thể xây dựng các mã SQL cho thủ tục lưu trữ và lưu trữ chúng trong CSDL riêng của chúng. Sau đó các thủ tục lưu trữ này sẽ được gọi tại ứng dụng. Khi xây dựng thủ tục

lưu trữ, các nhà phát triển vẫn cần phải tiến hành xác nhận các chuẩn đầu vào để loại bỏ mã SQL độc hại được tiêm vào đầu vào.

- SQL DOM: SQL DOM là bộ các lớp mà cho phép kiểm tra kiểu dữ liệu một cách tự động và kiểu ký tự thoát (escaping chars). Phương pháp này sử dụng việc đóng gói các truy vấn CSDL. Điều này làm thay đổi quá trình xây dựng câu truy vấn từ một quá trình không được kiểm soát sang một hệ thống có sử dụng API kiểm tra kiểu. Trong API, các nhà phát triển có thể áp dụng một cách có hệ thống phương pháp thực hành mã phòng thủ tốt nhất, ví như lọc đầu vào hay kiểm tra chặt chẽ kiểu dữ liệu của nhập liệu người dùng.

Dù phương pháp mã phòng thủ khá hiệu quả trong việc ngăn chặn Tấn công tiêm nhiễm SQL nhưng trong thực tế không thể tránh khỏi lỗi do con người. Phương pháp mã phòng thủ phụ thuộc vào ý thức chủ quan của con người, là các nhà phát triển ứng dụng, do đó sẽ xuất hiện những lỗ hổng do lỗi lập trình. Dù các nhà phát triển đều nỗ lực tạo ra các mã an toàn nhưng thực sự rất khó để có thể áp dụng mã phòng thủ một cách nghiêm ngặt và chính xác cho tất cả đầu vào.

1.3.2. Phương pháp phát hiện và ngăn chặn.

Các nhà phát triển đã nghiên cứu và đã đưa ra một loạt các đề xuất cho phương pháp phát hiện và ngăn ngừa nhằm hỗ trợ cho nhà phát triển bù đắp những thiếu sót trong phương pháp mã phòng thủ. Dưới đây là một số phương pháp phát hiện và ngăn ngừa chủ yếu:

Signature based: Giống với một phương pháp phát hiện mã độc của các chương trình anti-virus, phương pháp này sẽ cố gắng xây dựng tập hợp các mẫu Tấn công tiêm nhiễm SQL có thể có. Vì sử dụng tập hợp mẫu Tấn công tiêm nhiễm SQL nên cần phải thường xuyên cập nhật mẫu Tấn công tiêm nhiễm SQL mới. Nhà phát triển có thể sử dụng kỹ thuật kiểm thử hộp đen (Black Box Testing) để xây dựng tập hợp các khuôn mẫu các Tấn công tiêm nhiễm SQL có thể tấn công vào ứng dụng. Dựa vào tập hợp các khuôn mẫu trên để phát hiện và ngăn chặn các cuộc Tấn công tiêm nhiễm SQL.

Anomaly based: Trái với phương pháp Signature based, phương pháp này sẽ xây dựng các hành vi hợp lệ, bất kỳ mẫu hành vi nào nằm ngoài khoảng hành vi hợp lệ sẽ bị đánh dấu là bất hợp lệ và bị ngăn chặn. Rất nhiều kỹ thuật phát hiện và ngăn chặn Tấn công tiêm nhiễm SQL theo phương pháp này đã được giới thiệu. Dưới đây là một số kỹ thuật điển hình:

- AMNESIA: AMNESIA là một kỹ thuật dựa trên mô hình hóa. Kỹ thuật này kết hợp giữa phân tích tĩnh và giám sát thời gian chạy.

Trong giai đoạn tĩnh, AMNESIA sẽ sử dụng phương thức phân tích tĩnh để xây dựng mô hình truy vấn hợp lệ mà có thể được truy vấn từ các ứng dụng tương ứng. Trong giai đoạn động, AMNESIA sẽ tiếp nhận tất cả câu truy vấn được gửi từ ứng dụng trước khi chúng được gửi đến CSDL. AMNESIA sẽ kiểm tra tất cả câu truy vấn này dựa theo mô hình truy vấn hợp lệ đã được xây dựng trong giai đoạn tĩnh. Bất kỳ câu truy vấn nào được xác định là khác biệt so với mô hình sẽ bị coi là Tấn công tiêm nhiễm SQL và sẽ bị từ chối thi hành, không được gửi đến CSDL.

- Bộ lọc proxy (Proxy Filters): Kỹ thuật này sử dụng các quy tắc để tiến hành xác minh đầu vào trên dòng dữ liệu của ứng dụng web. Bằng cách sử dụng ngôn ngữ mô tả chính sách an ninh (Security Policy Descriptor Language), nhà phát triển có thể xây dựng các ràng buộc và xác định các biến đổi của các tham số ứng dụng khi chúng được truyền từ trang web (phía máy khách) tới máy chủ ứng dụng web.

Code analysis: Phương pháp này liên quan đến việc kiểm thử để phát hiện các lỗ hổng của ứng dụng. Bộ kiểm thử sẽ sinh ra một loạt các dạng Tấn công tiêm nhiễm SQL nhằm kiểm tra phản hồi của ứng dụng web. Một số bộ kiểm thử khá thông dụng như sqlmap, SqlDumper, SQLiX… 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 vá các lỗ hổng này.

1.4. Tóm Tắt.

Nội dung toàn chương một đã nêu lên được các kiến thức cơ bản về Tấn công tiêm nhiễm SQL, khái niệm về Tấn công tiêm nhiễm SQL, các cách thức của Tấn công tiêm nhiễm SQL và các phương pháp ngăn chặn Tấn công tiêm nhiễm SQL. Từ đó có thể nhận thấy:

Tấn công tiêm nhiễm SQL là một dạng tấn công tiêm nhiễm mã độc mà kẻ tấn công sẽ cố gắng khai thác lỗ hổng của chính ứng dụng web để tiến hành tiêm nhiễm mã độc vào câu truy vấn SQL của ứng dụng web nhằm truy cập trái phép vào cơ sở dữ liệu đằng sau ứng dụng web.

Tấn công tiêm nhiễm SQL vô cùng nguy hiểm vì kẻ tấn công không chỉ có thể ăn cắp được dữ liệu chứa thông tin nhạy cảm mà chúng còn có thể thay đổi dữ liệu, thậm chí là kiểm soát cả máy chủ mà CSDL đang chạy. Tấn công tiêm nhiễm SQL xảy ra trên các hệ quản trị CSDL quan hệ như MySQL, MS SQL, DB2, Oracle…

Tấn công tiêm nhiễm SQL có thể phân loại theo các tiêu chí như cơ chế tiêm nhiễm, mục đích tấn công và kỹ thuật tấn công.

Cơ chế tiêm nhiễm gồm: *tiêm nhiễm thông qua nhập liệu người dùng, tiêm nhiễm thông qua cookies, tiêm nhiễm second-order.*

Mục đích tấn công của kẻ tấn công có thể là *xác định các tham số có thể tiêm nhiễm, thực hiện tìm vết CSDL, xác định lược đồ CSDL, trích xuất dữ liệu, thêm hoặc thay đổi dữ liệu, thực hiện từ chối dịch vụ, Tránh né phát hiện, vượt qua xác thực, thực thi câu lệnh từ xa, thực hiện leo thang đặc quyền.*

Kỹ thuật tấn công phổ biến gồm: *tautologies, chú thích cuối dòng, truy vấn Union, truy vấn Piggy-Backed, suy luận.*

Các phương pháp ngăn chặn chủ yếu gồm mã phòng thủ và phương pháp phát hiện và ngăn chặn.

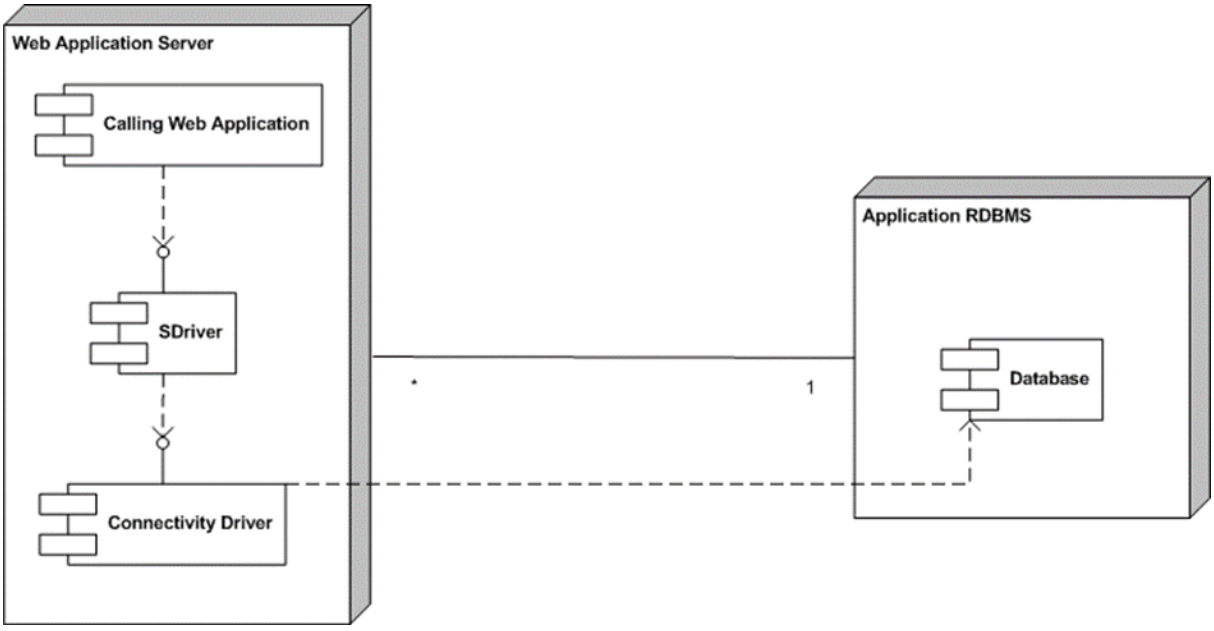
Phương pháp mã phỏng thủ có một số kỹ thuật điển hình sau: *thực hành mã phòng thủ,* t*ham số hóa truy vấn, SQL DOM.*

Phương pháp phát hiện và ngăn chặn được phân thành ba loại sau: *signature based, anomaly based, code analysis.*

CHƯƠNG 2 PHƯƠNG PHÁP SDRIVER TRONG CHỐNG TẤN CÔNG TIÊM NHIỄM SQL

2.1. Phương pháp chống tấn công tiêm nhiễm SQL sử dụng các khuôn mẫu hợp lệ theo bối cảnh, SDriver.

Dr. Dimitris Mitropoulos và Prof. Diomidis Spinellis đã đề xuất kỹ thuật chống tấn công tiêm nhiễm SQL bằng khuôn mẫu hợp lệ theo bối cảnh (location-specific signatures prevent SQL injection attack) [4]. Ý tưởng của kỹ thuật này là dựa vào kiến trúc điển hình của một ứng dụng web bao gồm ít nhất một ứng dụng đang chạy trên một máy chủ web và cơ sở dữ liệu ở phía sau. Giữa hai tầng này, trong hầu hết các trường hợp sẽ có một trình điều khiển kết nối cơ sở dữ liệu dựa trên các giao thức như ODBC (Open Database Connectivity) hoặc JDBC (Java Database Connectivity). Một trình điều khiển được gọi là SDriver sẽ được thêm vào giữa ứng dụng web và trình điều khiển kết nối tới cơ sở dữ liệu. Điểm mấu chốt của SDriver chính là tất cả câu lệnh SQL đều được xác định bởi vị trí câu truy vấn (bối cảnh) và phiên bản rút bỏ dữ liệu đầu vào của câu truy vấn. Các đặc điểm trên của câu truy vấn sẽ được phân tích trong suốt giai đoạn huấn luyện (training phase), từ đó xây dựng mô hình các câu truy vấn hợp lệ. Trong giai đoạn chạy, SDriver sẽ kiểm tra tất cả câu truy vấn được gửi từ ứng dụng web. Câu truy vấn bị đánh dấu là không hợp lệ và bị loại bỏ nếu SDriver không tìm thấy câu truy vấn hợp lệ tương ứng với nó trong mô hình trên.



Hình 2. 1 Kiến trúc đề xuất của SDriver [4, pp. 5].

SDriver là trong suốt và không phụ thuộc vào ứng dụng, CSDL hay trình điều khiển kết nối của ứng dụng. Bản thân SDriver cũng không phải là một trình

điều khiển kết nối mà là trung gian giữa ứng dụng web và trình điều khiển kết nối. SDriver chỉ đóng vai trò phát hiện và ngăn chặn Tấn công tiêm nhiễm SQL.

Khác với các kỹ thuật phát hiện và ngăn chặn Tấn công tiêm nhiễm SQL dựa vào khuôn mẫu hợp lệ khác, SDriver gắn khuôn mẫu hợp lệ với bối cảnh. Bối cảnh ở đây chính là Stack trace. Chi tiết về vai trò của Stack trace sẽ được trình bày ở mục 2.3.

2.2. Cách thức hoạt động của SDriver.

Để phát hiện và ngăn chặn tấn công tiêm nhiễm SQL thì SDriver cần phải được huấn luyện để xây dựng các câu truy vấn hợp lệ. Trong quá trình huấn luyện, tất cả câu truy vấn của ứng dụng web cần được thực thi, SDriver sẽ xác định các đặc trưng của từng câu truy vấn và lưu trữ trong CSDL các mẫu truy vấn hợp lệ. Lưu ý là trong quá trình này, SDriver sẽ giả định là tất cả câu truy vấn là hợp lệ và không có bất kỳ một tấn công tiêm nhiễm SQL nào xảy ra. Do vậy quá trình đào tạo nên được thực hiện trong chế độ không trực tuyến (offline). Sau quá trình huấn luyện, ứng dụng web sẽ được chuyển sang chế độ chạy trực tuyến (chế độ production). Trong chế độ này, SDriver sẽ dựa vào CSDL đã được xây dựng bên trên để phát hiện và loại bỏ tấn công tiêm nhiễm SQL.

