

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM**

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**HỒ NHỰT HỒNG QUANG - 52300246**

**NGUYỄN CÔNG TOÀN - 52200271**

**PHÁT HIỆN VÀ NGĂN CHẶN DDOS LÊN MÁY CHỦ  
WEB GIẢ LẬP, BẰNG MACHINE LEARNING  
TRONG MẠNG SDN**

**BÁO CÁO CUỐI KÌ**

**BẢO MẬT MẠNG**

**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**



**HỒ NHỰT HỒNG QUANG - 52300246**  
**NGUYỄN CÔNG TOÀN - 52200271**

**PHÁT HIỆN VÀ NGĂN CHẶN DDOS LÊN MÁY CHỦ  
WEB GIẢ LẬP, BẰNG MACHINE LEARNING  
TRONG MẠNG SDN**

**BÁO CÁO CUỐI KÌ**

**BẢO MẬT MẠNG**

Người hướng dẫn  
**TS. Trần Chí Thiện**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2025**

## LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn quý giảng viên đã luôn tận tâm giảng dạy, hướng dẫn và truyền đạt kiến thức một cách chi tiết, giúp chúng em hiểu sâu hơn về chuyên ngành. Sự tận tụy của thầy, cùng với sự hỗ trợ và tạo điều kiện của nhà trường, đã giúp chúng em có cơ hội hoàn thành dự án này, góp phần gắn kết giữa học tập – thực hành – nghiên cứu ứng dụng trong thực tế nghề nghiệp. Bên cạnh đó chúng em cũng chân thành cảm ơn phía nhà trường luôn tạo cơ hội lắng nghe sinh viên, giúp đỡ sinh viên khi sinh viên cần điều đó quá tuyệt vời. Chúng em xin chân thành cảm ơn ạ!

TP. Hồ Chí Minh, ngày 21 tháng 11 năm 2025

Tác giả

(Ký tên và ghi rõ họ tên)

Quang

Hồ Nhật Hồng Quang

Toan

Nguyễn Công Toàn

## **CÔNG TRÌNH ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Chúng em xin cam đoan đây là công trình nghiên cứu của riêng chúng em và được sự hướng dẫn khoa học của **TS. Trần Chí Thiện**. Các nội dung nghiên cứu, kết quả trong đề tài 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 dự 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 dự án của mình. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do chúng em gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 21 tháng 11 năm 2025

Tác giả

(Ký tên và ghi rõ họ tên)

Quang

Hồ Nhật Hồng Quang

Toan

Nguyễn Công Toàn

# **PHÁT HIỆN VÀ NGĂN CHẶN DDOS LÊN MÁY CHỦ WEB GIẢ LẬP BẰNG MACHINE LEARNING TRONG MẠNG SDN**

## **TÓM TẮT**

Tấn công từ chối dịch vụ phân tán (DDoS) hiện đang là một trong những mối đe dọa nghiêm trọng nhất đối với an ninh mạng trên toàn cầu. Những cuộc tấn công này có khả năng gây ra sự gián đoạn nghiêm trọng cho các dịch vụ và ứng dụng quan trọng bằng cách tràn ngập hệ thống mục tiêu với lưu lượng truy cập cực kỳ lớn, vượt quá khả năng xử lý của hệ thống đó. Những tổn thất này không chỉ gây ảnh hưởng trực tiếp đến hoạt động của các dịch vụ mà còn có thể làm giảm độ tin cậy của các hệ thống mạng, dẫn đến thiệt hại về tài chính và uy tín. Với sự phát triển nhanh chóng của công nghệ mạng và sự gia tăng không ngừng của các thiết bị Internet of Things (IoT), các cuộc tấn công DDoS ngày càng trở nên phức tạp và mạnh mẽ hơn, khiến việc bảo vệ các hệ thống mạng trở nên khó khăn và cần phải có các giải pháp phát hiện và giảm thiểu hiệu quả hơn.

Báo cáo này tập trung vào việc khai thác tiềm năng của công nghệ học máy (Machine Learning - ML) trong việc phát hiện các cuộc tấn công DDoS trong môi trường mạng định nghĩa bằng phần mềm (SDN - Software-Defined Networking). Mạng SDN, với khả năng lập trình linh hoạt và điều khiển tập trung, là một lựa chọn lý tưởng để triển khai các phương pháp phát hiện tấn công DDoS, đặc biệt là khi sự phân tán của tấn công ngày càng tinh vi và khó phát hiện. Mục tiêu chính của đề án là xây dựng một hệ thống SDN, mô phỏng các cuộc tấn công DDoS, và thử nghiệm các phương pháp phát hiện và giảm thiểu tấn công. Hệ thống sẽ sử dụng các thuật toán học máy để phân tích và phát hiện những bất thường trong lưu lượng mạng, từ đó nhận diện các cuộc tấn công DDoS trong thời gian thực.

Kết quả thực nghiệm của nghiên cứu chỉ ra rằng phương pháp phát hiện và giảm thiểu tấn công DDoS dựa trên học máy kết hợp đã đạt được tỷ lệ phát hiện cao đối với các loại tấn công DDoS phổ biến, đồng thời giảm thiểu được tác động của chúng đối với hiệu suất của hệ thống. Các thuật toán học máy đã chứng minh hiệu quả trong việc phân loại lưu lượng mạng và phát hiện các mẫu tấn công, giúp giảm thiểu tình trạng gián đoạn dịch vụ do tấn công DDoS gây ra. Ngoài ra, nghiên cứu còn đề xuất một số giải pháp tối ưu để cải thiện khả năng phòng chống DDoS trong môi trường SDN, nhằm đảm bảo sự ổn định và an toàn cho các hệ thống mạng hiện đại.

## MỤC LỤC

LỜI CẢM ƠN.....	3
DANH MỤC HÌNH ẢNH.....	8
DANH MỤC BẢNG BIỂU .....	9
DANH MỤC CÁC CHỮ VIẾT TẮT.....	10
CHƯƠNG 1. MỞ ĐẦU VÀ TỔNG QUAN ĐỀ TÀI .....	11
1.1 Lý do chọn đề tài .....	11
1.2 Mục tiêu thực hiện đề tài .....	12
1.3 Tổng quan về đề tài .....	12
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT.....	14
2.1 Mạng định nghĩa bằng phần mềm (software defined network SDN):.....	14
2.1.1 So sánh một số điểm khác nhau giữa Mạng định nghĩa bằng phần mềm và mạng truyền thống.....	17
2.2 Tấn công từ chối dịch vụ phân tán (distributed denial of service – DDOS) .....	21
2.2.1 Một số loại tấn công DDoS phổ biến: .....	23
2.3 Thuật toán decision tree.....	27
2.3.1 Cấu Trúc Của Cây Quyết Định: .....	27
2.3.2 Tiêu chí chia tách: Gini Impurity: .....	28
2.3.3 Quá Trình Xây Dựng Cây Quyết Định (Decision Tree): .....	29
2.3.4 Độ Quan Trọng Của Đặc Trưng (Feature Importance): .....	30
2.3.5 Vấn Đề Overfitting Trong Cây Quyết Định: .....	30
2.3.6 Random Forest - Phương Pháp Tập Hợp để Giải Quyết Overfitting: .....	31
2.3.7 Cân Bằng Lớp (Class Balance) và Hiệu Suất Mô Hình: .....	32
CHƯƠNG 3: MÔ HÌNH ĐỀ XUẤT.....	33
3.1 Tổng Quan Hệ Thống .....	33
3.2 Kiến Trúc Chi Tiết.....	33
3.2.1 Bộ Thu Thập Dữ Liệu (Collector): .....	33
3.2.2 Trích Xuất Đặc Trưng (Feature Engineering): .....	34
3.3.2 Mô Hình Học Máy (Machine Learning Model): .....	34

3.3 Qui Trình End-to-End.....	35
3.3.1 Chuẩn Bị Dữ Liệu: .....	35
3.3.2 Huấn Luyện Mô Hình (Model Training):.....	36
3.3.3 Dự Đoán Real-Time: .....	37
CHƯƠNG 4: MÔ PHÒNG .....	39
4.1 Thu thập dữ liệu cho dataset.....	39
4.1.1 Thu thập lưu lượng bình thường qua mạng .....	39
4.1.2 Thu thập lưu lượng giả lập DDOS qua mạng .....	40
4.2 Huấn luyện model machine learning .....	41
4.3 Giả lập một webserver và áp dụng machine learning để phát hiện và ngăn chặn DDOS.	43
4.3.1 Yêu cầu cơ bản trước khi chạy .....	43
4.3.2 Toàn bộ quá trình ngăn chặn DDOS.....	44
CHƯƠNG 5. KẾT LUẬN .....	50
5.1 Kết luận.....	50
5.2 Hướng phát triển .....	51
TÀI LIỆU THAM KHẢO .....	52

## DANH MỤC HÌNH ẢNH

Hình 1: Kiến trúc mạng định nghĩa bằng phần mềm .....	14
Hình 2: OpenSDN .....	16
Hình 3: SDN via Hypervisor-based Overlay Network.....	17
Hình 4: Tấn công từ chối dịch vụ phân tán .....	22
Hình 5: Minh họa SYN Flood Attack.....	24
Hình 6: Minh họa HTTP Flood Attack.....	25
Hình 7: Sơ đồ cấu trúc cây quyết định .....	28
Hình 8: Chuẩn Bị Dữ Liệu .....	35
Hình 9: Huấn Luyện Mô Hình.....	37
Hình 10: Dự Đoán Real-Time .....	38

## **DANH MỤC BẢNG BIỂU**

Bảng 1: So sánh giữa Mạng Định Nghĩa Bằng Phần Mềm và Mạng Truyền Thống.....	19
Bảng 2: Sự khác nhau giữa DoS và DDoS .....	23

## DANH MỤC CÁC CHỮ VIẾT TẮT

# CHƯƠNG 1. MỞ ĐẦU VÀ TỔNG QUAN ĐỀ TÀI

## 1.1 Lý do chọn đề tài

Trong kỷ nguyên số hiện nay, công nghệ mạng phát triển nhanh chóng và nhu cầu kinh doanh trực tuyến ngày càng tăng cao. Các dịch vụ mạng đóng vai trò quan trọng trong việc cung cấp thông tin và hỗ trợ các hoạt động kinh tế, xã hội. Tuy nhiên, sự phát triển này cũng kéo theo những mối đe dọa an ninh mạng, đặc biệt là các cuộc tấn công DDoS (Distributed Denial of Service). Các cuộc tấn công này có thể gây gián đoạn nghiêm trọng cho các dịch vụ mạng, dẫn đến tổn thất lớn về tài chính, phá hủy dữ liệu và thậm chí là thiệt hại về uy tín của các tổ chức. DDoS đã trở thành một trong những mối nguy hiểm lớn nhất đối với an ninh mạng, gây ảnh hưởng rộng rãi từ các doanh nghiệp, tổ chức tài chính đến cơ quan chính phủ.

Việc phát hiện và giảm thiểu các cuộc tấn công DDoS đã trở thành một thách thức lớn trong lĩnh vực an ninh mạng. Các phương pháp phát hiện tấn công truyền thống dựa trên việc phân tích lưu lượng mạng hoặc phát hiện bất thường lưu lượng vẫn gặp phải nhiều hạn chế trong việc nhận diện các cuộc tấn công nhanh chóng và chính xác. Các cuộc tấn công DDoS hiện đại ngày càng tinh vi hơn, làm cho việc phát hiện trở nên khó khăn. Chính vì vậy, việc tìm kiếm các phương pháp phát hiện tấn công hiệu quả hơn là một vấn đề nghiên cứu quan trọng. Mạng định nghĩa bằng phần mềm (SDN) với khả năng lập trình mạng và quản lý tập trung đã mở ra cơ hội mới để phát hiện và ngăn chặn tấn công DDoS một cách hiệu quả.

SDN không chỉ cho phép quản trị viên mạng có thể điều khiển lưu lượng từ một điểm duy nhất mà còn cung cấp khả năng linh hoạt trong việc điều chỉnh các chiến lược bảo vệ mạng. Bằng cách tích hợp các công nghệ phân tích lưu lượng, học máy và các công cụ giám sát như Snort, OpenFlow, SDN có thể phát hiện các dấu hiệu tấn công và giảm thiểu tác động của chúng. Với sự tiến bộ của các thuật toán học máy như SVM, Decision Tree, Random Forest và Deep Learning, việc phát hiện các cuộc tấn công DDoS có thể được tự động hóa, giúp giảm thiểu sự gián đoạn và bảo vệ các dịch vụ mạng.

## 1.2 Mục tiêu thực hiện đề tài

Mục tiêu của đề tài này là áp dụng các thuật toán học máy (Machine Learning) trong việc phát hiện và giảm thiểu các cuộc tấn công DDoS trong môi trường SDN. Cụ thể, mục tiêu của đề tài bao gồm:

- Xây dựng mô hình mạng SDN: Tạo ra một hệ thống SDN mô phỏng các cuộc tấn công DDoS
- Thu thập và phân tích dữ liệu lưu lượng mạng: Thu thập các đặc trưng lưu lượng mạng bình thường và lưu lượng mạng trong trường hợp giả định tấn công DDoS để làm dữ liệu huấn luyện cho các mô hình học máy.
- Phát triển hệ thống phát hiện tấn công DDoS: Áp dụng thuật toán học máy Random Forest để phát hiện các cuộc tấn công DDoS từ dữ liệu lưu lượng mạng bất thường lên một máy chủ Web.
- Giảm thiểu tác động của tấn công DDoS: Phát triển biện ngăn chặn và giảm thiểu tấn công, bao gồm tối ưu hóa chiến lược quản lý lưu lượng mạng và điều chỉnh các thông số của hệ thống SDN để cải thiện khả năng phục hồi và bảo vệ mạng.

## 1.3 Tổng quan về đề tài

Tấn công DDoS là một trong những mối đe dọa lớn đối với an ninh mạng hiện nay, và việc phát hiện các cuộc tấn công này một cách chính xác và nhanh chóng là vấn đề quan trọng trong lĩnh vực bảo mật. Các phương pháp phát hiện tấn công truyền thống chủ yếu dựa trên việc phân tích đặc điểm lưu lượng hoặc tìm kiếm bất thường trong lưu lượng mạng. Tuy nhiên, những phương pháp này gặp nhiều hạn chế trong việc đối phó với các tấn công tinh vi và phức tạp, khi mà các mẫu lưu lượng có thể thay đổi liên tục và khó nhận diện.

