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 17 tháng 12 năm 2025

Tác giả

(Ký tên và ghi rõ họ tên)

Quang

Hồ Nhựt 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 17 tháng 12 năm 2025

Tác giả

(Ký tên và ghi rõ họ tên)

Quang

Hồ Nhựt 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 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 đồ 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

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

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 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

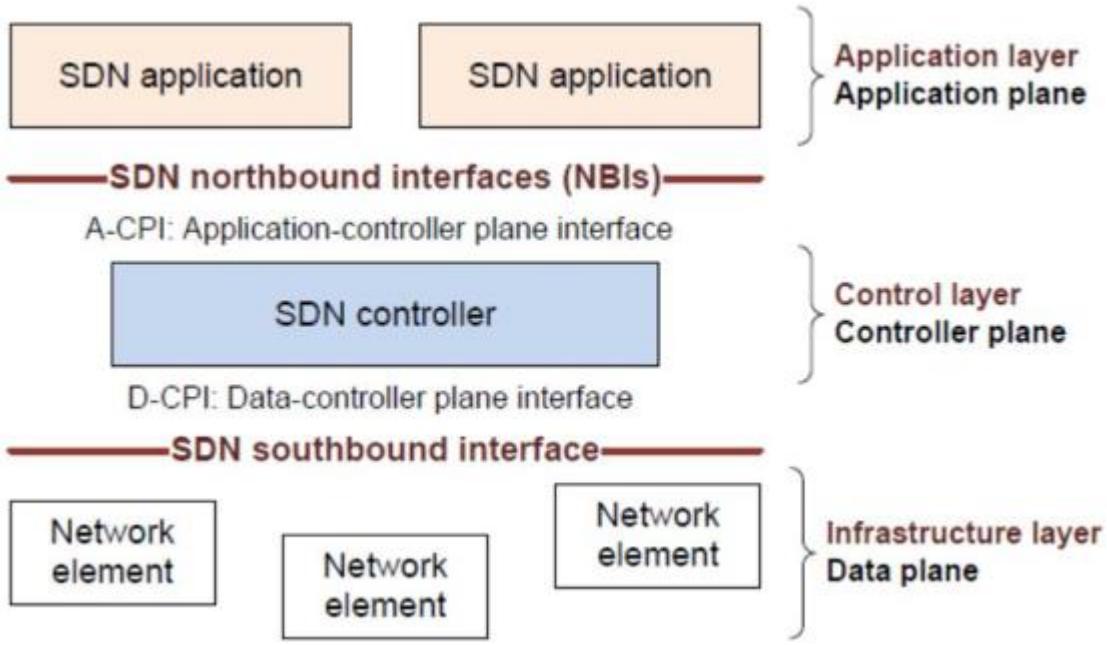
và phát hiện các dấu hiệu của tấn công. Các phương pháp dựa trên học máy sẽ phân tích lưu lượng mạng và tự động nhận diện các cuộc tấn công DDoS, từ đó giúp hệ thống phản ứng kịp thời để ngăn chặn hoặc giảm thiểu tác động của tấn công.

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

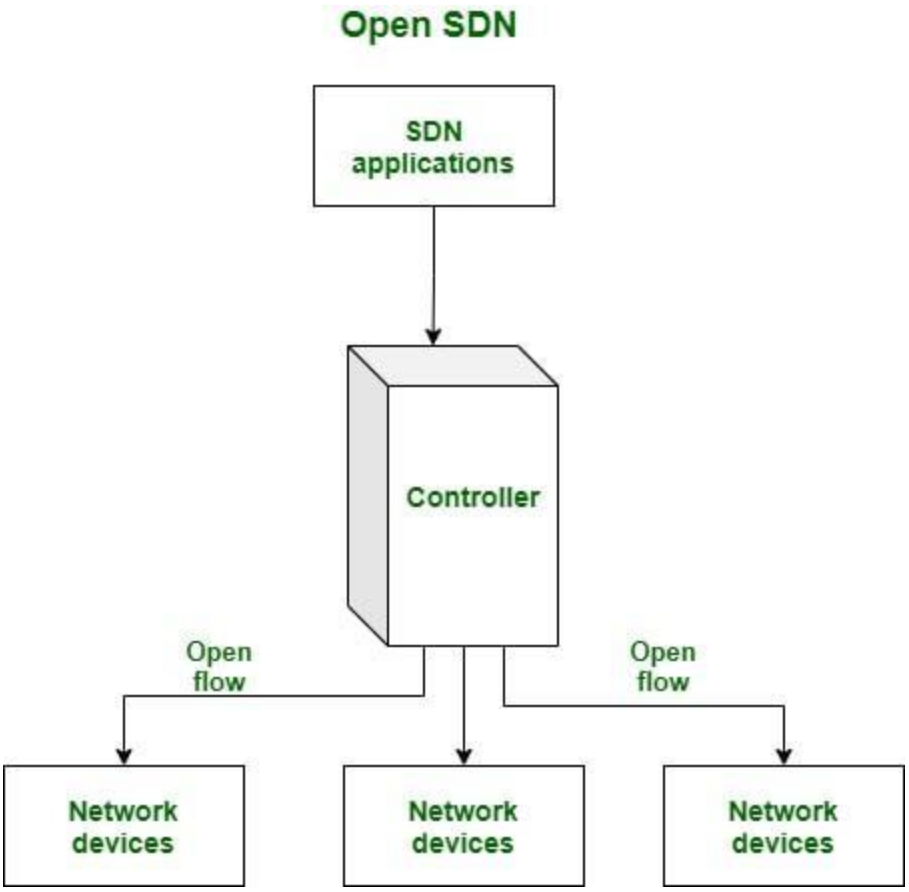
(control plane) riêng biệt. Lớp điều khiển của các switch trao đổi thông tin về cấu trúc mạng và từ đó xây dựng bảng chuyển tiếp (forwarding table) để quyết định cách thức chuyển tiếp gói dữ liệu. Mạng SDN là phương pháp tách rời lớp điều khiển khỏi các switch và chuyển giao chức năng này cho một bộ 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.

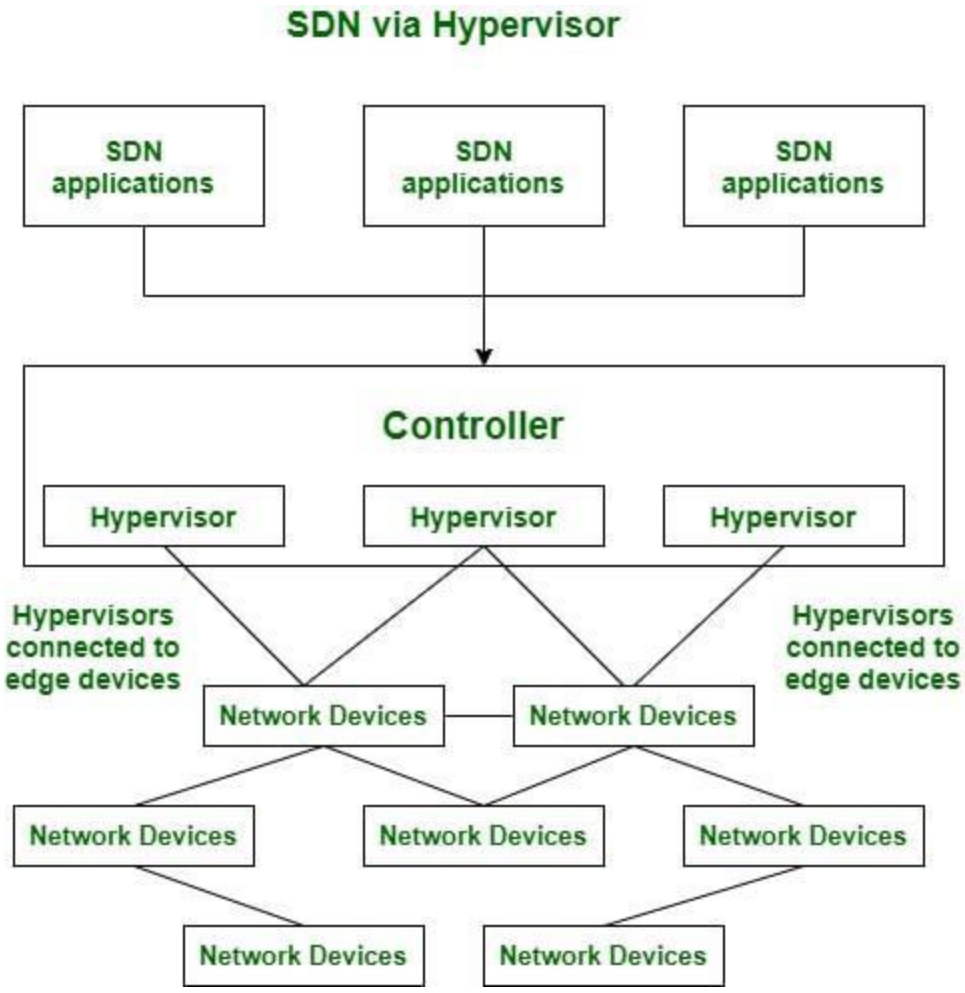
Các lớp giao tiếp thông qua một tập hợp các giao diện được gọi là northbound APIs và southbound APIs. Một số mô hình khác nhau của SDN:

- 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



Hình 3: SDN via Hypervisor-based Overlay Network

- Hybrid SDN là sự kết hợp của Mạng truyền thống với Mạng được định nghĩa bằng phần mềm trong một mạng duy nhất để hỗ trợ các loại chức năng khác nhau trên một mạng.

2.1.1 So sánh một số điểm khác nhau giữa Mạng định nghĩa bằng phần mềm và mạng truyền thống.

Tiêu chí	Mạng truyền thống	SDN (Software-Defined Networking)	Khác biệt chính

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

Chi phí	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 mạnh: 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 Lý do lựa chọn SDN trong phòng chống tấn công DDoS

Trong bối cảnh các cuộc tấn công từ chối dịch vụ phân tán (DDoS) ngày càng gia tăng về quy mô, cường độ và mức độ tinh vi, các kiến trúc mạng truyền thống bộc lộ nhiều hạn chế trong việc phát hiện và ứng phó kịp thời. Do đó, việc lựa chọn mạng định nghĩa bằng phần mềm (Software-Defined Networking – SDN) làm nền tảng triển khai giải pháp chống DDoS là cần thiết và có cơ sở khoa học.