2.2.1 Chế độ huấn luyện.

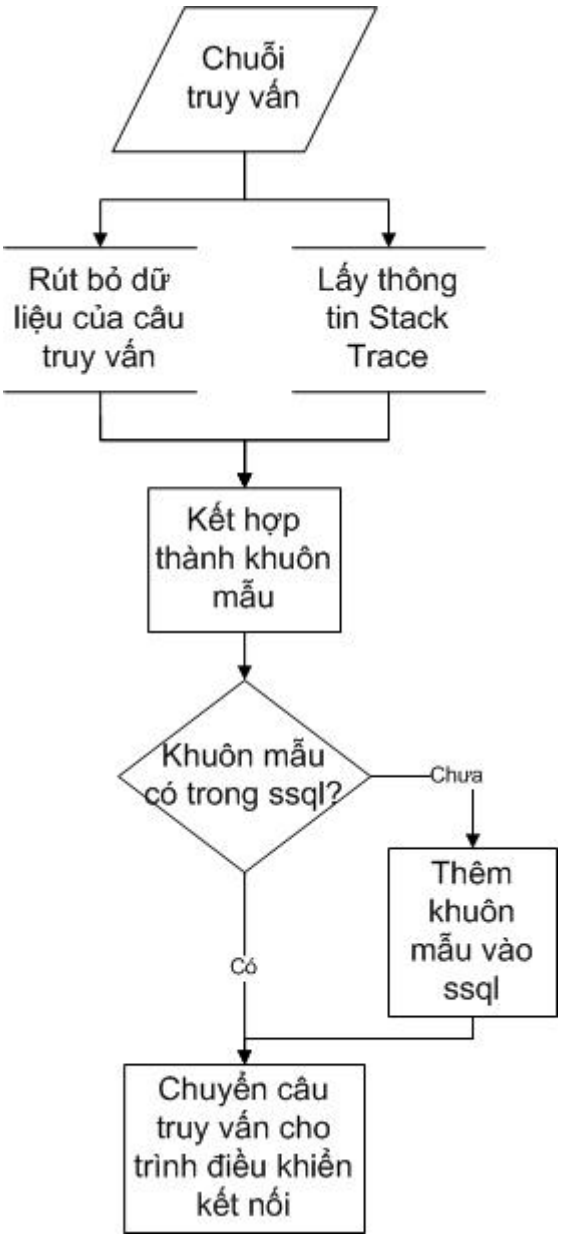
Trong chế độ này, tất cả câu truy vấn sẽ được xác định bằng cách kết hợp các đặc trưng sau:

- Thông tin về stack trace của câu truy vấn.
- Các từ khóa SQL.
- Các bảng và các trường có trong câu truy vấn.

Sau khi kết hợp các đặc trưng trên của câu truy vấn sẽ thu được khuôn mẫu hợp lệ. Tập hợp khuôn mẫu hợp lệ sẽ có dạng sau [1, pp. 6]:

$$S = \{\omega : \omega = (k, a_1, a_2, ...), k \in K, a_i \in (L \cup M \cup N)\}$$

Trong đó, S là tập hợp khuôn mẫu hợp lệ, ω là khuôn mẫu hợp lệ, K là tập hợp stack trace của câu truy vấn, L là tập hợp các từ khóa SQL tương ứng, M là tập hợp các bảng tương ứng và N là tập hợp các trường tương ứng.



Hình 2. 2 Chế độ huấn luyện của SDriver.

Để có thể kết hợp được các đặc trưng trên, khi 1 câu truy vấn được gửi từ ứng dụng web tới SDriver sẽ trải qua quá trình xử lý sau:

Câu truy vấn sẽ được rút bỏ dữ liệu, loại bỏ số và chuỗi đầu vào. Ví dụ như câu truy vấn "`SELECT accounts FROM users WHERE login='user' AND pass='abc123'`" qua quá trình rút bỏ dữ liệu sẽ trở thành "`SELECT accounts FROM users WHERE login= AND pass=`".

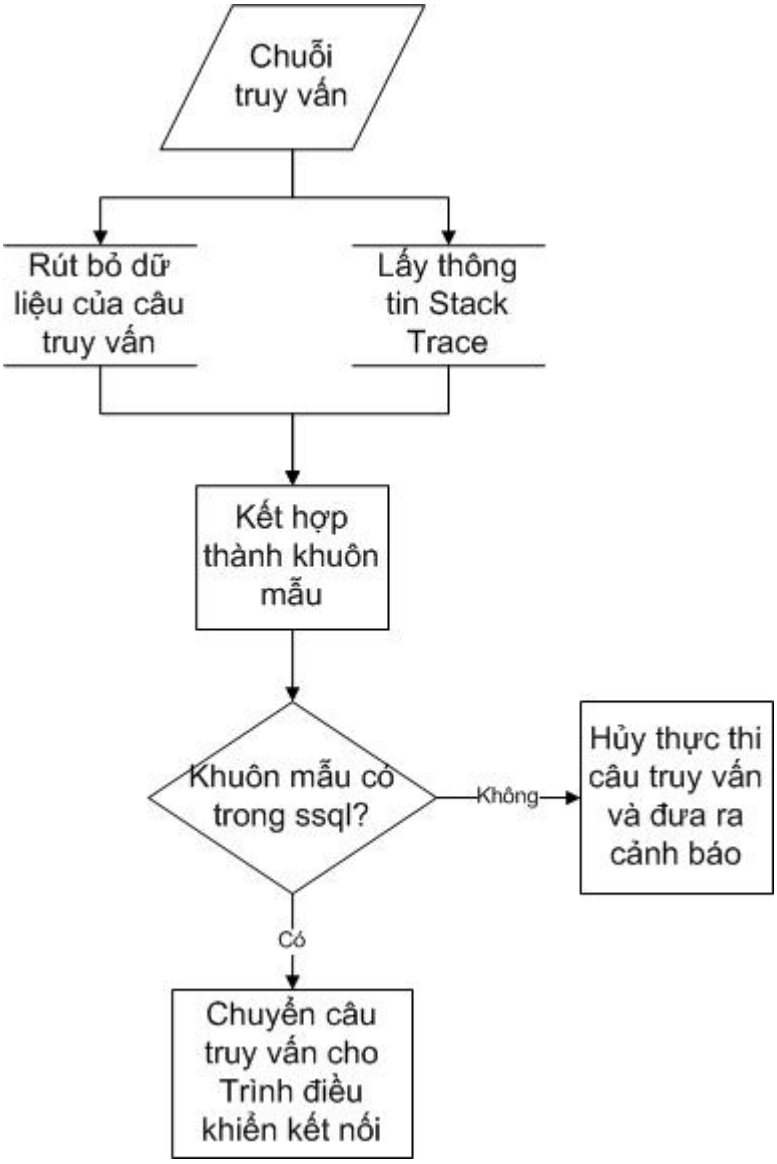
SDriver sẽ thu thập thông tin về stack trace của câu truy vấn bằng cách lần ngược lại ngăn xếp được gọi, cho đến khi tìm được vị trí nơi câu truy vấn được thực thi hay câu lệnh ban đầu.

Sau khi kết hợp thông tin stack trace và câu truy vấn rút bỏ dữ liệu thu được khuôn mẫu hợp lệ, SDriver sẽ lưu trữ khuôn mẫu hợp lệ trong một CSDL được gọi là ssql. Trong chế độ thực thi, SDriver sẽ sử dụng ssql để kiểm tra liệu 1 câu truy vấn là hợp lệ hay không.

2.2.2 Chế độ thực thi.

Trong chế độ thực thi, SDriver sẽ sử dụng các khuôn mẫu hợp lệ đã được lưu trữ trong CSDL ssql trong quá trình huấn luyện để xác định một câu truy vấn là hợp lệ hay không. Các bước thực hiện cũng không có gì quá khác biệt với chế độ huấn luyện.

Hình 2.3 dưới đây thể hiện các bước thực hiện của chế độ thực thi.



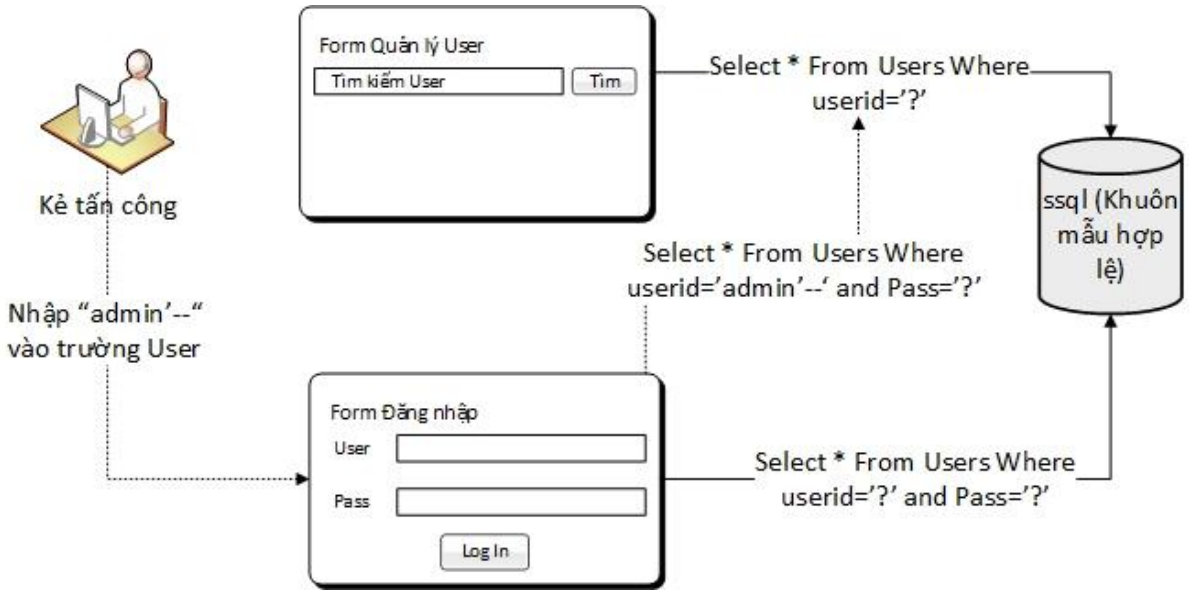
Hình 2. 3 Chế độ thực thi của SDriver.

Ta có thể thấy điểm khác biệt duy nhất giữa hai chế độ là bước kiểm tra câu truy vấn có phù hợp với 1 khuôn mẫu hợp lệ trong CSDL ssql hay không.

Nếu SDriver không tìm thấy khuôn mẫu hợp lệ tương ứng với câu truy vấn, thay vì chèn nó vào trong ssql như chế độ huấn luyện, SDriver sẽ chặn câu truy vấn, đồng thời ghi lại thông báo lỗi hay cảnh báo về tấn công tiêm nhiễm SQL. Nếu tìm thấy khuôn mẫu hợp lệ tương ứng, SDriver sẽ chuyển tiếp câu truy vấn đến trình điều khiển kết nối phía sau để thực thi.

2.3. Stack trace.

Trong kỹ thuật chống tấn công tiêm nhiễm SQL sử dụng các khuôn mẫu hợp lệ theo bối cảnh thì stack trace đóng vai trò là "bối cảnh". Stack trace tạo ra sự khác biệt giữa kỹ thuật này với các kỹ thuật chống tấn công tiêm nhiễm SQL sử dụng khuôn mẫu hợp lệ khác. Stack trace gồm thông tin chi tiết về tất cả phương thức và vị trí gọi (vị trí dòng lệnh), từ phương thức của ứng dụng nơi câu truy vấn được thực thi cho đến phương thức mục tiêu của trình điều khiển kết nối. Mỗi câu truy vấn sẽ có thông tin về stack strace là không giống nhau. Do vậy sử dụng stack trace để cấu thành đặc trưng của câu truy vấn sẽ tạo ra sự khác biệt. Câu truy vấn hợp lệ sẽ trở nên khó giả mạo hơn. Kẻ tấn công dù có thể suy diễn được câu truy vấn hợp lệ nhưng không thể giả mạo được stack trace thì cũng khó có cơ hội vượt qua được SDriver. Sau đây là một ví dụ về vai trò của stack trace.



Hình 2. 4 Ví dụ vai trò của stack trace.

Trong ví dụ này, ta có hai form của ứng dụng web là form Login, là form đăng nhập của người dùng, và form User Manager, là form quản lý người dùng. Trong form Login có câu lệnh thực thi câu truy vấn sau:

```
"Select * From User Where id='?' AND pass='?'"
```

Ký tự "?" trong câu truy vấn sẽ là chuỗi nhập liệu của người dùng vào form Login, có hai trường nhập liệu là trường "id" và trường "pass". Trong form User Manager có câu lệnh thực thi câu truy vấn sau:

```
"Select * From User Where id='?'"
```

Câu truy vấn này sẽ thực thi tìm kiếm user có "id" được nhập vào trong trường iduser của form. Ta thấy điểm khác biệt duy nhất giữa hai câu truy vấn này là điều kiện "`AND pass='?'`".

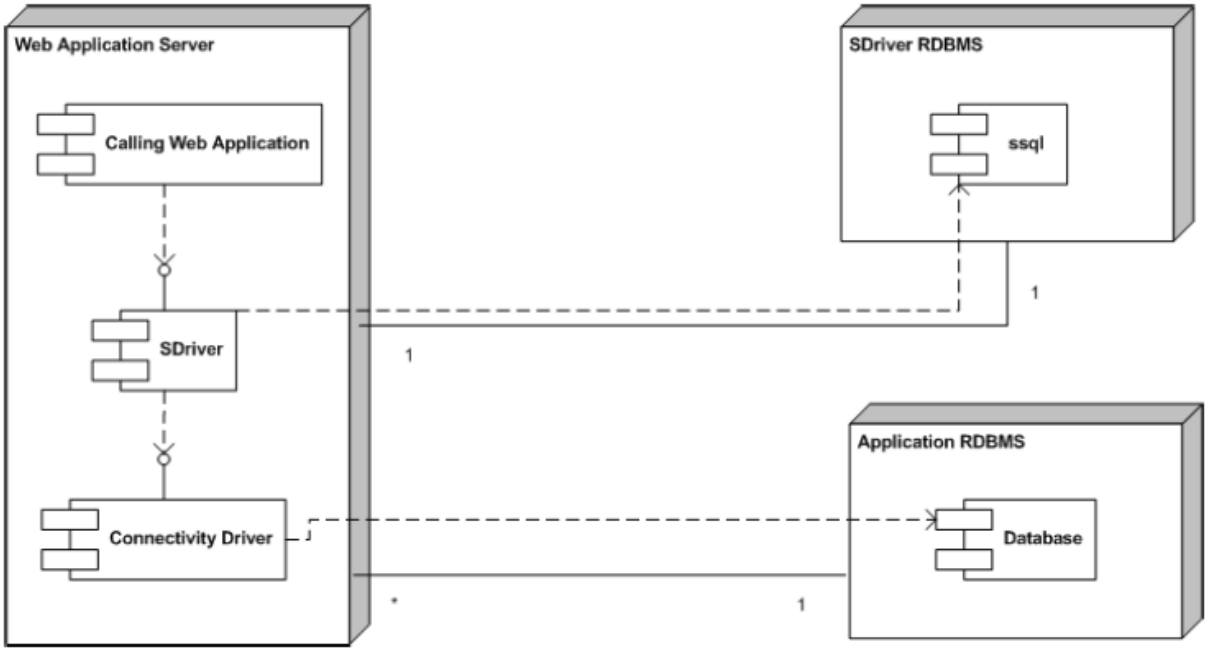
Với kỹ thuật chống tấn công tiêm nhiễm SQL sử dụng khuôn mẫu hợp lệ không có stack trace thì chỉ có bản thân câu truy vấn được lưu trữ trong CSDL SSQL, CSDL lưu trữ khuôn mẫu hợp lệ. Nếu kẻ tấn công nhập vào trường "id" trong form Login là "`id'--`" thì câu lệnh sẽ trở thành "`Select * From User Where id='?'`", như đã trình bày ở chương một các chuỗi đằng sau dấu chú thích "`--`" sẽ bị bỏ qua. Khi đó câu truy vấn trở thành truy vấn hợp lệ, kẻ tấn công có thể đăng nhập thành công mà không cần có mật khẩu.