SDN là một kiến trúc mạng linh hoạt với khả năng lập trình trực tiếp và quản lý tập trung, điều này giúp tối ưu hóa việc phát hiện và ngăn chặn các cuộc tấn công DDoS. SDN cung cấp một nền tảng lý tưởng để áp dụng các phương pháp phát hiện tấn công dựa trên học máy, giúp phát hiện các bất thường trong lưu lượng mạng một cách tự động và chính xác hơn. Trong môi trường SDN, các công cụ như Snort và OpenFlow có thể được sử dụng để giám sát lưu lượng

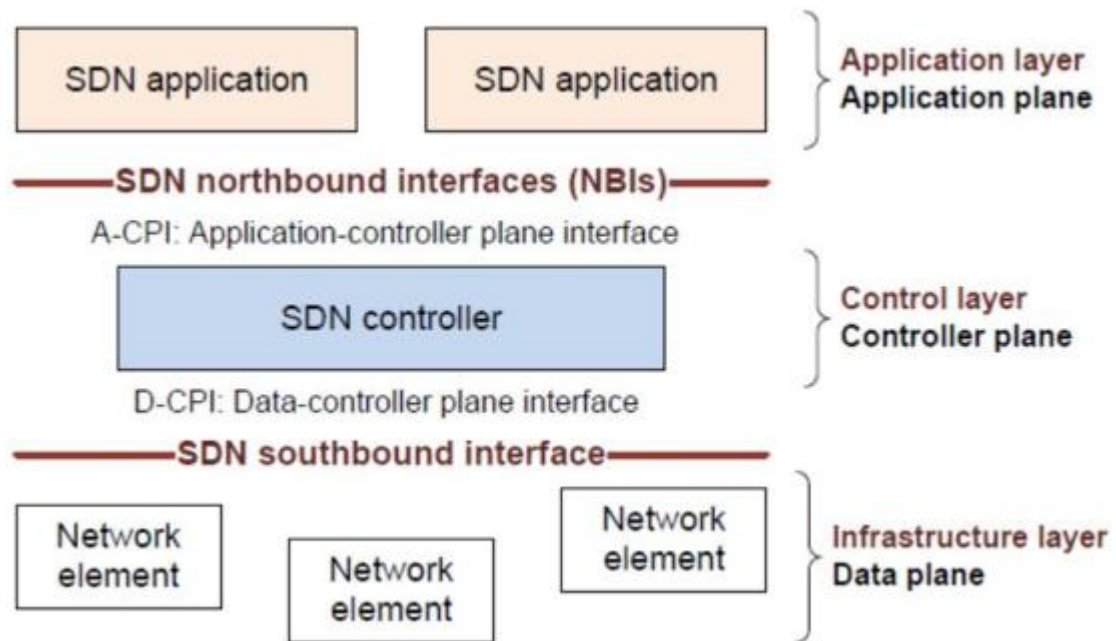
và phát hiện các dấu hiệu của tấn công. Các phương pháp dựa trên học máy sẽ phân tích lưu lượng mạng và tự động nhận diện các cuộc tấn công DDoS, từ đó giúp hệ thống phản ứng kịp thời để ngăn chặn hoặc giảm thiểu tác động của tấn công.

Đề tài này sẽ nghiên cứu và phát triển một hệ thống SDN sử dụng học máy để phát hiện và giảm thiểu các cuộc tấn công DDoS. Dữ liệu thu thập từ các cuộc tấn công DDoS sẽ được sử dụng để huấn luyện các mô hình học máy, giúp phát hiện các tấn công trong thời gian thực. Hệ thống cũng sẽ áp dụng các biện pháp giảm thiểu tấn công như điều chỉnh lưu lượng mạng và tối ưu hóa các chiến lược bảo vệ, giúp giảm thiểu sự gián đoạn và bảo vệ các dịch vụ mạng trong môi trường SDN.

## CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

### 2.1 Mạng định nghĩa bằng phần mềm (software defined network SDN):

Mạng định nghĩa bằng phần mềm (SDN) là một kiến trúc mạng mới, linh hoạt và dễ dàng quản lý, với khả năng mở rộng và tiết kiệm chi phí, phù hợp với yêu cầu của các ứng dụng đòi hỏi băng thông cao và tính năng động trong môi trường mạng hiện đại. Kiến trúc này phân tách các chức năng điều khiển và chuyển tiếp dữ liệu, tạo điều kiện cho việc điều khiển mạng thông qua lập trình trực tiếp và trừu tượng hóa cơ sở hạ tầng mạng, giúp các ứng dụng và dịch vụ có thể dễ dàng tương tác với hệ thống mạng. SDN cho phép các quản trị viên mạng có thể quản lý lưu lượng dữ liệu từ một bảng điều khiển trung tâm mà không cần quan tâm đến từng switch trong hệ thống. Bộ điều khiển SDN tập trung sẽ chỉ đạo các thiết bị chuyển mạch thực hiện các nhiệm vụ mạng theo yêu cầu, đồng thời quản lý toàn bộ kết nối giữa các máy chủ và thiết bị trong mạng.



Hình 1: Kiến trúc mạng định nghĩa bằng phần mềm

Kiến trúc của SDN được chia thành ba lớp chính: lớp ứng dụng (Application Plane), lớp điều khiển (Control Plane) và lớp cơ sở hạ tầng (Data Plane), như được mô tả trong hình 1.

Trong mạng truyền thống, mỗi switch đều có một lớp dữ liệu (data plane) và lớp điều khiển

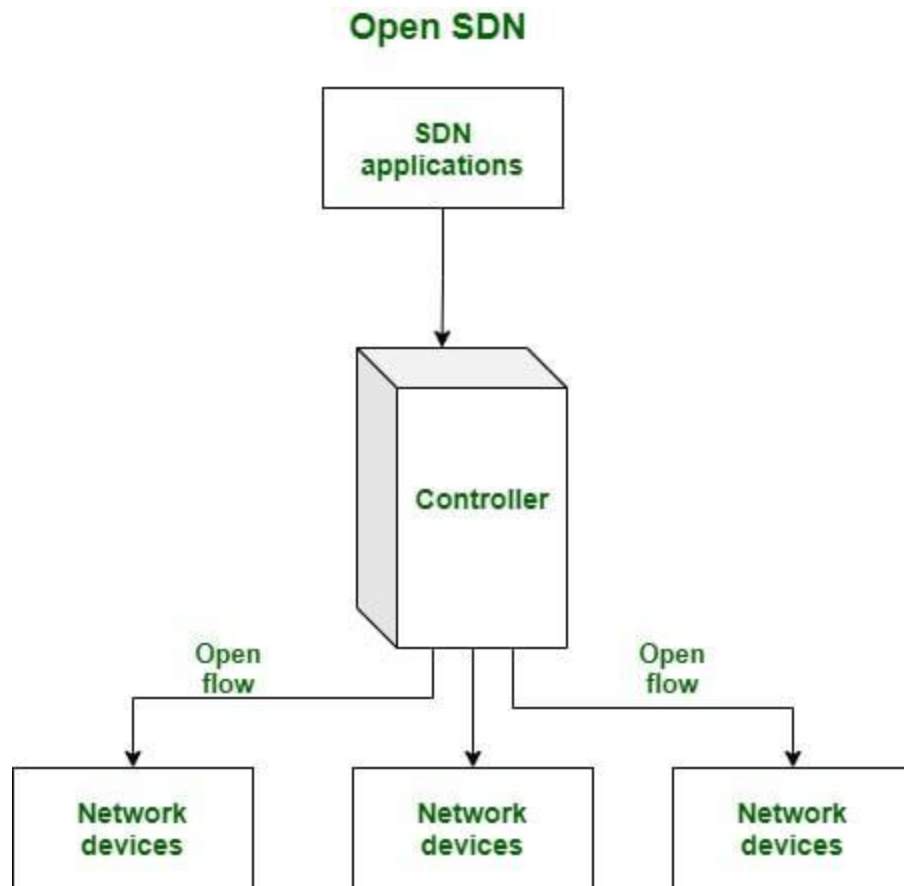
(control plane) riêng biệt. Lớp điều khiển của các switch trao đổi thông tin về cấu trúc mạng và từ đó xây dựng bảng chuyển tiếp (forwarding table) để quyết định cách thức chuyển tiếp gói dữ liệu. Mạng SDN là phương pháp tách rời lớp điều khiển khỏi các switch và chuyển giao chức năng này cho một bộ điều khiển tập trung gọi là SDN controller. Nhờ vậy, các quản trị viên mạng có thể quản lý lưu lượng mạng thông qua bảng điều khiển tập trung mà không cần trực tiếp can thiệp vào từng switch. Lớp dữ liệu vẫn nằm trong các switch, và khi gói dữ liệu đi qua switch, việc chuyển tiếp được quyết định dựa trên các mục nhập trong bảng luồng (flow table) đã được cấu hình trước bởi bộ điều khiển. Mỗi bảng luồng bao gồm các trường khớp (như cổng đầu vào và header gói tin) và các hướng dẫn thực thi. Gói tin sẽ được khớp với các mục nhập trong bảng luồng dựa trên các trường khớp, và các hướng dẫn tương ứng sẽ được thực thi. Những hướng dẫn này có thể bao gồm việc chuyển tiếp gói tin qua một hoặc nhiều cổng, loại bỏ gói tin, hoặc thêm các tiêu đề vào gói tin. Nếu gói tin không tìm thấy mục nhập khớp trong bảng luồng, switch sẽ yêu cầu bộ điều khiển cung cấp mục luồng mới, và sau đó tiếp tục chuyển tiếp hoặc loại bỏ gói tin dựa trên mục luồng này.

Một kiến trúc SDN điển hình gồm ba lớp chính:

- Lớp ứng dụng (Application Layer): Chứa các ứng dụng mạng như hệ thống phát hiện xâm nhập, tường lửa, và cân bằng tải.
- Lớp điều khiển (Control Layer): Bao gồm bộ điều khiển SDN, đóng vai trò là bộ não của mạng và giúp trừu tượng hóa phần cứng cho các ứng dụng.
- Lớp cơ sở hạ tầng (Infrastructure Layer): Bao gồm các switch vật lý tạo thành lớp dữ liệu, thực hiện việc chuyển tiếp các gói dữ liệu.

Các lớp giao tiếp thông qua một tập hợp các giao diện được gọi là northbound APIs và southbound APIs. Một số mô hình khác nhau của SDN:

- Open SDN (hình 2): được triển khai bằng cách sử dụng switch OpenFlow. Đây là một cách triển khai đơn giản của SDN. Trong Open SDN, bộ điều khiển giao tiếp với các switch bằng cách sử dụng southbound APIs thông qua giao thức OpenFlow.



Hình 2: OpenSDN

- SDN via APIs: các chức năng trong các thiết bị từ xa như switch được gọi bằng các phương pháp truyền thống như SNMP hoặc CLI hoặc thông qua các phương pháp mới như Rest API. Ở đây, các thiết bị được cung cấp với điểm điều khiển cho phép bộ điều khiển thao tác các thiết bị từ xa bằng cách sử dụng các API.
- SDN via Hypervisor-based Overlay Network (hình 3): cấu hình của các thiết bị vật lý không thay đổi. Thay vào đó, các mạng lớp phủ dựa trên Hypervisor được tạo ra trên mạng vật lý. Chỉ có các thiết bị ở biên của mạng vật lý được kết nối với các mạng ảo hóa, do đó che giấu thông tin của các thiết bị khác trong mạng vật lý



Cơ chế ra quyết định	Mỗi switch/router tự quyết định (distributed control)	Một controller trung tâm điều khiển toàn mạng (centralized control)	SDN tập trung, truyền thống phân tán
Kiến trúc hệ thống	Thiết bị mạng thông minh (switch = brain)	Controller là bộ não, switch trở thành thiết bị chuyển tiếp đơn giản	SDN tách control plane và data plane
Thu thập dữ liệu lưu lượng (flow)	Khó khăn, phải dùng SNMP, NetFlow, syslog	Lấy trực tiếp qua OpenFlow API (JSON)	SDN thu thập dữ liệu dễ hơn và chuẩn hóa
Tốc độ thu thập số liệu	Chậm (5–10 phút mỗi chu kỳ)	<1 giây (real-time)	SDN nhanh hơn ~100 lần
Tầm nhìn lưu lượng toàn mạng	Không đầy đủ, mỗi switch chỉ thấy phần của nó	Controller nhìn toàn bộ topologies và flows	SDN có global visibility
Khả năng chặn tấn công DDoS	Chậm: cần SSH vào từng switch, chỉnh ACL thủ công	Nhanh: <1 giây qua API tự động	SDN nhanh hơn ~300 lần
Cấu hình thiết bị	Thủ công, phải SSH từng switch/router	Lập trình tự động bằng REST API, Python, OpenFlow	SDN giảm tải công việc quản trị
Tích hợp Machine Learning	Khó: dữ liệu thô, thiếu chuẩn, phân tán	Dễ: controller cung cấp dữ liệu đầy đủ và chuẩn	SDN phù hợp cho hệ thống phát hiện tấn công bằng ML
Khả năng mở rộng (scalability)	Khó: nhiều switch = nhiều cấu hình	Dễ: 1 controller quản lý hàng trăm switch	SDN mở rộng tốt, truyền thống mở rộng chậm

<b>Chi phí</b>	Rẻ ban đầu, dùng thiết bị cũ	Tốn đầu tư ban đầu (controller, đào tạo)	SDN đắt hơn nhưng hiệu quả vận hành cao hơn
----------------	------------------------------	--	---

Bảng 1: So sánh giữa Mạng Định Nghĩa Bằng Phần Mềm và Mạng Truyền Thống

***Ưu điểm của SDN (Software-Defined Networking):***

- Quản lý tập trung: SDN tách riêng phần điều khiển (control plane) và phần xử lý dữ liệu (data plane), nhờ đó toàn bộ hệ thống mạng có thể được điều phối và giám sát từ một điểm trung tâm duy nhất.
- Mức độ lập trình linh hoạt: Khi lớp điều khiển tách biệt, mạng có khả năng lập trình cao hơn, cho phép cấu hình và vận hành mạng một cách chủ động, nhanh chóng và tối ưu.
- Khả năng ảo hóa mạng: SDN hỗ trợ ảo hóa các tài nguyên mạng vật lý, tạo thành nhiều mạng ảo phục vụ từng ứng dụng hoặc dịch vụ riêng, giúp khai thác tài nguyên hiệu quả hơn.
- Triển khai tính năng nhanh chóng: Nhờ đặc tính lập trình, SDN rút ngắn quá trình triển khai, cập nhật và thử nghiệm các chức năng mạng mới, từ đó giảm đáng kể thời gian và chi phí.
- Tương thích với điện toán đám mây: SDN đóng vai trò quan trọng trong môi trường cloud, cho phép phân bổ và điều phối tài nguyên mạng một cách linh hoạt, phù hợp nhu cầu của các dịch vụ đám mây.
- Tăng cường bảo mật: Việc quản trị tập trung giúp SDN dễ dàng giám sát và kiểm soát truy cập, nhờ đó triển khai các biện pháp an ninh mạng hiệu quả hơn và phản ứng nhanh trước sự cố.

***Nhược điểm của SDN:***

- Phụ thuộc vào bộ điều khiển trung tâm: SDN phụ thuộc vào bộ điều khiển trung tâm để quản lý và kiểm soát mạng. Nếu bộ điều khiển bị lỗi hoặc tấn công, toàn bộ mạng sẽ bị ảnh hưởng

- **Đòi hỏi nhân lực có chuyên môn cao:** Triển khai và quản lý SDN yêu cầu kỹ sư phải có kiến thức sâu về lập trình mạng, kiến trúc hệ thống và bảo mật, điều mà không phải tổ chức nào cũng có sẵn.
- **Gia tăng rủi ro bảo mật:** Việc tập trung quyền điều khiển vào một điểm duy nhất khiến SDN trở thành mục tiêu hấp dẫn cho tin tặc. Nếu controller bị tấn công, toàn bộ mạng sẽ bị ảnh hưởng.
- **Rủi ro về bảo mật:** Tập trung điều khiển mạng tại một điểm duy nhất cũng làm tăng nguy cơ bị tấn công và xâm nhập bảo mật
- **Chi phí đầu tư ban đầu lớn:** Các doanh nghiệp cần trang bị thiết bị mới, phần mềm điều khiển và đào tạo nhân sự, dẫn đến chi phí ban đầu cao hơn so với mô hình mạng truyền thống.
- **Phụ thuộc vào nhà cung cấp:** Nhiều giải pháp SDN mang tính độc quyền theo từng hãng, khiến doanh nghiệp có nguy cơ bị “khóa chặt” vào một hệ sinh thái công nghệ duy nhất.

