# Nghiên cứu và Phát triển Hệ thống Phòng thủ API Chủ động: Kiến trúc Purple Team Tự động hóa Tích hợp Trí tuệ Nhân tạo Phân loại Mã độc

## Tóm tắt Điều hành

Trong bối cảnh an ninh mạng hiện đại, Application Programming Interfaces (API) đã trở thành huyết mạch của chuyển đổi số, kết nối các dịch vụ vi mô, ứng dụng di động và hệ thống đám mây. Tuy nhiên, sự gia tăng về số lượng và độ phức tạp của API đã mở rộng bề mặt tấn công, khiến các phương pháp bảo mật truyền thống dựa trên chữ ký (signature-based) trở nên kém hiệu quả trước các mối đe dọa tinh vi và biến đổi không ngừng.1 Báo cáo này trình bày một đề án tốt nghiệp chuyên sâu dành cho sinh viên chuyên ngành an toàn thông tin, tập trung vào việc xây dựng một hệ sinh thái bảo mật khép kín theo mô hình **Purple Team tự động hóa**. Dự án đề xuất phát triển ba thành phần cốt lõi tương tác chặt chẽ: (1) Một môi trường giả lập chứa các lỗ hổng API được thiết kế có chủ đích (Vulnerable Target), (2) Một hệ thống tấn công tự động (Automated Attack Simulator) thực hiện vai trò Red Team để kiểm thử và tạo dữ liệu huấn luyện, và (3) Một cơ chế phòng thủ chủ động (AI-Driven WAF) đóng vai trò Blue Team, sử dụng học máy (Machine Learning) để phát hiện, phân loại và ngăn chặn các cuộc tấn công trong thời gian thực.

Nghiên cứu này không chỉ giải quyết bài toán phát hiện xâm nhập mà còn tập trung vào khả năng **nhận diện định danh loại hình tấn công** (Attack Classification) thông qua các thuật toán học máy giám sát đa lớp (Multi-class Classification), từ đó cung cấp phản ứng phòng thủ chính xác và giảm thiểu tỷ lệ báo động giả. Bằng cách kết hợp quy trình tấn công giả lập và phòng thủ tự động, dự án tạo ra một vòng lặp phản hồi liên tục (feedback loop), cho phép hệ thống "tự học" và thích ứng với các mẫu tấn công mới, mô phỏng quy trình vận hành của một trung tâm điều hành an ninh (SOC) hiện đại nhưng được thu nhỏ và tự động hóa.3

## 1. Giới thiệu và Phạm vi Nghiên cứu

### 1.1 Bối cảnh An ninh API và Sự dịch chuyển sang DevSecOps

Sự phát triển của kiến trúc Microservices và các ứng dụng Cloud-native đã đưa API trở thành mục tiêu hàng đầu của tội phạm mạng. Theo báo cáo của Akamai và Salt Security, lưu lượng tấn công vào API tăng trưởng nhanh hơn nhiều so với lưu lượng API hợp pháp. Các lỗ hổng như Broken Object Level Authorization (BOLA), Excessive Data Exposure, và Injection (SQLi, XSS) đang trở thành những vector tấn công phổ biến nhất.2

Trong quy trình phát triển phần mềm truyền thống, kiểm thử bảo mật thường được thực hiện ở giai đoạn cuối (Penetration Testing), dẫn đến việc phát hiện lỗi muộn và chi phí khắc phục cao. Mô hình DevSecOps yêu cầu bảo mật phải được tích hợp vào mọi giai đoạn ("Shift Left"). Tuy nhiên, việc thực hiện kiểm thử xâm nhập (Pentest) thủ công liên tục là bất khả thi về mặt nhân lực và thời gian. Do đó, nhu cầu về một giải pháp **Purple Teaming tự động hóa**—nơi các công cụ tấn công và phòng thủ hoạt động song song và liên tục—trở nên cấp thiết để đảm bảo an ninh cho API ngay từ giai đoạn phát triển.3

### 1.2 Mục tiêu và Ý nghĩa của Đồ án

Đồ án này được thiết kế nhằm mục đích cung cấp cho sinh viên cơ hội tiếp cận toàn diện với vòng đời bảo mật API. Mục tiêu tối thượng là xây dựng một hệ thống có khả năng tự động hóa quy trình kiểm thử (Red Teaming) và tự động hóa quy trình phòng thủ (Blue Teaming) dựa trên dữ liệu.

