TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN
LỚP CỬ NHÂN TÀI NĂNG

LƯƠNG TẤN KHANG

PHÁT HIỆN VÀ PHÒNG THỦ TRƯỚC CÁC CUỘC TẤN CÔNG TỪ CHỐI DỊCH VỤ TRONG SOFTWARE-DEFINED NETWORKING DỰA TRÊN CÁC PHƯƠNG THỨC HỌC MÁY VÀ HỌC SÂU

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN CNTT

TP.HCM, NĂM 2020

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN
LỚP CỬ NHÂN TÀI NĂNG

LƯƠNG TẤN KHANG - 1612865

PHÁT HIỆN VÀ PHÒNG THỦ TRƯỚC CÁC CUỘC TẤN CÔNG TỪ CHỐI DỊCH VỤ TRONG SOFTWARE-DEFINED NETWORKING DỰA TRÊN CÁC PHƯƠNG THỨC HỌC MÁY VÀ HỌC SÂU

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN CNTT

GIÁO VIÊN HƯỚNG DẪN
TS. TRẦN TRUNG DŨNG
ThS. LÊ GIANG THANH

KHÓA 2016 - 2020

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

...

Tp.HCM, Ngày .... tháng .... năm 2020

Giáo viên hướng dẫn

NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

...

Khóa luận đáp ứng yêu cầu của Khóa luận cử nhân CNTT.

Tp.HCM, Ngày .... tháng .... năm 2020

Giáo viên phản biện

Lời cảm ơn

Để hoàn thành khóa luận này tôi tỏ lòng biết ơn sâu sắc đến thầy TS. Trần Trung Dũng và thầy ThS. Lê Giang Thanh. Các thầy đã tận tình giúp tôi trong suốt quá trình nghiên cứu và hoàn thành khóa luận này.

Tôi chân thành cảm ơn quý thầy cô trong trường Đại học Khoa học Tự nhiên thành phố Hồ Chí Minh đã giảng dạy và truyền đạt kiến thức cho tôi trong suốt quãng thời gian học tập và nghiên cứu ở trường.

Tôi xin dành lời cảm ơn riêng cho thầy cô khoa Công nghệ thông tin đã tạo điều kiện cơ sở vật chất cho tôi có thể tiến hành các nghiên cứu và thí nghiệm tại phòng bộ môn Mạng máy tính.

Sinh viên thực hiện

Mục lục

Danh mục từ khóa

CNN	Convolutional Neural Network
DDoS	Distributed Denial of Service
DoS	Denial of Service
DNN	Deep neural Network
DT	Decision Tree
IDS	Intrusion Detection Systems
IoT	Internet of Things
LSVM	Linear Support Vector Machine
NB	Naïve Bayes
NN	Neural Network
ReLU	Rectified linear unit
RF	Random forests
RNN	Recurrent neural network
SDN	Software-Defined Networking
SVM	Support Vector Machine

Danh sách hình vẽ

Danh sách bảng

Lời mở đầu

Trong thời đại phát triển nhanh chóng của Internet hiện nay, các phương thức bảo mật và an toàn thông tin đang ngày càng được chú trọng. Trong đó, tấn công từ chối dịch vụ (DoS) và tấn công từ chối dịch vụ phân tán (DDoS) đang trở thành mối đe dọa lên toàn thể các máy chủ trên thế giới. Từ đó, các phương thức và dịch vụ phòng chống, giảm thiểu thiệt hại cũng được ra mắt với rất nhiều các chức năng khác nhau.

Một số vấn đề nổi trội khi thực hiện phòng thủ trước tấn công DoS/DDoS đó chính là chi phí cài đặt và vận hành hệ thống khá tốn kém. Bên cạnh đó, các phương pháp tấn công ngày càng đa dạng với quy mô ngày một lớn, khiến cho các phương pháp phòng chống nhanh chóng trở nên lỗi thời và bị vô hiệu hóa. Trong khóa luận này, tôi hi vọng có thể giải quyết được các vấn đề trên, cung cấp một giải pháp phòng thủ trước tấn công DoS/DDoS trong mạng Software defined network (SDN) hợp lý, gọn nhẹ và dễ dàng triển khai.

Tôi đã tiến hành tìm hiểu các nghiên cứu học thuật cũng như một số các sản phẩm liên quan đến vấn đề này trên thị trường. Từ đó, xem xét khả năng ứng dụng học sâu và học máy vào giải pháp phát hiện tấn công DoS/DDoS. Sau đó tôi thiết kế và xây dựng một proof-of-concept cho phần mềm nhận diện tấn công DoS/DDoS và một proof-of-concept cho hệ thống triển khai trong mạng SDN.

Văn bản này trình bày tổng kết lại quá trình tìm hiểu và xây dựng Hệ thống phát hiện và phòng thủ trước các cuộc tấn công từ chối dịch vụ và từ chối dịch vụ phân tán trong môi trường mạng SDN. Toàn bộ văn bản được chia thành các chương với nội

dung như sau:

Chương 1. Giới thiệu

Trình bày chi tiết nguyên nhân và động lực của tôi khi thực hiện khóa luận này, cũng như mục tiêu đặt ra và hướng tiếp cận của tôi đối với vấn đề.

Chương 2. Cơ sở lý thuyết

Trình bày những khái niệm cơ bản, giúp người đọc có cái nhìn tổng quan về cơ sở lý thuyết của khóa luận cũng như làm nền tảng cho những chương tiếp theo.

Chương 3. Các công trình liên quan

Trình bày những tìm hiểu của tôi về các nghiên cứu phát hiện và phòng thủ trước tấn công DoS/DDoS đã có. Tìm ưu nhược điểm trong các nghiên cứu này.

Chương 4. Mô hình giải pháp của đề tài

Trình bày và giới thiệu giải pháp của tôi đề xuất cho một hệ thống phát hiện và phòng thủ trước tấn công DoS/DDoS.

Chương 5. Mô hình mạng thử nghiệm

Áp dụng giải pháp đề xuất ở chương 4 vào mạng SDN trong môi trường mạng thử nghiệm.

Chương 6. Thử nghiệm và phân tích kết quả

Kiểm lỗi và độ chính xác của hệ thống khi áp dụng vào mạng SDN

Chương 7. Tổng kết

Tổng kết lại quá trình thực hiện khóa luận, những điểm đạt và chưa đạt, đồng thời đề xuất hướng phát triển trong tương lai.

Chương 1

Giới thiệu

1.1 Tổng quan về đề tài

Phát hiện và phòng thủ trước tấn công DoS/DDoS không phải là một lĩnh vực mới. Kể từ khi cuộc tấn công DDoS lần đầu tiên xuất hiện vào năm 1999, thì tấn công DDoS đang ngày càng trở nên đa dạng và phức tạp. Từ đó, loại tấn công mạng này được xem như là một vấn nạn, một mối đe dọa rất lớn đối với các hệ thống mạng và hệ thống máy chủ trên thế giới. Phát hiện tấn công DoS/DDoS theo đánh giá của các chuyên gia và các tổ chức an ninh mạng hiện nay là bài toán rất khó. Nguyên nhân là vì các cách thức và các cuộc tấn công ngày càng đa dạng và xuất hiện với mức độ dày đặc. Bên cạnh đó, với nền tảng vạn vật kết nối (IoT) phát triển mạnh mẽ, nhu cầu về một mạng lưới đa năng và uyển chuyển dẫn đến sự áp dụng mạnh mẽ Software Defined Network (SDN) vào nền tảng này. Từ đó, phát hiện tấn công DoS/DDoS trong SDN đang là đề tài được chú trọng và nhiều người nghiên cứu.

Các phương thức phát hiện tấn công DDoS hiện nay chủ yếu hiện nay dựa trên việc theo dõi mật độ dữ liệu di chuyển trong hệ thống mạng. Với phương pháp này, việc nhận diện hệ thống đang bị tấn công mật độ cao (high-rate) rất dễ dàng. Tuy nhiên, phương pháp này dễ bị nhầm lẫn giữa tấn công với quá tải do nhu cầu sử dụng tăng cao đột ngột và rõ ràng là không hiệu quả trong việc phát hiện tấn công mật độ thấp

(low-rate). Hiện nay, trong môi trường SDN, phương pháp theo dõi này được áp dụng chủ yếu, ở đó, người ta sẽ theo dõi các dữ liệu thống kê các luồng mở do các thiết bị OpenFlow switch trả về. Bên cạnh đó, các phương pháp thống kê và phân tích luồng dữ liệu cũng được áp dụng rộng rãi. Đây là phương pháp được đánh giá tốt hiện nay, tuy nhiên, để có thể chọn được các đặc trưng thống kê và phân tích hiệu quả, phương pháp này cần một người nghiên cứu có trình độ về thống kê nhất định. Ngoài ra, mỗi khi áp dụng các phương pháp thống kê và phân tích mới, các nhà nghiên cứu phải bỏ công sức làm lại từ đầu, từ đó dẫn đến tốn kém về mặt thời gian và nhân lực.

Hiện nay, các nghiên cứu áp dụng học máy và học sâu trong phát hiện tấn công DoS/DDoS đang xuất hiện ngày một nhiều với các thành tựu đáng kể. Hai phương pháp này đã cho thấy khả năng nhận diện tấn công vượt trội của mình. Từ đó đã thúc đẩy tôi nghiên cứu một hệ thống phát hiện tấn công DoS/DDoS dựa trên nền tảng của các nghiên cứu học máy và học sâu trước đó. Sau đó áp dụng vào hệ thống mạng SDN để có thể phát hiện sớm tấn công DoS/DDoS và giảm thiểu thiệt hại do nó gây ra. Sau cùng, để kiểm thử hệ thống, tôi xây dựng lên hai kịch bản phòng thủ quan trọng: phòng thủ từ nguồn tấn công và phòng thủ từ tấn công bên ngoài.

1.2 Bài toán nghiên cứu của đề tài

Trong khóa luận này, hai mục tiêu chính mà tôi hướng đến là nhận diện và phòng thủ trước tấn công DoS/DDoS trong môi trường mạng SDN.

Trong bài toán nhận diện, tôi áp dụng các phương pháp, các mô hình học máy và học sâu đã được nghiên cứu trước đó để tìm ra được mô hình tốt nhất, cũng như đánh giá ưu và nhược điểm của các mô hình trong các hoàn cảnh khác nhau. Áp dụng huấn luyện nhiều mô hình, tôi tìm ra được mô hình học máy có hướng tiếp cận đơn giản mà kết quả thu được lại tốt nhất là mô hình Decision Tree.

Trong bài toán phòng thủ, tận dụng tính uyển chuyển và tính tập trung của mạng SDN, tôi đề xuất một giải pháp đơn giản mà hiệu quả trong việc ngăn chặn các luồng

tấn công DDoS được chẩn đoán từ mô hình học máy và học sâu.

Kết hợp hai mục tiêu trên, tôi xây dựng nên một hệ thống nhận diện và phòng thủ toàn diện trước tấn công DoS/DDoS trong môi trường mạng SDN.

1.3 Mục tiêu đặt ra và hướng tiếp cận vấn đề

Những mục tiêu mà chúng tôi muốn đạt được như sau:

- Hoàn thành thành một phần mềm proof-of-concept trong việc phân loại tấn công DoS/DDoS.
- Triển khai một hệ thống phòng thủ proof-of-concept trong mạng SDN.
- Huấn luyện trên các mô hình học máy và học sâu khác nhau.
- Dễ dàng thay đổi mô hình nhận diện tấn công trong hệ thống.
- Dễ dàng mở rộng và áp dụng vào các nền tảng mạng khác so với SDN.
- Bảo đảm phần mềm có thể hoạt động theo thời gian thực trên một hệ phần cứng cơ bản.
- Cung cấp giao diện thống kê lưu lượng mạng theo thời gian thực trên nền tảng web.

Từ những mục tiêu trên, hướng tiếp cận của chúng tôi như sau:

- Tập trung xây dựng hệ thống nhận diện và phòng thủ cơ bản, sau đó mới tối ưu hóa sau.
- Ưu tiên sử dụng các mô hình học máy và học sâu đơn giản mà hiệu quả.
- Ưu tiên sử dụng những công cụ mô phỏng mạng SDN gọn nhẹ.

1.4 Đối tượng nghiên cứu

Những đối tượng nghiên cứu chính trong khóa luận này bao gồm:

– Lý thuyết và cách thức tấn công từ chối dịch vụ và từ chối dịch vụ phân tán.
– Lý thuyết và mô hình máy học và học sâu.
– Lý thuyết và ứng dụng mạng Software defined network.
– Các tập dữ liệu về tấn công DoS/DDoS đang có hiện nay.
– Các giải pháp phát hiện tấn công DoS/DDoS bằng các phương pháp học máy và học sâu.
– Các giải pháp phát hiện và phòng thủ trước tấn công DoS/DDoS trong mạng SDN.

Trên đây là toàn bộ nội dung của chương 1 của khóa luận. Qua những gì đã trình bày, tôi hy vọng có thể giúp người đọc có được cái nhìn tổng quát nhất về những vấn đề còn tồn đọng, nguyên nhân, động lực đã thúc đẩy tôi bắt tay vào thực hiện khóa luận này. Bên cạnh đó, tôi cũng đã trình bày những mục tiêu, hướng tiếp cận mà tôi đã vạch ra trước khi bước vào việc tìm hiểu những đối tượng nghiên cứu chính.

Trong chương tiếp theo, tôi sẽ trình bày cơ sở lý thuyết và các khái niệm cơ bản nhất về tấn công từ chối dịch vụ và từ chối dịch vụ phân tán, học máy, học sâu và mạng SDN, giúp người dùng có cái nhìn tổng quan về lý thuyết, dễ nắm bắt hơn khi đọc khóa luận này.

Chương 2

Cơ sở lý thuyết

2.1 Tấn công từ chối dịch vụ (DoS) và tấn công từ chối dịch vụ phân tán (DDoS)

2.1.1 Tấn công từ chối dịch vụ

Tấn công từ chối dịch vụ là gì?

Theo định nghĩa của [7], một cuộc tấn công từ chối dịch vụ diễn ra khi người dùng hợp lệ không có khả năng truy cập thông tin từ các hệ thống, các thiết bị hoặc các tài nguyên mạng do sự can thiệp của nhân tố mạng độc hại. Những dịch vụ bị ảnh hưởng bao gồm thư điện tử, các trang mạng, các tài khoản trực tuyến (ví dụ như ngân hàng), hay các dịch vụ khác dựa trên tài nguyên máy tính hoặc tài nguyên mạng. Tình trạng ”từ chối dịch vụ” là khi hệ thống hay mạng lưới của nạn nhân bị làm ngập lụt cho đến lúc không còn khả năng phản hồi, sụp đổ, hay ngăn chặn sự truy cập từ những người dùng hợp lệ. Tấn công từ chối dịch vụ có thể gây hại cho các tổ chức cả về tiền bạc lẫn thời gian trong lúc dịch vụ của họ không thể truy cập được.

Có rất nhiều phương thức để thực hiện tấn công DoS. Phương thức thông thường diễn ra là khi kẻ tấn công làm ngập lụt mạng lưới hệ thống với lưu lượng. Với phương

thức này, kẻ tấn công gửi hàng loạt yêu cầu đến máy chủ nạn nhân, làm cho nó quá tải. Những yêu cầu dịch vụ này là bất hợp lệ và có địa chỉ giả mạo, sẽ gây nhầm lẫn cho máy chủ khi nó cố gắng xác thực người gửi yêu cầu. Khi những yêu cầu thừa thải được xử lý liên tục, máy chủ sẽ bị quá tải, từ đó tạo nên trạng thái từ chối dịch vụ đối với các yêu cầu hợp lệ.

Những loại tấn công DoS thường gặp

Trong tấn công Smurf [7], những kẻ tấn công gửi các gói tin broadcast ICMP đến một số lượng đáng kể các máy chủ với các địa chỉ IP nguồn giả mạo, thứ thuộc về máy chủ nạn nhân. Các máy chủ nhận những gói tin này sẽ phản hồi và từ đó làm ngập lụt máy chủ nạn nhân với những gói tin phản hồi này.

Tấn công ngập lụt gói tin SYN [7] diễn ra khi một kẻ tấn công gửi một yêu cầu để kết nối đến máy chủ nạn nhân nhưng lại không hoàn thành kết nối đó thông qua việc bắt tay ba bước – một phương thức được dùng trong mạng lưới TCP/IP để tạo kết nối giữa máy chủ và máy khách. Việc không hoàn thành kết nối sẽ dẫn tới cổng kết nối giữ trạng thái ”bị chiếm giữ” và không khả dụng cho các kết nối sau này. Kẻ tấn công cứ liên tiếp tạo các kết nối, chiếm giữ tất cả các cổng và từ đó ngăn chặn những người dùng hợp lệ có thể tạo các kết nối mới.

Những mạng lưới cá nhân có thể bị ảnh hưởng bởi các tấn công DoS mà không phải là mục tiêu trực tiếp. Nếu nhà cung cấp dịch vụ Internet hoặc nhà cung cấp dịch vụ đám mây bị nhắm đến và bị tấn công, thì mạng lưới cũng sẽ trải qua việc mất dịch vụ.

Tấn công ngập lụt UDP [8] diễn ra dựa trên sự không tin cậy của kết nối UDP. Trong tấn công này, rất nhiều gói tin UDP với các cổng ngẫu nhiên được gửi tới nạn nhân. Khi nạn nhân nhận gói tin tại một cổng, nó sẽ tìm kiếm ứng dụng đang lắng nghe cổng đó. Khi nó không tìm được, nó sẽ phản hồi với một gói tin ICMP báo lỗi. Khi một lượng lớn gói tin UDP độc hại được nhận, nạn nhân sẽ tiêu thụ rất nhiều tài nguyên trong việc phản hồi với các gói tin ICMP. Điều này sẽ ngăn chặn nạn nhân

phản hồi những người dùng hợp lệ.

Slowris [8] là loại tấn công cho phép một máy tính làm sập các máy chủ web chỉ với một lượng băng thông nhỏ nhưng lại không ảnh hưởng đến các dịch vụ khác của nạn nhân. Slowris thực hiện việc này bằng cách giữ càng nhiều kết nối đến máy chủ web nạn nhân càng lâu càng tốt. Nó sẽ tạo kết nối đến với máy chủ web nạn nhân, tuy nhiên nó chỉ gửi một phần của yêu cầu. Nó liên tục gửi các HTTP header nhưng không bao giờ hoàn thành chúng. Nạn nhân sẽ liên tục giữ những kết nối độc hại này được mở, từ đó dẫn đến quá tải giới hạn kết nối, dẫn đến việc từ chối dịch vụ tới các người dùng hợp lệ.

2.1.2 Tấn công từ chối dịch vụ phân tán

Tấn công từ chối dịch vụ phân tán là gì?

Có rất nhiều định nghĩa tấn công DDoS từ các nghiên cứu khác nhau, tuy nhiên chung quy đều có chung suy nghĩ và ý nghĩa. Theo như [7], tấn công từ chối dịch vụ diễn ra khi nhiều máy tính vận hành cùng lúc với nhau để tấn công vào một mục tiêu. Những kẻ tấn công thường tận dụng mạng lưới Botnet – một nhóm các thiết bị truy cập Internet bị chiếm giữ - để dẫn tới một tấn công quy mô lớn. Những kẻ tấn công sẽ tìm kiếm và chiếm giữa các mạng lưới hay thiết bị kết nối Internet có bảo mật kém, từ đó tạo nên mạng Botnet. Tấn công DDoS tạo một lượng yêu cầu cực lớn đến nạn nhân, từ đó gia tăng sức mạnh tấn công, đồng thời rất khó để phòng thủ bởi tính đa dạng và rộng khắp từ các nguồn tấn công.

Nam-Seok Ko và các cộng sự [9] giải thích rằng tấn công DDoS là sự kết hợp của việc sử dụng một lượng cực lớn lưu lượng mạng bởi kẻ tấn công nhắm vào hệ thống mục tiêu thông qua việc vận hành đồng thời một lượng lớn máy tính được phân tán trên Internet. Theo như nghiên cứu này, lưu lượng tấn công sẽ ngăn cản người dùng thực tế truy cập vào tài nguyên của nạn nhân bằng việc thâu tóm tất cả băng thông mạng và tài nguyên bên trong hệ thống.

Fallah và các cộng sự [10] thì cho rằng, tấn công DDoS bắt đầu bằng việc kẻ tấn công sẽ truy tìm các máy chủ có các lỗi bảo mật trên Internet, để có thể xâm nhập và điều khiển chúng gửi một lượng lớn các gói tin đến máy chủ nạn nhân. Hơn nữa, khi kẻ tấn công thành công, chúng đã có thể sử dụng toàn bộ tài nguyên máy chủ như CPU, stack space trong giao thức phần mềm, hoặc Internet link capacity, từ đó nạn nhân không thể cung cấp dịch vụ đến với người dùng thực tế của nó.

Từ góc nhìn của doanh nghiệp, Yoon [11] giải thích rằng tấn công DDoS được thể hiện bằng việc kẻ tấn công bên ngoài trên Internet sử dụng BotMaster điều khiển từ xa các Botnet nhằm thực hiện tấn công vào các trang web quan trọng từ đó các dịch vụ kinh doanh thiết yếu như Internet banking, e-government, e-trading, e-commerce, vv, bị vô hiệu hóa đối với người dùng.

