# PHÁT HIỆN TẤN CÔNG SQL INJECTION BẰNG HỌC MÁY

**Trần Đắc Tốt*, Nguyễn Trung Kiên,**
**Trương Hữu Phúc, Lê Ngọc Sơn, Trần Thị Bích Vân**
*Trường Đại học Công nghiệp Thực phẩm TP.HCM*
*Email: *tottd@hufi.edu.vn*
Ngày nhận bài: 10/6/2022; Ngày chấp nhận đăng: 13/7/2022

## TÓM TẮT

Lỗ hổng SQL injection đang được xem là một trong những lỗ hổng bảo mật có độ nguy hiểm thuộc dạng cao nhất và liên tục nằm trong top 10 OWASP. Các cuộc tấn công SQL injection luôn gây ảnh hưởng nghiêm trọng tới các doanh nghiệp hay các trang web cá nhân và chúng vẫn đang tăng dần từng ngày. Do đó nhu cầu áp dụng các kỹ thuật học máy vào việc phát hiện lỗ hổng bảo mật này là một trong những ưu tiên hàng đầu hiện nay. Trong bài báo này chúng tôi sử dụng thuật toán Multinomial Naïve Bayes và K-Nearest neighbors để phân loại dữ liệu được lưu trên file logs dưới dạng văn bản nhằm phát hiện dấu hiệu SQL injection. Kết quả là thuật toán Multinomial Naïve Bayes có độ chính xác cao hơn so với thuật toán K-Nearest Neighbors trong việc phát hiện tấn công SQL injection.

*Từ khóa*: Thuật toán Multinomial Naïve Bayes, thuật toán K-Nearest Neighbors, tấn công SQL injection, tấn công dạng injection, lỗ hổng injection.

## 1. MỞ ĐẦU

Ngày nay, do nhu cầu người dùng càng mở rộng hầu hết các ứng dụng có kết nối với cơ sở dữ liệu ra đời. Đi kèm với nhiều nguy cơ bảo mật tồn tại, dẫn đến nhiều nguy hại cho ứng dụng hay cho người dùng. Dữ liệu của người dùng trên ứng dụng được lưu trữ tại cơ sở dữ liệu và để lấy được dữ liệu đưa lên ứng dụng thì ứng dụng sẽ giao tiếp với cơ sở dữ liệu thông qua một ngôn ngữ gọi là SQL [1]. Việc lợi dụng sơ hở của ứng dụng để thực thi những câu truy vấn SQL ngoài ý muốn của lập trình viên, kỹ thuật này gọi là SQL injection. Theo OWASP [2] và báo cáo của Edgescan [3], dựa vào bảng xếp hạng lỗ hổng bảo mật trong những năm vừa qua, SQL injection đã xuất hiện từ rất lâu và luôn thuộc nhóm đứng đầu những lỗ hổng nghiêm trọng nhất.

Trong thực tế, để xác định một cuộc tấn công SQL injection tự động bằng tường lửa ứng dụng web (WAF) [4, 5] thì rất khó có thể bao quát tất cả, vì SQL injection cũng giống như một truy vấn thông thường, WAF cũng chỉ làm việc dựa trên việc so sánh. Hiện nay, việc áp dụng học máy [6, 7] vào trong việc phát hiện tấn công đang ngày càng phổ biến. Nhiều thuật toán được áp dụng và đưa ra kết quả ngày càng khả quan [8]. Trong bài báo này, nhóm tác giả đề xuất sử dụng thuật toán Multinomial Naïve Bayes và K-Nearest Neighbors (KNN) để phát hiện truy vấn có phải là một SQL injection hay không?

Bài báo này có cấu trúc như sau: Phần 2 sẽ mô tả về SQL injection và trình bày về hai thuật toán được sử dụng cũng như phương pháp thực hiện. Phần 3 chúng tôi sẽ thực nghiệm và đưa ra kết quả.