Trong khi đó với kỹ thuật chống tấn công tiêm nhiễm SQL sử dụng stack trace, khuôn mẫu hợp lệ của từng câu truy vấn sẽ bao hàm cả thông tin stack trace của riêng chúng. Vậy nên dù kẻ tấn công có thể biến đổi câu truy vấn trong form Login giống với câu truy vấn trong form User Manager thì cũng không thể vượt qua được SDriver.

2.4. Mô phỏng hoạt động của SDriver.

2.4.1 Môi trường mô phỏng.

SDriver được triển khải trên nền tảng ngôn ngữ Java, tuy vậy SDriver có thể triển khai trên các môi trường khác. Hình 2.5 dưới đây mô tả kiến trúc thực tế của SDriver khi triển khai [1, pp.8].



Hình 2. 5 Kiến trúc thực tế của SDriver.

Trong môi trường mô phỏng, trình điều khiển kết nối được sử dụng là JDBC. CSDL của ứng dụng web là MySQL. CSDL, ssql, lưu trữ khuôn mẫu hợp lệ của SDriver là MySQL. Ứng dụng web được sử dụng để mô phỏng được triển khai trên nền tảng JSP, Servlet.

SDriver không phụ thuộc vào ứng dụng web hay trình điều khiển kết nối (Connectivity Driver) mà sẽ được chèn vào chúng. Để làm được điều này, mã nguồn của ứng dụng cần phải được thay đổi ở phần kết nối tới trình điều khiển. Thay vì kết nối tới trình điều khiển kết nối, ứng dụng web sẽ kết nối tới SDriver và bản thân SDriver cần phải thiết lập kết nối tới trình điều khiển kết nối ở phía sau, trong mô phỏng là JDBC. Trong môi trường mô phỏng, ứng dụng web sẽ kết nối tới JDBC nên đoạn mã kết nối tới JDBC sẽ là:

```
Class.forName("com.mysql.jdbc.Driver");
String ConnectionURL = "jdbc:mysql://" + hostName + ":3306/" + dbName;
```

Để ứng dụng web kết nối tới SDriver thì đoạn mã trên sẽ trở thành:

```
Class.forName("org.SDriver");
String ConnectionURL = "jdbc:SDriver:org.gjt.mm.mysql.Driver:mysql://" + hostName + ":3306/" + dbName;
```

CSDL ssql được thiết kế có một bảng duy nhất là bảng signatures. Bảng signatures cũng chỉ có một trường duy nhất là trường "ID" có kiểu chuỗi. Các

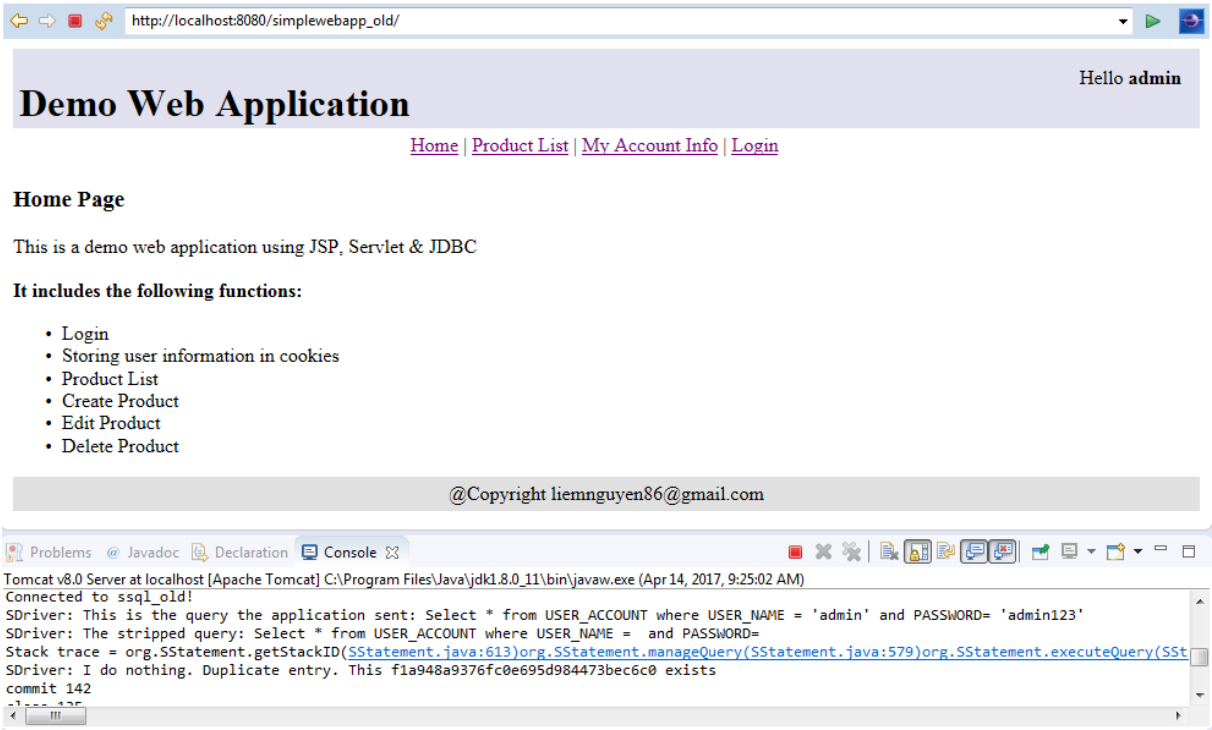
khuôn mẫu hợp lệ được lưu trữ vào ssql sẽ dưới dạng giá trị hàm băm MD5. Hai lý do chính cho việc sử dụng hàm băm cho khuôn mẫu hợp lệ là:

- Thông tin stack strace của câu truy vấn có thể rất dài, việc sử dụng hàm băm sẽ rút gọn được thông tin lưu trữ nhưng vẫn đảm bảo được tính duy nhất.
- Tốc độ SDriver truy vấn vào ssql để tìm kiếm khuôn mẫu hợp lệ sẽ được cải thiện đáng kể so với việc lưu trữ thông tin stack trace và câu truy vấn rút bỏ dữ liệu nguyên bản.

Như đã trình bày ở phần trên, SDriver có hai chế độ hoạt động là chế độ huấn luyện và chế độ thực thi. Để thay đổi chế độ hoạt động của SDriver ta thay đổi giá trị trong file mode.txt. Nếu dòng đầu tiên của file mode.txt là "training mode" thì SDriver sẽ hoạt động trong chế độ huấn luyện. Nếu là "production mode" thì SDriver sẽ hoạt động trong chế độ thực thi. Tùy vào môi trường chạy mà file mode.txt có thể đặt ở các nơi khác nhau, trong môi trường mô phỏng mode.txt có thể được đặt trong thư mục cài đặt eclipse.

2.4.2 Chạy mô phỏng hoạt động của SDriver.

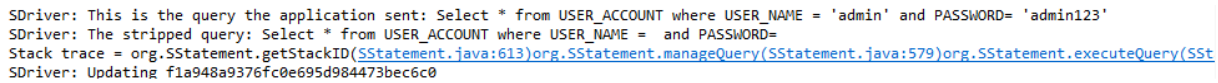
Hình 2.6 dưới đây là giao diện ứng dụng web demo cho mô phỏng.



Hình 2. 6 Giao diện ứng dụng web demo

Ứng dụng web demo sẽ có các chứng năng cơ bản như login, liệt kê danh sách sản phẩm, thêm, sửa, xóa sản phẩm, mã nguồn của trang được lấy từ trang *o7planning.org*.

Trước hết, SDriver cần được trải qua quá trình huấn luyện để xây dựng được CSDL các khuôn mẫu hợp lệ. Trong file mode.txt ta để giá trị dòng đầu tiên là "training mode". Sau đó, ta lần lượt chạy các câu truy vấn của ứng dụng web bằng cách kích hoạt lần lượt các chức năng của nó.



Hình 2. 7 Một kết quả khi SDriver hoạt động ở chế độ huấn luyện.

Ở hình 2.7 trên là kết quả khi kích hoạt chức năng login của ứng dụng web demo. Câu truy vấn được gửi từ ứng dụng web là

"Select * from USER_ACCOUNT where USER_NAME = 'admin' and PASSWORD= 'admin123'"

Sau khi được rút bỏ dữ liệu (stripped query) thì câu truy vấn trở thành

"Select * from USER_ACCOUNT where USER_NAME = and PASSWORD="

Thông tin stack trace của câu truy vấn là:

"org.SStatement.getStackID(SStatement.java:613)org.SStatement.manageQuery(SStatement.java:579)org.SStatement.executeQuery(SStatement.java:543)simplewebapp.utils.DBUtils.findUser(DBUtils.java:37)simplewebapp.servlet.DoLoginServlet.doGet(DoLoginServlet.java:52)simplewebapp.servlet.DoLoginServlet.doPost(DoLoginServlet.java:104)javax.servlet.http.HttpServlet.service(HttpServlet.java:648)javax.servlet.http.HttpServlet.service(HttpServlet.java:729)org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:292)org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:207)org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:52)org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:240)org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:207)simplewebapp.filter.JDBCFilter.doFilter(JDBCFilter.java:96)org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:240)org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:207)org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:212)org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:106)org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:502)org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:141)org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:79)org.apache.catalina.valves.AbstractAccessLogValve.invoke(AbstractAccessLogValve.java:616)org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:88)org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:522)org.apache.coyote.http11.AbstractHttp11Processor.process(AbstractHttp11Processor.java:1095)org.apache.coyote.AbstractProtocol$AbstractConnectionHandler.process(AbstractProtocol.java:672)org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.doRun(NioEndpoint.java:1500)org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.run(NioEndpoint.java:1456)java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:61)java.lang.Thread.run(Thread.java:745)".

Sau khi kết hợp thông tin stack trace và câu truy vấn rút bỏ dữ liệu, giá trị MD5 của khuôn mẫu hợp lệ sẽ là "`f1a948a9376fc0e695d984473bec6c0`".

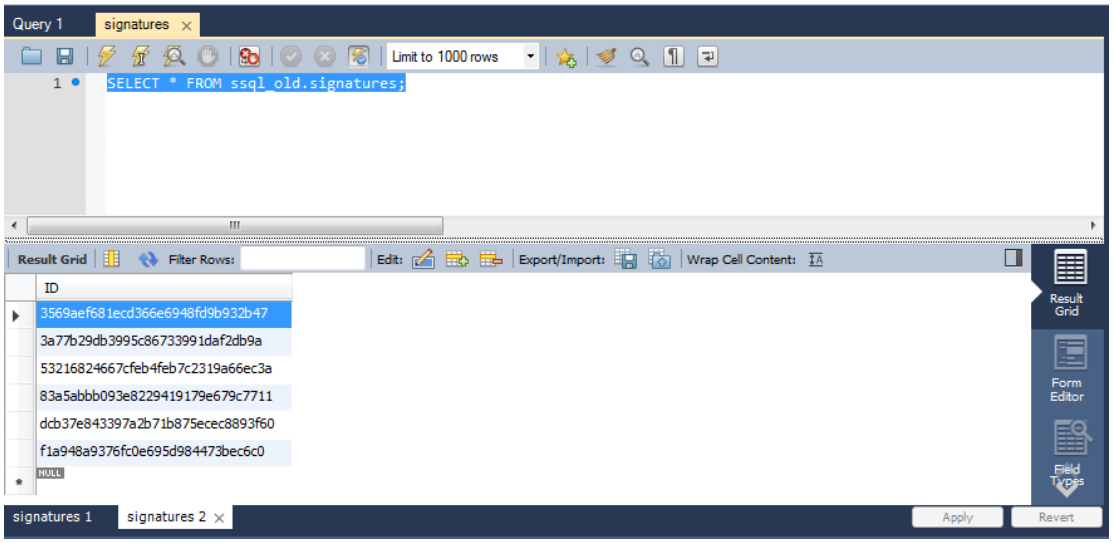
Giá trị MD5 sẽ được chèn vào ssql, SDrive cũng sẽ đưa ra thong báo "`Updating f1a948a9376fc0e695d984473bec6c0`".

Nếu trong trường hợp khâu mẫu hợp lệ đã tồn tại trong ssql thì SDriver sẽ đưa ra thông báo trùng lặp và sẽ không thực hiện thao tác chèn vào ssql.

```
SDriver: This is the query the application sent: Select * from USER_ACCOUNT where USER_NAME = 'admin' and PASSWORD= 'admin123'
SDriver: The stripped query: Select * from USER_ACCOUNT where USER_NAME =  and PASSWORD=
Stack trace = org.SStatement.getStackID(SStatement.java:613)org.SStatement.manageQuery(SStatement.java:579)org.SStatement.execu
SDriver: I do nothing. Duplicate entry. This f1a948a9376fc0e695d984473bec6c0 exists
```

Hình 2. 8 Thông báo trùng lặp của SDriver.

Khi tất cả câu truy vấn đã được cập nhật khuôn mẫu hợp lệ tương ứng trong ssql, bảng signatures của ssql sẽ như hình 2.9 bên dưới.



Hình 2. 9 Bảng signatures của ssql

Sau khi xây dựng được khuôn mẫu hợp lệ cho tất cả câu truy vấn, ta chuyển SDriver sang chế độ thực thi, chuyển dòng đầu tiên của file mode.txt sang "production mode". Trong chế độ thực thi, SDriver sẽ hoạt động như một bộ lọc nhằm phát hiện và ngăn chặn tấn công tiêm nhiễm SQL. Khi nhận một câu truy vấn từ ứng dụng web, SDriver sẽ tìm kiếm khuôn mẫu hợp lệ tương ứng với câu truy vấn trong ssql. Câu truy vấn được xác định là hợp lệ khi tồn tại khuôn mẫu hợp lệ tương ứng với nó, khi đó SDriver sẽ đưa ra thông báo "`NO NEED TO WORRY`", và nó sẽ chuyển câu truy vấn sang trình điều khiển kết nối để thực thi. Trong trường hợp không có khuôn mẫu hợp lệ tương ứng nào, câu truy vấn sẽ được coi là tấn công tiêm nhiễm SQL và bị từ chối thực thi. Đồng thời, SDriver sẽ đưa ra thông báo "`ATTACKING!!! This 219e776510cf52ef4f1d68252298a262 does`

not exists!!! NO RESULTS...YOU ARE ATTACKING!". Dưới đây là một số ví dụ các kỹ thuật tấn công, SDriver sẽ phát hiện và ngăn chặn chúng.

Kỹ thuật tautologies: Chuỗi độc hại "admin' OR 1=1 --" sẽ được chèn vào trường User name trong form Login, trường Password nhập tùy ý. Kết quả như sau:

"SDriver: This is the query the application sent: Select * from USER_ACCOUNT where USER_NAME = 'admin' OR 1=1 --' and PASSWORD= 'gvbcx'

SDriver: The stripped query: Select * from USER_ACCOUNT where USER_NAME = OR 1= gvbcx'

Stack trace = org.SStatement.getStackID(SStatement.java:613)org.SStatement…

SDriver 666: ATTACKING!!! This e3ca49a454c3faadd5818b8aacb85c9 does not exists!!!

