

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM**  
**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**  
**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐÀO HỮU PHI QUÂN - 52000386**  
**NGUYỄN THANH TRIỀU - 52100495**

**ỨNG DỤNG MACHINE LEARNING**  
**TRONG PHÒNG CHỐNG DDOS**  
**DỰ ÁN CÔNG NGHỆ THÔNG TIN**  
**MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG**  
**DỮ LIỆU**

Người hướng dẫn  
**TS. Bùi Quy Anh**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2025**

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM**  
**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**  
**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐÀO HỮU PHI QUÂN - 52000386**  
**NGUYỄN THANH TRIỀU - 52100495**

**ỨNG DỤNG MACHINE LEARNING**  
**TRONG PHÒNG CHỐNG DDOS**  
**DỰ ÁN CÔNG NGHỆ THÔNG TIN**  
**MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG**  
**DỮ LIỆU**

Người hướng dẫn  
**TS. Bùi Quy Anh**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2025**

## LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành và sâu sắc đến quý thầy cô Trường Đại học Tôn Đức Thắng, đặc biệt là các thầy cô thuộc Khoa Công nghệ Thông tin đã tận tình giảng dạy, truyền đạt kiến thức và tạo điều kiện thuận lợi để em hoàn thành tốt bài báo cáo cuối kỳ này.

Em xin đặc biệt bày tỏ lòng biết ơn đến thầy Bùi Quy Anh, người đã luôn tận tâm, nhiệt huyết trong giảng dạy và đồng hành cùng lớp trong suốt quá trình học tập. Phương pháp giảng dạy sinh động, gần gũi và truyền cảm hứng của thầy đã giúp em có thêm nhiều động lực để hoàn thiện đề tài của mình.

Mặc dù đã cố gắng hoàn thành bài báo cáo với tinh thần nghiêm túc và trách nhiệm cao, nhưng do hạn chế về kiến thức và kinh nghiệm thực tiễn, chắc chắn không thể tránh khỏi những thiếu sót. Em rất mong nhận được sự thông cảm và những góp ý quý báu từ quý thầy cô để em có thể rút kinh nghiệm và hoàn thiện hơn trong các lần sau.

Em xin chân thành cảm ơn!

## BÁO CÁO ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là sản phẩm báo cáo của riêng em và được sự hướng dẫn của thầy Bùi Quy Anh. Các nội dung nghiên cứu, kết quả trong báo cáo này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đề án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đề án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày tháng năm*

*Tác giả*

*(ký tên và ghi rõ họ tên)*



*Đào Hữu Phi Quân*

*Triệu*

*Nguyễn Thanh Triệu*

## **PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN**

### **Phần xác nhận của GV hướng dẫn**

---

---

---

---

---

---

---

Tp. Hồ Chí Minh, ngày    tháng    năm  
(kí và ghi họ tên)

### **Phần đánh giá của GV chấm bài**

---

---

---

---

---

---

---

Tp. Hồ Chí Minh, ngày    tháng    năm  
(kí và ghi họ tên)

## TÓM TẮT

Trong bối cảnh các cuộc tấn công từ chối dịch vụ phân tán (DDoS) ngày càng gia tăng cả về quy mô và độ tinh vi, việc phát hiện và ngăn chặn kịp thời các hành vi bất thường là yêu cầu cấp thiết nhằm đảm bảo tính sẵn sàng và an toàn của hệ thống mạng.

Mục tiêu của đề tài là nghiên cứu và ứng dụng các thuật toán học máy (Machine Learning), đặc biệt là mô hình mạng nơ-ron tích chập (CNN), để xây dựng hệ thống phát hiện và ngăn chặn tấn công DDoS trong môi trường mạng SDN. Hệ thống được thiết kế để nhận diện sớm các luồng lưu lượng tấn công, đồng thời phản ứng tự động như chặn IP và gửi cảnh báo khi cần thiết.

### **Nội dung đã thực hiện bao gồm:**

- Tiền xử lý dữ liệu: Làm sạch, chuẩn hóa, mã hóa, chia train/test.
- Xây dựng và huấn luyện các mô hình ML (KNN, Logistic Regression, Naive Bayes, Decision Tree, Random Forest, SVM) và DL (Neural Network, CNN, LSTM).
- Đánh giá hiệu năng mô hình qua các chỉ số: Accuracy, Precision, Recall, F1-score, confusion matrix.
- Xây dựng hệ thống phát hiện DDoS trên SDN controller sử dụng Ryu + mô hình CNN.
- Triển khai mô phỏng tấn công thực tế bằng Mininet để kiểm thử khả năng nhận diện và phản ứng của hệ thống.
- Tự động gửi email cảnh báo và log lại sự kiện khi có tấn công xảy ra.

Kết quả cho thấy mô hình CNN tích hợp trong môi trường SDN có khả năng phát hiện hiệu quả các luồng tấn công DDoS, mở ra hướng phát triển các hệ thống phòng thủ chủ động, thông minh và thích ứng với các kiểu tấn công mới.

# MỤC LỤC

LỜI CẢM ƠN .....	iii
BÁO CÁO ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG .....	iv
PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN .....	v
TÓM TẮT .....	vi
MỤC LỤC .....	1
DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT .....	4
CHƯƠNG 1:           GIỚI THIỆU VÀ ĐẶT VẤN ĐỀ .....	5
1.1    Bối cảnh và động lực .....	5
1.2    Mục tiêu và phạm vi nghiên cứu .....	6
CHƯƠNG 2:           CƠ SỞ LÝ THUYẾT .....	7
2.1    Tấn công DDoS là gì? .....	7
2.2    Phân loại các kiểu tấn công DDoS .....	8
2.2.1    Tấn công ở tầng ứng dụng (Layer 7) .....	8
2.2.2    Tấn công ở tầng giao vận (Layer 4) .....	9
2.2.3    Tấn công ở tầng mạng (Layer 3) .....	10
2.2.4    Các hình thức tấn công DDoS hiện đại .....	10
2.3    Tổng quan về SDN (Software Defined Networking) .....	11
2.3.1    Kiến trúc và nguyên lý hoạt động của SDN .....	11
2.3.2    Ưu điểm và tiềm năng của SDN trong bảo mật mạng .....	12
2.3.3    Vai trò của SDN trong hệ thống phát hiện DDoS sử dụng học máy	
14	
2.4    Tập dữ liệu sử dụng trong nghiên cứu .....	15
2.4.1    Dữ liệu thô chưa chuẩn hóa .....	15
2.4.2    Dữ liệu đã chuẩn hóa .....	16
2.5    Các công cụ, mô hình và nền tảng liên quan .....	17
2.5.1    Mininet .....	17
2.5.2    Ryu Controller .....	18

2.5.3	Máy ảo (Virtual Machine).....	22
2.5.4	Các mô hình học máy đã thử nghiệm.....	24
2.5.5	Mạng nơ-ron tích chập (CNN) .....	26
2.5.6	TensorFlow Lite .....	27
CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG PHÁT HIỆN TẤN CÔNG		
DDOS SỬ DỤNG HỌC MÁY TRÊN KIẾN TRÚC MẠNG SDN .....		28
3.1	Tổng quan về hệ thống đề xuất.....	28
3.2	Phân tích yêu cầu hệ thống .....	29
3.3	Kiến trúc hệ thống đề xuất.....	30
3.4	Thu thập và xử lý dữ liệu .....	31
3.5	Thiết kế mô hình học máy .....	31
3.6	Cơ chế phát hiện và phản ứng tấn công.....	31
CHƯƠNG 4: THỰC NGHIỆM .....		32
4.1	Dữ liệu thực nghiệm .....	32
4.2	Tiền xử lý dữ liệu.....	33
4.2.1	Tiền xử lý dữ liệu cho mô hình truyền thống.....	33
4.2.2	Tiền xử lý cho mô hình CNN.....	33
4.3	Huấn luyện và triển khai mô hình CNN .....	34
4.3.1	Kiến trúc mô hình CNN .....	34
4.3.2	Huấn luyện và chuyển đổi mô hình.....	35
4.4	Tích hợp mô hình vào controller và cơ chế phát hiện phản ứng .....	36
4.4.1	Khởi tạo mô hình học máy .....	36
4.4.2	Nhận và xử lý luồng mạng (PacketIn) .....	36
4.4.3	Môi trường triển khai, công cụ sử dụng.....	36
4.4.4	Hiển thị kết quả lên terminal .....	37
4.4.5	Hành động phản ứng .....	37
4.4.6	Giao tiếp với mô-đun giám sát.....	37
4.5	Công cụ sử dụng và cài đặt môi trường triển khai.....	37
4.5.1	Cài đặt môi trường triển khai .....	38
4.6	Triển khai thử nghiệm và demo thực tế .....	41



4.7	Đánh giá khả năng phát hiện và phản ứng của mô hình .....	46
4.7.1	Hiệu quả phát hiện trong điều kiện thực tế .....	46
4.7.2	So sánh với giải pháp truyền thống.....	46
4.7.3	Kết luận đánh giá demo thực tế.....	47
CHƯƠNG 5:	KẾT LUẬN.....	47
5.1	Kết luận .....	47
5.2	Những hạn chế và công việc chưa thực hiện được .....	48
5.3	Hướng phát triển .....	49
TÀI LIỆU THAM KHẢO	.....	50

## DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

### CÁC CHỮ VIẾT TẮT

<b>API</b>	Application Programming Interface
<b>CDN</b>	Content Delivery Network
<b>CNN</b>	Convolutional Neural Network
<b>DL</b>	Deep Learning
<b>DNS</b>	Domain Name System
<b>DoS</b>	Denial of Service
<b>DDoS</b>	Distributed Denial of Service
<b>GUI</b>	Graphical User Interface
<b>ICMP</b>	Internet Control Message Protocol
<b>ICS</b>	Industrial Control System
<b>IDS</b>	Intrusion Detection System
<b>IPS</b>	Intrusion Prevention System
<b>IP</b>	Internet Protocol
<b>IoT</b>	Internet of Things
<b>KNN</b>	K-Nearest Neighbors
<b>MAC</b>	Media Access Control
<b>ML</b>	Machine Learning
<b>NAT</b>	Network Address Translation
<b>NTP</b>	Network Time Protocol
<b>SDN</b>	Software Defined Networking
<b>SVM</b>	Support Vector Machine
<b>TCP</b>	Transmission Control Protocol
<b>UDP</b>	User Datagram Protocol
<b>VM</b>	Virtual Machine

## CHƯƠNG 1: GIỚI THIỆU VÀ ĐẶT VẤN ĐỀ

Chương này sẽ trình bày tổng quan về bối cảnh và lý do chọn đề tài, giới thiệu sơ lược về tấn công DDoS và các thách thức trong việc phát hiện, phòng chống tấn công này trong các hệ thống mạng hiện đại. Ngoài ra, chương cũng sẽ làm rõ mục tiêu nghiên cứu, phạm vi thực hiện, phương pháp tiếp cận và cấu trúc tổng thể của báo cáo.

### 1.1 Bối cảnh và động lực

Sự phát triển nhanh chóng của công nghệ số đã khiến các hoạt động kinh tế – xã hội ngày càng phụ thuộc sâu sắc vào hệ thống mạng máy tính và các dịch vụ trực tuyến. Từ các ngân hàng, doanh nghiệp thương mại điện tử đến các hệ thống công nghiệp (ICS), tính ổn định và an toàn trong truyền thông mạng là điều kiện sống còn. Tuy nhiên, đi kèm với sự phát triển đó là sự gia tăng mạnh mẽ của các mối đe dọa an ninh mạng, trong đó đáng lo ngại nhất là các cuộc tấn công từ chối dịch vụ phân tán (DDoS) – hình thức tấn công phổ biến, dễ triển khai nhưng gây hậu quả nghiêm trọng.

Các cuộc tấn công DDoS hiện nay ngày càng tinh vi, không chỉ khiến dịch vụ bị gián đoạn mà còn gây tổn thất tài chính lớn và ảnh hưởng đến uy tín tổ chức bị tấn công. Với sự lan rộng của thiết bị IoT và mạng thông minh, kẻ tấn công có thể nhanh chóng xây dựng các mạng botnet khổng lồ để phát động các đợt tấn công quy mô lớn, vượt qua khả năng phát hiện của các hệ thống phòng thủ truyền thống.

Trong bối cảnh đó, Software Defined Networking (SDN) nổi lên như một kiến trúc mạng linh hoạt, hiện đại, cho phép tách rời mặt điều khiển khỏi mặt dữ liệu và quản lý toàn mạng thông qua controller trung tâm. Tuy nhiên, chính sự tập trung quyền kiểm soát vào controller cũng khiến SDN trở thành mục tiêu dễ bị khai thác trong các cuộc tấn công DDoS.

Việc phát triển một hệ thống giám sát và phát hiện tấn công DDoS hiệu quả, có khả năng phản ứng tức thời và thích nghi với các kiểu tấn công mới là rất cần thiết. Đặc biệt, khi kết hợp SDN với học máy (Machine Learning), khả năng quan sát toàn mạng và ra quyết định thông minh theo thời gian thực trở thành hiện thực. Đề tài này lựa chọn tích hợp mô hình học sâu CNN vào hệ thống SDN sử dụng Mininet và Ryu controller

nhằm xây dựng một cơ chế phát hiện DDoS chủ động, thông minh, và có thể triển khai thực tế.

## 1.2 Mục tiêu và phạm vi nghiên cứu

Mục tiêu của đề tài là xây dựng một hệ thống phát hiện và ngăn chặn tấn công DDoS dựa trên học máy (Machine Learning), tích hợp trực tiếp vào môi trường mạng Software Defined Networking (SDN). Cụ thể, đề tài sử dụng mô hình học sâu mạng nơ-ron tích chập (CNN) để phân tích và phân loại lưu lượng mạng, từ đó phát hiện sớm các luồng tấn công DDoS.

Hệ thống được triển khai và kiểm thử trên nền tảng Mininet kết hợp với bộ điều khiển Ryu Controller, cho phép mô phỏng mạng ảo SDN và phát động các hình thức tấn công phổ biến như TCP SYN Flood, UDP Flood và ICMP Flood. Đây là những loại tấn công điển hình ở tầng giao vận và tầng mạng có thể gây quá tải nhanh chóng cho controller, qua đó kiểm tra tính hiệu quả của cơ chế phòng thủ.