## 2. MÔ TẢ VÀ GIẢI PHÁP

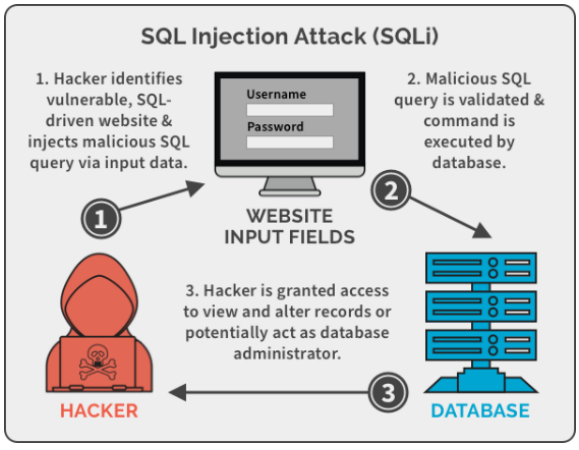
Tấn công SQL injection là việc chèn những truy vấn SQL thông qua dữ liệu đầu vào từ máy khách đến ứng dụng. Một cuộc khai thác SQL injection thành công có thể đọc dữ liệu nhạy cảm từ cơ sở dữ liệu, sửa đổi cơ sở dữ liệu (thêm/xoá/sửa), thực thi các hoạt động trên cơ sở dữ liệu và một số trường hợp đưa ra lệnh thực thi cho hệ điều hành [1].

Faisal Yudo Hernawan và các cộng sự [10] đã đề xuất xây dựng một hệ thống hoạt động như một proxy để ngăn chặn các cuộc tấn công chèn SQL bằng cách sử dụng kết hợp phương pháp SQL Injection Free Secure (SQL-IF) và Naïve Bayes. Kết quả thử nghiệm cho thấy phương pháp lai này có thể cải thiện độ chính xác của phòng chống tấn công SQL injection.

Naghmeh Moradpoor Sheykhkanloo và các đồng sự [7] đã đề xuất một mô hình dựa trên Mạng Nơ-ron (NN) để phát hiện và phân loại các cuộc tấn công SQL injection. Mô hình đề xuất được xây dựng từ ba yếu tố: 1) URL, 2) phân loại URL và 3) mô hình NN. Mô hình đề xuất giúp phát hiện một URL lành tính hoặc độc hại, và xác định kiểu tấn công SQL injection cho mỗi URL độc hại.

Muhammad Amirulluqman Azman và các tác giả [8] đề xuất kỹ thuật học máy để phát hiện tấn công SQL injection. Nghiên cứu dựa trên việc chuẩn bị các bộ dữ liệu để huấn luyện và thử nghiệm. Bộ dữ liệu huấn luyện được sử dụng để thiết lập cơ sở kiến thức và bộ dữ liệu kiểm tra được sử dụng để đánh giá độ chính xác. Kết quả cho thấy rằng kỹ thuật được đề xuất tạo ra độ chính xác cao trong việc nhận dạng các lưu lượng truy cập web độc hại hay lành tính.

Một cuộc tấn công SQL injection thường được thể hiện qua các bước sau (Hình 1).




*Hình 1.* Quy trình để tấn công SQL injection

Đầu tiên, kẻ tấn công dò quét lỗ hổng trên ứng dụng web của nạn nhân, gửi những đoạn mã độc hại tới cơ sở dữ liệu thông qua ứng dụng web. Máy chủ Web (web server) sau khi tiếp nhận câu truy vấn và gửi nó đến máy chủ ứng dụng thì câu truy vấn độc hại tiếp tục được chuyển từ máy chủ ứng dụng đến máy chủ cơ sở dữ liệu (giả định website tồn tại lỗ hổng SQL injection). Tại máy chủ cơ sở dữ liệu, câu truy vấn độc hại sẽ được thực thi. Yêu cầu không hợp lệ hoặc thông tin nhạy cảm của người dùng sẽ được trả về và hiển thị trên ứng dụng web. Cuối cùng, kẻ tấn công nhận được thông tin trả về và bắt đầu thực thi một câu truy vấn độc hại khác.

Trong các phần tiếp theo nhóm tác giả sẽ trình bày thuật toán Multinomial Naïve Bayes và K-Nearest Neighbors và sử dụng chúng để phân loại ra các truy vấn xem chúng có phải là SQL injection hay không.