Các mục tiêu cụ thể bao gồm:

1. **Mô phỏng Môi trường Thực tế:** Xây dựng một ứng dụng API hoàn chỉnh với các lỗi bảo mật điển hình để làm đối tượng nghiên cứu, thay vì sử dụng các bộ dữ liệu tĩnh thiếu tính thực tế.
2. **Tự động hóa Tấn công (Attack Automation):** Phát triển module có khả năng tự động dò quét, khai thác lỗ hổng và sinh ra các mẫu tấn công đa dạng để làm giàu dữ liệu huấn luyện.6
3. **Ứng dụng Trí tuệ Nhân tạo trong Phòng thủ:** Huấn luyện các mô hình học máy để phân tích cú pháp và ngữ nghĩa của HTTP Request, từ đó nhận diện chính xác loại mã độc (SQL Injection, XSS, Command Injection, v.v.) thay vì chỉ chặn dựa trên từ khóa đơn giản.7
4. **Phòng chống Chủ động (Active Defense):** Tích hợp mô hình AI vào một Middleware (WAF) để chặn các request độc hại ngay lập tức trước khi chúng chạm tới logic xử lý của ứng dụng.

### 1.3 Cấu trúc Tổng thể của Dự án

Dự án được chia thành bốn phân hệ chính, hoạt động trong một quy trình khép kín:

* **Phân hệ 1: Target System (Mục tiêu):** Một Web API được code có chủ đích để chứa lỗ hổng.
* **Phân hệ 2: Attack Engine (Tấn công):** Bộ công cụ automation scripts thực hiện fuzzing và injection.
* **Phân hệ 3: AI Brain (Trí tuệ):** Pipeline xử lý dữ liệu, huấn luyện mô hình và API dự đoán.
* **Phân hệ 4: Defense Shield (Phòng thủ):** Middleware tích hợp trong Target System để chặn tấn công dựa trên chỉ đạo của AI Brain.

## 2. Cơ sở Lý thuyết và Tổng quan Công nghệ

### 2.1 Các Lỗ hổng API Nghiêm trọng (OWASP API Security Top 10)

Để xây dựng một hệ thống phòng thủ hiệu quả, sinh viên cần nắm vững bản chất của các lỗ hổng mà hệ thống sẽ đối mặt. Đồ án sẽ tập trung vào việc mô phỏng và ngăn chặn các lỗ hổng sau, dựa trên danh sách OWASP API Security Top 10 2023 2:

| **Lỗ hổng** | **Mô tả Cơ chế** | **Nguy cơ** |
| --- | --- | --- |
| **API1:2023 Broken Object Level Authorization (BOLA)** | API không kiểm tra quyền truy cập của người dùng đối với đối tượng cụ thể (ví dụ: ID trong URL). Kẻ tấn công thay đổi ID để truy cập dữ liệu của người khác. | Lộ lọt dữ liệu cá nhân, chiếm quyền tài khoản. |
| **API2:2023 Broken Authentication** | Cơ chế xác thực yếu, cho phép credential stuffing, brute-force, hoặc giả mạo token (JWT không ký hoặc ký yếu). | Truy cập trái phép, mạo danh người dùng. |
| **API3:2023 Broken Object Property Level Authorization** | Cho phép người dùng xem hoặc chỉnh sửa các trường dữ liệu nhạy cảm mà họ không có quyền (Mass Assignment, Excessive Data Exposure). | Thăng đặc quyền (Privilege Escalation), lộ dữ liệu. |
| **API8:2023 Security Misconfiguration** | Cấu hình bảo mật lỏng lẻo, thông báo lỗi chứa stack trace, không bật các header bảo mật, hoặc cho phép các phương thức HTTP không cần thiết. | Cung cấp thông tin cho kẻ tấn công để khai thác sâu hơn. |
| **Injection (SQLi, NoSQLi, Command Injection)** | Dữ liệu đầu vào không được lọc (sanitize) được chuyển trực tiếp xuống trình biên dịch (interpreter) của cơ sở dữ liệu hoặc hệ điều hành.9 | Mất dữ liệu, chiếm quyền điều khiển server (RCE). |

### 2.2 Mô hình Purple Teaming và Automation

Purple Teaming là sự hợp tác giữa Red Team (tấn công) và Blue Team (phòng thủ) nhằm tối ưu hóa năng lực an ninh của tổ chức. Trong mô hình truyền thống, Red Team tấn công và sau đó báo cáo; Blue Team khắc phục. Quá trình này thường rời rạc và chậm chạp.