Thứ nhất, SDN cung cấp khả năng điều khiển tập trung, cho phép quản trị viên có cái nhìn toàn cục về trạng thái mạng. Trong khi mạng truyền thống phân tán chức năng điều khiển trên từng thiết bị (router, switch), SDN tách biệt rõ ràng mặt phẳng điều khiển (control plane) và mặt phẳng chuyển tiếp dữ liệu (data plane). Nhờ đó, bộ điều khiển SDN (SDN Controller) có thể nhanh chóng phát hiện các luồng lưu lượng bất thường – đặc trưng của tấn công DDoS – dựa trên thông tin thu thập từ toàn mạng, thay vì xử lý cục bộ và rời rạc như các mạng truyền thống.

Thứ hai, SDN có khả năng phản ứng linh hoạt và theo thời gian thực. Khi phát hiện tấn công, SDN cho phép cập nhật hoặc sinh động các luật chuyển tiếp (flow rules) trên switch một cách nhanh chóng để chặn, giới hạn băng thông hoặc chuyển hướng lưu lượng tấn công. Ngược lại, trong mạng truyền thống, việc thay đổi chính sách phòng thủ thường đòi hỏi cấu hình thủ công trên từng thiết bị, mất nhiều thời gian và khó đáp ứng yêu cầu phản ứng tức thời trước các cuộc tấn công DDoS quy mô lớn.

Thứ ba, SDN dễ dàng tích hợp với các kỹ thuật phát hiện thông minh, đặc biệt là các phương pháp học máy (Machine Learning). Nhờ giao diện lập trình mở (Open API), SDN cho

phép kết nối trực tiếp với các hệ thống IDS/IPS, các mô hình phát hiện bất thường hoặc các thuật toán phân tích lưu lượng nâng cao. Điều này giúp nâng cao độ chính xác trong việc phát hiện DDoS so với các cơ chế dựa trên chữ ký (signature-based) truyền thống vốn khó thích nghi với các kiểu tấn công mới.

Thứ tư, SDN nâng cao khả năng mở rộng và tự động hóa trong phòng chống DDoS. Khi lưu lượng mạng tăng đột biến, SDN có thể tự động phân phối lại tài nguyên, điều chỉnh chính sách định tuyến hoặc kích hoạt các cơ chế giảm tải mà không cần can thiệp thủ công. Đây là ưu điểm nổi bật so với mạng truyền thống, vốn có khả năng mở rộng hạn chế và phụ thuộc nhiều vào cấu hình tĩnh.

Cuối cùng, SDN giúp giảm chi phí vận hành và nâng cao hiệu quả quản lý mạng. Thay vì đầu tư nhiều thiết bị bảo mật chuyên dụng cho từng phân đoạn mạng, SDN cho phép triển khai các giải pháp phòng chống DDoS ở mức phần mềm, tập trung và linh hoạt hơn, phù hợp với các hệ thống mạng hiện đại như trung tâm dữ liệu, cloud và hạ tầng doanh nghiệp lớn.

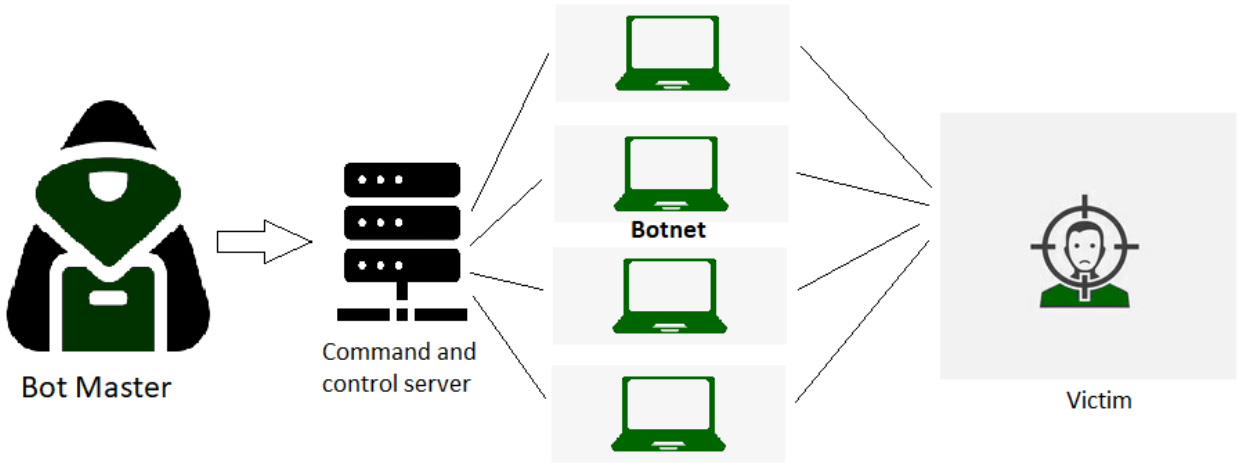
Từ các phân tích trên, có thể thấy rằng SDN là một nền tảng phù hợp và hiệu quả hơn so với mạng truyền thống trong việc phát hiện, ngăn chặn và giảm thiểu tác động của các cuộc tấn công DDoS, đặc biệt trong các môi trường mạng quy mô lớn và yêu cầu tính linh hoạt cao.

2.3 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 một	Khó phòng thủ do lưu lượng phân tán từ

khăn khi phòng thủ	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

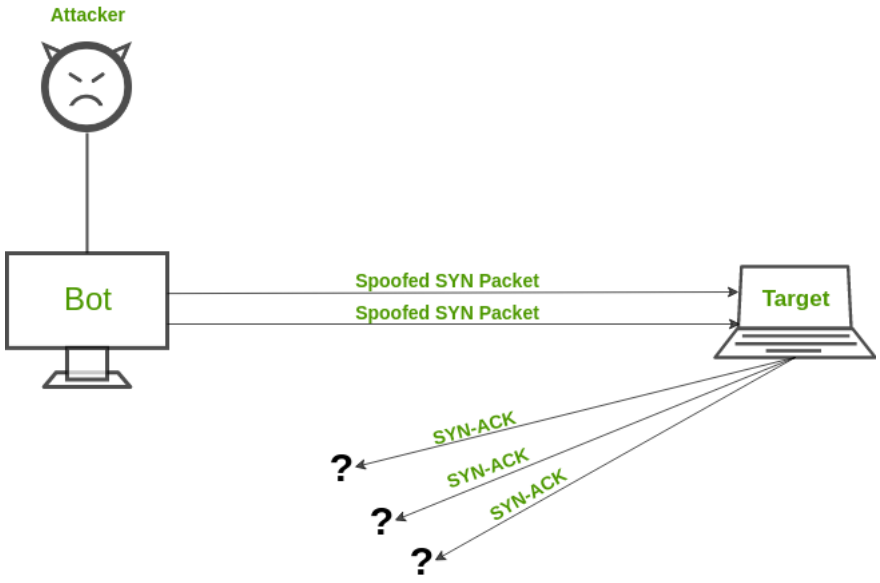
Một cuộc tấn công từ chối dịch vụ phân tán (DDoS) xảy ra khi một dịch vụ trực tuyến hợp lệ bị gửi quá nhiều yêu cầu vượt quá khả năng xử lý thông thường của nó. Chẳng hạn, một trang web chỉ có thể tiếp nhận và xử lý một lượng yêu cầu nhất định mỗi phút; khi lưu lượng truy cập vượt ngưỡng này, trang có 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 lượng truy cập ồ ạt từ nhiều nguồn, khiến mục tiêu nhanh chóng bị quá tải và không thể duy trì dịch vụ.

2.3.1 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:

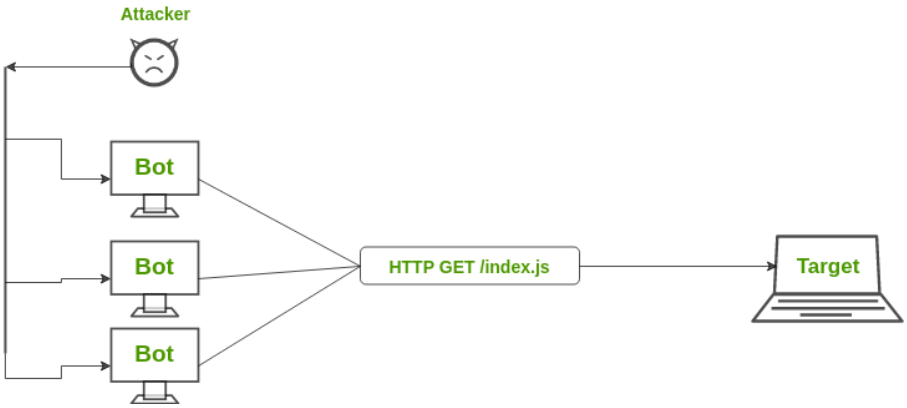
+ Gửi gói SYN: Kẻ tấn công gửi nhiều gói SYN đến máy chủ, thường là từ các địa chỉ IP giả mạo để che giấu danh tính thực.

+ 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 Tuyến Hố Đen: Trong định tuyến hố đen, lưu lượng mạng được điều hướng đến một "hố 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.

2.4 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.

2.4.1 Lý do lựa chọn thuật toán Decision Tree:

Trong bài toán phát hiện và phân loại tấn công DDoS, việc lựa chọn thuật toán học máy phù hợp đóng vai trò quan trọng trong việc đảm bảo độ chính xác, tốc độ xử lý và khả năng triển khai thực tế. Thuật toán Decision Tree (cây quyết định) được lựa chọn trong nghiên cứu này thay cho các thuật toán khác như SVM, k-NN, Neural Network hay Naive Bayes vì những ưu điểm nổi bật sau.

Thứ nhất, Decision Tree có khả năng diễn giải cao (high interpretability). Mô hình cây quyết định biểu diễn quá trình ra quyết định dưới dạng các nút và nhánh logic rõ ràng, cho phép người quản trị mạng dễ dàng hiểu được nguyên nhân tại sao một luồng lưu lượng bị phân loại là bình thường hay tấn công DDoS. Điều này đặc biệt quan trọng trong các hệ thống an ninh mạng, nơi yêu cầu tính minh bạch và khả năng giải thích cao hơn so với các mô hình "hộp đen" như mạng nơ-ron sâu (Deep Neural Networks).