SDriver 562: NO RESULTS...YOU ARE ATTACKING!".

Kỹ thuật chú thích cuối dòng (End-of-line comment): Chuỗi độc hại "admin'--" sẽ được chèn vào trường User name trong from Login, trường Password nhập tùy ý. Kết quả như sau:

"SDriver: This is the query the application sent: Select * from USER_ACCOUNT where USER_NAME = 'admin'--' and PASSWORD= 'abc'

SDriver: The stripped query: Select * from USER_ACCOUNT where USER_NAME = abc'

Stack trace = org.SStatement.getStackID(SStatement.java:613)org.SStatement.manageQuery(SStatement.java

ATTACKING!!! This 43f9a4cd8bb1b43fb71578cd568172 does not exists!!!

SDriver 562: NO RESULTS...YOU ARE ATTACKING!".

Kỹ thuật truy vấn Union: Chuỗi độc hại "' union select * from USER_ACCOUNT --" sẽ được chèn vào trường User name trong form Login, trường Password nhập tùy ý. Kết quả như sau:

"SDriver: This is the query the application sent: Select * from USER_ACCOUNT where USER_NAME = '' union select * from USER_ACCOUNT --' and PASSWORD= 'abc'

SDriver: The stripped query: Select * from USER_ACCOUNT where USER_NAME = union select * from USER_ACCOUNT abc'

Stack trace = org.SStatement.getStackID(SStatement.java:613)org.SStatement.manageQuery(SStatement.java

SDriver 666: ATTACKING!!! This c4bd54e8854c3cc448dea46be98f9f1 does not exists!!!

SDriver 562: NO RESULTS...YOU ARE ATTACKING!".

Kỹ thuật truy vấn Piggy-Backed: Chuỗi độc hại "';SHUTDOWN;--" sẽ được chèn vào trường Password, trường User name nhập tùy ý. Kết quả như sau:

"SDriver: This is the query the application sent: Select * from USER_ACCOUNT where USER_NAME = 'admin' and PASSWORD= '';SHUTDOWN;--'

SDriver: The stripped query: Select * from USER_ACCOUNT where USER_NAME = and PASSWORD= ;SHUTDOWN;'

Stack trace = org.SStatement.getStackID(SStatement.java:613)org.SStatement.manageQuery(SStatement.java

SDriver 666: ATTACKING!!! This 6fb255bd8c48bf49491b35cf4e341fc7 does not exists!!!

SDriver 562: NO RESULTS...YOU ARE ATTACKING!".

Với các ví dụ về tấn công tiêm nhiễm SQL trên thì SDriver đều hoạt động tốt, nhưng SDriver có thực sự tốt không nếu hoạt động trong môi trường thực tế. Chương 3 sẽ phân tích và đánh giá chi tiết hoạt động của SDriver.

2.5. Tóm tắt.

Nội dung chương hai đã làm rõ được kỹ thuật chống tấn công tiêm nhiễm SQL sử dụng các khuôn mẫu hợp lệ theo bối cảnh về kiến trúc, cách thức hoạt động của SDriver.

SDriver là trình điều khiển được chèn vào giữa ứng dụng web và trình điều khiển kết nối. Bản thân SDriver không phải là một trình điều khiển kết nối mà nó đóng vai trò như một bộ lọc.

SDriver có hai chế độ hoạt động là chế độ huấn luyện và chế độ thực thi. Trong chế độ huấn luyện, SDriver sẽ xây dựng CSDL về các khuôn mẫu hợp lệ. Trong chế độ thực thi, SDriver sẽ đóng vai trò phát hiện và ngăn chặn tấn công tiêm nhiễm SQL.

SDriver sẽ rút bỏ dữ liệu đầu vào của câu truy vấn và thu thập thông tin về stack strace của câu truy vấn. Kết hợp hai đặc trưng trên sẽ thu được khuôn mẫu hợp lệ tương ứng của câu truy vấn.

Stack trace gồm thông tin chi tiết về tất cả phương thức và vị trí gọi (vị trí dòng lệnh), từ phương thức của ứng dụng nơi câu truy vấn được thực thi cho đến phương thức mục tiêu của trình điều khiển kết nối. Mỗi câu truy vấn sẽ có thông tin về stack trace riêng biệt.

Stack trace khiến việc giả mạo câu truy vấn hợp lệ trở nên vô cùng khó khăn.

CHƯƠNG 3 ĐỀ XUẤT CẢI TIẾN CHỐNG TẤN CÔNG TIÊM NHIỄM SQL SỬ DỤNG CÁC KHUÔN MẪU HỢP LỆ THEO BỐI CẢNH

3.1. Phân tích hoạt động của SDriver.

Như nội dung đã được trình bày ở mục 2.2, SDriver có hai chế độ hoạt động là huấn luyện và thực thi. SDriver xây dựng CSDL các khuôn mẫu hợp lệ trong chế độ huấn luyện và sử dụng CSDL đó để phát hiện và ngăn chặn tấn công tiêm nhiễm SQL trong chế độ thực thi. Hai chế độ hoạt động của SDriver đều sử dụng chung hai bước chính là trích xuất đặc trưng của câu truy vấn và so sánh với khuôn mẫu hợp lệ trong CSDL ssql. Cũng giống như các phương pháp chống tấn công tiêm nhiễm SQL sử dụng khuôn mẫu hợp lệ khác, quá trình trích xuất đặc trưng của câu truy vấn của SDriver rất quan trọng. Trong SDriver thì đặc trưng của câu truy vấn gồm có thông tin về stack trace và câu truy vấn rút bỏ dữ liệu. Thông tin của stack trace sẽ được thu thập bằng cách lần ngược lại các phương thức được gọi cho đến vị trí của mã nguồn thực thi câu truy vấn, xem ví dụ về stack trace ở tiểu mục 2.4.2 bên trên. Do vậy thông tin stack trace là không thể giả mạo được. Trong mục 2.3 bên trên cũng đã trình bày về việc kẻ tấn công không thể sử dụng một câu truy vấn hợp lệ này để giả mạo một câu truy vấn hợp lệ khác để đánh lừa SDriver. Kẻ tấn công muốn vượt qua được SDriver thì phải đánh lừa được SDriver trong quá trình rút bỏ dữ liệu của câu truy vấn, SDriver có thể sẽ loại bỏ nhầm chuỗi độc hại trong khi rút bỏ dữ liệu của câu truy vấn. Quá trình rút bỏ dữ liệu của câu truy vấn được xử lý qua các bước sau:

1. Xóa bỏ các ký tự được cho là thông thường: “+”, “-” và “.”.
2. Xóa bỏ các chữ số đằng sau các phép toán “< , = , >”.
3. Xóa bỏ thành phần chú thích “/* */”.
4. Xóa bỏ chuỗi nhập liệu nằm giữa cặp dấu ngoặc đơn “‘”.

```
/**strippedDownQuery strips the query down by removing strings, comments and numbers using regex */
@SuppressWarnings({ "static-access", "static-access" })
public String strippedDownQuery(String sql) {
    Matcher tmpMatcher = this.trivia.matcher(sql);
    sql = tmpMatcher.replaceAll("");
    tmpMatcher = this.numbers.matcher(sql);
    sql = tmpMatcher.replaceAll("$1");
    tmpMatcher = this.comments.matcher(sql);
    sql = tmpMatcher.replaceAll("");
    tmpMatcher = this.escapeChar.matcher(sql);
    sql = tmpMatcher.replaceAll("");
    System.out.println("SDriver: The stripped query: " +sql);
    return sql;
}
```

Hình 3. 1 Phương thức rút bỏ dữ liệu của câu truy vấn.

```
//Initializing patterns
trivia = Pattern.compile("\\+|-|\\.");
escapeChar = Pattern.compile("\\'(?:.|[\\n\\r])*?\\'");
numbers = Pattern.compile("([<=>,\\(]+\\s*)(?:[0-9A-Fa-f])+");
comments = Pattern.compile("/\\*(?:.|[\\n\\r])*?\\*/");
```

Hình 3. 2 Các mẫu được loại bỏ khỏi câu truy vấn.

Bước một và ba chủ yếu là loại bỏ các ký tự, chuỗi được cho là "thừa", không phải là đặc trưng của câu truy vấn. Trong khi bước hai và bốn là loại bỏ các ký tự số, chuỗi nhập liệu của người dùng. Ví dụ:

Đầu vào là câu truy vấn

"SELECT a.OrderID, a.OrderDate, a.CusID, a.Amount FROM tblOrder a WHERE (datediff(curdate(),a.OrderDate)) < 30 AND a.CusID='abc' /* day la cau truy van select */".

Thì câu truy vấn rút bỏ dữ liệu sẽ là

"SELECT aOrderID, aOrderDate, aCusID, aAmount FROM tblOrder a WHERE (datediff(curdate(),aOrderDate)) AND aCusID=".

Quay lại ví dụ về kỹ thuật tấn công tiêm nhiễm SQL sử dụng chú thích cuối dòng ở mục 2.4 trên, ta có thể thấy dấu chú thích "--" đã được loại bỏ trong câu truy vấn rút bỏ dữ liệu. Thay vì nhập "admin'--", nhập "admin'--'" vào trường Login, câu truy vấn rút bỏ dữ liệu lúc này sẽ là "Select * from USER_ACCOUNT where USER_NAME = and PASSWORD=" và SDriver đưa ra thông báo "NO NEED TO WORRY". Tức là SDriver đã không phát hiện ra đây là một tấn công tiêm nhiễm SQL. Nguyên nhân của vấn đề này là tại bước một trong quá trình rút bỏ dữ liệu của câu truy vấn. Bản thân ký tự "-" là một ký tự thông thường, hay là ký tự dấu âm, nhưng nếu hai ký tự "-" đứng ngay cạnh nhau "--" thì nó sẽ trở thành dấu chú thích cuối dòng, toàn bộ chuỗi đứng sau nó cho đến cuối dòng sẽ được đánh dấu là chú thích và sẽ bị hệ quản trị CSDL bỏ qua khi thực thi câu truy vấn.

Một vấn đề nữa nằm ở bước ba khi loại bỏ thành phần chú thích nằm giữa "/*" và "*/". Bản thân các dấu chú thích này nếu nằm trong cặp ngoặc đơn thì chúng sẽ được giải mã như những ký tự thông thường chứ không phải là dấu chú thích. Vậy nên nếu lồng ghép dấu "/*" và "*/" vào các cặp ngoặc đơn, ví dụ như "'/*' AND '*/'" thì chuỗi "AND" sẽ không được tính như một thành phần chú thích mà là một toán tử của SQL. Do vậy ở bước ba, SDriver có thể sẽ loại bỏ những chuỗi mã độc hại.

Ở bước thứ bốn, SDriver chỉ chú tâm vào việc loại bỏ các chuỗi nằm trong cặp dấu ngoặc đơn mà không quan tâm đến việc có bao nhiêu chuỗi đã bị

loại bỏ. Như trong ví dụ về các kỹ thuật tấn công tiêm nhiễm SQL dưới đây, lẽ ra trong câu truy vấn hợp lệ chỉ có hai cặp ngoặc đơn, tức là có hai chuỗi bị loại bỏ, nhưng SDriver đã loại bỏ tất cả các chuỗi, chỉ cần nó nằm trong 1 cặp ngoặc đơn. Đây cũng là một lỗ hổng của SDriver mà kẻ tấn công có thể lợi dụng.

Sau đây là một số ví dụ về các kỹ thuật tấn công tiêm nhiễm SQL, bằng cách sử dụng các lỗ hổng trên của SDriver để vượt qua SDriver.

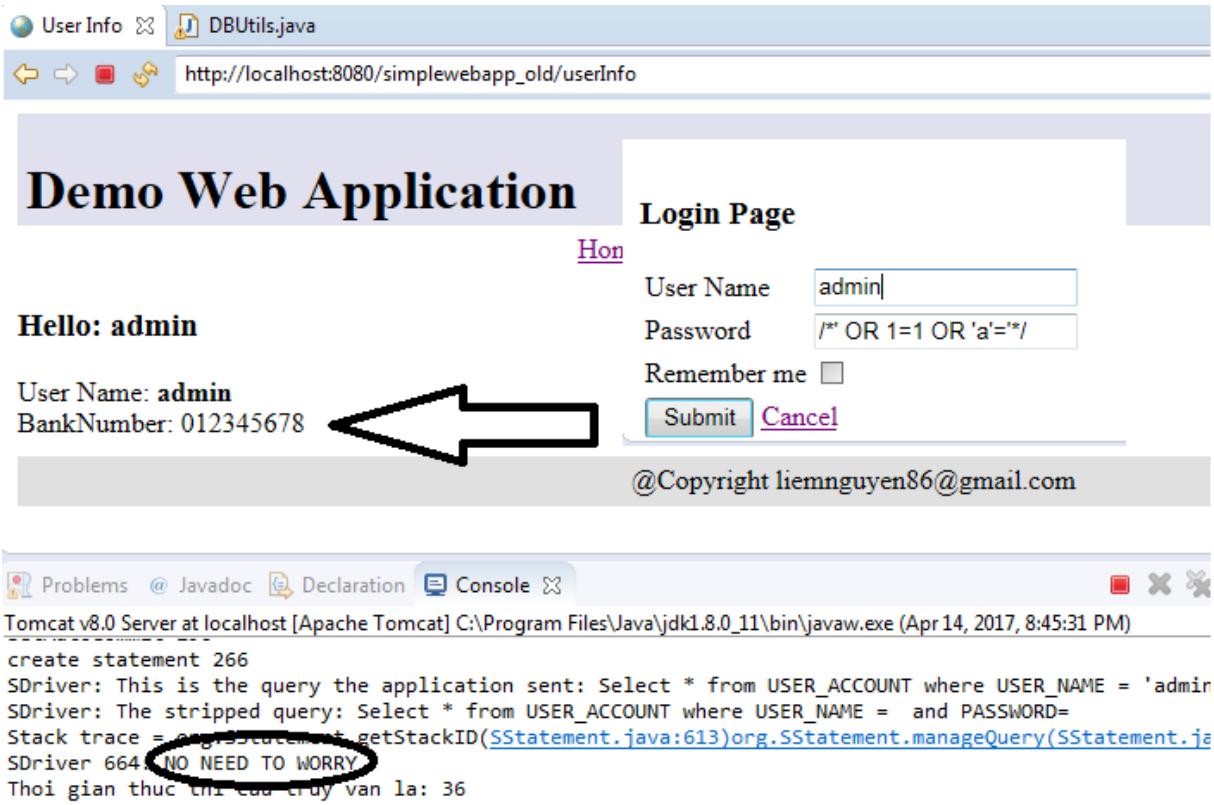
Kỹ thuật tautologies: Nhập "`admin`" vào trường User name trong form Login, trường Password chèn chuỗi độc hại "`/*' OR 1=1 OR 'a'='*/`". Kết quả như sau:

"SDriver: This is the query the application sent: Select * from USER_ACCOUNT where USER_NAME = 'admin' and PASSWORD= '/*' OR 1=1 OR 'a'='*/'

SDriver: The stripped query: Select * from USER_ACCOUNT where USER_NAME = and PASSWORD=

Stack trace = org.SStatement.getStackID(SStatement.java:613)org.SStatement.manageQuery(SStatemen

SDriver 664: NO NEED TO WORRY".