## 2.1. Thuật toán

### *2.1.1. Multinomial Naïve Bayes*

Phân lớp Bayes là một giải thuật thống kê, có thể dự đoán xác suất của một phần tử dữ liệu thuộc vào một lớp có tỉ lệ bao nhiêu so với xác suất [9]. Phân lớp Bayes được đưa ra dựa trên nền tảng của định lý Bayes (tác giả của định lý là Thomas Bayes), công thức (1). Thuật toán này được ứng dụng rất nhiều trong các lĩnh vực học máy dùng để đưa các dự đoán có độ chính xác cao [10].

Thuật toán còn được thực thi trong trường hợp phát hiện chèn query của SQL. Cốt lõi của Naïve Bayes là nó giả định tính độc lập giữa tất cả các dữ liệu. Nó yêu cầu một số lượng nhỏ dữ liệu để huấn luyện và cực kỳ nhanh chóng so với cách khác.

$$\mathrm{P}(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (1)$$

Trong đó:

- P(A|B) là xác suất xảy ra của một sự kiện ngẫu nhiên A khi biết sự kiện liên quan tới B đã xảy ra.
- P(B|A) là xác suất xảy ra B khi biết A xảy ra
- P(A) là xác suất xảy ra của riêng A mà không quan tâm đến B.
- P(B) là xác suất xảy ra của riêng B mà không quan tâm đến A.

Multinomial Naive Bayes: Mô hình này chủ yếu được sử dụng trong phân loại văn bản mà feature vectors được tính bằng Bags of word.

Giả sử có 2 đoạn văn bản như sau và tìm ra vector tương ứng

Vb1: “Don’t go tonight”

Vb2: “Lover is gone, that is not easy ”

Ta có từ điển từ 2 văn bản như sau:

[“go”, “tonight”, “is”, “gone”]

Với mỗi văn bản sẽ tạo được thành 1 vector với độ dài bằng với độ dài của từ điển

=> Vb1: [1, 1, 0, 0] (có 1 từ “go” và 1 từ “tonight”)

=> Vb2: [0, 0, 2, 1] (có 2 từ “is” và 1 từ “gone”)

Lúc này, mỗi văn bản được biểu diễn bởi một vector có độ dài d chính là số từ trong từ điển. Giá trị của thành phần thứ i trong mỗi vector chính là số lần từ thứ i xuất hiện trong văn bản đó. Khi đó, $P(x_i|c)$ tỉ lệ với tần suất từ thứ i (hay feature thứ i cho trường hợp tổng quát) xuất hiện trong các văn bản của lớp c. Giá trị này có thể được tính bằng cách (2):

$$\lambda_{ci} = \mathrm{P}(x_i|c) = \frac{N_{ci}}{N_c} \quad (2)$$

Trong đó:

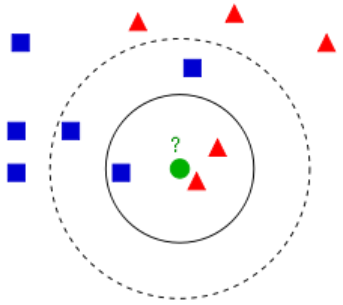
- $N_{ci}$ là tổng số lần từ thứ i xuất hiện trong các văn bản của lớp c, nó được tính là tổng của tất cả các thành phần thứ i của các feature vectors ứng với lớp c
- $N_c$ là tổng số từ (kể cả lặp) xuất hiện trong class c. Nói cách khác, nó bằng tổng độ dài của toàn bộ các văn bản thuộc vào lớp c. Có thể suy ra rằng $N_C = \sum_{i=1}^{d} N_{ci}$, từ đó $\sum_{i=1}^{d} \lambda_{ci} = 1$.

*2.1.2. K-Nearst Neighbors*

KNN (K-Nearst Neighbors) là một trong những thuật toán học có giám sát đơn giản trong học máy [11]. Thuật toán này hoạt động bằng cách tìm ra lớp (nhãn) của một đối tượng thông qua k hàng xóm gần nó nhất.

Đối với thuật toán này, nó sẽ đặt các điểm nằm trong không gian 2 chiều và nó cho rằng là những điểm dữ liệu tương tự nhau sẽ có vị trí gần nhau, việc của chúng ta là tìm k điểm dữ liệu gần với dữ liệu cần kiểm tra.