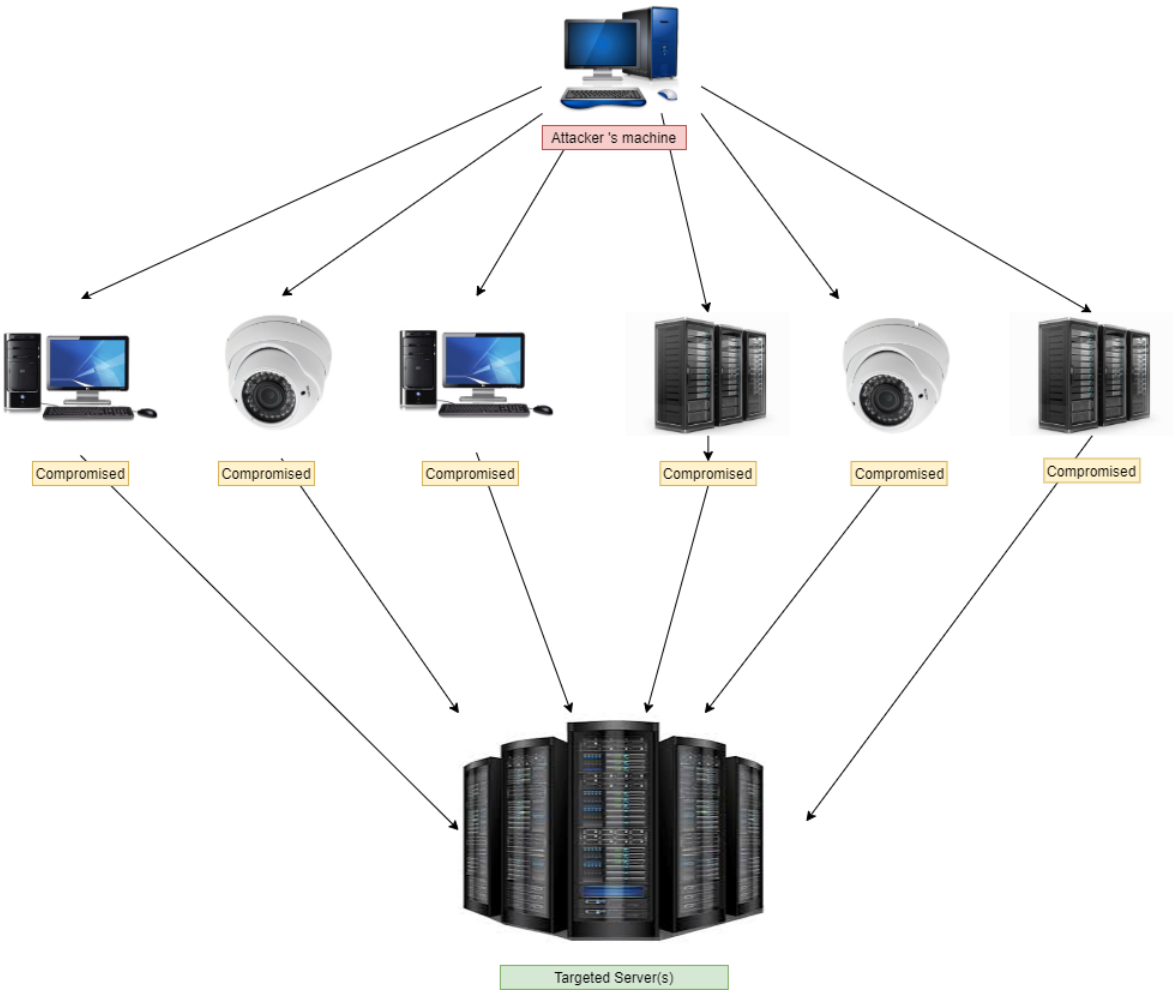
Theo [7], các cuộc tấn công DDoS đã gia tăng cường độ khi ngày càng có nhiều thiết bị IoT ra đời. Các thiết bị IoT thường sử dụng mật khẩu mặc định và không được bảo mật kỹ càng khiến chúng rất dễ bị xâm nhập và khai thác. Việc lây nhiễm các thiết bị IoT thường không được nhiều người chú ý, kẻ tấn công có thể xâm nhập và điều khiển hàng trăm nghìn thiết bị IoT, từ đó thực hiện một cuộc tấn công DDoS quy mô lớn mà người sở hữu thiết bị không hay biết hề hay biết.

Trên thực tế hiện nay, mặc cho các cuộc tấn công DDoS rất khó ngăn chặn và giảm thiểu thiệt hại thì vẫn chưa có một phương thức phòng chống tiêu chuẩn nào được đưa ra. Điều này phần lớn là do DDoS cố gắng bắt chước lưu lượng thông thường nhưng với mật độ yêu cầu khổng lồ.

Mô hình tấn công DDoS được mô tả như trong hình 2.1.

Các loại tấn công từ chối địch vụ phân tán thường gặp

Nhìn chung, theo như các định nghĩa thì tấn công DDoS được thực hiện khi có rất nhiều thiết bị cùng lúc thực hiện tấn công DoS. Một số loại tấn công DDoS có thể liệt kê như sau:



Hình 2.1: Mô hình tấn công DDoS

Tấn công khuếch đại NTP [8], kẻ tấn công sẽ khai thác những máy chủ NTP công khai để làm quá tải nạn nhân với lưu lượng UDP. Loại tấn công này được xem là tấn công khuếch đại cưỡng đoạt bởi vì tỷ lệ phản hồi trong kịch bản này nằm ở đoạn 1:20 đến 1:200 và hơn thế nữa. Điều này có nghĩa là một kẻ tấn công nắm một lượng lớn máy chủ NTP có thể dễ dàng tạo nên tấn công DDoS với băng thông và lưu lượng cực lớn.

Tấn công ngập lụt HTTP [8] là loại tấn công mà kẻ tấn công sẽ gửi những yêu cầu GET/POST HTTP trông giống các yêu cầu hợp lệ tới máy chủ web hoặc ứng dụng. Tấn công này không gửi các gói tin dị thường, kỹ thuật giả mạo hay phản chiếu, và yêu cầu ít băng thông hơn các kiểu tấn công khác để làm sập máy chủ nạn nhân. Đây

là loại tấn công hiệu quả nhất khi nó buộc máy chủ hay ứng dụng phải sử dụng tối đa tài nguyên cho mỗi yêu cầu.

Các đợt tấn công DoS/DDoS điển hình

Theo [12] ghi nhận, vào năm 2016 thế giới chứng kiến một đợt tấn công DDoS ”lớn nhất từ trước đến nay” nhắm vào Dyn – một đơn vị cung cấp dịch vụ DNS của Hoa Kỳ. Cuộc tấn công này là hệ quả từ việc bảo mật yếu kém từ các thiết bị IoT, cho phép kẻ tấn công chiếm quyền kiểm soát và cài đặt botnet. Đỉnh điểm của cuộc tấn công lên đến 1.2Tbps.

Theo [13], vào năm 2018, Github – một công ty cung cấp dịch vụ lưu trữ mã nguồn lớn nhất thế giới có trụ sở tại Hoa Kỳ, đã trải qua một đợt tấn công DDoS lớn nhất lịch sử được ghi nhận, đỉnh điểm lên đến 1.35Tbps. Đây là hệ quả của việc bảo mật yếu kém của các máy chủ Memcached, giúp kẻ tấn công dễ dàng chiếm quyền kiểm soát và thực hiện cuộc tấn công khuếch đại. Kiểu tấn công này được ghi nhận tương tự như tấn công khuếch đại NTP nhưng với quy mô và sức mạnh lớn hơn gấp nhiều lần.

Bảng 2.1 liệt kê một số cuộc tấn công DDoS quy mô lớn được ghi nhận cho đến nay.

2.2 Mạng định nghĩa bằng phần mềm (Software Defined Network)

2.2.1 Software defined network là gì

Khái niệm Software defined network (SDN) nguyên thủy được đặt ra để trình bày ý tưởng và làm việc với Openflow [17] tại đại học Standford (Standford, CA, USA).

Theo D.Kreutz và các cộng sự [1], như định nghĩa ban đầu, SDN đề cập đến một kiến trúc mạng, mà ở đó trạng thái chuyển tiếp (forwarding state) trong plane dữ liệu

Đơn vị bị tấn công	Mô tả	Thời gian	Lưu lượng cao nhất
Github	Dịch vụ lưu trữ dạng web cho việc quản lý mã nguồn	2018	1.35 Tbps [13]
BBC	Đài truyền thông Anh	2015	602 Gbps [14]
Cloudflare	Một công ty Hoa Kỳ chuyên cung cấp các dịch vụ an ninh mạng	2014	~400 Gbps [15]
Spamhaus	Một tổ chức phi lợi nhuận chuyên theo dõi những kẻ spam email và các hoạt động liên quan đến spam	2013	~300Gbps [16]
Dyn	Nhà cung cấp dịch vụ DNS	2016	1.2Tbps [12]

Bảng 2.1: Một số cuộc tấn công DDoS quy mô lớn

(data plane) được quản lý bằng một plane được điều khiển từ xa (remotely controlled plane) tách rời. Kiến trúc SDN được định nghĩa dựa trên 4 yếu tố.

– Control plane và data plane được tách rời. Những tính năng điều khiển được di chuyển ra khỏi những thiết bị mạng, thì những thiết bị mạng này trở thành phần chuyển tiếp gói tin đơn giản.

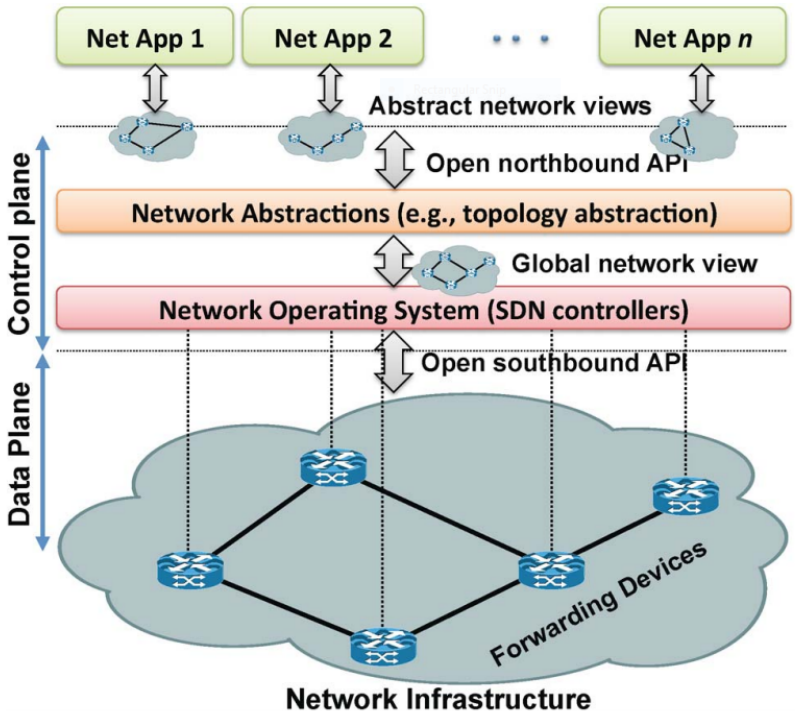
– Quyết định chuyển tiếp được dựa trên luồng (flow), thay vì dựa trên đích đến. Một luồng được định nghĩa là một tập hợp các giá trị trường gói tin đóng vai trò là tiêu chí khớp (bộ lọc) và tập hợp hành động (chỉ dẫn). Trong bối cảnh SDN/Openflow, một luồng là một chuỗi những gói tin giữa một nguồn và một đích. Tất cả gói tin trong một luồng nhận những chính sách dịch vụ giống hệt nhau tại thiết bị chuyển tiếp. Sự trừu tượng của luồng cho phép thống nhất những loại hành động khác nhau của những thiết bị mạng, bao gồm các router, switch, firewall, middleboxes. Lập trình luồng cho phép sự linh hoạt chưa từng có, chỉ giới hạn ở khả năng của các flow-table được triển khai.

– Điều khiển logic được di chuyển ra khỏi các thực thể bên trong, được gọi là SDN controller hay Network operating system (NOS). NOS là nền tảng phần mềm được chạy trên công nghệ máy chủ và cung cấp những tài nguyên thiết yếu trừu

tượng để tạo điều kiện cho việc lập trình các thiết bị chuyển tiếp dựa trên một chế độ mạng trừu tượng và tập trung logic. Mục đích của nó thì giống với hệ điều hành truyền thống.

– Mạng có thể lập trình được thông qua những ứng dụng phần mềm ở tầng đầu của NOS, mà ở đó tương tác với những thiết bị ở data plane. Đây là đặc tính cơ bản của SDN được xem là giá trị chính của nó.

Hình 2.2 thể hiện kiến trúc và sự trừu tượng cơ bản của SDN.



Hình 2.2: Kiến trúc của SDN [1]

Openflow [17] được đề xuất để tiêu chuẩn giao tiếp giữa các switch và controller trong kiến trúc SDN [18]. Các tác giả nhận định rằng, thật khó để cộng đồng nghiên cứu mạng có thể kiểm tra ý tưởng mới ở các phần cứng hiện tại. Việc này diễn ra bởi vì mã nguồn của phần mềm chạy trên các switch không thể được tùy chỉnh và các kiến trúc mạng thì luôn cố định, dẫn tới những ý tưởng mạng mới không thể được kiểm tra trên những cài đặt lưu lượng thực tế. Bằng việc xác định các đặc trưng thông thường trên những bảng luồng trong các switch Ethernet, các tác giả đã cung cấp cách điều

khiển switch mà không cần quan tâm đến mã nguồn trên các thiết bị đó.

Openflow lần đầu được triển khai trong khuôn viên mạng đại học [17]. Ngày nay, có ít nhất 9 trường đại học ở Hoa Kỳ triển khai công nghệ này. Mục đích của Openflow là cung cấp nền tảng mà có thể cho phép các nhà nghiên cứu chạy thử nghiệm những sản phẩm mạng. Tuy nhiên, SDN và Openflow được gắn kết như chiến lược để gia tăng chức năng chức năng của mạng trong khi giảm thiểu chi phí và độ phức tạp phần cứng. Openflow Networking Foundation (ONF) được thành lập vào năm 2011 bởi Deutsche Telekom, Facebook, Google, Microsoft, Verizon và Yahoo để khuyến khích áp dụng mạng dựa trên SDN và Openflow. Hiện tại, ONF có hơn 95 thành viên bào gồm thành viên sáng lập.

Bảng 2.2 liệt kê một số công ty cung cấp Openflow switch trên thị trường.

Switch company	**Series**
Arista	Arista extensible modular operating system (EOS), Arista 7124FX application switch
Ciena	Ciena Coredirector running firmware version 6.1.1
Cisco	Cisco cat6k, catalyst 3750, 6500 series
HP	HP procurve series- 5400 zl, 8200 zl, 6200 yl, 3500 yl, 6600
NEC	NEC IP8800
Open vSwitch	Software switch.

Bảng 2.2: Một số công ty cung cấp OpenFlow Switch

Trong khuôn khổ của khóa luận này, Open vSwitch sẽ được sử dụng là một thành phần chính trong triển khai SDN, được giới thiệu chi tiết trong mục 2.2.2.

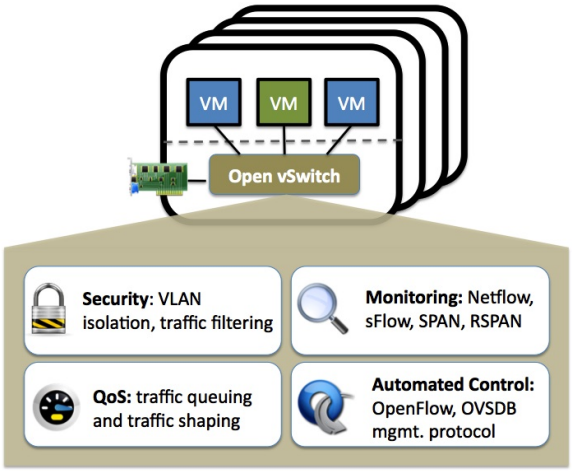
2.2.2 Open vSwitch

Open vSwitch (OVS) là gì?

Open vSwitch là switch phần mềm đa lớp bản quyền mã nguồn mở Apache 2. Mục tiêu là để triển khai một nền tảng sản phẩm switch chất lượng hỗ trợ các giao diện quản lý tiêu chuẩn và mở ra chức năng chuyển tiếp để mở rộng và kiểm soát bằng lập

trình.

OVS rất thích hợp với chức năng như một switch ảo trong môi trường máy ảo. Ngoài việc hiển thị các giao diện điều khiển và hiển thị tiêu chuẩn cho lớp mạng ảo, nó còn được thiết kế để hỗ trợ phân phối trên nhiều máy chủ vật lý. OVS hỗ trợ nhiều công nghệ ảo hóa dựa trên Linux bao gồm Xen/XenServer, KVM và Virtualbox. Hình 2.3 thể hiện kiến trúc tổng quan và chức năng của OVS.



Hình 2.3: Kiến trúc tổng quan và chức năng của OVS [2]

2.2.3 Vai trò của SDN controller

Tổng quan

Dựa theo các định nghĩa đã được đưa ra, SDN controller là hệ quả của việc tách rời Data plane và Control plane trong hệ thống mạng. Nó có vai trò trung tập điều khiển tất cả các hoạt động của mạng bằng việc điều khiển các switch và router thông qua flow-table. Nói cách khác, SDN controller như là bộ não của toàn bộ hệ thống. Để có thể giao tiếp với các thiết bị mạng, SDN controller sử dụng một giao thức chuẩn là Openflow như đã đề cập tại mục 2.2.1. Hình 2.2 cho thấy, SDN controller như một cầu nối giữa các thiết bị mạng và các ứng dụng ở phần trên.

Hiện nay có rất nhiều phần mềm hay framework hỗ trợ việc lập trình SDN controller như OpenDaylight, ONOS, POX, Ryu, vv. Trong khuôn khổ của dự án này,

Ryu framework được sử dụng như thành phần chính để lập trình và thực thi SDN controller, được giới thiệu chi tiết bên dưới mục ”Framework RYU”.

Framework RYU

Theo [19], Ryu Controller là một dự án mã nguồn mở dưới giấy phép của Apache 2.0, được viết hoàn toàn dựa trên Python, được hỗ trợ và triển khai bởi trung tâm dữ liệu đám mây NTT. Phần mã nguồn chính có thể được tìm thấy trên Github, được cung cấp và hỗ trợ bởi Open Ryu Community. Nó hỗ trợ quản trị mạng NETCONF và OF-config, còn được biết đến là Openflow. Ryu đã được kiểm định với OVS switch, Hewlett Packard, IBM và NEC. Hiện tại nó đang hỗ trợ đến phiên bản Openflow 1.5.

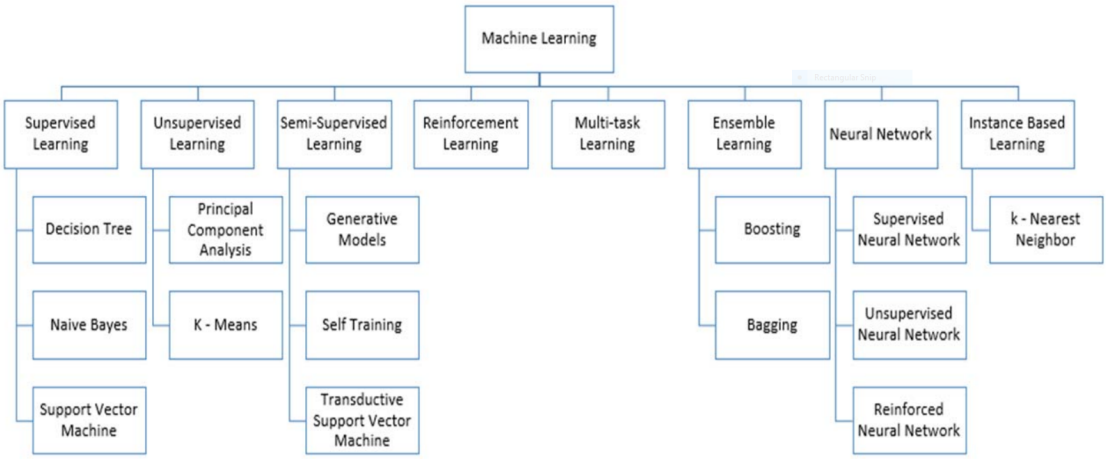
Cũng như các SDN controller khác, Ryu cũng tạo ra các gói tin Openflow, quản lý các sự kiện liên quan tới những gói tin đến và đi. Nó còn có một danh sách phong phú các thư viện hỗ trợ xử lý hoạt động của các gói tin. Về hỗ trợ các giao thức southbound, Ryu hợp tác với các giao thức như Xflow (Netflow và Sflow), OF-config, NETCONF, Open vSwitch Database Protocol, vv, VLAN và GRE cũng được hỗ trợ bởi những thư viện gói tin của Ryu.

2.3 Học máy (Machine Learning)

2.3.1 Học máy là gì?

Theo A.Dey [3] trích dẫn, học máy được dùng để dạy máy móc cách để xử lý dữ liệu hiệu quả hơn. Đôi khi, sau khi xem xét dữ liệu, chúng ta không thể giải thích được mẫu hoặc trích xuất thông tin từ chúng. Trong trường hợp này, chúng ta có thể áp dụng học máy. Với vô số tập dữ liệu (dataset) hiện có, nhu cầu về học máy ngày càng tăng. Rất nhiều ngành công nghiệp từ dược liệu cho đến quân sự đều ứng dụng học máy để trích xuất thông tin liên quan từ dữ liệu. Công việc chính của học máy là học từ dữ liệu. Có rất nhiều nghiên cứu đã tìm ra được cách để máy móc có thể học từ chính chúng [20], [21]. Có rất nhiều nhà toán học và lập trình viên ứng dụng nhiều hướng

tiếp cận để tìm ra giải pháp cho vấn đề này. Một vài trong số chúng được biểu thị trong hình 2.4.



