**01**

**01**

**Th.S Trương Tấn Khoa**

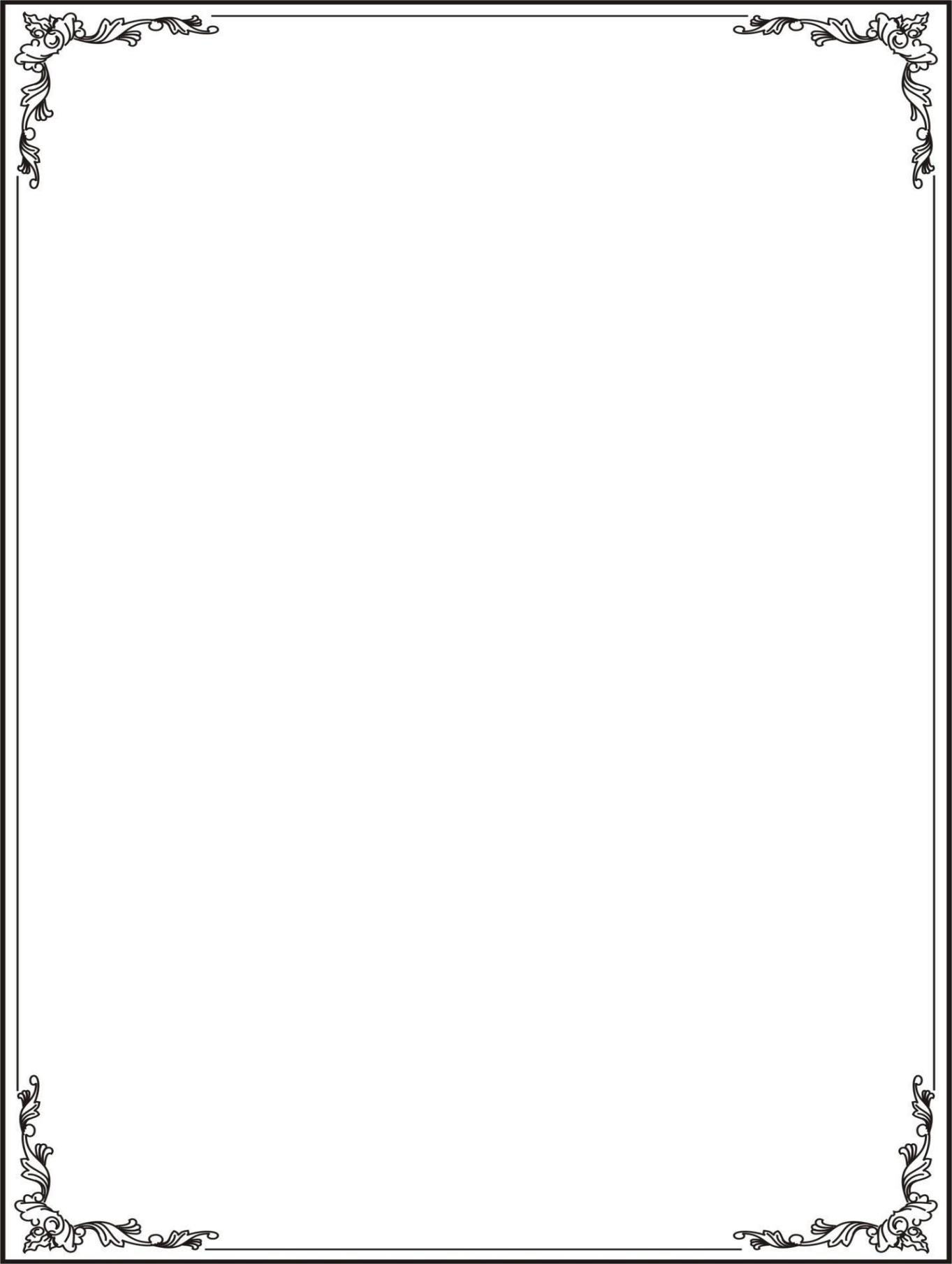
**Phan Cảnh Tuấn Đạt - 3122410076**

**Trần Khánh Huyền - 3122410156**

**Dương Bình Minh - 3122410236**

**Nguyễn Hoàng Mai Vy - 3122410490**

**Huỳnh Văn Phú - 3122560057**



**ỦY BAN NHÂN DÂN THÀNH PHỐ HỒ CHÍ MINH TRƯỜNG ĐẠI HỌC SÀI GÒN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BÁO CÁO ĐỒ ÁN**

**MÔN: AN TOÀN BẢO MẬT DỮ LIỆU TRONG HTTT (841120)**

**TÊN ĐỀ TÀI**

**XÂY DỰNG WEB APPLICATION FIREWALL THÍCH ỨNG DỰA TRÊN LOG TẤN CÔNG**

**Nhóm báo cáo:**

**Nhóm lớp:**

**GVHD:**

**Danh sách thành viên:**

**TP Hồ Chí Minh, tháng 10 năm 2024**



**TÊN ĐỀ TÀI: Xây dựng Web Application Firewall thích ứng dựa trên log tấn công**

**NHÓM: 01**

**THÀNH VIÊN:**

|  |  |  |  |
| --- | --- | --- | --- |
| **MÃ SV** | **HỌ TÊN** | **TỈ LỆ PHẦN TRĂM** | **CÔNG VIỆC THỰC HIỆN**  **(Liệt kê công việc dự kiến thực hiện trong đồ án)** |
| 3122410076 | Phan Cảnh Tuấn Đạt | 12,5% |  |
| 3122410156 | Trần Khánh Huyền | 12,5% |  |
| 3122410236 | Dương Bình Minh | 12,5% |  |
| 3122410490 | Nguyễn Hoàng Mai Vy | 12,5% |  |
| 3122560057 | Huỳnh Văn Phú | 12,5% |  |

**MỤC LỤC**

# **DANH MỤC THUẬT NGỮ**

.pkl (Pickle): là một định dạng tệp trong Python dùng để lưu trữ các đối tượng (objects) - như model machine learning, dataframe, list, dictionary, hoặc bất kỳ cấu trúc dữ liệu nào - dưới dạng nhị phân.

.csv (Comma-Separated Values): là định dạng văn bản (text file) dùng để lưu trữ dữ liệu dạng bảng — giống như bảng Excel — trong đó các cột được ngăn cách bằng dấu phẩy (,).

Raw

Malicious

BOM

non-malicious

preprocessing

payload

# **DANH MỤC HÌNH ẢNH**

# **LỜI MỞ ĐẦU**

# **Chương 1. Giới thiệu chung**

## **Lý do chọn đề tài**

Trong bối cảnh công nghệ thông tin phát triển mạnh mẽ, các ứng dụng web ngày càng trở thành mục tiêu hấp dẫn của tội phạm mạng. Hàng loạt hình thức tấn công như SQL Injection, Cross-site Scripting (XSS), Path Traversal, Remote Code Execution (RCE),… liên tục được phát hiện và lợi dụng để chiếm đoạt dữ liệu hoặc làm gián đoạn dịch vụ. Các hệ thống tường lửa truyền thống chủ yếu hoạt động ở tầng mạng (Network Layer) chỉ có khả năng lọc địa chỉ IP, cổng hoặc giao thức, chưa đủ khả năng nhận diện và ngăn chặn các mối đe dọa phức tạp ở tầng ứng dụng. Vì vậy, nhu cầu về một tường lửa ứng dụng web (Web Application Firewall – WAF) ngày càng trở nên cấp thiết. Đề tài “Web Application Firewall thích ứng dựa trên log tấn công” được lựa chọn với mong muốn xây dựng một mô hình tường lửa không chỉ phát hiện và chặn các cuộc tấn công thông qua các quy tắc có sẵn, mà còn có khả năng học hỏi và thích ứng dựa trên dữ liệu log thực tế. Hệ thống này có thể tự cải thiện hiệu quả phát hiện theo thời gian, góp phần tăng cường tính an toàn cho các ứng dụng web hiện nay.

* 1. **Mục tiêu**

Đề tài hướng tới việc xây dựng và triển khai một hệ thống WAF thích ứng, kết hợp giữa phương pháp truyền thống dựa trên luật (rule-based) và phương pháp học máy (machine learning).

Cụ thể, các mục tiêu bao gồm:

* Xây dựng kiến trúc WAF cơ bản gồm các thành phần: engine phát hiện tấn công, module ghi log, và API quản trị.
* Sử dụng mô hình học máy (sử dụng thuật toán như Logistic Regression, XGBoost hoặc LightGBM) để phân tích và phân loại lưu lượng web thành hợp lệ hoặc tấn công.
* Thu thập và xử lý dữ liệu log để huấn luyện, đánh giá và cải thiện độ chính xác của hệ thống.
* Thiết kế giao diện quản trị (admin-ui) đơn giản cho phép quản lý log, theo dõi kết quả phát hiện và cập nhật mô hình.
* Tích hợp cơ chế tự học giúp hệ thống thích ứng với các kiểu tấn công mới chưa có trong bộ quy tắc ban đầu.
  1. **Phạm vi**

Phạm vi của đề tài tập trung vào việc phát triển và mô phỏng hoạt động của một Web Application Firewall trong môi trường cục bộ (local), bao gồm:

* Backend được xây dựng bằng Flask (Python), chịu trách nhiệm xử lý request, phát hiện tấn công và ghi log.
* Module học máy trong thư mục machine\_learning/ đảm nhiệm việc huấn luyện và dự đoán hành vi tấn công.
* Admin UI được thiết kế bằng HTML/CSS/JS nhằm hiển thị log, tình trạng tấn công và cấu hình hệ thống.
* Dữ liệu log mô phỏng các request hợp lệ và tấn công (SQLi, XSS, Path Traversal, …).
* Hệ thống được triển khai và thử nghiệm ở mức ứng dụng web nhỏ, chưa áp dụng vào môi trường thực tế quy mô lớn.