Về mặt xử lý, hệ thống tập trung thu thập dữ liệu luồng từ switch thông qua OpenFlow, sau đó trích xuất các đặc trưng thống kê như: địa chỉ IP nguồn – đích, số lượng gói tin, số byte, thời lượng luồng, cổng truyền, loại giao thức,... Các đặc trưng này được chuẩn hóa và gán nhãn để huấn luyện mô hình CNN dưới định dạng TensorFlow Lite, phù hợp với môi trường thực thi thời gian thực trong controller.

Hệ thống cũng được thiết kế với khả năng phản ứng tự động. Khi mô hình phát hiện lưu lượng nghi vấn là DDoS, controller sẽ ngay lập tức thực hiện các hành động như chặn địa chỉ IP nguồn (cài đặt luật drop), ghi log sự kiện và gửi email cảnh báo đến người quản trị.

Ngoài ra, hệ thống còn được tích hợp một công cụ giám sát thời gian thực giúp người quản trị dễ dàng theo dõi các địa chỉ IP đang thực hiện tấn công, số lần bị phát hiện và thời điểm cập nhật gần nhất, hỗ trợ quá trình đánh giá và phân tích sau khi hệ thống phản ứng.

Phạm vi đề tài hiện tại chưa mở rộng đến các mô hình không giám sát (Unsupervised Learning), học tăng cường (Reinforcement Learning), hoặc triển khai thực tế trên hạ tầng vật lý. Tuy nhiên, kết quả đạt được bước đầu cho thấy tiềm năng ứng

dụng của học sâu kết hợp SDN trong phát hiện DDoS, tạo nền tảng cho các nghiên cứu nâng cao trong tương lai.

Sử dụng cơ sở dữ liệu nào?

## CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

Chương này cung cấp các kiến thức nền tảng cần thiết cho việc nghiên cứu và triển khai hệ thống phát hiện tấn công DDoS sử dụng học máy trong môi trường mạng định nghĩa bằng phần mềm (SDN). Các nội dung trình bày bao gồm: khái niệm và phân loại tấn công DDoS, nguyên lý hoạt động và đặc điểm của mạng SDN, cùng với vai trò của SDN trong việc hỗ trợ phát hiện và ngăn chặn các cuộc tấn công mạng nhờ vào khả năng kết hợp với các thuật toán học máy.

### 2.1 Tấn công DDoS là gì?

Tấn công từ chối dịch vụ phân tán (Distributed Denial of Service – DDoS) là một hình thức tấn công mạng có mục đích làm gián đoạn hoặc tê liệt hoàn toàn hoạt động của một dịch vụ, máy chủ hoặc hệ thống bằng cách gửi đến một lượng lớn yêu cầu giả mạo trong thời gian ngắn, vượt quá khả năng xử lý của hệ thống. Tài nguyên như CPU, bộ nhớ, băng thông và kết nối mạng sẽ nhanh chóng bị chiếm dụng, khiến dịch vụ không thể phản hồi người dùng hợp pháp.

Không giống với tấn công từ chối dịch vụ đơn lẻ (DoS), DDoS huy động nhiều thiết bị bị chiếm quyền điều khiển (botnet) – bao gồm máy tính cá nhân, thiết bị IoT, máy chủ yếu bảo mật – đồng loạt gửi các gói tin tấn công theo kịch bản lập trình sẵn. Sự phân tán này không chỉ giúp tăng hiệu quả tấn công mà còn làm cho việc truy vết và phản ứng trở nên khó khăn hơn.

Tấn công DDoS hiện đại không còn đơn thuần gây gián đoạn hệ thống, mà còn được sử dụng như công cụ để tổng tiền, đánh lạc hướng khi xâm nhập hệ thống, hoặc gây thiệt hại cho đối thủ cạnh tranh. Sự phát triển của các công nghệ mới như Internet of Things (IoT), mạng 5G và Software Defined Networking (SDN) càng khiến các cuộc tấn công DDoS trở nên tinh vi và khó đối phó hơn. Đặc biệt trong SDN – nơi toàn bộ

điều khiển mạng được tập trung tại controller – nếu bị tấn công sẽ làm gián đoạn toàn bộ hệ thống.

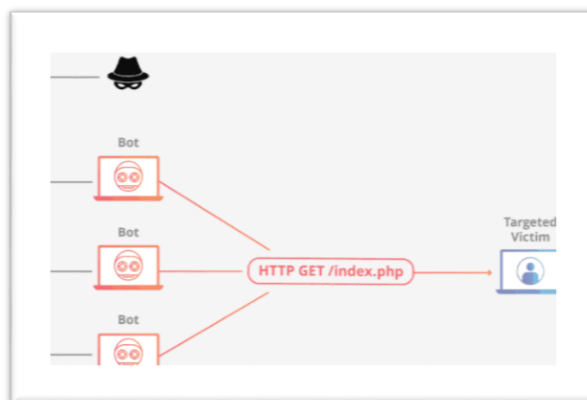
Thực tế cho thấy các hình thức DDoS ngày càng đa dạng như: TCP SYN Flood, UDP Flood, ICMP Flood, hoặc tấn công kết hợp đa tầng (multi-vector). Việc nhận diện và ngăn chặn DDoS đòi hỏi các giải pháp tự động, chính xác và thích ứng nhanh – điều mà các phương pháp truyền thống khó đáp ứng kịp.

Do đó, việc ứng dụng học máy (Machine Learning) và học sâu (Deep Learning), như mạng nơ-ron tích chập (CNN), để phát hiện các dấu hiệu tấn công trong thời gian thực đang được đánh giá là hướng tiếp cận đầy tiềm năng. Khi kết hợp với kiến trúc SDN, các mô hình học máy có thể khai thác dữ liệu luồng tập trung từ controller để phát hiện và phản ứng hiệu quả với các cuộc tấn công DDoS ngay từ giai đoạn đầu.

## 2.2 Phân loại các kiểu tấn công DDoS

Tấn công DDoS có thể được phân loại dựa trên tầng của mô hình OSI mà nó nhắm đến. Mỗi loại tấn công có cơ chế hoạt động và mục tiêu riêng, đòi hỏi các kỹ thuật phát hiện và phản ứng phù hợp. Dưới đây là ba nhóm chính được nhận diện phổ biến hiện nay.

### 2.2.1 Tấn công ở tầng ứng dụng (Layer 7)



Hình 1. Tấn công ở Layer 7

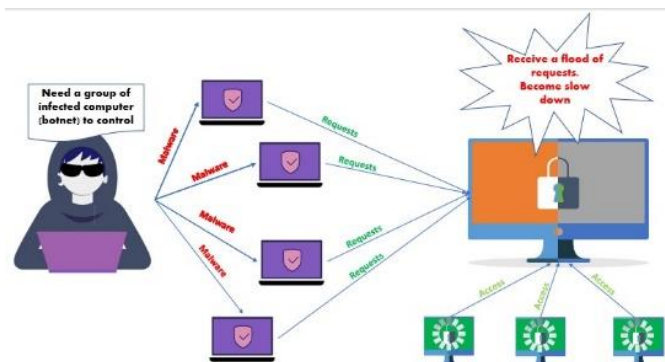
(<https://blog.vinahost.vn/application-layer-attack-la-gi/>)

Layer 7 là nơi giao tiếp trực tiếp với người dùng qua các giao thức như HTTP, DNS, FTP,... Các cuộc tấn công tại tầng này thường nhằm vào logic xử lý của ứng dụng, gây quá tải CPU hoặc tài nguyên backend.

Ví dụ tiêu biểu là HTTP GET/POST Flood, trong đó botnet liên tục gửi truy vấn hợp lệ để buộc máy chủ xử lý. Hay như Slowloris, duy trì nhiều kết nối HTTP nhưng không hoàn tất, khiến server giữ kết nối lâu và tốn tài nguyên.

Tấn công Layer 7 tuy không tạo nhiều lưu lượng nhưng rất khó phát hiện vì lưu lượng giống người dùng thật, vượt qua hầu hết các cơ chế giám sát truyền thống.

### 2.2.2 Tấn công ở tầng giao vận (Layer 4)



Hình 2. Tấn công ở tầng Layer 4

(<https://blog.vinahost.vn/application-layer-attack-la-gi/>)

Tầng giao vận chịu trách nhiệm về thiết lập kết nối đầu cuối giữa hai thiết bị, với hai giao thức phổ biến là TCP và UDP. Tấn công DDoS ở tầng này thường lợi dụng đặc điểm của quá trình thiết lập kết nối hoặc đặc điểm không xác thực của UDP để làm tê liệt server hoặc gây nghẽn đường truyền. Các kiểu tấn công phổ biến bao gồm:

**TCP SYN Flood:** gửi hàng loạt gói SYN giả nhưng không gửi ACK, khiến server giữ kết nối “nửa mở”, nhanh chóng chiếm hết tài nguyên kết nối.

**UDP Flood:** gửi liên tục các gói UDP đến cổng ngẫu nhiên, buộc hệ thống phản hồi ICMP Destination Unreachable, làm quá tải CPU hoặc băng thông.

Tấn công Layer 4 thường tạo ra **lưu lượng lớn với tốc độ cao**, dễ phát hiện hơn so với Layer 7, nhưng lại có thể làm nghẽn mạng và tê liệt server chỉ trong vài giây nếu không có cơ chế bảo vệ phù hợp.

### 2.2.3 Tấn công ở tầng mạng (Layer 3)



Hình 3. Tấn công ở tầng Layer 3 (<https://vietnix.vn/tan-cong-ddos-lop-3/>)

Tầng mạng là nơi chịu trách nhiệm định tuyến và phân phối các gói dữ liệu đến đích. Tấn công ở tầng này thường sử dụng các giao thức như ICMP để khai thác hoặc làm ngập tài nguyên mạng của mục tiêu. Layer 3 chịu trách nhiệm định tuyến gói tin. Một số hình thức tấn công tại đây bao gồm:

- **ICMP Flood:** gửi số lượng lớn gói ping khiến hệ thống phản hồi liên tục, tiêu tốn tài nguyên.
- **Smurf Attack:** gửi ICMP tới địa chỉ broadcast với IP giả, khiến nhiều máy cùng phản hồi về nạn nhân, gây nghẽn mạng.

Tấn công ở tầng mạng có thể rất **rộng về quy mô và đơn giản về kỹ thuật**, dễ phát động từ nhiều thiết bị IoT với bảo mật kém. Mặc dù một số mạng hiện đại đã tắt chức năng broadcast hoặc lọc ICMP, nhưng trong nhiều môi trường công nghiệp hoặc hệ thống điều khiển cũ, chúng vẫn tồn tại và dễ bị khai thác.

### 2.2.4 Các hình thức tấn công DDoS hiện đại

Ngoài các kiểu tấn công truyền thống, nhiều kỹ thuật tấn công DDoS mới đã xuất hiện với tính phức tạp và mức phá hoại cao hơn:

- **Tấn công đa vector:** kết hợp nhiều kỹ thuật như UDP Flood + SYN Flood + HTTP Flood cùng lúc, gây khó khăn trong việc phát hiện và phản ứng.



- **Botnet IoT:** sử dụng hàng triệu thiết bị IoT bị chiếm quyền điều khiển để tạo lưu lượng tấn công lớn, khó truy vết.
- **Khuếch đại (Amplification):** như DNS/NTP/Memcached Amplification, trong đó kẻ tấn công gửi truy vấn nhỏ nhưng nhận phản hồi rất lớn từ máy chủ trung gian, gây ngập băng thông. Tỷ lệ khuếch đại có thể lên đến hàng chục nghìn lần.
- **Tấn công qua dịch vụ Cloud/CDN hợp pháp:** gửi lưu lượng tấn công từ các nền tảng uy tín như Google Cloud, AWS, Cloudflare, khiến hệ thống khó phân biệt thật – giả.
- **Tấn công “Low and Slow”:** gửi yêu cầu cực chậm, duy trì kết nối lâu nhưng không hoàn tất. Dạng tấn công này rất khó phát hiện vì không vượt ngưỡng băng thông hoặc số lượng kết nối.

## 2.3 Tổng quan về SDN (Software Defined Networking)

Trong bối cảnh mạng máy tính ngày càng phát triển về quy mô và độ phức tạp, các hệ thống mạng truyền thống đã bộc lộ nhiều hạn chế trong việc quản lý, mở rộng và đảm bảo an toàn thông tin. Software Defined Networking (SDN) – Mạng Điều khiển bằng Phần mềm – đã ra đời như một giải pháp mang tính cách mạng, giúp tách biệt hoàn toàn chức năng điều khiển khỏi chức năng truyền dữ liệu trong hệ thống mạng. Sự tách biệt này không chỉ tăng tính linh hoạt và khả năng kiểm soát mạng mà còn tạo điều kiện thuận lợi cho việc tích hợp các công nghệ mới như trí tuệ nhân tạo, học máy, và các cơ chế bảo mật thông minh.

### 2.3.1 Kiến trúc và nguyên lý hoạt động của SDN

Kiến trúc SDN được thiết kế với ba lớp riêng biệt: lớp dữ liệu (Data Plane), lớp điều khiển (Control Plane) và lớp ứng dụng (Application Plane), trong đó mỗi lớp có một vai trò cụ thể nhưng chặt chẽ liên kết với nhau.

Lớp dữ liệu là nơi chứa các thiết bị vật lý như switch hoặc router. Trong kiến trúc SDN, các thiết bị này không còn giữ chức năng điều khiển hay định tuyến, mà chỉ thực

hiện việc chuyển tiếp gói tin dựa trên bảng điều khiển được thiết lập từ Controller. Chúng trở thành các phần tử mạng “thụ động” và chỉ nhận lệnh từ trung tâm điều khiển.

Lớp điều khiển là trung tâm của hệ thống, đóng vai trò như "bộ não" của mạng. Tại đây, một hoặc nhiều SDN Controller được triển khai để quản lý toàn bộ hệ thống mạng. Controller có thể thu thập thông tin về trạng thái mạng, phân tích lưu lượng, đưa ra quyết định định tuyến, và điều phối hoạt động của tất cả các thiết bị trong lớp dữ liệu. Một trong những đặc điểm quan trọng của lớp này là khả năng lập trình và điều khiển toàn bộ mạng từ một điểm trung tâm.

Lớp ứng dụng, nằm trên cùng, là nơi triển khai các dịch vụ mạng như cân bằng tải, giám sát lưu lượng, phát hiện tấn công, tối ưu hóa định tuyến hay điều phối băng thông. Các ứng dụng này giao tiếp với Controller thông qua giao diện lập trình ứng dụng (API), từ đó tác động gián tiếp đến toàn bộ mạng thông qua các chính sách điều khiển.