***Ứng dụng quan trọng của SDN bao gồm:***

- **Quản lý và điều khiển mạng tập trung:** SDN cho phép toàn bộ hệ thống mạng được quản trị từ một điểm trung tâm, giúp đơn giản hóa quá trình cấu hình, triển khai và nâng cao hiệu quả vận hành.
- **Ảo hóa tài nguyên mạng:** Nhờ khả năng lập trình, SDN hỗ trợ ảo hóa các phần tử mạng vật lý, giúp phân bổ tài nguyên linh hoạt cho từng dịch vụ và ứng dụng, từ đó tối ưu hóa hiệu suất sử dụng.
- **Hỗ trợ điện toán đám mây:** SDN là nền tảng quan trọng trong môi trường cloud, giúp quản lý tài nguyên mạng một cách tự động, linh hoạt và phù hợp với đặc thù thay đổi nhanh của các dịch vụ đám mây.
- **Tăng cường an ninh mạng:** Với cơ chế điều khiển tập trung, SDN cho phép xây dựng và triển khai nhanh các chính sách bảo mật như tường lửa, phân đoạn mạng, phát hiện và ngăn chặn tấn công, đồng thời nâng cao khả năng giám sát.
- **Ứng dụng trong mạng di động:** SDN giúp tối ưu phân bổ tài nguyên, linh hoạt điều chỉnh chính sách và hỗ trợ triển khai hạ tầng mạng di động hiệu quả hơn.

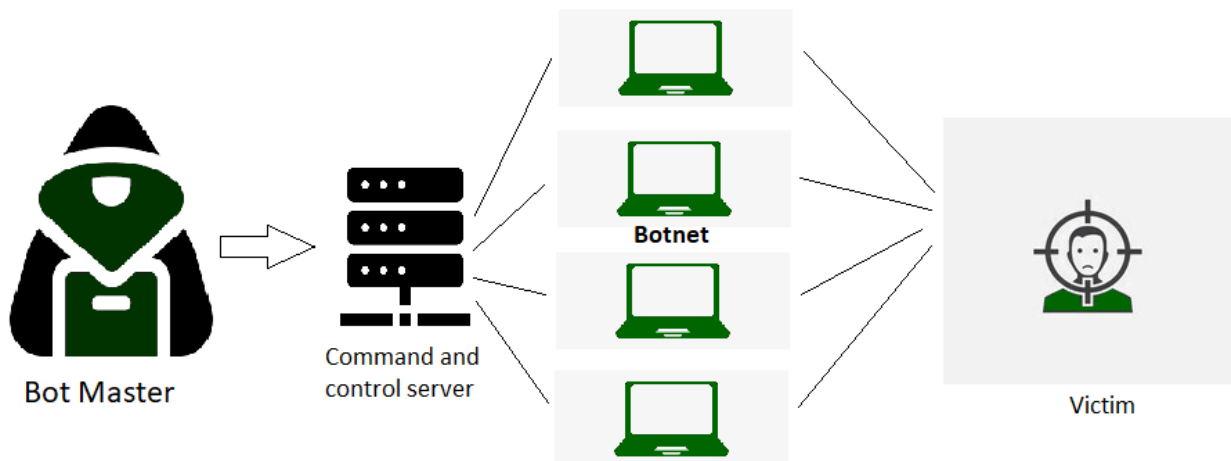
- Dành cho Internet of Things (IoT): Tính linh hoạt và khả năng điều khiển tập trung của SDN giúp kết nối, quản lý và bảo vệ số lượng lớn thiết bị IoT trong các hệ thống phức tạp.
- Mạng doanh nghiệp: SDN giúp các doanh nghiệp triển khai và vận hành mạng nội bộ dễ dàng hơn, đồng thời nâng cao mức độ bảo mật và hiệu quả sử dụng tài nguyên.
- Trung tâm dữ liệu: Trong các data center quy mô lớn, SDN cho phép tối ưu đường truyền, tăng hiệu suất khai thác tài nguyên mạng và mở rộng hạ tầng một cách đơn giản và hiệu quả.

## **2.2 Tấn công từ chối dịch vụ phân tán (distributed denial of service – DDoS)**

Tấn công Từ chối Dịch vụ Phân tán (Distributed Denial of Service – DDoS) là một dạng mở rộng của tấn công DoS, trong đó kẻ tấn công điều khiển nhiều hệ thống đã bị xâm nhập (thường là máy tính nhiễm mã độc hoặc botnet) để đồng loạt gửi lượng lớn yêu cầu đến một mục tiêu cụ thể.

DDoS lợi dụng số lượng lớn máy chủ và kết nối Internet để làm quá tải tài nguyên của nạn nhân. Đây được xem là một trong những hình thức tấn công nguy hiểm nhất trên môi trường mạng hiện nay. Khi một trang web bị gián đoạn hoạt động hoặc bị đánh sập, nguyên nhân phổ biến thường là do bị tấn công DDoS.

Trong kiểu tấn công này, kẻ tấn công tạo ra lưu lượng truy cập cực lớn gửi đến máy chủ mục tiêu, khiến hệ thống không thể xử lý kịp và dẫn đến tình trạng tê liệt hoặc ngừng hoạt động hoàn toàn.



Hình 4: Tấn công từ chối dịch vụ phân tán

DoS là viết tắt của Denial of Service (Từ chối Dịch vụ). Đây là một loại tấn công vào một dịch vụ nhằm làm gián đoạn chức năng bình thường của nó và ngăn cản người dùng khác truy cập vào dịch vụ đó. Mục tiêu phổ biến nhất của một cuộc tấn công DoS là dịch vụ trực tuyến như trang web, mặc dù các cuộc tấn công cũng có thể được thực hiện chống lại mạng, máy tính, hoặc thậm chí một chương trình đơn lẻ. Bảng 2 so sánh một số thông tin cơ bản giữa DoS và DDos.

Tiêu chí	DoS (Denial of Service)	DDoS (Distributed Denial of Service)
Khái niệm	Tấn công Từ chối Dịch vụ từ một nguồn duy nhất.	Tấn công Từ chối Dịch vụ Phân tán từ nhiều nguồn khác nhau.
Số lượng hệ thống tham gia tấn công	Chỉ một hệ thống thực hiện tấn công.	Nhiều hệ thống/botnet cùng tham gia tấn công.
Nguồn gửi lưu lượng	Lưu lượng tấn công đến từ một vị trí duy nhất.	Lưu lượng được gửi từ nhiều vị trí khác nhau.
Tốc độ tấn công	Mức độ tấn công thường yếu hơn, tốc độ chậm hơn.	Tạo ra lưu lượng lớn và tốc độ tấn công mạnh hơn.
Mức độ khó	Dễ bị ngăn chặn vì chỉ có	Khó phòng thủ do lưu lượng phân tán từ

khả năng phòng thủ	một nguồn tấn công.	nhiều thiết bị.
Công cụ tấn công	Một thiết bị hoặc công cụ DoS duy nhất.	Sử dụng mạng botnet với số lượng lớn thiết bị bị điều khiển.
Khả năng truy vết	Dễ xác định nguồn tấn công.	Rất khó truy vết do nguồn phân tán.
Các dạng tấn công phổ biến	- Tràn bộ đệm - Ping of Death / ICMP Flood - Teardrop Attack - Flooding	- Volumetric Attack - Phân mảnh gói tin - Tấn công tầng ứng dụng (Layer 7) - Tấn công dựa trên giao thức

Bảng 2: Sự khác nhau giữa DoS và DDoS

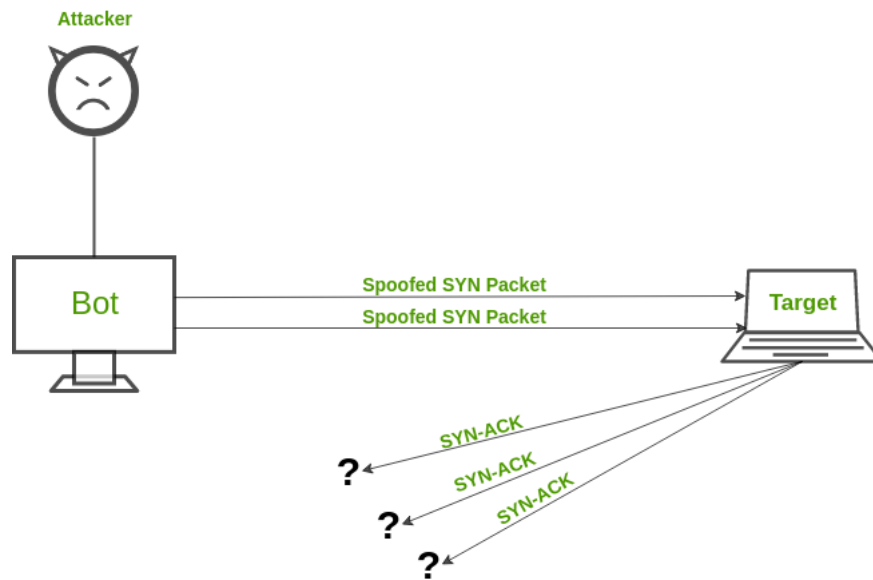
Một cuộc tấn công từ chối dịch vụ phân tán (DDoS) xảy ra khi một dịch vụ trực tuyến hợp lệ bị gửi quá nhiều yêu cầu vượt quá khả năng xử lý thông thường của nó. Chẳng hạn, một trang web chỉ có thể tiếp nhận và xử lý một lượng yêu cầu nhất định mỗi phút; khi lưu lượng truy cập vượt ngưỡng này, trang có thể hoạt động chậm, không ổn định hoặc ngừng đáp ứng hoàn toàn. Tình trạng quá tải này có thể xuất phát từ một cuộc tấn công, nhưng cũng có thể xảy ra do nhu cầu truy cập thật — ví dụ như các trang thương mại điện tử quá tải trong ngày Black Friday hoặc hệ thống bán vé bị nghẽn khi mở bán vé cho một sự kiện lớn.

Một cuộc tấn công DDoS thường được triển khai thông qua nhiều thiết bị bị xâm nhập và điều khiển từ xa ở nhiều vị trí khác nhau trên thế giới. Tập hợp các thiết bị này được gọi là botnet (Hình 4). Khác với tấn công DoS truyền thống — vốn chỉ dựa vào một thiết bị hoặc một kết nối Internet để tạo lưu lượng — tấn công DDoS tạo ra lưu lượng truy cập ồ ạt từ nhiều nguồn, khiến mục tiêu nhanh chóng bị quá tải và không thể duy trì dịch vụ.

### 2.2.1 Một số loại tấn công DDoS phổ biến:

- Tấn công SYN Flood (Hình 5): SYN Flood là một dạng tấn công từ chối dịch vụ (DoS) lợi dụng lỗ hổng trong cơ chế bắt tay ba bước của giao thức TCP (TCP three-way handshake). Trong kiểu tấn công này, kẻ tấn công gửi một số lượng lớn gói SYN đến máy chủ nhưng không hoàn tất quy trình kết nối. Điều này khiến máy chủ phải

dành tài nguyên để chờ phản hồi hoàn tất kết nối, dẫn đến trạng thái cạn kiệt tài nguyên. Khi tài nguyên bị chiếm giữ bởi các kết nối “nửa vời”, máy chủ không thể xử lý các yêu cầu hợp lệ từ người dùng thật, gây ra tình trạng gián đoạn hoặc ngừng hoạt động.

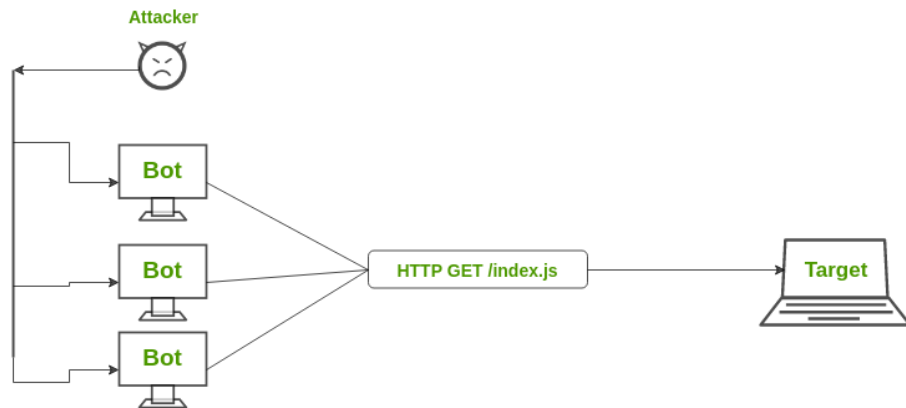


Hình 5: Minh họa SYN Flood Attack

Trong một cuộc tấn công SYN Flood, kẻ tấn công gửi một lượng lớn các gói SYN tới máy chủ, nhưng không bao giờ gửi gói ACK để hoàn tất quy trình bắt tay. Điều này khiến máy chủ liên tục giữ các kết nối nửa chừng, làm cạn kiệt tài nguyên hệ thống. Cụ thể:

- + Gửi gói SYN: Kẻ tấn công gửi nhiều gói SYN đến máy chủ, thường là từ các địa chỉ IP giả mạo để che giấu danh tính thực.
  - + SYN-ACK: Máy chủ phản hồi bằng các gói SYN-ACK và chờ đợi gói ACK từ máy khách.
  - + Không nhận được ACK: Máy chủ không bao giờ nhận được gói ACK cuối cùng, khiến nó tiếp tục giữ các kết nối nửa chừng trong một khoảng thời gian nhất định.
- Tấn công HTTP Flood (Hình 6): HTTP Flood là một hình thức tấn công từ chối dịch vụ (DoS) hoặc tấn công từ chối dịch vụ phân tán (DDoS) nhắm trực tiếp vào tầng ứng

dụng của giao thức HTTP. Trong kiểu tấn công này, kẻ tấn công gửi một số lượng lớn yêu cầu HTTP giống như các truy cập hợp lệ, khiến máy chủ web phải xử lý quá nhiều request đồng thời. Khi tài nguyên bị tiêu hao vượt mức, máy chủ không còn khả năng đáp ứng các yêu cầu thật từ người dùng, dẫn đến chậm phản hồi hoặc ngừng hoạt động hoàn toàn.



Hình 6: Minh họa HTTP Flood Attack

Trong một cuộc tấn công HTTP Flood, kẻ tấn công gửi một lượng lớn các yêu cầu HTTP đến máy chủ mục tiêu, làm ngập nó với lưu lượng truy cập giả mạo. Các yêu cầu này có thể là:

- + HTTP GET requests: Kẻ tấn công gửi các yêu cầu GET để tải các trang web hoặc tài nguyên cụ thể.
- + HTTP POST requests: Kẻ tấn công gửi các yêu cầu POST để gửi dữ liệu đến máy chủ, thường là thông qua các biểu mẫu hoặc API.