# **Chương 2. Tìm hiểu về tường lửa**

## **Khái niệm tường lửa**

### **Định nghĩa tường lửa**

Tường lửa là một thiết bị bảo mật mạng có chức năng phân tách mạng nội bộ đáng tin cậy với các mạng bên ngoài được xem là không an toàn, chẳng hạn như Internet. Thiết bị này kiểm soát lưu lượng mạng vào và ra dựa trên các quy tắc bảo mật được thiết lập sẵn. Tường lửa đóng vai trò then chốt trong việc bảo vệ hệ thống mạng khỏi truy cập trái phép, các hoạt động độc hại và các mối đe dọa tiềm ẩn, đồng thời có thể tồn tại dưới nhiều hình thức khác nhau như phần cứng, phần mềm, dịch vụ phần mềm (SaaS) hoặc được triển khai trên môi trường điện toán đám mây công cộng hoặc riêng tư.

Tường lửa thực hiện việc kiểm tra các gói tin trong mạng và áp dụng các chính sách bảo mật nhằm ngăn chặn người dùng không được phép hoặc dữ liệu độc hại xâm nhập hay rời khỏi hệ thống mạng. Với vai trò như một “người gác cổng”, tường lửa phân tích từng gói tin và quyết định cho phép hoặc chặn lại dựa trên các quy tắc đã được thiết lập, qua đó đảm bảo chỉ những luồng dữ liệu an toàn và hợp lệ mới được phép đi qua.

Bên cạnh các chức năng cốt lõi, tường lửa thế hệ mới (Next-Generation Firewall – NGFW) còn được trang bị nhiều tính năng nâng cao nhằm tăng cường khả năng bảo vệ mạng. Các tính năng này bao gồm kiểm tra sâu gói tin (Deep Packet Inspection), giám sát và kiểm soát ứng dụng, phát hiện và ngăn chặn xâm nhập, phòng chống phần mềm độc hại, lọc địa chỉ URL, cùng nhiều khả năng bảo mật khác.[1]

### **Vai trò của tường lửa trong bảo mật mạng**

Việc triển khai tường lửa có thể nâng cao đáng kể mức độ an toàn của hệ thống mạng cũng như dữ liệu nhạy cảm. Tường lửa mang lại khả năng bảo vệ trước các mối đe dọa và cuộc tấn công mạng phổ biến, trở thành một thành phần không thể thiếu trong bối cảnh các hình thức tấn công mạng ngày càng tinh vi.

Bên cạnh việc đảm bảo an ninh mạng, tường lửa còn giúp các tổ chức đáp ứng các yêu cầu tuân thủ và thực hành theo những tiêu chuẩn tốt nhất của ngành. Nhiều khung quy định yêu cầu bắt buộc phải sử dụng tường lửa để bảo vệ thông tin khách hàng nhạy cảm. Thông qua việc triển khai tường lửa, tổ chức có thể đáp ứng các yêu cầu tuân thủ này và tránh các hình phạt tiềm ẩn. [1]

Tường lửa đóng vai trò:

* Lọc gói tin (Packet Filtering): Tường lửa kiểm tra các gói dữ liệu đi vào hoặc ra khỏi mạng và áp dụng các quy tắc được thiết lập sẵn để quyết định cho phép hoặc chặn lại. Việc lọc này dựa trên các tiêu chí như địa chỉ IP nguồn và đích, cổng (port) và giao thức truyền thông.
* Kiểm tra trạng thái (Stateful Inspection): Tường lửa có khả năng theo dõi trạng thái của các kết nối đang hoạt động và đưa ra quyết định dựa trên ngữ cảnh của lưu lượng mạng. Nhờ đó, tường lửa có thể hiểu rõ bối cảnh của từng gói dữ liệu và thực hiện các hành động kiểm soát chính xác hơn.
* Ủy quyền truy cập và biên dịch địa chỉ mạng (Proxying và NAT): Tường lửa có thể hoạt động như một máy chủ trung gian (proxy) giữa mạng nội bộ và mạng bên ngoài, giúp che giấu cấu trúc của hệ thống mạng bên trong. Cơ chế NAT (Network Address Translation) cho phép nhiều thiết bị trong mạng nội bộ cùng chia sẻ một địa chỉ IP công cộng duy nhất.
* Hỗ trợ mạng riêng ảo (VPN): Nhiều tường lửa được tích hợp chức năng VPN nhằm bảo mật các kết nối truyền dữ liệu qua Internet. VPN mã hóa dữ liệu, đảm bảo quá trình trao đổi thông tin giữa người dùng từ xa và mạng nội bộ được an toàn và bảo mật. [2]

### **Các loại tường lửa**

**Tường lửa lọc gói tin (Packet Filtering Firewall)**

Loại tường lửa này kiểm tra từng gói dữ liệu đi qua và thực hiện lọc dựa trên các tham số như địa chỉ IP nguồn và đích, số hiệu cổng (port), và loại giao thức. Mặc dù có cấu trúc đơn giản và chi phí thấp, tường lửa lọc gói tin không thể phân tích nội dung bên trong của gói dữ liệu, do đó kém hiệu quả hơn trong việc đối phó với các cuộc tấn công tinh vi.

**Tường lửa ủy quyền (Proxy Firewall)**

Tường lửa ủy quyền là một trong những dạng tường lửa ra đời sớm, hoạt động như một cổng trung gian giữa hai mạng cho một ứng dụng cụ thể. Máy chủ proxy có thể cung cấp các chức năng bổ sung như lưu trữ tạm thời nội dung (caching) và tăng cường bảo mật bằng cách ngăn chặn các kết nối trực tiếp từ bên ngoài vào mạng nội bộ. Tuy nhiên, điều này có thể làm giảm hiệu suất truyền tải và giới hạn số lượng ứng dụng mà hệ thống có thể hỗ trợ.

**Tường lửa kiểm tra trạng thái (Stateful Inspection Firewall)**

Hiện nay được xem là dạng tường lửa truyền thống, tường lửa kiểm tra trạng thái cho phép hoặc chặn lưu lượng dựa trên trạng thái, cổng và giao thức. Nó theo dõi toàn bộ hoạt động của một kết nối từ khi mở đến khi đóng. Quyết định lọc được thực hiện dựa trên các quy tắc do quản trị viên xác định kết hợp với ngữ cảnh, tức là sử dụng thông tin từ các kết nối và gói dữ liệu liên quan trước đó.

**Tường lửa ứng dụng web (Web Application Firewall – WAF)**

Tường lửa ứng dụng web hoạt động như một trung gian giữa mạng nội bộ và mạng bên ngoài, xử lý tất cả các yêu cầu giao tiếp thay cho mạng nội bộ. WAF mang lại mức độ bảo mật cao do có khả năng kiểm tra nội dung của các gói dữ liệu và loại bỏ dữ liệu độc hại hoặc không được phép. Tuy nhiên, việc phụ thuộc vào máy chủ proxy có thể làm tăng độ trễ và ảnh hưởng đến hiệu suất mạng.

**Tường lửa quản lý mối đe dọa hợp nhất (Unified Threat Management – UTM Firewall)**

Thiết bị UTM thường tích hợp các chức năng của tường lửa kiểm tra trạng thái với hệ thống ngăn chặn xâm nhập (IPS) và chống virus. Ngoài ra, UTM còn có thể bao gồm nhiều dịch vụ mở rộng khác, thậm chí được quản lý thông qua nền tảng đám mây. Ưu điểm chính của UTM là sự đơn giản và dễ sử dụng trong quản lý bảo mật tổng thể.

**Tường lửa thế hệ mới (Next-Generation Firewall – NGFW)**

Tường lửa thế hệ mới (NGFW) là thiết bị bảo mật mạng cung cấp nhiều tính năng vượt trội so với tường lửa truyền thống. Trong khi tường lửa truyền thống chỉ thực hiện kiểm tra trạng thái lưu lượng, NGFW bổ sung các khả năng như nhận diện và kiểm soát ứng dụng, hệ thống ngăn chặn xâm nhập (IPS), lọc URL theo vị trí địa lý và độ uy tín, cùng với trí tuệ về mối đe dọa (threat intelligence). NGFW giúp đơn giản hóa công tác quản trị và giảm thiểu độ phức tạp nhờ các chính sách bảo mật thống nhất, bảo vệ toàn diện trong suốt vòng đời của mối đe dọa.

**Tường lửa ứng dụng trí tuệ nhân tạo (AI-powered Firewall)**

Tường lửa ứng dụng trí tuệ nhân tạo (AI) và học máy (ML) để nâng cao khả năng bảo vệ và giám sát mạng. Khác với tường lửa truyền thống chỉ dựa vào các quy tắc được xác định sẵn, tường lửa AI hoạt động theo thời gian thực để phân tích lưu lượng mạng động, nhận diện các mẫu hành vi bất thường và hỗ trợ tự động hóa quá trình quản lý vòng đời của chính sách tường lửa.