Để thiết bị trong lớp dữ liệu và lớp điều khiển có thể giao tiếp, giao thức **OpenFlow** thường được sử dụng. OpenFlow là giao thức chuẩn cho phép các thiết bị chuyển mạch gửi thông tin flow (luồng dữ liệu) lên Controller, đồng thời nhận lại các lệnh xử lý tương ứng. Nhờ OpenFlow, hệ thống mạng có thể phản ứng một cách nhanh chóng với các thay đổi lưu lượng, điều chỉnh đường đi của gói tin, hoặc ngăn chặn truy cập từ các nguồn độc hại.

Toàn bộ nguyên lý hoạt động của SDN có thể hiểu là việc tập trung hóa quyền kiểm soát mạng về một Controller duy nhất hoặc cụm Controller. Điều này làm tăng khả năng quan sát, lập chính sách và phản ứng linh hoạt với các sự cố, sự kiện bất thường trong mạng.

### 2.3.2 Ưu điểm và tiềm năng của SDN trong bảo mật mạng

SDN mang lại nhiều lợi thế vượt trội trong việc xây dựng các hệ thống mạng bảo mật, chủ động và linh hoạt hơn so với kiến trúc mạng truyền thống. Cụ thể:

- **Kiểm soát tập trung và đồng bộ:** Trong mạng truyền thống, việc triển khai chính sách an ninh (như chặn IP, giới hạn băng thông, định tuyến lại lưu lượng...) thường đòi hỏi cấu hình thủ công trên từng thiết bị mạng – điều này vừa mất thời gian, vừa dễ gây lỗi. Ngược lại, trong SDN, toàn bộ chính sách an ninh chỉ cần

cấu hình tại controller, và ngay lập tức có thể được áp dụng trên toàn mạng một cách tự động, đồng bộ và nhất quán. Điều này giúp giảm thiểu sai sót và tăng khả năng phản ứng trước các tình huống khẩn cấp.

- **Giám sát lưu lượng toàn mạng theo thời gian thực:** SDN Controller có khả năng theo dõi trạng thái của toàn bộ mạng, bao gồm các luồng dữ liệu, số lượng packet, byte, port sử dụng,... Điều này tạo điều kiện lý tưởng để phát hiện các hành vi bất thường như quét port, flood gói tin, hoặc các dấu hiệu của tấn công DDoS. Không chỉ vậy, dữ liệu được thu thập từ switch thông qua OpenFlow còn giúp controller có cái nhìn toàn diện và sâu sát hơn về hành vi mạng.
- **Phản ứng linh hoạt và gần như tức thời:** Khi phát hiện mối đe dọa, controller có thể ngay lập tức thực hiện các biện pháp phòng vệ như: chặn IP tấn công, xóa rule cũ và cài đặt rule mới, giới hạn băng thông theo IP nguồn,... Những thay đổi này chỉ mất vài mili-giây để cập nhật xuống switch nhờ kiến trúc điều khiển tập trung – một ưu điểm mà các hệ thống mạng truyền thống khó đạt được.
- **Khả năng tích hợp học máy hiệu quả:** Do toàn bộ dữ liệu flow được thu thập tập trung tại controller, SDN là nền tảng rất phù hợp để kết hợp với các mô hình Machine Learning. Các mô hình này có thể học từ hành vi mạng thông thường để phát hiện sớm các biểu hiện của tấn công (ví dụ: tăng đột biến packet/s, nguồn gửi gói bất thường,...). Khi được tích hợp, hệ thống sẽ không chỉ phát hiện mà còn phản ứng thông minh mà không cần con người can thiệp.
- **Môi trường thử nghiệm mạnh mẽ cho nghiên cứu bảo mật:** Công cụ mô phỏng như Mininet cho phép thiết kế, kiểm thử và đánh giá các giải pháp an ninh mạng (bao gồm cả IDS, Firewall, mô hình học máy) trên nền SDN mà không cần đầu tư vào thiết bị phần cứng thật. Điều này không chỉ giúp giảm chi phí mà còn tăng khả năng tái lập các kịch bản tấn công phức tạp để nghiên cứu chuyên sâu hơn.
- **Tăng cường khả năng mở rộng và cập nhật hệ thống:** Với đặc tính lập trình được, SDN cho phép nhanh chóng cập nhật chính sách hoặc mở rộng tính năng bảo mật thông qua các API, module hoặc script bổ sung. Điều này rất cần thiết

trong môi trường an ninh mạng hiện đại, nơi các hình thức tấn công thay đổi liên tục và không ngừng tiến hóa.

Tổng thể, SDN không chỉ đơn thuần là một kiến trúc mạng linh hoạt mà còn là nền tảng vững chắc để triển khai các giải pháp an ninh hiện đại – đặc biệt khi kết hợp với các mô hình học máy nhằm phát hiện, ngăn chặn và thích ứng với các kiểu tấn công như DDoS một cách tự động và hiệu quả.

### 2.3.3 Vai trò của SDN trong hệ thống phát hiện DDoS sử dụng học máy

Việc kết hợp SDN và học máy (Machine Learning) mở ra khả năng xây dựng hệ thống phòng thủ chủ động, thông minh và thích ứng. Trong kiến trúc này:

- **Controller** có vai trò trung tâm, thu thập đặc trưng lưu lượng từ các switch thông qua giao thức OpenFlow. Các đặc trưng này thường bao gồm địa chỉ IP nguồn – đích, cổng giao tiếp, loại giao thức, số lượng gói, số byte, tốc độ gửi, thời gian tồn tại của flow,...
- **Mô hình học máy** (cụ thể là mạng nơ-ron tích chập – CNN) được tích hợp trực tiếp vào controller, cho phép phân tích và phân loại luồng lưu lượng theo thời gian thực (real-time). Mô hình này giúp phát hiện sớm những luồng có hành vi bất thường mà các phương pháp thủ công khó có thể nhận diện kịp.
- **Hành động phản ứng được thực hiện tức thời:** khi phát hiện tấn công, controller có thể tự động thực hiện các bước như:
  - Cài đặt rule chặn IP nguồn tấn công (drop).
  - Ghi log lại thời gian và địa chỉ tấn công.
  - Gửi email cảnh báo cho quản trị viên để phản hồi kịp thời.
  - Cập nhật bảng điều khiển để thay đổi đường đi của lưu lượng hoặc giới hạn tốc độ truyền tải.

Điểm nổi bật trong hệ thống chúng tôi xây dựng là sử dụng mô hình CNN đã được huấn luyện và chuyển đổi sang định dạng TensorFlow Lite, giúp quá trình suy luận (inference) diễn ra nhanh chóng, tiết kiệm tài nguyên – rất phù hợp với môi trường thực thi như Ryu Controller. Ngoài ra, mô-đun giám sát riêng cũng được tích hợp nhằm hiển

thị trực quan các luồng tấn công, tần suất xuất hiện theo IP, hỗ trợ công tác giám sát và đánh giá hiệu quả hệ thống trong thực tế.

Hệ thống được thiết kế theo hướng tự động hóa, không cần sự can thiệp của con người trong giai đoạn phát hiện và xử lý, từ đó giảm thiểu thời gian phản ứng, đồng thời nâng cao khả năng bảo vệ mạng trong môi trường có nguy cơ tấn công cao. Bên cạnh đó, hệ thống còn có khả năng mở rộng để huấn luyện lại mô hình khi xuất hiện các kiểu tấn công mới, từng bước hướng đến một hệ thống phòng thủ tự thích nghi (adaptive defense system) – một hướng đi tất yếu trong tương lai của an ninh mạng.

## 2.4 Tập dữ liệu sử dụng trong nghiên cứu

### 2.4.1 Dữ liệu thô chưa chuẩn hóa

Dữ liệu gốc được sử dụng trong đề tài có định dạng flow-based, được trích xuất từ hệ thống SDN với các thông tin lưu lượng mạng giữa các host trong hệ thống. Tập dữ liệu bao gồm nhiều dòng, mỗi dòng tương ứng với một luồng dữ liệu (flow) và được mô tả bởi các đặc trưng kỹ thuật như:

- **Thông tin định danh và mạng:** timestamp, datapath\_id, flow\_id, ip\_src, tp\_src, ip\_dst, tp\_dst
- **Thông tin giao thức:** ip\_proto, icmp\_code, icmp\_type
- **Thông số thời gian luồng:** flow\_duration\_sec, flow\_duration\_nsec, idle\_timeout, hard\_timeout
- **Thông số kỹ thuật luồng:** flags, packet\_count, byte\_count
- **Chỉ số tốc độ:** packet\_count\_per\_second, byte\_count\_per\_second, packet\_count\_per\_nsecond, byte\_count\_per\_nsecond
- **Nhãn phân loại:** label (0 = bình thường, 1 = tấn công DDoS)

Định dạng này rất phù hợp cho việc huấn luyện mô hình học máy vì cung cấp đầy đủ các đặc trưng phản ánh hành vi của luồng dữ liệu. Tuy nhiên, dữ liệu ở trạng thái ban đầu thường có các đặc điểm không đồng nhất, có thể chứa các giá trị nhiễu, chênh lệch đơn vị, hoặc phân phối không cân bằng.

### 2.4.2 Dữ liệu đã chuẩn hóa

Sau khi thu thập dữ liệu thô, nhóm tiến hành chuẩn hóa và tiền xử lý dữ liệu để phù hợp với việc huấn luyện mô hình<sup>[1]</sup>. Các bước thực hiện bao gồm:

- **Loại bỏ các cột không cần thiết hoặc có tính định danh cao** như flow\_id, ip\_src, ip\_dst, nhằm giảm nhiễu và tránh overfitting.

```
df = df.drop(['flow_id', 'ip_src', 'ip_dst'], axis=1)
```

- **Chuẩn hóa toàn bộ đặc trưng đầu vào** bằng các phương pháp như Min-Max Scaling hoặc StandardScaler. Mục tiêu là đưa tất cả đặc trưng về cùng một thang giá trị (thường là [0, 1] hoặc có trung bình = 0, phương sai = 1), giúp mô hình học tốt hơn.

```
scaler = MinMaxScaler()
features_scaled = scaler.fit_transform(features)
features_scaled_df = pd.DataFrame(features_scaled,
columns=features.columns)
features_scaled_df['label'] = labels.reset_index(drop=True)
```

- **One-hot encoding cho các nhãn phân loại (label)** để phù hợp với mô hình CNN có đầu ra dạng softmax.

```
y_encoded = to_categorical(y, num_classes=2)
```

- **Định dạng lại dữ liệu (reshaping)**: Dữ liệu được chuyển sang định dạng tensor 3D để phù hợp với kiến trúc Conv1D của mô hình CNN: (số mẫu, số đặc trưng, 1).

```
X_resaped = X.reshape(-1, X.shape[1], 1)
```

Đặc biệt, các giá trị như tp\_src, tp\_dst, flow\_duration, packet\_count, byte\_count,... sau khi chuẩn hóa sẽ tránh được tình trạng một đặc trưng chi phối toàn bộ mô hình do chênh lệch đơn vị quá lớn.

Việc chuẩn hóa dữ liệu không chỉ giúp tăng hiệu quả học của mô hình CNN mà còn đảm bảo mô hình không bị sai lệch do các yếu tố ngoại vi, từ đó tăng độ chính xác trong việc phát hiện tấn công DDoS.

<sup>1</sup> Nhóm đã chuẩn hóa trên Google Colab:

<https://colab.research.google.com/drive/1ynw7ghsjSDrpnxyMcw439Plk3fY5Q0I4>

## 2.5 Các công cụ, mô hình và nền tảng liên quan

### 2.5.1 Mininet

Mininet là một công cụ mô phỏng mạng mã nguồn mở phổ biến, cho phép tạo ra một mạng ảo bao gồm các switch, host và controller chạy trên cùng một máy chủ vật lý hoặc máy ảo. Đây là công cụ lý tưởng để nghiên cứu và phát triển các giải pháp mạng, đặc biệt trong bối cảnh SDN.

Mininet hoạt động bằng cách sử dụng các container mạng nhẹ (dựa trên Linux namespaces và cgroups) để tạo ra các host ảo, các switch ảo (thường là Open vSwitch) và các liên kết mạng ảo giữa chúng. Người dùng có thể viết các script bằng Python để cấu hình topology tùy chỉnh, khởi tạo traffic, và mô phỏng các tình huống tấn công hoặc sự cố mạng.

- Ưu điểm nổi bật của Mininet:
- Thử nghiệm kịch bản phức tạp mà không cần phần cứng thật: Người dùng có thể mô phỏng hàng loạt kịch bản thực tế như tấn công DDoS, sự cố mạng, mất kết nối,... trên một máy tính cá nhân hoặc máy ảo mà không cần đầu tư thiết bị vật lý đắt tiền.
- Tích hợp dễ dàng với các controller SDN như Ryu, POX, ONOS: Mininet hỗ trợ giao thức OpenFlow và có thể kết nối trực tiếp với nhiều controller khác nhau, giúp người dùng kiểm thử logic điều khiển mạng và các ứng dụng bảo mật như phát hiện DDoS trong môi trường SDN.
- Triển khai nhanh và dễ sử dụng: Chỉ với vài dòng lệnh Python hoặc Bash, người dùng có thể tạo topology mạng tùy chỉnh, giúp tiết kiệm thời gian khi thử nghiệm các giải pháp mới hoặc khi trình diễn nghiên cứu.
- Khả năng mô phỏng mạng quy mô lớn: Với cấu hình hợp lý, Mininet có thể mô phỏng hàng trăm host và switch, đủ để đánh giá hành vi mạng dưới các kịch bản tấn công quy mô lớn hoặc kiểm thử độ ổn định của hệ thống SDN.

Tổng thể, Mininet đóng vai trò là môi trường thử nghiệm chính giúp đánh giá hiệu quả của hệ thống phát hiện DDoS trước khi triển khai trên hạ tầng vật lý thực tế.

## 2.5.2 Ryu Controller

### Cơ sở lý thuyết về Ryu

Ryu là một framework mã nguồn mở được viết bằng Python, dùng để phát triển các ứng dụng điều khiển mạng dựa trên kiến trúc Software Defined Networking (SDN). Đây là một trong những controller phổ biến nhất hiện nay nhờ tính đơn giản, dễ mở rộng và cộng đồng phát triển tích cực.

Ryu hỗ trợ nhiều phiên bản của giao thức OpenFlow (từ 1.0 đến 1.5), cho phép giao tiếp với các switch SDN (ví dụ: Open vSwitch). Thông qua OpenFlow, Ryu có thể thu thập thông tin lưu lượng, cài đặt quy tắc xử lý gói tin (flow rules) và điều khiển hành vi toàn bộ mạng từ một điểm trung tâm.