Trong đồ án này, khái niệm "Purple Team" được hiện thực hóa thông qua Automation (Tự động hóa).3

* **Automated Red Teaming:** Script tấn công chạy liên tục theo lịch trình hoặc sự kiện (CI/CD trigger), mô phỏng các hành vi của kẻ tấn công thực tế (Tactics, Techniques, and Procedures - TTPs).
* **Feedback Loop:** Dữ liệu từ các cuộc tấn công (thành công hay thất bại) được ghi nhận và nạp ngay lập tức vào hệ thống AI để "dạy" cho Blue Team ảo cách nhận diện. Đây là cốt lõi của sự "kết hợp" mà yêu cầu đồ án đặt ra.

### 2.3 Trí tuệ Nhân tạo trong Phân loại Mã độc

Việc sử dụng AI để phát hiện tấn công web thường dựa trên bài toán **Phân loại Văn bản (Text Classification)** hoặc **Phát hiện Bất thường (Anomaly Detection)**.

* **Feature Extraction (Trích xuất đặc trưng):** HTTP request là dữ liệu phi cấu trúc. Để AI hiểu được, cần chuyển đổi chúng thành vector số. Các kỹ thuật phổ biến bao gồm TF-IDF (Term Frequency-Inverse Document Frequency) để đánh trọng số cho các từ khóa nguy hiểm (như SELECT, UNION, SCRIPT, DROP) và N-grams để bắt các chuỗi ký tự liền kề đặc trưng của mã độc.7
* **Mô hình Học máy:**
  + *Random Forest / Decision Trees:* Hiệu quả cao trong việc phân loại dựa trên các luật (rules) học được từ dữ liệu, dễ giải thích (explainable AI).12
  + *Support Vector Machines (SVM):* Tốt cho việc phân chia ranh giới giữa dữ liệu sạch và dữ liệu tấn công trong không gian nhiều chiều.8
  + *Deep Learning (CNN/LSTM):* Có thể sử dụng để học các mẫu phức tạp hơn trong chuỗi URL hoặc Body dài, nhưng yêu cầu tài nguyên tính toán lớn hơn.14

## 3. Thiết kế Kiến trúc Hệ thống (System Architecture)

Hệ thống được thiết kế theo kiến trúc Microservices, cho phép các thành phần phát triển và triển khai độc lập. Dưới đây là sơ đồ kiến trúc logic và chức năng chi tiết của từng module.

### 3.1 Sơ đồ Tổng quan

Hệ thống bao gồm 4 khối chính giao tiếp với nhau qua HTTP và Shared Database/Logs:

1. **The Vulnerable Target (API Mục tiêu):** Nơi chứa các endpoint bị lỗi.
2. **The Automated Attacker (Máy Tấn công):** Nơi sinh ra lưu lượng sạch và độc hại.
3. **The Intelligence Core (Bộ não AI):** Nơi huấn luyện và cung cấp API dự đoán.
4. **The Active Defense Middleware (Lớp Phòng thủ):** Cổng bảo vệ gắn trực tiếp vào Target.

### 3.2 Chi tiết Thành phần 1: The Vulnerable Target (Website Giả lập)

Đây là "bệnh nhân" để sinh viên thực hành. Thay vì dùng các framework có sẵn như DVWA hay Juice Shop, sinh viên nên tự xây dựng một API đơn giản để hiểu sâu về cách code gây ra lỗi.

* **Công nghệ:** Python Flask hoặc FastAPI (đơn giản, dễ demo, hỗ trợ mạnh về AI sau này). Cơ sở dữ liệu SQLite hoặc PostgreSQL.
* **Chức năng Nghiệp vụ:** Một ứng dụng quản lý sách hoặc sinh viên đơn giản.
  + POST /auth/login: Đăng nhập.
  + GET /books: Tìm kiếm sách.
  + POST /books/{id}/comment: Bình luận.
  + GET /user/{id}/profile: Xem thông tin cá nhân.
* **Kịch bản Lỗ hổng (Vulnerability Injection):**
  + *SQL Injection:* Tại endpoint /books, tham số tìm kiếm q được nối chuỗi trực tiếp vào câu lệnh SQL: f"SELECT \* FROM books WHERE title LIKE '%{q}%'".15
  + *Reflected XSS:* Tại endpoint /books, trả về tham số q trong thông báo lỗi hoặc kết quả tìm kiếm mà không qua hàm escape().9
  + *BOLA (IDOR):* Tại endpoint /user/{id}/profile, backend lấy thông tin theo id trên URL mà không kiểm tra xem id đó có trùng với user đang đăng nhập (từ token) hay không.2
  + *Security Misconfiguration:* Bật chế độ Debug (DEBUG=True trong Flask) để lộ stack trace khi gặp lỗi 500.