Hình 3. 3 Ví dụ kỹ thuật tấn công tautologies.

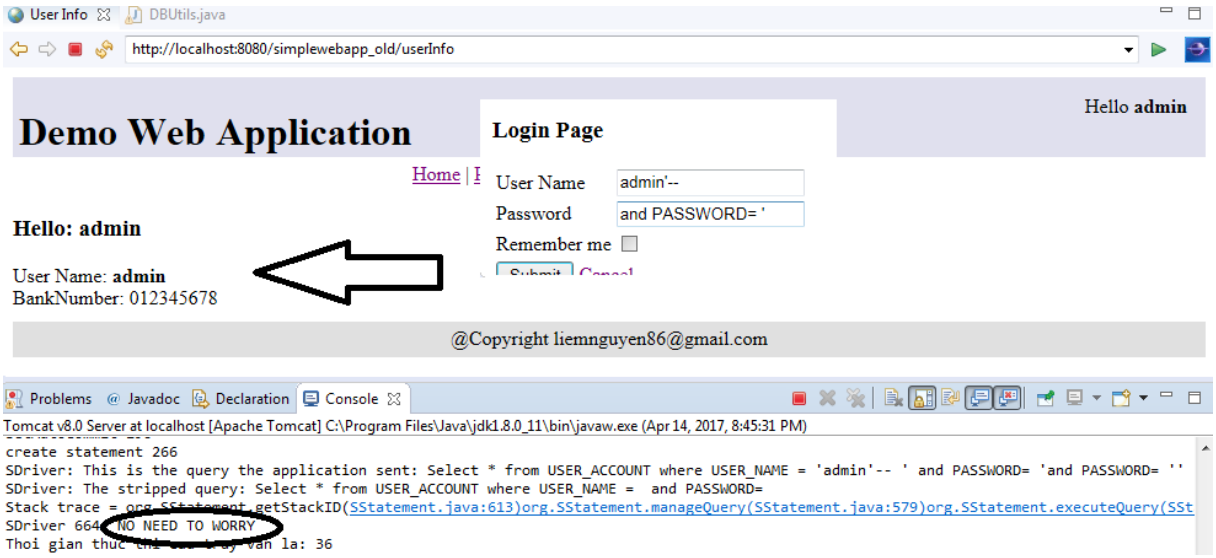
Kẻ tấn công dã giả dạng thành công câu truy vấn hợp lệ và vượt qua được SDriver, với điều kiện luôn đúng "OR 1=1" sẽ luôn có kết quả được trả về khi CSDL thực thi câu truy vấn độc hại trên.

Kỹ thuật chú thích cuối dòng: Chuỗi độc hại "`admin'-- `" sẽ được chèn vào trường User name trong from Login, trường Password nhập "and PASSWORD= '". Kết quả như sau:

"`SDriver: This is the query the application sent: Select * from USER_ACCOUNT where USER_NAME = 'admin'-- ' and PASSWORD= 'and PASSWORD= ''`

`SDriver: The stripped query: Select * from USER_ACCOUNT where USER_NAME = and PASSWORD= Stack trace = org.SStatement.getStackID(SStatement.java:613)org.SStatement.manageQuery(SStatemen`

`SDriver 664: NO NEED TO WORRY`".



Hình 3. 4 Ví dụ kỹ thuật tấn công chú thích cuối dòng.

Kẻ tấn công đã vượt qua được SDriver bằng cách giả dạng chính câu truy vấn hợp lệ. Với câu truy vấn đã được thay đổi, toàn bộ chuỗi đằng sau dấu chú thích "--" sẽ bị bỏ qua khi CSDL thực thi câu truy vấn. Từ đó kết quả sẽ luôn trả về bản ghi có USER_NAME là "admin".

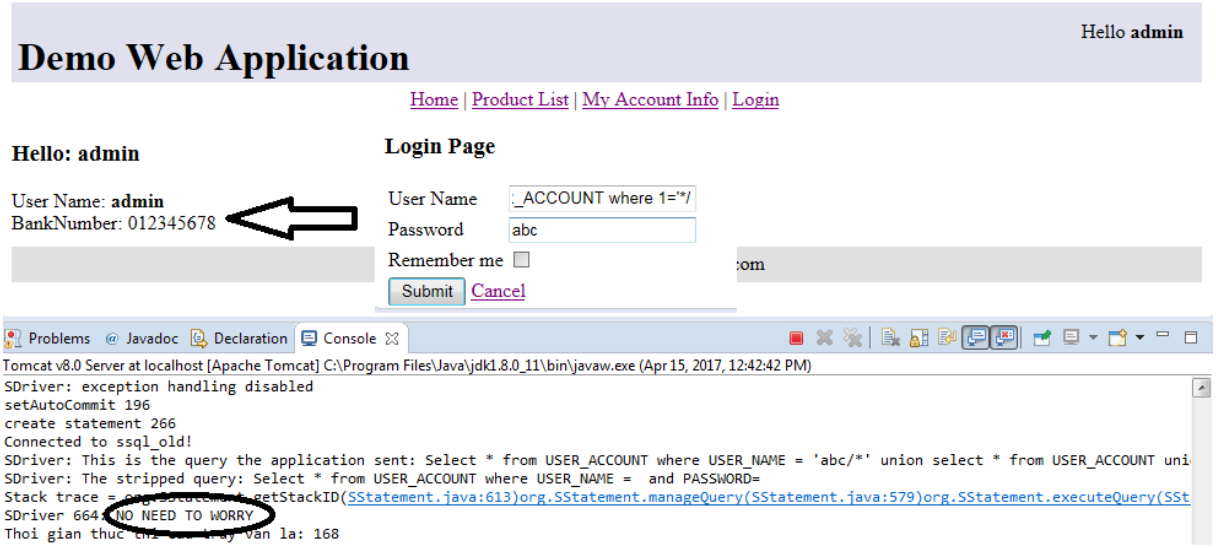
Kỹ thuật truy vấn Union: Chuỗi độc hại "`abc/*' union select * from USER_ACCOUNT union select * from USER_ACCOUNT WHERE 'a'='*/`" sẽ được chèn vào trường User name trong form Login, trường Password nhập tùy ý. Kết quả như sau:

"`SDriver: This is the query the application sent: Select * from USER_ACCOUNT where USER_NAME = 'abc/*' union select * from USER_ACCOUNT union select * from USER_ACCOUNT where 'a'='*/' and PASSWORD= 'abc'`

`SDriver: The stripped query: Select * from USER_ACCOUNT where USER_NAME = and PASSWORD=`

Stack trace = org.SStatement.getStackID(SStatement.java:613)org.SStatement.manageQuery(SStatemen

SDriver 664: NO NEED TO WORRY!".



Hình 3. 5 *Ví dụ kỹ thuật tấn công truy vấn union.*

Kẻ tấn công đã vượt qua được SDriver. Với mã SQL "`union select * from USER_ACCOUNT`", kẻ tấn công có thể thu được thông tin của tất cả user. Tuy nhiên để vượt qua được SDriver thì vẫn cần thêm mã "`select * from USER_ACCOUNT union select * from USER_ACCOUNT where 'a'='*/`" để khớp nối với đoạn mã "`and PASSWORD= 'abc`". Nếu đoạn mã "`and PASSWORD= 'abc`" không được khớp nối sẽ gây ra lỗi khi thực thi câu truy vấn độc hại tại CSDL, lúc này SDriver sẽ tiến hành rollback việc thực thi câu truy vấn, dẫn đến tấn công tiêm nhiễm SQL sẽ thất bại.

Kỹ thuật truy vấn Piggy-Backed: Chuỗi độc hại "`admin/*'; SHUTDOWN;--'*/`" sẽ được chèn vào trường User name, trường Password nhập tùy ý. Kết quả như sau:

"SDriver: This is the query the application sent: Select * from USER_ACCOUNT where USER_NAME = 'admin/*'; SHUTDOWN;-- '*/' and PASSWORD= 'abc'

SDriver: The stripped query: Select * from USER_ACCOUNT where USER_NAME = and PASSWORD=

Stack trace = org.SStatement.getStackID(SStatement.java:613)org.SStatement.manageQuery(SStatemen

SDriver 664: NO NEED TO WORRY!".

***Hình 3. 6** Ví dụ kỹ thuật tấn công truy vấn piggy-backed*

Kẻ tấn công tiếp tục vượt qua được SDriver, tuy vậy lệnh SHUTDOWN sẽ không được thực thi là do phương thức gọi lệnh thực thi truy vấn trong trường hợp trên là excuteQuery, phương thức này chỉ truy vấn để lấy dữ liệu chứ không thực hiện bất kỳ thay đổi nào trên CSDL.

Như vậy SDriver vẫn có thể bị vượt qua bằng cách lợi dụng lỗ hổng khi rút bỏ dữ liệu của câu truy vấn. Để giải quyết vấn đề thì cần một cơ chế rút bỏ dữ liệu của câu truy vấn tốt hơn.

3.2. Đề xuất cải tiến.

Như đã phân tích ở trên, vấn đề của SDriver nằm ở cơ chế rút bỏ dữ liệu của câu truy vấn. Do vậy, một cơ chế rút bỏ dữ liệu của câu truy vấn mới sẽ được đề xuất.

3.2.1 Cơ chế rút bỏ dữ liệu của câu truy vấn mới.

Cơ chế rút bỏ dữ liệu của câu truy vấn được đề xuất không chỉ đảm bảo trích xuất ra được đặc trưng của câu truy vấn mà còn đảm bảo không loại bỏ nhầm những mã độc hại. Sau đây là một số vấn đề cần xem xét khi đề xuất cơ chế rút bỏ dữ liệu của câu truy vấn mới.

Đầu tiên, ta xem xét hai câu truy vấn sau:

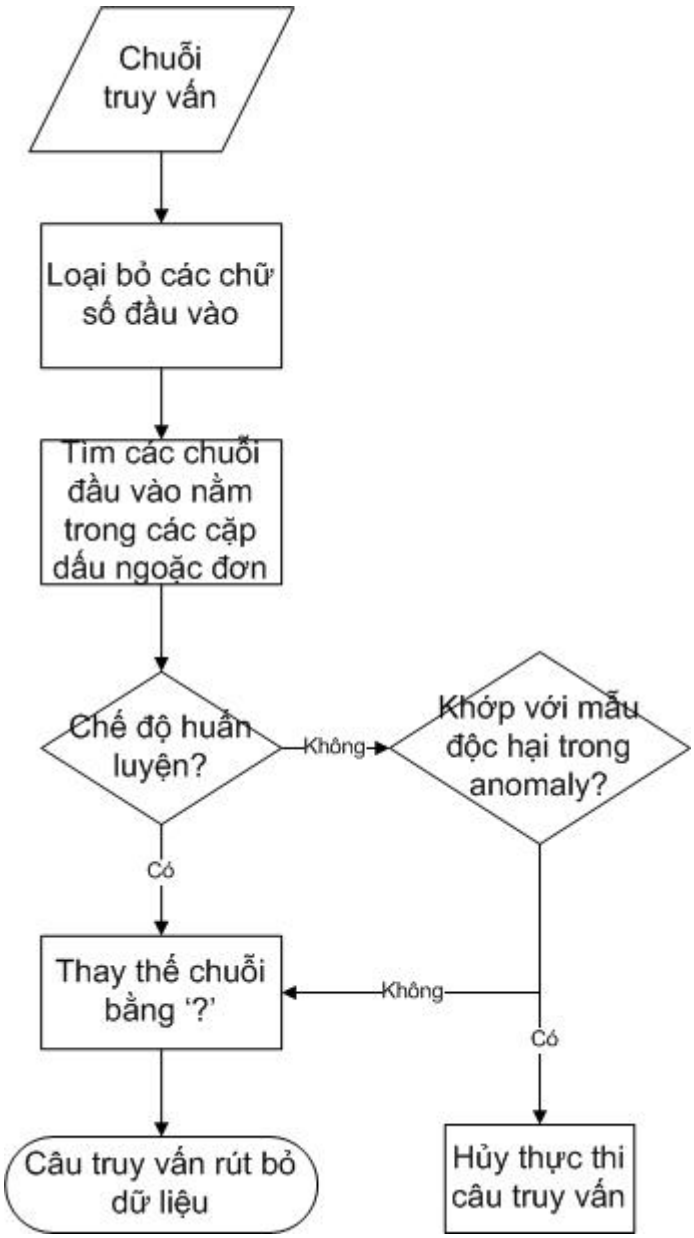
(1) "`Select userid, username from USER_ACCOUNT where USER_NAME = '?' and PASSWORD= '?'`".

(2) "`Select a.userid, a.username from USERS a where a.userid = '?' and a.password = '?' /*dang nhap */`".

Tuy hai câu truy vấn trên có sự khác biệt nhưng khi thực thi chúng sẽ trả về kết quả giống hệt nhau. Sự khác biệt ở đây có chăng là do phong cách lập trình của nhà phát triển. Nhà phát triển có thể sử dụng biệt danh cho bảng, như câu truy vấn (2) biệt danh cho bảng "USERS" là "a", và sử dụng nó để xác định trường nào thuộc bảng nào, như (2) "a.userid" xác định trường "userid" thuộc bảng USERS. Hay nhà phát triển cũng có thể thêm các chú thích, kiểu như "/* dang nhap */", để làm rõ ý nghĩa chức năng câu truy vấn. Tức là các ký tự "+", "-", "." hay các chuỗi được đánh dấu là chú thích cũng có thể coi là đặc trưng của một câu truy vấn, và không nên loại bỏ chúng khỏi câu truy vấn. Ngay cả khi kẻ tấn công phỏng đoán được cấu trúc của câu truy vấn thì cũng khó có thể phỏng đoán được những đặc trưng này.

Tiếp đến ta cần xem xét đến vấn đề loại bỏ chuỗi đầu vào. Khi tiến hành loại bỏ chuỗi đầu vào thì cần xác định chính xác có bao nhiêu chuỗi đầu vào đã được loại bỏ và vị trí của chúng trong câu truy vấn hợp lệ. Để làm được điều đó, thay vì hoàn toàn xóa bỏ chuỗi đầu vào, ta sẽ thay thế chúng bằng một ký tự đặc biệt để "đánh dấu" vị trí của chuỗi bị loại bỏ, ví dụ là ký tự '?'. Nếu có sự khác biệt ký tự "đánh dấu" trên về vị trí và số lượng trong câu truy vấn so với khuôn mẫu hợp lệ thì câu truy đó được coi là độc hại.