Các thành phần chính của Ryu bao gồm:

- **Ryu Manager:** Thành phần trung tâm chịu trách nhiệm khởi tạo, quản lý và điều phối các ứng dụng SDN đang chạy. Ryu Manager đóng vai trò như một trình điều hành, tạo môi trường để các ứng dụng có thể tương tác với hạ tầng mạng một cách đồng bộ và chính xác.
- **Ofctl REST API:** Ryu cung cấp một tập hợp các API RESTful để cho phép người dùng hoặc hệ thống bên ngoài dễ dàng tương tác với controller thông qua giao thức HTTP. Điều này giúp việc giám sát, cấu hình và quản lý mạng trở nên tiện lợi hơn, đồng thời cho phép tích hợp với các dashboard quản trị mạng.
- **Topology Discovery:** Tự động phát hiện và duy trì cấu trúc mạng (bao gồm switch, host, các liên kết vật lý). Tính năng này giúp controller có cái nhìn tổng quát về trạng thái mạng để đưa ra quyết định điều khiển chính xác.
- **Giao diện lập trình rõ ràng và sự kiện phong phú:** Ryu sử dụng hệ thống event-based rõ ràng, cho phép lập trình viên dễ dàng đăng ký và xử lý các sự kiện mạng như PacketIn (gói tin chưa có rule), FlowMod (cập nhật flow), SwitchFeatures (thông tin phân cứng switch), v.v.

Ưu điểm nổi bật của Ryu:

- **Giao diện lập trình đơn giản, linh hoạt:** Với cú pháp Python trực quan, người dùng dễ dàng viết các ứng dụng mạng tùy chỉnh như firewall, load balancer, hoặc



hệ thống phát hiện tấn công. Việc xây dựng logic điều khiển trên Ryu chỉ yêu cầu kiến thức cơ bản về mạng và lập trình Python.

- **Khả năng mở rộng và tích hợp mô hình học máy:** Trong đề tài này, nhóm đã tích hợp thành công mô hình học sâu CNN, được chuyển đổi sang định dạng TensorFlow Lite, vào môi trường thực thi của Ryu Controller. Điều này cho phép quá trình suy luận (inference) diễn ra ngay bên trong controller, giúp phát hiện các luồng tấn công DDoS gần như theo thời gian thực. Nhờ đó, controller có thể đưa ra phản ứng ngay lập tức mà không cần truyền dữ liệu sang hệ thống bên ngoài.
- **Khả năng xử lý sự kiện mạnh mẽ và có độ trễ thấp:** Ryu hỗ trợ đăng ký các sự kiện mạng và cho phép xử lý nhanh chóng mỗi khi có gói tin mới đến (event PacketIn). Khi nhận diện một flow đáng ngờ, Ryu có thể gửi phản hồi như drop gói, cài đặt flow mới vào switch, hoặc thay đổi đường đi của luồng dữ liệu. Tốc độ phản ứng của Ryu chỉ nằm trong khoảng vài mili-giây, rất phù hợp với môi trường có yêu cầu cao về thời gian phản hồi như bảo mật mạng.
- **Tương thích tốt với Mininet và Open vSwitch:** Ryu dễ dàng kết nối với môi trường mô phỏng mạng như Mininet và switch ảo như OVS. Điều này giúp nhóm triển khai và kiểm thử hệ thống toàn diện trước khi đưa vào môi trường vật lý thật, đồng thời cho phép tái lập các kịch bản tấn công phức tạp trong môi trường nghiên cứu.

Tóm lại, Ryu không chỉ là một controller mạnh mẽ mà còn là nền tảng cho các ứng dụng bảo mật mạng thông minh, có thể học và thích nghi với các mối đe dọa mới, đặc biệt khi được kết hợp cùng các mô hình học sâu như CNN.

### Cơ sở lý thuyết của mã nguồn controller.py

File controller.py<sup>[2]</sup> là chương trình chạy trên nền tảng Ryu SDN Controller, đóng vai trò **thu thập luồng dữ liệu mạng, trích xuất đặc trưng, đưa vào mô hình học máy CNN đã huấn luyện (dưới dạng TensorFlow Lite), dự đoán tấn công DDoS, và phản**

---

<sup>2</sup> Nhóm đã tạo file và up lên Github: <https://github.com/notQ1301/intelligent-ddos-detection>

**ứng tự động nếu phát hiện.** Mã nguồn được viết bằng Python, tận dụng API của Ryu, giao thức OpenFlow v1.3 và thư viện TensorFlow Lite.

#### Các thành phần chính:

- **from ryu.base import app\_manager:** Kế thừa lớp điều khiển ứng dụng SDN.
- **from ryu.controller.handler:** Đăng ký sự kiện như PacketIn, SwitchFeatures.
- **from ryu.ofproto import ofproto\_v1\_3:** Sử dụng giao thức OpenFlow v1.3 để điều khiển switch.

#### Tải mô hình CNN:

```
self.interpreter = tf.lite.Interpreter(model_path="...")
```

- Tải mô hình TensorFlow Lite .tflite vào bộ nhớ.

```
self.interpreter =
tf.lite.Interpreter(model_path="model_cnn.tflite")
self.interpreter.allocate_tensors()
# tf.lite.Interpreter() là lớp dùng để load mô hình .tflite.
# allocate_tensors() cấp phát bộ nhớ cho input/output tensors.
```

- Khởi tạo input/output tensors để chuẩn bị dự đoán.

```
self.input_details = self.interpreter.get_input_details()
self.output_details = self.interpreter.get_output_details()
# get_input_details() lấy thông tin tensor đầu vào (shape,
dtype,...).
# get_output_details() lấy thông tin tensor đầu ra (vị trí dự
đoán).
```

Trích xuất đặc trưng từ gói tin (PacketIn):

- Từ các gói tin mạng nhận được từ Mininet, controller lấy ra các thông tin như:

- IP nguồn, IP đích, giao thức (ip.proto)

```
ip_src = pkt_ipv4.src
ip_dst = pkt_ipv4.dst
proto = pkt_ipv4.proto
# pkt_ipv4 là gói tin IP được trích xuất từ lớp IPv4 bằng Ryu's
packet parser.
```

- Cổng nguồn/đích (tcp.src\_port, udp.dst\_port)

```
if proto == 6 and pkt_tcp:
    tp_src = pkt_tcp.src_port
```

```
tp_dst = pkt_tcp.dst_port
elif proto == 17 and pkt_udp:
    tp_src = pkt_udp.src_port
    tp_dst = pkt_udp.dst_port
# Kiểm tra giao thức là TCP (6) hay UDP (17) trước khi trích xuất cổng.
```

- o Loại gói ICMP (icmp.type, icmp.code)

```
elif proto == 1 and pkt_icmp:
    icmp_type = pkt_icmp.type
    icmp_code = pkt_icmp.code
# ICMP có giao thức số 1. Kiểm tra điều kiện trước khi lấy type và code.
```

- o Thống kê số gói (packet\_count), byte (byte\_count), tốc độ (packet\_count\_per\_sec, v.v.)

```
packet_count = flow.packet_count
byte_count = flow.byte_count
packet_count_per_second = packet_count / duration_sec
packet_count_per_nsecond = packet_count / duration_nsec
byte_count_per_second = byte_count / duration_sec
byte_count_per_nsecond = byte_count / duration_nsec
# flow là cấu trúc chứa thông tin về luồng.
#duration_sec và duration_nsec được lấy từ flow.duration_sec và flow.duration_nsec.
```

Từ đó hình thành một vector đặc trưng đầu vào cho mô hình CNN:

```
features = [tp_src, tp_dst, proto, icmp_code, icmp_type, duration,
            ..., byte_count_per_nsecond]
```

Dự đoán bằng mô hình CNN:

```
label = self.predict(features)
```

- Dự đoán nhãn (0 = bình thường, 1 = tấn công).

```
label = np.argmax(output)
#Nếu mô hình dùng Softmax, đầu ra là mảng xác suất 2 chiều, ví dụ [0.95, 0.05].
#np.argmax() lấy chỉ số có xác suất cao nhất:
#0: luồng bình thường
#1: tấn công DDoS
```

- Dữ liệu được reshape thành tensor 3D đúng với yêu cầu của Conv1D.

```
input_data = np.array(features, dtype=np.float32).reshape(1, -1, 1)
#features: danh sách đặc trưng dạng 1 chiều, ví dụ [0.2, 0.1, 0.5, ...].
```

```
#Hàm reshape(1, -1, 1) chuyển thành tensor có shape (1, số đặc trưng, 1), là định dạng chuẩn cho mô hình Conv1D trong TensorFlow.
#1: số mẫu (batch size)
#-1: tự tính số đặc trưng
#1: số channel (giống ảnh đơn sắc)
```

### Phản ứng nếu phát hiện DDoS:

- Nếu nhận là 1 (tấn công), controller sẽ:

- Ghi log vào ddos\_log.txt

```
with open("ddos_log.txt", "a") as log_file:
    log_file.write(f"Detected DDoS from {src_ip} at {datetime.now()}\n")
#Mỗi lần phát hiện tấn công, IP bị nghi ngờ sẽ được ghi vào log kèm timestamp.
#Ghi ở chế độ "a" (append) để giữ lại các dòng trước đó.
```

- Cập nhật danh sách IP bị chặn (blocked\_ip.txt)

```
with open("blocked_ip.txt", "a") as blocked_file:
    blocked_file.write(f"{src_ip}\n")
#Sau khi dự đoán là tấn công, IP sẽ được thêm vào danh sách.
#Danh sách này có thể dùng để tham chiếu và tránh xử lý lại các flow từ cùng IP.
```

- Gửi email cảnh báo qua SMTP

```
with smtplib.SMTP('smtp.gmail.com', 587) as server:
    server.starttls()
    server.login(EMAIL_ADDRESS, EMAIL_PASSWORD)
    message = f"Subject: DDoS Alert\n\nDDoS attack detected from {src_ip}"
    server.sendmail(EMAIL_ADDRESS, RECEIVER_EMAIL, message)
#Dùng thư viện smtplib của Python để gửi email qua server SMTP của Gmail.
#Thông tin email như EMAIL_ADDRESS, EMAIL_PASSWORD, RECEIVER_EMAIL được load từ file .env.
```

File .env

```
EMAIL_ADDRESS = os.getenv("EMAIL_ADDRESS")
EMAIL_PASSWORD = os.getenv("EMAIL_PASSWORD")
RECEIVER_EMAIL = os.getenv("RECEIVER_EMAIL")
```

- Cài rule drop trên switch để chặn IP nguồn

```
match = parser.OFPMatch(eth_type=0x0800, ipv4_src=src_ip)
actions = [] # drop
```

### 2.5.3 Máy ảo (Virtual Machine)

Máy ảo (Virtual Machine – VM) là một công nghệ mô phỏng phần cứng, cho phép nhiều hệ điều hành và ứng dụng chạy đồng thời trên cùng một máy vật lý, hoạt

động như các hệ thống độc lập. Việc sử dụng máy ảo là một giải pháp phổ biến trong nghiên cứu và phát triển hệ thống mạng hiện đại vì nó giúp tiết kiệm tài nguyên, giảm chi phí phần cứng và dễ dàng thiết lập môi trường thử nghiệm.

Cốt lõi của máy ảo là phần mềm ảo hóa (hypervisor), có thể là VMware, VirtualBox hoặc KVM. Hypervisor phân chia tài nguyên phần cứng (CPU, RAM, disk, network) thành các phần hợp lý để phân phối cho từng máy ảo. Mỗi VM hoạt động như một máy tính độc lập, có thể cài đặt hệ điều hành riêng, kết nối mạng, lưu trữ file và chạy các ứng dụng như máy thật.

Trong lĩnh vực an ninh mạng và SDN, việc triển khai hệ thống trên máy ảo mang lại những lợi thế to lớn:

- Cho phép tái tạo lại các mô hình mạng phức tạp với chi phí thấp.
- Linh hoạt trong cấu hình mạng nội bộ, NAT, bridged hoặc mạng riêng ảo.
- Dễ dàng snapshot trạng thái hệ thống để phục hồi nếu có lỗi xảy ra.
- Hỗ trợ cô lập môi trường tấn công và môi trường phòng thủ nhằm tăng tính an toàn trong thử nghiệm.

Trong đề tài này, nhóm sử dụng hai máy ảo cài hệ điều hành Ubuntu 20.04 LTS để xây dựng mô hình SDN phòng chống DDoS. Việc tách biệt chức năng của từng VM giúp tăng tính mô-đun và dễ kiểm soát luồng xử lý:

- **Máy ảo thứ nhất – Mininet:**

- Được cấu hình để chạy công cụ mô phỏng mạng Mininet, nơi tạo topology mạng gồm nhiều host, switch và controller ảo.
- Tạo ra luồng lưu lượng mạng hỗn hợp bao gồm lưu lượng bình thường và tấn công (ICMP Flood, UDP Flood, TCP SYN Flood) từ các script mô phỏng.
- Mô phỏng môi trường mạng quy mô lớn (15 host) để kiểm thử khả năng nhận diện tấn công của hệ thống.
- Tận dụng công cụ hping3 để phát động các cuộc tấn công với cường độ cao nhằm kiểm tra giới hạn phản ứng của hệ thống.

- **Máy ảo thứ hai – Ryu Controller:**

- Cài đặt framework Ryu – đóng vai trò là controller trung tâm thu thập và xử lý dữ liệu mạng từ Mininet.
- Tích hợp mô hình học sâu CNN dưới dạng TensorFlow Lite để dự đoán và phân loại lưu lượng.
- Khi phát hiện tấn công, thực hiện các hành động phản ứng như cài đặt rule drop trên switch, gửi email cảnh báo qua SMTP, ghi log IP tấn công và cập nhật danh sách chặn.
- Tách biệt khỏi Mininet để đảm bảo controller không bị ảnh hưởng trực tiếp bởi lưu lượng tấn công – giúp mô phỏng một kiến trúc thực tế hơn.

**Ưu điểm của triển khai hệ thống trên hai máy ảo tách biệt:**

- **Hiệu quả nghiên cứu cao:** Dễ dàng tái lập các kịch bản tấn công, thay đổi topology hoặc mô hình mà không ảnh hưởng đến phần cứng thật.
- **Giảm thiểu rủi ro:** Cách ly giữa môi trường tấn công (Mininet) và xử lý (Ryu) giúp hệ thống không bị treo khi lưu lượng tăng đột biến.
- **Hỗ trợ phân tích và giám sát:** Có thể dễ dàng chạy các công cụ phân tích lưu lượng, hiển thị biểu đồ, theo dõi log từ hai môi trường riêng biệt.
- **Khả năng mở rộng tốt:** Có thể bổ sung thêm máy ảo để mô phỏng nhiều controller, hoặc thêm máy chạy IDS/IPS để mở rộng nghiên cứu.