### 3.3 Chi tiết Thành phần 2: The Automated Attacker (Công cụ Tự động hóa)

Đây là thành phần "Red Team ảo". Nhiệm vụ của nó là bắn phá Target để kiểm tra độ bền vững và quan trọng hơn là **tạo ra bộ dữ liệu (dataset)** cho AI học.

* **Công nghệ:** Python, thư viện requests, faker (tạo dữ liệu giả), pytest.
* **Cơ chế Hoạt động:**
  + **Traffic Generator (Benign):** Sử dụng faker để tạo ra hàng nghìn request hợp lệ: đăng nhập đúng, tìm kiếm từ khóa bình thường, xem profile của chính mình. Gán nhãn Label = 0 (Benign).
  + **Attack Simulator (Malicious):**
    - *Payload Injection:* Tích hợp danh sách các payload tấn công nổi tiếng (Fuzzing Lists) như SecLists hoặc PayloadsAllTheThings. Ví dụ: ' OR 1=1 -- (SQLi), <script>alert(1)</script> (XSS), ../../etc/passwd (Path Traversal).16
    - *Logic Exploitation:* Script tự động thay đổi ID trong URL để thử tấn công BOLA.
    - Gán nhãn Label = 1 (SQLi), Label = 2 (XSS), Label = 3 (Path Traversal), v.v.
* **Lưu trữ:** Mọi request gửi đi (URL, Headers, Body, Method) và nhãn tương ứng được ghi vào một file CSV (training\_data.csv) hoặc Database Log để làm nguyên liệu cho AI.

### 3.4 Chi tiết Thành phần 3: The Intelligence Core (Bộ não AI)

Phân hệ này chịu trách nhiệm "học" từ dữ liệu mà Attacker tạo ra và cung cấp khả năng nhận diện.

* **Công nghệ:** Python scikit-learn, pandas, joblib.
* **Quy trình Xử lý (Pipeline):**
  1. **Data Ingestion:** Đọc dữ liệu từ CSV/Database.
  2. **Preprocessing:** Làm sạch dữ liệu, giải mã URL encoding (ví dụ: %20 -> space), chuyển về chữ thường.
  3. **Feature Engineering (Trích xuất đặc trưng):**
     + Sử dụng **TF-IDF Vectorizer** để chuyển đổi payload (body/url) thành vector số. Cấu hình N-gram (ví dụ: 1-3 grams) để bắt các cụm từ như union select hay script src.8
     + Trích xuất các đặc trưng thống kê: Độ dài request, số lượng ký tự đặc biệt, entropy.11
  4. **Model Training:** Huấn luyện mô hình phân loại đa lớp (Multi-class Classifier).
     + Input: Vector đặc trưng của request.
     + Output: Loại tấn công (Normal, SQLi, XSS, v.v.).
     + Thuật toán đề xuất: **Random Forest** (độ chính xác cao, ít bị overfitting) hoặc **SVM**.12
  5. **Model Serialization:** Lưu mô hình đã train ra file .pkl để WAF có thể load nhanh mà không cần train lại mỗi lần khởi động.20

### 3.5 Chi tiết Thành phần 4: The Active Defense Middleware (WAF)

Đây là lớp "phòng chống" được nhúng trực tiếp vào ứng dụng mục tiêu hoặc đứng trước nó như một Reverse Proxy.

* **Công nghệ:** Flask Middleware (@app.before\_request) hoặc một microservice độc lập.
* **Logic Phòng thủ:**
  1. Bắt chặn (Intercept) mọi HTTP Request đến server.
  2. Trích xuất dữ liệu (URL, Body).
  3. Tiền xử lý dữ liệu giống hệt như lúc train AI.
  4. Gọi Model AI để dự đoán.
  5. **Hành động (Enforcement):**
     + Nếu AI dự đoán là Normal: Cho phép request đi qua (pass).
     + Nếu AI dự đoán là Malicious (ví dụ: SQLi):
       - Chặn request (trả về HTTP 403 Forbidden).
       - Ghi log cảnh báo: "Phát hiện tấn công SQL Injection từ IP X".
       - Trả về thông báo cho Attacker (tùy chọn): "Request blocked by AI Shield".