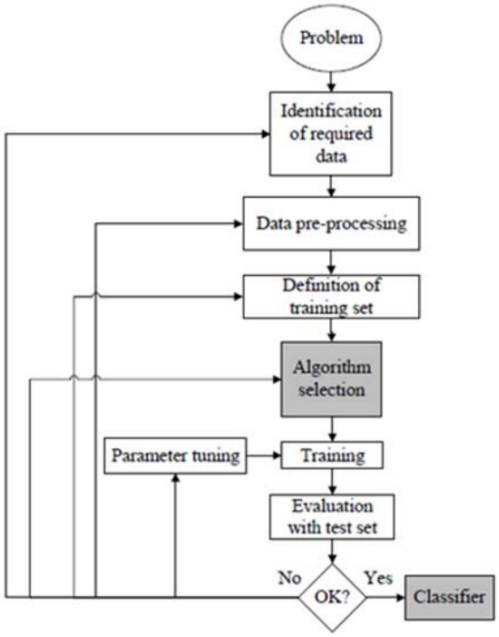
Hình 2.4: Một số giải thuật học máy [3]

2.3.2 Mô hình học máy

Theo như hình 2.4, học máy được phân ra thành rất nhiều loại như Supervised learning (học có giám sát), Unsupervised learning (học không có giám sát), Semi-supervised learning (học bán giám sát), vv. Trong khuôn khổ của khóa luận này, tôi chỉ giới thiệu một số giải thuật là Support Vector Machine, Naïve Bayes, Decision Tree và Random Forest trong nhóm Học có giám sát, vì đây là các giải thuật được tôi áp dụng vào khóa luận.

Học có giám sát

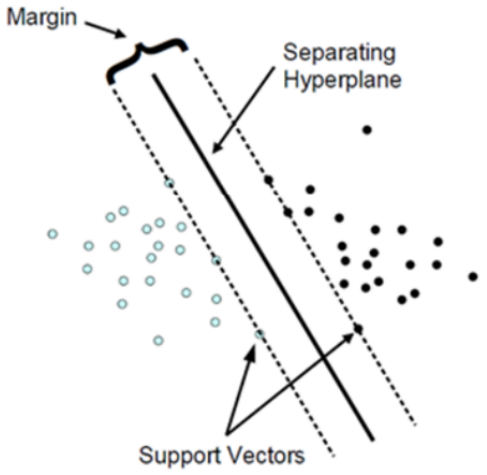
Các thuật toán học có giám sát là những thuật toán cần sự hỗ trợ từ bên trong. Đầu vào của dữ liệu cần chia ra thành tập huấn luyện (train set) và tập kiểm tra (test set). Train set có giá trị đầu ra, thứ cần được dự đoán hoặc phân loại. Tất cả các thuật toán đều được học hình mẫu từ train set và sau đó vận dụng chúng vào test set để đánh giá [22]. Giải thuật học có giám sát được biểu thị trong hình 2.5.



Hình 2.5: Giải thuật học máy có giám sát [3]

Support Vector Machine

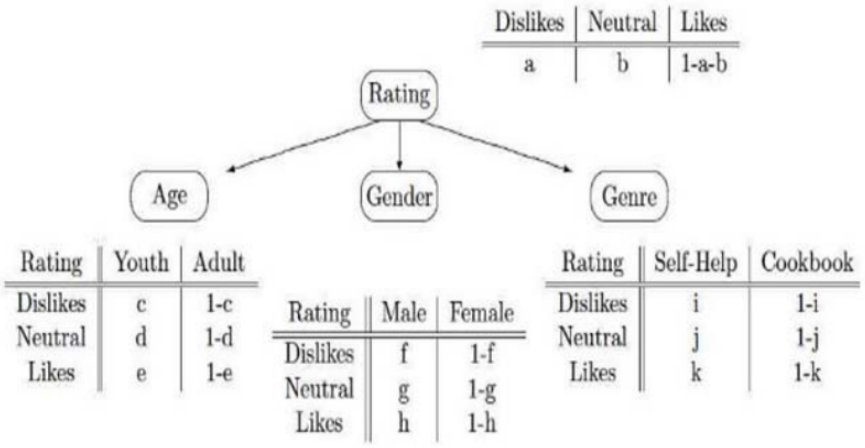
Một kỹ thuật học máy state-of-the-art được sử dụng rộng rãi là Support Vector Machine (SVM). Nó được dùng chủ yếu trong bài toán phân loại. SVM hoạt động dựa trên nguyên lý của tính toán biên. Các biên được vẽ sao cho khoảng cách giữa biên đến các lớp là lớp nhất và vì vậy giảm thiểu sai số phân loại. Mô tả SVM được biểu thị trong hình 2.6.



Hình 2.6: Mô tả mô hình SVM [3]

Naïve Bayes

Naïve Bayes được áp dụng chủ yếu trong lĩnh vực phân loại văn bản. Nó được sử dụng chủ yếu cho việc phân cụm và phân lớp. Kiến trúc cơ bản của Naïve Bayes phụ thuộc vào xác suất có điều kiện. Nó tạo ra cây dựa trên xác suất xảy ra của chúng. Những cây này được gọi là Bayesian Network. Ví dụ cho Bayesian Network được biểu thị trong hình 2.7.



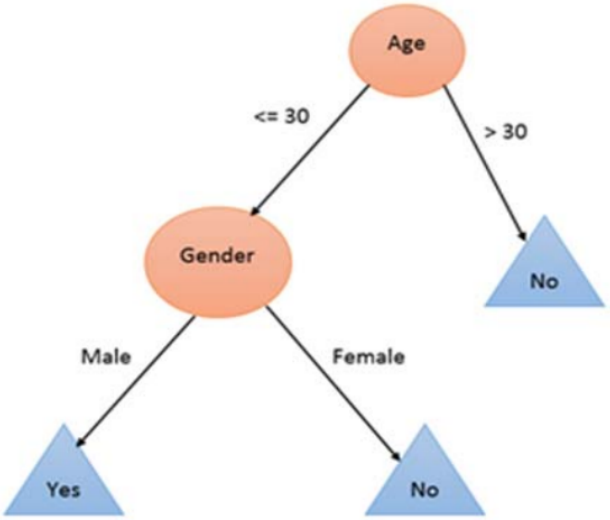
Hình 2.7: Một ví dụ về Bayesian Network [3]

Decision Tree

Decision trees (những cây quyết định) là những loại cây gom nhóm các thuộc tính dựa trên giá trị của chúng. Decision tree được sử dụng chủ yếu cho việc phân lớp. Môi cây chứa nhiều nút và nhiều nhánh. Mỗi nút biểu diễn cho những thuộc tính trong một nhóm để phân lớp và mỗi nhánh biểu diễn một giá trị mà nút đó có thể lấy. Một ví dụ cho Decision tree trong hình 2.8.

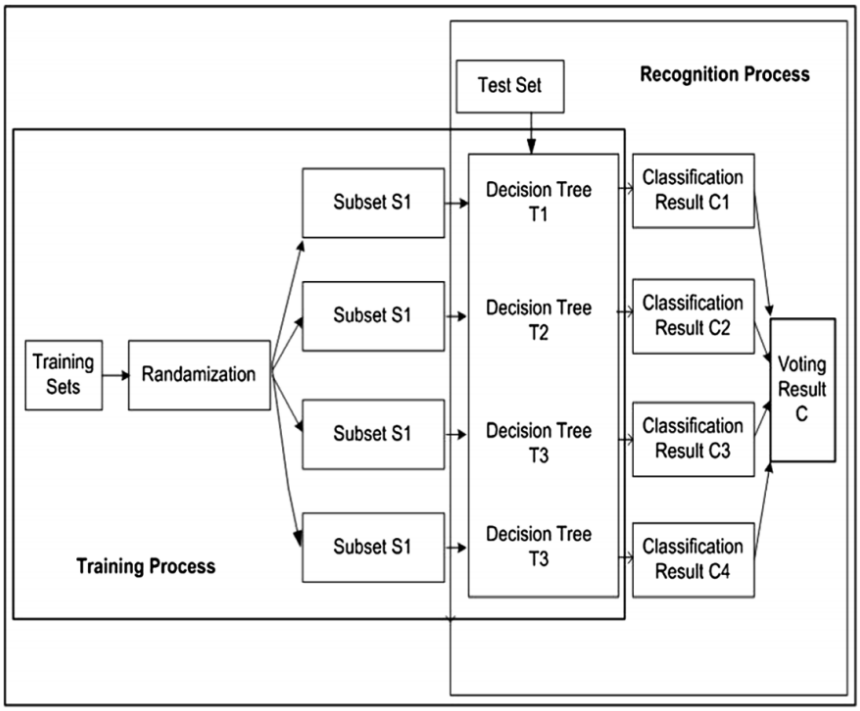
Random Forest

Random forest được đề xuất lần đầu bởi Leo Breiman từ đại học California vào năm 2001. Nó kết hợp nhiều máy phân lớp đơn giản (các cây quyết định) độc lập với nhau. Kết quả của phân lớp của một mẫu được đưa ra bởi nhiều cây quyết đinh, kết quả cuối



Hình 2.8: Một ví dụ cho Decision Tree

cùng sẽ được bình chọn dựa trên kết quả của các cây quyết định này [23].Toàn bộ quy trình phân lớp dựa trên Random Forest được biểu diễn như trong hình 2.9.



Hình 2.9: Quá trình máy phân lớp Random forest

2.3.3 Thư viện scikit-learn

Scikit-learn là thư viện cung cấp sẵn một lượng lớn các giải thuật học máy có giám sát và không có giám sát thông qua giao diện lập trình Python.

Đây là thư viện mã nguồn mở, giấy phép cấp bởi BSD và được phân phối sẵn trên nhiều nền tảng hệ điều hành Linux.

Scikit-learn được phát triển lần đầu bởi David Cournapeau như một dự án mùa hè tại Google năm 2007. Sau đó Matthieu Brucher tham gia dự án và bắt đầu sử dụng thư viện này như một phần trong luận văn của anh ấy. Năm 2010, INRIA tham gia và lần đầu xuất bản phiên bản đầu tiên (v0.1 beta) vào cuối tháng 1 năm 2010. Dự án hiện tại có trên 30 nhà phát triển đóng góp và được tài trợ trả phí từ INRIA, Google, Tinyclues và Python software foundation.

2.4 Học sâu (Deep learning)

2.4.1 Học sâu là gì

Theo X. Du và các cộng sự [24], học sâu trong quá khứ được phát triển từ mạng nơ-ron nhân tạo, và bây giờ nó trở nhành một lĩnh vực thịnh hành của học máy. Nghiên cứu về mạng nơ-ron nhân tạo bắt đầu từ những năm 1940. McCulloch và các cộng sự [25] đã đề xuất mô hình McCulloch-Pitts với việc phân tích và tổng hợp những nét đặc trưng của mạng nơ-ron. Khái niệm học sâu được đưa ra lần đầu vào năm 2006. Sau đó, học sâu vẫn tiếp tục được phát triển mạnh. Hiện tại, có rất nhiều tên tuổi nổi bật như Geoffrey Hinton, Yoshua Bengio, Yann LeCun và Andrew Ng. Họ là những người dẫn đầu về hướng nghiên cứu học sâu.

Một số doanh nghiệp, ví dụ như Google và Facebook đã có rất nhiều nghiên cứu đạt được thành công và áp dụng trong nhiều lĩnh vực khác nhau. Đơn cử như Google’s AlphaGo đã đánh bại Lee Sedol – kỳ thủ cờ vây chuyên nghiệp người Hàn Quốc, cho thấy khả năng học rất mạnh của học sâu. Hơn nữa, Google’s DeepDream và một phần

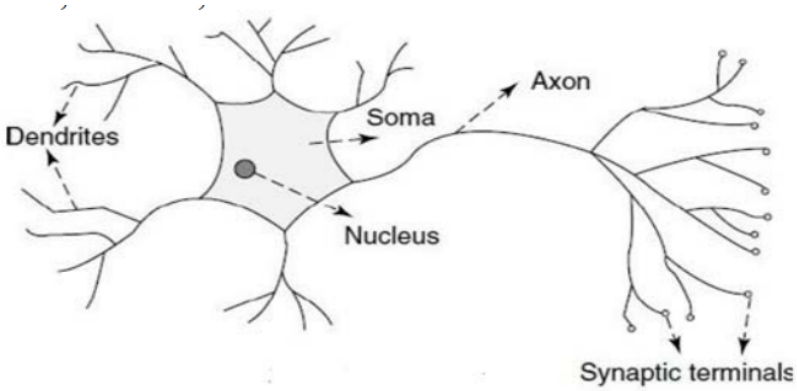
mềm tuyệt vời, thứ không chỉ phân loại hình ảnh mà còn tạo ra được những hình vẽ nhân tạo kỳ lạ dựa trên chính kiến thức của nó. Bên cạnh đó, Facebook với DeepText – một kỹ thuật hiểu văn bản dựa trên học sâu, có thể phân loại một lượng dữ liệu khổng lồ, cung cấp dịch vụ phản hồi sau khi xác định những đoạn trò chuyện của người dùng và xóa các đoạn thư spam.

2.4.2 Mô hình học sâu

Khái niệm và phân loại

Mạng nơ-ron - Neural Network

Mạng nơ-ron (hay mạng nơ-ron nhân tạo hay ANN) được kế thừa từ khái niệm nơ-rơn sinh học. Một nơ-rơn như một cấu trúc trong não. Để hiểu được mạng nơ-ron, người ta cần phải hiểu cách hoạt động của nơ-ron. Một nơ-ron bao gồm 4 phần chính là dendrites, nucleus, soma và axon (Hình 2.10).

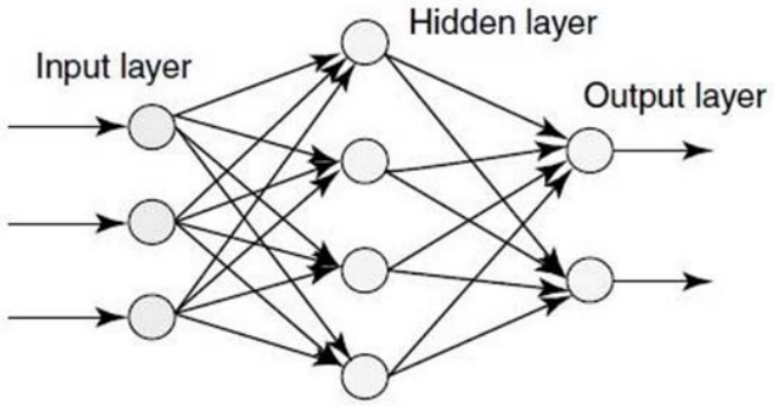


Hình 2.10: Mạng nơ-ron sinh học

Các dendrite nhận tín hiệu điện. Soma xử lý tín hiệu điện. Đầu ra của quá trình được vận chuyển bởi axon đến cổng dendrite nơi mà đầu ra được gửi tiếp đến nơ-ron tiếp theo. Necleus là trái tim của nơ-ron. Kết nối bên trong nơ-ron được gọi là mạng nơ-ron nơi mà các xung điện di chuyển khắp não.

Một mạng nơ-ron nhân tạo cũng hoạt động tương tự. Nó hoạt động với 3 tầng. Tầng đầu vào (input) nhận dữ liệu đầu vào (tương tự dendrite). Những tầng ẩn thì xử

lý dữ liệu (tương tự soma và axon). Cuối cùng, tầng đầu ra (output) gửi kết quả đã được tính toán (tương tự dendrite terminal). Cấu trúc ví dụ của mạng nơ-ron nhân tạo được biểu thị trong hình 2.11



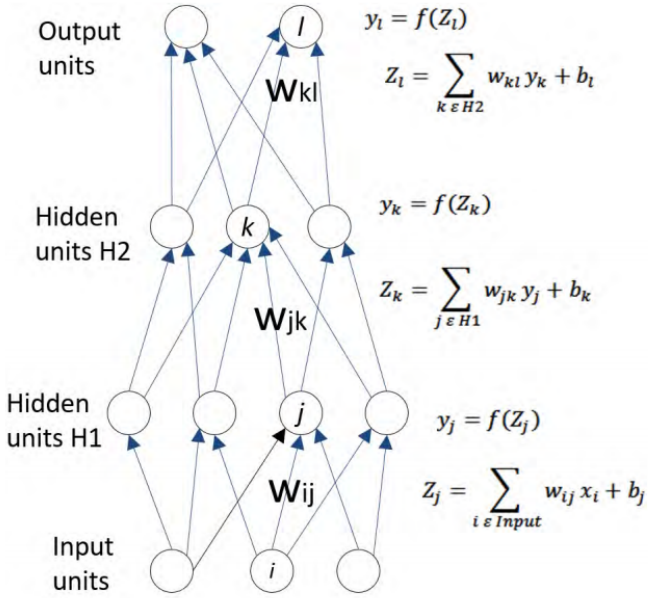
Hình 2.11: Mạng nơ-ron nhân tạo

Theo Shrestha và các cộng sự [4], mạng nơ-ron nhân có thể được chia ra thành các loại sau:

- Mạng nơ-ron suy luận tiến (Feedforward Neural Network)
- Mạng nơ-ron hồi quy (Recurrent Neural Network (RNN))
- Mạng chức năng cơ sở xuyên tâm (Radial Basis Function Neural Network)
- Mạng nơ-ron ánh xạ đặc trưng tự tổ chức (Kohonen Self Organizing Neural Network)
- Mạng nơ-ron mô-đun

Trong mạng nơ-ron suy luận tiến, những luồng thông tin chỉ đi theo 1 chiều từ lớp đầu vào đến lớp đầu ra (đi qua các nút ẩn nếu có). Chúng không có tạo thành bất cứ vòng lặp hay luận lùi. Hình 2.12 biểu thị một triển khai cụ thể của mạng nơ-ron suy tiến đa lớp với các giá trị và các hàm được tính toán theo chiều tiến. Z là tổng trọng số của các đầu vào và y biểu diễn một hàm không tuyến tính f của Z ở mỗi lớp. W biểu diễn cho những trọng số giữa 2 đơn vị trong những lớp liền kề biểu diễn bởi ký tự chỉ

số dưới và b biểu diễn cho giá trị sai khác của đơn vị.

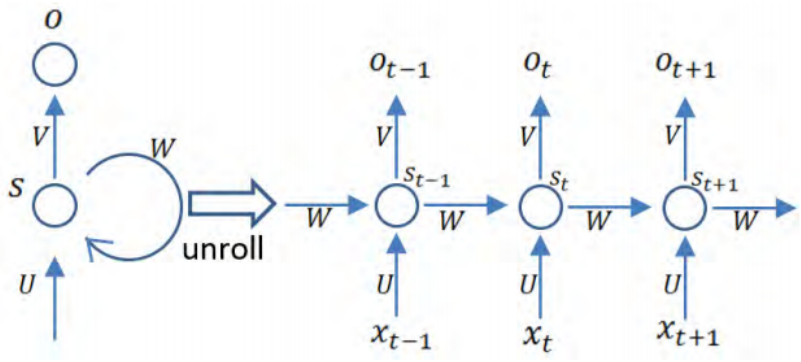


Hình 2.12: Mạng nơ-ron suy luận tiến [4]

Không giống như mạng nơ-ron suy luận tiến, việc xử lý đơn vị trong RNN có vòng lặp. Đầu ra của lớp trước trở thành đầu vào cho lớp sau, thứ đơn thuần chỉ là 1 lớp trong mạng, vì vậy chính đầu ra của lớp cũng trở thành chính đầu vào của nó biểu diễn một vòng tự lặp. Điều này cho phép mạng có ký tức về những trạng thái trước đó và sử dụng nó để ảnh hướng lên đầu ra hiện tại. Một lợi thế lớn của đặc điểm này so với mạng nơ-ron suy tiến là RNN có thể nhận chuỗi đầu vào và trả về chuỗi đầu ra, vì vậy nó rất hữu ích cho các ứng dụng cần xử lý đầu vào dạng chuỗi theo thời gian như nhận diện giọng nói, phân loại từng khung hình của video, vv. Hình 2.13 biểu diễn một triển khai của RNN. Trong đó x^t biểu diễn cho đầu vào tại thời điểm t. U, V và W là các tham số học được chia sẽ bởi tất cả bước. O_t là đầu ra tại thời điểm t. S_t biểu diễn trạng thái tại thời điểm t.

Mạng chức năng cơ sở xuyên tâm được dùng trong phân lớp, xấp xỉ hàm, những vấn đề dự doán theo thời gian, vv.

Mạng nơ-ron ánh xạ đặc trưng tự tổ chức có thể tự tổ chức mô hình mạng dựa trên dữ liệu đầu vào bằng việc học không giám sát.



Hình 2.13: Mạng nơ-ron hồi quy [4]

Mạng nơ-ron mô-đun tách một mạng lớn thành các mô-đun mạng nơ-ron nhỏ độc lập. Những mạng nhỏ hơn giải quyết công việc khác nhau và sau đó kết hợp tại một điểm đầu ra của toàn bộ mạng.

Một số khái niệm về các hàm huấn luyện

Hàm kích hoạt (Activation function)

Theo [26], những hàm kích hoạt là những hàm dùng trong mạng nơ-ron để tính toán tổng trọng số của đầu vào và những sai khác (bias), thứ quyết định nơ-ron được thực hiện hay không. Nó thao tác với dữ liệu thông qua quá trình gradient thường là giảm gradient và sau đó sinh ra kết quả cho mạng nơ-ron mà có chứa tham số cho dữ liệu. Hàm kích hoạt có thể tuyến tính hoặc không tuyến tính tùy thuộc vào hàm mà nó biểu diễn và được dùng để điều khiển đầu ra cho mạng nơ-ron. Một số hàm kích hoạt biểu diễn trong công thức (2.1) và (2.2).