Thứ hai, Decision Tree có tốc độ huấn luyện và suy luận nhanh, phù hợp với môi trường mạng thời gian thực. So với các thuật toán phức tạp như SVM hay Deep Learning, Decision Tree không đòi hỏi quá trình tối ưu hóa phức tạp hoặc tài nguyên tính toán lớn. Điều này giúp

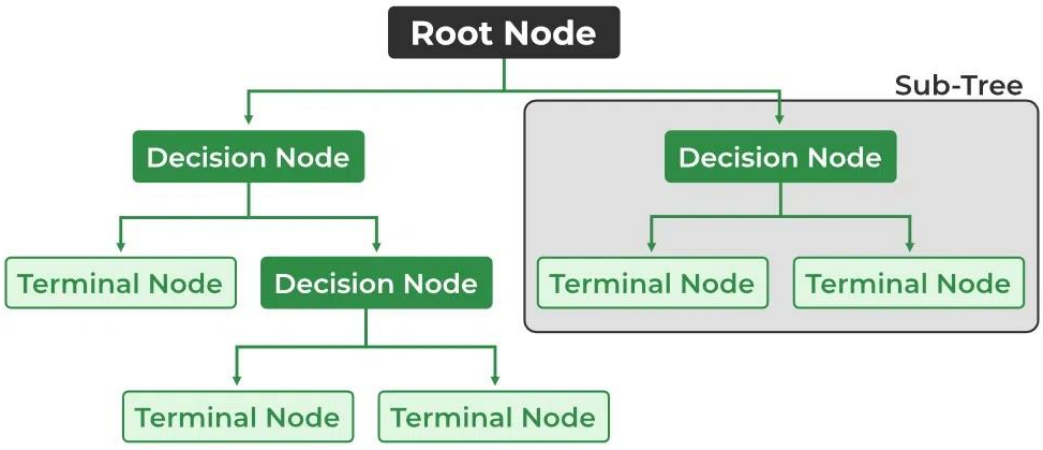
thuật toán có thể được tích hợp trực tiếp vào bộ điều khiển SDN để đưa ra quyết định chặn hoặc điều hướng lưu lượng DDoS gần như tức thời.

Thứ ba, Decision Tree xử lý tốt dữ liệu không tuyến tính và không yêu cầu chuẩn hóa dữ liệu. Trong thực tế, các đặc trưng lưu lượng mạng (số gói, tốc độ gói, entropy, tỷ lệ SYN/ACK, v.v.) thường có mối quan hệ phi tuyến và phân bố không đồng đều. Decision Tree có thể xử lý hiệu quả các dạng dữ liệu này mà không cần bước tiền xử lý phức tạp như chuẩn hóa hay chuẩn hóa đặc trưng, điều mà nhiều thuật toán khác yêu cầu.

Thứ tư, Decision Tree hoạt động tốt với tập dữ liệu có kích thước vừa và nhiều đặc trưng rời rạc, phù hợp với các bộ dữ liệu mạng phổ biến trong nghiên cứu DDoS. So với k-NN (nhạy cảm với nhiễu và kích thước dữ liệu lớn) hoặc Naive Bayes (giả định độc lập giữa các đặc trưng), Decision Tree linh hoạt hơn và phản ánh tốt hơn bản chất thực tế của lưu lượng mạng.

Cuối cùng, Decision Tree dễ mở rộng và tích hợp. Thuật toán này là nền tảng cho các mô hình mạnh hơn như Random Forest và Gradient Boosting, cho phép nghiên cứu mở rộng trong tương lai mà vẫn giữ được tính nhất quán về phương pháp. Đồng thời, việc triển khai Decision Tree trong các hệ thống SDN – IDS/IPS có độ phức tạp thấp và dễ bảo trì hơn so với nhiều thuật toán khác.

*2.4.2 Cấu Trúc Của Cây Quyết Định (*Decision Tree)*:*



Hình 7: Sơ đồ cấu trúc cây quyết định

Hình 7 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.

2.4.3 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.

2.4.4 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 Đệ 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.

2.4.5 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.

2.4.6 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.

2.4.7 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.

2.4.8 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% lấp 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.

2.4.9 Lý do lựa chọn Floodlight SDN Controller:

Trong nghiên cứu và triển khai hệ thống phát hiện – phòng chống tấn công DDoS dựa trên SDN, việc lựa chọn bộ điều khiển mạng (SDN Controller) đóng vai trò then chốt, ảnh hưởng trực tiếp đến khả năng giám sát, điều khiển lưu lượng và tích hợp các thuật toán phát hiện. Trong số các SDN Controller phổ biến hiện nay như OpenDaylight, ONOS, Ryu hay POX, Floodlight được lựa chọn nhờ các ưu điểm phù hợp với mục tiêu và phạm vi nghiên cứu.

Thứ nhất, Floodlight có kiến trúc đơn giản, dễ triển khai và cấu hình. So với OpenDaylight hay ONOS – vốn có kiến trúc phức tạp, nhiều module và yêu cầu cấu hình cao – Floodlight gọn nhẹ hơn, phù hợp cho môi trường nghiên cứu, thử nghiệm và mô phỏng. Điều này giúp

giảm thời gian làm quen, triển khai nhanh hệ thống SDN và tập trung vào bài toán chính là phát hiện và giảm thiểu tấn công DDoS.

Thứ hai, Floodlight cung cấp khả năng lập trình linh hoạt thông qua API REST và OpenFlow, cho phép dễ dàng thu thập thống kê lưu lượng (flow statistics, port statistics) từ các switch. Đây là cơ sở quan trọng để trích xuất đặc trưng lưu lượng mạng phục vụ cho các thuật toán học máy như Decision Tree trong việc phát hiện hành vi DDoS. So với POX (đã cũ và ít được cập nhật), Floodlight ổn định hơn và hỗ trợ tốt hơn cho các phiên bản OpenFlow phổ biến.

Thứ ba, Floodlight hoạt động hiệu quả trong môi trường mô phỏng Mininet, là nền tảng thường được sử dụng trong các nghiên cứu học thuật về SDN và an ninh mạng. Floodlight có khả năng tương thích cao với các switch ảo của Mininet, giúp dễ dàng xây dựng các kịch bản tấn công DDoS (SYN Flood, UDP Flood, ICMP Flood,…) và đánh giá hiệu quả của các cơ chế phòng chống được đề xuất.

Thứ tư, Floodlight phù hợp với các hệ thống SDN quy mô nhỏ và trung bình, vốn là phạm vi triển khai phổ biến trong các nghiên cứu và thử nghiệm học thuật. Trong khi ONOS và OpenDaylight được thiết kế hướng đến mạng carrier-grade hoặc mạng doanh nghiệp quy mô lớn, Floodlight đáp ứng tốt yêu cầu kiểm soát tập trung, phản ứng nhanh và độ trễ thấp trong môi trường thí nghiệm.

Cuối cùng, Floodlight có cộng đồng tài liệu và ví dụ triển khai rõ ràng, hỗ trợ tốt cho việc học tập, nghiên cứu và mở rộng. Nhiều công trình nghiên cứu về phát hiện DDoS trên SDN đã sử dụng Floodlight như một nền tảng tham chiếu, giúp dễ dàng so sánh, đối chiếu và đánh giá kết quả nghiên cứu.

2.4.10 Lý do lựa chọn thuật toán Random Forest:

Trong bài toán phát hiện và phân loại tấn công DDoS trong môi trường mạng SDN, dữ liệu lưu lượng mạng thường có đặc điểm nhiễu cao, phân bố không đồng đều và quan hệ phi tuyến phức tạp. Do đó, việc lựa chọn mô hình học máy cần đảm bảo độ chính xác, tính ổn định và khả năng triển khai thực tế. Random Forest được lựa chọn thay cho các mô hình cây khác như Decision Tree đơn lẻ, Extra Trees hay Gradient Boosting nhờ các ưu điểm sau.

Thứ nhất, Random Forest khắc phục hiện tượng overfitting của Decision Tree đơn lẻ.

Trong khi một cây quyết định riêng lẻ dễ học quá chi tiết theo dữ liệu huấn luyện, Random Forest sử dụng cơ chế ensemble kết hợp nhiều cây quyết định được huấn luyện trên các tập dữ liệu con ngẫu nhiên (bagging) và các tập đặc trưng con. Nhờ đó, mô hình có khả năng tổng quát hóa tốt hơn, đặc biệt quan trọng trong việc phát hiện các dạng tấn công DDoS mới hoặc biến thể chưa từng xuất hiện trong dữ liệu huấn luyện.

Thứ hai, Random Forest có độ chính xác và độ ổn định cao hơn so với các mô hình cây khác. So với Extra Trees (Extremely Randomized Trees) – nơi các ngưỡng chia được chọn hoàn toàn ngẫu nhiên – Random Forest vẫn giữ được sự cân bằng giữa tính ngẫu nhiên và khả năng học cấu trúc dữ liệu, giúp mô hình ít bị dao động khi dữ liệu đầu vào thay đổi hoặc có nhiễu.

Thứ ba, Random Forest xử lý hiệu quả dữ liệu nhiều chiều và phi tuyến. Các đặc trưng lưu lượng mạng trong bài toán DDoS (tốc độ gói, số lượng luồng, entropy địa chỉ IP, tỷ lệ SYN/ACK,…) thường có mối quan hệ phức tạp. Random Forest có khả năng mô hình hóa các mối quan hệ này mà không cần giả định tuyến tính, vượt trội hơn so với nhiều mô hình cây đơn giản hoặc các phương pháp dựa trên ngưỡng cố định.

Thứ tư, Random Forest có khả năng đánh giá mức độ quan trọng của đặc trưng (feature importance). Điều này cho phép xác định các đặc trưng nào đóng vai trò quan trọng nhất trong việc phát hiện DDoS, hỗ trợ quá trình phân tích, tối ưu mô hình và giải thích kết quả. Đây là ưu điểm lớn so với một số mô hình cây khác hoặc các mô hình "hộp đen" như mạng nơ-ron sâu.