## 4. Phương pháp Triển khai và Chi tiết Kỹ thuật

Phần này hướng dẫn chi tiết cách sinh viên thực hiện đồ án, từ việc setup môi trường đến viết code cụ thể cho từng module.

### 4.1 Giai đoạn 1: Xây dựng Nền tảng (Target & Attacker)

Bước 1: Phát triển Vulnerable API (Target)

Sinh viên cần viết code Python Flask. Dưới đây là ví dụ minh họa logic lỗi SQL Injection để nhúng vào đồ án:

Python

# target\_app.py  
from flask import Flask, request  
import sqlite3  
  
app = Flask(\_\_name\_\_)  
  
@app.route('/api/search', methods=)  
def search\_product():  
 query = request.args.get('q')  
 conn = sqlite3.connect('database.db')  
 cursor = conn.cursor()  
 # LỖI BẢO MẬT: Nối chuỗi trực tiếp tạo điều kiện cho SQL Injection  
 sql\_command = f"SELECT \* FROM products WHERE name LIKE '%{query}%'"  
 try:  
 cursor.execute(sql\_command)  
 results = cursor.fetchall()  
 return {"data": results}, 200  
 except Exception as e:  
 return {"error": str(e)}, 500 # LỖI BẢO MẬT: Lộ thông tin lỗi hệ thống

*Phân tích:* Đoạn code trên vi phạm nguyên tắc sử dụng Parameterized Query và để lộ thông tin lỗi (Information Leakage), tạo điều kiện cho module tấn công khai thác.10

Bước 2: Xây dựng Bộ sinh dữ liệu và Tấn công (Attacker)

Sử dụng thư viện faker và requests. Script này cần chạy độc lập.

Python

# attacker\_bot.py  
import requests  
import random  
import csv  
from faker import Faker  
  
fake = Faker()  
TARGET\_URL = "http://localhost:5000/api/search"  
  
# Danh sách payload tấn công mẫu  
sqli\_payloads =  
xss\_payloads = ["<script>alert(1)</script>", "<img src=x onerror=alert(1)>"]  
  
def generate\_traffic():  
 with open('dataset.csv', 'a', newline='', encoding='utf-8') as f:  
 writer = csv.writer(f)  
   
 # 1. Sinh traffic sạch  
 for \_ in range(100):  
 keyword = fake.word()  
 requests.get(TARGET\_URL, params={'q': keyword})  
 writer.writerow([keyword, 'Normal']) # Gán nhãn 0  
  
 # 2. Sinh traffic tấn công SQLi  
 for payload in sqli\_payloads:  
 requests.get(TARGET\_URL, params={'q': payload})  
 writer.writerow() # Gán nhãn 1  
  
 # 3. Sinh traffic tấn công XSS  
 for payload in xss\_payloads:  
 requests.get(TARGET\_URL, params={'q': payload})  
 writer.writerow() # Gán nhãn 2  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 generate\_traffic()

*Lưu ý:* Sinh viên cần mở rộng danh sách payload bằng cách tải về từ các nguồn GitHub như SecLists để dataset đủ lớn (khuyến nghị > 10,000 mẫu).16

### 4.2 Giai đoạn 2: Phát triển Trí tuệ Nhân tạo (The Brain)

Đây là phần phức tạp nhất, yêu cầu kiến thức về Data Science.

Bước 1: Tiền xử lý và Vector hóa (Preprocessing)

Sử dụng TfidfVectorizer của scikit-learn để chuyển đổi chuỗi text thành số. Cần cấu hình analyzer='char' hoặc analyzer='word' tùy thuộc vào chiến lược. Với SQLi/XSS, phân tích cấp ký tự (char-level n-grams) thường hiệu quả hơn vì nó bắt được các mẫu cú pháp nhỏ như OR, --, <s.8

**Bước 2: Huấn luyện Mô hình Phân loại (Training)**

Python

# train\_model.py  
import pandas as pd  
from sklearn.feature\_extraction.text import TfidfVectorizer  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import classification\_report  
import joblib  
  
# 1. Load dữ liệu  
df = pd.read\_csv('dataset.csv', names=['payload', 'label'])  
  
# 2. Vector hóa dữ liệu  
vectorizer = TfidfVectorizer(min\_df=0.0, analyzer="char", sublinear\_tf=True, ngram\_range=(1,3))  
X = vectorizer.fit\_transform(df['payload'])  
y = df['label']  
  
# 3. Chia tập train/test  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# 4. Huấn luyện Random Forest  
model = RandomForestClassifier(n\_estimators=100)  
model.fit(X\_train, y\_train)  
  