***Đặc điểm của tấn công HTTP Flood:***

- + Tấn công tầng ứng dụng: Khác với các tấn công DoS hoặc DDoS ở tầng mạng, tấn công HTTP Flood nhằm vào tầng ứng dụng, nơi các yêu cầu HTTP được xử lý.
- + Khó phát hiện: Vì các yêu cầu HTTP trông giống như lưu lượng truy cập hợp lệ, việc phát hiện và phân biệt giữa lưu lượng truy cập hợp lệ và tấn công trở nên khó khăn.
- + Lợi dụng tài nguyên: Kẻ tấn công lợi dụng việc xử lý các yêu cầu HTTP tiêu tốn tài nguyên của máy chủ như CPU, bộ nhớ và băng thông mạng.

### ***Ví dụ về tấn công HTTP Flood:***

- + HTTP GET Flood: Trong dạng tấn công này, kẻ tấn công gửi một lượng rất lớn yêu cầu GET đến máy chủ web nhằm truy xuất các trang hoặc tài nguyên tĩnh. Việc xử lý số lượng lớn yêu cầu GET liên tục khiến máy chủ phải tiêu tốn nhiều tài nguyên, dẫn đến tình trạng quá tải và không thể phục vụ người dùng hợp lệ.
- + HTTP POST Flood: Kẻ tấn công gửi một lượng lớn các yêu cầu POST với dữ liệu lớn hoặc phức tạp đến máy chủ, khiến nó phải xử lý và lưu trữ dữ liệu này, từ đó làm chậm hoặc làm hỏng dịch vụ.

### ***Hậu quả của tấn công HTTP Flood:***

- + Từ chối dịch vụ: Khi máy chủ bị quá tải bởi các yêu cầu giả mạo, nó không thể xử lý các yêu cầu hợp lệ từ người dùng thực, dẫn đến tình trạng từ chối dịch vụ.
- + Giảm hiệu suất: Ngay cả khi máy chủ không bị hoàn toàn từ chối dịch vụ, hiệu suất của nó cũng sẽ giảm đáng kể, làm chậm quá trình xử lý yêu cầu và phản hồi người dùng.

### ***Biện pháp phòng chống:***

- + Giới hạn tốc độ yêu cầu: Áp dụng các giới hạn tốc độ yêu cầu trên máy chủ để giảm thiểu tác động của lưu lượng truy cập giả mạo.
- + Sử dụng tường lửa ứng dụng web (WAF): Sử dụng WAF để phát hiện và ngăn chặn các yêu cầu HTTP giả mạo.
- + Phân tán tài nguyên: Sử dụng các dịch vụ phân tán tài nguyên như CDN (Content Delivery Network) để giảm tải cho máy chủ gốc.

### ***Một số kỹ thuật giảm nhẹ có thể được sử dụng, như:***

- + Định Tuyển Hồ Đen: Trong định tuyển hồ đen, lưu lượng mạng được điều hướng đến một "hồ đen," nơi cả lưu lượng độc hại và không độc hại đều bị mất. Biện pháp này hiệu quả khi một máy chủ đang phải đối mặt với một cuộc tấn công DDoS, chuyển hết lưu lượng để duy trì thời gian hoạt động của mạng.
- + Giới Hạn Tốc Độ: Điều này liên quan đến việc kiểm soát tốc độ lưu lượng được gửi hoặc nhận bởi một giao diện mạng. Nó đặc biệt hiệu quả trong việc làm chậm các công cụ thu thập dữ liệu web và các nỗ lực đăng nhập mật khẩu mạnh. Tuy nhiên, việc giới

hạn tốc độ một mình có thể không đủ để ngăn chặn các cuộc tấn công DDoS phức tạp.

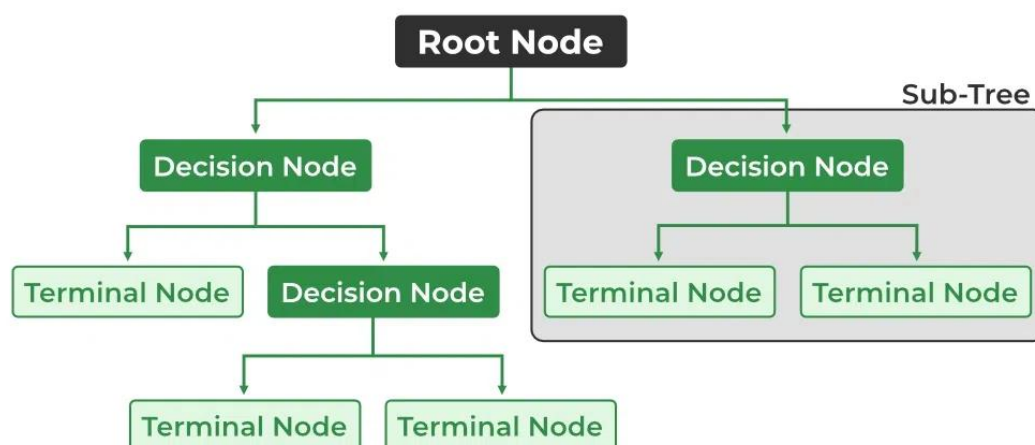
+ **Cấm/Cho Truy Cập Theo Danh Sách Đen/Trắng:** Cấm truy cập liên quan đến việc chặn các địa chỉ IP, URL, tên miền, vv., được liệt kê như là mối đe dọa, trong khi cho phép lưu lượng từ các nguồn khác. Ngược lại, việc cho truy cập theo danh sách trắng chỉ cho phép các địa chỉ IP, URL, tên miền, vv., được chỉ định trong danh sách, từ chối quyền truy cập của tất cả các nguồn khác.

## 2.3 Thuật toán decision tree

Cây quyết định (Decision Tree) là một thuật toán học có giám sát được sử dụng để phân loại dữ liệu bằng cách xây dựng một cấu trúc cây logic. Mỗi nút trong cây đại diện cho một điều kiện kiểm tra trên một đặc trưng (feature), mỗi nhánh biểu diễn kết quả của điều kiện đó, và các nút lá chứa nhãn lớp cuối cùng. Trong bài toán phát hiện tấn công DDoS, cây quyết định được sử dụng để phân loại luồng mạng thành hai lớp: lưu lượng bình thường (normal) hoặc lưu lượng tấn công (ddos).

Để giải quyết vấn đề overfitting của cây đơn lẻ và tăng độ chính xác, Random Forest được sử dụng – một phương pháp tập hợp (ensemble method) kết hợp 300 cây quyết định độc lập hoạt động song song.

### 2.3.1 Cấu Trúc Của Cây Quyết Định:



## Hình 7: Sơ đồ cấu trúc cây quyết định

Hình 15 mô tả cấu trúc cơ bản của thuật toán Decision Tree. Một số thuật ngữ được giải thích như sau:

- **Root Node:** Là nút ở trên cùng của cây, đại diện cho toàn bộ tập dữ liệu. Đây là điểm khởi đầu của quá trình ra quyết định.
- **Decision/Internal Node:** Nút biểu thị một lựa chọn liên quan đến một đặc trưng đầu vào. Việc phân nhánh từ các nút nội bộ kết nối chúng với các nút lá hoặc các nút nội bộ khác.
- **Leaf/Terminal Node:** Nút không có nút con, chỉ ra một nhãn lớp hoặc một giá trị số.
- **Splitting:** Quá trình chia một nút thành hai hoặc nhiều nút con bằng cách sử dụng tiêu chí chia tách và một đặc trưng được chọn.
- **Branch/Sub-Tree:** Một phần nhỏ của cây quyết định bắt đầu từ một nút nội bộ và kết thúc tại các nút lá.

### 2.3.2 Tiêu chí chia tách: Gini Impurity:

Trong quá trình xây dựng cây, thuật toán phải chọn phép chia tách tối ưu tại mỗi nút. Gini Impurity là tiêu chí được sử dụng để đánh giá mức độ thuần nhất của dữ liệu sau mỗi phép chia.

Gini Impurity đo lường mức độ hỗn loạn của phân phối lớp trong một tập dữ liệu con. Nó phản ánh xác suất một mẫu được chọn ngẫu nhiên sẽ bị phân loại sai nếu được gán nhãn theo phân phối lớp hiện tại.

**Công thức:** 
$$\text{Gini}(S) = 1 - \sum_{i=1}^c p_i^2$$

Trong đó:

- S là tập dữ liệu tại nút hiện tại
- $p_i$  là tỷ lệ (xác suất) của lớp i trong tập S
- c là số lượng lớp (trong trường hợp phát hiện DDoS,  $c = 2$ : normal và ddos)

Giá trị Gini Impurity nằm trong khoảng từ 0 đến 0.5 (đối với hai lớp). Giá trị 0 chỉ ra rằng nút là hoàn toàn thuần nhất (pure), tức là tất cả các mẫu trong nút đều thuộc cùng một lớp, không cần chia tách thêm. Giá trị 0.5 chỉ ra rằng nút có độ hỗn loạn cao nhất, có nghĩa là

hai lớp phân bố đều bằng nhau.

Thuật toán lựa chọn phép chia tách sao cho mức giảm Gini Impurity (Gini gain) là lớn nhất. Điều này có nghĩa là phép chia tách tốt nhất là phép chia mà làm cho các nút con trở nên thuần nhất hơn so với nút cha.

Gini Impurity được lựa chọn vì nó có tính toán đơn giản (chỉ sử dụng bình phương, không có logarithm), khiến nó nhanh hơn các tiêu chí khác như Entropy, đặc biệt là khi cần huấn luyện nhiều cây như trong Random Forest.

### **2.3.3 Quá Trình Xây Dựng Cây Quyết Định (Decision Tree):**

Quá trình xây dựng cây quyết định tuân theo một thuật toán phân vùng tổng hợp (recursive partitioning) nhằm tối ưu hóa việc phân chia dữ liệu dần dần.

- Bước 1 - Chọn Đặc Trưng Tối Ưu: Tại nút hiện tại, thuật toán kiểm tra toàn bộ các đặc trưng khả dụng. Đối với mỗi đặc trưng, thuật toán tìm ngưỡng (threshold) tối ưu sao cho phép chia dữ liệu theo ngưỡng đó sẽ mang lại mức giảm Gini Impurity (hay Gini gain) cao nhất. Đặc trưng và ngưỡng tốt nhất (cho phép đạt Gini gain lớn nhất) sẽ được lựa chọn cho nút này.
- Bước 2 - Thực Hiện Phép Chia Tách: Dữ liệu được chia thành hai tập con dựa trên điều kiện ở bước 1. Các mẫu thỏa mãn điều kiện sẽ đi vào nút con trái (left child), còn lại sẽ đi vào nút con phải (right child). Mỗi nút con giờ đây chứa dữ liệu có độ thuần nhất cao hơn so với nút cha.
- Bước 3 - Lặp Lại Quá Trình Đề Quy: Thuật toán áp dụng bước 1 và 2 cho từng nút con một cách đệ quy. Mỗi nút sẽ tiếp tục bị chia cho đến khi đạt một điều kiện dừng:
  - Nút trở nên hoàn toàn thuần nhất (tất cả mẫu thuộc cùng một lớp), hoặc
  - Độ sâu của cây đạt giới hạn tối đa (max\_depth), hoặc
  - Gini gain bằng 0 hoặc rất nhỏ (không có lợi ích từ phép chia), hoặc
  - Số lượng mẫu còn lại tại nút nhỏ hơn một ngưỡng tối thiểu.
- Bước 4 - Tạo Nút Lá: Khi một nút không thể hoặc không nên chia tách thêm, nó trở thành nút lá. Nhãn lớp của nút lá được gán là lớp xuất hiện nhiều nhất trong tập mẫu tại nút đó. Để dự đoán một mẫu mới, ta bắt đầu từ nút gốc, theo các điều kiện tại mỗi nút cho đến khi đạt được một nút lá, và nhãn của nút lá đó chính là dự đoán.

### 2.3.4 Độ Quan Trọng Của Đặc Trưng (Feature Importance):

Sau khi huấn luyện xong cây, có thể tính toán mức độ quan trọng của mỗi đặc trưng trong quá trình dự đoán. Một đặc trưng được coi là quan trọng nếu nó được sử dụng để chia tách ở các nút gần gốc và mang lại mức giảm Gini Impurity đáng kể.

Độ quan trọng của một đặc trưng được tính bằng cách tổng hợp tất cả các mức giảm Gini Impurity (Gini gain) mà đặc trưng đó mang lại trong toàn bộ cây. Đặc trưng nào có tổng Gini gain cao nhất sẽ có độ quan trọng cao nhất. Điều này có ý nghĩa thực tiễn lớn: nó giúp ta xác định những đặc trưng nào của luồng mạng là chỉ báo quan trọng nhất để phân biệt giữa lưu lượng bình thường và lưu lượng tấn công DDoS.

### 2.3.5 Vấn Đề Overfitting Trong Cây Quyết Định:

Overfitting là hiện tượng mô hình học quá kỹ chi tiết của dữ liệu huấn luyện, bao gồm cả những nhiễu (noise) và bất thường, thay vì học những quy luật và mô hình tổng quát. Khi xảy ra overfitting, mô hình sẽ hoạt động rất tốt trên dữ liệu huấn luyện nhưng hiệu suất sẽ giảm đáng kể trên dữ liệu mới hoặc dữ liệu kiểm tra (test data).

Cây quyết định đơn lẻ có xu hướng dễ overfitting, đặc biệt là khi cây được phép phát triển quá sâu và chi tiết. Một cây quá sâu có thể tạo ra các quy tắc phân loại quá phức tạp, mỗi quy tắc áp dụng chỉ cho một số ít mẫu huấn luyện, dẫn đến việc mô hình không khái quát tốt đối với dữ liệu không nhìn thấy trước.

Có hai phương pháp chính để giải quyết vấn đề overfitting:

- Pre-pruning (Cắt Tỉa Sớm): Phương pháp này dừng quá trình xây dựng cây trước khi nó phát triển quá sâu. Bằng cách đặt các giới hạn như độ sâu tối đa (max\_depth), số lượng mẫu tối thiểu cần thiết để thực hiện phép chia ở một nút (min\_samples\_split), hoặc số lượng mẫu tối thiểu cần có ở một nút lá (min\_samples\_leaf), ta có thể kiểm soát độ phức tạp của cây. Pre-pruning đơn giản, hiệu quả về mặt tính toán, nhưng có thể dừng lại quá sớm.
- Post-pruning (Cắt Tỉa Sau): Phương pháp này cho phép cây phát triển hoàn toàn (hoặc gần như hoàn toàn), sau đó loại bỏ các nhánh hoặc nút mà không giúp cải thiện hiệu suất trên một tập dữ liệu xác thực hoặc dựa trên một tiêu chí đánh giá. Post-pruning thường cho kết quả tốt hơn nhưng đòi hỏi chi phí tính toán lớn hơn.

### **2.3.6 Random Forest - Phương Pháp Tập Hợp để Giải Quyết Overfitting:**

Random Forest là một phương pháp học tập tập hợp (ensemble learning method) được thiết kế để giải quyết một số nhược điểm của cây quyết định đơn lẻ, đặc biệt là vấn đề overfitting và sự không ổn định.