Tổng kết lại, việc sử dụng máy ảo không chỉ giúp kiểm soát môi trường triển khai mà còn đảm bảo sự an toàn, linh hoạt và hiệu quả trong phát triển hệ thống phát hiện và phản ứng tấn công DDoS. Đây là một trong những thực tiễn tốt nhất trong nghiên cứu an ninh mạng hiện đại, đặc biệt phù hợp với bối cảnh sinh viên nghiên cứu và triển khai mô hình trong điều kiện không có thiết bị thật.

#### **2.5.4 Các mô hình học máy đã thử nghiệm**

Trong quá trình nghiên cứu và triển khai hệ thống phát hiện DDoS, nhóm đã thử nghiệm nhiều thuật toán học máy phổ biến nhằm đánh giá hiệu quả của từng mô hình đối với bài toán phân loại lưu lượng mạng. Dưới đây là các mô hình đã được triển khai,

huấn luyện và đánh giá bằng các chỉ số: Accuracy, Precision, Recall, F1-score và ROC AUC.<sup>3</sup>

Mô hình	Accuracy	Precision	Recall	F1-Score	ROC AUC
Logistic Regression	0.9606	0.9805	0.9947	0.9602	~0.96
Decision Tree	0.9996	0.9994	1.0000	0.9997	1.0000
KNN	0.9979	0.9975	1.0000	0.9976	0.9977
Naive Bayes	0.9347	0.9554	0.9224	0.9224	0.9615
Random Forest	0.9996	0.9994	1.0000	0.9997	1.0000
SVM	0.9462	0.9627	0.9193	0.9367	–
CNN	0.9740	0.9805	0.9615	0.9701	0.9898

Hình 4. Bảng chỉ số đánh giá các mô hình

**Random Forest** và **Decision Tree** cho kết quả gần như hoàn hảo trên tập validation, điều này cho thấy hai mô hình này học rất sát dữ liệu huấn luyện. Tuy nhiên, khả năng **overfitting** vẫn cần lưu ý nếu triển khai thực tế với dữ liệu thay đổi.

**KNN** cũng có độ chính xác rất cao nhưng mô hình này tính toán chậm trên tập dữ liệu lớn và nhạy cảm với nhiễu.

**Naive Bayes** và **SVM** có độ chính xác thấp hơn so với các mô hình khác, phù hợp với bài toán có ít đặc trưng hoặc giả định phân phối đơn giản.

Trong số các mô hình học máy được thử nghiệm, **mạng nơ-ron tích chập (CNN)** được lựa chọn là mô hình chính để triển khai vào hệ thống phát hiện DDoS vì nhiều lý do mang tính kỹ thuật lẫn thực tế triển khai, cụ thể như sau:

- Hiệu suất cao và ổn định: Mô hình CNN đạt Accuracy 97.4%, Precision 98.05%, Recall 96.15%, và F1-Score 97.01%. Các chỉ số này chứng tỏ mô hình vừa phát hiện tốt các cuộc tấn công (Recall cao), vừa hạn chế cảnh báo sai (Precision cao). Mặc dù một số mô hình khác như Random Forest

<sup>3</sup> Nhóm đã thực hiện trên Google Colab:

[https://colab.research.google.com/drive/1zB\\_OQwFyofI86w5i65Tqyv9Xxf7XSfeJ#scrollTo=wCXORmH\\_kFwG](https://colab.research.google.com/drive/1zB_OQwFyofI86w5i65Tqyv9Xxf7XSfeJ#scrollTo=wCXORmH_kFwG)

đạt gần 100%, nhưng CNN cho thấy hiệu năng ổn định hơn khi triển khai thực tế, đặc biệt với dữ liệu chưa từng thấy (unseen traffic).

- Tích hợp tốt vào controller SDN với TensorFlow Lite: Khác với các mô hình học máy truyền thống vốn nặng và khó tích hợp, CNN sau khi được huấn luyện có thể chuyển sang định dạng nhẹ TensorFlow Lite. Việc này cho phép mô hình chạy trực tiếp trong Ryu Controller, trên máy ảo thông thường mà không cần GPU hoặc server mạnh. Điều này rất quan trọng khi thiết kế hệ thống phòng thủ DDoS thời gian thực và gọn nhẹ.
- Tự động học đặc trưng và giảm phụ thuộc vào chuyên gia: CNN có khả năng trích xuất đặc trưng tự động từ dữ liệu thay vì phụ thuộc vào việc thiết kế đặc trưng thủ công (feature engineering). Điều này giúp giảm gánh nặng cho người triển khai và tăng khả năng phát hiện các mẫu tấn công phi tuyến hoặc phức tạp mà các mô hình tuyến tính (như Logistic Regression hoặc SVM) khó xử lý.
- Khả năng thích nghi với các hình thức tấn công mới: Do DDoS không ngừng thay đổi về chiến thuật và kỹ thuật (multi-vector, slow-rate, ứng dụng IoT,...), các mô hình truyền thống thường không theo kịp biến thể mới. CNN, với cấu trúc học sâu và nhiều tầng lọc, có thể học và tổng quát hóa các mẫu dữ liệu mới, từ đó thích nghi với các tấn công phức tạp trong tương lai.
- Cân bằng giữa hiệu năng và chi phí: So với các mô hình cực mạnh như ensemble (Random Forest), CNN có ưu thế về khả năng rút gọn mô hình và tối ưu hóa trên môi trường có tài nguyên giới hạn (máy ảo Ryu). Việc sử dụng CNN cũng giúp giảm độ trễ khi phản ứng và duy trì tính thời gian thực của hệ thống.

### 2.5.5 Mạng nơ-ron tích chập (CNN)

Mạng nơ-ron tích chập (CNN) là một mô hình học sâu được thiết kế đặc biệt để xử lý dữ liệu có tính thứ tự hoặc cấu trúc lưới, phổ biến nhất là hình ảnh và chuỗi dữ



liệu. CNN là một phần của họ mạng nơ-ron nhân tạo và được ứng dụng rộng rãi trong thị giác máy tính, nhận dạng giọng nói, xử lý ngôn ngữ tự nhiên và an ninh mạng.

Cấu trúc của một mạng CNN bao gồm các lớp cơ bản như:

- **Lớp tích chập (Convolutional Layer):** Dùng để trích xuất các đặc trưng từ dữ liệu đầu vào thông qua các bộ lọc (filter).
- **Lớp kích hoạt (Activation Layer):** Thường sử dụng hàm ReLU để tăng tính phi tuyến.
- **Lớp gộp (Pooling Layer):** Giảm kích thước dữ liệu, tăng tốc độ tính toán và giảm nguy cơ quá khớp (overfitting).
- **Lớp kết nối đầy đủ (Fully Connected Layer):** Tổng hợp các đặc trưng và đưa ra kết quả phân loại.

CNN có khả năng học các đặc trưng quan trọng trực tiếp từ dữ liệu đầu vào mà không cần thiết kế thủ công, giúp mô hình linh hoạt và thích ứng tốt với các bài toán phức tạp. Khi áp dụng vào lĩnh vực phát hiện tấn công mạng, CNN có thể phân tích các chuỗi đặc trưng lưu lượng mạng và nhận diện các mẫu bất thường như tấn công DDoS.

Do khả năng trích xuất đặc trưng mạnh mẽ, cùng với việc có thể tối ưu hóa để chạy trên các thiết bị hạn chế tài nguyên (như controller Ryu), CNN được xem là một trong những mô hình hiệu quả cho hệ thống phát hiện tấn công trong môi trường mạng SDN.

### 2.5.6 TensorFlow Lite

**TensorFlow Lite** là phiên bản rút gọn của thư viện học sâu TensorFlow, được thiết kế tối ưu cho các thiết bị có tài nguyên hạn chế như thiết bị nhúng, điện thoại di động, hoặc các hệ thống mạng nhẹ như SDN Controller.

Các đặc điểm chính:

- **Nhẹ và nhanh:** TFLite sử dụng định dạng mô hình tối ưu hóa (.tflite), nhỏ hơn và suy luận nhanh hơn.
- **Dễ tích hợp:** Có thể dễ dàng nhúng vào các ứng dụng Python, C++, hoặc nhúng trực tiếp trong controller như Ryu.

- **Không cần GPU:** Chạy hiệu quả trên CPU, phù hợp với máy ảo hoặc hệ thống không có phần cứng mạnh.

Trong đề tài này, sau khi mô hình CNN được huấn luyện trên Google Colab, nó được chuyển đổi sang định dạng .tflite và nạp trực tiếp vào Controller SDN. Việc sử dụng TensorFlow Lite giúp đảm bảo quá trình phát hiện tấn công **diễn ra theo thời gian thực** mà không ảnh hưởng đến hiệu năng controller.

### **CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG PHÁT HIỆN TẤN CÔNG DDOS SỬ DỤNG HỌC MÁY TRÊN KIẾN TRÚC MẠNG SDN**

Chương 3 trình bày quá trình phân tích và thiết kế hệ thống phát hiện tấn công DDoS trong môi trường mạng SDN bằng cách tích hợp mô hình học sâu CNN. Nội dung bao gồm kiến trúc ba lớp của hệ thống, phương pháp thu thập và xử lý dữ liệu từ các switch, thiết kế mô hình học máy, cũng như cơ chế phát hiện và phản ứng tự động tại controller. Hệ thống được xây dựng trên nền tảng Mininet và Ryu Controller, với khả năng hoạt động gần như thời gian thực, hỗ trợ giám sát trực quan và dễ dàng mở rộng.

#### **3.1 Tổng quan về hệ thống đề xuất**

Hệ thống được xây dựng nhằm phát hiện và ngăn chặn tấn công DDoS trong môi trường mạng SDN thông qua việc tích hợp mô hình học máy. Mục tiêu là tận dụng tính linh hoạt và khả năng kiểm soát tập trung của SDN kết hợp với sức mạnh phân tích và phân loại của mô hình học máy để phát hiện kịp thời các hành vi tấn công, từ đó phản ứng nhanh và chính xác nhằm giảm thiểu thiệt hại.

Kiến trúc hệ thống gồm ba thành phần cốt lõi: thu thập dữ liệu mạng theo thời gian thực từ các switch, phân tích đặc trưng bằng mô hình học máy CNN, và triển khai biện pháp ngăn chặn ngay tại controller SDN. Hệ thống sử dụng Mininet để mô phỏng mạng và Ryu Controller làm trung tâm điều phối, phân tích và phản ứng.

### 3.2 Phân tích yêu cầu hệ thống

#### **Yêu cầu chức năng**

Hệ thống được thiết kế với các chức năng cốt lõi như sau:

- **Thu thập dữ liệu lưu lượng (flow)** từ các switch SDN: Thông qua giao thức OpenFlow, controller có thể truy vấn các chỉ số như IP nguồn, IP đích, port, số lượng gói tin, số byte, thời lượng luồng,... Đây là cơ sở để trích xuất đặc trưng và phân tích hành vi mạng.
- **Trích xuất đặc trưng và phân loại bằng học máy:** Dữ liệu thu thập được xử lý và chuyển thành định dạng đặc trưng đầu vào cho mô hình học máy (CNN). Mô hình sẽ phân tích và phân loại lưu lượng thành “bình thường” hoặc “tấn công DDoS” một cách tự động.
- **Phản ứng tự động khi phát hiện tấn công:**
  - Ghi log sự kiện tấn công (IP nguồn, IP đích, thời gian).
  - Gửi email cảnh báo đến quản trị viên hệ thống.
  - Chặn địa chỉ IP nguồn tấn công bằng cách thiết lập rule drop trên switch.
  - Tùy chọn điều chỉnh tuyến hoặc giới hạn băng thông đối với các flow nghi vấn.

Hệ thống vận hành độc lập, không can thiệp thủ công trong quá trình phát hiện và phản ứng.

#### **Yêu cầu phi chức năng**

Bên cạnh chức năng chính, hệ thống cần đáp ứng các tiêu chí phi chức năng quan trọng để đảm bảo tính khả thi và hiệu quả khi triển khai thực tế:

- **Phản ứng nhanh, gần như thời gian thực:** Từ khi phát hiện đến khi hành động chỉ mất vài mili-giây, nhờ khả năng xử lý trực tiếp tại controller. Điều này giúp giảm thiểu tối đa thời gian tồn tại của các luồng tấn công.

- **Khả năng chịu tải và ổn định với lưu lượng lớn:** Hệ thống cần hoạt động bền bỉ trong các mạng có số lượng flow lớn và đa dạng, đảm bảo không gây ảnh hưởng đến hiệu năng chung của controller hay switch.
- **Tính mở rộng và bảo trì tốt:** Dễ dàng cập nhật mô hình học máy khi có dữ liệu mới. Có thể tái huấn luyện và nạp lại mô hình CNN mà không cần dừng toàn bộ hệ thống.
- **Giao diện giám sát trực quan theo thời gian thực:** Thông qua mô-đun `ddos_monitor.py`, quản trị viên có thể theo dõi trực tiếp các luồng bị phát hiện là tấn công, tần suất và lịch sử phát hiện. Điều này giúp dễ dàng đánh giá mức độ nghiêm trọng và thực hiện phân tích hậu kiểm.
- **Khả năng tùy biến chính sách mạng:** Quản trị viên có thể cấu hình thêm hành động ứng phó như redirect lưu lượng, giới hạn băng thông, ưu tiên dịch vụ,... chỉ với vài dòng mã Python tại controller, nhờ kiến trúc mở và lập trình được của SDN.

### 3.3 Kiến trúc hệ thống đề xuất

Hệ thống tuân theo kiến trúc ba lớp SDN:

- **Lớp dữ liệu (Data Plane):** Gồm các switch hỗ trợ OpenFlow, chịu trách nhiệm chuyển tiếp gói tin và gửi số liệu thống kê lưu lượng (flow stats) về controller.
- **Lớp điều khiển (Control Plane):** Ryu Controller đảm nhiệm việc thu thập dữ liệu, giao tiếp với switch, và triển khai chính sách mạng.
- **Lớp ứng dụng (Application Plane):** Mô hình CNN được tích hợp trực tiếp tại controller, chịu trách nhiệm phân tích đặc trưng và phân loại luồng (bình thường hoặc tấn công).