Cuối cùng, Random Forest dễ triển khai và tích hợp trong hệ thống SDN thời gian thực. So với các mô hình tăng cường như Gradient Boosting hoặc XGBoost – vốn yêu cầu tinh chỉnh siêu tham số phức tạp và thời gian huấn luyện dài – Random Forest có cấu trúc đơn giản hơn, tốc độ suy luận nhanh và phù hợp để triển khai trong bộ điều khiển SDN nhằm đưa ra quyết định chặn hoặc điều hướng lưu lượng DDoS kịp thời.

2.4.11 Lý do sử dụng CalibratedClassifierCV:

Trong hệ thống phát hiện tấn công DDoS dựa trên Machine Learning, xác suất dự đoán (prediction probability) đóng vai trò quan trọng trong việc ra quyết định chặn hoặc cho phép lưu lượng, đặc biệt khi mô hình được tích hợp vào bộ điều khiển SDN để phản ứng theo thời

gian thực. Tuy nhiên, nhiều thuật toán phân loại phổ biến như Decision Tree, Random Forest hay SVM thường cho xác suất dự đoán không được hiệu chỉnh tốt (poorly calibrated), tức là giá trị xác suất không phản ánh chính xác mức độ tin cậy thực sự của dự đoán. Do đó, việc sử dụng CalibratedClassifierCV là cần thiết.

Thứ nhất, CalibratedClassifierCV giúp hiệu chỉnh xác suất đầu ra của mô hình phân loại. Thay vì chỉ quan tâm đến nhãn dự đoán (normal/DDoS), hệ thống cần một giá trị xác suất đáng tin cậy để so sánh với ngưỡng (threshold). CalibratedClassifierCV sử dụng các phương pháp hiệu chỉnh như Platt Scaling hoặc Isotonic Regression nhằm biến đổi xác suất thô của mô hình thành xác suất có ý nghĩa thống kê hơn.

Thứ hai, CalibratedClassifierCV đặc biệt phù hợp với các mô hình ensemble như Random Forest. Mặc dù Random Forest thường cho độ chính xác cao, nhưng xác suất dự đoán của nó có xu hướng bị lệch (overconfident hoặc underconfident). So với việc sử dụng trực tiếp predict_proba() từ mô hình gốc, CalibratedClassifierCV giúp cải thiện đáng kể độ tin cậy của xác suất, từ đó nâng cao hiệu quả quyết định chặn DDoS trong môi trường SDN.

Thứ ba, CalibratedClassifierCV hỗ trợ đánh giá xác suất một cách khách quan thông qua cross-validation. Khác với các phương pháp hiệu chỉnh thủ công hoặc chia dữ liệu cố định, CalibratedClassifierCV tích hợp sẵn cơ chế cross-validation, giúp giảm nguy cơ overfitting trong quá trình hiệu chỉnh và đảm bảo mô hình có khả năng tổng quát hóa tốt hơn trên dữ liệu chưa thấy.

Thứ tư, CalibratedClassifierCV phù hợp với các hệ thống ra quyết định dựa trên ngưỡng (threshold-based decision). Trong bài toán này, Floodlight controller chỉ chặn luồng lưu lượng khi xác suất tấn công vượt quá một giá trị nhất định (ví dụ 0.07). Nếu xác suất không được hiệu chỉnh, việc lựa chọn threshold sẽ mang tính chủ quan và kém ổn định. Việc sử dụng CalibratedClassifierCV giúp threshold có ý nghĩa rõ ràng hơn và dễ điều chỉnh trong thực nghiệm.

Cuối cùng, CalibratedClassifierCV đơn giản, ổn định và dễ tích hợp. So với các phương pháp hiệu chỉnh phức tạp hoặc các mô hình xác suất sâu (Bayesian Neural Networks), CalibratedClassifierCV có chi phí tính toán thấp, dễ triển khai trong pipeline Machine Learning hiện có và phù hợp với hệ thống phát hiện DDoS thời gian thực.

2.4.12 Lý do lựa chọn công cụ CICFlowMeter:

Trong hệ thống phát hiện tấn công DDoS dựa trên Machine Learning, việc trích xuất đặc trưng lưu lượng mạng một cách chính xác và nhất quán là bước then chốt quyết định hiệu quả của mô hình học máy. Trong số các công cụ phân tích lưu lượng mạng hiện nay như Tshark, Argus hay NetFlow, CICFlowMeter được lựa chọn nhờ các ưu điểm phù hợp với mục tiêu nghiên cứu.

Thứ nhất, CICFlowMeter được phát triển bởi Canadian Institute for Cybersecurity (CIC), là đơn vị cung cấp các bộ dữ liệu chuẩn trong lĩnh vực an ninh mạng như CICIDS2017, CICDDoS2019. Do đó, công cụ này đảm bảo tính tương thích cao với các bộ dữ liệu chuẩn, giúp kết quả thực nghiệm có độ tin cậy và dễ so sánh với các nghiên cứu liên quan.

Thứ hai, CICFlowMeter trích xuất đặc trưng ở mức flow thay vì packet, phù hợp với các hệ thống phát hiện DDoS trong môi trường SDN. Các đặc trưng flow-level như số gói, tốc độ gói, thời gian tồn tại flow, độ lệch chuẩn inter-arrival time, tỷ lệ byte/packet… phản ánh rõ hành vi bất thường của tấn công DDoS, đồng thời giảm đáng kể khối lượng dữ liệu so với phân tích packet-level.

Thứ ba, CICFlowMeter cung cấp tập đặc trưng phong phú và có ý nghĩa cao, bao gồm các đặc trưng thống kê theo hai chiều (forward/backward), các chỉ số thời gian, kích thước gói, và hành vi TCP flag. Những đặc trưng này đã được kiểm chứng hiệu quả trong nhiều nghiên cứu về phát hiện DDoS và tấn công mạng nói chung, giúp cải thiện độ chính xác của các mô hình học máy như Decision Tree hay Random Forest.

Thứ tư, CICFlowMeter dễ tích hợp vào pipeline xử lý dữ liệu realtime. Công cụ hỗ trợ đầu vào từ file pcap và xuất ra file CSV chuẩn hóa, thuận tiện cho việc tiền xử lý, huấn luyện và dự đoán realtime trong hệ thống SDN. So với các công cụ khác yêu cầu cấu hình phức tạp hoặc phụ thuộc thiết bị mạng chuyên dụng, CICFlowMeter linh hoạt và phù hợp với môi trường mô phỏng Mininet.

Cuối cùng, CICFlowMeter là công cụ mã nguồn mở, ổn định và được sử dụng rộng rãi trong nghiên cứu học thuật. Việc sử dụng một công cụ phổ biến giúp nâng cao khả năng tái lập (reproducibility) của nghiên cứu, đồng thời hỗ trợ việc mở rộng hoặc so sánh kết quả với các công trình trước đó.

2.4.13 Lý do lựa chọn công cụ TCPDUMP:

Trong hệ thống phát hiện và giảm thiểu tấn công DDoS dựa trên SDN và Machine Learning, việc thu thập lưu lượng mạng theo thời gian thực là bước quan trọng nhằm đảm bảo dữ liệu đầu vào phản ánh chính xác trạng thái hoạt động của mạng. Trong số các công cụ bắt gói phổ biến như Wireshark, Tshark hay các hệ thống giám sát chuyên dụng, TCPDUMP được lựa chọn nhờ các ưu điểm phù hợp với yêu cầu của hệ thống.

Thứ nhất, TCPDUMP là công cụ dòng lệnh nhẹ, hiệu năng cao và hoạt động ổn định. Công cụ này cho phép bắt gói trực tiếp ở mức kernel thông qua thư viện libpcap, với độ trễ thấp và tiêu thụ tài nguyên hệ thống nhỏ. Điều này đặc biệt quan trọng trong kịch bản thu thập lưu lượng mạng realtime khi xảy ra tấn công DDoS, nơi lưu lượng tăng đột biến và yêu cầu hệ thống giám sát không trở thành nút nghẽn.

Thứ hai, TCPDUMP hỗ trợ lọc gói linh hoạt thông qua biểu thức BPF (Berkeley Packet Filter). Nhờ khả năng lọc ngay từ tầng thấp, TCPDUMP có thể chỉ thu thập các gói tin cần thiết (ví dụ TCP, UDP, ICMP hoặc theo cổng dịch vụ), giúp giảm đáng kể khối lượng dữ liệu thô và tối ưu quá trình xử lý tiếp theo. So với Wireshark – vốn thiên về giao diện đồ họa và phân tích thủ công – TCPDUMP phù hợp hơn cho các hệ thống tự động.

Thứ ba, TCPDUMP dễ dàng tích hợp vào các script và pipeline xử lý dữ liệu. Công cụ này cho phép ghi trực tiếp dữ liệu bắt được vào file pcap, tạo điều kiện thuận lợi cho các bước xử lý tiếp theo như trích xuất đặc trưng bằng CICFlowMeter hoặc phân tích offline. Việc tích hợp TCPDUMP vào các script shell giúp hệ thống có thể tự động hóa hoàn toàn quá trình thu thập dữ liệu.

Thứ tư, TCPDUMP phù hợp với môi trường mô phỏng và thử nghiệm SDN. Trong các kịch bản sử dụng Mininet, TCPDUMP có thể dễ dàng bắt gói trên các interface ảo mà không cần cấu hình phức tạp, cho phép quan sát chính xác lưu lượng giữa các host và switch trong quá trình tấn công DDoS và phản ứng của Floodlight controller.

Cuối cùng, TCPDUMP là công cụ mã nguồn mở, phổ biến và đáng tin cậy. Việc sử dụng TCPDUMP giúp đảm bảo tính tái lập của thí nghiệm, đồng thời thuận tiện cho việc so sánh và mở rộng nghiên cứu với các công trình liên quan trong lĩnh vực an ninh mạng.