**Tường lửa ảo (Virtual Firewall)**

Tường lửa ảo thường được triển khai dưới dạng một thiết bị ảo (virtual appliance), có thể được lưu trữ tại chỗ trong môi trường đám mây riêng dựa trên các nền tảng như VMware ESXi, Microsoft Hyper-V, KVM, OpenStack hoặc Nutanix. Ngoài ra, tường lửa ảo cũng có thể được triển khai trong môi trường đám mây công cộng như Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP) hay Oracle Cloud Infrastructure (OCI). Với tường lửa ảo, doanh nghiệp có thể bảo vệ ứng dụng và dữ liệu của mình trên môi trường đa đám mây (multicloud) thông qua chính sách bảo mật thống nhất, quản lý tập trung và khả năng phòng thủ nâng cao trước các mối đe dọa. [1]

## **Web Application Firewall (WAF)**

### **Khái niệm WAF**

Tường lửa ứng dụng web (Web Application Firewall – WAF) là một lớp phòng thủ bảo mật quan trọng đối với các trang web, ứng dụng di động và API. WAF có nhiệm vụ giám sát, lọc và chặn các gói dữ liệu ra vào ứng dụng web nhằm bảo vệ chúng khỏi các mối đe dọa tiềm ẩn. Các WAF được thiết kế (hoặc huấn luyện) để phát hiện và ngăn chặn những lỗ hổng bảo mật nguy hiểm thường xuất hiện trong lưu lượng truy cập web. Nhờ đó, WAF trở thành yếu tố không thể thiếu đối với các doanh nghiệp trực tuyến như bán lẻ, ngân hàng, y tế và mạng xã hội — những lĩnh vực cần bảo vệ dữ liệu nhạy cảm khỏi truy cập trái phép. WAF có thể được triển khai dưới nhiều hình thức khác nhau, bao gồm dạng mạng (network-based), dạng máy chủ (host-based) hoặc dạng đám mây (cloud-based), giúp cung cấp khả năng giám sát dữ liệu ở tầng ứng dụng HTTP.

Do các ứng dụng web, ứng dụng di động và API thường dễ bị tấn công, gây gián đoạn hoạt động hoặc làm cạn kiệt tài nguyên, tường lửa ứng dụng web được thiết kế để chống lại các hình thức khai thác phổ biến trên web như bot độc hại. WAF giúp bảo vệ hệ thống trước các mối đe dọa ảnh hưởng đến tính sẵn sàng, bảo mật hoặc tài nguyên của ứng dụng, bao gồm cả các cuộc tấn công **zero-day**, **bot**, và **phần mềm độc hại (malware)**. [3]

* + 1. **Khái quát cách hoạt động**

Một WAF hoạt động bằng cách kiểm tra các yêu cầu HTTP và áp dụng các quy tắc được thiết lập sẵn để phát hiện lưu lượng độc hại. WAF có thể tồn tại dưới dạng phần mềm, thiết bị phần cứng hoặc dịch vụ. Cơ chế của WAF tập trung vào việc phân tích các thành phần chính trong quá trình trao đổi HTTP như sau:

* Yêu cầu GET: Dùng để truy xuất dữ liệu từ máy chủ.
* Yêu cầu POST: Dùng để gửi dữ liệu đến máy chủ nhằm thay đổi trạng thái của hệ thống.
* Yêu cầu PUT: Dùng để gửi dữ liệu nhằm cập nhật hoặc tạo mới thông tin trên máy chủ.
* Yêu cầu DELETE: Dùng để xóa dữ liệu khỏi máy chủ.

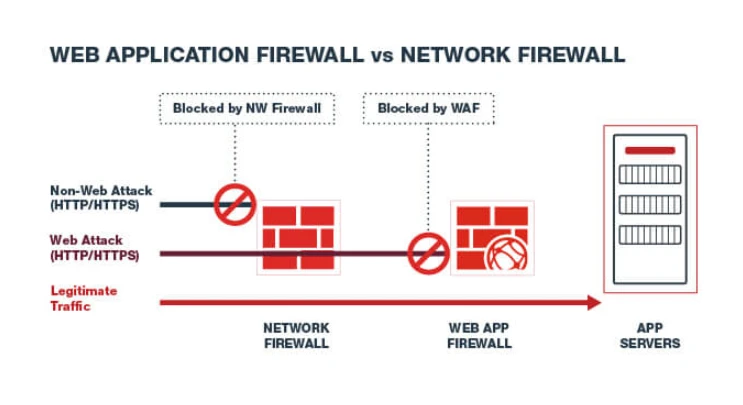
Ngoài ra, WAF còn phân tích phần header, chuỗi truy vấn (query string) và nội dung (body) của các yêu cầu HTTP để phát hiện các mẫu hành vi độc hại. Nếu phát hiện yêu cầu có dấu hiệu tấn công, WAF sẽ chặn yêu cầu đó và gửi cảnh báo đến đội ngũ bảo mật để xử lý.

### **So sánh WAF với Network Firewall**

Tường lửa ứng dụng web (Web Application Firewall – WAF) và tường lửa mạng (Network Firewall) đều đóng vai trò quan trọng trong việc bảo vệ hệ thống, tuy nhiên chúng hoạt động ở những tầng khác nhau của mô hình OSI và có mục tiêu bảo vệ riêng biệt.

WAF tập trung vào việc bảo vệ các ứng dụng web, API và lưu lượng HTTP/HTTPS khỏi các mối đe dọa ở tầng ứng dụng (Layer 7). Nó được đặt giữa người dùng bên ngoài và ứng dụng web để giám sát, phân tích toàn bộ lưu lượng truy cập, đồng thời phát hiện và chặn các yêu cầu độc hại trước khi chúng có thể gây ảnh hưởng đến máy chủ hoặc dữ liệu người dùng. WAF thường được sử dụng để ngăn chặn các hình thức tấn công phổ biến như SQL Injection, Cross-Site Scripting (XSS), tấn công DDoS tầng 7, hay xâm nhập thông qua lỗ hổng ứng dụng web.

Trong khi đó, Network Firewall bảo vệ mạng nội bộ (LAN) khỏi truy cập trái phép và các cuộc tấn công đến từ bên ngoài. Thiết bị này hoạt động ở tầng mạng và tầng truyền tải (Layer 3 và Layer 4), nơi diễn ra quá trình truyền tải dữ liệu, giúp kiểm soát và lọc các gói tin dựa trên địa chỉ IP, cổng, hoặc giao thức. Mục tiêu chính của tường lửa mạng là tạo ranh giới an toàn giữa vùng mạng tin cậy và vùng mạng kém an toàn, ngăn chặn các hành vi tấn công như Man-in-the-Middle (MITM), truy cập trái phép, hay leo thang đặc quyền trong hệ thống.



Hình 2.1. Khác biệt giữa WAF và Network Firewall

## **Phương pháp phát hiện tấn công trong WAF**

### **Signature-based (Dựa trên mẫu tấn công)**

Cơ chế phát hiện dựa trên mẫu tấn công là phương pháp được sử dụng phổ biến trong hầu hết các hệ thống WAF hiện nay. Theo đó, chính sách bảo mật của WAF được xây dựng dựa trên một tập hợp các mẫu tấn công – mỗi mẫu tấn công là một mẫu đặc trưng của một dạng tấn công đã được nhận diện trước đó. WAF sẽ tiến hành so sánh các yêu cầu từ phía người dùng và phản hồi từ máy chủ với danh sách chữ ký này nhằm phát hiện những mẫu trùng khớp. Khi phát hiện hành vi đáng ngờ, hệ thống có thể phát cảnh báo hoặc tự động chặn lưu lượng truy cập tùy theo cấu hình bảo mật đã thiết lập.

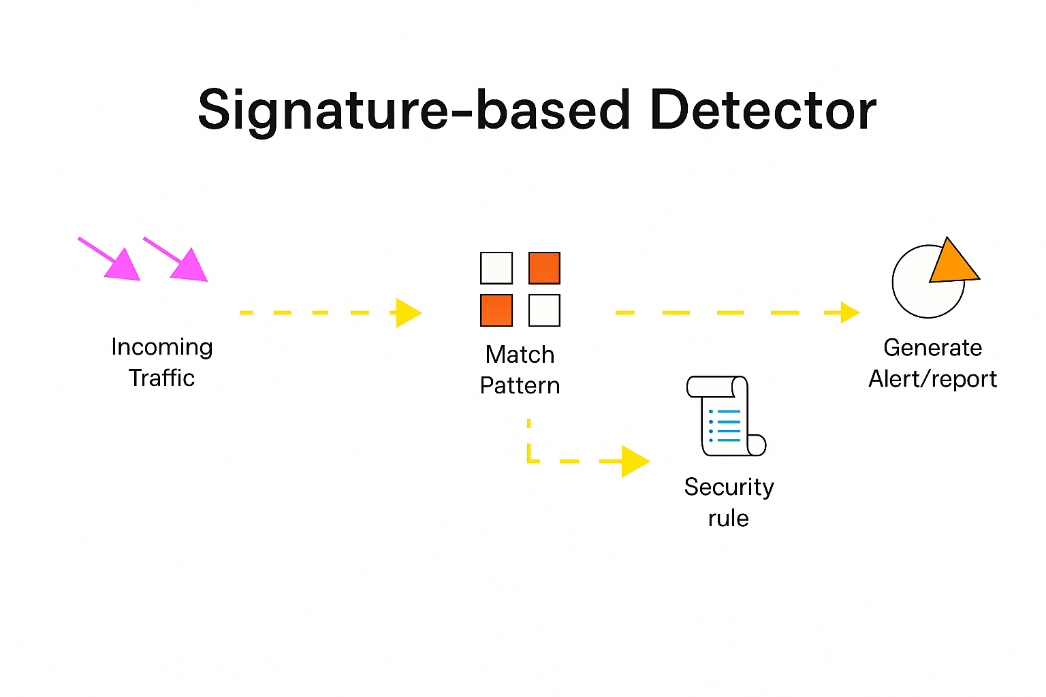
Ưu điểm:

Phương pháp dựa trên mẫu tấn công đã được áp dụng từ lâu trong các hệ thống tường lửa và phần mềm chống mã độc. Ưu điểm nổi bật của nó là dễ dàng triển khai và cập nhật khi xuất hiện các hình thức tấn công mới. Mỗi mẫu tấn công chỉ mô tả một dạng tấn công cụ thể, vì vậy việc nhận diện và xử lý tương đối chính xác và hiệu quả, đặc biệt trong những trường hợp cần ngăn chặn một mối đe dọa xác định rõ ràng. Nhờ tính đặc thù này, tỷ lệ cảnh báo sai cũng được giảm thiểu đáng kể.