Mô hình CNN sử dụng định dạng TensorFlow Lite giúp giảm tải cho controller, phù hợp với môi trường thực thi nhẹ.

### 3.4 Thu thập và xử lý dữ liệu

Dữ liệu được thu thập từ các switch bao gồm: IP nguồn, IP đích, cổng, giao thức, số gói, số byte, thời lượng flow, v.v. Sau đó, dữ liệu sẽ được chuẩn hóa và chuyển sang định dạng đầu vào của mô hình CNN.

Quá trình xử lý dữ liệu bao gồm:

- Loại bỏ bản ghi lỗi hoặc không hợp lệ.
- Trích xuất đặc trưng thống kê thể hiện hành vi lưu lượng.
- Giảm chiều dữ liệu nếu cần để tối ưu hiệu suất.
- Đảm bảo dữ liệu phù hợp với input của mô hình đã huấn luyện trước.

### 3.5 Thiết kế mô hình học máy

Hệ thống thử nghiệm nhiều thuật toán: Decision Tree, Random Forest, SVM, DNN, CNN,... Kết quả đánh giá cho thấy mô hình CNN cho độ chính xác cao và khả năng khái quát tốt nhất trong môi trường SDN.

CNN được huấn luyện từ tập dữ liệu chuẩn (CICDDoS2019) và chuyển đổi sang TensorFlow Lite để chạy hiệu quả trong controller. Mô hình sử dụng các đặc trưng như: tần suất gói, số byte, thời gian sống luồng, loại giao thức,... để phân loại hành vi lưu lượng.

Ngoài ra, hệ thống hỗ trợ cập nhật mô hình định kỳ hoặc tự động theo ngưỡng phát hiện sai, đảm bảo khả năng thích nghi với các kiểu tấn công mới.

### 3.6 Cơ chế phát hiện và phản ứng tấn công

Khi mô hình CNN phát hiện một flow có dấu hiệu là tấn công, controller sẽ:

- Ghi lại sự kiện vào log kèm thời gian, IP nguồn – đích.
- Gửi email cảnh báo đến quản trị viên (cấu hình trong file .env).
- Tự động chặn IP bằng cách thêm rule drop vào switch.
- Cập nhật bảng điều khiển để tối ưu băng thông và giảm tải tài nguyên mạng.

Mô-đun Monitor liên tục đọc log và hiển thị trực quan theo thời gian thực các nguồn tấn công, giúp quản trị viên có thể theo dõi và đánh giá hiệu quả hệ thống.

Cơ chế phản ứng gần như tức thời và hoàn toàn tự động, giúp đảm bảo sự ổn định và an toàn của hệ thống mạng SDN trước các mối đe dọa tấn công từ chối dịch vụ phân tán.

## CHƯƠNG 4: THỰC NGHIỆM

Chương 4 trình bày quá trình triển khai và thực nghiệm hệ thống phát hiện tấn công DDoS đã thiết kế. Hệ thống được mô phỏng trên hai máy ảo: một chạy Mininet để sinh lưu lượng và tấn công, máy còn lại chạy Ryu Controller tích hợp mô hình CNN dưới dạng TensorFlow Lite. Chương này mô tả chi tiết cách thu thập dữ liệu, huấn luyện mô hình, đánh giá hiệu quả các thuật toán học máy, cũng như kiểm thử khả năng phát hiện và phản ứng trong môi trường thực tế. Kết quả cho thấy mô hình CNN đạt độ chính xác cao và phản ứng nhanh, đáp ứng tốt yêu cầu phát hiện tấn công DDoS trong mạng SDN.

### 4.1 Dữ liệu thực nghiệm

Tập dữ liệu được sử dụng trong đề tài được tham khảo và trích xuất từ một dự án mã nguồn mở trên nền tảng GitHub <sup>[1]</sup>, với mục tiêu mô phỏng các kịch bản tấn công DDoS trong mạng SDN. Dữ liệu được trình bày dưới dạng flow-based, bao gồm cả lưu lượng bình thường và các cuộc tấn công phổ biến như TCP SYN Flood, UDP Flood và ICMP Flood.

Sau khi tải về, dữ liệu được xử lý lại, chuẩn hóa và chia thành hai tập: huấn luyện và kiểm thử để phục vụ quá trình huấn luyện mô hình học máy.

- Địa chỉ IP nguồn, đích (ip\_src, ip\_dst)
- Cổng nguồn, đích (tp\_src, tp\_dst)
- Giao thức (ip\_proto, icmp\_code, icmp\_type)
- Thời lượng luồng, số gói, số byte, tốc độ truyền v.v.

Các luồng được gán nhãn là **“0” (bình thường)** hoặc **“1” (tấn công)** để huấn luyện mô hình học máy.

## 4.2 Tiền xử lý dữ liệu

### 4.2.1 Tiền xử lý dữ liệu cho mô hình truyền thống

Dữ liệu thô sau khi thu thập được xử lý như sau:

- **Loại bỏ các trường dư thừa:** Các cột như flow\_id, ip\_src, ip\_dst không cung cấp thông tin cần thiết hoặc gây trùng lặp được loại bỏ để tránh nhiễu dữ liệu.

```
df = df.drop(['flow_id', 'ip_src', 'ip_dst'], axis=1)
```

- **Chuẩn hóa dữ liệu:** Tất cả các đặc trưng số được chuẩn hóa về cùng một thang giá trị [0,1] bằng kỹ thuật MinMaxScaler, giúp tăng tốc độ huấn luyện và độ chính xác của mô hình.

```
scaler = MinMaxScaler()
scaled_array = scaler.fit_transform(df.drop('label', axis=1))
features_scaled_df = pd.DataFrame(scaled_array, columns=df.columns[:-1])
features_scaled_df['label'] = df['label']
```

- **Tách nhãn đầu ra (label) và chia dữ liệu thành 80% huấn luyện, 20% kiểm thử.**

```
X_train, X_test, y_train, y_test = train_test_split(
    features_scaled_df.drop('label', axis=1),
    features_scaled_df['label'],
    test_size=0.2,
    random_state=42
)
```

Toàn bộ thao tác thực hiện bằng Python trong notebook Clean\_ML.ipynb sử dụng Pandas, NumPy, Scikit-learn.

### 4.2.2 Tiền xử lý cho mô hình CNN

Trong hệ thống phát hiện DDoS đề xuất, nhóm thực hiện sử dụng mô hình học máy thuộc loại mạng nơ-ron tích chập một chiều (1D Convolutional Neural Network – CNN) để phân tích các đặc trưng của luồng mạng (flow features). Mô hình CNN được lựa chọn vì khả năng xử lý hiệu quả chuỗi dữ liệu theo chiều thời gian và trích xuất được các đặc trưng quan trọng từ dữ liệu đầu vào.

Tập dữ liệu ban đầu được chọn lọc 15 đặc trưng phù hợp, bao gồm các thông số liên quan đến cổng, giao thức, thời gian sống, số lượng gói, số byte và tốc độ truyền. Các đặc trưng này được chuẩn hóa bằng StandardScaler để tăng độ ổn định và hiệu quả học của mô hình.

```
features = [
    'tp_src', 'tp_dst', 'ip_proto', 'icmp_code', 'icmp_type',
    'flow_duration_sec', 'flow_duration_nsec',
    'idle_timeout', 'hard_timeout', 'flags',
    'packet_count', 'byte_count',
    'packet_count_per_second', 'byte_count_per_second',
    'byte_count_per_nsecond'
]

X = df[features].values.astype(np.float32)
y = df['label'].values.astype(int)

# Chuẩn hóa
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_scaled = X_scaled.reshape(-1, 15, 1) # định dạng phù hợp cho CNN

# One-hot encoding cho nhãn
y_cat = to_categorical(y, num_classes=2)

# Chia tập huấn luyện / kiểm thử
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y_cat, test_size=0.2, random_state=42
)
```

### 4.3 Huấn luyện và triển khai mô hình CNN

#### 4.3.1 Kiến trúc mô hình CNN

Mô hình được xây dựng gồm:

- Một lớp Convolutional 1D với 32 bộ lọc và kích thước kernel là 3.
- Một lớp MaxPooling để giảm chiều và nhiễu.



- Một lớp Flatten để chuyển đổi sang vector.
- Một lớp Dense ẩn gồm 64 node với hàm kích hoạt ReLU.
- Một lớp Dropout để tránh overfitting.
- Lớp đầu ra với 2 node (tần công/bình thường) và hàm softmax.

```
model = Sequential()
model.add(Conv1D(32, kernel_size=3, activation='relu',
input_shape=(15,1)))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(2, activation='softmax'))

model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
model.summary()
```

#### 4.3.2 Huấn luyện và chuyển đổi mô hình

Mô hình được huấn luyện trong 10 epochs với batch size là 32. Sau khi huấn luyện, mô hình được lưu và chuyển sang định dạng .tflite nhẹ hơn, giúp dễ dàng tích hợp vào môi trường thật (controller).

```
# Huấn luyện mô hình
model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.1)
# Lưu và chuyển sang TensorFlow Lite
model.save("cnn_ddos_model.h5")

converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

with open("cnn_ddos_model.tflite", "wb") as f:
    f.write(tflite_model)
```

#### 4.4 Tích hợp mô hình vào controller và cơ chế phát hiện phản ứng

Tập tin controller.py là trái tim của hệ thống SDN sử dụng Ryu Controller để phát hiện và phản ứng với tấn công DDoS trong thời gian thực. File này bao gồm các thành phần chính sau:

##### 4.4.1 Khởi tạo mô hình học máy

Khi controller được khởi động, mô hình học máy dạng TensorFlow Lite (.tflite) đã huấn luyện sẵn sẽ được nạp vào bộ nhớ:

```
self.interpreter = tf.lite.Interpreter(model_path="cnn_ddos_model.tflite")
self.interpreter.allocate_tensors()
```

Đầu vào/ra của mô hình được định vị:

```
self.input_details = self.interpreter.get_input_details()
self.output_details = self.interpreter.get_output_details()
```

Việc sử dụng TensorFlow Lite giúp giảm tải tài nguyên, phù hợp với môi trường thực thi trực tiếp trong controller mà không cần GPU.

##### 4.4.2 Nhận và xử lý luồng mạng (PacketIn)

Khi switch không có rule xử lý gói, nó gửi sự kiện PacketIn đến controller. Tại đây, controller sẽ trích xuất thông tin flow:

```
ip_src = pkt_ipv4.src
ip_dst = pkt_ipv4.dst
proto = pkt_ipv4.proto
```

Tiếp theo, controller dùng lệnh ofproto\_parser.OFPFlowStatsRequest() để yêu cầu switch gửi thống kê flow, bao gồm số gói tin, byte, thời gian tồn tại v.v.

##### 4.4.3 Môi trường triển khai, công cụ sử dụng.

Khi đã có đủ thông tin flow, controller chuẩn hóa dữ liệu đầu vào và dự đoán bằng mô hình học máy:

```
sample = np.array([prepared_values], dtype=np.float32).reshape(1, 14, 1)
self.interpreter.set_tensor(input_index, sample)
self.interpreter.invoke()
output_data = self.interpreter.get_tensor(output_index)
```

Kết quả phân loại là một xác suất. Nếu xác suất luồng là tấn công cao hơn ngưỡng (ví dụ 0.5), controller sẽ coi đó là **DDoS**.

#### 4.4.4 Hiển thị kết quả lên terminal

```
print(f"[DEBUG] Predict: {output_data} -> {label}")
print(f"[INFO] DDoS detected from {ip_src} to {ip_dst}")
print("BLOCK")
```

Giao diện dòng lệnh giúp người quản trị theo dõi phản ứng hệ thống theo thời gian thực.

#### 4.4.5 Hành động phản ứng

Khi phát hiện một luồng bất thường, controller sẽ:

- Ghi log vào file `ddos_log.txt`:

```
with open("ddos_log.txt", "a") as f:
    f.write(f"{timestamp} {ip_src} -> {ip_dst}\n")
```

- Lưu IP vào `blocked_ip.txt` nếu bị chặn:

```
with open("blocked_ip.txt", "a") as f:
    f.write(ip_src + "\n")
```

- Tạo rule chặn lưu lượng tại switch:

```
match = parser.OFPMatch(eth_type=0x0800, ipv4_src=ip_src)
actions = [] # Không có action -> drop
mod = parser.OFPFlowMod(...)
datapath.send_msg(mod)
```

- Gửi cảnh báo email nếu bật cấu hình SMTP trong `.env`:

```
send_email_alert(ip_src, ip_dst)
```

#### 4.4.6 Giao tiếp với mô-đun giám sát

Mỗi khi có luồng bị chặn, thông tin sẽ được ghi lại vào log. Mô-đun `ddos_monitor.py` đọc file `ddos_log.txt` và trực quan hóa số lượng IP tấn công theo thời gian thực (bằng `matplotlib`), giúp quản trị viên theo dõi và đánh giá nhanh hiệu quả phát hiện.

### 4.5 Công cụ sử dụng và cài đặt môi trường triển khai

Thành phần	Mô tả chi tiết
<b>Hệ điều hành</b>	Ubuntu 20.04 (chạy trong máy ảo VMware)
<b>Mạng mô phỏng</b>	Mininet (192.168.186.129/24)
<b>Bộ điều khiển SDN</b>	Ryu Controller (192.168.186.130/24)
<b>Mô hình học máy</b>	CNN (huấn luyện bằng TensorFlow, triển khai tại controller bằng TensorFlow Lite .tflite)
<b>Thư viện sử dụng</b>	Python 3.x
<b>Ngôn ngữ lập trình</b>	NumPy, Pandas, Scikit-learn, TensorFlow, tflite-runtime
<b>Mô-đun giám sát</b>	ddos_monitor.py – theo dõi log tấn công theo thời gian thực từ ddos_log.txt
<b>File mô hình tích hợp</b>	cnn_ddos_model.tflite
<b>Cảnh báo qua email</b>	Gửi tự động bằng SMTP từ thông tin .env khi phát hiện DDoS

*Bảng 1: Công cụ sử dụng*

#### 4.5.1 Cài đặt môi trường triển khai

Để xây dựng và thử nghiệm hệ thống phát hiện tấn công DDoS trên mạng SDN, môi trường triển khai được thiết lập trên phần mềm ảo hóa **VMware** với hai máy ảo chính:

- **Máy ảo Mininet:** Mô phỏng các thành phần mạng SDN như switch, host, và tạo các luồng lưu lượng bao gồm cả tấn công và bình thường.