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 trên kiến trúc SDN (Software Defined Networking) với Floodlight controller. Chuỗi xử lý dữ liệu theo thời gian thực như sau:

- Bộ Thu Thập Dữ Liệu (Capture & Convert): Quay PCAP theo cửa sổ thời gian bằng tcpdump và chuyển đổi PCAP sang CSV đặc trưng bằng CICFlowMeter.
- Bộ Tiền Xử Lý và Trích Xuất Đặc Trưng (Feature Engineering): Chuẩn hoá schema theo feature_schema.py, xử lý thiếu/ngoại lệ, chuẩn hoá thang đo.
- Mô Hình Học Máy (ML Model): Sử dụng Random Forest (bọc CalibratedClassifierCV) để dự đoán xác suất DDoS theo luồng, sau đó điều khiển Floodlight đẩy flow rule chặn IP nguồn tấn công.

3.2 Kiến Trúc Chi Tiết

3.2.1 Bộ Thu Thập Dữ Liệu (Collector):

Chức năng: Ghi nhận gói tin từ interface Mininet, xoay file PCAP theo cửa sổ thời gian và chuyển đổi sang bảng đặc trưng cho mô hình ML.

Qui trình hoạt động:

1. Quay PCAP: Script capture_tcpdump.sh sử dụng tcpdump để quay gói trên interface (mặc định: s1-eth1) và xoay file theo cửa sổ thời gian (mặc định: 15 giây) vào thư mục output/pcap_in.
2. Chuyển PCAP → CSV: Script pcap_processor.sh theo dõi thư mục output/pcap_in, với mỗi file PCAP mới:
   - Gọi CICFlowMeter CLI để sinh ra các CSV đặc trưng tạm thời.
   - Gộp các CSV tạm vào một file hợp nhất output/final_csv/Predict.csv (giữ một header duy nhất), đồng thời đánh dấu .done cho PCAP đã xử lý.

3. Tần suất xử lý: capture_tcpdump.sh xoay PCAP theo cửa sổ (mặc định 15s). pcap_processor.sh quét và xử lý PCAP mới theo chu kỳ (mặc định 10s sleep giữa các vòng lặp).

3.2.2 Trích Xuất Đặc Trưng (Feature Engineering):

Mục đích: Chuẩn hoá các trường của CICFlowMeter về đúng schema mô hình, xử lý giá trị thiếu/ngoại lệ và chuẩn hoá để suy luận ổn định.

Các đặc trưng được tạo:

- Destination Port, Flow Duration
- Total Fwd Packets, Total Length of Fwd Packets
- Flow Bytes/s, Flow Packets/s
- Flow IAT Mean, Flow IAT Std
- Fwd IAT Mean, Bwd IAT Mean
- Fwd Packet Length Mean, Bwd Packet Length Mean
- Packet Length Std, Packet Length Max, Packet Length Min
- ACK Flag Count, FIN Flag Count, PSH Flag Count

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

Xử lý Giá Trị Bất Thường:

- Chuẩn hoá tên cột theo ánh xạ CICFlowMeter → schema, tự bổ sung các cột thiếu (điền NaN chờ impute).
- Chuyển tất cả đặc trưng sang số, thay ±inf → NaN.
- Điền khuyết bằng median theo cột (median tính từ dữ liệu train và lưu trong metadata.pkl), sau đó fallback 0 nếu còn thiếu.
- Chuẩn hoá bằng StandardScaler (fit trên train, áp dụng lại ở realtime qua metadata.pkl)

3.2.3 Mô Hình Học Máy (Machine Learning Model):

Thuật toán: RandomForestClassifier được hiệu chỉnh xác suất bằng CalibratedClassifierCV (cv=5). Huấn luyện và đánh giá trong ML_trainer.py.

Qui 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

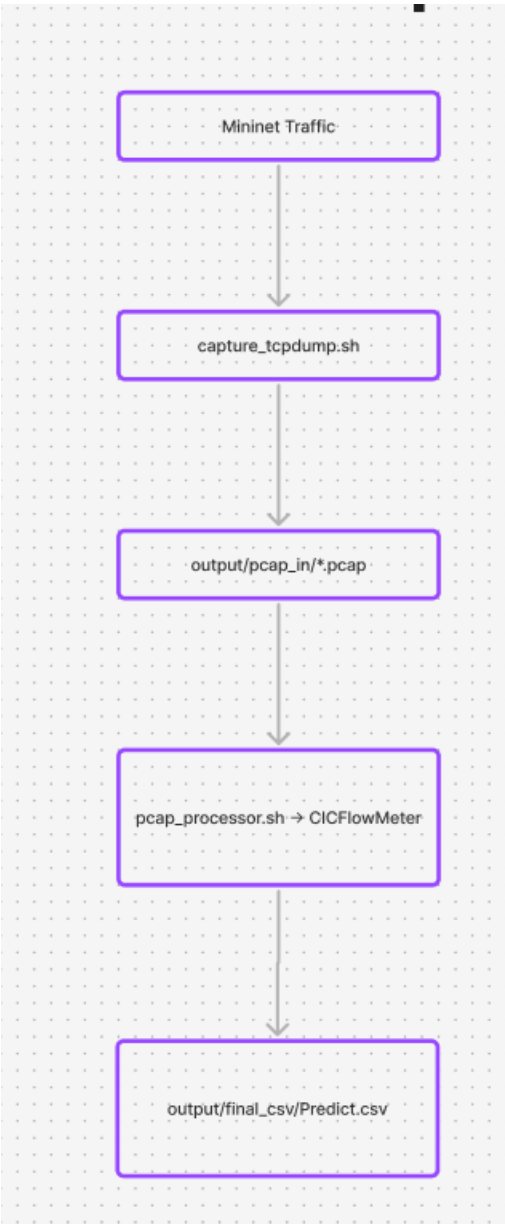
Model Persistence:

- Model được lưu vào `model.pkl`
- Metadata được lưu vào `metadata.pkl`:
    - feature_names: danh sách đặc trưng (17 cột).
    - medians: median theo cột để impute ở realtime.
    - scaler: đối tượng StandardScaler để chuẩn hoá ở realtime.
    - threshold: ngưỡng mặc định (0.5).
    - class_mapping: ánh xạ nhãn gốc → nhị phân (0: benign/normal, 1: attack).

3.3 Qui Trình End-to-End

Qui 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:

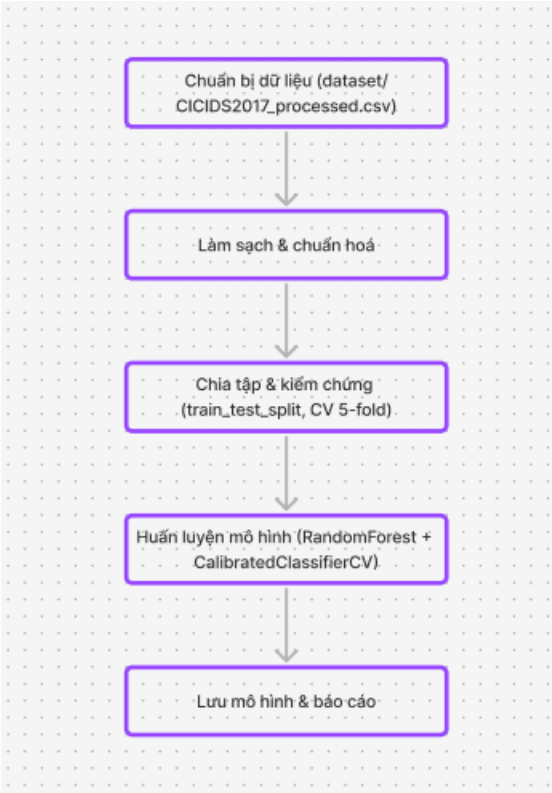
- Tạo topology Mininet và sinh traffic bình thường/DDoS.
- Bắt gói theo cửa sổ bằng capture_tcpdump.sh → file PCAP luân phiên tại output/pcap_in.

- Chuyển PCAP sang CSV đặc trưng bằng pcap_processor.sh → gộp vào output/final_csv/Predict.csv.
- Với huấn luyện mô hình: sử dụng sẵn dataset/CICIDS2017_processed.csv

***Yêu cầu**:*

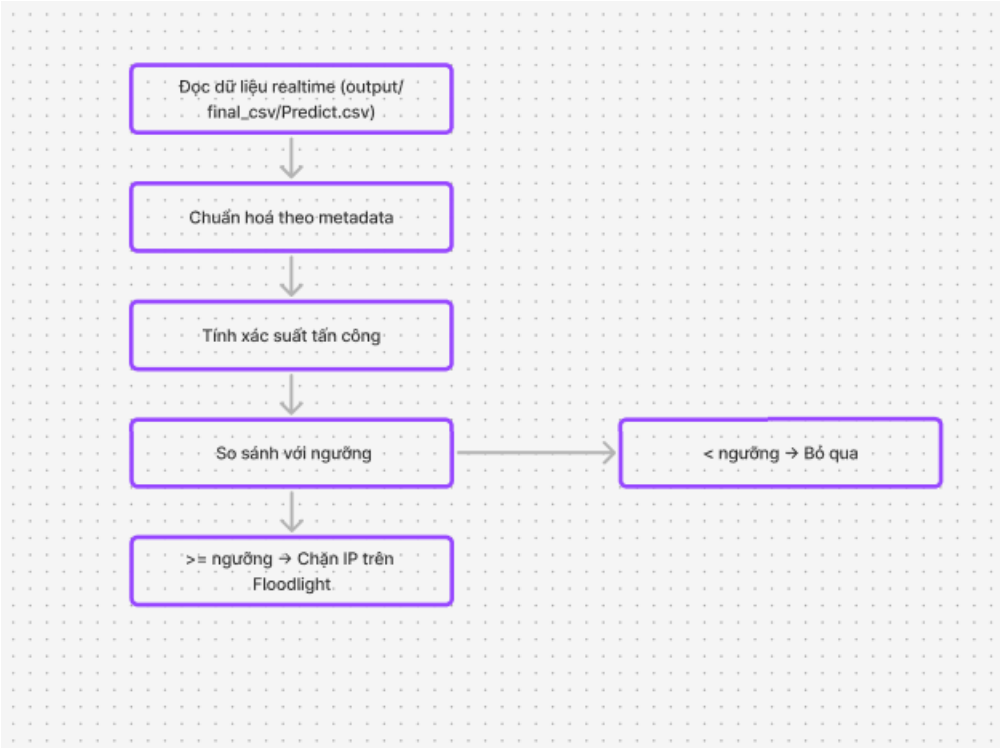
- Floodlight controller đang chạy; topology Mininet hoạt động và có traffic.
- Có đủ quyền để chạy tcpdump và các script (yêu cầu sudo).
- Dữ liệu train: sử dụng dataset/CICIDS2017_processed.csv (mặc định) hoặc CSV CICFlowMeter tự thu thập và gắn nhãn.