Thuật toán KNN tìm ra được đầu ra dựa vào thông tin của k điểm dữ liệu training gần nó nhất, nếu k = 1 thì đầu ra của một điểm dữ liệu chính là đầu ra của một điểm dữ liệu đã biết gần nhất, hoặc là có thể tìm ra bằng cách bầu chọn theo số phiếu, hoặc cũng có thể tìm ra bằng cách đánh trọng số khác nhau cho các điểm gần nhất.




*Hình 2.* Phân tích cách chọn k trong KNN

Ví dụ với Hình 2 mẫu thử nghiệm là điểm hình tròn màu xanh lá cây phải được phân loại thành hình vuông màu xanh hoặc hình tam giác màu đỏ, nếu chọn k = 3 (là đường tròn nét liền) thì có thể thấy rõ ràng có 2 điểm đỏ và 1 điểm xanh dương, ta có thể gán cho mẫu thử nghiệm là màu đỏ hình tam giác. Nhưng nếu chọn k = 5 (đường tròn nét đứt) thì mẫu thử nghiệm sẽ là màu xanh hình vuông vì có 3 trong 5 điểm liền kề là hình vuông. Nếu cho k = 4 thì sẽ gồm 2 màu đỏ hình tam giác và 2 màu xanh dương hình vuông.

Vì vậy, việc quan trọng là phải tìm ra k phù hợp với bộ dữ liệu, giá trị của k càng lớn thì càng giảm được việc nhiễu là ảnh hưởng tới việc phân loại nhưng nó sẽ làm cho ranh giới giữa các lớp sẽ ít phân biệt hơn.

Đối với thuật toán này thì có một nhược điểm là nó có xu hướng bị lệch về một lớp có tỷ lệ các điểm dữ liệu xuất hiện nhiều hơn. Có nghĩa là các điểm dữ liệu của cùng một lớp xuất hiện thường xuyên hơn có xu hướng chiếm tỉ lệ xuất hiện cao hơn trong việc dự đoán lớp của một điểm dữ liệu mới.

Phân loại KNN có hai giai đoạn [1]:

- Đầu tiên là xác định những người hàng xóm gần nhất
- Thứ hai là xác định lớp bằng cách sử dụng những người hàng xóm.

## 2.2. Phương pháp thực hiện

Hiện nay có rất nhiều phương pháp để phát hiện các cuộc tấn công SQL injection, tuy nhiên không có phương pháp nào có thể phát hiện hoàn toàn đúng tất cả các loại tấn công SQL injection, mỗi một phương pháp đều có các điểm mạnh cũng như điểm yếu riêng. Trong bài báo này hai thuật toán Multinomial Naive Bayes và K-Nearest Neighbour được thử nghiệm nhằm phát hiện các cuộc tấn công SQL injection. Phương pháp thực hiện như sau:

***Bước 1*:** Thu thập dữ liệu (Data Collection): Đây là giai đoạn thu thập dữ liệu trước khi tiến hành chạy thuật toán để thực hiện huấn luyện từ dữ liệu thu thập được. Các dữ liệu được thu thập từ nhật ký (logs) của các cuộc tấn công SQL injection trong máy chủ. Chúng được chia ra 2 file, một file là tập các dòng logs bị nhiễm SQL injection, còn một file là tập hợp các dòng log và dữ liệu không bị nhiễm.

Ví dụ file thứ nhất:

/index.php?id='+and+order+by+1+--+-+

/login/username='+or+1+=+1&password=123123

Ví dụ file thứ hai tập hợp những dữ liệu không bị nhiễm:

/wordpress/wp-admin/

/wordpress/wp-login.php

***Bước 2:*** Tạo dữ liệu để huấn luyện (Data Creation)

Ở giai đoạn này, dữ liệu tấn công và dữ liệu bình thường được chuyển đổi thành dạng vector số bằng cách tính toán số lượng từ khóa được coi là đã tồn tại query SQL tồn tại trong mỗi dữ liệu tấn công và dữ liệu bình thường. Các từ khóa này được nhóm thành một số loại thường gặp cụ thể là SQL Comments, SQL Operators, SQL Logical, và SQL Keywords. Danh sách các từ khóa phổ biến được sử dụng được hiển thị trong bảng dưới đây (Bảng 1).