– Hàm Softmax:

$$f(x) = \frac{exp(x_i)}{\sum_j exp(x_j)} \tag{2.1}$$

– Hàm ReLU:

$$f(x) = max(0, x) \tag{2.2}$$

Hàm mất mát (Loss function)

Hàm mất mát đơn giản là hàm dùng để đánh giá mức độ tốt của mô hình giải thuật. Nếu dự đoán tốt hàm sẽ cho ra giá trị thấp, ngược lại hàm cho ra giá trị cao. Công thức của hàm mất mát Categorical Cross Entroy biểu diễn ở công thức (2.3).

$$Loss = -\sum_{i=1}^{outputsize} y_i \cdot log\hat{y}_i \tag{2.3}$$

2.4.3 Thư viện Tensorflow và Keras

Tensorflow hiện nay là bộ thư viện học sâu nổi tiếng nhất được phát triển bởi Google. Nó lần đầu tiên được công bố vào cuối năm 2015, trong khi phiên bản ổn định đầu tiên ra đời năm 2017. Tensorflow là thư viện mã nguồn mở dưới giấy phép của Apache Opensource. Người ta có thể sử dụng, tinh chỉnh, và phân phối lại phiên bản tinh chỉnh mà không cần phải trả một khoản nào cho Google.

Keras là một thư viện mã nguồn dưới bản quyền MIT. Keras cung cấp một API ở mức cao, nằm trên các thư viện học sâu như Tensorflow, Theano, CNTK với mục tiêu giúp đơn giản hóa việc tiếp cận nền tảng học sâu. Keras được xây dựng với mục đích cung cấp giao diện thân thiện, mô-đun hóa, và dễ mở rộng.

Năm 2017, Google's Tensorflow team quyết định hỗ trợ Keras nằm trực tiếp trong phần nhân của thư viện tensorflow.

Chương 3

Các công trình liên quan

Trong chương này, chúng tôi sẽ tiến hành tìm hiểu, khảo sát các công trình liên quan đến chủ đề của khóa luận.

3.1 Phát hiện tấn công DDoS với phương pháp thống kê

R. Doriguzzi-Corin và các cộng sự [6] cho rằng, đo lường các đặc tính thống kê của các thuộc tính lưu lượng mạng là phương pháp thông dụng để phát hiện tấn công DDoS, thường liên quan đến việc quan sát sự biến đổi entropy của trường header trong các gói tin. Bằng định nghĩa này, entropy là đo lường của sự đa dạng hay sự ngẫu nhiên trong tập dữ liệu. Phương thức phát hiện tấn công DDoS dựa trên entropy đã được đưa ra ở các nghiên cứu học thuật vào những năm 2000, dựa trên giả sử rằng, trong suốt quá trình tấn công DDoS băng thông, tính ngẫu nhiên của đặc trưng lưu lượng biến đổi đột ngột. Lý do là đặc trưng của các cuộc tấn công DDoS băng thông thường có một số lượng lớn kẻ tấn công, thông thường là các thiết bị bị xâm nhập gửi một lượng lớn yêu cầu đến nạn nhân. Do đó, các cuộc tấn công này tường gây ra sự sụt giảm trong việc phân phối một số thuộc tính lưu lượng truy cập, chẳng hạn như địa chỉ IP đích, hoặc sự gia tăng trong việc phân phối các thuộc tính khác, ví dụ như địa

chỉ IP nguồn. Cách xác định tấn công DDoS thường phụ thuộc vào giá trị trung bình của ngưỡng của các chỉ số phân phối này.

Feinstein và các cộng sự [27] đã trình bày kỹ thuật phát hiện DDoS dựa trên việc tính toán entropy của IP nguồn và phân phối Chi bình phương. Tác giả đã quan sát sự thay đổi trong entropy IP nguồn và thống kê Chi bình thương thông qua sự biến động của các lưu lượng hợp lệ thì nhỏ, so với độ lệch do lưu lượng tấn công gây ra. Tương tự vậy, P. Bojovic và các cộng sự [28] đã kết hợp entropy với đặc tính lưu lượng băng thông để phát hiện tấn công DDoS băng thông.

Một hạn chế thường thấy ở kỹ thuật dựa trên entropy là sự bắt buộc chọn lấy một ngưỡng phát hiện chính xác. Các hệ thống mạng khác nhau sẽ có độ lệch về băng thông khác nhau, dẫn đến thách thức trong việc áp dụng một ngưỡng chính xác sao cho giảm tối thiểu các tỷ lệ nhận diện sai trong các kịch bản tấn công khác nhau. Một giải pháp được đề xuất bởi Kumar và các cộng sự [29] là gán linh động giá trị ngưỡng để có thể tự động thích nghi với lưu lượng mạng biến động thông thường.

3.2 Phát hiện DDoS với phương pháp học máy và học sâu

3.2.1 Một số dataset được sử dụng hiện nay

Hiện nay có rất nhiều dataset dành cho lĩnh vực an toàn thông tin ra đời, trong đó, phần lớn đề cập đến vấn đề phát hiện và phòng thủ trước tấn công DoS/DDoS. M. Ring và các cộng sự [30] đã có một khảo sát chi tiết các dataset liên quan đến an toàn thông tin, trong phần này tôi chỉ trích xuất ra các dataset liên quan đến tấn công DoS/DDoS.

Booters [31]. Booters là dịch vụ tấn công từ chối dịch vụ được sự dụng bởi các tin tặc. Santanna và các cộng sự [31] đã công bố dataset sử dụng 9 loại tấn công của Booters nhắm vào các máy tính trong mạng lưới của họ. Kết quả được ghi lại dưới dạng gói tin có lên tới 250GB lưu lượng mạng. Từng gói tin riêng lẻ thì không được dán

nhãn, tuy nhiên các loại tấn công khác nhau được tách ra các tệp khác nhau. Dataset được công khai tuy nhiên tên của các loại tấn công bị ẩn đi do vấn đề về quyền riêng tư.

ISCX 2012 [32], được tạo ra vào năm 2012 bởi Canadian Institute for Cybersecurity (CIC) bằng việc bắt các gói tin trong môi trường mạng mô phỏng trong vòng 1 tuần. Tác giả sử dụng các phương pháp động để tạo ra dataset với các lưu lượng bình thường cũng như lưu lượng độc hại. Tác giả chia thành 2 phiên bản. Phiên bản alpha định nghĩa các kịch bản tấn công, phiên bản beta định nghĩa các kịch bản của người dùng thông thường bao gồm gửi email, lướt web, vv. Dựa vào phương pháp này, tác giả đã tạo ra dataset khá hoàn thiện, trong đó có các loại tấn công phổ biến mà bao gồm cả tấn công DoS và DDoS.

CIC-IDS-2017 [33] và CSE-CIC-IDS-2018 [34] đây là hai dataset cũng được thực hiện bởi CIC. Trong đó, CIC-IDS-2017 được công bố vào năm 2017, tác giả sử dụng các kỹ thuật tương tự như khi xây dựng dataset ISCX 2012. Tuy nhiên với phiên bản mới này, tác giả đã bổ sung nhiều loại tấn công hơn, chi tiết hơn về cách dán nhãn tấn công cũng như giới thiệu phần mềm phân tích lưu lượng mạng CICFlowMeter [35]. Với kết quả của phần mềm này, các nhà nghiên cứu có thể dễ dàng áp dụng các phương pháp học máy và học sâu mà không cần phải tiển xử lý hàng trăm GB thông tin các gói tin. Hơn nữa, năm 2018, tác giả đã cho ra đời dataset CSE-CIC-IDS-2018 (gọi tắt là CICIDS2018), với việc kết hợp với AWS cloud service, để có thể tạo ra một mạng mô phỏng phức tạp sát với mạng lưới thực tế. Từ đó, tác giả áp dụng nhiều kịch bản tấn công làm cho dataset này trở thành dataset đầy đủ và tốt nhất hiện nay.

DARPA [36]. DARPA là dataset được dùng rộng rãi trong phát hiện xâm nhập mạng, được tạo bởi MIT Lincoln Lab trong mạng mô phỏng. Dataset chứa nhiều loại tấn công như DoS, buffer overflow, port scan, rootkits.

KDD CUP 99 [37] dựa trên DARPA dataset và trở thành dataset được sử dụng rộng rãi. Dataset này không chứa đựng thông tin gói tin hay thông tin flow, mà đơn giản là chỉ chứa các chỉ mục. Ở đó bao gồm các thuộc tính của các kết nối TCP, các

thuộc tính cấp cao như số lần đăng nhập sai, và hơn 20 thuộc tính khác nữa.

NSL-KDD [38]. NSL-KDD là phiên bản nâng cấp của KDD CUP 99 với việc loại bỏ phần lớn các thông tin dư thừa. Dataset này có 150,000 điểm dữ liệu được chia ra thành tập huấn luyện và tập kiểm tra.

3.2.2 Một số công trình sử dụng phương pháp học máy

Có nhiều công trình nghiên cứu sử dụng học máy vào việc phát hiện tấn công DoS/DDoS. He và các cộng sự [39] đã sử dụng 9 loại giải thuật học máy gồm Linear Regression, SVM (linear, RBF, Polynomial kernel), Decision Tree, Naïve Bayes, Random Forest, K-means, Gaussian EM. Trong đó kết quả tốt nhất đạt được là độ chính xác cao (99.7%) và tỷ lệ nhận diện sai thấp (<0.07%) với giải thuật Linear SVM.

Tương tự, R. Doshi và các cộng sự [40] đã áp dụng các phương pháp học máy để phát hiện tấn công từ chối dịch vụ tại nguồn trong hệ thống mạng các thiết bị IoT. Ở đây, tác giả chủ yếu phát hiện các thiết bị IoT bị lợi dụng để trở thành công cụ tấn công DDoS. Kết quả cho thấy, các phương pháp học máy cho độ chính xác rất cao và trong đó mô hình Linear SVM cũng được đánh giá rất tốt với độ chính xác lên đến 99.1%.

3.2.3 Một số công trình sử dụng phương pháp học sâu

Song song với các công trình nghiên cứu áp dụng học máy, các công trình áp dụng học sâu cũng xuất hiện khá nhiều gần đây. Trong đó, một số công trình cũng thực hiện so sánh các kết quả của hai phương pháp này để có cái nhìn tổng quan hơn.

Koay và các cộng sự [41] đã để xuất phương pháp kết hợp ba mô hình như RNN, MLP và ADT tạo nên mô hình E3ML. Tác giả huấn luyện và kiểm thử các mô hình trên hai dataset là ISCX2012 và DARPA. So sánh kế quả thu được, dù kết hợp các mô hình lại với nhau nhưng kết quả của E3ML vẫn tương đương so với mô hình học sâu RNN. Cho thấy được khả năng học vượt trội của mô hình học sâu này.

Yin và các cộng sự [42] đã so sánh kết quả huấn luyện trên dataset NSL-KDD ở

mô hình học sâu RNN với các mô hình máy học J48, ANN, Random forest và SVM. Kết quả cho thấy RNN có độ chính xác cao hơn các phương pháp máy học.

Min và các cộng sự [43] đã kết hợp CNN và Random forest để huấn luyện mô hình của họ gọi là TR-IDS trên dataset ISCX2012 và so sánh kết quả với các phương pháp học máy khác như SVM, NN, CNN, RF, kết quả cho thấy mô hình kết hợp của họ cho kết quả với độ chính xác cao (99.13%).

Wu và các cộng sự [44] đã giới thiệu hệ thống phát hiện xâm nhập sử dụng CNN để phân loại đa lớp. Họ huấn luyện mô hình của mình trên dataset NSL-KDD, mà ở đó dataset này được mã hóa sang dạng mảng 11x11. Kết quả sau cùng được so sánh với mô hình RNN thì có độ chính xác cao hơn khi phân loại tấn công DoS, tuy nhiên mô hình được để xuất này lại có độ phức tạp gấp 20 lần so với mô hình RNN.

Yuan và các cộng sự [5] đã kết hợp CNN và RNN và đề xuất phương pháp được đặt trên là DeepDefense. Tác giả sử dụng kỹ thuật sliding-window để xử lý dữ liệu thô từ dataset ISCX2012 để chuyển từ dạng packet-based sang dạng window-based. Tác giả chia dataset thành hai phần lớn và bé. Phần lớn là các gói tin được ghi nhận vào ngày 15 và phần bé là các gói tin được ghi nhận vào ngày 14. Như báo cáo, mô hình 3LSTM đạt độ chính xác cao nhất trong phần dataset lớn với 98.41%, trong khi đó GRU đạt độ chính xác cao nhất với 98.417% trong phần dataset còn lại.

Doriguzzi-Corin và các cộng sự [6] đã sử dụng mô hình CNN với trọng tâm là Conv1D và đặt tên là LUCID. Các tác giả nhắm đến mục tiêu xây dựng mô hình nhỏ gọn để có thể triển khai trên các thiết bị hạn chế phần cứng. Tác giả huấn luyện mô hình của mình trên một loạt các dataset như ISCX2012, CICIDS2017 và CICIDS2018. Ở đây, tác giả chỉ thu thập trên mỗi gói tin 11 đặc trưng và ứng với 100 gói tin trong thời gian 100 giây, tác giả xem đó là 1 luồng. Với việc khai thác khả năng của Conv1D, ReLU, Max pooling, theo như báo cáo, tác giả đã đạt được kết quả rất tốt khi so sánh với các nghiên cứu khác trên các tập dataset tương ứng. Ở đó, với việc kết hợp huấn luyện trên cả ba dataset nêu trên, báo cáo đạt được độ chính xác lên tới 99.50%, với mức độ nhỏ gọn hơn đến 40 lần so với mô hình DeepDefense của [5].

Ram B. Basnet và các cộng sự [45] áp dụng mô hình mạng nơ-ron sâu vào dataset dataset CSE-CIC-IDS2018 để phân loại nhiều loại tấn công. Tác giả sử dụng 2 lớp ẩn dense layer, với lớp đầu tiên có số đơn vị trùng với số đặc trưng được ghi nhận trong dataset và lớp thứ 2 có 128 đơn bị. Thay vì sử dụng phương thức kiểm tra bằng cách tách dataset thành tập huấn luyện và tập kiểm thử, tác giả sử dụng phương pháp n-fold cross-validation. Theo báo cáo, với mô hình này, khi phân loại các tấn công DoS/DDoS cho ra kết quả luôn lớn hơn 99%.

3.3 Phát hiện và phòng thủ DDoS trong SDN

Trong môi trường SDN, có nhiều phương pháp đã được đưa ra để phát hiện và giảm thiểu thiệt hai do DDoS gây ra.

Kim và các cộng sự [47] đã đề xuất phương pháp dự đoán lưu lượng bình thường dựa trên ngưỡng của luồng. Phương pháp sử dụng Cisco's NetFlow Technology, để phát hiện lưu lượng mạng bởi tính năng phát hiện được xây dựng bằng các đặt trưng lưu lượng và ngưỡng được thiết lập. Tuy nhiên, phương pháp này lại phụ thuộc quá nhiều vào kinh nghiệm thực tiễn của người nghiên cứu.

Manso và các cộng sự [48] đã áp dụng hệ thống phát hiện xâm nhập mã nguồn mở SNORT vào mạng SDN để cảnh báo sớm sự xâm nhập và tấn công DDoS từ nguồn. Trong nghiên cứu này, tác giả chủ yếu đưa ra các luật có thể áp dụng vào SNORT để phát hiện sớm việc các thiết bị trong mạng bị lợi dụng thành công cụ tấn công DDoS.

Niyaz và các cộng sự [49] đề xuất phương pháp sử dụng mô hình học sâu với Stacked Autoencoder. Đầu tiên họ xây dựng môi trường mô phỏng mạng SDN để thu thập thông tin các gói tin, thông tin các luồng, tạo nên một dataset riêng gồm các luồng bình thường và luồng tấn công. Sau đó họ phân tích dữ liệu thu được và tiến hành huấn luyện. Đặc biệt, trong dataset của họ có phân loại các loại tán công khác nhau như SYN flood, UDP flood, TCP flood, vv. Kết quả báo cáo là độ chính xác phân loại đa lớp đạt kết quả hầu hết trên 90%, bên cạnh đó độ chính xác của phân loại nhị

phân đạt được kết quả lên đến 99%.

Li và các cộng sự [50] đã áp dụng mô hình học sâu RNN ở nghiên cứu [5] vào mạng SDN. Tác giả đã đề xuất mô hình bắt gói tin và phân loại luồng bằng các mô-dun kết hợp với SDN controller. Sau cùng để ngăn chặn cuộc tấn công, tác giả chặn hai chiều đối với những luồng có IP được phân loại là độc hại.

Chương 4

Mô hình giải pháp của đề tài

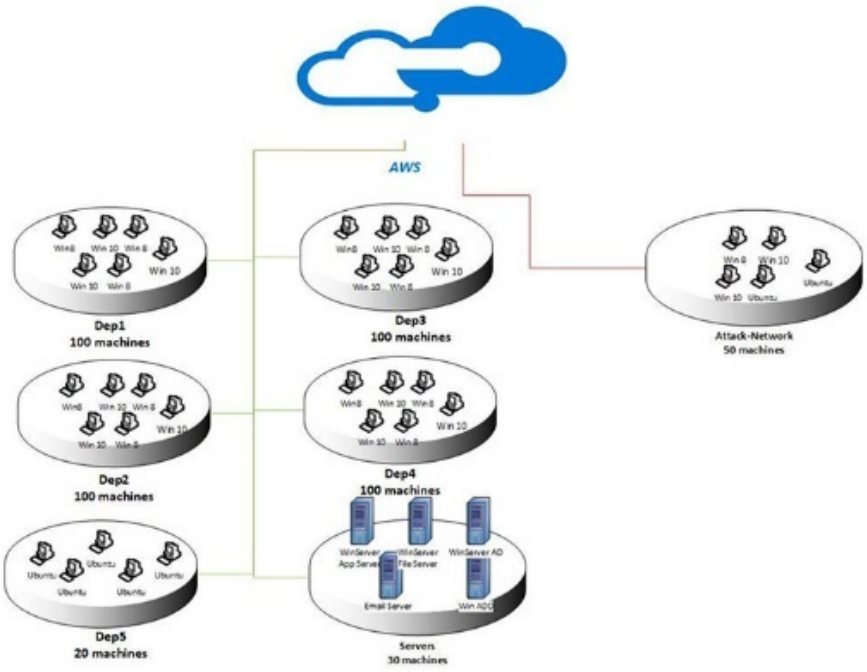
Ở chương trước, chúng tôi đã trình bày những khảo sát về các công trình liên quan tới nghiên cứu của chúng tôi. Từ đó, trong chương này, chúng tôi sẽ đề xuất một mô hình giải pháp cho bài toán phát hiện và phòng thủ trước tấn công DoS/DDoS dựa trên mục tiêu và hướng tiếp cận vấn đề mà chúng tôi đã nêu ra ở Chương Giới thiệu.

4.1 Tập dữ liệu dùng để huấn luyện: CICIDS2018

CSE-CIC-IDS2018 (gọi tắt CICIDS2018) [34] là dataset bao gồm nhiều loại tấn công giúp xây dựng hệ thống phát hiện xâm nhập. Đây là dự án kết hợp giữa Communications Security Establishment (CSE) & the Canadian Institute for Cybersecurity (CIC). Theo báo cáo, để thực hiện tạo dựng dataset này, tác giả đã xây dựng 7 hệ thống mạng với 500 máy tính trên nền tảng AWS. Mô hình mạng này được biểu thị trong hình 4.1.

Các loại tấn công có trong dataset này bao gồm tấn công brute-force, tấn công heartbleed, Botnet, tấn công DoS, tấn công DDoS, tấn công Web, xâm nhập mạng từ bên trong.

Dataset được chia thành hai loại, thứ nhất là tập các tệp PCAP ghi nhận lưu lượng mạng và các mô tả về IP nguồn, IP đích và Protocol của các luồng tấn công và luồng



Hình 4.1: Mô hình mạng trong CSE-CIC-IDS2018

hợp lệ, thứ hai là tập các tệp CSV được sinh ra từ phần mềm CICFlowMeter [35] kèm với nhãn của các luồng đã được đánh dấu sẵn.

Do kích thước của các tệp PCAP rất lớn (lên tới hàng chục GB), nên trong khuôn khổ của khóa luận này, tôi sử dụng các tệp CSV đã được tạo sẵn với kích thước nhỏ hơn rất nhiều.

Dataset được chia ra thành từng ngày, mỗi ngày sẽ có một số loại tấn công được triển khai. Vì vậy, tôi chỉ chọn ra những ngày có tấn công DoS/DDoS được triển khai.

Danh sách các ngày được chọn trong bảng 4.1.

4.1.1 Tiền xử lý và thống kê dữ liệu

Tiền xử lý

Dữ liệu trong dataset không thật sự hoàn hảo, lý do đến từ phần CICFlowMeter vẫn còn một vài lỗi dẫn đến xuất hiện một số bản ghi trong dữ liệu chứa giá trị vô cực (Inf) hoặc không phải số (NaN). Vì vậy, bước cần thiết là loại bỏ các bản ghi lỗi này

Ngày	Mô tả
Thurs-15-02-2018	DoS-GoldenEye DoS-Slowloris
Fri-16-02-2018	DoS-SlowHTTPTest DoS-Hulk
Tues-20-02-2018	DDoS attacks-LOIC-HTTP DDoS-LOIC-UDP
Wed-21-02-2018	DDOS-LOIC-UDP DDOS-HOIC

Bảng 4.1: Danh sách các ngày được chọn trong dataset CICIDS2018

ra khỏi dataset.