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 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:

```
1: root@Ubuntu: /media/source
root@Ubuntu:/media/source# python3 machinelearning/ML_trainer.py --csv dataset/CICIDS2017_processed.csv
2025-12-16 10:37:35,758 | INFO | CV ROC AUC (5-fold): mean=0.9998 std=0.0004
```

```
2025-12-16 10:38:48,774 | INFO | AUC: 0.9998
2025-12-16 10:38:48,774 | INFO | Confusion Matrix:
[[600   0]
 [  3 597]]
2025-12-16 10:38:48,821 | INFO | Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       600
           1       1.00      0.99      1.00       600

    accuracy                           1.00      1200
   macro avg       1.00      1.00      1.00      1200
weighted avg       1.00      1.00      1.00      1200
```

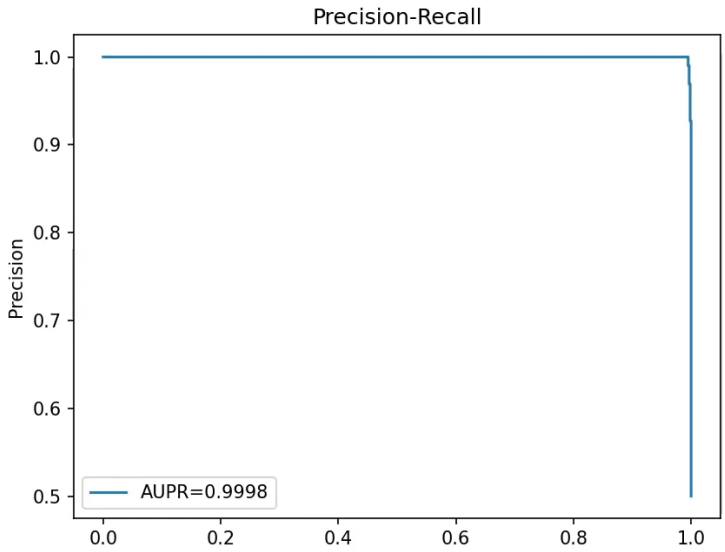
```
           1       1.00      0.99      1.00       600

    accuracy                           1.00      1200
   macro avg       1.00      1.00      1.00      1200
weighted avg       1.00      1.00      1.00      1200

2025-12-16 10:38:51,857 | INFO | Saved plots: model_eval_{roc,pr,cm}.png
2025-12-16 10:39:04,056 | INFO | Saved feature importances to feature_importances.csv
2025-12-16 10:39:07,387 | INFO | Saved model to model.pkl
2025-12-16 10:39:07,389 | INFO | Saved metadata to metadata.pkl
2025-12-16 10:39:07,389 | INFO | Training complete. ACC/AUC logged above. Threshold fixed at 0.50
```

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

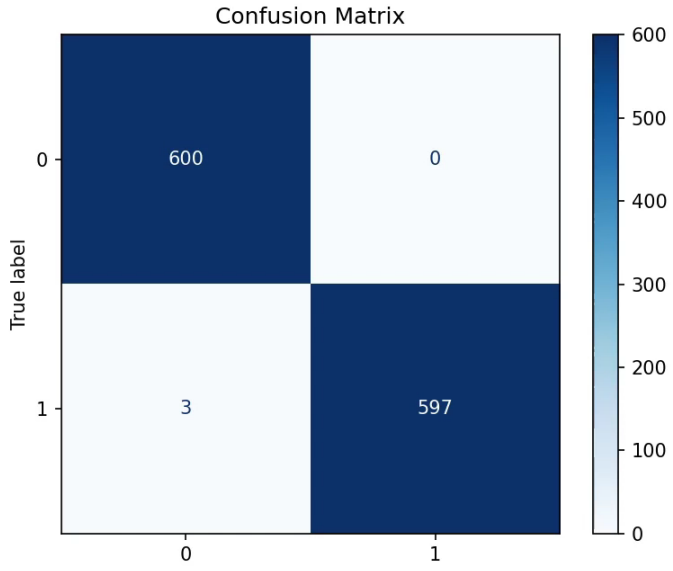
```
root@Ubuntu:/media/source# ls
CICFlowMeter  controller  feature_importances.csv  metadata.pkl  model_eval_cm.png  model_eval_roc.png  output      venv
ciclib        dataset     machinelearning          mininet       model_eval_pr.png  model.pkl           processing
```

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.

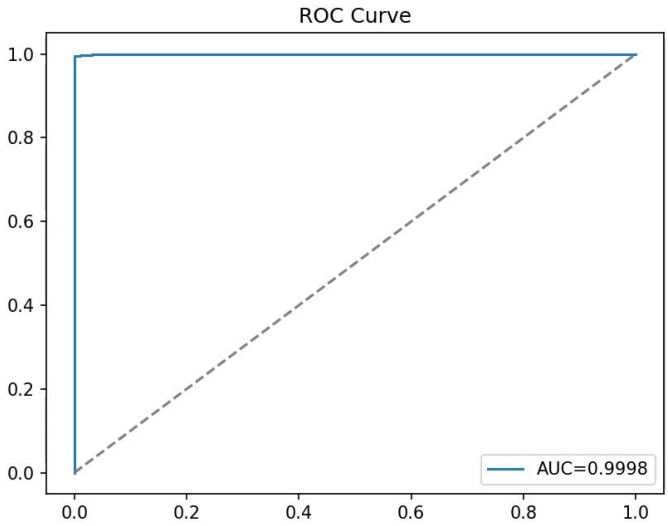
Hình ảnh của các file ".png", đầu tiên là model_eval_pr.png:



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.

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

4.2.1 Yêu cầu cơ bản trước khi chạy

Sau khi huấn luyện xong, để tiến hành chạy mô phỏng quy trình trên, ta cần 4 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 capture_tcpdump.sh để thu thập traffic qua mạng chuyển nó thành file .pcap để terminal C phân tích.
- Terminal C: chạy file pcap_processor.sh để phân tích các file .pcap được bắt thành file .csv, sau đó tổng hợp các dữ liệu (các cột feature và giá trị tương ứng của nó) vào 1 file .csv chung tên Predict.csv để mô hình ML đọc trong "thời gian thực"
- Terminal D: chạy topology.py để tạo một hạ tầng mạng ảo hóa được Floodlight nhận và quản lý

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

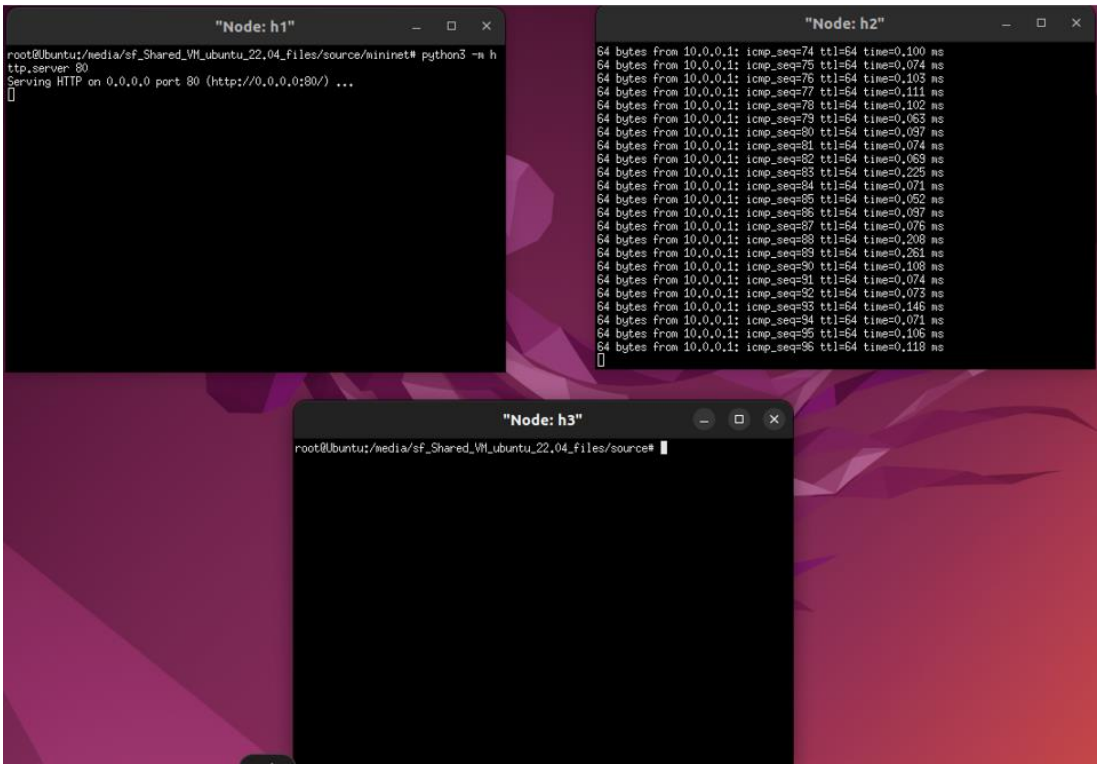
Sau đó chạy lệnh để khởi động Floodlight trên *http://localhost:8080*, để mô hình ML có thể đẩy các rule chặn port lên API của Floodlight.