Thay vì xây dựng một cây quyết định duy nhất, Random Forest xây dựng nhiều cây quyết định độc lập (thường hàng trăm hoặc hàng ngàn). Mỗi cây được huấn luyện trên một mẫu bootstrap từ dữ liệu gốc – tức là một tập mẫu được lấy ngẫu nhiên từ dữ liệu huấn luyện với phép lặp lại (sampling with replacement). Do cách lấy mẫu này, mỗi cây sẽ nhìn thấy một phiên bản hơi khác nhau của dữ liệu huấn luyện, dẫn đến các cây khác nhau.

Khi dự đoán một mẫu mới, Random Forest cho tất cả các cây dự đoán kết quả của chúng. Đối với bài toán phân loại như phát hiện DDoS, kết quả cuối cùng được quyết định bằng cách bỏ phiếu đa số (majority voting): lớp nào được các cây vote nhiều nhất sẽ là dự đoán cuối cùng.

#### ***Ưu điểm của Random Forest:***

- Giảm Phương Sai (Variance): Bằng cách kết hợp dự đoán từ nhiều cây, Random Forest giảm thiểu phương sai, do đó giảm thiểu hiệu ứng overfitting. Các lỗi ngẫu nhiên của các cây có xu hướng triệt tiêu lẫn nhau.
- Tăng Độ Chính Xác: Tổng thể, Random Forest thường đạt độ chính xác cao hơn so với một cây đơn lẻ, nhất là khi các cây trong rừng có sự đa dạng.
- Ổn Định Hơn: Random Forest ít bị ảnh hưởng bởi những thay đổi nhỏ trong dữ liệu huấn luyện. Nếu một vài mẫu thay đổi, nhiều cây sẽ vẫn đưa ra dự đoán chính xác.
- Cung Cấp Độ Quan Trọng Đặc Trưng: Random Forest có thể tính độ quan trọng của từng đặc trưng bằng cách tổng hợp độ quan trọng từ tất cả các cây. Điều này giúp hiểu rõ hơn tầm quan trọng của các đặc trưng khác nhau.
- Xử Lý Dữ Liệu Không Tuyến Tính: Random Forest có khả năng bắt giữ các mối quan hệ phi tuyến tính trong dữ liệu mà các mô hình tuyến tính (như hồi quy logistic) không thể.

#### ***Nhược điểm:***

- Chậm Hơn: Việc huấn luyện và dự đoán với Random Forest chậm hơn so với cây đơn lẻ, do phải xử lý nhiều cây.
- Khó Giải Thích Hơn: Một cây đơn lẻ dễ hình dung và giải thích, nhưng 300 cây hoạt động đồng thời rất khó để hiểu chi tiết các quyết định.

### **2.3.7 Cân Bằng Lớp (Class Balance) và Hiệu Suất Mô Hình:**

Để một mô hình phân loại như Random Forest hoạt động tốt, đặc biệt là trong các tác vụ phát hiện bất thường như DDoS detection, dữ liệu huấn luyện cần phải cân bằng giữa các lớp. Dữ liệu cân bằng có nghĩa là số lượng mẫu của mỗi lớp là tương đương nhau.

Khi dữ liệu không cân bằng (imbalanced), ví dụ như 90% lớp lệnh lưu lượng bình thường (label=0) và chỉ 10% là lưu lượng tấn công (label=1), mô hình sẽ có xu hướng học cách phân loại hầu hết các mẫu thành lớp đa số. Điều này là do các thuật toán học máy thường tối ưu hóa độ chính xác tổng thể, và nếu phân loại mọi thứ thành "bình thường" (lớp đa số), độ chính xác sẽ rất cao (90%), nhưng khả năng phát hiện tấn công sẽ cực kỳ tệ (False Negative rất cao).

Đối với bài toán phát hiện tấn công DDoS, việc bỏ sót các tấn công (False Negative) là vô cùng nguy hiểm hơn việc báo động giả (False Positive). Do đó, dữ liệu huấn luyện cần chứa đủ lượng mẫu tấn công để mô hình có thể học và nhận diện các quy tắc phân biệt giữa lưu lượng bình thường và tấn công.

Nếu dữ liệu huấn luyện chỉ chứa toàn bộ lưu lượng bình thường (label=0) mà không có bất kỳ lưu lượng tấn công (label=1) nào, mô hình sẽ không có khả năng học các đặc điểm của tấn công, và do đó sẽ không thể phát hiện tấn công mới.

## CHƯƠNG 3: MÔ HÌNH ĐỀ XUẤT

### 3.1 Tổng Quan Hệ Thống

Hệ thống phát hiện tấn công DDoS (Distributed Denial-of-Service) được xây dựng dựa trên kiến trúc SDN (Software Defined Networking) sử dụng Floodlight controller. Hệ thống bao gồm ba thành phần chính:

1. Bộ Thu Thập Dữ Liệu (Flow Collector): Lấy thông tin luồng mạng từ Floodlight controller
2. Bộ Tiền Xử Lý và Trích Xuất Đặc Trưng (Feature Engineering): Chuyển đổi dữ liệu thô thành các đặc trưng có ý nghĩa
3. Mô Hình Học Máy (ML Model): Sử dụng Random Forest để phân loại luồng mạng

### 3.2 Kiến Trúc Chi Tiết

#### 3.2.1 Bộ Thu Thập Dữ Liệu (Collector):

**Chức năng:** Thực hiện polling liên tục tới Floodlight REST API để lấy thông tin luồng mạng từ các switch OpenFlow.

**Quy trình hoạt động:**

1. Kết nối tới Floodlight: Collector gửi HTTP GET request đến endpoint `/wm/core/switch/all/flow/json`` trên Floodlight controller (mặc định: `http://127.0.0.1:8080``)
2. Lấy dữ liệu flow: Server trả về JSON chứa danh sách các switch và các flow rules đang active trên mỗi switch.
3. Phân tích từng flow: Đối với mỗi flow, collector trích xuất:
4. Lọc flow IPv4:
5. Ghi CSV: Các flow đã lọc được ghi vào file `FlowStatsfile.csv`` với:

Polling Interval: Mặc định mỗi 10 giây thu thập một lần (có thể điều chỉnh bằng `--interval``)

### ***3.2.2 Trích Xuất Đặc Trưng (Feature Engineering):***

**Mục đích:** Chuyển đổi các trường dữ liệu thô từ OpenFlow flow statistics thành các đặc trưng (features) có ý nghĩa cho mô hình học máy.

***Các đặc trưng được tạo:***

- Đặc Trưng Từ Thống Kê Thô:
- Đặc Trưng Dẫn Xuất:
- Đặc Trưng Phân Loại:

Tổng cộng: 17 đặc trưng được sử dụng để huấn luyện mô hình.

***Xử lý Giá Trị Bất Thường:***

- Giá trị NaN/Inf được thay thế bằng median của cột tương ứng
- Các cột bị skew cao (skewness > 1.2) được áp dụng log1p transformation để làm phẳng phân phối
- Tất cả các đặc trưng được chuẩn hóa bằng StandardScaler (mean=0, std=1)

### ***3.3.2 Mô Hình Học Máy (Machine Learning Model):***

**Thuật toán:** Random Forest Classifier với CalibratedClassifierCV

***Quy Trình Huấn Luyện:***

- Bước 1 - Chia dữ liệu
- Bước 2 - Cross-Validation
- Bước 3 - Tính độ quan trọng đặc trưng
- Bước 4 - Huấn luyện mô hình cuối
- Bước 5 - Tối ưu threshold

***Model Persistence:***

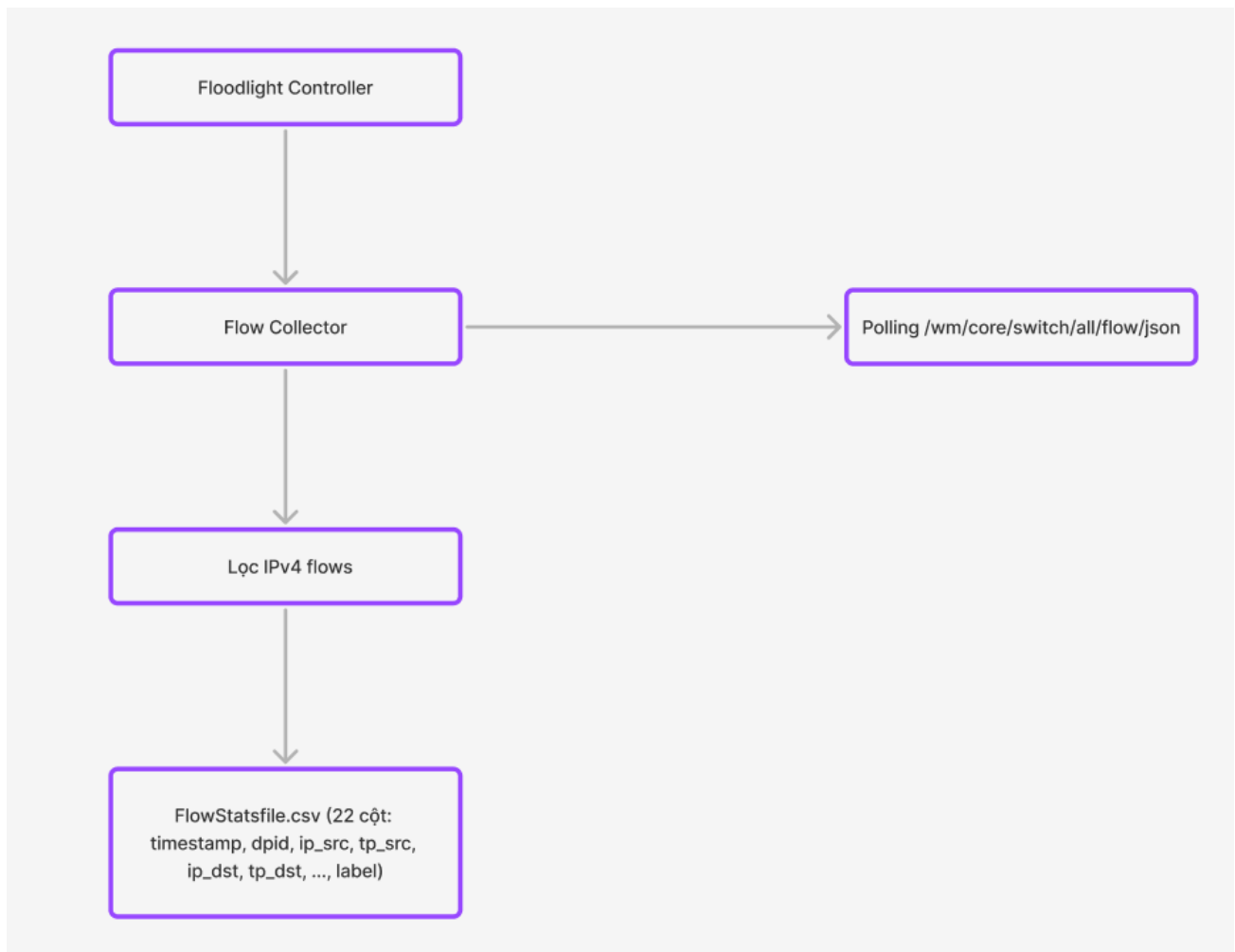
- Model được lưu vào `model.pkl`
- Metadata được lưu vào `metadata.pkl`:
- Danh sách đặc trưng (features)

- Giá trị median của mỗi đặc trưng
- StandardScaler object (để chuẩn hóa dữ liệu test)
- Các cột cần log1p transform
- Optimal threshold (Ngưỡng tối ưu)

### 3.3 Quy Trình End-to-End

Quy trình end-to-end mô tả toàn bộ vòng đời của mô hình từ lúc bắt đầu cho đến khi phát hiện DDoS:

#### 3.3.1 Chuẩn Bị Dữ Liệu:



Hình 8: Chuẩn Bị Dữ Liệu

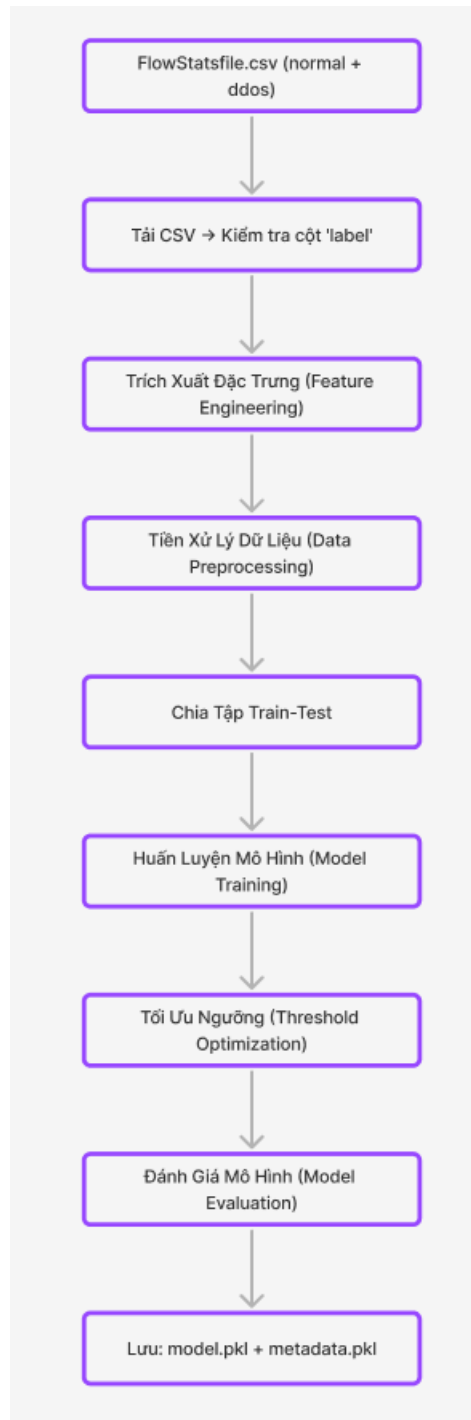
### ***Chi tiết:***

- Collector gửi HTTP gửi yêu cầu tới ``/wm/core/switch/all/flow/json``
- Floodlight trả về JSON với tất cả flows trên tất cả switches
- Collector lọc ra chỉ những flows IPv4 (`eth_type=0x800`)
- Ghi thêm label: ``--label 0`` (normal traffic) hoặc ``--label 1`` (DDoS traffic)