Cuối cùng là những chuỗi đã được loại bỏ trong bước rút bỏ dữ liệu của câu truy vấn phải được lọc để đảm bảo không loại bỏ nhầm chuỗi độc hại. Trong SDriver cũ, các chuỗi bị loại bỏ khỏi câu truy vấn trong quá trình rút bỏ dữ liệu đã không được kiểm tra. Chính điều này đã dẫn đến việc loại bỏ nhầm những chuỗi độc hại khỏi câu truy vấn. Do vậy trong cơ chế rút bỏ dữ liệu của câu truy vấn được đề xuất, các chuỗi bị loại bỏ sẽ phải trải qua một lần sàng lọc. Một CSDL lưu trữ các dạng tấn công tiêm nhiễm đã biết sẽ được sử dụng để lọc các chuỗi bị loại bỏ. Nếu chuỗi bị loại bỏ khớp với một mẫu tấn công tiêm nhiễm thì nó có khả năng là một chuỗi độc hại. Lúc này, dù câu truy vấn có khớp với một khuôn mẫu hợp lệ thì cũng sẽ bị ngăn chặn.

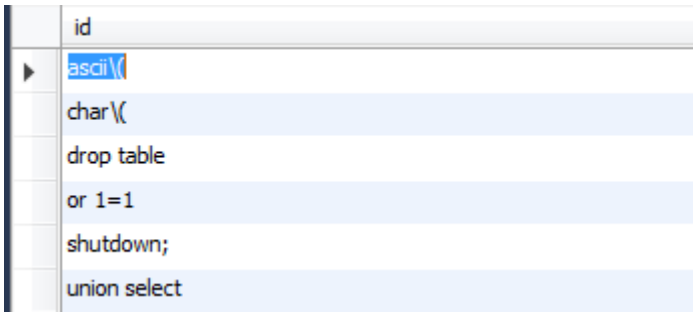


Hình 3. 7 Cơ chế rút bỏ dữ liệu của câu truy vấn được đề xuất.

Các bước của cơ chế rút bỏ dữ liệu của câu truy vấn đề xuất được thể hiện như hình 3.7 trên. Trong SDriver cũ, chế độ huấn luyện và chế độ thực thi sử dụng chung một cơ chế rút bỏ dữ liệu của câu truy vấn. Với cơ chế rút bỏ dữ liệu của câu truy vấn mới, chế độ huấn luyện và chế độ thực thi sẽ có sự khác biệt khi rút bỏ dữ liệu của câu truy vấn. Trong chế độ huấn luyện, với giả định là các chuỗi đầu vào hoàn toàn bình thường thì các chuỗi bị loại bỏ sẽ không phải trải qua bước "khớp với mẫu độc hại" và sẽ đưa thẳng đầu ra là câu truy vấn rút bỏ dữ liệu. Trong chế độ thực thi, các chuỗi bị loại bỏ mới phải trải qua bước "khớp với mẫu độc hại" để đảm bảo SDriver không loại bỏ nhầm bất kì chuỗi độc hại nào.

3.2.2 Triển khai cơ chế rút bỏ dữ liệu được đề xuất.

Một bảng, gọi là bảng anomaly, sẽ được thêm vào trong CSDL ssql. Bảng này bao gồm một trường duy nhất là trường "id". Trường "id" sẽ chứa các mẫu tấn công tiêm nhiễm đã biết. Tùy vào kiểu CSDL của ứng dụng web mà các mẫu này có thể có sự khác biệt.



Hình 3. 8 Một số mẫu tấn công tiêm nhiễm SQL trong bảng anomaly.

Trong quá trình triển khai đã nảy sinh một vấn đề là hiện nay có nhiều kỹ thuật để lẩn tránh việc phát hiện mã độc hại, như cố tình thêm nhiều ký tự khoảng trắng, viết chữ hoa, thường xen kẽ… Do vậy trước khi tiến hành khớp chuỗi bị loại bỏ với các mẫu tấn công tiêm nhiễm thì cần chuẩn hóa chuỗi bằng cách đưa toàn bộ chuỗi về chữ thường, loại bỏ các khoảng trắng đứng cạnh nhau.

Ngoài các mẫu tấn công tiêm nhiễm có sẵn trong bảng anomaly, các mẫu mới có thể được tự động thêm vào trong quá trình chạy ở chế độ thực thi. Khi SDriver phát hiện câu truy vấn không khớp với khuôn mẫu hợp lệ thì đó là câu truy vấn độc hại và bị ngăn chặn, đồng thời nó sẽ được thêm vào bảng anomaly.

3.3. Tóm tắt.

SDriver vẫn có thể để lọt những mã độc. Vấn đề nằm ở cơ chế rút bỏ dữ liệu của câu truy vấn, SDriver đã loại bỏ cả các mã độc.

Các ký tự thông thường hay chuỗi chú thích cũng có thể coi là những đặc trưng riêng của câu truy vấn, bản thân chúng mang phong cách lập trình riêng của nhà phát triển.

Khi loại bỏ các chuỗi đầu vào nằm trong cặp ngoặc đơn thì thay thế bằng một ký tự đặc biệt. Tác dụng của điều này là xác định vị trí và số lượng của chuỗi bị xóa bỏ. Bất kỳ sự khác biệt nào về vị trí, số lượng chuỗi bị xóa bỏ giữa câu truy vấn được gửi từ ứng dụng web với khuôn mẫu hợp lệ đều sẽ được coi là câu truy vấn độc hại, trong chế độ thực thi của SDriver.

Các chuỗi bị loại bỏ phải được lọc để đảm bảo không loại bỏ nhầm chuỗi độc hại.

Một bảng gọi là anomaly chứa các mẫu tấn công tiêm nhiễm sẽ được thêm vào CSDL ssql.

CHƯƠNG 4 KẾT QUẢ THỰC NGHIỆM ĐÁNH GIÁ

4.1. Mô phỏng thực nghiệm SDriver với cơ chế rút bỏ dữ liệu mới.

Sau đây SDriver với cơ chế mới sẽ được gọi tắt là SDriver đề xuất. Môi trường mô phỏng thực nghiệm SDriver đề xuất như sau:

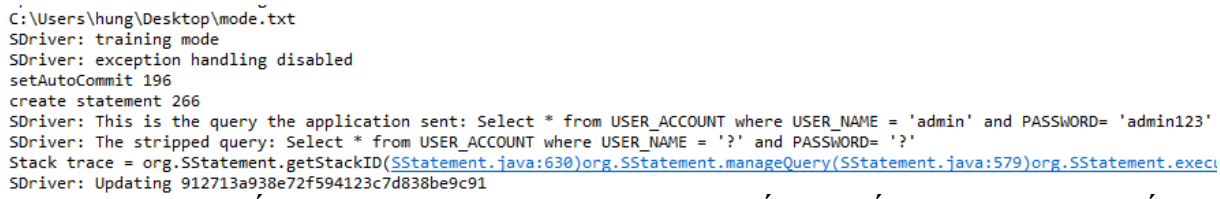
- Ứng dụng web được sử dụng để thực nghiệm được xây dựng dựa trên nền tảng JSP và Servlet.
- Sử dụng eclipse để làm môi trường chạy ứng dụng web.
- Trình điều khiển kết nối là JBDC
- CSDL cho ứng dụng web và SDriver là MySQL. CSDL ứng dụng web là data_JDBC, CSDL cho SDriver là ssql.

Các bước chạy mô phỏng thực nghiệm:

1. Đặt chế độ hoạt động của SDriver là huấn luyện. Chuyển dòng đầu tiên của file mode.txt thành “training mode”.
2. Lần lượt thực thi các câu truy vấn bằng cách chạy các chức năng của ứng dụng web. Tiến hành quan sát các câu truy vấn được rút bỏ dữ liệu.
3. Khi toàn bộ câu truy vấn đã có được khuôn mẫu hợp lệ tương ứng trong CSDL ssql thì dừng chế độ huấn luyện.
4. Chuyển chế độ hoạt động của SDriver sang thực thi. Chuyển dòng đầu tiên của file mode.txt thành “production mode”.
5. Lần lượt thực thi các câu truy vấn bằng đầu vào hợp lệ. Tiến hành quan sát kết quả.
6. Lần lượt thử nghiệm các kỹ thuật tấn công đã vượt qua được SDriver. Quan sát kết quả.

Dưới đây là chi tiết quá trình chạy mô phỏng thực nghiệm hoạt động của SDriver với cơ chế rút bỏ dữ liệu mới.

Ở chế độ huấn luyện: Lần lượt thực thi các câu truy vấn để huấn luyện cho SDriver.



Hình 4. 1 Một kết quả khi SDriver hoạt động ở chế độ huấn luyện với cơ chế rút bỏ dữ liệu mới.

Với cơ chế rút bỏ dữ liệu mới, các chuỗi trong cặp ngoặc đơn đã không bị xóa bỏ hoàn toàn, thay vào đó là “‘?’”. Do vậy kẻ tấn công không thể thoải mái

chèn các cặp dấu ngoặc đơn vào đầu vào được. Trong hình 3.5, câu truy vấn vẫn chưa có được khuôn mẫu hợp lệ tương ứng trong ssql, nên SDriver đã tiến hành chèn thêm khuôn mẫu, dưới dạng giá trị MD5, vào ssql. Trong trường hợp, câu truy vấn đã có khuôn mẫu hợp lệ tương ứng trong ssql, SDriver sẽ đưa ra thông báo trùng lặp và không cần thiết phải làm gì thêm: "`SDriver: I do nothing. Duplicate entry. This 912713a938e72f594123c7d838be9c91 exists`".

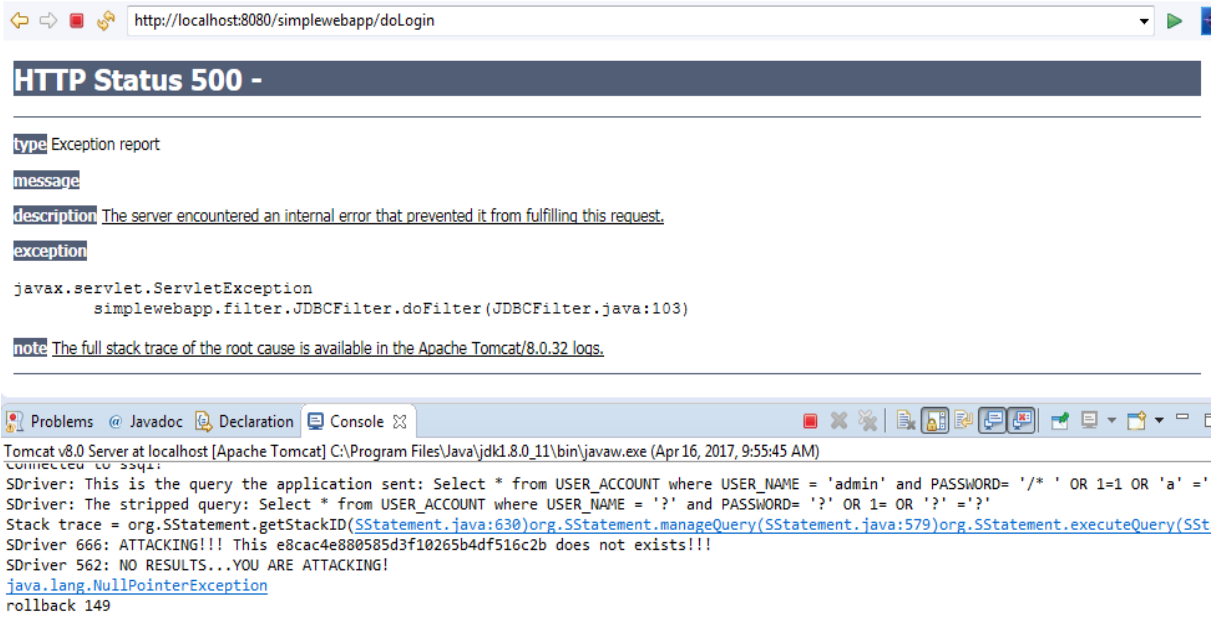
Ở chế độ thực thi: Khi có đầy đủ khuôn mẫu hợp lệ của tất cả câu truy vấn của ứng dụng web, chuyển SDriver sang chế độ thực thi, thay đổi dòng đầu tiên trong file mode.txt thành "production mode". Với đầu vào bình thường (hợp lệ) SDriver sẽ đưa thông báo "`NO NEED TO WORRY!`" và chuyển tiếp câu truy vấn cho trình điều khiển kết nối để thực thi.

```
JDBC Filter
Open Connection for: /doLogin
C:\Users\hung\Desktop\mode.txt
SDriver: production mode
SDriver: exception handling disabled
setAutoCommit 196
create statement 266
SDriver: This is the query the application sent: Select * from USER_ACCOUNT where USER_NAME = 'ADMIN' and PASSWORD= 'Ã dfd'
SDriver: The stripped query: Select * from USER_ACCOUNT where USER_NAME = '?' and PASSWORD= '?'
Stack trace = org.SStatement.getStackID(SStatement.java:630)org.SStatement.manageQuery(SStatement.java:579)org.SStatement.ex
SDriver 664: NO NEED TO WORRY
Thoi gian thuc hien cau truy van la: 3 ms
```

Hình 4. 2 Kết quả khi SDriver không phát hiện truy vấn bất thường.

Thử nghiệm một số kỹ thuật tấn công tiêm nhiễm SQL đã vượt qua được SDriver ở mục 3.1:

Kỹ thuật tấn công Tautologies: Nhập "`admin`" vào trường User name trong form Login, trường Password chèn chuỗi độc hại "`/*' OR 1=1 OR 'a'='*/`". Kết quả như sau:

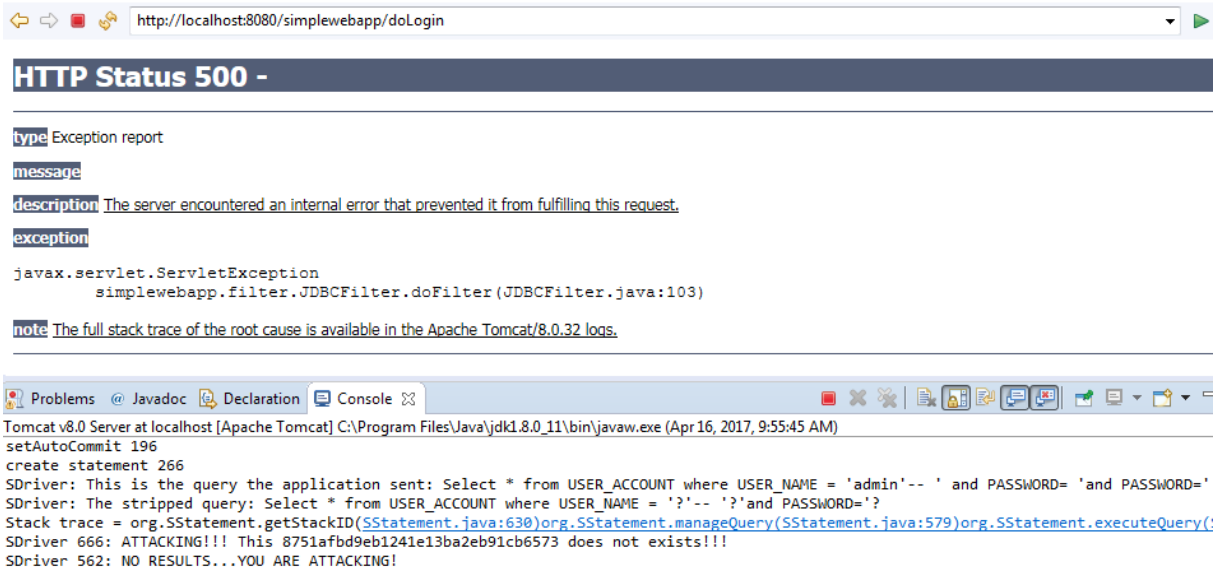


Hình 4. 3 Tấn công Tautologies đã bị phát hiện và ngăn chặn.