Giải thuật tiền xử lý dữ liệu được mô tả trong lưu đồ 4.2.

Thống kê

Sau khi tiền xử lý dữa liệu, số lượng luồng hợp lệ và tấn công còn lại được ghi trong bảng 4.2.

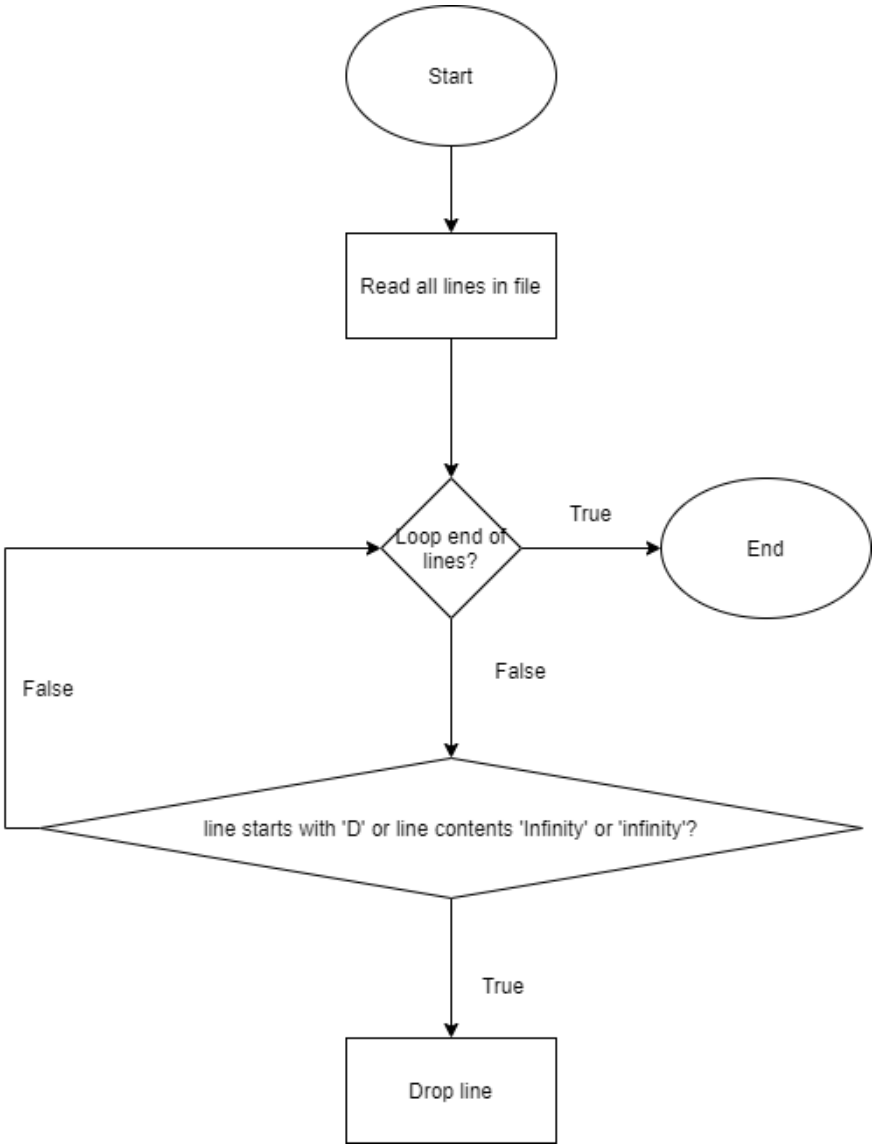
Ngày	Số luồng hợp lệ	Số luồng tấn công
Thurs-15-02-2018	987,980	52,498
Fri-16-02-2018	446,772	601,802
Tues-20-02-2018	7,313,104	576,191
Wed-21-02-2018	360,833	687,742

Bảng 4.2: Bảng thống kê dữ liệu trong CICIDS2018

4.1.2 Xử lý dữ liệu

Phân chia dữ liệu thành các tập con

Vì mỗi một ngày chứa lưu lượng của một loại tấn công khác nhau, nên trước khi gộp dữ liệu các ngày lại với nhau, tôi tiến hành phân chia dữ liệu trên từng ngày thành 2



Hình 4.2: Lưu đồ giải thuật tiền xử lý dataset

tập là tập huấn luyện và tập kiểm tra với tỷ lệ 8:2. Sau đó, tôi ghép các tập con này lại và thu được dataset đặt tên là **IDS-train** tương ứng với tập huấn luyện và **IDS-test** tương ứng với tập kiểm tra được thống kê trong bảng 4.3. Trước khi ghép các dữ liệu lại với nhau, tôi đã loại bỏ trường ”Timestamp”, vì trường này không quan trọng cho việc phân tích và huấn luyện.

Tập	Số luồng hợp lệ	Số luồng tấn công
IDS-train	7,287,007	1,534,586
IDS-test	1,821,682	383,647

Bảng 4.3: Các tập dữ liệu con sau khi phân chia

Phân tích, tối ưu hóa dữ liệu

Sau khi đã có dataset cuối cùng, tôi tiến hành phân tích và tối ưu dữ liệu để phục vụ cho việc huấn luyện.

Phân tích

Tổng số đặc trưng mà CICFlowMeter trích xuất là 84, tuy nhiên khi tác giả công bố dataset đã bỏ đi một vài đặc trưng như Source IP, Source Port, Dest IP, vv, và thêm một đặc trưng được tôi loại bỏ là ”Timestamp” nên tổng số đặc trưng còn lại là 78. Chi tiết các đặc trưng này trong bảng 4.4.

STT	Đặc trưng	Mô tả
1	Dst Port	Cổng của địa chỉ IP đích trong gói tin IP
2	Protocol	Giao thức mạng với 6 là TCP, 17 là UDP
3	Flow Duration	Tổng thời gian của luồng
4	Tot Fwd Pkts	Tổng số gói tin trong luồng theo chiều tới (fwd)
5	Tot Bwd Pkts	Tổng số gói tin trong luồng theo chiều lui (bwd)
6	TotLen Fwd Pkts	Tổng độ dài của các gói tin fwd
7	TotLen Bwd Pkts	Tổng độ dài của các gói tin bwđ
8	Fwd Pkt Len Max	Độ dài max gói tin fwd
9	Fwd Pkt Len Min	Độ dài min gói tin fwd
10	Fwd Pkt Len Mean	Mean độ dài gói tin fwd
11	Fwd Pkt Len Std	Độ lệch chuẩn độ dài gói tin fwd
12	Bwd Pkt Len Max	Độ dài max gói tin bwd
13	Bwd Pkt Len Min	Độ dài min gói tin bwd
14	Bwd Pkt Len Mean	Mean độ dài gói tin bwd

15	Bwd Pkt Len Std	Độ lệch chuẩn độ dài gói tin bwd
16	Flow Byts/s	Tổng số bytes trong luồng / giây
17	Flow Pkts/s	Tổng số gói tinn trong luồng / giây
18	Flow IAT Mean	Mean IAT luồng
19	Flow IAT Std	Độ lệch chuẩn IAT luồng
20	Flow IAT Max	Max IAT luồng
21	Flow IAT Min	Min IAT luồng
22	Fwd IAT Tot	Tổng số IAT fwd
23	Fwd IAT Mean	Mean IAT fwd
24	Fwd IAT Std	Độ lệch chuẩn IAT fwd
25	Fwd IAT Max	Max IAT fwd
26	Fwd IAT Min	Min IAT fwd
27	Bwd IAT Tot	Tổng số IAT bwd
28	Bwd IAT Mean	Mean IAT bwd
29	Bwd IAT Std	Độ lệch chuẩn IAT bwd
30	Bwd IAT Max	Max IAT bwd
31	Bwd IAT Min	Min IAT bwd
32	Fwd PSH Flags	Tổng số cờ PSH fwd
33	Bwd PSH Flags	Tổng số cờ PSH bwd
34	Fwd URG Flags	Tổng số cờ URG fwd
35	Bwd URG Flags	Tổng số cờ URG bwd
36	Fwd Header Len	Tổng độ dài header các gói tin fwd
37	Bwd Header Len	Tổng độ dài header các gói tin bwd
38	Fwd Pkts/s	Tổng số gói tin fwd / giây
39	Bwd Pkts/s	Tổng số gói tin bwd / giây
40	Pkt Len Min	Độ dài min gói tin trong luồng
41	Pkt Len Max	Độ dài max gói tin trong luồng

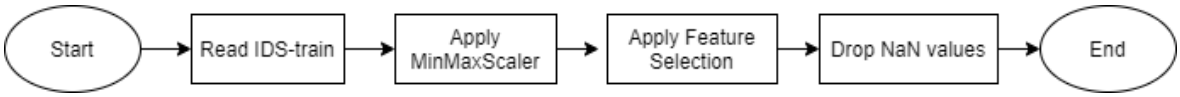
42	Pkt Len Mean	Độ dài mean gói tin trong luồng
43	Pkt Len Std	Độ lệch chuẩn độ dài gói tin trong luồng
44	Pkt Len Var	Phương sai độ dài gói tin trong luồng
45	FIN Flag Cnt	Tổng số cờ FIN
46	SYN Flag Cnt	Tổng số cờ SYN
47	RST Flag Cnt	Tổng số cờ RST
48	PSH Flag Cnt	Tổng số cờ PSH
49	ACK Flag Cnt	Tổng số cờ ACK
50	URG Flag Cnt	Tổng số cờ URG
51	CWE Flag Count	Tổng số cờ CWE
52	ECE Flag Cnt	Tổng số cờ ECE
53	Down/Up Ratio	Lưu lượng fwd / lưu lượng bwd
54	Pkt Size Avg	Kích thước trung bình của gói tin
55	Fwd Seg Size Avg	Kích thước trung bình gói tin fwd
56	Bwd Seg Size Avg	Kích thước trung bình gói tin bwd
57	Fwd Byts/b Avg	Số bytes/bulk trung bình fwd
58	Fwd Pkts/b Avg	Số gói tin/bulk trung bình fwd
59	Fwd Blk Rate Avg	Tỷ lệ bulk fwd trung bình
60	Bwd Byts/b Avg	Số bytes/bulk bwd
61	Bwd Pkts/b Avg	Số gói tin/bulk bwd
62	Bwd Blk Rate Avg	Tỷ lệ bulk bwd trung bình
63	Subflow Fwd Pkts	Số gói tin trong luồng con fwd
64	Subflow Fwd Byts	Số bytes trong luồng con fwd
65	Subflow Bwd Pkts	Số gói tin trong luồng con bwd
66	Subflow Bwd Byts	Số bytes trong luồng con bwd
67	Init Fwd Win Byts	TCP window size của gói tin đầu tiên fwd
68	Init Bwd Win Byts	TCP window size của gói tin đầu tin bwd

69	Fwd Act Data Pkts	Số gói tin thật sự có payload fwd
70	Fwd Seg Size Min	Kích thước header nhỏ nhất fwd
71	Active Mean	Mean thời gian active
72	Active Std	Độ lệch chuẩn thời gian active
73	Active Max	Max thời gian active
74	Active Min	Min thời gian active
75	Idle Mean	Mean thời gian idle
76	Idle Std	Độ lệch chuẩn thời gian idle
77	Idle Max	Max thời gian idle
78	Idle Min	Min thời gian idle

Bảng 4.4: Danh sách các đặc trưng của luồng gói tin được rút trích

Tối ưu hóa

Tôi sử dụng phương pháp Feature Selection với hàm ChiSquare để chọn ra các đặc trưng thật sự cần thiết cho việc huấn luyện. Tôi sử dụng thư viện scikit-learn để thực hiện. Giải thuật được mô tả trong sơ đồ hình 4.3.



Hình 4.3: Lưu đồ tối ưu hóa dataset

Từ đó tôi loại bỏ được 11 đặc trưng gồm Dst Port, Bwd PSH Flags, Fwd URG Flags, Bwd URG Flags, CWE Flag Count, Fwd Byts/b Avg, Fwd Pkts/b Avg, Fwd Blk Rate Avg, Bwd Byts/b Avg, Bwd Pkts/b Avg, Bwd Blk Rate Avg. Từ đó còn lại 67 đặc trưng cho việc huấn luyện.

4.2 Các mô hình huấn luyện

Trước khi áp dụng huấn luyện, tôi chuyển các giá trị âm trong dữ liệu thành 0, sau đó thực hiện normalize dữ liệu bằng hàm normalize trong thư viện Keras. Hàm này sử dụng L2 norm để normalize dữ liệu.

4.2.1 Các chỉ số đánh giá mô hình

Trong bài toán hiện tại tôi đang giải quyết là bài toán phân lớp nhị phân, vì vậy kết quả đầu ra là Positive (Attack) hay Negative (Benign). Ma trận nhầm lẫn (Confusion matrix) của tôi sẽ có dạng như bảng 4.5. Với TP, FP, TN, FN tương ứng với True Positive, False Positive, True Negative, False Negative.

		Predicted	
		Negative	*Positive*
True	*Negative*	# of TN	# of FP
	Positive	# of FN	# of TP

Bảng 4.5: Mẫu ma trận nhầm lẫn

Khi đó, công thức của các chỉ số đánh giá như sau.

Accuracy:

$$Acc = \frac{TP + TN}{Total}$$

Precision (hay Positive predictive value):

$$Pre = \frac{TP}{TP + FP}$$

Recall:

$$Rec = \frac{TP}{TP + FN}$$

F1-score:

$$F1 = 2\frac{Pre \cdot Rec}{Pre + Rec}$$

4.2.2 Mô hình học máy

Trong phần này, tôi giới thiệu chi tiết các mô hình học máy gồm Support Vector Machine, Naïve Bayes, Decision Tree.

Support Vector Machine

Tham khảo từ nghiên cứu [51] của A. Patle và cộng sự, SVM phân loại lớp bằng việc tạo ra một siêu phẳng (hyperplane) N chiều. SVM là một thực thể toán học, một giải thuật để tối đa hóa một hàm toán học cụ thể đối với một tập dữ liệu nhất định. SVM có 4 khái niệm cơ bản:

- Siêu phẳng phân chia (Separating hyperplane)
- Siêu phẳng biên cực đại (Maximum-margin hyperplane)
- Biên mềm (Soft Margin)
- Hàm cốt lõi (Kernel function)

Biên

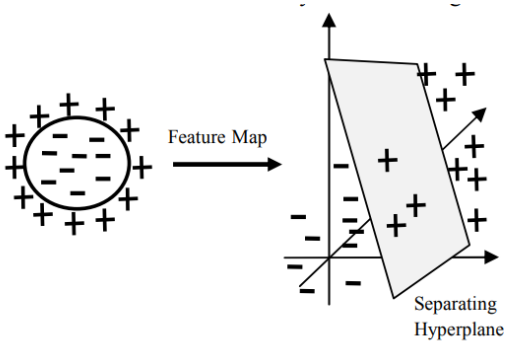
Biên là khoảng cách giữa siêu phẳng đến hai điểm dữ liệu gần nhất tương ứng với các phân lớp.

Siêu phẳng phân chia

Đường thẳng chia không gian thành 2 phần, và ở không gian ba chiều chúng ta cần mặt phẳng để phân chia không gian. Định nghĩa chung cho ranh giới phân chia trong không gian N chiều là siêu phẳng. Vì vậy, siêu phẳng phân chia là ranh giới về cơ bản phân chia các mẫu khác nhau trong tập dữ liệu. Hình 4.4 ví dụ một siêu phẳng phân chia.

Siêu phẳng biên cực đại (siêu phẳng tối ưu)

Siêu phẳng biên cực đại nghĩa là lựa chọn ranh giới ở chính giữa. Nói cách khác, ranh giới được chọn để phân chia hai lớp phải đáp ứng được biên là lớn nhất.



Hình 4.4: Ví dụ siêu phẳng phân chia

Theo [52], cách giải thích của Vapnik và cộng sự trình bày về khái niệm này như sau. Ta có một tập huấn luyện được gán nhãn

$$(y_1, x_1), ..., (y_l, x_l), y_i \in -1, 1 \tag{4.1}$$

được xem là có thể phân chia tuyến tính nếu tồn tại một vec-tơ **w** một đại lượng vô hướng b sao cho các bất bất đẳng thức

$$\begin{aligned} w \cdot x_i + b \geq 1 \qquad & if \qquad y_i = 1, \\ w \cdot x_i + b \leq 1 \qquad & if \qquad y_i = -1 \end{aligned} \tag{4.2}$$

đều thỏa tất cả phần tử trong tập huấn luyện (4.1). Viết lại bất đẳng thức (4.2)

$$y_i(w \cdot x_i + b) \geq 1, \qquad i = 1, ..., l \tag{4.3}$$

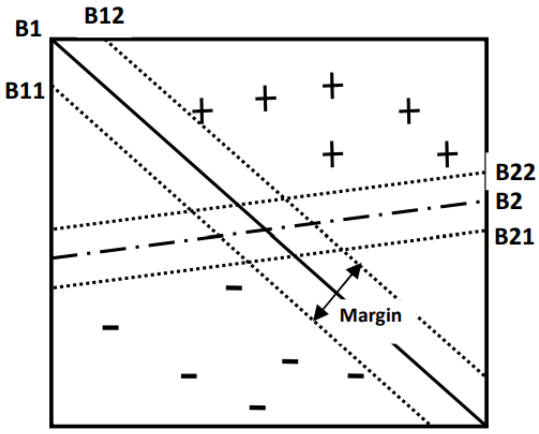
Siêu phẳng tối ưu

$$w_0 \cdot x + b_0 = 0 \tag{4.4}$$

là siêu phẳng duy nhất chia dữ liệu huấn luyện với biên lớn nhất: nó xác định hướng của **w**/|**w**| sao cho khoảng cách khoảng cách hình chiếu của vec-tơ huấn luyện của hai lớp khác nhau là tối đa.

Biên mềm

SVM có thể chịu được lỗi trong dữ liệu bằng cách cho phép một vài dữ liệu bị phân lớp sai. Để làm được vậy, giải thuật SVM chỉnh sửa và thêm vào "biên mềm" (Hình 4.5). Cơ bản, điều này cho phép một số điểm dữ liệu di chuyển qua rìa của siêu phẳng phân chia mà không ảnh hưởng đến kết quả cuối cùng. Trong hình 4.5, siêu phẳng là B1 và B2. B1 tốt hơn B2 vì nó tối đa hóa biên.



Hình 4.5: Ví dụ biên mềm

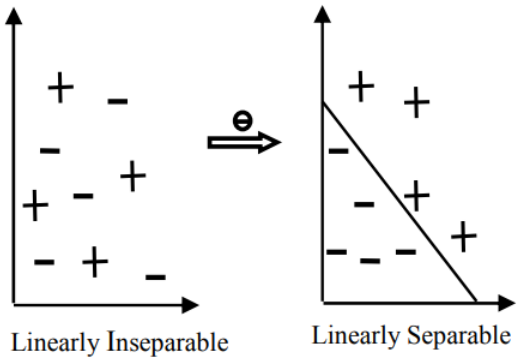
Hàm cốt lõi

Hàm cốt lõi là thủ thuật toán học cho phép SVM phân lớp trong không gian hai chiều của một tập dữ liệu ở không gian một chiều. Nói chung, một hàm cốt lõi ánh xạ dữ liệu từ một không gian chiều thấp đến một không gian có chiều cao hơn. Công thức bên dưới và hình 4.6 thể hiện cách hàm cốt lõi ánh xạ dữ liệu.

$$\langle x_1 \cdot x_2 \rangle \leftarrow K(x_1, x_2) = \langle \Phi(x_1) \cdot \Phi(x_2) \rangle$$

Hàm cốt lõi tuyến tính (Linear Kernel Function)

Hàm cốt lõi tuyến tính được biểu diễn như sau: $K(x, x_j) = x \cdot x^T$



Hình 4.6: Cách hàm cốt lõi ánh xạ dữ liệu

Naïve Bayes

Theo nghiên cứu của Rish và các cộng sự [53], giải thuật phân lớp Naïve được diễn giải như sau.

Cho $X = X_1, ..., X_n$ là một vec-tơ những biến ngẫn nhiên, được gọi là đặc trưng, với mỗi đặc tính lấy giá trị tiền miền D_i của nó. Tập tất cả các vec-tơ đặc trưng được ký hiệu $\Omega = D_1 \times ... \times D_n$. Cho C là một biến ngẫu nhiên không quan sát ký hiệu cho phân lớp của mẫu, C có thể giữa một giá trị trong m giá trị $c \in 0, ..., m-1$.

Một hàm $g : \Omega \rightarrow 0, ..., m-1$, với $g(x) = C$ ký hiệu một *khái niệm* được học.

Một máy phân lớp được định nghĩa bởi một hàm $h : \Omega \rightarrow 0, ..., m-1$ (một hypothesis) gán một lớp cho bất cứ mẫu được cho nào. Một hướng tiếp cận thông dụng là liên kết mỗi lớp i với một hàm phân biệt $f_i(X), i = 0, ..., m-1$, và để cho máy phân lớp chọn phân lớp với hàm phân biệt lớn nhất trên một mẫu được cho: $h(x) = argmax_{i \in 0,...,m-1} f_i(x)$.