*Bảng 1*. Danh sách các từ khóa phổ biến trong các cuộc tấn công SQL injection

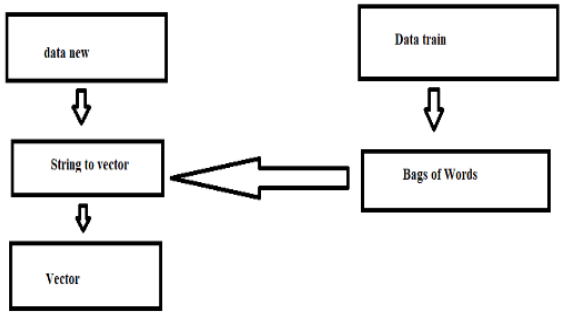
| | |
|---|---|
| Comment | --, #, -+, ++, --, -'', /*, */, /**/ |
| Operator | <, >, ==, != <=, >=, <<, >>, \|\|, && |
| Logical | OR, AND, NOR, XOR |
| Keywords | UNION, SELECT, ORDER,<br>CONCAT, GROUP,<br>INFORMATION_SCHEMA.TABLES, TABLE_NAME, TABLE_SCHEMA,<br>GROUP_CONCAT, COLUMN_NAME,<br>INFORMATION_SCHEMA.COLUMNS, LIMIT, COUNT, CHAR, BY,<br>SLEEP, BENCHMARK,<br>LIKE, WAITFOR, LOAD_FILE, DECLARE, INSERT, UPDATE, FROM,<br>DATABASE, WHERE,<br>EXEC, HEX, DELAY, DESC, FALSE, COUNT, EXPORT_SET, ORD |

Tập dữ liệu này sẽ được sử dụng làm dữ liệu đào tạo để ngăn chặn các cuộc tấn công đưa vào query SQL. Sau khi đã kiểm tra các request từ các phương thức GET (tham số truyền sau url), POST (tham số truyền dưới body) tương ứng khớp với các keyword trong danh mục SQL Comment, SQL Operator, SQL Logical, SQL Operator và SQL Keyword thì chúng ta sẽ đưa vào xử lý dữ liệu. Nếu bị nhiễm SQL injection thì chúng ta sẽ đánh số là 1, còn nếu không bị nhiễm thì đánh số là 0 vào vector số đã đếm số lượng SQL injection tồn tại.

***Bước 3:*** Huấn luyện (Train Data)

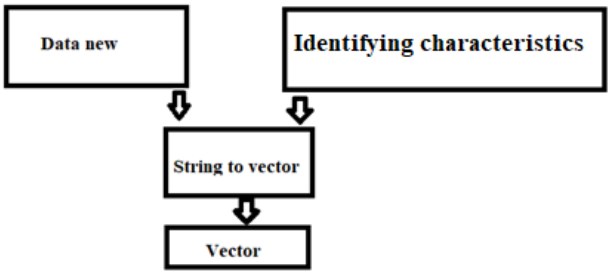
Từ những vector số được chuyển đổi từ bước 2 thì ta có thể đưa vào Multinomial Naïve Bayes để xử lý những dữ liệu đó;

Đối với mô hình Multinomial Naive Bayes sẽ dùng bộ dữ liệu huấn luyện trích xuất các từ trong bộ dữ liệu đó thành một bộ từ điển. Với một chuỗi dữ liệu sẽ được biểu diễn thành một vector có độ dài bằng độ dài của bộ từ điển và giá trị của phần tử thứ i trong vector đó là số lần từ thứ i xuất hiện trong chuỗi dữ liệu đó (Hình 3).




*Hình 3.* Quá trình chuyển data sang dạng vector của Multinomial Naive Bayes

Đối với mô hình K-Nearest Neighbors sẽ dựa vào số lượng đặc điểm do người dùng đưa ra. Trong bài này sẽ sử dụng công thức Euclide để tính khoảng cách. Với một chuỗi dữ liệu được biểu diễn thành một vector có độ dài là tổng số lượng đặc điểm và giá trị của phần tử thứ i trong vector (là giá trị đúng hoặc sai của đặc điểm thứ i. Giá trị đúng tương ứng là 1 và giá trị sai tương ứng là 0) (Hình 4). Trong mô hình này k có giá trị là 7, tức là sẽ lấy k truy vấn SQL có đặc trưng gần với truy vấn đang xét nhất.