Hạn chế:

Tuy nhiên, để đảm bảo hiệu quả, hệ thống cần liên tục cập nhật khi xuất hiện các kỹ thuật tấn công mới. Theo thời gian, số lượng mẫu gia tăng nhanh chóng khiến WAF phải xử lý hàng nghìn mẫu khác nhau, dẫn đến tiêu tốn tài nguyên và làm giảm hiệu suất của ứng dụng. Ngoài ra, việc tích hợp quá nhiều chữ ký có thể khiến một số lưu lượng hợp lệ bị chặn nhầm, làm tăng tỷ lệ cảnh báo sai.

Giải pháp lý tưởng là chỉ duy trì những mẫu thực sự cần thiết và phù hợp với môi trường ứng dụng, song việc cân bằng giữa mức độ bảo mật và hiệu năng không hề đơn giản. Một số nhà cung cấp dịch vụ đã thử nghiệm ứng dụng trí tuệ nhân tạo và machine learning để tự động phân tích hệ thống trong giai đoạn triển khai ban đầu, nhằm tối ưu hóa chính sách bảo mật. Tuy nhiên, phương pháp này vẫn chưa đảm bảo mang lại kết quả tối ưu trong mọi trường hợp và có thể làm tăng chi phí triển khai ban đầu. [5][6]



Hình 2.1. Phương pháp Signature-based

* + 1. **Rule-based (Dựa trên quy tắc)**

Phương pháp phát hiện dựa trên quy tắc là một hướng tiếp cận mới nổi trong lĩnh vực WAF, được phát triển nhờ vào sự hỗ trợ của trí tuệ nhân tạo (AI). So với mô hình dựa trên mẫu tấn công, cách tiếp cận này phụ thuộc nhiều hơn vào công nghệ tự động hóa và ít cần can thiệp thủ công.

Thông thường, mẫu (signature) được tạo ra bằng cách ghi nhận cấu trúc cú pháp của mã nguồn tấn công. Trong khi đó, quy tắc (rule) không được xác định trực tiếp từ mẫu tấn công cụ thể. Thay vào đó, AI sẽ phân tích nhiều mẫu tấn công khác nhau và sử dụng công cụ suy luận (inference engine) để rút ra các quy tắc mang tính khái quát – có khả năng đại diện cho nhiều mẫu tấn công. Do đó, một quy tắc duy nhất có thể bao quát hàng trăm mẫu riêng lẻ, có thể hiểu rằng quy tắc chính là “mẫu của các mẫu tấn công”.

Ưu điểm:

Trong khi một chính sách bảo mật dựa trên mẫu có thể cần đến 2.000 – 8.000 mẫu, thì một chính sách bảo mật dựa trên quy tắc chỉ cần vài chục quy tắc để phát hiện số lượng tấn công tương đương.

Ưu điểm lớn nhất của WAF dựa trên quy tắc là tốc độ và hiệu năng xử lý. Theo kết quả thử nghiệm nội bộ của nhóm nghiên cứu tại Penta Security, một hệ thống WAF với 27 quy tắc có thể chặn thành công 95% các cuộc tấn công, tương đương với hiệu quả của hệ thống dùng 8.000 chữ ký. Về mặt hiệu năng, hệ thống dùng 27 quy tắc chỉ làm giảm 20% tốc độ xử lý, trong khi hệ thống dùng 8.000 chữ ký khiến hiệu năng giảm đến 50%.

Bên cạnh đó, WAF dựa trên quy tắc dễ bảo trì hơn nhiều. Sau khi bộ quy tắc ban đầu được thiết lập, hệ thống gần như không cần cập nhật thủ công thường xuyên; nhà cung cấp dịch vụ chỉ cần điều chỉnh hoặc bổ sung quy tắc mới khi cần thiết.

Ngoài ra, WAF loại này còn có thể ngăn chặn một số cuộc tấn công zero-day, do các quy tắc được xây dựng đủ linh hoạt để phát hiện những biến thể nhỏ trong mẫu tấn công.

Hạn chế:

Một số chuyên gia vẫn ưa chuộng hệ thống dựa trên mẫu, bởi chúng mang tính bảo thủ và cho phép can thiệp thủ công nhiều hơn. Trong khi đó, phương pháp dựa trên quy tắc phụ thuộc nhiều vào trí tuệ nhân tạo, khiến một số người cảm thấy thiếu quyền kiểm soát trực tiếp.

Ngoài ra, việc chỉ sử dụng số lượng quy tắc ít cũng có thể khiến người dùng nghi ngờ về khả năng phát hiện tấn công. Tuy nhiên, cả kết quả thử nghiệm trong phòng lab lẫn thực tế triển khai đều cho thấy rằng WAF dựa trên quy tắc không hề thua kém về năng lực phát hiện so với phương pháp truyền thống.

A diagram of a computer system

AI-generated content may be incorrect.

Hình 2.2. Khác biệt giữa Signature-based và Rule-based WAF

### **Anomaly-based (Dựa trên hành vi bất thường)**

Các quy tắc phát hiện bất thường trong WAF tìm kiếm những mẫu hành vi không bình thường trong lưu lượng truy cập. Những mẫu này có thể bao gồm sự gia tăng đột ngột về lưu lượng, các yêu cầu đến từ những vị trí bất thường, hoặc các yêu cầu chứa ký tự lạ. Khi một hành vi bất thường được phát hiện, quy tắc của WAF có thể chặn hoặc ghi lại yêu cầu đó để phục vụ cho việc điều tra sau này.

Các quy tắc phát hiện bất thường có khả năng phát hiện các cuộc tấn công mới hiệu quả hơn so với các quy tắc dựa trên chữ ký, vì chúng không bị giới hạn trong các mẫu tấn công đã biết. Tuy nhiên, quy tắc phát hiện bất thường cũng có thể tạo ra các cảnh báo sai (false positives).

Ưu điểm:

Phòng thủ chủ động trước các mối đe dọa chưa biết: Có khả năng nhận diện hành vi bất thường từ bất kỳ nguồn nào, giúp phát hiện hiệu quả các mối đe dọa mới và chưa từng được nhận dạng. Cách tiếp cận chủ động này đặc biệt quan trọng trong môi trường an ninh mạng luôn thay đổi.

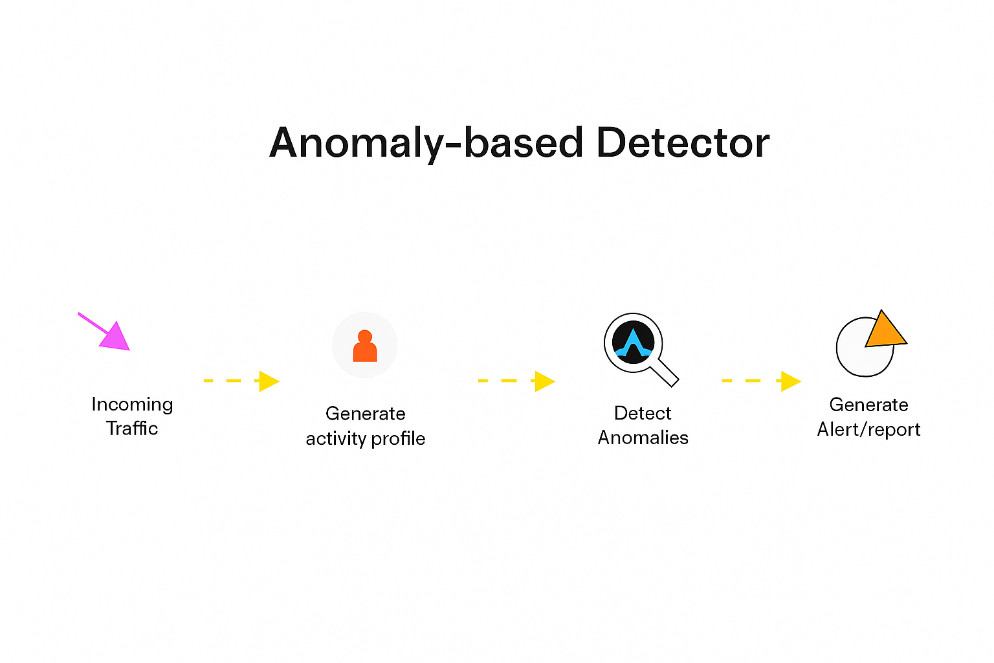
Giảm khối lượng bảo trì: Cơ chế phát hiện dựa trên bất thường có khả năng thích nghi với môi trường ứng dụng, giúp giảm nhu cầu cập nhật thường xuyên và hạn chế sự can thiệp thủ công trong việc duy trì hoạt động của WAF.

Bảo vệ toàn diện: Bằng cách giám sát các hành vi bất thường, hệ thống có thể phát hiện các mối đe dọa vượt qua cơ chế nhận dạng dựa trên chữ ký, tăng cường mức độ bảo mật bằng cách bao quát nhiều rủi ro tiềm ẩn và giảm khả năng bỏ sót các phương thức tấn công phức tạp.

Hạn chế:

Tăng tỷ lệ cảnh báo sai: Mặc dù phát hiện tốt các hoạt động bất thường, hệ thống dựa trên bất thường có thể tạo ra nhiều cảnh báo sai hơn. Các hành vi bình thường nhưng khác biệt so với chuẩn ban đầu có thể bị nhận diện nhầm là mối đe dọa, đòi hỏi sự kiểm tra kỹ lưỡng hơn.