Máy phân lớp Bayes $h*(x)$ được sử dụng như những hàm phân biệt xác xuất có điều kiện của các vec-tơ đặc trưng được cho. Áp dụng luật Bayes, hàm phân biệt:

$$f_i^*(x) = P(X = x|C = i)P(C = i)$$

với $P(X = x|C = i)$ được gọi là *phân phối xác xuất có điều kiện*. Vì vậy máy phân

lớp Bayes:

$$h^*(x) = argmax_i P(X = x|C = i)P(C = i)$$

Tuy nhiên, trực tiếp ước lượng P(X=x|C=i) thường khó trong các tập đặc trưng có không gian bậc cao. Vì vậy để đơn giản, ta sử dụng *native Bayes* (NB(x)) với hàm phân biệt được định nghĩa như sau:

$$f_i^{NB} = \Pi_{j=1}^n P(X_j = x_j|C = i)P(C = i)$$

Decision Tree

Theo nghiên cứu của Somvanshi và các cộng sự [54], Decision Tree (DT) được diễn giải như sau.

Để tạo ra các nút, DT sử dụng phương pháp tăng thông tin để xác định thuộc tính phù hợp trong cây. Từ mức thông tin cao nhất chúng ta có thể chọn ra thuộc tính. Có những giải thuật DT khác nhau mà trong đó ID3 (được giới thiệu bởi QUINLAN vào năm 1986) là giải thuật quan trọng dựa trên entropy thông tin để tạo ra giải thuật học có giám sát.

ID3 sử dụng thông tin tăng cường để quyết định phân chia thuộc tính. Đưa ra một tập hợp các kết quả, dữ liệu không chắc chắn biểu diễn trong tập dữ liệu được đo bởi công thức

$$Entropy(S) = -\sum p(x)log2p(x$$

Với S là tập dữ liệu cho mỗi entropy được tính toán, X là tập hợp của các lớp trong tập dữ liệu, P(x) là xác suất của số phần tử trong lớp X đến số phần tử trong tập S. Khi I(S) = 0 thì tập dữ liệu được phân lớp hoàn hảo.

Huấn luyện mô hình trên tập IDS-train

Để đơn giản trong việc triển khai huấn luyện, tôi sử dụng thư viện Scikit-learn.

Với giải thuật SVM tôi huấn luyện với hàm cốt lõi tuyến tính (LSVM) với hàm LinearSVC. Bên cạnh đó tôi sử dụng hàm GaussianNB cho giải thuật Naïve Bayes (NB), hàm DecisionTreeClassifier cho giải thuật Decision Tree (DT) và hàm RandomForestClassifier cho giải thuật Random Forest (RF).

Kiểm thử mô hình trên tập IDS-test

Kiểm thử các mô hình đã huấn luyện ở trên, tôi thu được kết quả được thể hiện trong bảng 4.6. Dựa vào kết quả trong bảng này, tôi nhận thấy giải thuật Decision Tree cho kết quả các chỉ số cao hơn hẳn các giải thuật còn lại, cho thấy tính hiệu quả của cách tiếp cận đơn giản mà giải thuật này áp dụng.

Mô hình	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
LSVM	95.67	88.05	86.91	87.48
NB	67.69	34.92	99.31	51.67
DT	**99.97**	**99.91**	**99.94**	**99.92**
RF	99.83	99.11	99.94	99.52

Bảng 4.6: Kết quả huấn luyện các mô hình học máy trên tập IDS-Test

4.2.3 Mô hình học sâu

Ở các nghiên cứu [6], [5] với việc áp dụng các mô hình CNN và RNN mang lại kết quả tốt. Tuy nhiên, các nghiên cứu đó sắp xếp và xử lý dữ liệu theo thời gian (time series), từ đó việc xử dụng mạng nơ-ron tích chập rất hiệu quả trong việc rút trích đặc trưng. Trong khi đó, trong khóa luận này, đặc trưng của luồng dữ liệu đã được tôi xử lý như trình bày trong mục 4.1.1, vì vậy, tôi sẽ áp dụng mô hình DNN như trong nghiên cứu [45] để huấn luyện, và dùng mô hình này đại diện cho tiếp cận học sâu mà tôi muốn hướng tới.

Trước hết, tôi sẽ trình bày về các mô hình được đề xuất trong nghiên cứu [6] và [5].

Chi tiết mô hình DeepDefense và LUCID

Mô hình DeepDefense [5]

Trong nghiên cứu này, tác giả sử dụng nhiều mô hình RNN khác nhau để đánh giá về độ chính xác trên hai dataset lớn (Data15) và dataset nhỏ (Data14) được thu thập từ dataset ISCX 2012 [32]. Các mô hình gồm LSTM, GRU, CNNLSTM (kết hợp CNN để rút trích đặt trưng trước khi đến lớp LSTM), và 3LSTM. Các mô hình này đều cho kết quả state-of-the-art trên dataset mà tác giả huấn luyện, mà ở đó, mô hình 3LSTM có kiến trúc phức tạp nhất cho kết quả với độ chính xác cao nhất trên dataset lớn (98.41%), trong khi mô hình GRU đơn giản hơn đạt được độ chính xác cao nhất trong dataset nhỏ (98.342%). Kiến trúc các mô hình này được miêu tả như bảng trong hình 4.7.

Table III: Specifications of Different RNN models

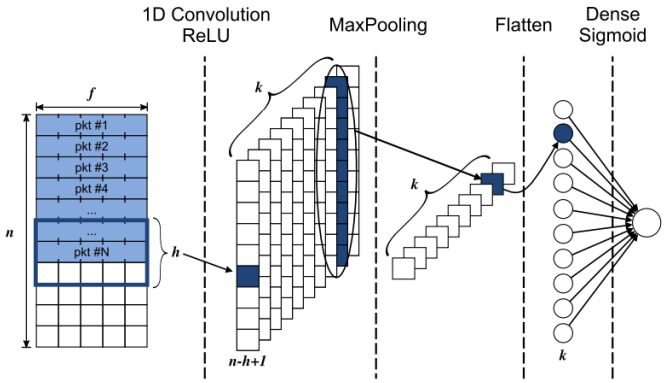
ModelName	LSTM	CNNLSTM	GRU	3LSTM
# LSTM/GRU layer	4	4	4	6
# neuron	64	64	64	64
activation function	tanh	tanh	tanh	tanh
# CNN Layer	/	2	/	/
# neuron	/	128	/	/
activation function	/	relu	/	/
# Fully Connected Layer	2	2	2	2
# neuron	128, 1	128, 1	128, 1	128, 1
activation function	relu, sigmoid	relu, sigmoid	relu, sigmoid	relu, sigmoid

Hình 4.7: Kiến trúc các mô hình của DeepDefense [5]

Mô hình LUCID [6]

Trong nghiên cứu này, tác giả tận dụng sự đơn giản mà hiệu quả của lớp Conv1D và Max Pooling trong mạng CNN. Để xây dựng dataset, tác giả rút trích 11 đặc trưng của các gói tin và sắp xếp theo trình tự thời gian (time-series) với số lượng n cố định. Với các siêu tham số n (số lượng packet tối đa của 1 mẫu), t (thời gian quan sát tối đa 1 mẫu), k (số kernel của Conv1D), h (số filter của Conv1D), m (kích thước của Max Pooling), tác giả có thể tùy chỉnh các thông số này sao cho thu được mô hình có độ

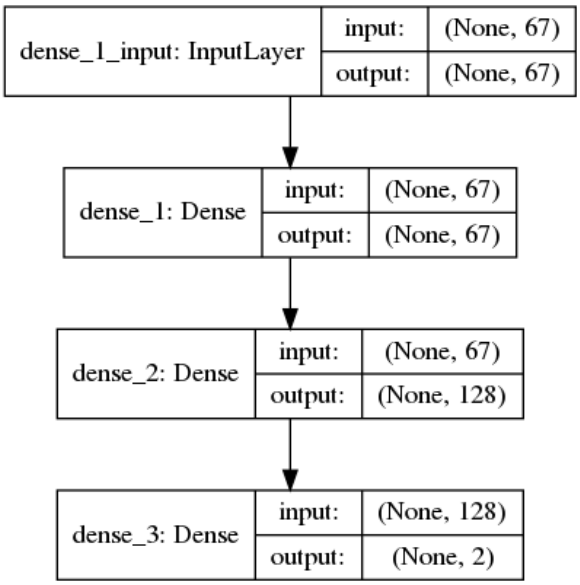
chính xác cao nhất. Cuối cùng với bộ siêu tham số n = 100, t = 100, k = 64, h = 3, m = 98, tác giả thu được độ chính xác mô hình trên dataset ISCX 2012 [32] là 98.88%. So với mô hình DeepDefense ở trên, mô hình LUCID có độ chính xác cao hơn và đơn giản hơn nhiều lần. Kiến trúc của mô hình LUCID được thể hiện trong hình 4.8.



Hình 4.8: Kiến trúc mô hình của LUCID [6]

Chi tiết mô hình DNN

Dựa theo nghiên cứu của Ram B. Basnet và các cộng sự [45], mô hình tôi áp dụng được thể hiện trong hình 4.9.



Hình 4.9: Mô hình học sâu DNN

Trong đó,

– Lớp Input: đầu vào có dạng (67,1) tương ứng với dạng của một bản ghi trong dataset có 67 đặc trưng của một luồng gói tin.

– Lớp Dense_1: lớp này có 67 đơn vị giúp rút trích đặc tính của từng đặc trưng trong dữ liệu. Hàm kích hoạt được dùng là ReLU.

– Lớp Dense_2: lớp này có 128 đơn vị. Hàm kích hoạt được dùng là ReLU.

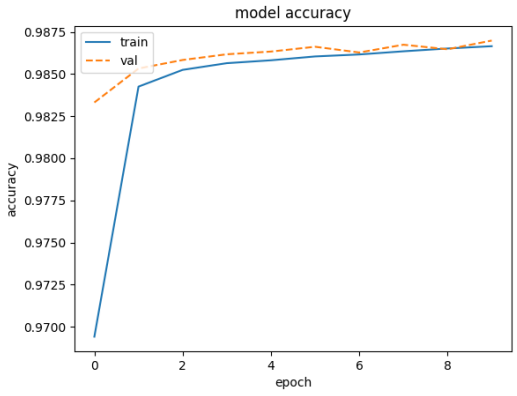
– Lớp Fully conntected (dense 3): lớp này có 2 đơn vị để để phân lớp Benign hay Attack. Hàm kích hoạt được dùng là Softmax.

Huấn luyện mô hình DNN trên tập IDS-train

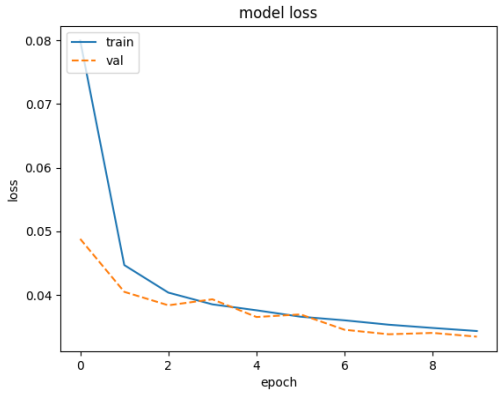
Để huấn luyện mô hình, tôi sử dụng hàm tối ưu là Adam [55], hàm mất mát là CategoricalCrossEntropy.

Trước khi huấn luyện, tôi tách tập *IDS-train* thành hai phần là phần huấn luyện và phần xác thực với tỷ lệ 8:2. Sau khi huấn luyện với 10 epoch, tôi thu được kết quả với độ chính xác là 99.59% trên tập xác thực.

Biểu đồ trong hình 4.10 thể hiện độ chính xác và biểu đồ trong hình 4.11 thể hiện sự mất mát trong quá trình huấn luyện.



Hình 4.10: Độ chính xác mô hình DNN trong quá trình huấn luyện



Hình 4.11: Độ mất mát mô hình DNN trong quá trình huấn luyện

Kiểm thử mô hình DNN trên tập IDS-test

Kiểm thử mô hình đã huấn luyện ở trên, tôi thu được kết quả đạt độ chính xác 99.22%. Bảng 4.7 biểu thị ma trận nhầm lẫn.

		Predicted	
		Negative	*Positive*
True	*Negative*	1,813,764	7,988
	Positive	9,291	374,356

Bảng 4.7: Ma trận nhầm lẫn mô hình DNN trên tập IDS-Test

Từ ma trận nhầm lẫn, tôi có được các chỉ số đánh giá mô hình của mình trong bảng 4.8.

Mô hình	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-score (%)**
DNN	99.22	97.91	97.58	97.74

Bảng 4.8: Kết quả mô hình DNN trên tập IDS-test

So sánh kết quả mô hình DNN với DeepDefense và LUCID

Vì các mô hình DeepDefense [5] và LUCID [6] chưa được tôi cài đặt và kiểm thử trên một tập dữ liệu, vì vậy tôi sẽ so sánh kết quả của mô hình DNN của tôi với các kết quả được công bố trong các nghiên cứu trên. Kết quả so sánh được biểu thị trong bảng 4.9.

Mô hình	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-score (%)**
DNN	**99.22**	97.91	97.58	97.74
LUCID	98.88	**98.27**	**99.52**	**98.89**
DeepDefense	98.41	98.34	98.48	98.41

Bảng 4.9: So sánh kết quả của mô hình DNN với DeepDefense và LUCID

Từ số liệu trong bảng 4.9, tôi nhận thấy, kết quả của các mô hình trên các tập kiểm thử của chính họ đều đạt được kết quả tốt và đạt được state-of-the-art.

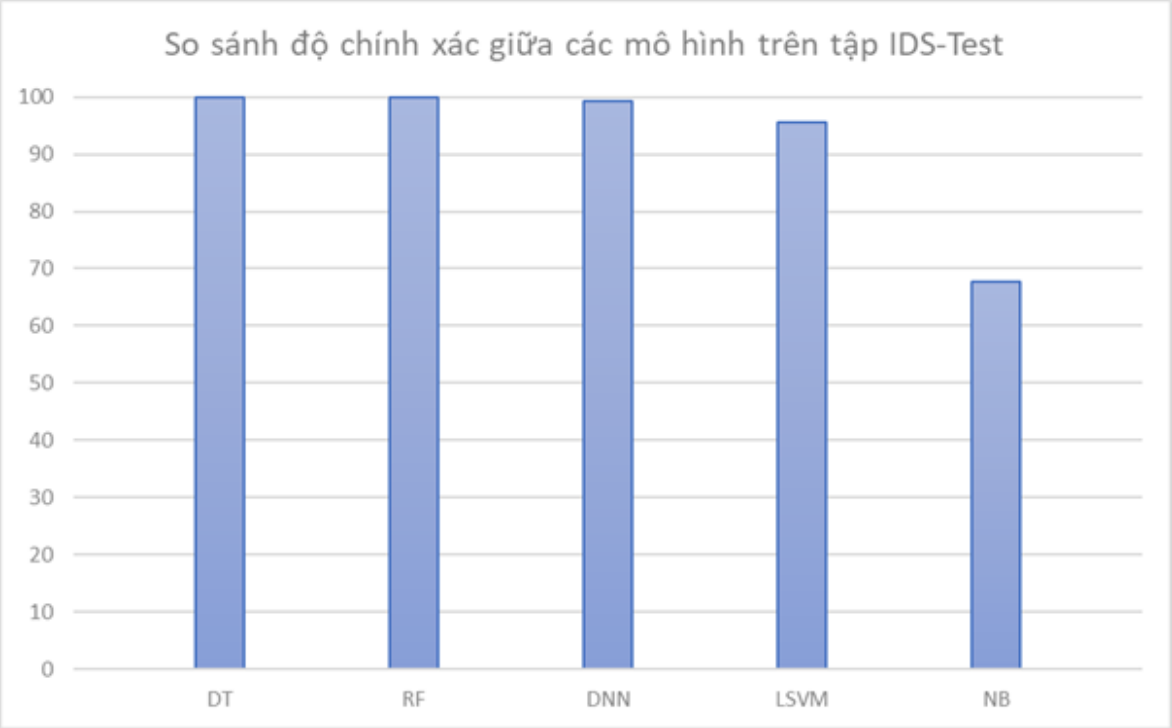
4.2.4 So sánh đánh giá các mô hình trên tập IDS-test

Trong phần này, tôi sẽ so sánh các chỉ số đánh giá và thời gian thực thi giữa các mô hình học máy với mô hình học sâu DNN.

So sánh đánh giá độ chính xác

Mô hình	Accuracy (%)
DT	**99.97**
RF	**99.83**
DNN	99.22
LSVM	95.67
NB	67.69

Bảng 4.10: So sánh độ chính xác của các mô hình trên tập IDS-Test



Hình 4.12: So sánh độ chính xác các mô hình trên tập IDS-test

Từ bảng 4.10 và biểu đồ hình 4.12, có thể thấy, hai mô hình học máy có tiếp cận đơn giản là Decision Tree và Random Forest có độ chính xác cao hơn mô hình DNN. Điều này cho thấy sự đơn giản mà hiệu quả của các mô hình học máy trong việc nhận

diện tấn công DoS/DDoS so với mô hình học sâu DNN có kiến trúc phức tạp và đòi hỏi nhiều tài nguyên tính toán hơn.

So sánh và đánh giá thời gian thực thi

Trong phần này tôi sẽ đánh giá thời gian thực thi giữa hai mô hình Decision Tree và DNN. Để đánh giá và so sánh thời gian thực thi của hai mô hình, tôi sử dụng phần cứng với CPU Intel Core I7-8700 và bộ nhớ RAM 32GB. Tổng số mẫu trong IDS-Test là 2,205,399 mẫu. Kết quả được biểu thị trong bảng 4.11.

Mô hình	Thời gian thực thi (giây)	Thời gian thực thi trung bình 1 mẫu (giây)
DT	**0.4**	$2 \cdot 10^{-7}$
DNN	8.5	$4 \cdot 10^{-6}$

Bảng 4.11: So sánh thời gian thực thi giữa DT và DNN

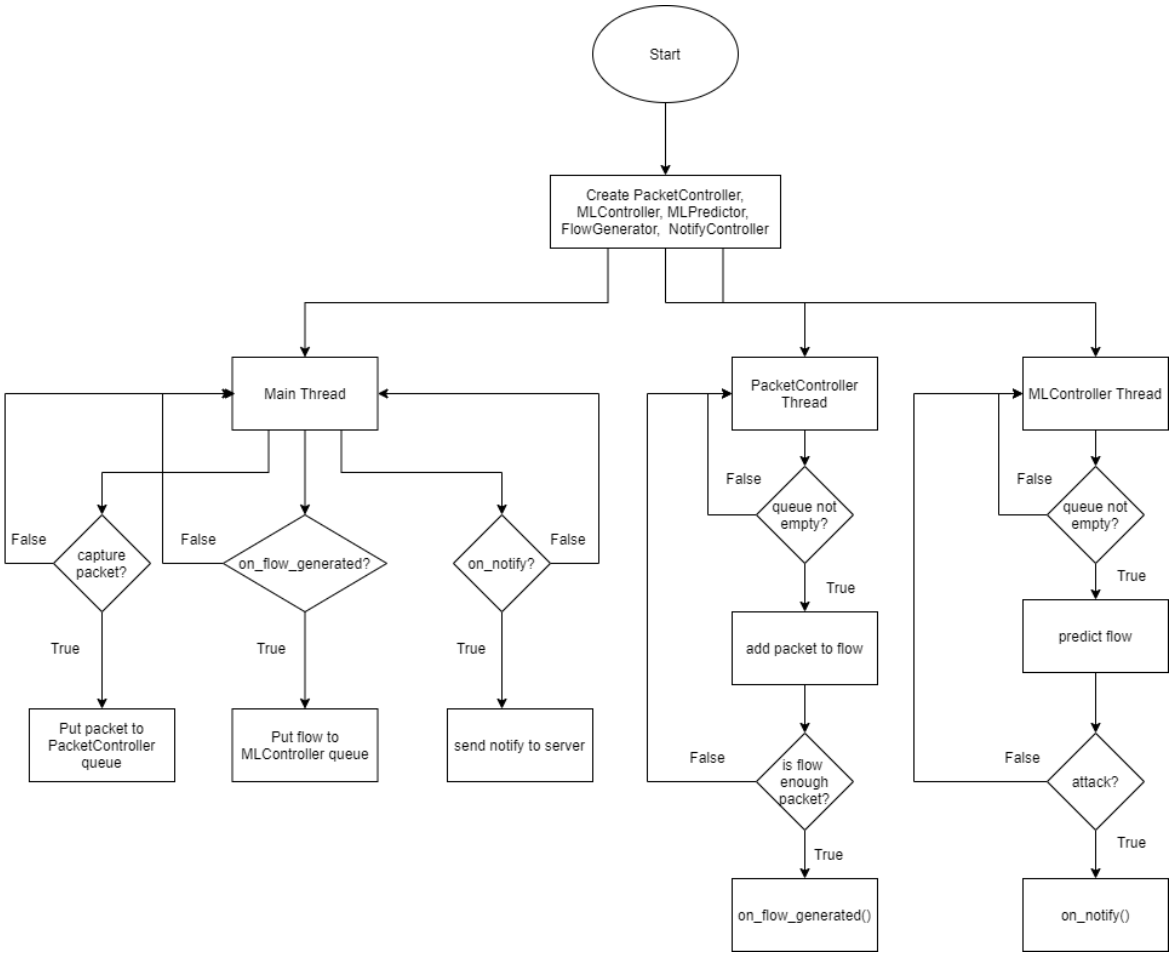
Từ kết quả trong bảng 4.11, tôi nhận thấy thời gian thực thi mô hình Decision Tree nhanh hơn đáng kể so với mô hình DNN (gấp 22 lần). Từ đó cho thấy, Decision Tree không những vượt trội hơn DNN về độ chính xác mà còn vượt trội hơn cả về thời gian thực thi.