# 5. Đánh giá  
y\_pred = model.predict(X\_test)  
print(classification\_report(y\_test, y\_pred))  
  
# 6. Lưu mô hình để dùng cho WAF  
joblib.dump(model, 'waf\_model.pkl')  
joblib.dump(vectorizer, 'waf\_vectorizer.pkl')

*Phân tích:* Sử dụng RandomForest giúp mô hình có khả năng tổng quát hóa tốt. Việc lưu model ra file .pkl là bắt buộc để tích hợp vào ứng dụng.13

### 4.3 Giai đoạn 3: Tích hợp và Phòng thủ (Active Defense)

Sử dụng Middleware trong Flask để chặn request dựa trên kết quả từ AI.

Python

# waf\_middleware.py  
from flask import request, abort, jsonify  
import joblib  
import numpy as np  
  
# Load bộ não AI đã được huấn luyện  
model = joblib.load('waf\_model.pkl')  
vectorizer = joblib.load('waf\_vectorizer.pkl')  
  
def ai\_firewall():  
 # Lấy dữ liệu từ request  
 payload\_to\_check = ""  
 if request.method == 'GET':  
 payload\_to\_check = request.args.get('q', '') # Lấy tham số q  
 elif request.method == 'POST':  
 payload\_to\_check = request.data.decode('utf-8')  
  
 if payload\_to\_check:  
 # Chuyển đổi dữ liệu sang vector  
 vector\_input = vectorizer.transform([payload\_to\_check])  
 # AI dự đoán  
 prediction = model.predict(vector\_input)  
   
 # Nếu không phải là 'Normal', chặn ngay lập tức  
 if prediction!= 'Normal':  
 print(f" Blocked attack type: {prediction} from IP: {request.remote\_addr}")  
 response = jsonify({  
 "error": "Request Blocked",  
 "reason": f"AI Shield detected malicious pattern: {prediction}"  
 })  
 response.status\_code = 403  
 return response  
  
# Tích hợp vào app chính  
# app.before\_request(ai\_firewall)

Middleware này hoạt động như một chốt chặn. Nếu AI nói "đây là SQL Injection", hàm ai\_firewall sẽ trả về lỗi 403 và ngắt request ngay lập tức, bảo vệ hàm search\_product bên dưới khỏi bị khai thác, dù code của search\_product vẫn còn lỗ hổng.21

## 5. Đánh giá và Kiểm thử Hệ thống

### 5.1 Chỉ số Đánh giá Hiệu quả AI

Để chứng minh đồ án thành công, sinh viên cần đưa ra các bảng số liệu cụ thể. Các chỉ số quan trọng bao gồm:

* **Precision (Độ chính xác):** Tỷ lệ số lần AI chặn đúng tấn công trên tổng số lần chặn. Precision thấp đồng nghĩa với việc chặn nhầm người dùng thật (False Positive) - điều tối kỵ trong WAF.
* **Recall (Độ phủ):** Tỷ lệ số cuộc tấn công bị phát hiện trên tổng số cuộc tấn công thực tế. Recall thấp nghĩa là bỏ lọt tội phạm (False Negative).
* **F1-Score:** Trung bình điều hòa của Precision và Recall.

| **Loại Tấn công** | **Precision** | **Recall** | **F1-Score** |
| --- | --- | --- | --- |
| Normal | 0.99 | 0.98 | 0.98 |
| SQL Injection | 0.95 | 0.92 | 0.93 |
| XSS | 0.94 | 0.90 | 0.92 |
| **Tổng thể** | **0.96** | **0.93** | **0.94** |
| *(Bảng minh họa kết quả mong đợi)* |  |  |  |

### 5.2 Kiểm thử Hiệu năng (Performance Overhead)

Đo lường độ trễ (Latency) mà WAF thêm vào hệ thống.

* Thời gian phản hồi khi không có WAF: ~50ms.
* Thời gian phản hồi khi có AI WAF: ~70ms.
* *Kết luận:* AI thêm vào khoảng 20ms độ trễ để tính toán vector và dự đoán. Đây là mức chấp nhận được cho hầu hết ứng dụng web.23

### 5.3 Kịch bản Demo Hội đồng

Sinh viên cần chuẩn bị kịch bản demo trực quan:

1. **Chưa bật WAF:** Chạy script tấn công SQL Injection -> Show database bị dump hoặc đăng nhập thành công với admin' --.
2. **Bật AI WAF:** Chạy lại script tấn công cũ -> Nhận lỗi 403 Forbidden.
3. **Nhận diện:** Show log trên Dashboard thấy dòng chữ: "Detected: SQL Injection".
4. **Tấn công mới (Zero-day simulation):** Thử biến đổi payload SQLi một chút (ví dụ dùng /\*\*/ thay cho dấu cách). Nếu AI được train tốt với N-grams, nó vẫn sẽ nhận diện được và chặn.