Độ phức tạp và yêu cầu tài nguyên cao: Hệ thống này cần khả năng tính toán mạnh và các thuật toán tiên tiến để phát hiện chính xác các bất thường, điều này có thể làm tăng yêu cầu về phần cứng, phần mềm và ảnh hưởng đến hiệu năng cũng như khả năng mở rộng của hệ thống. [6]



Hình 2.2. Phương pháp Anomaly-based

### **Machine Learning / AI trong WAF**

Các hướng tiếp cận dựa trên học máy/học sâu (ML/DL) đã được áp dụng trong lĩnh vực nhận diện tấn công web và đạt được những thành công nhất định. Các kỹ thuật AI này có tiềm năng trong việc giải quyết các vấn đề an ninh mạng.

Một số hướng tiếp cận có thể kể đến:

* Xử lý Ngôn ngữ Tự nhiên (Natural Language Processing - NLP)

Các kỹ thuật NLP có thể cung cấp những cách hiệu quả để thu thập thông tin có giá trị từ việc phân tích các chuỗi từ, đặc biệt là từ URL và các tham số hoặc weblog vốn là thông tin dựa trên văn bản.

* Giải pháp Phát hiện Tấn công Web dựa trên DL (DL-WAD)

DL-WAD là một giải pháp mới được đề xuất, sử dụng học sâu (DL) và các kỹ thuật xử lý ngôn ngữ tự nhiên (NLP) để phát hiện tấn công web.

DL-WAD được thiết kế với cơ chế tiền xử lý dữ liệu nhằm phân biệt giữa các yêu cầu web bình thường và yêu cầu độc hại có chứa payload tấn công bao gồm nhiều loại tấn công web khác nhau.

Cơ chế tiền xử lý dữ liệu bao gồm các bước: chuẩn hóa dữ liệu (Data Normalization), nhúng từ (Word embedding) và token hóa từ (Word Tokenization).

Mô hình DL được sử dụng là Bidirectional Long Short-Term Memory (Bi-LSTM).

* Bi-LSTM

Bi-LSTM là một mô hình học sâu liên quan đến NLP, đã đạt được những kết quả tiềm năng trong phân loại. Mô hình này được sử dụng trong các tác vụ liên quan đến NLP nhờ khả năng xử lý dữ liệu theo cả hai hướng để hiểu rõ hơn mối quan hệ giữa các chuỗi. [7]

## **Kiến trúc và thành phần của WAF**

### **Các kiến trúc triển khai WAF phổ biến**

**Chế độ Inline hoặc Bridge**

Chế độ Inline hoặc Bridge thường được sử dụng trong các môi trường doanh nghiệp, nơi yếu tố bảo mật, hiệu năng và khả năng ngăn chặn mối đe dọa theo thời gian thực đóng vai trò then chốt. Kiến trúc này đảm bảo rằng lưu lượng truy cập được phân tích và lọc một cách chủ động, giúp nó trở nên lý tưởng cho các mạng có thông lượng cao và độ trễ thấp.

Trong mô hình này, WAF được đặt trực tiếp giữa máy khách và ứng dụng web (có thể ở mức vật lý hoặc logic) trong đường truyền dữ liệu. Điều này đồng nghĩa với việc toàn bộ lưu lượng vào và ra đều phải đi qua WAF. Nhờ đó, đội ngũ bảo mật có được khả năng quan sát và kiểm soát toàn bộ quá trình trao đổi dữ liệu. Ngoài ra, vị trí này cho phép WAF kiểm tra và chặn lưu lượng theo thời gian thực, mang lại khả năng phòng ngừa mối đe dọa chủ động.

Tuy nhiên, cách triển khai này có thể trở thành một điểm lỗi tiềm ẩn, vì vậy các cấu hình dự phòng như active-active hoặc active-passive thường được áp dụng. Nếu WAF gặp sự cố, lưu lượng có thể bị gián đoạn. Do đó, hệ thống sao lưu là yếu tố thiết yếu để đảm bảo tính sẵn sàng cao và tránh thời gian ngừng hoạt động.

**Cloud-based WAF**

WAF dựa trên đám mây là lựa chọn lý tưởng cho các doanh nghiệp cần khả năng bảo vệ mạnh mẽ mà không phải quản lý hạ tầng tại chỗ. Các giải pháp này được lưu trữ bởi các nhà cung cấp bên ngoài, và lưu lượng truy cập được định tuyến qua mạng của họ để kiểm tra trước khi đến máy chủ. Điều này có nghĩa là các mối đe dọa được loại bỏ từ sớm mà không tiêu tốn tài nguyên nội bộ.

Mô hình này có khả năng mở rộng cao và được triển khai nhanh chóng, phù hợp với các doanh nghiệp hoạt động toàn cầu. Theo Coherent Market Insights, mô hình triển khai WAF dựa trên đám mây sẽ chiếm khoảng 61,3% thị phần toàn cầu vào năm 2025. Ngoài ra, nhiều nhà cung cấp còn tích hợp thêm các dịch vụ bổ sung như bảo vệ DDoS, giám sát liên tục và cung cấp thông tin về mối đe dọa theo thời gian thực để nâng cao hiệu quả bảo vệ. Tuy nhiên, vì WAF hoạt động bên ngoài môi trường doanh nghiệp, nó có thể hạn chế khả năng quan sát và kiểm soát của tổ chức.

**Appliance-based WAF**

WAF dạng thiết bị phù hợp với các nhóm cần tuân thủ nghiêm ngặt các yêu cầu bảo mật hoặc đạt được các mục tiêu hiệu năng cụ thể. Giải pháp này hữu ích cho các tổ chức muốn kiểm soát hoàn toàn cách thức lọc lưu lượng và quản lý bảo mật. Được triển khai tại chỗ, dưới dạng thiết bị vật lý hoặc máy ảo, mô hình này cho phép tích hợp với các công cụ hiện có và tùy chỉnh toàn diện các quy tắc và chính sách bảo mật.

Tuy nhiên, giải pháp này đi kèm với khối lượng công việc vận hành lớn. Do đó, chi phí bảo trì định kỳ, cập nhật thủ công và khả năng mở rộng hạn chế cần được cân nhắc kỹ lưỡng trong quá trình triển khai. [8]

### **Các thành phần chính:**

**Công cụ Phát hiện (Detection Engine)**

Công cụ phát hiện có trách nhiệm nhận diện và phản ứng lại lưu lượng truy cập có khả năng gây hại nhắm vào ứng dụng web. Nó sử dụng sự kết hợp của nhiều kỹ thuật:

* Phát hiện dựa trên mẫu tấn công (Signature-based detection): Khớp với các mẫu tấn công đã biết như các payload SQL injection phổ biến hoặc các script XSS điển hình. Các mẫu này được cập nhật thường xuyên để theo kịp các mối đe dọa mới nổi.
* Phát hiện dựa trên Heuristic hoặc Bất thường (Heuristic or anomaly-based detection): Phân tích hành vi lưu lượng truy cập và đánh dấu các sai lệch so với mức sử dụng bình thường. Ví dụ, nếu một người dùng đột ngột gửi một số lượng lớn yêu cầu với các tham số truy vấn bất thường, đó có thể là dấu hiệu của một cuộc tấn công đang diễn ra.
* Phân tích hành vi (Behavioral analysis): Giúp xác định các mối đe dọa tinh vi hoặc đang phát triển mà không khớp với các chữ ký hiện có, chẳng hạn như các lỗ hổng zero-day.
* Giới hạn tốc độ và lọc dựa trên danh tiếng (Rate limiting and reputation-based filtering): Có thể được tích hợp để chặn các IP hoặc bot độc hại đã biết.

Hiệu quả của công cụ phát hiện phụ thuộc vào chất lượng của bộ quy tắc và khả năng thích ứng với các mối đe dọa thay đổi mà không tạo ra quá nhiều kết quả dương tính giả (false positives).

**Máy chủ Proxy (Proxy Server)**

Máy chủ proxy đóng vai trò là cổng nơi tất cả lưu lượng HTTP đi và đến ứng dụng web đều đi qua. Vai trò chính của nó là cho phép kiểm tra và kiểm soát hoàn toàn lưu lượng này:

* Chế độ Proxy ngược (Reverse proxy mode): WAF đứng giữa các máy khách bên ngoài và ứng dụng, xử lý tất cả lưu lượng đến. Đây là thiết lập phổ biến nhất cho WAFs.
* Chế độ Proxy trong suốt (Transparent proxy) hoặc Inline: Lưu lượng được kiểm tra mà không làm thay đổi định tuyến, điều này đơn giản hóa việc triển khai nhưng có thể giới hạn một số khả năng.
* Làm sạch Yêu cầu (Sanitize requests): Máy chủ proxy có thể loại bỏ hoặc sửa đổi các yếu tố có khả năng gây hại trước khi chúng đến máy chủ ứng dụng.
* Kích hoạt các tính năng: Máy chủ proxy cũng kích hoạt các tính năng như chấm dứt SSL (SSL termination), quản lý phiên (session management), và định hình lưu lượng (traffic shaping).

Bằng cách tách biệt máy chủ ứng dụng khỏi việc tiếp xúc trực tiếp với internet, máy chủ proxy giảm thiểu bề mặt tấn công và tăng cường cả bảo mật lẫn hiệu suất.

**Giao diện Quản lý (Management Interface)**

Giao diện quản lý cung cấp khả năng hiển thị và kiểm soát tất cả các hoạt động của WAF. Nó rất cần thiết để cấu hình hệ thống, phản ứng với các mối đe dọa và đảm bảo bảo vệ liên tục.