### ***Yêu cầu:***

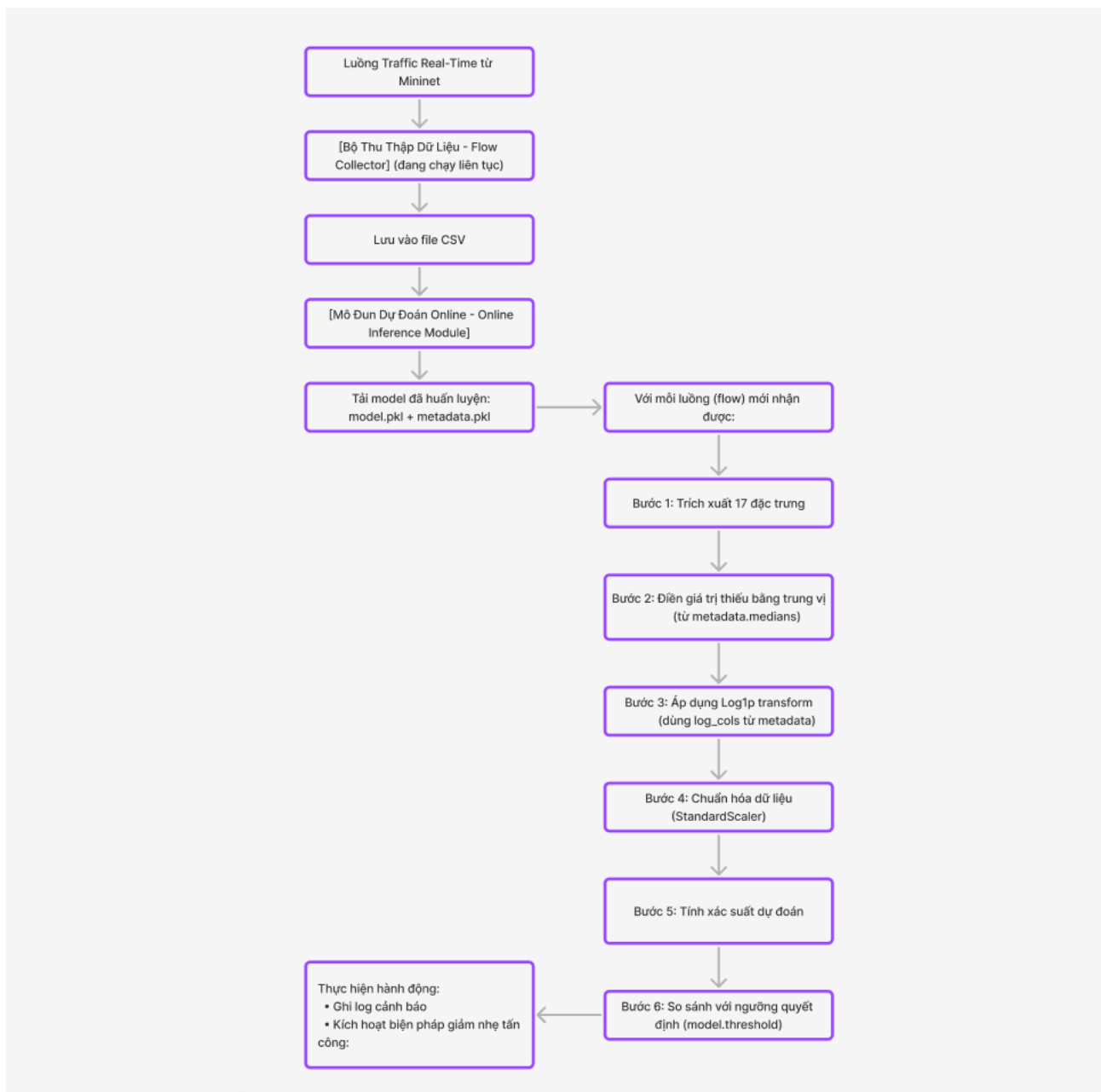
- Hệ thống Mininet + Floodlight controller đang chạy
- Topology với nhiều hosts/switches
- Phase Normal: Chạy ``collect_training_stats_floodlight.py --label 0`` + traffic generation (`generate_normal_traffic.py`)
- Phase DDoS: Chạy ``collect_training_stats_floodlight.py --label 1`` + DDoS attack (`generate_ddos_traffic.py`)
- Kết quả: FlowStatsfile.csv chứa cả normal (label = 0) và DDoS (label = 1)
- Giai Đoạn 2: Huấn Luyện Mô Hình (Model Training)

### ***3.3.2 Huấn Luyện Mô Hình (Model Training):***



Hình 9: Huấn Luyện Mô Hình

### 3.3.3 Dự Đoán Real-Time:



Hình 10: Dự Đoán Real-Time

## CHƯƠNG 4: MÔ PHỎNG

### 4.1 Thu thập dữ liệu cho dataset

#### 4.1.1 Thu thập lưu lượng bình thường qua mạng

Ta khởi chạy 2 file tương ứng là:

Collect\_training\_stats\_floodlight.py

Generate\_normal\_traffic.py

Hình ảnh minh họa chạy lần lượt các file:

```
root@Ubuntu:/media/sf_Shared_VM_ubuntu_22.04_files/source# python3 controller/collect_training_stats_floodlight.py --interval 5 --label 0
```

Tiếp theo là đến file tạo lưu lượng bình thường:

```
root@Ubuntu:/media/sf_Shared_VM_ubuntu_22.04_files/source# python3 mininet/generate_normal_traffic.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18
*** Adding switches:
s1 s2 s3 s4 s5 s6
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s2) (h5, s2) (h6, s2) (h7, s3) (h8, s3) (h9, s3)
(h10, s4) (h11, s4) (h12, s4) (h13, s5) (h14, s5) (h15, s5) (h16, s6) (h17, s6)
(h18, s6) (s1, s2) (s2, s3) (s3, s4) (s4, s5) (s5, s6)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18
*** Starting controller
c0
*** Starting 6 switches
s1 s2 s3 s4 s5 s6 ...
-----
-
Generating traffic ...
-----
-
Iteration n 1 ...
-----
-
generating ICMP traffic between h12 and h4 and TCP/UDP traffic between h12 and h1
h12 Downloading index.html from h1
h12 Downloading test.zip from h1
generating ICMP traffic between h8 and h12 and TCP/UDP traffic between h8 and h1
1
```

Khi file thu thập phát hiện lưu lượng tồn tại, nó sẽ bắt đầu ghi vào trong file dataset (file .csv):

```
Appended 18 flow rows (label=0)
```

**Lưu ý:** Ta tiến hành quá trình chạy 2 file này trong khoảng 5-10 phút, để đảm bảo lưu lượng bình thường được thu thập đủ để đáp ứng nhu cầu huấn luyện mô hình của ta.

#### 4.1.2 Thu thập lưu lượng giả lập DDOS qua mạng

Ta khởi chạy 2 file tương ứng là:

- Collect\_training\_stats\_floodlight.py
- Generate\_ddos\_traffic.py

Hình ảnh minh họa chạy lần lượt các file:

```
root@Ubuntu:/media/sf_Shared_VM_ubuntu_22.04_files/source# python3 controller/c
ollect_training_stats_floodlight.py --interval 5 --label 1
```

Tiếp theo là đến file tạo lưu lượng DDOS:

```
root@Ubuntu:/media/sf_Shared_VM_ubuntu_22.04_files/source# python3 mininet/gene
rate_ddos_traffic.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18
*** Adding switches:
s1 s2 s3 s4 s5 s6
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s2) (h5, s2) (h6, s2) (h7, s3) (h8, s3) (h9, s3
) (h10, s4) (h11, s4) (h12, s4) (h13, s5) (h14, s5) (h15, s5) (h16, s6) (h17, s
6) (h18, s6) (s1, s2) (s2, s3) (s3, s4) (s4, s5) (s5, s6)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18
*** Starting controller
c0
*** Starting 6 switches
s1 s2 s3 s4 s5 s6 ...
[*] Starting simple http.server on victim h2 (port 80)
[*] Selected attackers: ['h16', 'h3', 'h6', 'h8', 'h13']
[*] Starting tcpdump on victim h2 -> /tmp/h2_http_capture.pcap
[*] Launching ddos on attackers (concurrency per attacker = 10), duration_arg='
INFINITE'
- starting h16: /tmp/ddos_no_flood.sh 10.0.0.2 '' 10 &
- starting h3: /tmp/ddos_no_flood.sh 10.0.0.2 '' 10 &
- starting h6: /tmp/ddos_no_flood.sh 10.0.0.2 '' 10 &
- starting h8: /tmp/ddos_no_flood.sh 10.0.0.2 '' 10 &
- starting h13: /tmp/ddos_no_flood.sh 10.0.0.2 '' 10 &
[*] Attack running INFINITELY. Press Ctrl-C here to stop and cleanup.
^C[*] Cleanup: killing ddos scripts and curl on attackers, stopping tcpdump and
http.server on victim.
*** Stopping 1 controllers
c0
*** Stopping 23 links
.....
*** Stopping 6 switches
s1 s2 s3 s4 s5 s6
*** Stopping 18 hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18
*** Done
```

Khi file thu thập phát hiện lưu lượng tồn tại, nó sẽ bắt đầu ghi vào trong file dataset (file .csv):

```
Appended 7737 flow rows (label=1)
^CCollector stopped.
```

## 4.2 Huấn luyện model machine learning

Ta khởi chạy file tương ứng là:

- ML\_trainer.py

Các tham số ta có thể truyền vào trước khi chạy file:

```
root@Ubuntu:/media/sf_Shared_VM_ubuntu_22.04_files/source# python3 machinelearning/ML_trainer.py -h
usage: ML_trainer.py [-h] --csv CSV [--out_model OUT_MODEL] [--out_meta OUT_META]

options:
  -h, --help            show this help message and exit
  --csv CSV              Input CSV file (FlowStatsfile.csv from collector)
  --out_model OUT_MODEL
  --out_meta OUT_META
```

Khi khởi chạy file với các tham số truyền vào thích hợp, ta sẽ nhận được output sau:

```
root@Ubuntu:/media/sf_Shared_VM_ubuntu_22.04_files/source# python3 machinelearning/ML_trainer.py -h
usage: ML_trainer.py [-h] --csv CSV [--out_model OUT_MODEL] [--out_meta OUT_META]

options:
  -h, --help            show this help message and exit
  --csv CSV              Input CSV file (FlowStatsfile.csv from collector)
  --out_model OUT_MODEL
  --out_meta OUT_META
root@Ubuntu:/media/sf_Shared_VM_ubuntu_22.04_files/source# python3 machinelearning/ML_trainer.py --csv output/FlowStatsfile.csv
2025-11-22 16:29:13,014 | INFO | CV ROC AUC (5-fold): mean=0.9913 std=0.0122
2025-11-22 16:29:18,008 | INFO | Top Features: [('byte_count', np.float64(0.25086043974763117)), ('packet_count', np.float64(0.1841687270000303)), ('flow_durat
lon', np.float64(0.12857690555828155)), ('dst_port', np.float64(0.07809550088052616)), ('byte_count_per_nsecond', np.float64(0.07321931795411245)), ('src_port'
, np.float64(0.06551805220786265)), ('packet_count_per_nsecond', np.float64(0.06256770108825097)), ('flow_bytes_per_s', np.float64(0.05372791023898382)), ('flo
w_pkts_per_s', np.float64(0.05136941817009056)), ('avg_pkt_size', np.float64(0.03301524112848672))]
2025-11-22 16:29:53,452 | INFO | Accuracy: 0.9987
2025-11-22 16:29:53,452 | INFO | AUC: 1.0000
2025-11-22 16:29:53,453 | INFO | Confusion Matrix:
[[ 934   3]
 [  2 2920]]
2025-11-22 16:29:53,476 | INFO | Classification Report:
      precision    recall  f1-score   support

     0         1.00      1.00      1.00         937
     1         1.00      1.00      1.00        2922

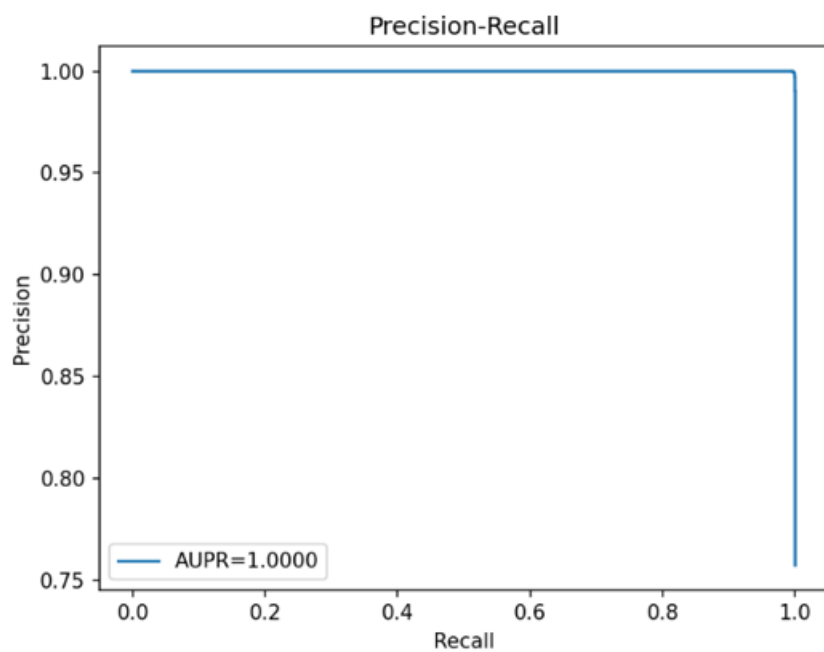
   accuracy          1.00
  macro avg          1.00
 weighted avg          1.00
2025-11-22 16:29:54,163 | INFO | Selected threshold=0.370 (F1=0.999)
2025-11-22 16:29:57,510 | INFO | Saved plots: model_eval_{roc,pr,cm}.png
2025-11-22 16:30:03,458 | INFO | Saved model to model.pkl
2025-11-22 16:30:03,458 | INFO | Saved metadata to metadata.pkl
2025-11-22 16:30:03,459 | INFO | Training complete.
2025-11-22 16:30:03,459 | INFO | Final ACC=0.9987 | AUC=1.0000
```

Sau khi check lại các file được output ra từ quá trình chạy, ta được (các file màu xanh lá):

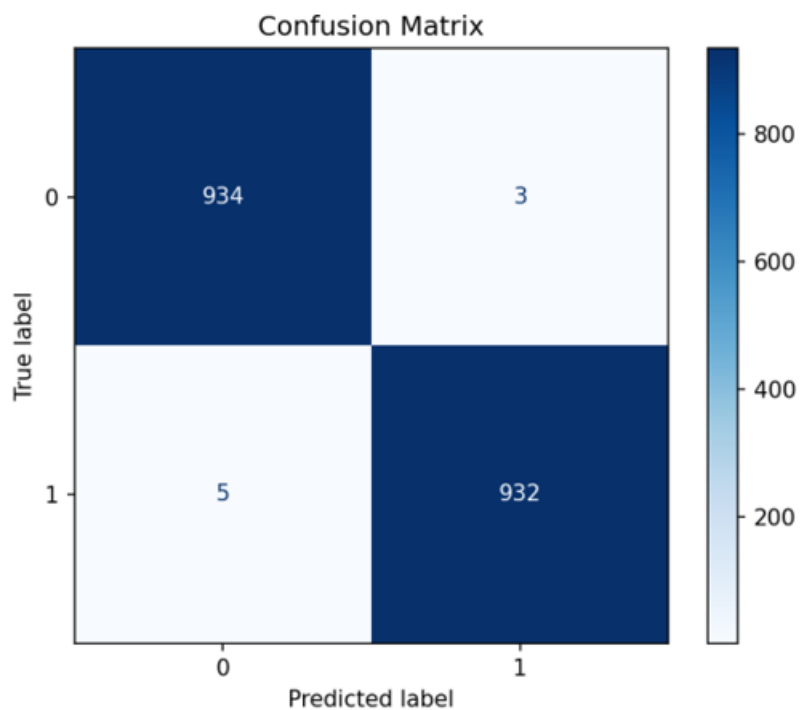
```
root@Ubuntu:/media/sf_Shared_VM_ubuntu_22.04_files/source# ls
backup_ML      feature_importances.csv  metadata.pkl  model_eval_cm.png  model_eval_roc.png  output
controller    machinelearning          mininet      model_eval_pr.png  model.pkl            venv
```

Trong đó 3 file có đuôi “.png” chính là các hình ảnh minh họa cho mức độ hiệu quả nhận dạng của mô hình ML này dựa trên dataset ta tạo từ trước.