## 6. Thảo luận và Hướng phát triển

### 6.1 Lợi ích của Giải pháp

Giải pháp này vượt trội so với WAF truyền thống (dựa trên Regex) ở chỗ nó có khả năng "học" và khái quát hóa. Nếu Regex chỉ chặn chính xác chuỗi UNION SELECT, thì AI có thể chặn UnIoN/\*\*/SeLeCt dựa trên sự tương đồng về vector đặc trưng mà không cần viết luật mới. Hơn nữa, việc tự động hóa quy trình tấn công giúp sinh ra dữ liệu liên tục, làm cho mô hình ngày càng thông minh hơn theo thời gian (Purple Team Loop).

### 6.2 Hạn chế và Tấn công vào AI (Adversarial Attacks)

Mô hình AI có thể bị đánh lừa bởi các kỹ thuật **Adversarial Machine Learning**. Ví dụ, kẻ tấn công có thể chèn thêm các từ khóa "lành tính" (như hello, world, contact) vào payload độc hại để làm loãng vector đặc trưng, khiến AI nhầm tưởng đó là request bình thường. Một hướng phát triển tương lai là áp dụng kỹ thuật **Adversarial Training** - chủ động sinh ra các mẫu gây nhiễu này và dạy AI cách nhận biết chúng.24

### 6.3 Hướng mở rộng Đồ án

* **Real-time Learning:** Triển khai cơ chế Online Learning để model cập nhật trọng số ngay lập tức sau mỗi lần phát hiện sai (cần người quản trị gán nhãn lại).
* **Dashboard trực quan:** Xây dựng giao diện Grafana/Elasticsearch (ELK Stack) để vẽ biểu đồ tấn công theo thời gian thực.
* **Containerization:** Đóng gói toàn bộ 4 module vào Docker Compose để triển khai "1-click deployment".

## 7. Kết luận

Đồ án "Hệ thống Phòng thủ API Chủ động Dựa trên Mô hình Purple Team Tự động hóa và Trí tuệ Nhân tạo" là một đề tài mang tính thời sự, thực tiễn và hàm lượng công nghệ cao, rất phù hợp cho sinh viên an ninh mạng. Dự án không chỉ dừng lại ở việc phát hiện lỗ hổng mà còn tiến tới việc tự động hóa và phòng thủ chủ động - xu hướng tất yếu của ngành DevSecOps. Bằng việc kết hợp kiến thức về lập trình an toàn (Secure Coding), kiểm thử xâm nhập (Pentest), tự động hóa (Automation) và khoa học dữ liệu (Data Science), sinh viên sẽ chứng minh được năng lực toàn diện của một kỹ sư an ninh mạng hiện đại.

Hệ thống đề xuất cung cấp một khung làm việc (framework) linh hoạt, có thể mở rộng để phát hiện nhiều loại tấn công khác nhau, đóng góp một giải pháp hiệu quả cho bài toán bảo mật API đang ngày càng nhức nhối trong kỷ nguyên số.

Ghi chú: Báo cáo này tổng hợp các phương pháp và kỹ thuật từ các tài liệu nghiên cứu uy tín về Purple Teaming 3, An ninh API 2, Học máy trong An ninh mạng 7, và các kỹ thuật lập trình phòng thủ.9