* Quản lý chính sách (Policy management): Cho phép quản trị viên định nghĩa và tùy chỉnh các quy tắc để lọc lưu lượng, chặn tấn công và ghi lại sự kiện.
* Cấu hình chính sách bảo mật (Security policy configuration): Cho phép quản trị viên đặt các mục tiêu bảo vệ rộng, ví dụ như chặn tất cả các yêu cầu khớp với mẫu OWASP Top 10, và xác định mức độ thực thi—chặn, cảnh báo, hoặc ghi log.
* Quản lý quy tắc (Rules management): Hỗ trợ kiểm soát chi tiết cách WAF phản hồi với các loại lưu lượng truy cập cụ thể. Các quy tắc có thể được xây dựng tùy chỉnh hoặc dựa trên thư viện định nghĩa trước và điều chỉnh cho từng ứng dụng.
* Cảnh báo và báo cáo (Alerting and reporting): Thông báo cho các đội ngũ về các mối đe dọa được phát hiện, các nỗ lực bị chặn, và tình trạng hệ thống.
* Phân tích lưu lượng (Traffic analytics): Giúp xác định xu hướng tấn công và các lỗ hổng tiềm ẩn.
* Tích hợp với các hệ thống bên ngoài: Cho phép giám sát tập trung và phối hợp phản hồi (ví dụ: với SIEMs, threat intelligence feeds).
* Quản lý người dùng và vai trò (User and role management): Đảm bảo chỉ những nhân viên được ủy quyền mới có thể sửa đổi cấu hình hoặc truy cập dữ liệu nhạy cảm.[9]

## **Một số thách thức và vấn đề bảo mật liên quan**

### **Các hạn chế về hiệu suất và độ chính xác**

Dương tính giả (False positives): WAF có thể chặn lưu lượng truy cập hoặc yêu cầu hợp pháp nếu nó nhầm lẫn chúng là hoạt động độc hại. Điều này có thể gây ra vấn đề cho người dùng đang cố gắng truy cập ứng dụng. Quản trị viên có thể cần phải xem xét và thủ công cho phép (whitelist) các lưu lượng này.

Ảnh hưởng đến Hiệu suất (Performance impact): WAF có thể thêm gánh nặng (overhead) cho ứng dụng và ảnh hưởng đến hiệu suất của nó. Đây là một mối quan tâm đối với các ứng dụng quan trọng có khối lượng lưu lượng truy cập cao hoặc các yêu cầu nghiêm ngặt về hiệu suất.

### **Các thách thức về cấu hình và chuyên môn**

Yêu cầu Cấu hình Cẩn thận: WAF yêu cầu cấu hình cẩn thận để đạt được hiệu quả. Nếu WAF không được cấu hình chính xác, nó có thể không phát hiện và chặn được lưu lượng độc hại, khiến ứng dụng dễ bị tấn công.

Phức tạp và Tốn kém: WAF cần nhiều công sức hơn để thiết lập chính xác so với tường lửa tiêu chuẩn. Cần có kiến thức về ứng dụng để đảm bảo WAF bảo vệ ứng dụng một cách phù hợp.

Yêu cầu Chuyên gia: Trong thực tế, WAF đòi hỏi các chuyên gia về an ninh và ứng dụng để thiết lập và tận dụng tối đa hiệu quả. Nếu không có chuyên môn nội bộ, tổ chức sẽ phải thuê ngoài công việc này, khiến WAF trở thành một giải pháp không hề rẻ.

### **Hạn chế về phạm vi bảo vệ**

Bảo vệ Giới hạn: WAF chỉ bảo vệ chống lại các cuộc tấn công dựa trên web và có thể không hiệu quả đối với các loại tấn công khác.

Bỏ qua (Bypassing): WAF có thể bị qua mặt nếu kẻ tấn công sử dụng một kiểu tấn công chưa biết hoặc tấn công Zero-Day mà WAF chưa được cấu hình để phát hiện. Điều này nhấn mạnh tầm quan trọng của việc thường xuyên cập nhật và duy trì WAF để đảm bảo tính hiệu quả chống lại các loại mối đe dọa mới.

Không bao trùm toàn bộ các mối đe dọa: Mặc dù nguồn tài liệu mới này đề cập WAF có thể không hiệu quả chống lại SQL injection hoặc XSS, các nguồn trước đây trong cuộc hội thoại của chúng ta đã chỉ ra rằng các mối đe dọa này (SQL Injection, Cross-Site Scripting - XSS) là mục tiêu mà WAF được thiết kế để bảo vệ. Tuy nhiên, nếu WAF bị qua mặt bởi các biến thể hoặc zero-day thì khả năng bảo vệ sẽ bị hạn chế.

# **Chương 3. Giới thiệu về ứng dụng**

## **Giới thiệu tổng quan**

Ứng dụng Web Application Firewall (WAF) là hệ thống tường lửa sử dụng phương pháp Machine Learning trong WAF, với đặc điểm thích ứng được xây dựng nhằm tăng cường khả năng phát hiện và ngăn chặn tấn công web trong thời gian thực, đồng thời tự học và thích ứng dựa trên các log tấn công thu thập được.

Hệ thống kết hợp các kỹ thuật truyền thống (rule-based) và học máy (Machine Learning) để:

* Phát hiện những mẫu tấn công đã biết thông qua rules.json và biểu thức regex.
* Phân tích, đánh giá các request chưa từng gặp bằng mô hình XGBoost, giúp phát hiện tấn công mới.
* Tự động ghi log, huấn luyện và cập nhật mô hình để cải thiện độ chính xác qua thời gian.

## **Mô tả hệ thống**

* + 1. **Các thành phần chính của hệ thống**

1. **Flask**

Flask là một microframework web viết bằng [Python](https://200lab.io/blog/python-la-gi/) cực kì tối giản nhưng hiệu quả và được sử dụng để xây dựng API backend cho hệ thống.

Flask mang đến cho các lập trình viên quyền kiểm soát nhiều hơn, không bị ràng buộc bởi các công cụ hay thư viện mặc định mà  có thể tự do chọn các thư viện bên ngoài sao cho phù hợp nhất cho dự án. Flask đặt biệt phù hợp với những ai ưa thích sự linh hoạt và cấu trúc tối giản.

Các tính năng chính của Flask:

* Nhẹ và Linh Hoạt: Bạn có thể chọn chính xác những thành phần mà mình cần, không bị ép buộc phải dùng cấu trúc có sẵn.
* Server Tích Hợp: Flask có một server tích hợp sẵn, giúp kiểm thử ứng dụng ngay trên máy local.
* Jinja2 Templating Engine: Tích hợp công cụ Jinja2, hỗ trợ bạn tạo các trang web động một cách hiệu quả.
* Hỗ Trợ Tiện Ích Mở Rộng: Flask có nhiều tiện ích mở rộng, như xác thực, ORM, ...

A blue and black logo

AI-generated content may be incorrect.

Trong hệ thống này, Flask đóng vai trò trung gian giữa giao diện người dùng và các module xử lý ở tầng dưới (bao gồm phân tích log, mô hình học máy, và WAF engine).  
Cụ thể, Flask đảm nhiệm các chức năng:

* Tiếp nhận các yêu cầu HTTP (request) từ phía người dùng hoặc ứng dụng web.
* Gửi dữ liệu cần phân tích đến mô hình học máy hoặc module WAF để xử lý.
* Trả kết quả phân loại (ví dụ: *malicious* hoặc *benign*) về cho giao diện quản trị.
* Cung cấp các endpoint RESTful API cho phép tích hợp hệ thống với các ứng dụng khác.
* Ngoài ra, Flask hỗ trợ ghi log hoạt động, giúp quản trị viên theo dõi các yêu cầu đến và kết quả phân tích theo thời gian thực.

1. **Admin UI (Giao diện quản trị)**

Giao diện quản trị (Admin UI) được phát triển bằng HTML, CSS và JavaScript, đặt trong thư mục admin-ui/.  
Phần này cho phép người dùng hoặc quản trị viên:

* Quản lý các rule của WAF.
* Xem chi tiết các log được ghi lại bởi WAF.
* Cấu hình hoặc cập nhật các quy tắc bảo mật (security rules) được lưu trong file rules.json.

Admin UI hoạt động độc lập với backend nhưng giao tiếp thông qua các API do Flask cung cấp.

1. **Mô hình học máy**
2. **WAF**
   * 1. **Sơ đồ chức năng**
     2. **Luồng hoạt động**

**A diagram of a diagram

AI-generated content may be incorrect.**

## **Huấn luyện mô hình học máy**

* + 1. **Tập dữ liệu**

Tập dữ liệu được sử dụng cho quá trình huấn luyện bao gồm các tập hợp pattern thường xuất hiện trong các phương thức tấn công như SQL Injection, XSS, Shell, Reverse Shell và các payloads truy cập hợp lệ được tổng hợp lại thành các file văn bản .txt:

* SQLCollection.txt
* XSSCollection.txt
* ShellCollection.txt
* non-maliciousCollection.txt

Các tập dữ liệu được thu thập từ các nguồn hỗ trợ học thuật, nghiên cứu như Kaggle, OWASP và các sự kiện Capture the Flags (CTF) được công bố trên repository Github: <https://github.com/foospidy/payloads/tree/master>.

Mô tả chung về tập dữ liệu:

* SQLCollection.txt: SQLCollection.txt là tập hợp các payload (chuỗi tấn công) thường gặp trong các cuộc tấn công SQL Injection. Các payload trong file được thu thập từ nguồn học thuật, bài viết bảo mật và các cuộc thi CTF; mục đích là làm dataset mẫu cho huấn luyện mô hình phân loại payload (malicious vs benign). File chứa các biểu thức và chuỗi đại diện cho các kỹ thuật SQLi phổ biến như: boolean tautologies, comment injection, string-termination + boolean, subqueries, hàm DBMS đặc thù, chuỗi ghép/obfuscation, v.v.