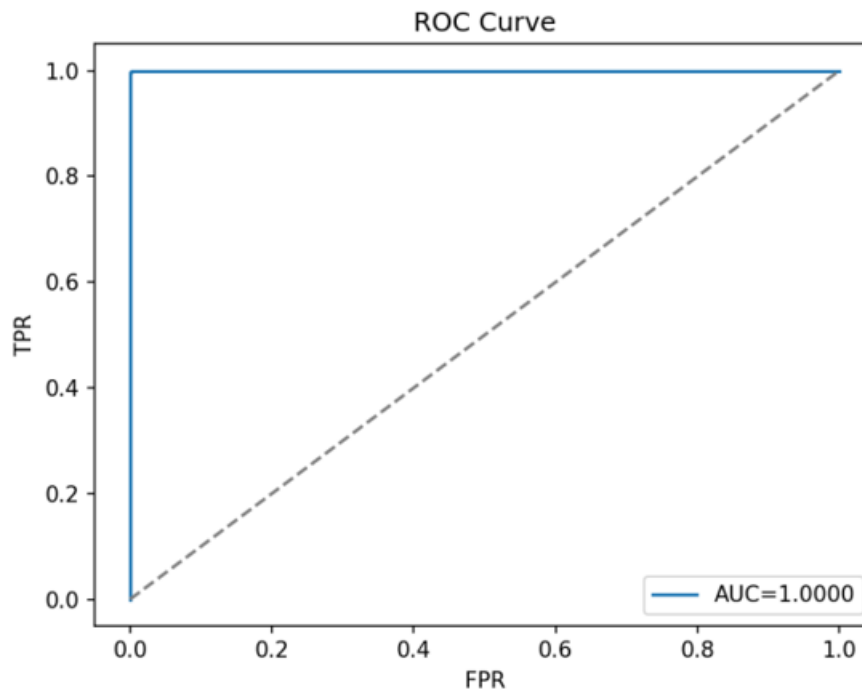
Hình ảnh của các file “.png”, đầu tiên là model\_eval\_pr.png:



File model\_eval\_cm.png:



File model\_eval\_roc.png



Còn 2 file “màu xanh lá” còn lại khi train xong là:

Model.pkl: Chứa mô hình học máy đã train, dùng cho inference: load bằng pickle rồi gọi model.predict / model.predict\_proba để phân loại lưu lượng.

Metadata.pkl: Chứa dữ liệu tiền xử lý và siêu thông tin cần thiết để áp pipeline inference giống lúc train: keys: "features" (list tên feature), "medians" (điền khuyết), "scaler" (StandardScaler object), "log\_cols" (các cột apply log1p), "threshold" (ngưỡng xác suất → nhãn). Dùng để chuyển đổi dòng flow mới (impute, log transform, scale) rồi map sang tập feature đúng thứ tự trước khi feed vào model.

## 4.3 Giả lập một webserver và áp dụng machine learning để phát hiện và ngăn chặn DDOS.

### 4.3.1 Yêu cầu cơ bản trước khi chạy

Sau khi huấn luyện xong, để tiến hành chạy mô phỏng quy trình trên, ta cần 3 terminal mở ở thư mục root (/source) như sau:

- Terminal A: gọi file để chạy ML trong thời gian thực để phát hiện và block traffic DDOS.
- Terminal B: chạy file thu thập dữ liệu thời gian thực và ghi vào file “PredictFlowStatsfile.csv” để đút cho ML đang chạy bên terminal A, từ đó mô hình có thể đánh giá đâu là traffic DDOS.
- Terminal C: chạy lệnh để khởi tạo 1 topology để mô phỏng mạng lưới các thiết bị. Sau đó trong terminal này, ta chạy các giao diện terminal nhỏ khác cho các host của các thiết bị.

**Lưu ý:** Để chạy được quy trình này một cách chính xác và ổn định, phải đảm bảo đã thành công cài đặt được Floodlight bản 1.2 trên Ubuntu 22.04, chạy Python 3.10 và dùng JDK 1.8

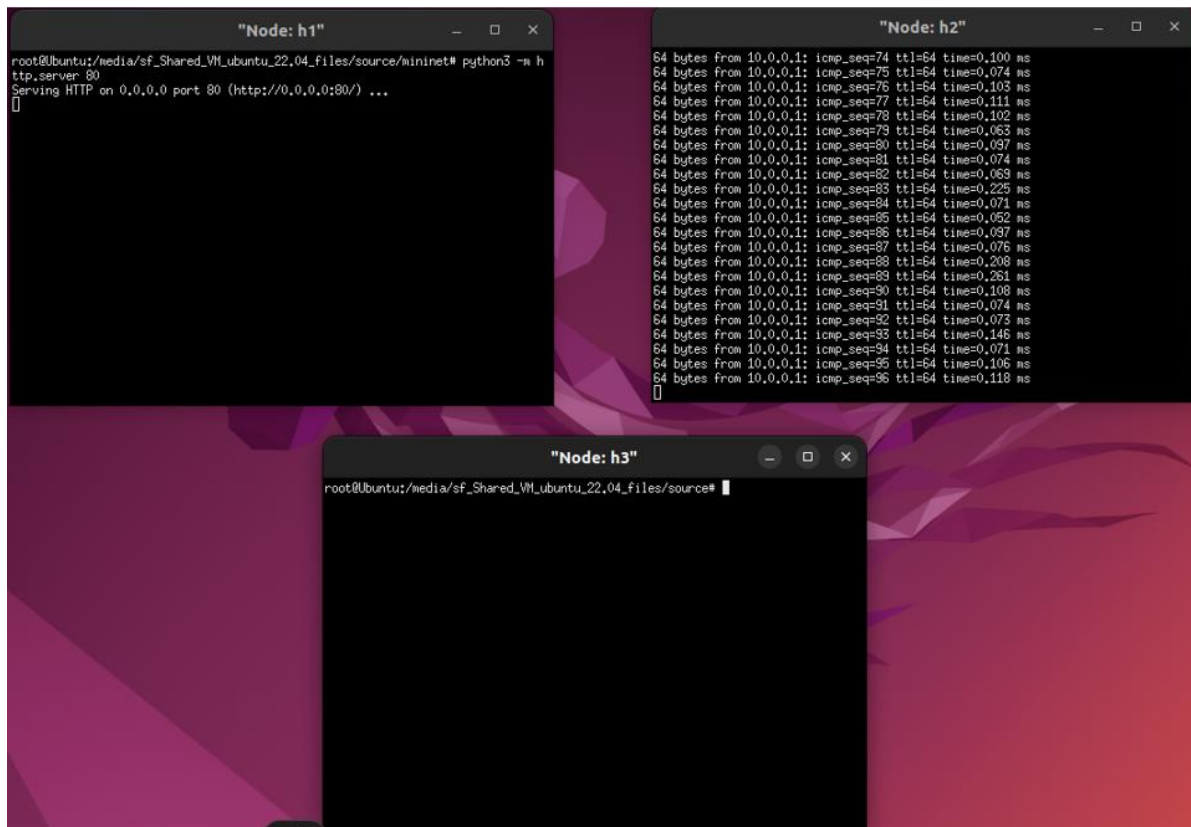
Sau đó chạy lệnh để khởi động Floodlight trên *http://localhost:8080*, để mô hình ML có thể đẩy các rule chặn port lên API của Floodlight.

### 4.3.2 Toàn bộ quá trình ngăn chặn DDOS

Đầu tiên ta chạy file topology.py bên terminal C để tạo 1 mạng lưới mô phỏng.

```
root@Ubuntu:/media/sf_Shared_VM_ubuntu_22.04_files/source# python3 mininet/topology.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(10.00Mbit) (10.00Mbit) (h1, s1) (10.00Mbit) (10.00Mbit) (h2, s1) (10.00Mbit) (
10.00Mbit) (h3, s1) (10.00Mbit) (10.00Mbit) (h4, s2) (10.00Mbit) (10.00Mbit) (h
5, s2) (10.00Mbit) (10.00Mbit) (h6, s2) (10.00Mbit) (10.00Mbit) (h7, s3) (10.00
Mbit) (10.00Mbit) (h8, s3) (10.00Mbit) (10.00Mbit) (h9, s3) (10.00Mbit) (10.00M
bit) (h10, s4) (10.00Mbit) (10.00Mbit) (h11, s4) (10.00Mbit) (10.00Mbit) (h12,
s4) (s1, s2) (s2, s3) (s3, s4)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ... (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.
00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit)
*** Starting CLI:
mininet>
```

Sau đó ta tạo 3 terminal nhỏ bằng công cụ *Xterm* để giả lập 3 host trên mạng lưới.



```
"Node: h1"
root@Ubuntu:/media/sf_Shared_VM_ubuntu_22_04_files/source/mininet# python3 -m h
tup.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...

"Node: h2"
64 bytes from 10.0.0.1: icmp_seq=74 ttl=64 time=0.100 ms
64 bytes from 10.0.0.1: icmp_seq=75 ttl=64 time=0.074 ms
64 bytes from 10.0.0.1: icmp_seq=76 ttl=64 time=0.103 ms
64 bytes from 10.0.0.1: icmp_seq=77 ttl=64 time=0.111 ms
64 bytes from 10.0.0.1: icmp_seq=78 ttl=64 time=0.102 ms
64 bytes from 10.0.0.1: icmp_seq=79 ttl=64 time=0.063 ms
64 bytes from 10.0.0.1: icmp_seq=80 ttl=64 time=0.097 ms
64 bytes from 10.0.0.1: icmp_seq=81 ttl=64 time=0.074 ms
64 bytes from 10.0.0.1: icmp_seq=82 ttl=64 time=0.069 ms
64 bytes from 10.0.0.1: icmp_seq=83 ttl=64 time=0.225 ms
64 bytes from 10.0.0.1: icmp_seq=84 ttl=64 time=0.071 ms
64 bytes from 10.0.0.1: icmp_seq=85 ttl=64 time=0.052 ms
64 bytes from 10.0.0.1: icmp_seq=86 ttl=64 time=0.097 ms
64 bytes from 10.0.0.1: icmp_seq=87 ttl=64 time=0.076 ms
64 bytes from 10.0.0.1: icmp_seq=88 ttl=64 time=0.208 ms
64 bytes from 10.0.0.1: icmp_seq=89 ttl=64 time=0.261 ms
64 bytes from 10.0.0.1: icmp_seq=90 ttl=64 time=0.108 ms
64 bytes from 10.0.0.1: icmp_seq=91 ttl=64 time=0.074 ms
64 bytes from 10.0.0.1: icmp_seq=92 ttl=64 time=0.073 ms
64 bytes from 10.0.0.1: icmp_seq=93 ttl=64 time=0.146 ms
64 bytes from 10.0.0.1: icmp_seq=94 ttl=64 time=0.071 ms
64 bytes from 10.0.0.1: icmp_seq=95 ttl=64 time=0.106 ms
64 bytes from 10.0.0.1: icmp_seq=96 ttl=64 time=0.118 ms

"Node: h3"
root@Ubuntu:/media/sf_Shared_VM_ubuntu_22_04_files/source#
```

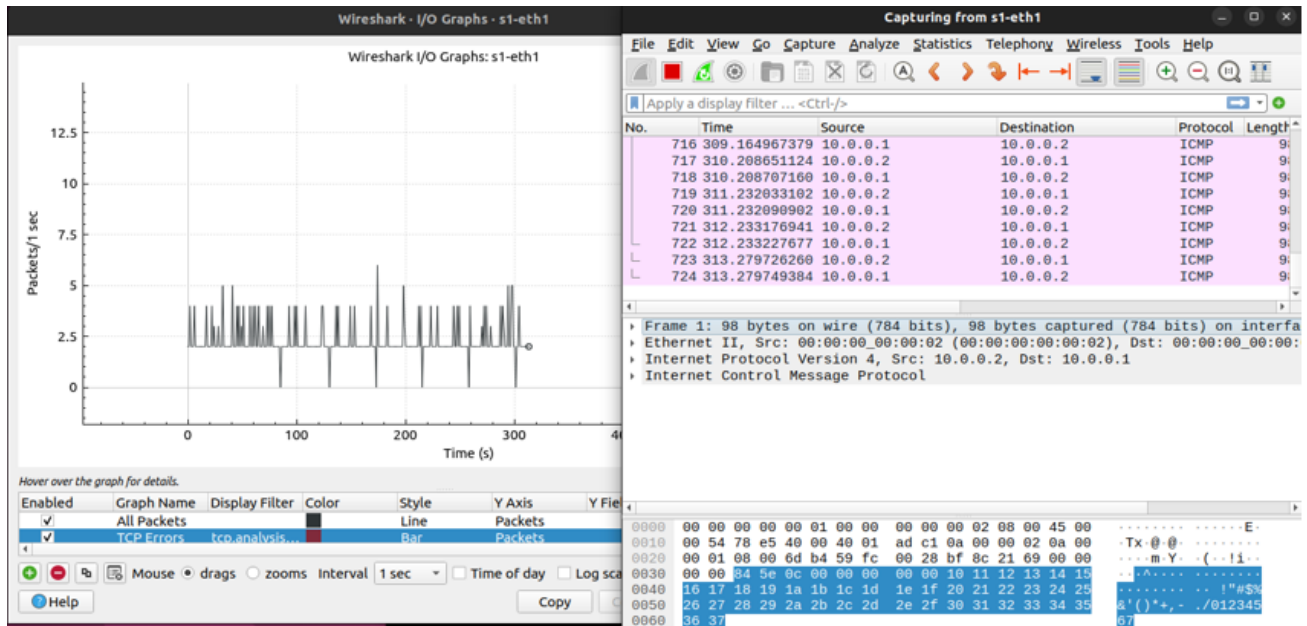
Trong đó:

h1: đảm nhiệm làm webserver ta cho chạy trên port 80

h2: thực hiện ping đều đến h1 để giả lập traffic bình thường

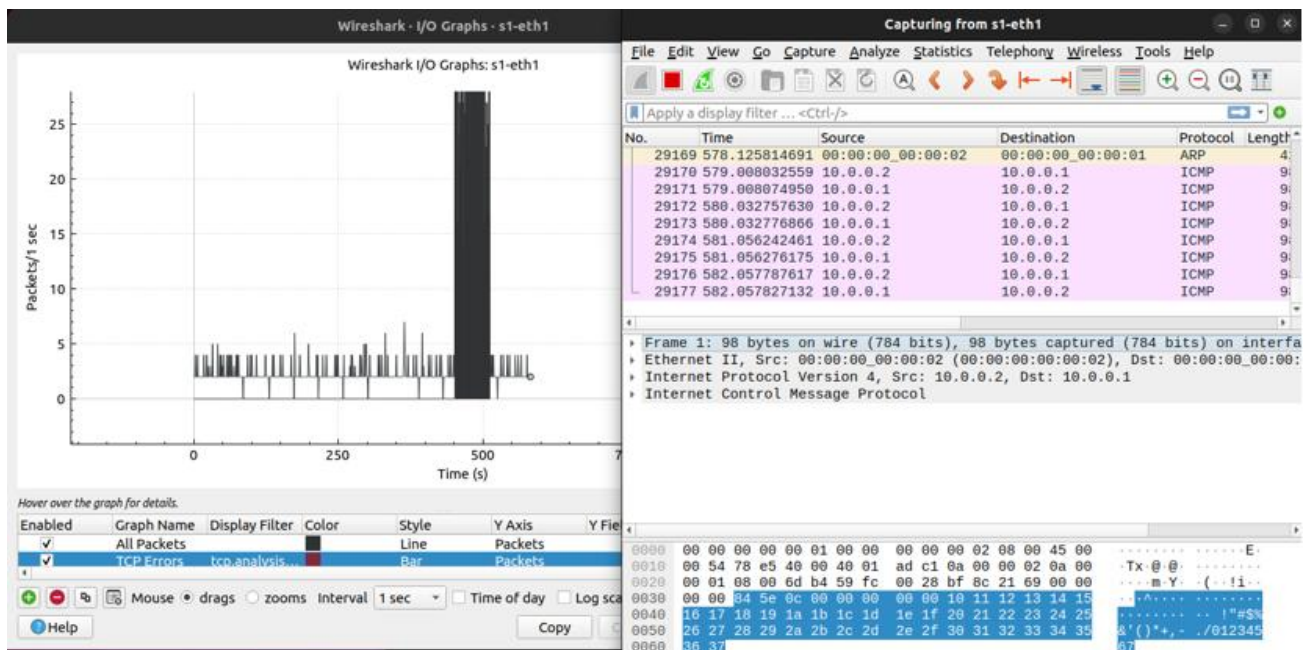
h3: sẵn sàng thực hiện ddos (chạy file ddos.sh)

Ta dùng công cụ như Wireshark để xem lưu lượng của h1 trước khi bị DDOS sẽ như thế nào.