*Hình 4.* Quá trình chuyển data sang dạng vector của K-Nearest Neighbors

## 3. KẾT QUẢ VÀ THẢO LUẬN

### 3.1. Bộ dữ liệu

Bộ dữ liệu được sử dụng trong bài này là bộ dữ liệu HttpParamsDataset [12], trong bộ dữ liệu này có các chuỗi dữ liệu là giá trị trong các tham số http request. Bộ dữ liệu này có tổng cộng là 31067 dữ liệu trong đó có 19304 dữ liệu bình thường và 11763 dữ liệu độc hại. Trong 11763 dữ liệu độc hại gồm có các loại tấn công: SQL injection attacks, Cross-Site Scripting, Command injection, Path Traversal attacks. Trong đó SQL injection attacks gồm có: 10852, Cross-Site Scripting có: 532, Command injection có: 89, Path Traversal attacks có: 290. Bộ dữ liệu này được tạo ra từ các nguồn khác như: CSIC2010 dataset [13], sqlmap được dùng để tạo ra các paypload tấn công SQL injection, xssya được dùng để tạo ra các payload Cross-site-Scripting, Vega Scanner tạo ra payload Command injection và Path Traversal và cuối cùng FuzzDB repository chứa các payload Cross-Site scripting, Command injection và Path traversal.

Từ bộ dữ liệu trên sẽ chọn ra hai bộ dữ liệu để sử dụng trong đó có hai bộ dữ liệu để huấn luyện và một dữ liệu để thực hiện test. Trong hai bộ dữ liệu dùng để huấn luyện thì bộ dữ liệu thứ nhất tổng cộng có 19730 trong đó SQL injection là 9865 và normal là 9865. Trong bộ dữ liệu huấn luyện thứ hai tổng cộng có 20105 trong đó có 7235 SQL injection và 12870 normal. Bộ dữ liệu dùng để test có tổng cộng là 10051 trong đó có 6434 dữ liệu normal và 3617 dữ liệu SQL injection.

### 3.2. Thực nghiệm

*3.2.1. Thử nghiệm thuật toán Multinomial Naive Bayes với dữ liệu huấn luyện 19730*

Thử nghiệm thuật toán Multinomial Naive Bayes được huấn luyện với bộ dữ liệu có tổng cộng là 19730 trong đó SQL injection là 9865 và normal là 9865. Thực hiện kiểm tra với bộ dữ liệu test có số lượng là 10051 trong đó có 6434 dữ liệu normal và 3617 dữ liệu SQL injection (Bảng 2 và 3).

*Bảng 2.* Kết quả thử nghiệm thuật toán Multinomial Naive Bayes

| Actual Class | Predicted Class | |
|---|---|---|
| | Positive | Negative |
| Positive | TP = 3586 | FN = 3 |
| Negative | FP = 31 | TN = 6431 |

*Bảng 3.* Độ chính xác của thuật toán Multinomial Naive Bayes

| Performance | Value |
|---|---|
| Accuracy = (TP + TN) / (TP + FP + FN + TN) | 0.996617 |
| Precision = TP / (TP + FP) | 0.991429 |
| Recall = TP / (TP + FN) | 0.999164 |

*3.2.2. Thử nghiệm thuật toán Multinomial Naive Bayes với dữ liệu huấn luyện 20105*

Thử nghiệm thuật toán Multinomial Naive Bayes được huấn luyện với bộ dữ liệu có tổng cộng có 20105 trong đó có 7235 SQL injection và 12870 normal. Thực hiện kiểm tra với bộ dữ liệu test có số lượng là 10051 trong đó có 6434 dữ liệu normal và 3617 dữ liệu SQL injection (Bảng 4 và 5).