4.2.2 Toàn bộ quá trình ngăn chặn DDOS

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

```
root@Ubuntu:/media/sf_Shared_VM_ubuntu_22.04_files/source# python3 mininet/topo
logy.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.



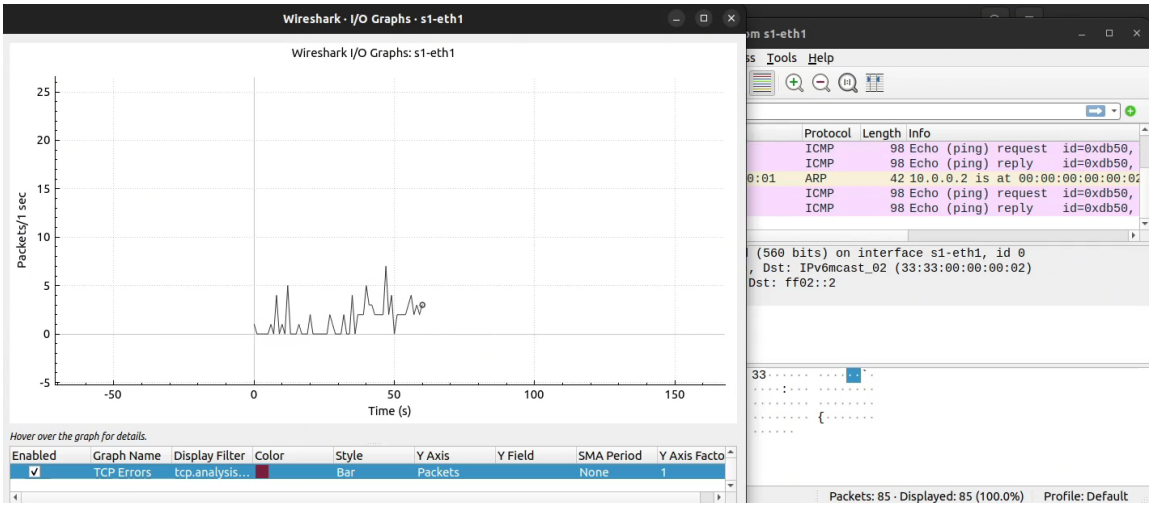
Trong đó:

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

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

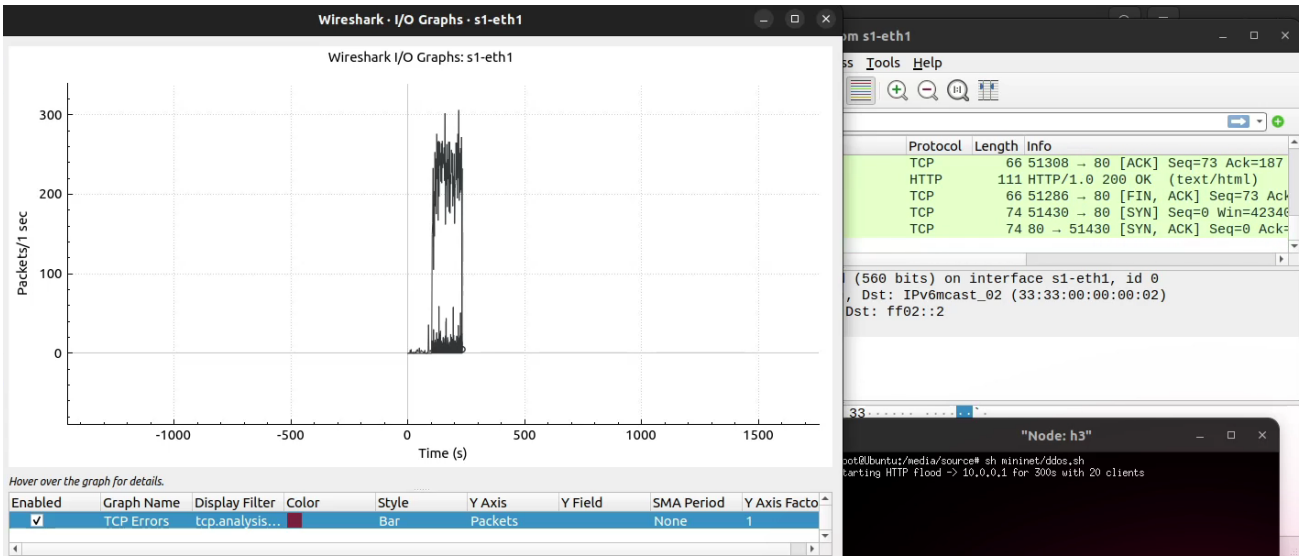
h3: sẵn sàng thực hiện ddos

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.

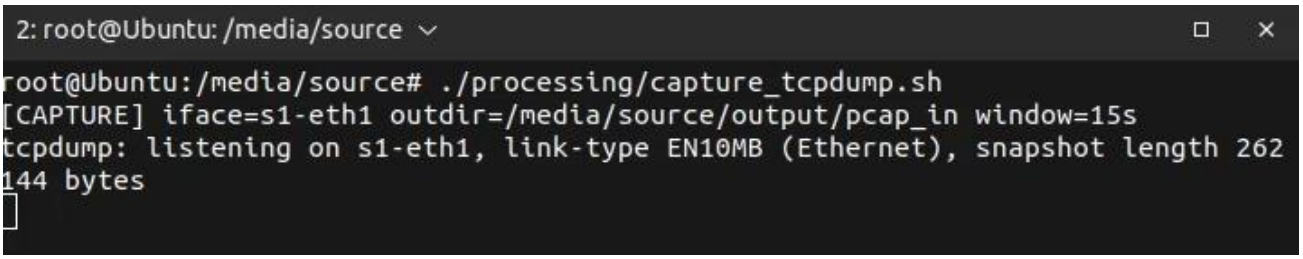
Sau đó, ta thử tấn công DDOS trên h3 để xem biểu đồ lưu lượng thay đổi thế nào.



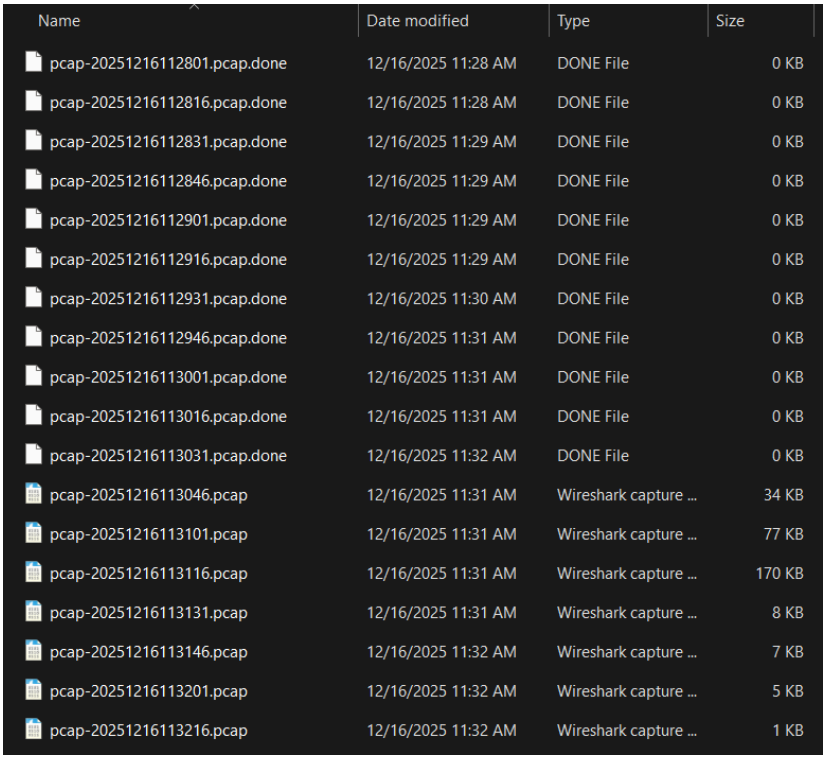
Ta quan sát được rằng có sự biến thiên cao diễn ra. h1 bây giờ đang bị DDOS bằng HTTP flood bởi h3 (h3 được cấu hình để thực hiện ddos trong 300s, mục đích là để giả lập cho ta thấy sự biến thiên về lưu lượng trong 1 khoảng thời gian.)

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 chuyển đổi traffic thành file .pcap như sau:

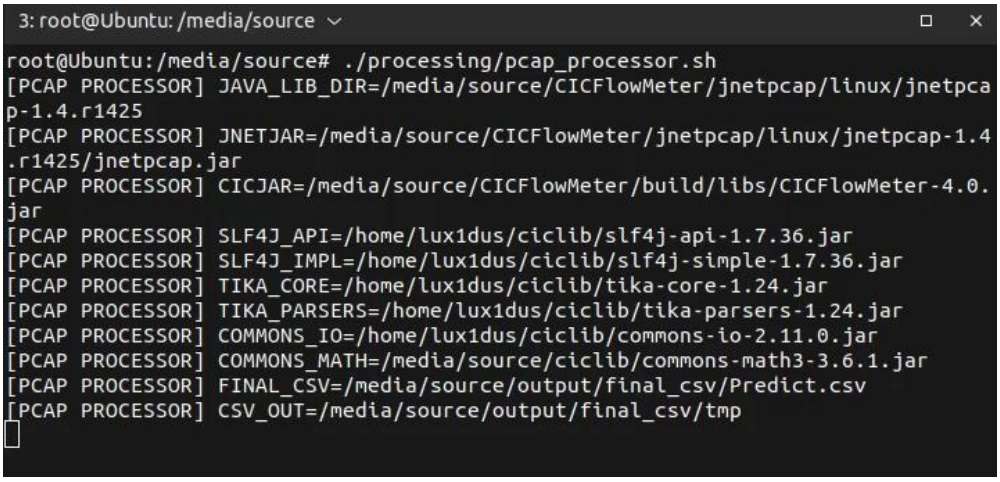


Các lưu lượng đang được ghi vào từng file .pcap.

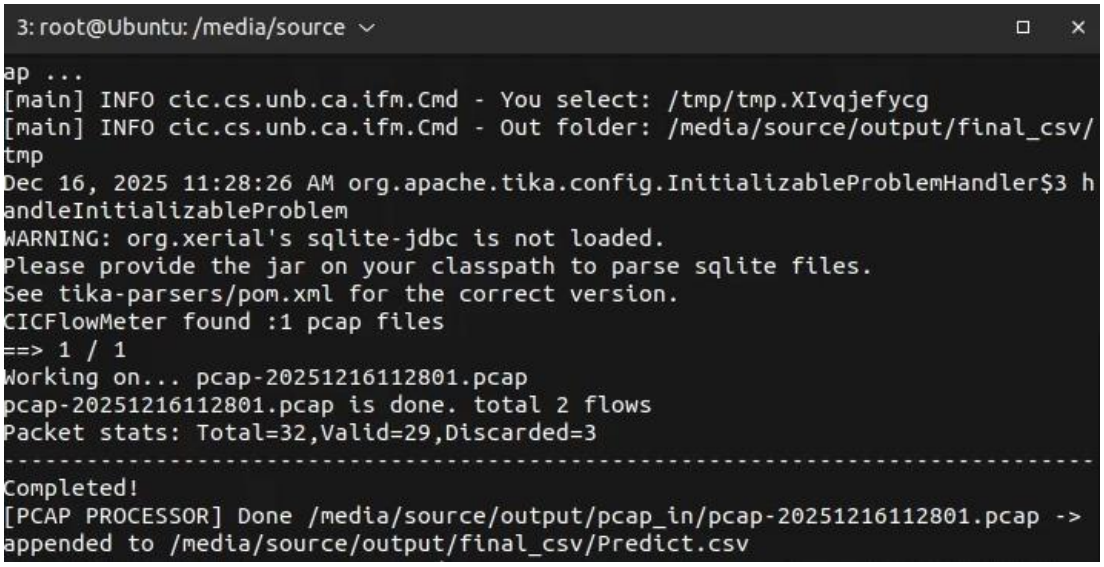


Name	Date modified	Type	Size
pcap-20251216112801.pcap.done	12/16/2025 11:28 AM	DONE File	0 KB
pcap-20251216112816.pcap.done	12/16/2025 11:28 AM	DONE File	0 KB
pcap-20251216112831.pcap.done	12/16/2025 11:29 AM	DONE File	0 KB
pcap-20251216112846.pcap.done	12/16/2025 11:29 AM	DONE File	0 KB
pcap-20251216112901.pcap.done	12/16/2025 11:29 AM	DONE File	0 KB
pcap-20251216112916.pcap.done	12/16/2025 11:29 AM	DONE File	0 KB
pcap-20251216112931.pcap.done	12/16/2025 11:30 AM	DONE File	0 KB
pcap-20251216112946.pcap.done	12/16/2025 11:31 AM	DONE File	0 KB
pcap-20251216113001.pcap.done	12/16/2025 11:31 AM	DONE File	0 KB
pcap-20251216113016.pcap.done	12/16/2025 11:31 AM	DONE File	0 KB
pcap-20251216113031.pcap.done	12/16/2025 11:32 AM	DONE File	0 KB
pcap-20251216113046.pcap	12/16/2025 11:31 AM	Wireshark capture ...	34 KB
pcap-20251216113101.pcap	12/16/2025 11:31 AM	Wireshark capture ...	77 KB
pcap-20251216113116.pcap	12/16/2025 11:31 AM	Wireshark capture ...	170 KB
pcap-20251216113131.pcap	12/16/2025 11:31 AM	Wireshark capture ...	8 KB
pcap-20251216113146.pcap	12/16/2025 11:32 AM	Wireshark capture ...	7 KB
pcap-20251216113201.pcap	12/16/2025 11:32 AM	Wireshark capture ...	5 KB
pcap-20251216113216.pcap	12/16/2025 11:32 AM	Wireshark capture ...	1 KB

Tiếp theo ta chạy file chuyển đổi .pcap thành .csv bên terminal C như sau:



Ta có thể thấy file này đang thành công tương tác và đọc các file .pcap, sau khi xử lý xong sẽ chuyển các file .pcap vào 1 thư mục tên “pcap_done” để không tự động đọc lại file đó nữa. Rồi tổng hợp các thông tin vào 1 file Predict.csv



Sau đó bên terminal A, ta chạy file realtime_floodlight_ML.py với các tham số cần thiết.