Ta nhận xét rằng, lưu lượng biến thiên nhỏ, minh chứng cho hoạt động bình thường.

Sau đó, ta thử tấn công DDOS trên h3 để xem biểu đồ lưu lượng thay đổi thế nào.



Ta quan sát được rằng có sự biến thiên cao diễn ra. h1 bây giờ đang bị DDOS bằng HTTP flood bởi h3 (h3 được cấu hình để thực hiện ddos trong 60s, mục đích là để giả lập cho ta thấy sự biến thiên về lưu lượng trong 1 khoảng thời gian.)

Giờ ta sẽ bắt đầu áp dụng mô hình ML vào để ngăn chặn DDOS.

Đầu tiên bên terminal B, ta sẽ khởi chạy file thu thập lưu lượng thực tế như sau:

```
root@Ubuntu:/home/lux1dus/DDOS_ML_2025/live_collector# python3 collect_realtime_traffic_ml.py
Collector starting (Floodlight) url=http://127.0.0.1:8080, interval=5s, out=output/PredictFlowStatsfile.csv
2025-11-22T10:25:56.527762 appended 2 rows -> output/PredictFlowStatsfile.csv
2025-11-22T10:26:02.550978 appended 2 rows -> output/PredictFlowStatsfile.csv
2025-11-22T10:26:08.578024 appended 2 rows -> output/PredictFlowStatsfile.csv
```

Ta thấy được các lưu lượng bình thường đang được ghi vào file PredictFlowStatsfile.csv

Sau đó bên terminal A, ta chạy file realtime\_floodlight\_ML.py, file có các tham số có thể truyền vào như:

```
root@Ubuntu:/media/sf_Shared_VM_ubuntu_22.04_files/source# python3 controller/realtime_floodlight_ML.py -h
usage: realtime_floodlight_ML.py [-h] [--model MODEL]
                                [--predict-file PREDICT_FILE]
                                [--floodlight FLOODLIGHT]
                                [--threshold THRESHOLD] [--timeout TIMEOUT]
                                [--interval INTERVAL]
                                [--required-hits REQUIRED_HITS]
                                [--detect-window DETECT_WINDOW]
                                [--cooldown COOLDOWN]
                                [--monitored-net MONITORED_NET]
                                [--port-fallback-count PORT_FALLBACK_COUNT]
                                [--port-to-block PORT_TO_BLOCK]
                                [--http-ports [HTTP_PORTS ...]]
                                [--block-per-victim] [--once]
                                [--min-flows MIN_FLOWS]
                                [--min-total-bytes MIN_TOTAL_BYTES]
                                [--min-avg-pkt-rate MIN_AVG_PKT_RATE]
                                [--grace-seconds GRACE_SECONDS]

ML -> Floodlight static flow pusher
```

Ta tiến hành truyền các tham số cần thiết vào và chạy file:

```

root@Ubuntu:/media/sf_Shared_VM_ubuntu_22.04_files/source# python3 controller/r
ealtime_floodlight_ML.py --model model.pkl --predict-file /home/luxidus/DDOS_ML
_2025/live_collector/output/PredictFlowStatsfile.csv --interval 3 --detect-wind
ow 12
2025-11-22 17:27:22,663 INFO: Loading model from model.pkl
2025-11-22 17:27:25,941 INFO: Using static flow endpoint: /wm/staticflowpusher/
json
2025-11-22 17:27:27,262 INFO: Computed probabilities 4 rows (threshold=0.280)
2025-11-22 17:27:27,285 INFO: HTTP candidate flows in window: 0
2025-11-22 17:27:27,299 INFO: Found 4 switches: ['00:00:00:00:00:00:01', '00
:00:00:00:00:00:02', '00:00:00:00:00:00:03', '00:00:00:00:00:00:04']
2025-11-22 17:27:31,675 INFO: Computed probabilities 4 rows (threshold=0.280)
2025-11-22 17:27:31,680 INFO: HTTP candidate flows in window: 0
2025-11-22 17:27:31,691 INFO: Found 4 switches: ['00:00:00:00:00:00:01', '00
:00:00:00:00:00:02', '00:00:00:00:00:00:03', '00:00:00:00:00:00:04']

```

Sau khi cả terminal A và B đều chạy tốt, ta sẽ bắt đầu thực hiện DDOS lại bằng h3 lên h1 để xem ML hành động thế nào.

Ngay khi vừa chạy lệnh DDOS trên h3, terminal A và B đều phản ứng như sau:

Bên terminal B đã cập nhật các luồng DDOS vào file PredictFlowStatsfile.csv:

```

2025-11-22T10:28:51.564977 appended 1648 rows -> output/PredictFlowStatsfile.cs
v
2025-11-22T10:28:57.850698 appended 1236 rows -> output/PredictFlowStatsfile.cs
v

```

Bên terminal A đã nhận diện được sự thay đổi đột ngột về traffic HTTP:

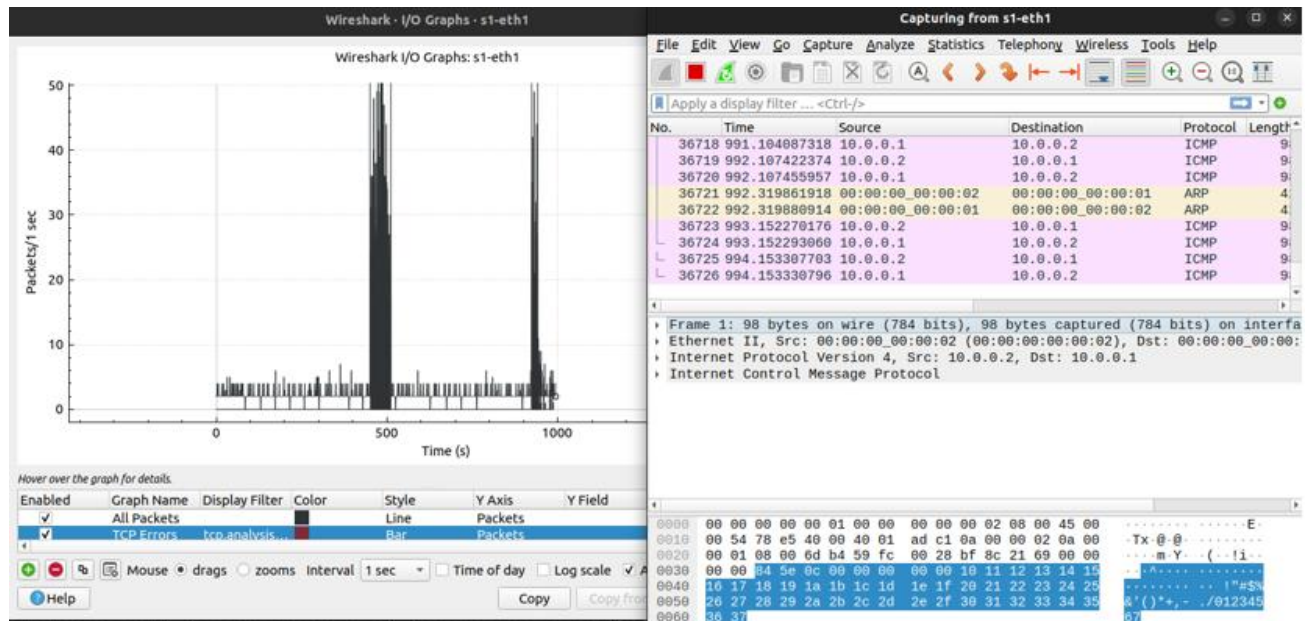
```

2025-11-22 17:28:54,852 INFO: HTTP candidate flows in window: 822
2025-11-22 17:28:54,882 INFO: Found 4 switches: ['00:00:00:00:00:00:01', '00
:00:00:00:00:00:02', '00:00:00:00:00:00:03', '00:00:00:00:00:00:04']
2025-11-22 17:29:00,806 INFO: Computed probabilities 2884 rows (threshold=0.280
)
2025-11-22 17:29:00,815 INFO: HTTP candidate flows in window: 1441
2025-11-22 17:29:00,843 INFO: Found 4 switches: ['00:00:00:00:00:00:01', '00
:00:00:00:00:00:02', '00:00:00:00:00:00:03', '00:00:00:00:00:00:04']
2025-11-22 17:29:01,211 INFO: Push flow block_http_10_0_0_3_any_1763807340 -> s
tatus 200

```

Ở dòng cuối của terminal A, đã cho thấy rằng mô hình ML đã tiến hành đẩy rule chặn host đang thực hiện DDOS (h3) lên webserver (h1)

Ta có thể quan sát trực quan hơn việc ngăn chặn DDOS diễn ra thành công thông qua Wireshark:



Ta có thể thấy rõ khoảng thời gian DDOS lên h1 đã bị giảm đi đáng kể so với việc không ứng dụng ML. Có thể nói rằng, việc ngăn chặn DDOS dạng HTTP flood lên webserver (h1) đã thành công mỹ mãn!

## CHƯƠNG 5. KẾT LUẬN

### 5.1 Kết luận

Tấn công DDoS (Distributed Denial-of-Service) là một trong những mối đe dọa bảo mật nghiêm trọng nhất trong thời đại kỹ thuật số. Khác với các cuộc tấn công có mục đích đánh cắp dữ liệu, DDoS chủ yếu nhằm vào tính khả dụng (availability) của dịch vụ. Một cuộc tấn công DDoS thành công có thể làm cho các dịch vụ quan trọng trở nên hoàn toàn bất khả dụng trong vài giờ hoặc vài ngày, gây ra những hậu quả nghiêm trọng. Tác động trực tiếp bao gồm sự mất mát doanh thu do dịch vụ offline, tổn hại đến danh tiếng của công ty, và mất lòng tin từ khách hàng. Những tác động này không chỉ ảnh hưởng đến tài chính ngắn hạn mà còn có thể gây ra thiệt hại lâu dài đối với uy tín và vị thế cạnh tranh của tổ chức. Do đó, việc phát hiện DDoS sớm và nhanh chóng là một yêu cầu quan trọng không thể bỏ qua trong chiến lược bảo mật của bất kỳ tổ chức nào.

Các phương pháp phòng chống DDoS truyền thống đều có những hạn chế nhất định. Phương pháp dựa trên rule (rule-based) đòi hỏi các chuyên gia định nghĩa các quy tắc phát hiện thủ công, nhưng những quy tắc này có thể bị vượt qua dễ dàng bởi các attacker tinh vi. Phương pháp dựa trên chữ ký (signature-based) chỉ có thể phát hiện các cuộc tấn công đã biết trước đó, hoàn toàn bất lực trước các loại DDoS mới chưa biết (zero-day attacks). Ngược lại, Machine Learning mang đến một cách tiếp cận hoàn toàn khác biệt: thay vì phải định nghĩa quy tắc thủ công, các mô hình ML học trực tiếp từ dữ liệu để xác định các pattern và đặc tính phân biệt giữa lưu lượng bình thường và tấn công. Điều này cho phép hệ thống không chỉ phát hiện được các cuộc tấn công đã biết mà còn có khả năng nhận diện các loại DDoS mới, chưa được quan sát trước đó. Hơn nữa, ML có thể tự động thích ứng với sự thay đổi của lưu lượng mạng theo thời gian mà không cần can thiệp thủ công từ quản trị viên. Khi kết hợp nhiều đặc trưng (features) của luồng mạng, các mô hình ML đạt được độ chính xác cao và có khả năng giảm thiểu false positive (báo động giả), giúp tiết kiệm tài nguyên xử lý và tránh những hành động phòng chống không cần thiết.

## 5.2 Hướng phát triển

### a. Phát hiện khi mô hình "quên" cách phát hiện

- Vấn đề hiện tại: Theo thời gian, kẻ tấn công thay đổi cách tấn công → mô hình không còn chính xác.
- Cách giải quyết:
  - Thêm một hệ thống để phát hiện khi lưu lượng mạng bắt đầu thay đổi
  - Khi phát hiện thấy sự thay đổi → tự động huấn luyện lại mô hình bằng dữ liệu mới
  - Điều này giúp mô hình luôn cập nhật với các chiến thuật mới của kẻ tấn công
- Mục tiêu: Mô hình luôn giữ độ chính xác trên 90% dù kẻ tấn công thay đổi chiến lược

### b. Phát Hiện Những Hành Vi Bất Thường

- Vấn đề hiện tại: Hệ thống chỉ phát hiện DDoS, nhưng không phát hiện các hành vi bất thường khác (ví dụ: ai đó cố đánh cắp dữ liệu).
- Cách giải quyết:
  - Xây dựng một hệ thống phát hiện những hành vi lạ:
    - Dùng dữ liệu lưu lượng bình thường để "học" thế nào là bình thường
    - Khi có hành vi lạ → báo cáo ngay
  - Lợi ích:
    - Phát hiện được các loại tấn công hoàn toàn mới
    - Phát hiện người nội bộ có hành vi xấu
    - Phát hiện những lỗi cấu hình trong mạng
- Mục tiêu: Phát hiện được 70-80% hành vi bất thường, báo động giả dưới 5%

## TÀI LIỆU THAM KHẢO

1. scikit-learn: Machine learning library (Random Forest, preprocessing)
2. Floodlight Controller: OpenFlow controller
3. Mininet: Network emulator
4. Pandas: Data manipulation
5. NumPy: Numerical computing
6. Matplotlib: Data visualization