4.3 Phát triển phần mềm phát hiện tấn công DoS/DDoS – IDS-DDoS

Sau khi huấn luyên các mô hình học máy và học sâu, tôi tiến hành viết phần mềm phát hiện tấn công mạng trong thời gian thực.

4.3.1 Kiến trúc của phần mềm

Ý tưởng của phần mềm là sẽ lắng nghe các gói tin trên một giao diện mạng, khi phát hiện một luồng tấn công, thông báo sẽ được gửi lên một máy chủ thông qua unix socket. Lưu đồ của phần mềm được thể hiện trong hình 4.13.



Hình 4.13: Lưu đồ phần mềm IDS-DDoS

Phần mềm được thiết kế chạy đa tiểu trình (multi-thread). Trong đó có 3 tiểu trình là MainThread, PacketControllerThread và MLControllerThread.

MainThread

Trong tiểu trình này, các gói tin sẽ được thu thập trên một giao diện mạng (một card mạng), cứ mỗi lần bắt được một gói tin, tiểu trình này sẽ đẩy thông tin gói tin đó vào trong hàng đợi (queue) của PacketControllerThread để chờ xử lý tạo ra các luồng gói tin (flow). Bên cạnh đó, tiểu trình này có hai hàm callback là **on_flow_generated** và **on_notify**. Mỗi khi PacketControllerThread tạo ra 1 flow mới, nó sẽ gọi hàm **on_flow_generated** để thông báo, khi nhận được thông báo MainThread sẽ đẩy flow này vào trong queue của MLControllerThread để chờ xử lý. Mỗi khi MLControllerThread xử lý xong, nó sẽ gọi hàm **on_notify** để thông báo cho MainThread.

Hàm **on_notify** ở MainThread có chức năng là sẽ đẩy thông điệp đến một socket server thông qua unix socket.

PacketControllerThread

Tiểu trình này sẽ có một vòng lặp vô tận kiểm tra xem queue có phần tử không, nếu có nó sẽ tiến hành lấy gói tin trong queue ra, biến đổi thành đối tượng lớp BasicPacketInfo, đối tượng này sẽ được xử lý trong lớp FlowGenerator. FlowGenerator sẽ liên tục đẩy các BasicPacketInfo vào đối tượng lớp BasicFlow cho đến khi nhận được gói tin kết thúc hoặc timeout. Cuối cùng nó sẽ đẩy đối tượng lớp BasicFlow vào hàm **on_flow_generated**.

MLControllerThread

Tương tự PacketControllerThread, tiểu trình này cũng có một vòng lặp để kiểm tra queue, nếu có phần tử (flow) nó sẽ tiến hành phân loại flow này, nếu là flow độc hại (tấn công DoS/DDoS) nó sẽ gọi hàm **on_notify**.

4.3.2 Rút trích các đặc trưng

Để rút trích đặc trưng của một luồng gói tin, tôi sử dụng 3 lớp là BasicPacketInfo, BasicFlow và FlowGenerator được mô tả như bên dưới. Toàn bộ giải thuật của các lớp này được tôi tham khảo mã nguồn phần mềm CICFlowMeter [35]. Lý do là vì dataset mà tôi sử dụng để huấn luyện được tạo ra từ phần mềm này.

Lớp BasicPacketInfo

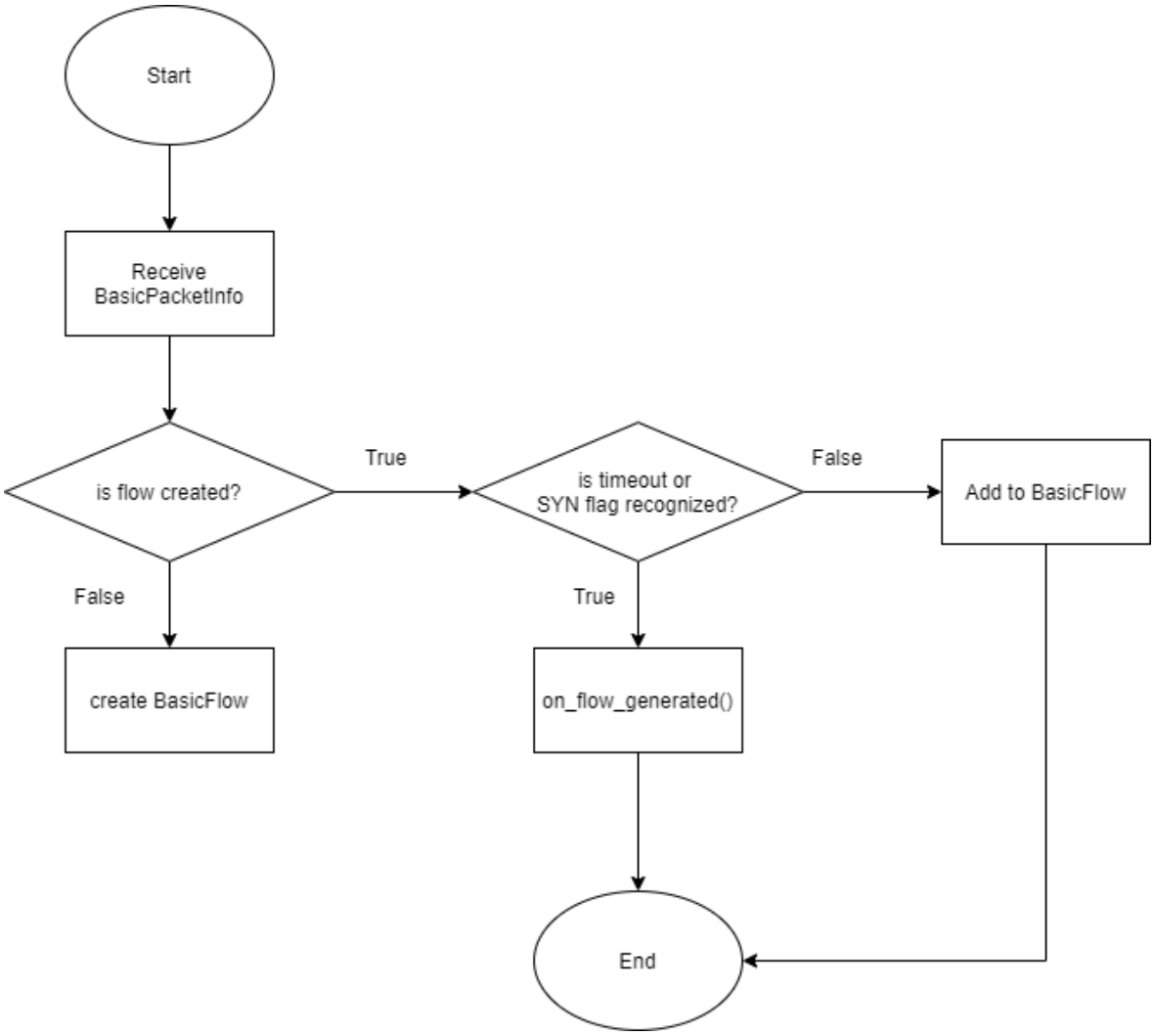
Gói tin mạng có rất nhiều thành phần, vì vậy cần phải phân tích và lấy ra các thông tin thiết yếu. Các thông tin này sẽ được đưa vào lớp BasicPacketInfo. Các thông tin đó gồm: địa chỉ IP nguồn, địa chỉ IP đích, cổng nguồn, cổng đích, giao thức, timestamp, số lượng bytes trong payload, các flag FIN, PSH, URG, ECE, SYN, ACK, CWR, RST, TCP window, số byte trong header.

Lớp BasicFlow

Từ thông tin của các BasicPacketInfo, lớp này sẽ tính toán các đặc trưng. Các đặc trưng này được liệt kê trong bảng 4.4. Các đặc trưng chủ yếu là các con số thống kê max, min, trung bình, độ lệch chuẩn và phương sai.

Lớp FlowGenerator

Lưu đồ của lớp này được thể hiện trong hình 4.14.



Hình 4.14: Lưu đồ lớp FlowGenerator

4.3.3 Sử dụng phần mềm

Phần mềm có giao diện console. Để chạy được cần cung cấp các thông tin gồm giao diện mạng, đường dẫn đến unixsocket server, tên mô hình học máy/học sâu, các đường

dẫn đến tệp lưu trữ mô hình.

python3 ids-ddos.py <network interface> <unixsocket path> <model name> <model path>

Ví dụ, lắng nghe trên eth0 và gửi qua socket /tmp/ids-ddos.

DT: *python3 ids-ddos.py eth0 /tmp/ids-ddos svm ./model/dt_model.pkl*

DNN: *python3 ids-ddos.py eth0 /tmp/ids-ddos svm ./model/dnn-model.json,./model/dnn-weights.hdf5*

4.4 Xây dựng SDN controller bằng RYU

SDN controller của tôi được thiết kế để làm các nhiệm vụ:

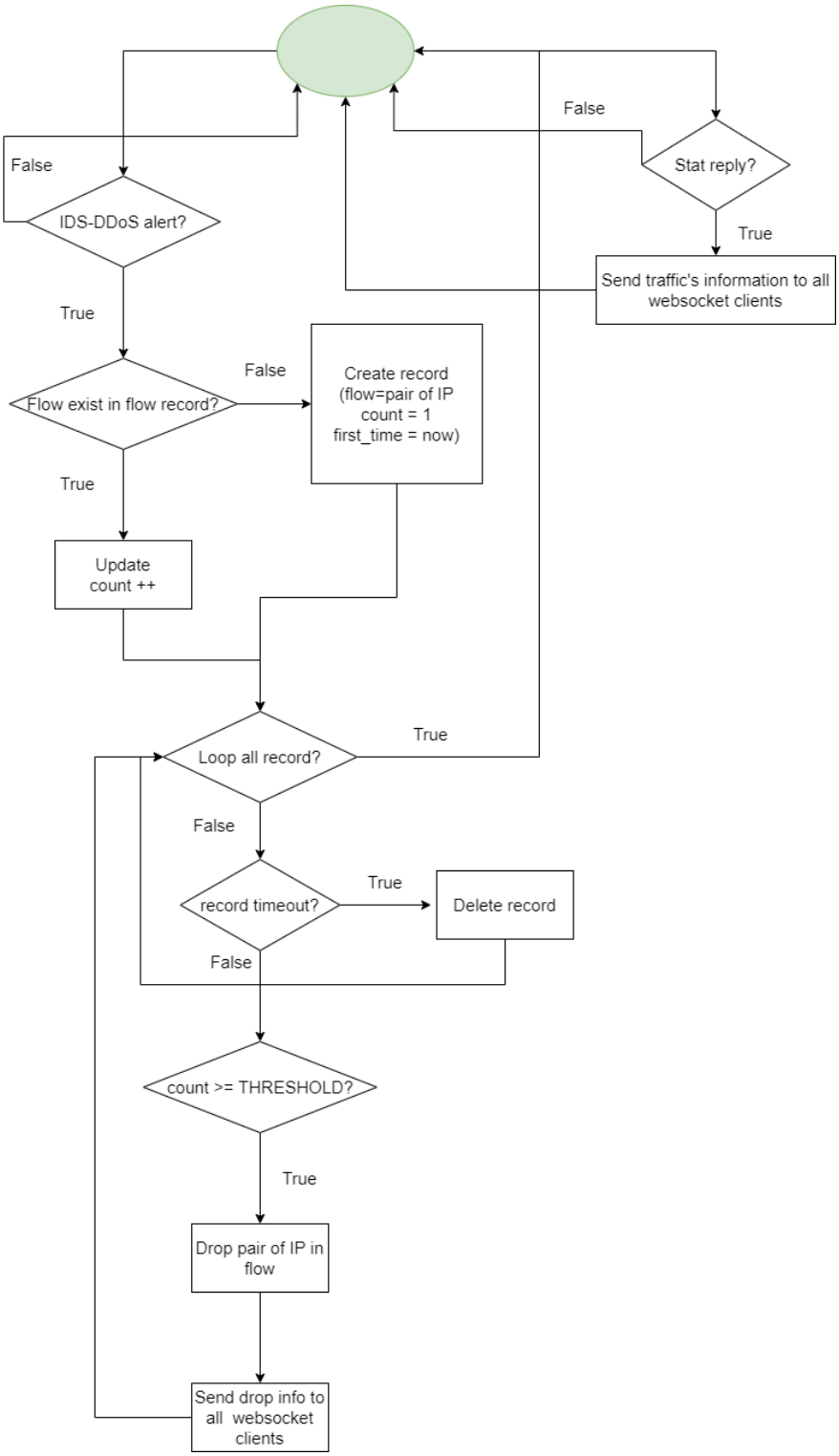
- Là một switch controller cho các host trong mạng SDN.
- Là một traffic monitor các gói tin đi và đến trong mạng, tín hiệu và dữ liệu này được gửi từ Open vSwitch.
- Là một unixsocket server nhận tín hiệu từ phần mềm IDS-DDoS.

Lưu đồ trong hình 4.15 thể hiện giải thuật của SDN controller, tuy nhiên tôi không thể hiện phần switch controller, vì đây là phần đơn giản được sao chép từ ví dụ của nhà phát triển.

SDN controller sẽ tạo ra 2 server

- Server unixsocket để chờ tín hiệu từ IDS-DDoS.
- Server websocket để gửi tín hiệu đến phần mềm trực quan ở máy khách.

Khi nhận được luồng có định dạng là ”IP1-IP2”từ IDS-DDoS, trong đó IP1 và IP2 là IP của 2 máy cần phải chặn không cho giao tiếp, controller sẽ tiến hành kiểm tra xem cặp IP này đã được ghi nhận trong bản ghi chưa, nếu chưa thì tạo, nếu có rồi thì cập nhật trạng thái số lượng. Sau đó, controller sẽ tiến hành quét tất cả các luồng trong bản ghi, nếu luồng nào đã timeout (thời gian hiện tại trừ first_time lớn hơn 1 hằng số



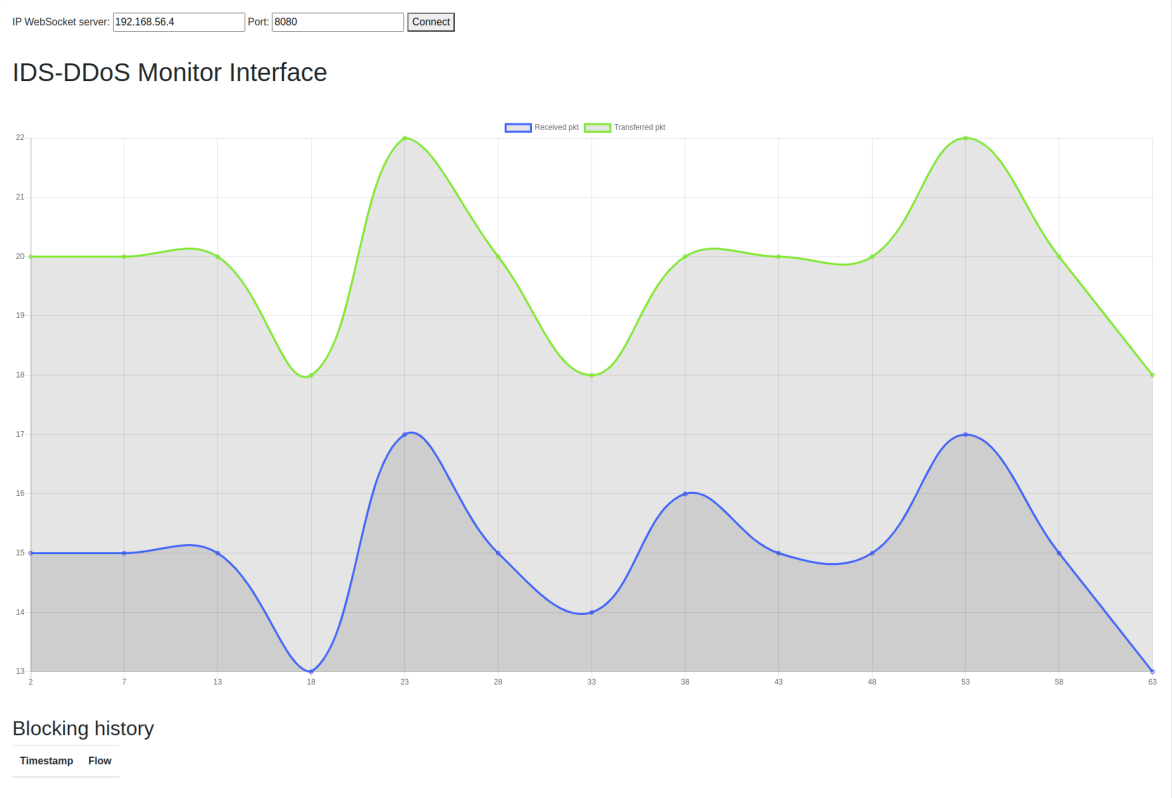
Hình 4.15: Lưu đồ của SDN controller

giây định trước) thì xóa luồng đó, ngược lại sẽ đi kiểm tra xem luồng tấn công này có được ghi nhận lớn hơn 1 ngưỡng hằng số định trước hay không, nếu có thì chặn luồng

này và gửi thông tin này đến tất cả các client đang kết nối vào websocket server, nếu không thì quay lại tiếp tục lắng nghe tín hiệu từ IDS-DDoS.

Tương tự, khi nhận được tín hiệu thống kê lưu lượng từ Open vSwitch, controller sẽ tiến hành broadcast đến tất cả client đang kết nối đến websocket server thông tin này.

Bên cạnh đó, tôi thiết kế một trang web kết nối đến websocket server để trực quan hóa các thông tin chặn luồng, lưu lượng mạng như trình bày ở trên. Hình 4.16 thể hiện giao diện của trang web này.



Hình 4.16: Giao diện web client thống kê lưu lượng

Trong đó,

– Trục tung là số lượng gói tin.

– Trục hoành là thời điểm kể từ khi truy cập vào công cụ theo dõi.

– Đường màu xanh lá biểu diễn số lượng gói tin được gửi từ trong mạng SDN ra ngoài.

– Đường màu xanh dương biểu diễn số lượng gói tin mạng SDN nhận từ bên ngoài.

– Bảng ”Blocking history”ghi lại các cặp IP đã chặn.

Chương 5

Mô hình mạng thử nghiệm

Để tạo mô phỏng các máy tính trong mạng tôi sử dụng 3 máy ảo với phần mềm Virtualbox.

Mạng mô phỏng được biểu diễn như trong hình 5.1.

Các máy ảo trong mạng sẽ gồm Inner Network Machine, Outer Attacker và Outter Webserver. Các máy này được nối với nhau qua 1 mạng ảo Host-only (D) của Virtualbox với địa chỉ đường mạng là 192.168.56.0. Thông tin về các máy này được ghi trong bảng 5.1.

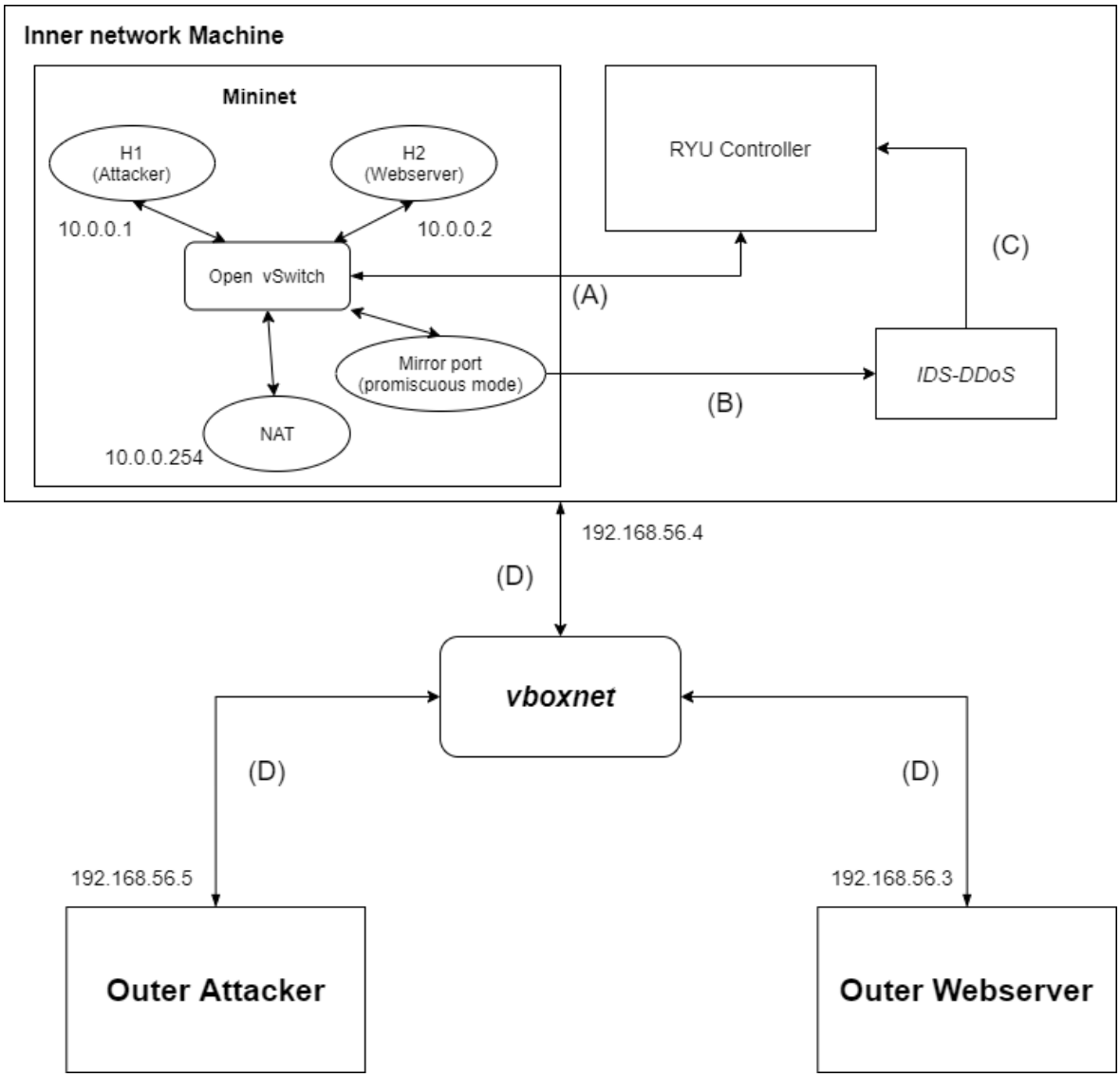
Inner network Machine

– Mạng SDN được kết nối 2 chiều với RYU Controller thông qua TCP localhost (A).