```
1: root@Ubuntu: /media/source
root@Ubuntu:/media/source# python3 controller/realtime_floodlight_ML.py --csv output/final_csv/Predict.csv --threshold 0.007
2025-12-16 11:28:33,290 | INFO | Realtime inference start: csv=output/final_csv/Predict.csv threshold=0.007 interval=2s
```

Sau khi cả 3 terminal A, B và C đều chạy tốt, ta sẽ bắt đầu thực hiện DDOS lại bằng h3 lên h1 để xem ML hành động thế nào.

Khi chạy lệnh DDOS trên h3, terminal A và C đều phản ứng như sau:

Bên terminal C đã cập nhật các luồng DDOS vào file Predict.csv:

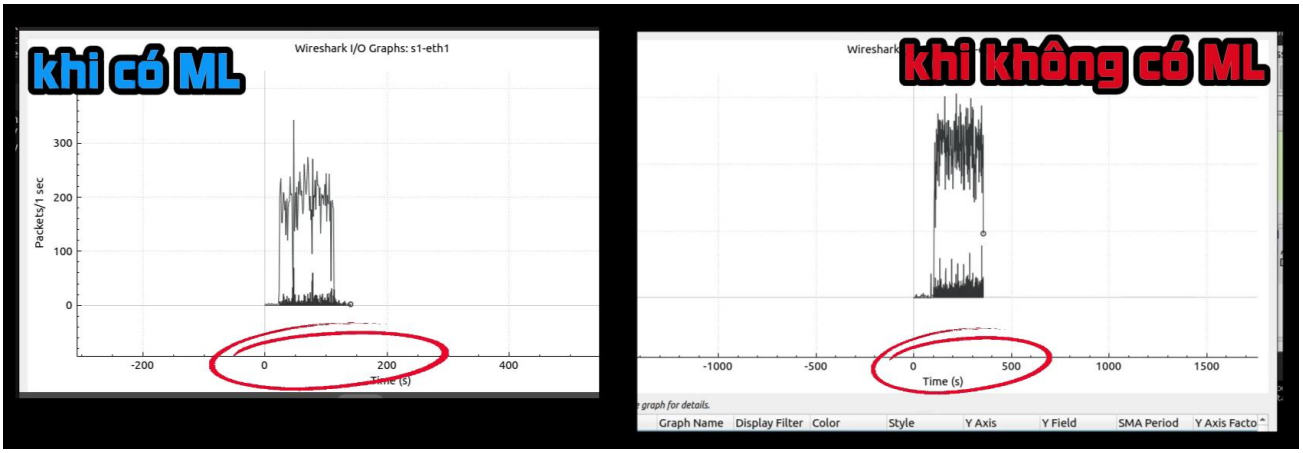
```
3: root@Ubuntu: /media/source
IN Flag
[main] INFO cic.cs.unb.ca.jnetpcap.FlowGenerator - Forward flow closed due to F
IN Flag
[main] INFO cic.cs.unb.ca.jnetpcap.FlowGenerator - Forward flow closed due to F
IN Flag
[main] INFO cic.cs.unb.ca.jnetpcap.FlowGenerator - Forward flow closed due to F
IN Flag
[main] INFO cic.cs.unb.ca.jnetpcap.FlowGenerator - Forward flow closed due to F
IN Flag
[main] INFO cic.cs.unb.ca.jnetpcap.FlowGenerator - Forward flow closed due to F
IN Flag
[main] INFO cic.cs.unb.ca.jnetpcap.FlowGenerator - Forward flow closed due to F
IN Flag
[main] INFO cic.cs.unb.ca.jnetpcap.FlowGenerator - Forward flow closed due to F
IN Flag
[main] INFO cic.cs.unb.ca.jnetpcap.FlowGenerator - Forward flow closed due to F
IN Flag
pcap-20251216112946.pcap is done. total 42 flows
Packet stats: Total=325,Valid=322,Discarded=3
```

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

```
1: root@Ubuntu: /media/source
root@Ubuntu:/media/source# python3 controller/realtime_floodlight_ML.py --csv output/final_csv/Predict.csv --threshold 0.007
2025-12-16 11:28:33,290 | INFO | Realtime inference start: csv=output/final_csv/Predict.csv threshold=0.007 interval=2s
2025-12-16 11:28:36,959 | INFO | Polled 1 rows, alerts=0 (threshold=0.007, pmax=0.003, pmean=0.003)
2025-12-16 11:28:52,057 | INFO | Polled 2 rows, alerts=0 (threshold=0.007, pmax=0.003, pmean=0.003)
2025-12-16 11:29:05,802 | INFO | Polled 3 rows, alerts=0 (threshold=0.007, pmax=0.003, pmean=0.003)
2025-12-16 11:29:18,075 | INFO | Polled 4 rows, alerts=0 (threshold=0.007, pmax=0.003, pmean=0.003)
2025-12-16 11:29:34,848 | INFO | Polled 5 rows, alerts=0 (threshold=0.007, pmax=0.003, pmean=0.003)
2025-12-16 11:29:48,497 | INFO | Polled 6 rows, alerts=0 (threshold=0.007, pmax=0.003, pmean=0.003)
2025-12-16 11:30:27,294 | INFO | Polled 7 rows, alerts=0 (threshold=0.007, pmax=0.003, pmean=0.003)
2025-12-16 11:31:15,647 | INFO | Using static flow endpoint: /wm/staticflowpusher/json
2025-12-16 11:31:15,727 | INFO | Block 10.0.0.3 on 00:00:00:00:00:00:00:02 -> 200
2025-12-16 11:31:15,794 | INFO | Block 10.0.0.3 on 00:00:00:00:00:00:00:04 -> 200
2025-12-16 11:31:15,859 | INFO | Block 10.0.0.3 on 00:00:00:00:00:00:00:03 -> 200
2025-12-16 11:31:15,926 | INFO | Block 10.0.0.3 on 00:00:00:00:00:00:00:01 -> 200
2025-12-16 11:31:15,926 | INFO | Attack detected: src=10.0.0.3 dst=10.0.0.1 proba=0.008
2025-12-16 11:31:15,961 | INFO | Polled 48 rows, alerts=13 (threshold=0.007, pmax=0.008, pmean=0.006)
```

Ở các dòng gần 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) lên tất cả các switch trong mạng

Ta có thể quan sát trực quan hơn việc ngăn chặn DDOS diễn ra thành công thông qua sự so sánh sau:



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 tiếu 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
7. Decision Tree https://www.geeksforgeeks.org/machine-learning/decision-tree/
8. SDN là gì? Tìm hiểu tổng quan về Software Defined Networking (SDN) từ A-Z SDN là gì? Ứng dụng, kiến trúc Software Defined Networking