**Một vài ví dụ điển hình (trích từ file):**

* + 'a' or 1=1--
  + "a"" or 1=1--"
  + or a = a
  + a' or 'a' = 'a
  + 1 or 1=1
  + '+(SELECT 1)+'
  + ' OR 'sp'='sp
  + superbug' AND 'a'='a
  + ' select dbms\_xmlgen.getxml('select \"a\" from sys.dual') from sys.dual;
  + ' select+dbms\_pipe.receive\_message((chr(95)||chr(96)||chr(97))+from+dual)

**Đặc điểm chính của tập dữ liệu**

Kiểu tấn công: chủ yếu là SQL Injection — bao gồm các dạng boolean-based, union-based, error-based, time-based (một số payload phức tạp có subquery/hàm DBMS).

Cấu trúc: nhiều payload có chuỗi ' hoặc " để phá vỡ literal trong truy vấn, tiếp đó là toán tử logic (OR, AND) sửa đổi điều kiện, hoặc chuỗi comment (--) để vô hiệu phần sau truy vấn.

Biến thể/obfuscation: có dạng ghép chuỗi ('+'(SELECT 1)+'), dùng hàm DBMS, hoặc dùng các biểu thức mã hóa (ví dụ chr()), mục tiêu là né detection dựa trên ký tự nguyên thủy.

Tính không đồng nhất: gồm cả payload ngắn (ví dụ 1 or 1=1) lẫn payload dài, có cú pháp đặc thù cho Oracle, MSSQL, MySQL, v.v.

Rủi ro false positive: một số chuỗi (ví dụ chứa từ or, and) có thể xuất hiện trong input hợp lệ → cần xử lý/normalization thận trọng khi gán nhãn.

* XSSCollection.txt: XSSCollection.txt chứa các payload đại diện cho các kỹ thuật tấn công Cross-Site Scripting (XSS). Các chuỗi trong file thể hiện nhiều biến thể XSS khác nhau (reflected, stored, DOM-based), dùng script tag, event handler, onerror/onload, payload dạng URI javascript:/data:, SVG/IMG injection, mã hóa ký tự (char codes / hex / percent-encoding) và các phương pháp obfuscation nhằm tránh nhận diện. File được thu thập từ nguồn học thuật/CTF và dùng làm dữ liệu huấn luyện để mô hình hóa khả năng phát hiện payload XSS.

**Một vài ví dụ điển hình (trích từ file):**

* data:text/html;alert(1)/\*
* <svg onload=eval(unescape(location))><title>\*/;alert(2);...
* "><img src=x onerror=confirm(1)>
* " onclick=alert(1)//<button ...
* ';alert(String.fromCharCode(88,83,83))//
* <SCRIPT>alert(String.fromCharCode(88,83,83)) </SCRIPT>
* <img src="http://i.imgur.com/P8mL8.jpg" onerror=eval(... )>
* "><BODY ONLOAD=alert(0x000123)>
* '%2Balert(0x000123)%2B' (URL-encoded variant)
* \"><sCriPt>alert(0x000123)</sCriPt>

**Đặc điểm chính của tập dữ liệu**

Chiến thuật tấn công phong phú: có cả payload trực tiếp chèn thẻ <script> lẫn kỹ thuật ẩn/obfuscate như String.fromCharCode, hex/decimal character references, %2B URL-encoded, hoặc chèn vào thuộc tính sự kiện (onerror/onload/onclick).

Nhiều dạng payload HTML/JS: chèn thẻ (script, img, svg), thuộc tính JS, inline JS sau dấu "> (break out of attribute), biểu thức javascript: và data: URIs, chú thích/đóng thẻ bất hợp pháp để phá ngữ cảnh HTML.

Obfuscation: dùng hàm tạo ký tự (String.fromCharCode), escape/encoding (URL-encoding, HTML entities), thay đổi case ký tự (sCriPt) để né signature tĩnh.

Ký tự đặc biệt nhiều: dấu ngoặc nhọn/nhọn <>, dấu nháy ' / ", dấu chấm phẩy ;, dấu ngoặc đơn (), ký tự % (encoded) — làm cho token hóa theo từ khó áp dụng, phù hợp trích xuất n-gram ký tự hoặc feature dựa trên ký tự đặc biệt.

Rủi ro false positive: nội dung HTML hợp lệ do người dùng nhập (ví dụ mô tả sản phẩm chứa HTML, tên file có chuỗi "script") có thể bị nhận dạng nhầm; cần chuyển đổi context và kết hợp nhiều đặc trưng để giảm FP.

* ShellCollection.txt: ShellCollection.txt chứa các payload/chuỗi mẫu liên quan tới các kỹ thuật tấn công thực thi lệnh trên hệ thống (Command Injection / Remote Code Execution / Shellshock / Directory Traversal, v.v.). Các dòng trong file biểu diễn nhiều dạng gọi lệnh shell hoặc biểu thức đánh lừa interpreter (ví dụ backticks, $(), eval, call tới system, Kernel.exec), đoạn payload khai thác lỗ hổng Shellshock, chuỗi chỉ đường tới file (file:///, .. traversal), và các ký tự/metacharacters đặc thù của shell (ví dụ ;, |, >, &, $(...), `...`). File dùng làm tập dữ liệu tấn công kiểu shell để huấn luyện mô hình phân loại payload độc hại.

**Một vài ví dụ điển hình (trích từ file):**

* () { 0; }; touch /tmp/blns.shellshock1.fail;
* () { \_; } >\_[$($())] { touch /tmp/blns.shellshock2.fail; }
* `touch /tmp/blns.fail`
* $(touch /tmp/blns.fail)
* @{[system "touch /tmp/blns.fail"]}
* eval("puts 'hello world'")
* `ls -al /`
* Kernel.exec("ls -al /")
* %x('ls -al /')
* File:///b'/../../../../file'
* \\..\\..\\..\\..\\file

**Đặc điểm chính của tập dữ liệu**

Sử dụng các metacharacters shell: backticks (`), $(), dấu chấm phẩy ;, pipe |, redirect >, dấu & — tất cả là chỉ dấu mạnh cho command injection.

Gọi API/những hàm thực thi hệ thống: system, eval, Kernel.exec, %x(...), exec, File:/// — biểu hiện nỗ lực chạy lệnh từ môi trường ứng dụng.

Các payload khai thác lỗ hổng cụ thể: dòng () { 0; }; ... là mẫu Shellshock (vulnerable bash function export exploit).

Directory traversal và path manipulation: ../../.., File:///..., escape backslashes \\..\\.. — thường dùng để truy cập file hệ thống.

Chuỗi chứa lệnh tạo file / thay đổi hệ thống: touch /tmp/..., ls -al /, rm,... (trong dataset có các ví dụ touch, ls).

Trích xuất payload từ nhiều ngữ cảnh: chuỗi có thể xuất hiện trong headers, cookie, form input, filename — nên detection cần xét context.

* non-maliciousCollection.txt: non-maliciousCollection.txt chứa các chuỗi hợp lệ (benign) đại diện cho các giá trị thường thấy trong ứng dụng web: số cổng, ID, tên thành phố, timestamp, mã sản phẩm, tên người, địa chỉ email/url được mã hóa, các token hay tham số hợp lệ khác. Mục đích của tập này là cung cấp ví dụ về input không độc hại để cân bằng tập huấn luyện và giúp mô hình học phân biệt giữa payload hợp lệ và payload tấn công.

**Một vài ví dụ điển hình (trích từ file):**

* 569993989 — có thể là một ID hoặc số điện thoại/ mã
* 46201 — mã bưu điện / số hiệu
* Indianapolis — tên thành phố (chuỗi chữ)
* 20354328 — mã/ticket/ID
* A8Cyj4uzrSgkGg4szKuHeI — token/khóa API/ID base62
* Primary — giá trị lựa chọn (select/enum)
* indiana%2Eedu — tên miền/địa chỉ có URL-encoded indiana.edu
* 100 — số lượng/giá trị số
* t — ký hiệu, boolean hoặc flag
* 1257138619112 — timestamp (ms)
* mc\_1094698\_601 — mã sản phẩm / mã phiên
* Catellier%2C+Michele+J. — tên người được URL-encoded (Catellier, Michele J.)
* 5 — số nhỏ (quantity/rating)

**Đặc điểm chính của tập dữ liệu**

Đa dạng kiểu dữ liệu hợp lệ: gồm số nguyên, chuỗi chữ, tên, mã, token, timestamp, URL-encoded values.

Nhiều giá trị mã hóa (encoded): có các chuỗi URL-encoded (%2E, +) đại diện cho dữ liệu được gửi qua query string/form.

Không có kí tự đặc biệt shell/SQL/XSS điển hình: hầu hết dòng không chứa metacharacters như ;, `, <script>, --, union select hoặc các từ khóa exec/system.

Chuỗi ngắn & cấu trúc có ý nghĩa: nhiều dòng có cấu trúc giống các trường form (city, postal code, product code,…).

* + 1. **Tiền xử lý dữ liệu**

Mục tiêu của bước tiền xử lý (preprocessing) là chuyển các tệp payload thô (SQL, XSS, Shell và non-malicious) thành một tập dữ liệu chuẩn, sạch, có nhãn và ở định dạng CSV dùng trực tiếp cho việc huấn luyện mô hình. Các yêu cầu chính:

* Loại bỏ dòng trống, BOM và kí tự gây hỏng định dạng CSV.
* Chuẩn hóa payload thành một dòng (single-line) để dễ đọc/ghi CSV.
* Gán nhãn is\_malicious cho mỗi payload (1 = malicious, 0 = benign).
* Loại bỏ bản sao (deduplicate) để tránh bias.
* Lưu tập kết quả ở dạng processed\_payloads.csv có header chuẩn.