*Bảng 4.* Kết quả thử nghiệm thuật toán Multinomual Naive Bayes 2

| Actual Class | Predicted Class | |
|---|---|---|
| | Positive | Negative |
| Positive | TP = 3586 | FN = 1 |
| Negative | FP = 31 | TN = 6433 |

*Bảng 5.* Độ chính xác của thuật toán Multinomial Naive Bayes 2

| Performance | Value |
|---|---|
| Accuracy = (TP + TN) / (TP + FP + FN + TN) | 0.996816 |
| Precision = TP / (TP + FP) | 0.991429 |
| Recall = TP / (TP + FN) | 0.999721 |
| F1 = 2*(Recall * Precision) / (Recall + Precision) | 0.995558 |

*3.2.3. Thử nghiệm thuật toán K-Nearest Neighbors với dữ liệu huấn luyện 19730*

Thử nghiệm thuật toán K-Nearest Neighbors được huấn luyện với bộ dữ liệu có tổng cộng là 19730 trong đó SQL injection là 9865 và normal là 9865. Thực hiện kiểm tra với bộ

dữ liệu test có số lượng là 10051 trong đó có 6434 dữ liệu normal và 3617 dữ liệu SQL injection (Bảng 6 và 7).

*Bảng 6*. Kết quả thử nghiệm thuật toán K-Nearest neighbor

| Actual Class | Predicted Class | |
|---|---|---|
| | Positive | Negative |
| Positive | TP = 3616 | FN = 964 |
| Negative | FP = 1 | TN = 5470 |

*Bảng 7*. Độ chính xác của thuật toán K-Nearest Neighbors

| Performance | Value |
|---|---|
| Accuracy = (TP + TN) / (TP + FP + FN + TN) | 0.903989 |
| Precision = TP / (TP + FP) | 0.999723 |
| Recall = TP / (TP + FN) | 0.789519 |
| F1 = 2*(Recall * Precision) / (Recall + Precision) | 0.882274 |

*3.2.4. Thử nghiệm thuật toán K-Nearest Neighbors với dữ liệu huấn luyện 20105*

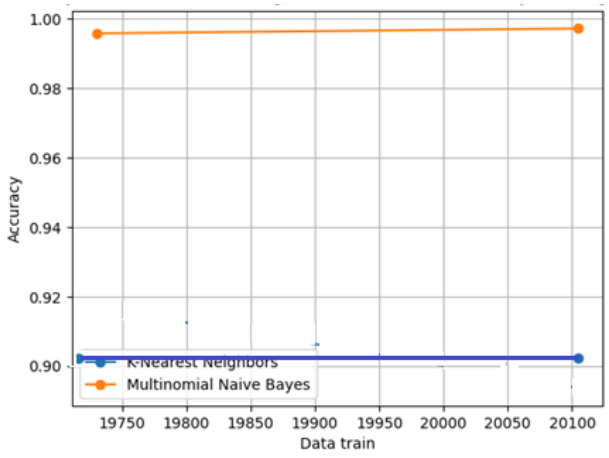
Thử nghiệm thuật toán K-Nearest Neighbors được huấn luyện với bộ dữ liệu có tổng cộng có 20105 trong đó có 7235 SQL injection và 12870 normal. Thực hiện kiểm tra với bộ dữ liệu test có số lượng là 10051 trong đó có 6434 dữ liệu normal và 3617 dữ liệu SQL injection (Bảng 8 và 9).

*Bảng 8*. Kết quả thử nghiệm thuật toán K-Nearest Neighbors 2

| Actual Class | Predicted Class | |
|---|---|---|
| | Positive | Negative |
| Positive | TP = 3616 | FN = 965 |
| Negative | FP = 1 | TN = 5469 |

*Bảng 9*. Độ chính xác của thuật toán K-Nearest Neighbors 2

| Performance | Value |
|---|---|
| Accuracy = (TP + TN) / (TP + FP + FN + TN) | 0.903890 |
| Precision = TP / (TP + FP) | 0.999723 |
| Recall = TP / (TP + FN) | 0.789347 |
| F1 = 2*(Recall * Precision) / (Recall + Precision) | 0.882166 |




*Hình 5.* Độ chính xác của thuật toán K-Nearest Neighbors và Multinomial Naive Bayes

## 4. KẾT LUẬN

Trong bài báo này nhóm tác giả đã nghiên cứu thuật toán Multinomial Naïve Bayes và thuật toán K-Nearest Neighbors cho bài toán phát hiện tấn công SQL injection, so sánh 2 thuật toán này như kết quả trên Hình 5. Các thí nghiệm, thực nghiệm cho thấy rằng hệ thống phát hiện được các cuộc tấn công SQl injection có tỷ lệ chính xác và hiệu quả cao. Các nghiên cứu trong tương lai sẽ tập trung vào cải thiện tốc độ và độ chính xác của 2 thuật toán này cho bài toán phát hiện tấn công SQL injection và thử nghiệm kết hợp với 2 thuật toán trên cho ứng dụng phát hiện SQL injection.