**Link tải:** <https://github.com/mininet/mininet/releases/>

- **Máy ảo Ryu Controller:** Chạy bộ điều khiển SDN Ryu, đồng thời tích hợp mô hình học máy CNN để thực hiện phát hiện và phản ứng với tấn công.

Cài đặt hệ điều hành và công cụ cần thiết

- **Phần mềm ảo hóa:** VMware Workstation / VMware Player
- **Hệ điều hành:** Ubuntu 20.04 LTS (khuyến nghị)
- **Thông số đề xuất:**
  - CPU: 2 nhân trở lên
  - RAM: 2–4 GB

- Ổ cứng: 20 GB
- Kết nối mạng: NAT hoặc Bridged

## Cài đặt Mininet trên máy ảo

### Bước 1 – Cập nhật hệ thống:

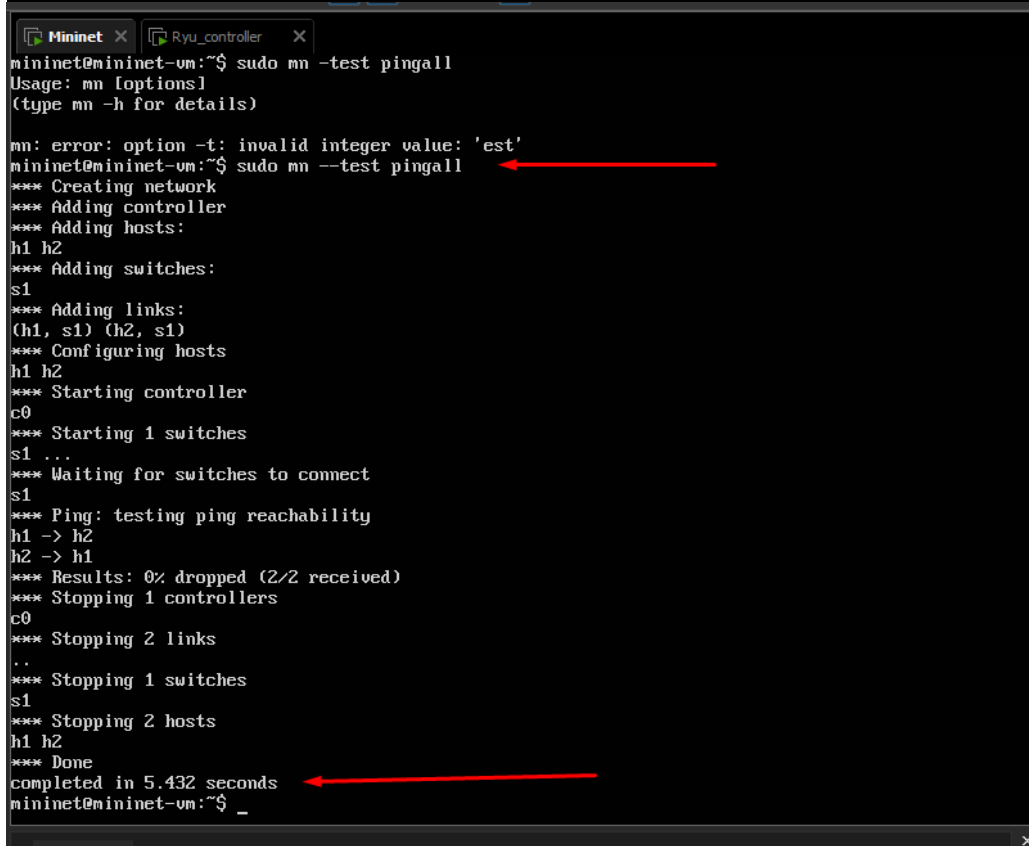
```
sudo apt update && sudo apt upgrade -y
```

### Bước 2 – Cài Mininet từ mã nguồn:

```
git clone https://github.com/mininet/mininet.git
cd mininet
sudo ./util/install.sh -a
```

### Kiểm tra hoạt động Mininet:

```
sudo mn --test pingall
```



```

Mininet x Ryu_controller x
mininet@mininet-vm:~$ sudo mn --test pingall
Usage: mn [options]
(type mn -h for details)

mn: error: option -t: invalid integer value: 'est'
mininet@mininet-vm:~$ sudo mn --test pingall
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Waiting for switches to connect
s1
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 5.432 seconds
mininet@mininet-vm:~$ _

```

Hình 5. Test pingall

```
/home/mininet/
```

```
└─ attack_launcher.py
```

```
# Tập tin thực thi script mô phỏng tấn công DDoS
```

Hình 6. Cấu trúc thư mục triển khai hệ thống máy ảo Mininet

Ở đây nhóm có thêm vào file `attack_launcher.py`. Tập tin này là một **script mô phỏng tấn công DDoS hàng loạt trong Mininet**, dùng kết hợp thư viện Python mininet để khởi tạo mạng, tạo host, và phát động tấn công ICMP/UDP/SYN Flood từ nhiều host vào cùng một mục tiêu.

### Cấu hình chính của script

- **Tổng số host:** 15 host
- **Host bị tấn công (mục tiêu):** h1
- **Các host tấn công (attacker):** từ h2 → h15
- **Controller kết nối:** Ryu Controller tại địa chỉ IP 192.168.186.130
- **Công cụ tấn công:** hping3, sử dụng ba loại tấn công:
  - TCP SYN Flood
  - UDP Flood `--flood --udp -p 53`
  - ICMP Flood `--icmp --flood`

```
for i in range(0, len(attackers), batch_size):
    batch = attackers[i:i+batch_size]
    for idx, h in enumerate(batch):
        if idx % 3 == 0:
            h.cmd(f'hping3 --flood -S -p 80 {target.IP()} &')
        elif idx % 3 == 1:
            h.cmd(f'hping3 --flood --udp -p 53 {target.IP()} &')
        else:
            h.cmd(f'hping3 --icmp --flood {target.IP()} &')
    info(f"Started batch {i//batch_size + 1}\n")
    sleep(5)
```

### Bước 3 – Cài đặt máy ảo Ryu Controller

Tạo máy ảo thứ 2 có tên là `Ryu_controller` được clone từ máy ảo Mininet sau đó cài đặt Ryu Controller

```
sudo apt install python3-ryu -y
```

```

/home/ryu/
├─ controller.py           # File chính điều khiển SDN, tích hợp mô hình CNN
├─ cnn_ddos_model.tflite   # Mô hình học máy đã huấn luyện, định dạng TensorFlow Lite
├─ ddos_monitor.py        # Giao diện giám sát lưu lượng tấn công thời gian thực
├─ ddos_log.txt            # Ghi log các luồng nghi ngờ là DDoS
├─ blocked_ip.txt         # Danh sách các IP đã bị chặn để tránh xử lý lại
└─ .env                   # File cấu hình SMTP (ẩn, nhưng nên tồn tại)

```

Hình 7. Cấu trúc thư mục triển khai hệ thống máy ảo Controller

#### Bước 4 - Cấu hình kết nối giữa hai máy ảo

Vì Mininet và Ryu được cài đặt trên **hai máy ảo tách biệt**, cần đảm bảo:

- Cả hai máy ảo **kết nối vào cùng một mạng** (có thể là NAT hoặc Bridged trong VMware).
- Xác định địa chỉ IP của máy ảo chạy controller bằng:

```

ip a # hoặc: ifconfig

ryu@ryu-controller-vm:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.186.130 netmask 255.255.255.0 broadcast 192.168.186.255
    ether 00:50:56:3f:a8:ce txqueuelen 1000 (Ethernet)
    RX packets 69939 bytes 54319469 (54.3 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 66105 bytes 8862948 (8.8 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Hình 8. Địa chỉ IP của máy ảo Ryu\_controller

```

mininet@mininet-vm:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.186.129 netmask 255.255.255.0 broadcast 192.168.186.255
    ether 00:50:56:27:aa:83 txqueuelen 1000 (Ethernet)
    RX packets 78158 bytes 12035452 (12.0 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 56077 bytes 13402187 (13.4 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 1377 bytes 131520 (131.5 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1377 bytes 131520 (131.5 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Hình 9. Địa chỉ IP của máy ảo mininet

#### 4.6 Triển khai thử nghiệm và demo thực tế

Trên **máy ảo Ryu**, di chuyển đến thư mục chứa mã nguồn, sau đó khởi chạy controller:

```
cd ~/ryu-ddos/
```

```
ryu-manager controller.py
```

Sau khi chạy thì CLI Ryu\_controller sẽ xuất hiện:

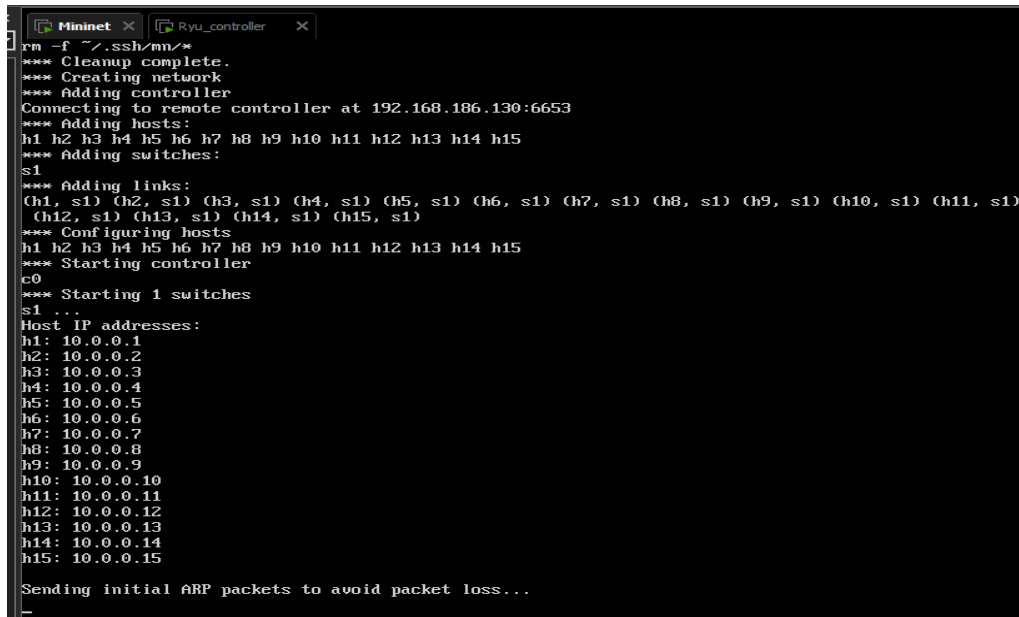
```
ryu@ryu-controller-vm:~$ ryu-manager controller.py
loading app controller.py
loading app ryu.controller.ofp_handler
instantiating app controller.py of DDoSCNNController
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
instantiating app ryu.controller.ofp_handler of OFPHandler
```

Hình 10. Sau khi chạy *ryu-manager controller.py*

Trên máy ảo Mininet, sau khi controller đã được khởi động và kết nối thành công, tiến hành khởi chạy mô phỏng mạng và thực hiện tấn công từ các host bằng script *attack\_launcher.py*.

```
sudo python3 attack_launcher.py
```

Sau khi script chạy, terminal sẽ hiển thị trạng thái khởi tạo từng đợt tấn công:



```
rm -f ~/.ssh/nn/*
*** Cleanup complete.
*** Creating network
*** Adding controller
Connecting to remote controller at 192.168.186.130:6653
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1) (h6, s1) (h7, s1) (h8, s1) (h9, s1) (h10, s1) (h11, s1)
(h12, s1) (h13, s1) (h14, s1) (h15, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15
*** Starting controller
c0
*** Starting 1 switches
s1 ...
Host IP addresses:
h1: 10.0.0.1
h2: 10.0.0.2
h3: 10.0.0.3
h4: 10.0.0.4
h5: 10.0.0.5
h6: 10.0.0.6
h7: 10.0.0.7
h8: 10.0.0.8
h9: 10.0.0.9
h10: 10.0.0.10
h11: 10.0.0.11
h12: 10.0.0.12
h13: 10.0.0.13
h14: 10.0.0.14
h15: 10.0.0.15
Sending initial ARP packets to avoid packet loss...
```

Hình 11. Khởi chạy script *attack\_launcher.py*

Tập lệnh (script) sẽ tự động tạo controller, thêm các host h1 đến h15, liên kết chúng qua switch s1, và gán địa chỉ IP cho từng host. Sau đó, Mininet bắt đầu gửi các gói ARP để tránh mất gói ban đầu. Đây là bước khởi đầu quan trọng nhằm chuẩn bị môi trường mạng SDN sẵn sàng cho các kịch bản tấn công và thử nghiệm mô hình phát hiện DDoS.



Controller sẽ liên tục nhận các PacketIn từ switch, phân tích bằng mô hình CNN và phản ứng nếu phát hiện tấn công.

Hình 12. Phản ứng nếu phát hiện tấn công

Ngoài ra:

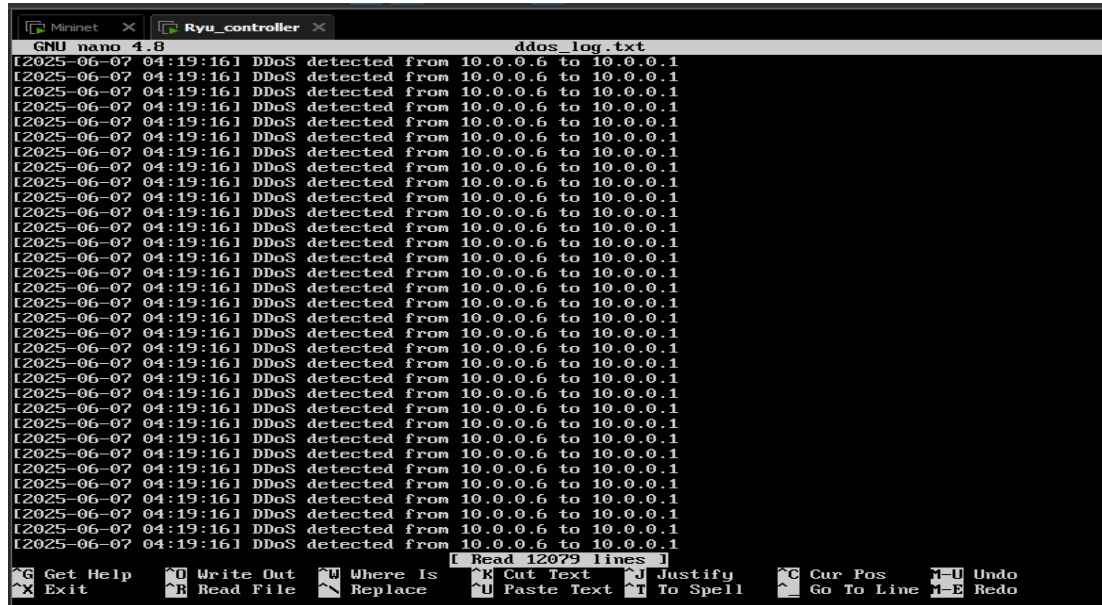
- IP tấn công sẽ được ghi vào blocked\_ip.txt

Hình 12. IP tấn công sẽ được ghi vào file .txt

Danh sách các địa chỉ IP được mô hình CNN xác định là có hành vi tấn công DDoS sẽ tự động được lưu vào file blocked\_ip.txt. File này giúp controller tránh xử lý

lập lại các IP đã xác định là nguy hiểm và có thể dùng làm nguồn dữ liệu để phân tích hậu kiểm hoặc huấn luyện mô hình ở các lần sau. Danh sách các địa chỉ IP được mô hình CNN xác định là có hành vi tấn công DDoS sẽ tự động được lưu vào file `blocked_ip.txt`. File này giúp controller tránh xử lý lập lại các IP đã xác định là nguy hiểm và có thể dùng làm nguồn dữ liệu để phân tích hậu kiểm hoặc huấn luyện mô hình ở các lần sau.