Các file đầu vào đều ở dạng raw, và sau khi xử lý thì ta sẽ thu được 1 file tổng hợp tất cả các payload theo một định dạng thống nhất. Quá trình xử lý dữ liệu bao gồm các bước chính như sau:

* **Đọc và giữ nguyên định dạng gốc:** Các tệp chứa payload độc hại (SQL, XSS, Shell) được đọc theo từng dòng, đồng thời chương trình đảm bảo không loại bỏ các khoảng trắng hoặc ký tự đặc biệt nhằm giữ nguyên cấu trúc ban đầu của payload. Việc này giúp mô hình học máy có thể học được các đặc trưng thực tế của các chuỗi tấn công.
* **Chuẩn hóa nội dung:** Các ký tự không cần thiết như ký tự BOM (\ufeff) được loại bỏ. Ngoài ra, các ký tự xuống dòng (\n, \r) được thay thế bằng dấu cách để đảm bảo mỗi payload chỉ nằm trên một dòng duy nhất trong tệp CSV, tránh phá vỡ cấu trúc dữ liệu.
* **Gán nhãn và phân loại:**
  + Các chuỗi đến từ tệp SQLCollection.txt, XSSCollection.txt và ShellCollection.txt được gán nhãn is\_malicious = 1, cùng với loại tấn công tương ứng là SQL, XSS hoặc SHELL.
  + Tập dữ liệu hợp lệ non-maliciousCollection.txt được gán nhãn is\_malicious = 0 và loại LEGAL. Chương trình được thiết kế linh hoạt để có thể xử lý cả trường hợp tệp hợp lệ có hoặc không có định dạng CSV chuẩn.
* **Loại bỏ trùng lặp:** Sau khi hợp nhất toàn bộ các tập dữ liệu, chương trình thực hiện loại bỏ các payload trùng nhau bằng cách sử dụng hàm deduplicate\_preserve\_order(). Việc này giúp giảm nhiễu trong quá trình huấn luyện, đồng thời đảm bảo mỗi payload chỉ xuất hiện một lần duy nhất trong tập huấn luyện.
* **Xuất dữ liệu:** Kết quả cuối cùng được ghi vào tệp processed\_payloads.csv, bao gồm bốn cột chính:
* index: chỉ số thứ tự payload
* payload: nội dung chuỗi
* is\_malicious: nhãn độc hại (1: có, 0: không)
* injection\_type: loại tấn công hoặc hợp lệ (SQL, XSS, SHELL, LEGAL)
  + 1. **Huấn luyện mô hình**

**3.3.3.1. SQL Injection**

Để xây dựng hệ thống phát hiện tấn công SQL Injection (SQLi) tự động, nhóm đã tiến hành huấn luyện các mô hình học máy sử dụng tập dữ liệu được tổng hợp từ nhiều nguồn đáng tin cậy, bao gồm các payload tấn công và các chuỗi truy vấn hợp lệ (non-malicious) đã được xử lý ở các bước trên. Mục tiêu của quá trình huấn luyện là giúp mô hình có khả năng phân biệt giữa các mẫu đầu vào hợp lệ và các mẫu có chứa đặc trưng tấn công SQL Injection dựa trên ngữ cảnh, từ khóa, và cấu trúc cú pháp đặc trưng của truy vấn.

Đặc trưng của dữ liệu được biểu diễn bằng phương pháp Bag-of-Words thông qua CountVectorizer trong thư viện *scikit-learn*. Bộ vector hóa này sử dụng n-gram (với phạm vi từ 1 đến 3) nhằm nắm bắt được không chỉ từ khóa đơn lẻ mà còn các cụm từ hoặc chuỗi ký tự liên tiếp thường xuất hiện trong payload tấn công (ví dụ: "or 1=1", "' UNION SELECT"). Việc này giúp mô hình nhận diện được ngữ cảnh và mô hình cú pháp của các câu lệnh SQL độc hại.

Sau khi trích xuất đặc trưng, nhóm tiến hành huấn luyện và đánh giá hiệu năng của nhiều mô hình khác nhau để lựa chọn mô hình phù hợp nhất cho hệ thống WAF. Các mô hình được triển khai bao gồm:

* Multinomial Naive Bayes (NB) – mô hình cơ bản thường được sử dụng cho bài toán phân loại văn bản;
* Support Vector Machine (SVM) – cho phép xác định biên phân tách tối ưu giữa các lớp;
* Logistic Regression (LR) – mô hình tuyến tính có khả năng dự đoán xác suất;
* Decision Tree (DT) – giúp diễn giải trực quan quá trình phân loại dựa trên cây quyết định;
* Bagging và AdaBoost – hai kỹ thuật Ensemble nhằm tăng độ ổn định và giảm phương sai mô hình;
* Random Forest (RF) – mở rộng từ nhiều cây quyết định nhằm giảm hiện tượng overfitting;
* Stacking Classifier – mô hình tổ hợp nhiều bộ học (RF, SVM, LR) nhằm tận dụng ưu điểm của từng thuật toán.

Sau khi huấn luyện, các mô hình được đánh giá thông qua các chỉ số như độ chính xác (Accuracy) và báo cáo phân loại (Classification Report), bao gồm độ chính xác, độ bao phủ và điểm F1-score cho từng lớp. Kết quả cho thấy mô hình Multinomial Naive Bayes đạt hiệu năng ổn định và tốc độ huấn luyện nhanh, do đó được lựa chọn làm mô hình chính cho hệ thống Web Application Firewall (WAF).

Cuối cùng, mô hình Naive Bayes cùng với bộ CountVectorizer được lưu trữ dưới dạng file nhị phân (nb\_classifier.pkl, count\_vectorizer.pkl) trong thư mục saved\_models. Các tệp này sẽ được nạp trực tiếp vào thành phần WAF trong quá trình chạy thực tế, cho phép hệ thống phân tích và phát hiện các truy vấn SQL độc hại trong thời gian thực mà không cần tái huấn luyện mô hình.

**3.3.3.2. XSS**

* + 1. **Kết quả huấn luyện**

## **Kiểm thử**

* + 1. **Truy cập hợp lệ**
    2. **Truy cập vi phạm**

# **Chương 4. Kết luận**

## **Mục tiêu đạt được**

## **Hạn chế**

**TÀI LIỆU THAM KHẢO**

[1] Cisco, "What Is a Firewall?", Cisco, [Online]. Available: https://www.cisco.com/site/us/en/learn/topics/security/what-is-a-firewall.html. [Accessed: Oct. 22, 2025].

[2] Kobalt.io, “The Role of Firewalls in Cybersecurity,” *Kobalt.io*, [Online]. Available: [https://kobalt.io/role-of-firewalls/](https://kobalt.io/role-of-firewalls/?utm_source=chatgpt.com). [Accessed: Oct. 22, 2025].

[3] Cisco, “What Is a Web Application Firewall (WAF)?”, *Cisco*, [Online]. Available: [https://www.cisco.com/site/us/en/learn/topics/security/what-is-web-application-firewall-waf.html](https://www.cisco.com/site/us/en/learn/topics/security/what-is-web-application-firewall-waf.html?utm_source=chatgpt.com). [Accessed: Oct. 22, 2025].

[4] Fortinet, “WAF vs. Firewall: Web Application & Network Firewalls,” *Fortinet CyberGlossary*, [Online]. Available: <https://www.fortinet.com/resources/cyberglossary/waf-vs-firewall>. [Accessed: Oct. 22, 2025]. ([fortinet.com](https://www.fortinet.com/resources/cyberglossary/waf-vs-firewall?utm_source=chatgpt.com))

[5] Penta Security Inc., “Signature-Based vs. Rule-Based WAFs: A Detailed Comparison,” *Penta Security Blog*, 14 Aug. 2020. [Online]. Available: [https://www.pentasecurity.com/blog/signature-based-vs-rule-based-wafs/](https://www.pentasecurity.com/blog/signature-based-vs-rule-based-wafs/?utm_source=chatgpt.com) [Accessed: Oct. 22, 2025].

[6] ModShield SB, “Demystifying WAF Rules: Signature-based vs. Anomaly Detection,” *ModShield SB Blog*, 23 May 2024. [Online]. Available: [https://www.modshieldsb.com/demystifying-waf-rules-signature-based-vs-anomaly-detection/](https://www.modshieldsb.com/demystifying-waf-rules-signature-based-vs-anomaly-detection/?utm_source=chatgpt.com). [Accessed: Oct. 22, 2025].

[7] P. V. Hau and D. T. T. Hien, "Enhancing Web Application Security: A Deep Learning and NLP-based Approach for Accurate Attack Detection," \*Journal of Science and Technology on Information security\*, no. 3.CS(20), pp. 77–87, 2023.

[8] Fortinet, “WAF Architecture,” *Fortinet CyberGlossary*, [Online]. Available: <https://www.fortinet.com/resources/cyberglossary/waf-architecture>. [Accessed: Oct. 22, 2025].

[9] Radware, “WAF Architecture: Components & 8 Key Considerations,” *Radware CyberPedia*, [Online]. Available: [https://www.radware.com/cyberpedia/application-security/waf-architecture/](https://www.radware.com/cyberpedia/application-security/waf-architecture/?utm_source=chatgpt.com). [Accessed: Oct. 22, 2025].

[10] Source Defense, “Limitations of WAF,” *Source Defense Glossary*, [Online]. Available: [https://sourcedefense.com/glossary/limitations-of-waf/](https://sourcedefense.com/glossary/limitations-of-waf/?utm_source=chatgpt.com). [Accessed: Oct. 22, 2025].