## TÀI LIỆU THAM KHẢO

1. W3schools. n. d.. SQL Introduction, from https://www.w3schools.com/sql/sql_intro.asp, last accessed 2021/06/20.
2. OWASP Top 10 team, Open Web Application Security Project (OWASP), OWASP Top Ten Project, SQL injection – OWASP, Available at: https://owasp.org/www-community/attacks/SQL_Injection.
3. Edgescan, Edgescan 2021 Stats Reports, edgescan cyber Security company (2021), Available at: https://info.edgescan.com/hubfs/Edgescan2021StatsReport.pdf.
4. Luca Demetrio, Andrea Valenza, Gabriele Costa, Giovanni Lagorio - WAF-A-MoLE: Evading web application firewalls through adversarial machine learning, Conference: 35th Annual ACM Symposium on Applied Computing At: Brno, Czech Republic (March 2020), DOI:10.1145/3341105.3373962.
5. Michael Ritter, Cyber Risk Services, Deloitte, Web Application Firewall Profiling and Evasion, Open Web Application Security Project (OWASP).
6. Padraig Cunningham, Sarah Jane Delany - K-Nearest neighbour classifiers ACM Computing Surveys **54** (6) (2007), DOI:10.1145/3459665.
7. Sheykhkanloo N. N. - A Learning-based Neural Network Model for the Detection and Classification of SQL Injection Attacks. International Journal of Cyber Warfare and Terrorism (2017). Available at: https://dl.acm.org/doi/10.4018/IJCWT.2017040102.

8. Azman M. A., Marhusin M. F. & Sulaiman R. - Machine Learning-Based Technique to Detect SQL Injection Attack, Journal of Computer Science **17** (3) (2021) 296-303. https://doi.org/10.3844/jcssp.2021.296.303.
9. Amit Banchhor, Tushar Vaidya - SQL injectiondetection using Baye's classification, International Journal of Advance Research in Science and Engineering **5** (1) (2016) 313-317.
10. Faisal Yudo Hernawan, Indra Hidayatulloh, Ipam Fuaddina Adam - Hybrid method integrating SQL-IF and Naïve Bayes for SQL injection attack avoidance, Journal of Engineering and Applied Technology **1** (2) (2020) 85-96.
11. Alex S. & Vishwanathan, S.V.N. - Introduction to Machine Learning, Published by the press syndicate of the University of Cambridge, Cambridge, United Kingdom (2008).
12. HTTP Param Dataset, https://github.com/Morzeux/HttpParamsDataset, last accessed 2021/06/20.
13. HTTP DATASET CSIC 2010, https://www.isi.csic.es/dataset/, last accessed 2021/06/20.
14. Extended Log File Format, https://www.w3.org/TR/WD-logfile.html, last accessed 2021/06/20.

**ABSTRACT**

MACHING LEARNING TO DETECT SQL INJECTION

Tran Dac Tot*, Nguyen Trung Kien,
Truong Huu Phuc, Le Ngoc Son, Tran Thi Bich Van
*Ho Chi Minh City University of Food Industry*
*Email: *tottd@hufi.edu.vn*

SQL Injection vulnerability is being considered as one of the most dangerous security vulnerabilities continuously in the top 10 OWASP. SQL Injection attacks have always seriously affected businesses or private websites and they are still increasing day by day. Due to the increasing demand, the application of machine learning techniques to detect this security breach is to solve the above problem. In this paper, we have used an algorithm called Multinomial Naïve Bayes, K-Nearest Neighbors on the log file in the form of text. As a result, the Multinomial Naïve Bayes algorithm has higher accuracy than the K-Nearest neighbors algorithm in SQL Injection attack.

*Keywords*: Multinomial Naïve Bayes, K-Nearest Neighbors, SQL Injection attack, Injection attack, Injection vulnerability.