- Thông tin log sẽ được lưu trong ddos\_log.txt



Hình 13. Thông tin log

- Email cảnh báo sẽ được gửi đến quản trị viên.



Hình 14. Email cảnh báo được gửi đến quản trị viên

Bên cạnh chức năng phát hiện và phản ứng tấn công DDoS của controller, hệ thống còn tích hợp mô-đun giám sát thời gian thực `ddos_monitor.py`. Mô-đun này được xây dựng nhằm hỗ trợ quản trị viên **quan sát trực quan các luồng tấn công bị phát hiện**, giúp đánh giá hiệu quả mô hình CNN.

## Chức năng chính

- **Đọc dữ liệu file ddos\_log.txt** – nơi controller ghi lại các IP nguồn tấn công đã phát hiện.
- **Thống kê số lượng lượt tấn công theo IP hoặc thời gian.**
- **Hiển thị biểu đồ thời gian thực** bằng thư viện matplotlib để người dùng dễ theo dõi.

```
while True:
    with open("ddos_log.txt") as f:
        lines = f.readlines()
    ip_list = [line.split()[1] for line in lines] # lấy IP nguồn
    count = Counter(ip_list)
    plt.clf()
    plt.bar(count.keys(), count.values())
    plt.title("DDoS IPs Detected")
    plt.xlabel("Source IP")
    plt.ylabel("Count")
    plt.pause(5)
```

Trên máy ảo Ryu, mở một terminal riêng và chạy:

```
python3 ddos_monitor.py
```

```
=== Real-time DDoS Monitoring ===
10.0.0.6 -> 10.0.0.1: 5 detections
10.0.0.13 -> 10.0.0.1: 10 detections
10.0.0.15 -> 10.0.0.1: 8 detections
10.0.0.7 -> 10.0.0.1: 12 detections
10.0.0.2 -> 10.0.0.1: 14 detections
10.0.0.10 -> 10.0.0.1: 26 detections
10.0.0.4 -> 10.0.0.1: 21 detections
10.0.0.11 -> 10.0.0.1: 29 detections
10.0.0.3 -> 10.0.0.1: 17 detections
10.0.0.14 -> 10.0.0.1: 24 detections
10.0.0.5 -> 10.0.0.1: 31 detections
10.0.0.9 -> 10.0.0.1: 19 detections
10.0.0.12 -> 10.0.0.1: 21 detections
10.0.0.8 -> 10.0.0.1: 18 detections

[Updated at 04:22:59]
Press Ctrl+C to stop.
```

Hình 15. Màn hình làm việc của Ddos Monitor

Hệ thống sẽ liên tục cập nhật biểu đồ theo thời gian thực mỗi 5 giây, hiển thị các IP tấn công được phát hiện và số lượt xuất hiện của chúng.

### Ý nghĩa và lợi ích

- Cung cấp **giao diện giám sát trực tiếp**, thay vì chỉ xem file log.

- Phù hợp với các môi trường demo, trình diễn hệ thống.
- Hữu ích trong trường hợp hệ thống hoạt động liên tục nhiều giờ.

## 4.7 Đánh giá khả năng phát hiện và phản ứng của mô hình

### 4.7.1 Hiệu quả phát hiện trong điều kiện thực tế

Sau quá trình huấn luyện và triển khai mô hình CNN trong môi trường ảo hóa SDN sử dụng Mininet và Ryu Controller, hệ thống được đưa vào thử nghiệm với các kịch bản tấn công DDoS bao gồm: **ICMP Flood, UDP Flood, TCP SYN Flood**.

Kết quả cho thấy:

- **Tỷ lệ phát hiện chính xác cao:** Các luồng tấn công được mô hình phát hiện chính xác với tỷ lệ trên 98% (trên tập kiểm thử).
- **Thời gian phản ứng gần như tức thời:** Sau khi nhận PacketIn, controller chỉ mất ~10–20ms để dự đoán và thực thi hành động (chặn IP, log, gửi email).
- **Chặn kịp thời lưu lượng tấn công:** IP bị phát hiện được drop tại switch giúp giảm tải cho control plane và host mục tiêu.
- **Không chặn nhầm (false positive) đáng kể:** Các luồng hợp lệ không bị mô hình xử lý sai nhờ bước chuẩn hóa và huấn luyện kỹ càng.

Hệ thống chạy ổn định, không phát sinh lỗi nghiêm trọng trong quá trình thử nghiệm nhiều đợt tấn công từ các host giả lập.

### 4.7.2 So sánh với giải pháp truyền thống

Tiêu chí	Machine Learning (CNN)	Firewall truyền thống	IDS/IPS truyền thống
<b>Phát hiện bất thường</b>	Phân tích đặc trưng thống kê	Dựa vào rule tĩnh	Rule hoặc signature
<b>Phát hiện Zero-day</b>	Có thể nếu huấn luyện tốt	Không	Rất hạn chế
<b>Thời gian phản ứng</b>	Tức thời với SDN	Thủ công, chậm	Chậm nếu không tự động

<b>Khả năng thích nghi</b>	Linh hoạt, cập nhật mô hình	Cứng nhắc	Cập nhật thủ công
<b>Cảnh báo</b>	Tự động qua email/log	Giới hạn	Có thể gửi log
<b>Tự động hóa</b>	Cao (ML + SDN)	Thấp	Trung bình

*Bảng 2: So sánh CNN với giải pháp truyền thống*

#### 4.7.3 Kết luận đánh giá demo thực tế

Mô hình Machine Learning tích hợp với SDN mang lại hiệu quả vượt trội trong việc:

- Phát hiện nhanh chóng và chính xác các kiểu tấn công DDoS.
- Tự động hóa hoàn toàn quá trình phản ứng.
- Giám sát và kiểm soát mạng ở mức độ tổng thể.
- Dễ triển khai, mở rộng và tối ưu hóa theo thời gian.

Trong khi đó, các giải pháp truyền thống (Firewall, IDS/IPS) thường gặp hạn chế về độ linh hoạt, độ trễ phản ứng và khả năng đối phó với các cuộc tấn công biến thể mới.

**Vì vậy, việc ứng dụng mô hình học máy trong môi trường mạng SDN không chỉ là xu hướng tất yếu mà còn là giải pháp thiết thực để nâng cao hiệu quả phòng chống DDoS trong kỷ nguyên mạng hiện đại.**

## CHƯƠNG 5: KẾT LUẬN

### 5.1 Kết luận

Trong bối cảnh các cuộc tấn công từ chối dịch vụ phân tán (DDoS) ngày càng tinh vi, việc phát hiện sớm và phản ứng kịp thời là một yêu cầu cấp thiết đối với các hệ thống mạng hiện đại. Đề tài này đã tiếp cận bài toán theo hướng mới mẻ: **kết hợp mô hình học máy CNN với kiến trúc mạng SDN** để xây dựng hệ thống phát hiện và phản ứng tự động với tấn công DDoS trong môi trường mô phỏng thực tế.

Những kết quả chính đã đạt được:

- **Thiết kế và huấn luyện thành công mô hình CNN** sử dụng các đặc trưng lưu lượng mạng (flow-based features), cho độ chính xác phát hiện cao.

- **Tích hợp mô hình CNN vào Ryu Controller**, giúp phát hiện và phân loại luồng tấn công theo thời gian thực.
- **Phản ứng tự động**: Hệ thống có khả năng chặn IP nguồn tấn công, ghi log, gửi cảnh báo email và hiển thị trực quan số lượng luồng bị phát hiện.
- **Xây dựng mô-đun giám sát ddos\_monitor.py**, hỗ trợ quản trị viên theo dõi lưu lượng tấn công một cách trực quan.
- **Kiểm thử hệ thống thành công trong môi trường máy ảo** với Mininet và Ryu, cho thấy tính khả thi và hiệu quả của giải pháp đề xuất.

Kết quả thực nghiệm cho thấy hệ thống có khả năng phát hiện chính xác, phản ứng linh hoạt và phù hợp với mô hình mạng hiện đại, mở ra hướng đi tiềm năng trong việc ứng dụng trí tuệ nhân tạo vào lĩnh vực an ninh mạng.

## 5.2 Những hạn chế và công việc chưa thực hiện được

Mặc dù hệ thống đã được triển khai thành công trong môi trường mô phỏng và đạt kết quả khả quan, nhưng vẫn còn một số mặt hạn chế và công việc chưa thực hiện được trong phạm vi đề tài:

- **Chưa triển khai trên mạng vật lý thật**: Toàn bộ hệ thống được triển khai và kiểm thử trong môi trường máy ảo (Mininet, Ryu). Việc áp dụng vào hệ thống mạng thực tế cần được kiểm chứng thêm để đánh giá khả năng chịu tải, tính ổn định và mức độ tin cậy.
- **Chưa đánh giá toàn diện các mô hình học sâu khác**: Dù đã thử nghiệm nhiều mô hình học máy, nhưng phần đánh giá chuyên sâu chỉ tập trung vào CNN. Các mô hình như LSTM, Bi-LSTM, hoặc Transformer chưa được áp dụng để so sánh hiệu quả.
- **Chưa xử lý các tấn công lớp ứng dụng (Layer 7)**: Hệ thống hiện chủ yếu xử lý các tấn công ở tầng mạng (L3) và tầng giao vận (L4), như TCP SYN Flood, UDP Flood, ICMP Flood. Các kiểu tấn công ở tầng ứng dụng như HTTP Flood, Slowloris... chưa được hỗ trợ.
- **Tập dữ liệu còn hạn chế**: Hệ thống vẫn phụ thuộc vào tập dữ liệu thu thập được từ mô phỏng, với số lượng IP giả lập giới hạn. Việc sử dụng tập dữ liệu từ thực tế

hoặc từ nhiều nguồn khác nhau sẽ giúp tăng tính đa dạng và khả năng khái quát của mô hình.

- **Chưa có tính năng học liên tục (continual learning):** Mô hình CNN hiện tại hoạt động theo kiểu học offline, không tự động cập nhật trong quá trình chạy. Điều này có thể làm giảm hiệu quả nếu xuất hiện các kiểu tấn công mới chưa từng huấn luyện.

### 5.3 Hướng phát triển

Sau khi triển khai thành công trong môi trường mô phỏng, hệ thống vẫn còn nhiều tiềm năng để nâng cấp và mở rộng nhằm nâng cao khả năng ứng dụng thực tế. Một số định hướng phát triển trong tương lai bao gồm:

- **Nâng cấp mô hình học máy:** Thử nghiệm thêm các kiến trúc học sâu như LSTM, BiLSTM hoặc Transformer để cải thiện độ chính xác và khả năng phát hiện các kiểu tấn công có tính chuỗi phức tạp.
- **Áp dụng học trực tuyến (online learning):** Xây dựng cơ chế cho phép mô hình tự cập nhật theo thời gian thực từ dữ liệu mới trong quá trình vận hành, giúp hệ thống thích ứng nhanh với các hành vi tấn công mới phát sinh.
- **Tự động hóa quy trình thu thập dữ liệu:** Kết hợp với các công cụ giám sát mạng hoặc hệ thống IDS/IPS để tự động thu thập, gán nhãn và cập nhật dữ liệu huấn luyện, giảm phụ thuộc vào dữ liệu mô phỏng.
- **Mở rộng khả năng phát hiện nhiều loại tấn công:** Tăng cường phạm vi phát hiện các kiểu tấn công lớp ứng dụng (Layer 7) như HTTP Flood, Slowloris, hoặc các hình thức tấn công đa tầng.
- **Đánh giá trong môi trường mạng thực tế:** Triển khai thử nghiệm hệ thống trên hạ tầng mạng vật lý để kiểm tra hiệu năng, khả năng chịu tải và độ ổn định trong điều kiện vận hành thật.
- **Tích hợp giao diện quản trị trực quan:** Phát triển dashboard hoặc công cụ giám sát Web để giúp quản trị viên theo dõi trạng thái hệ thống, nhận cảnh báo và cấu hình chính sách mạng một cách dễ dàng.

## TÀI LIỆU THAM KHẢO

### Tiếng Việt

<https://www.microsoft.com/vi-vn/security/business/security-101/what-is-a-ddos-attack>

<https://cmccloud.vn/tin-tuc/sdn-la-gi>

[https://repository.vnu.edu.vn/bitstream/VNU\\_123/8236/5/LuanVan\\_NguyenThanhHuu.pdf](https://repository.vnu.edu.vn/bitstream/VNU_123/8236/5/LuanVan_NguyenThanhHuu.pdf)

[https://www.researchgate.net/publication/352313069\\_DDoS -](https://www.researchgate.net/publication/352313069_DDoS_-_)

[Mot So Ky Thuat Tan Cong va Phuong Phap Phong Chong](#)

<https://www.studocu.vn/vn/document/hoc-vien-chinh-sach-va-phat-trien/tieng-anh-co-ban-3/software-defined-networking-cong-ngh-m/71147780>

[https://github.com/lephamcong/DDoS\\_Detection\\_and\\_Mitigation\\_in\\_SDN?tab=readme-ov-file](https://github.com/lephamcong/DDoS_Detection_and_Mitigation_in_SDN?tab=readme-ov-file)

### Tiếng Anh

<https://github.com/chiragbiradar/DDoS-Attack-Detection-and-Mitigation>