Như hình 3.7, SDriver đã phát hiện và ngăn chặn thành công cuộc tấn công tiêm nhiễm SQL. Kẻ tấn công đã chèn chuỗi độc hại vào trường Password, nhưng sau khi rút bỏ dữ liệu thì cả vị trí cũng như số lượng chuỗi bị xóa bỏ khác biệt so với câu truy vấn hợp lệ, đồng thời thừa ra thêm đoạn mã SQL. Chuỗi độc hại được chèn vào đã không bị xóa bỏ như ở phiên bản SDriver trước. SDriver đưa ra cảnh báo "`SDriver 666: ATTACKING!!! This e8cac4e880585d3f10265b4df516c2b does not exists!!!`

`SDriver 562: NO RESULTS...YOU ARE ATTACKING!`".

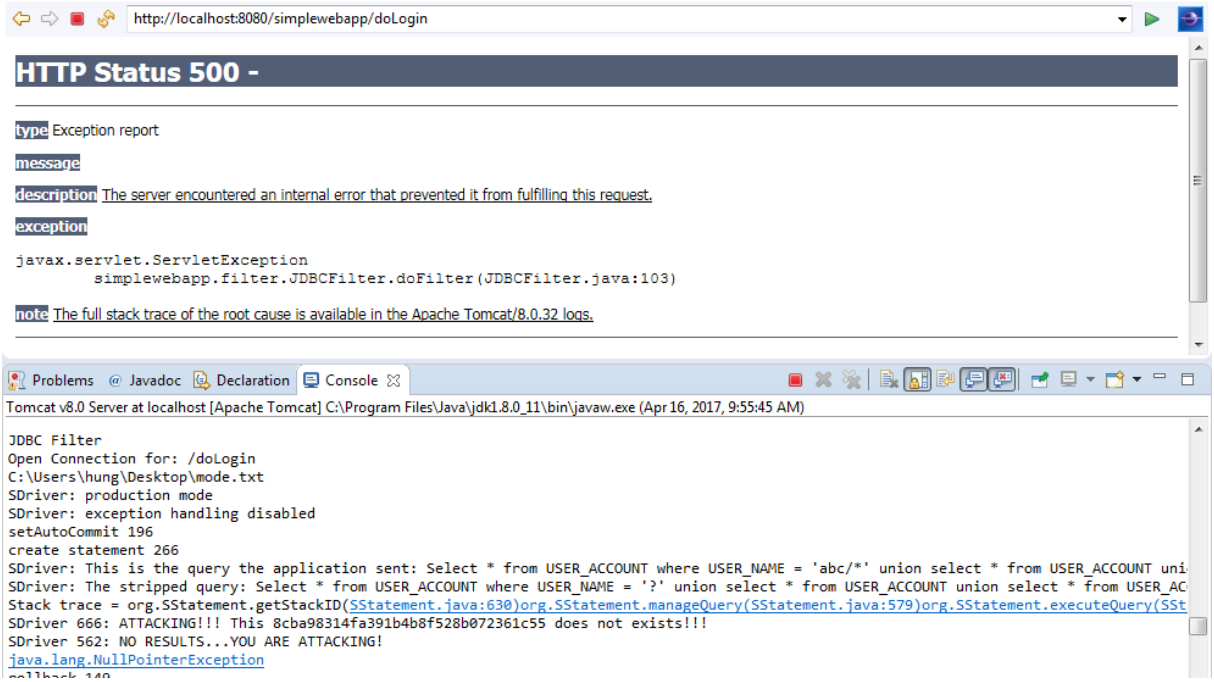
Kỹ thuật tấn công Chú thích cuối dòng: Chuỗi độc hại "`admin'--` " sẽ được chèn vào trường User name trong from Login, trường Password nhập "and PASSWORD= '". Kết quả như sau:



Hình 4. 4 Tấn công chú thích cuối dòng đã bị phát hiện và ngăn chặn.

Tương tự như tấn công Tautologies, kỹ thuật tấn công chú thích cuối dòng cũng bị SDriver phát hiện và ngăn chặn thành công.

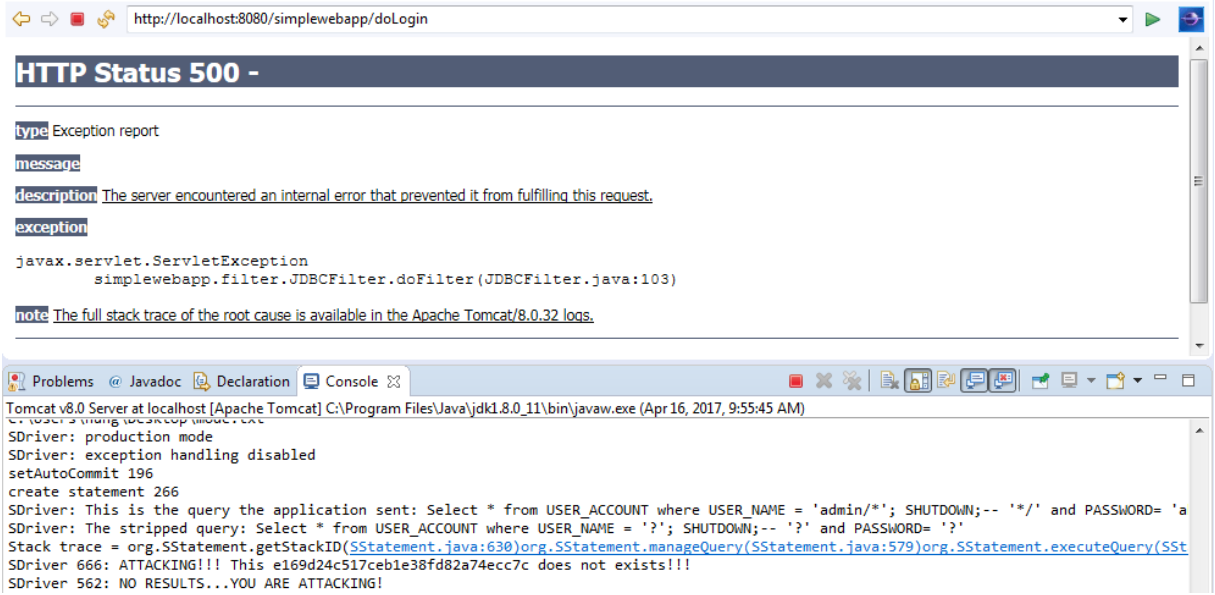
Kỹ thuật tấn công Union: Chuỗi độc hại "`abc/*' union select * from USER_ACCOUNT union select * from USER_ACCOUNT WHERE 'a'='*/`" sẽ được chèn vào trường User name trong form Login, trường Password nhập tùy ý. Kết quả như sau:



Hình 4. 5 Tấn công truy vấn Union đã bị phát hiện và ngăn chặn.

Như hình 3.9, ta có thể thấy toàn bộ chuỗi mã độc được chèn vào đã không bị xóa bỏ như trước, do vậy nó đã tạo nên sự bất thường trong câu truy vấn rút bỏ dữ liệu. Cuộc tấn công đã bị SDriver phát hiện và ngăn chặn.

Kỹ thuật tấn công truy vấn Piggy-Backed: Chuỗi độc hại "`admin/*'; SHUTDOWN;--'*/`" sẽ được chèn vào trường User name, trường Password nhập tùy ý. Kết quả như sau:



Hình 4. 6 Tấn công truy vấn Piggy-Backed đã bị phát hiện và ngăn chặn.

Tương tự như các cuộc tấn công trên, tấn công truy vấn Piggy-Backed từng vượt qua được SDriver đã bị ngăn chặn.

Trên đây là một số ví dụ về các kỹ thuật tấn công tiêm nhiễm SQL. Tuy nhiên để đánh giá được chính xác hoạt động của SDriver với cơ chế rút bỏ dữ liệu mới thì cần đánh giá trên hai khía cạnh là chi phí hoạt động và độ chính xác.

4.2. Đánh giá hoạt động của SDriver đề xuất.

Hoạt động của SDriver cũ được đánh giá theo hai khía cạnh là chi phí hoạt động và độ chính xác. Do vậy để so sánh hoạt động của SDriver đề xuất với SDriver cũ thì cần đánh giá hoạt động của SDriver đề xuất theo hai khía cạnh trên. Ngoài ra phương pháp đánh giá hoạt động của SDriver cũ sẽ được sử dụng để đánh giá hoạt động của SDriver đề xuất.

4.2.1. Đánh giá về chi phí hoạt động.

Chi phí hoạt động của SDriver được đánh giá dựa trên các tiêu chí về chi phí triển khai và hiệu năng của hệ thống.

Về chi phí triển khai, SDriver là trình điều khiển trung gian được chèn vào giữa ứng dụng web và trình điều khiển kết nối. Do vậy, lượng mã nguồn thay đổi rất ít, chỉ thay đổi lại chuỗi kết nối tới CSDL. Thứ nữa là việc huấn luyện cho SDriver cần theo những kịch bản thích hợp để chạy ứng dụng web, đảm bảo toàn bộ câu truy vấn của ứng dụng web có được khuôn mẫu hợp lệ tương ứng. Trong quá trình huấn luyện có thể sử dụng các công cụ kiểm thử tự động để quá trình huấn luyện SDriver trở nên đơn giản hơn. Vậy, SDriver không đòi hỏi phải chỉnh sửa lại nhiều mã nguồn của ứng dụng web, tuy nhiên cần có thời gian để huấn luyện cho SDriver. Vì trong chế độ huấn luyện, SDriver chạy với giả định là tất cả câu truy vấn là bình thường nên chế độ huấn luyện cần chạy trong môi trường không trực tuyến (offline) để tránh khả năng bị lọt những câu truy vấn độc hại khi chạy trực tuyến (online).

Về tiêu chí hiệu năng của hệ thống, hiệu năng của SDriver cũ và mới sẽ cùng được lần lượt kiểm thử trên cùng một hệ thống và sau đó sẽ được so sánh với nhau. Hệ thống được sử dụng để thử nghiệm là:

- Cấu hình máy tính: Bộ vi xử lý Intel Core i3 2.67Ghz, Ram 3GB.
- Môi trường: Java SE Develop Kit 8, Windows 7 32bit.
- CSDL: MySQL version 5.7.

Bảng 4. 1 Thời gian thực thi truy vấn của 2 phiên bản SDriver.

Chế độ	SDriver cũ (ms)	SDriver đề xuất (ms)	Tỷ lệ mới/cũ (%)
Huấn luyện	15.9375	15.5625	97.64706
Thực thi	3.8125	4.3125	113.1148

Bảng 4.1 trên thể hiện thời gian thực thi câu truy vấn, đơn vị mili giây (ms), của SDriver cũ và SDriver đề xuất ở cả hai chế độ huấn luyện và thực thi. Cột "Tỷ lệ mới/cũ" thể hiện so sánh tỷ lệ thời gian thực thi câu truy vấn của SDriver đề xuất với SDriver cũ.

Trong chế độ huấn luyện, thời gian thực thi câu truy vấn của SDriver đề xuất chỉ bằng 97,65% so với SDriver cũ. Điều này là do cơ chế rút bỏ dữ liệu mới đã lược bỏ bớt các thành phần xóa bỏ khỏi câu truy vấn và chỉ tập trung vào xóa bỏ chuỗi nhập liệu đầu vào.

Trong khi ở chế độ thực thi, thời gian thực thi câu truy vấn của SDriver đề xuất lại nhỉnh hơn, bằng 113,11 % so với SDriver cũ. Điều này không nằm ngoài dự liệu vì trong SDriver đề xuất, các chuỗi nhập liệu không chỉ bị loại bỏ khi rút bỏ dữ liệu mà còn phải trải qua một lần sàng lọc.

Ngoài ra ta có thể thấy với SDriver đề xuất, thời gian thực thi câu truy vấn ở chế độ huấn luyện lại cao hơn nhiều so với chế độ thực thi. Dù trong chế độ huấn luyện các chuỗi bị loại bỏ không phải trải qua bước khớp với mẫu tấn công tiêm nhiễm. Nguyên nhân là do trong chế độ huấn luyện các câu truy vấn sau khi được rút bỏ dữ liệu để tạo ra khuôn mẫu hợp lệ sẽ phải chèn thêm vào bảng "signatures" trong CSDL ssql.

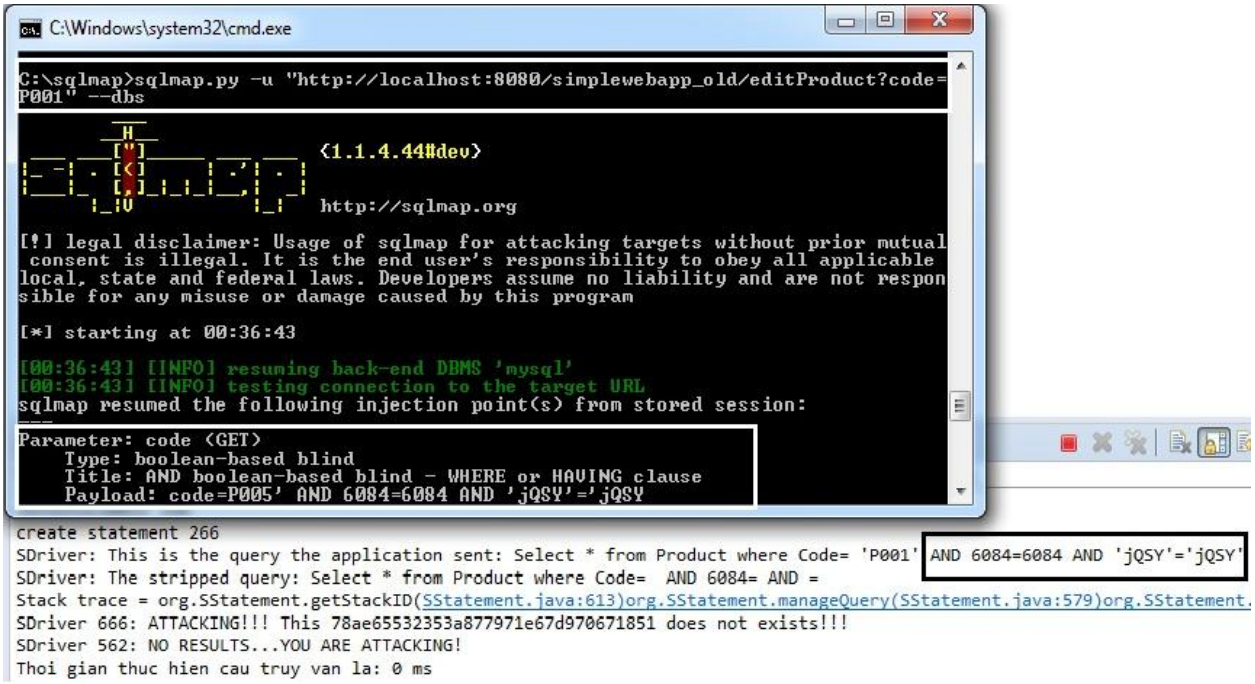
4.2.2. Đánh giá về độ chính xác.