#### Nguồn trích dẫn

1. API Security: Purple Teaming Exercises - ThreatX, truy cập vào tháng 11 29, 2025, <https://www.threatx.com/blog/api-security-purple-teaming-exercises/>
2. OWASP Top 10 API Security Risks. ( 2024 Updated) | by Robert Hackett | Medium, truy cập vào tháng 11 29, 2025, <https://medium.com/@robert-hackett/owasp-api-top-10-27455f3241f0>
3. Purple Team Exercise Framework: A Complete Guide to Collaborative Security Testing, truy cập vào tháng 11 29, 2025, <https://www.appsecure.security/blog/purple-team-exercise-framework>
4. OWASP API Security Top 10, truy cập vào tháng 11 29, 2025, <https://owasp.org/API-Security/>
5. What is a Purple Team - Wallarm, truy cập vào tháng 11 29, 2025, <https://www.wallarm.com/what/what-is-a-purple-team>
6. OpenAEV | Adversarial Exposure Validation Platform by Filigran, truy cập vào tháng 11 29, 2025, <https://filigran.io/platforms/openaev/>
7. Detecting SQL Injection Attacks using Machine Learning - CEUR-WS.org, truy cập vào tháng 11 29, 2025, <https://ceur-ws.org/Vol-3652/paper4.pdf>
8. vladan-stojnic/ML-based-WAF: Simple machine learning based web application firewall (WAF) created in python - GitHub, truy cập vào tháng 11 29, 2025, <https://github.com/vladan-stojnic/ML-based-WAF>
9. Cross-Site Scripting (XSS) Vulnerability Guide - Invicti, truy cập vào tháng 11 29, 2025, <https://www.invicti.com/learn/cross-site-scripting-xss>
10. Preventing SQL Injection Attacks With Python, truy cập vào tháng 11 29, 2025, <https://realpython.com/prevent-python-sql-injection/>
11. Analyzing HTTP requests for web intrusion detection - Digital ..., truy cập vào tháng 11 29, 2025, [https://digitalcommons.kennesaw.edu/cgi/viewcontent.cgi?referer=&httpsredir=1&article=1053&context=ccerp](https://digitalcommons.kennesaw.edu/cgi/viewcontent.cgi?referer&httpsredir=1&article=1053&context=ccerp)
12. prax-1/WebAppFirewall-with-ML - GitHub, truy cập vào tháng 11 29, 2025, <https://github.com/prax-1/WebAppFirewall-with-ML>
13. Serving your SciKit Learn Model as a Prediction API - Mike Polinowski, truy cập vào tháng 11 29, 2025, <https://mpolinowski.github.io/docs/IoT-and-Machine-Learning/AIOps/2023-06-17-scikit-learn-model-deployment/2023-06-17/>
14. Analyzing HTTP requests for web intrusion detection - Semantic Scholar, truy cập vào tháng 11 29, 2025, <https://www.semanticscholar.org/paper/Analyzing-HTTP-requests-for-web-intrusion-detection-Althubiti-Yuan/f3adfc7e7686114ce2cb1a1eb7dc22848fdf13ca>
15. python - Detect SQL injections in the source code - Stack Overflow, truy cập vào tháng 11 29, 2025, <https://stackoverflow.com/questions/27402426/detect-sql-injections-in-the-source-code>
16. Simplifying Custom Security Testing Protocols with Python - Penzzer, truy cập vào tháng 11 29, 2025, <https://www.we-fuzz.io/blog/simplifying-custom-security-testing-protocols-with-python>
17. APIFuzzer - PyPI, truy cập vào tháng 11 29, 2025, <https://pypi.org/project/APIFuzzer/>
18. Machine learning SQL injection detection - Python Forum, truy cập vào tháng 11 29, 2025, <https://python-forum.io/thread-6605.html>
19. geeknik/waf-bypass-tool: A robust, machine learning-powered tool for detecting and bypassing Web Application Firewalls. - GitHub, truy cập vào tháng 11 29, 2025, <https://github.com/geeknik/waf-bypass-tool>
20. A Flask API for serving scikit-learn models | by Amir Ziai | TDS Archive | Medium, truy cập vào tháng 11 29, 2025, <https://medium.com/data-science/a-flask-api-for-serving-scikit-learn-models-c8bcdaa41daa>
21. AIWAF Flask: Drop in Security Middleware with AI Anomaly Detection - Reddit, truy cập vào tháng 11 29, 2025, <https://www.reddit.com/r/flask/comments/1niw2ii/aiwaf_flask_drop_in_security_middleware_with_ai/>
22. Flask Middlewares - GeeksforGeeks, truy cập vào tháng 11 29, 2025, <https://www.geeksforgeeks.org/python/flask-middlewares/>
23. Web application firewall based on machine learning models - PeerJ, truy cập vào tháng 11 29, 2025, <https://peerj.com/articles/cs-2975.pdf>
24. The evolution of input security: From SQLi & XSS to prompt injection in large language models - ASAPP, truy cập vào tháng 11 29, 2025, <https://www.asapp.com/blog/the-evolution-of-input-security-from-sqli-xss-to-prompt-injection-in-large-language-models>
25. Security Considerations — Flask Documentation (3.1.x), truy cập vào tháng 11 29, 2025, <https://flask.palletsprojects.com/en/stable/web-security/>