Để đánh giá được độ chính xác của SDriver với cơ chế rút bỏ dữ liệu mới thì có hai cách thức thực hiện là: sử dụng bộ công cụ kiểm thử và đánh giá dựa trên các ứng dụng web thực tế. Kết quả đánh giá độ chính xác của SDriver đề xuất cũng sẽ được so sánh với SDriver cũ. Do vậy trong quá trình đánh giá sẽ lần lượt chạy thực nghiệm SDriver cũ và SDriver đề xuất, sau đó sẽ so sánh kết quả thu được.

Sử dụng bộ công cụ kiểm thử để đánh giá độ chính xác:

Bộ công cụ kiểm thử được sử dụng để đánh giá độ chính xác SDriver đề xuất là sqlmap, tải tại trang web https://sqlmap.org. Sqlmap là bộ công cụ kiểm thử tấn công tiêm nhiễm SQL cho các ứng dụng web, rất mạnh mẽ và đa năng. Sqlmap hỗ trợ kiểm thử nhiều loại tấn công tiêm nhiễm SQL khác nhau, và hỗ trợ kiểm thử nhiều loại CSDL khác nhau. Tuy nhiên sqlmap không hỗ trợ tự động dò quét toàn bộ ứng dụng web mà phải truyền tham số cho nó. Hai phương thức truyền dữ liệu phổ biến trên ứng dụng web là GET và POST. Đối với mỗi phương thức, ta cần truyền tham số thích hợp cho sqlmap để kiểm thử.

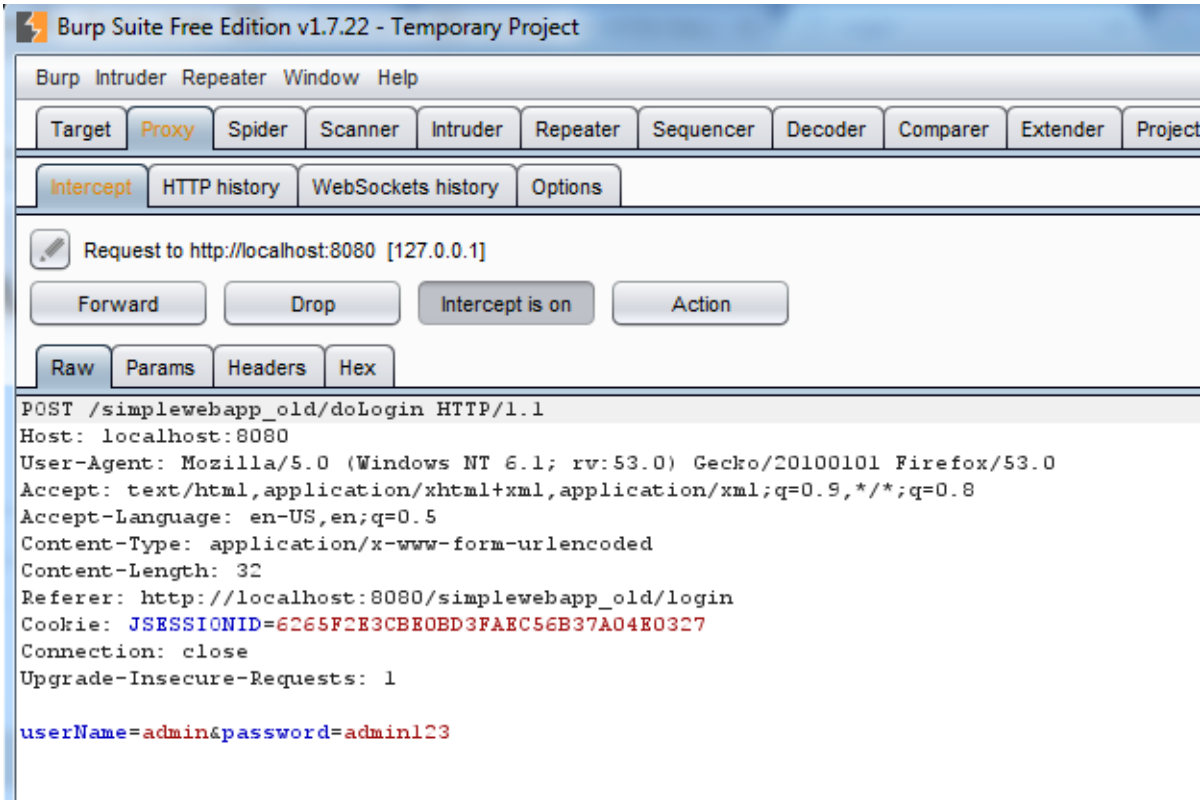
Đối với phương thức GET, truyền tham số trực tiếp trên đường link url của ứng dụng web. Ví dụ kiểm thử tấn công tiêm nhiễm SQL qua tham số "Code", với SDriver cũ, SDriver đề xuất thực hiện tương tự:



Hình 4. 7 Kiểm thử tấn công tiêm nhiễm SQL với tham số code theo phương thức GET.

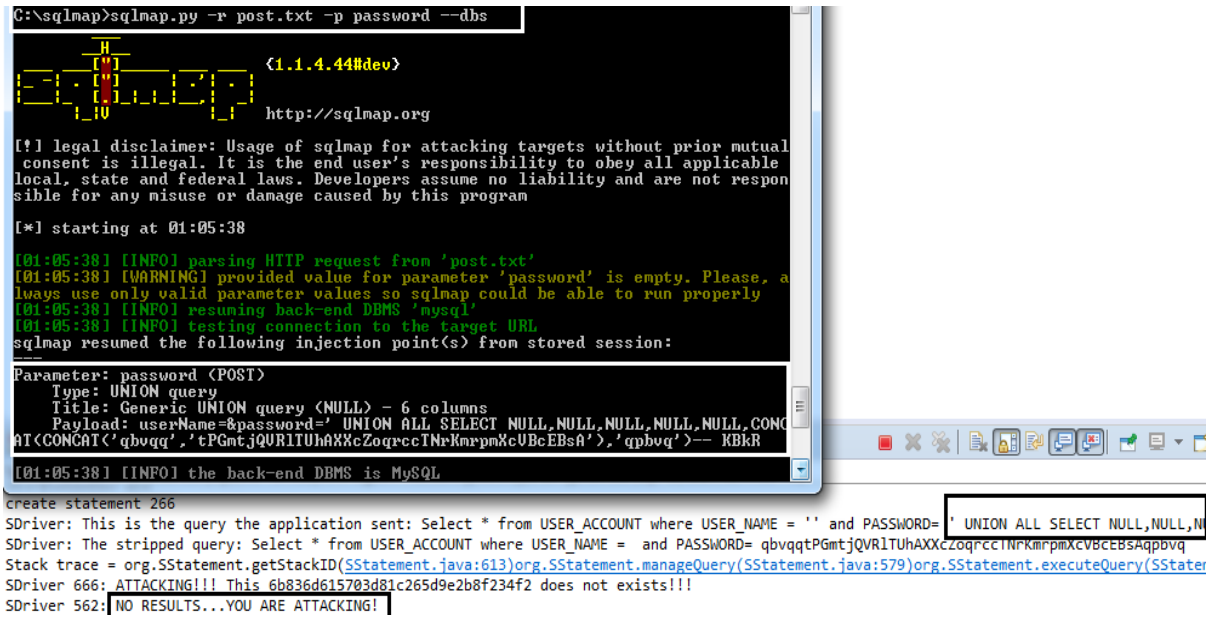
Theo hình trên, đoạn mã url được truyền vào sqlmap để kiểm thử là "http://localhost:8080/simplewebapp_old/editProduct?code=". Ngoài ra tham số "--dbs" cũng được truyền vào sqlmap với ý nghĩa là cố gắng lấy danh sách CSDL trên máy chủ. Sqlmap sẽ tự động tiêm nhiễm thông qua tham số "code". Như hình trên có một chuỗi mã độc được sqlmap cố gắng tiêm nhiễm là "AND 6084=6084 AND 'jQSY'='jQSY", nhưng SDriver đã phát hiện và ngăn chặn.

Đối với phương thức POST thì trước hết cần phải lấy được tham số POST của ứng dụng web. Để làm được điều này cần sử dụng một công cụ là Burp Suite, phiên bản Free Edition. Sau khi lấy được thông tin về phương thức POST, như hình dưới là thông tin POST của trang login, thì có thể lưu thông tin dưới dạng một file text, ví dụ post.txt, để sử dụng.



Hình 4. 8 Ví dụ thông tin phương thức POST của trang login.

Khi đã có thông tin phương thức POST thì tiến hành kiểm thử các tham số, như ví dụ trên là "userName" và "password". Cũng giống như với phương thức GET, khi tiến hành kiểm thử tấn công tiêm nhiễm với tham số "password" với phương thức POST, sqlmap sẽ tiến hành tự động tiêm nhiễm các mã độc hại, như hình dưới là một ví dụ. Mã độc hại "UNION ALL SELECT NULL, NULL, NULL,NULL,NULL,CONCAT(CONCAT ('qbvqq','tPGmtjQVRlTUhAXXcZoqrccTNrKmrpmXcVBcEBsA'),'qpbvq')-- KBkR" đã được tiêm vào tham số "password" nhưng SDriver đã phát hiện và ngăn chặn.



Hình 4. 9 Kiểm thử tấn công tiêm nhiễm SQL với tham số password theo phương thức POST.

Trên là hai ví dụ về cách thức sử dụng công cụ sqlmap để kiểm thử khả năng phát hiện và ngăn chặn tấn công tiêm nhiễm của SDriver cũ và SDriver đề xuất. Tiến hành sử dụng sqlmap để thăm dò lỗ hổng từ các tham số khác của ứng dụng web. Kết quả thu được là cả SDriver cũ và SDriver đề xuất đều phát hiện và ngăn chặn thành công tất cả chuỗi độc hại được sinh từ sqlmap.

Đánh giá qua các ứng dụng web thực tế:

Ứng dụng web thực tế được sử dụng để đánh giá là ba ứng dụng web có mã nguồn mở, được tải ở trang web http://www.codewithc.com. SDriver cũ và SDriver đề xuất sẽ lần lượt được triển khai trên ba ứng dụng web này. Một danh sách các chuỗi mã độc, thuộc nhiều kỹ thuật tấn công tiêm nhiễm khác nhau, sẽ được sử dụng để cố gắng tiêm nhiễm vào các tham số của các ứng dụng web. Bảng 4.2 dưới đây thể hiện kết quả phát hiện và ngăn chặn tấn công tiêm nhiễm SQL của SDriver cũ và SDriver đề xuất. Cột "tấn công tiêm nhiễm SQL" thể hiện số cuộc tấn công đã được thực hiện. Cột "ngăn chặn" thể hiện số cuộc tấn công mà SDriver ngăn chặn thành công, cột "tỷ lệ" là tỷ lệ % ngăn chặn thành công.

Bảng 4. 2 Kết quả ngăn chặn tấn công tiêm nhiễm SQL

Ứng dụng web	Tấn công tiêm nhiễm SQL	SDriver cũ		SDriver đề xuất	
		Ngăn chặn	Tỷ lệ (%)	Ngăn chặn	Tỷ lệ (%)
EMusic	110	101	91,8%	110	100%
Book store	130	124	95,4%	130	100%
Document Manager System	151	145	96%	151	100%

Trong quá trình kiểm thử hoạt động của SDriver trên các ứng dụng web, có nhiều loại tham số khác nhau và không phải loại tham số nào cũng thích hợp để thử nghiệm tấn công tiêm nhiễm. Ví dụ như tham số "search" trong ứng dụng "Document Manager System", có hai câu truy vấn được thực thi là "`select * from documentload where status !='deleted' and author='user1'`" và "`select * from documentshared where status !='deleted' and sharedto='user1'`". Trong cả hai câu truy vấn trên đều không hề có tham số "search", thay vào đó ứng dụng web sẽ truy vấn lấy toàn bộ danh sách "document" có liên quan đến user sau đó so sánh danh sách đó với chuỗi được truyền vào tham số search. Với các tham số kiểu như vậy thì các cuộc tấn công tiêm nhiễm SQL là vô nghĩa vì nó không thực sự tác động đến câu truy vấn. Kết quả là SDriver đã không phát hiện ra và không đưa ra cảnh báo. Do đó các cuộc tấn công tiêm nhiễm thử nghiệm vào kiểu tham số như trên sẽ không được tính vào bảng kết quả.

KẾT LUẬN

Kết quả đạt được:

Sau thời gian tìm hiểu và thực hiện đề tài: *"Chống tấn công tiêm nhiễm SQL sử dụng các khuôn mẫu hợp lệ theo bối cảnh"*. Nội dung bài luận văn đã đạt được các kết quả như sau:

- ✓ Hiểu được tổng quan về tấn công tiêm nhiễm SQL, các cách thức tấn công và các phương pháp ngăn chặn.
- ✓ Hiểu được cơ chế hoạt động của kỹ thuật chống tấn công tiêm nhiễm SQL sử dụng các khuôn mẫu hợp lệ theo bối cảnh - SDriver, áp dụng thực hiện mô phỏng hoạt động của SDriver.
- ✓ Tìm ra được vấn đề còn tồn tại của SDriver
- ✓ Đưa ra được đề xuất cải tiến
- ✓ Chạy mô phỏng SDriver với đề xuất cải tiến và đưa ra được đánh giá.

Nhìn chung, luân văn đã đạt được các mục tiêu nghiên cứu đã đề ra. Tuy nhiên luận văn vẫn cần phải đưa ra được những đánh giá có tính thuyết phục hơn, như mở rộng ứng dụng web, mở rộng số lượng câu truy vấn, thực thi các câu truy vấn có độ phức tạp cao…

Hướng phát triển tiếp theo: Nội dung luận văn có thể phát triển theo các hướng sau:

- ✓ Tiếp tục nghiên cứu cải tiến hiệu năng của kỹ thuật này.
- ✓ Nghiên cứu để triển khai trên nhiều nền tảng khác nhau.

Tài liệu tham khảo

1. Dimitris Mitropoulos and Diomidis Spinellis (2009), "SDriver: Location-Specific Signatures Prevent SQL Injection Attacks", *Computer & Security*, Volume 28, pp. 121-129.
2. Inyong Lee, Sangsoo Yeo, Soonki Jeong, Jongsub Moon (2012), "A novel method for SQL injection attack detection based on removing SQL query attribute values", *Mathematical and Computer Modelling*, Volume 55, pp. 58-68.
3. Open Web Application Security Project (2013), *OWASP Top 10 - 2013 The Ten Most Critical Web Application Security Risks*, United State.
4. William G.J. Halfond, Jeremy Viegas, and Alessandro Orso (2006), "A Classification of SQL Injection Attacks and Countermeasures", *Proceedings of the IEEE International Symposium on Secure Software Engineering*, (1), pp. 13-15.
5. William Stallings, Lawrie Brown (2014), "SQL Injection Attack", *Computer Security: Principles and Practice*, (3), pp. 163-168.