– Mirror port là 1 card mạng ảo ở chế độ promiscious dùng để chuyển tất cả traffic từ H1 và H2 vào đây. Card mạng này được phần mềm IDS-DDoS lắng nghe để bắt các gói tin và phân loại luồng (B).

– RYU Controller và phần mềm IDS-DDoS giao tiếp với nhau thông qua unix-socket (C).

Mạng SDN



Hình 5.1: Mạng mô phỏng

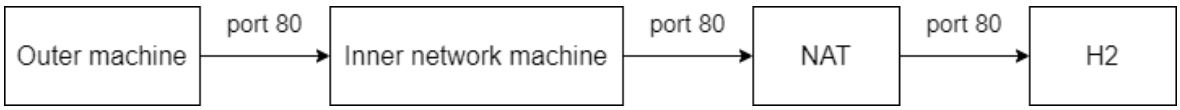
– H1, H2 là các host trong mạng SDN. Trong đó H1 dùng để chạy phần mềm tấn công DDoS Hulk, H1 dùng để làm 1 webserver.

– Để H1, H2 có thể giao tiếp với bên ngoài, Open vSwitch được trang bị thêm 1 thiết bị NAT.

– Ngược lại, để máy bên ngoài có thể giao tiếp với Webserver H2, các lệnh ”iptables nat” được dùng sao cho tất cả gói tin đi tới port 80 của máy Inner network machine sẽ được chuyển vào port 80 của thiết bị NAT trong SDN, từ đó, tất cả các gói tin đi vào port 80 sẽ được chuyển vào port 80 của máy H2. Mô hình như hình 5.2.

Máy	Vai trò	Địa chỉ IP	Cấu hình
Inner Network Machine	Chứa mạng ảo SDN được tạo bằng Mininet, RYU SDN Controller và phần mềm IDS-DDoS.	192.168.56.4	HĐH Ubuntu 18.04 64bit CPU 6 threads RAM 4GB
Outer Webserver	Chứa một trang web tĩnh tạo với SimpleHTTP python	192.168.56.3	HĐH Ubuntu server 18.04 32bit CPU 1 thread RAM 1GB
Outer Attacker	Dùng để chạy công cụ tấn công DDoS Hoic	192.168.56.5	HĐH Windows 7 32bit CPU 2 threads RAM 2GB

Bảng 5.1: Mô tả các máy tính trong mạng



Hình 5.2: Mô hình NAT port

Chương 6

Thử nghiệm và phân tích kết quả

Trong chương này, tôi sẽ xây dựng hai kịch bản để kiểm thử hoạt động của phần mềm IDS-DDoS trong mạng SDN. Hai kịch bản này là:

- Mạng SDN bị tấn công từ bên ngoài. Kịch bản này tương tự như các kịch bản tấn công mạng thường gặp. Trong trường hợp này, tôi sẽ sử dụng IDS-DDoS để cố gắng bảo vệ và giảm thiểu thiệt hại cho hệ thống mạng cũng như máy chủ web bên trong.
- Máy con trong mạng SDN bị lợi dụng tấn công ra bên ngoài. Kịch bản này tương tự như kịch bản các thiết bị IoT bị xâm nhập và trở thành botnet. Trong trường hợp này, tôi sẽ cố gắng ngăn chặn các thiết bị bên trong mạng SDN thực hiện tấn công đến các máy chủ bên ngoài.

Với phần mềm IDS-DDoS, tôi sử dụng mô hình LSVM để nhận diện các luồng tấn công. Mặc dù kết quả huấn luyện cho thấy LSVM có độ chính xác thấp hơn so với Decision Tree, Random Forest hay DNN (mục 4.2.4), tuy nhiên, khi áp dụng thực tế, LSVM lại nhận diện luồng tấn công tốt hơn. Kết luận này được tôi rút ra từ việc quan

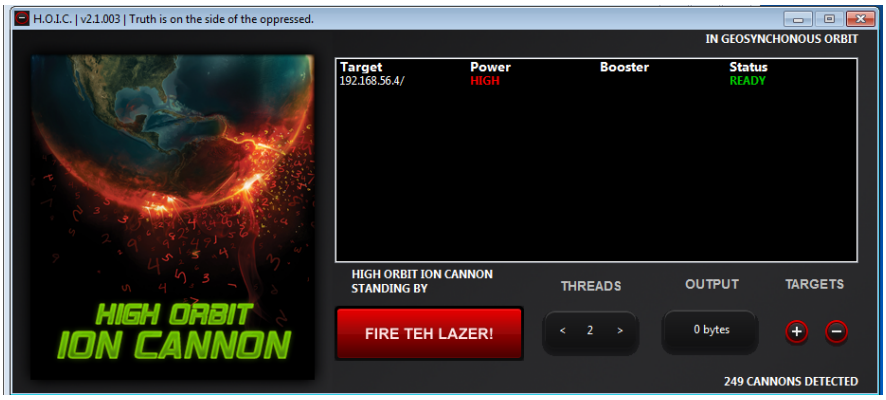
sát quá trình hoạt động của các mô hình trong môi trường thực nghiệm, phần này sẽ được tôi trình bày trong phần 6.3.

Các biểu đồ trong mục này được chụp từ công cụ theo dõi lưu lượng gói tin trong mạng được thiết kế trong mục 4.4.

6.1 Mạng SDN bị tấn công từ bên ngoài

Trong kịch bản này, Outer Attacker sẽ sử dụng công cụ HOIC để tấn công vào Web-server H2.

Giao diện công cụ HOIC trong hình 6.1. Vì đây là công cụ không dành cho học thuật và có thể bị lợi dụng để sử dụng với mục đích xấu nên tôi không chỉ ra nơi tham khảo hay cách tải xuống công cụ này.



Hình 6.1: Giao diện công cụ HOIC

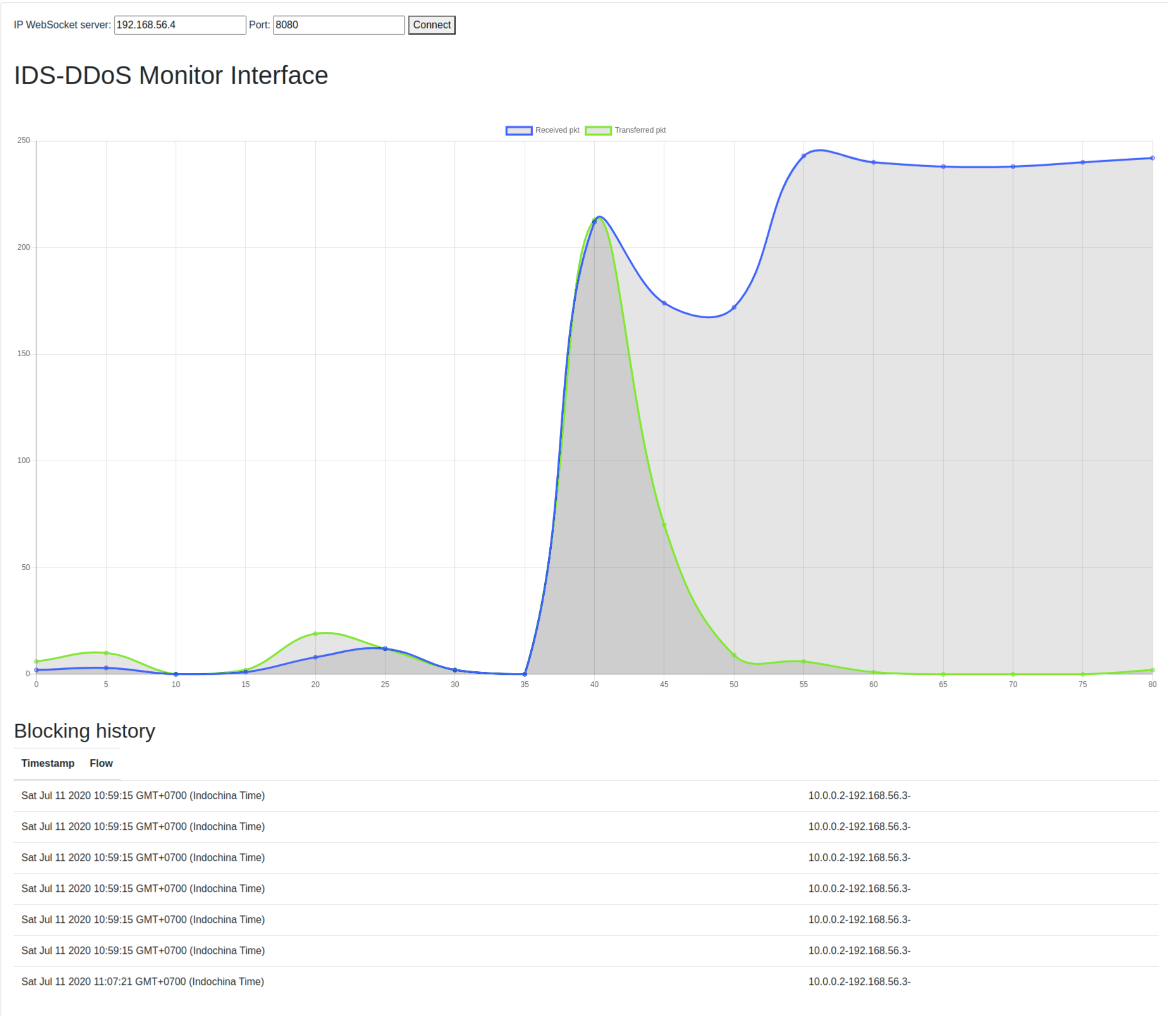
Quá trình tấn công được biểu thị trong các biểu đồ dưới đây.

Biểu đồ trong hình 6.2 mô tả trạng thái mạng trước và trong lúc bị tấn công.

Lúc đầu, nhìn theo trục gói tin, ta thấy được số lượng gói tin đến và gói tin đi vẫn còn thấp. Đường màu xanh lá biểu diễn số lượng gói tin đi, tức là các gói tin được phản hồi từ máy chủ web H2. Đường màu xanh dương biểu thị gói tin đến, tức là các gói tin được gửi tử Outer attacker.

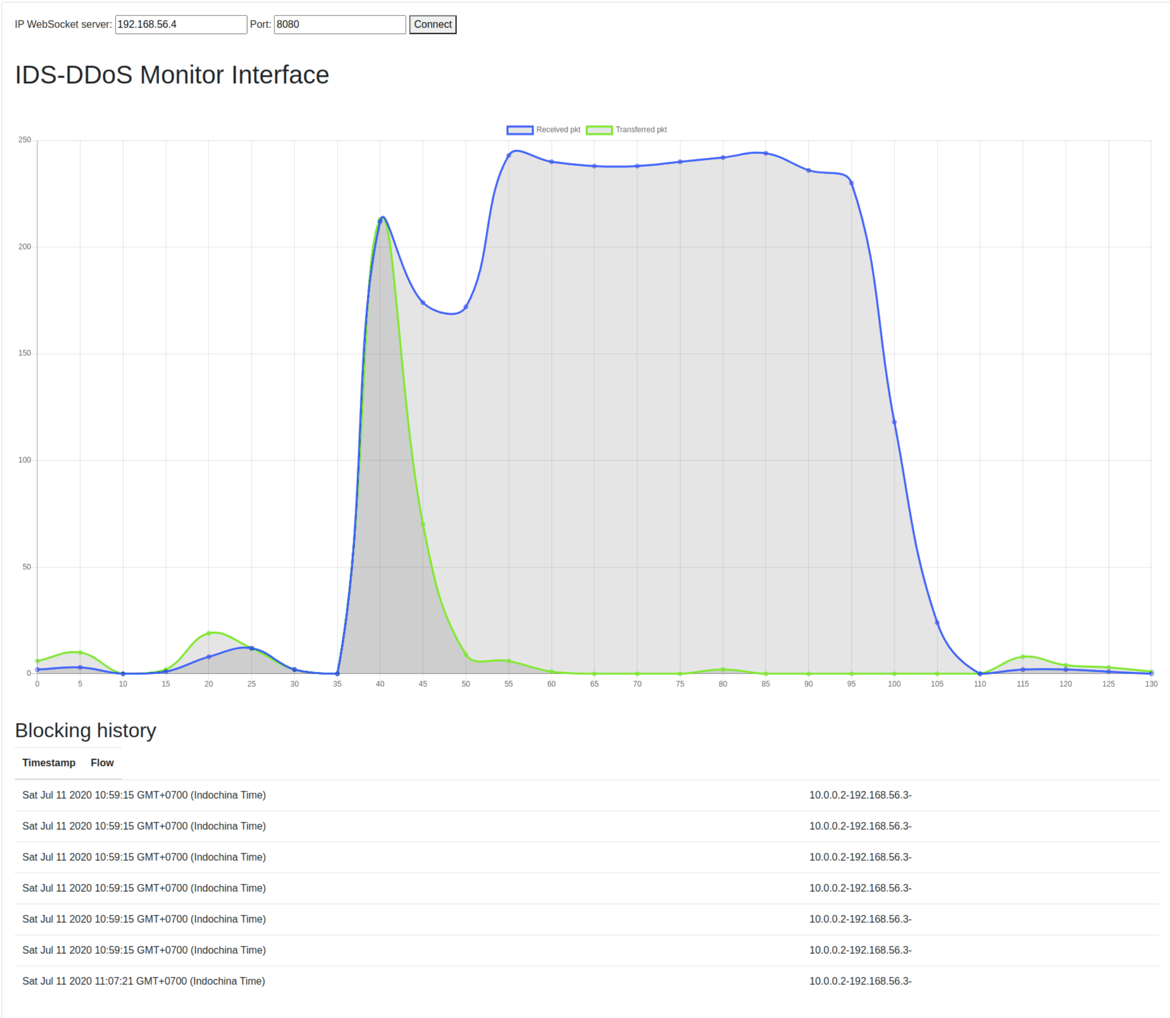
Tuy nhiên, khi tấn công diễn ra, ta thấy được số lượng gói tin đến và đi tăng lên

đột ngột trong một khoảng thời gian. Sau đó, số lượng gói tin đi (đường xanh lá) giảm xuống 0, trong khi số lượng gói tin đến (đường xanh dương) vẫn tăng. Nguyên nhân là do khi IDS-DDoS phát hiện tấn công và thông báo cho SDN controller chặn luồng IP, lúc này, máy chủ H2 bên trong mạng không thể nhận thêm yêu cầu từ attacker nên không thể phản hồi, trong khi đó attacker vẫn tiếp tục tấn công làm cho số lượng gói tin đến tăng không ngừng.



Hình 6.2: Kịch bản 1: Trạng thái mạng trước và trong khi tấn công

Sau khi kết thúc tấn công (hình 6.3), số lượng gói tin đến và đi đều giảm xuống 0.



Hình 6.3: Kịch bản 1: Trạng thái mạng sau khi tấn công

6.2 Máy con trong mạng SDN bị lợi dụng tấn công ra bên ngoài

Trong kịch bản này attacker H1 sẽ tấn công vào Outer Webserver bằng công cụ Hulk.

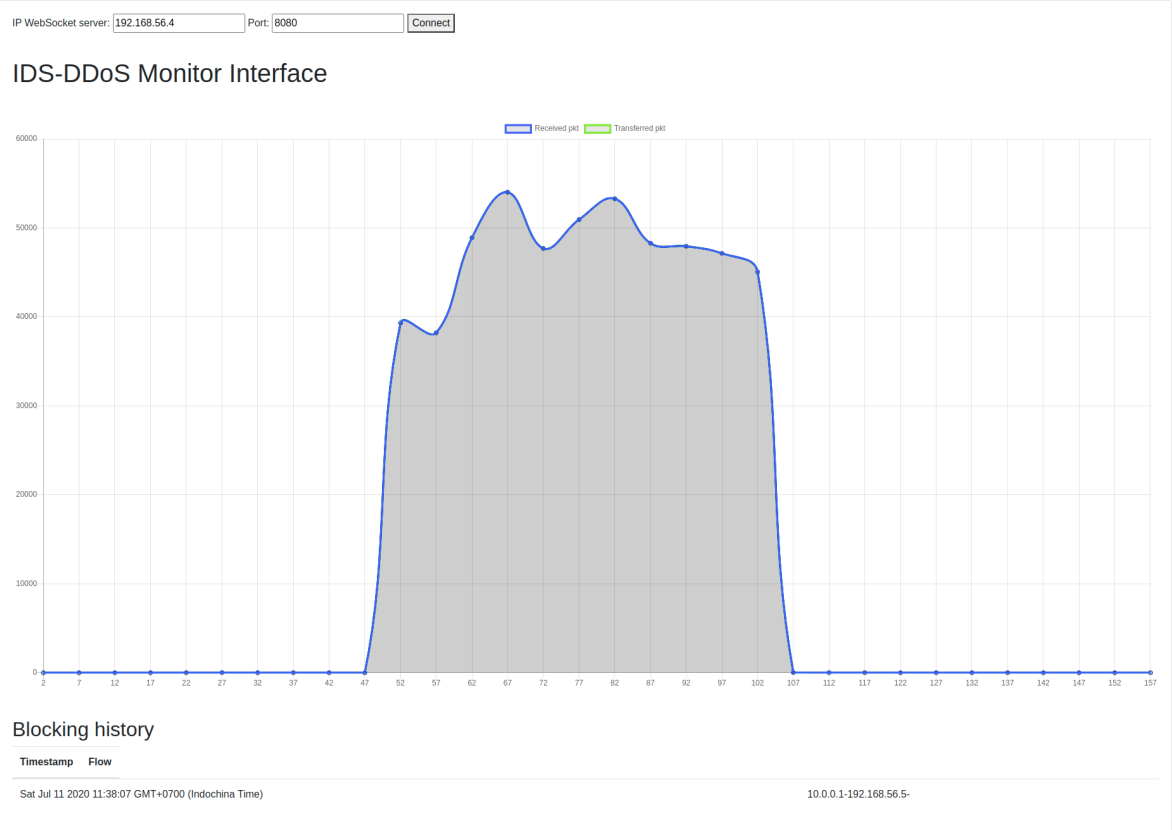
Hulk là công cụ giao diện console, với cùng lý do nêu trong mục 6.1, tôi cũng sẽ không chia sẽ cách tiếp cận công cụ này.

Quá trình tấn công được biểu thị trong biểu đồ 6.4 dưới đây.

Lúc đầu, nhìn theo trục gói tin, ta thấy được số lượng gói tin đến và gói tin đi vẫn

còn thấp.

Đường màu xanh dương biểu diễn gói tin đến, tức là các gói tin phản hồi từ Outer Webserver. Đường màu xanh lá biểu diễn gói tin đi, tức là các gói tin được gửi từ Attacker H1.



Hình 6.4: Kịch bản 2: Trạng thái mạng trong quá trình H1 tấn công

Tuy nhiên, khi H1 bắt đầu tấn công, ta thấy số lượng gói tin đến và đi tăng lên đột ngột và tăng rất cao. Tuy nhiên sau một khoảng thời gian, số lượng gói tin này giảm xuống còn 0. Lý do là vì khi IDS-DDoS phát hiện tấn công và thông báo cho SDN controller để chặn luồng IP, cho nên H1 không thể tiếp tục tạo các kết nối đến Outer Webserver. Bên cạnh đó, H1 là máy con trong mạng SDN, nên các gói tin bị chặn tại nguồn tấn công, ta không thể quan sát được số lượng gói tin đi (đường xanh lá) tiếp tục tăng như trong kịch bản 6.1 dù cho H1 vẫn tiếp tục tấn công.

6.3 Kết luận kết quả kiểm thử

Từ hai kịch bản thí nghiệm trên, tôi nhận thấy phần mềm IDS-DDoS có khả năng nhận diện và phòng thủ trước tấn công DDoS. Tuy nhiên, khi so sánh giữa hai biểu đồ 6.2 và 6.3, tôi nhận thấy, phần mềm không nhận diện tốt được các luồng tấn công từ công cụ Hulk.

Tôi đã tìm hiểu nguyên nhân và phát hiện ra rằng, trong dataset CICIDS2018 mà tôi sử dụng ở mục 4.1, các luồng tấn công Hulk mà tác giả ghi nhận rất khác so với luồng tấn công do công cụ Hulk được tôi sử dụng. Trong đó, các gói tin và thời gian của luồng mà công cụ Hulk do tôi sử dụng sẽ nhỏ hơn nhiều so với tác giả. Lý do này đã khiến cho công cụ IDS-DDoS bị nhầm lẫn hầu hết các luồng tấn công Hulk là luồng hợp lệ. Thử nghiệm với mô hình học sâu DNN, tôi cũng được kết quả tương tự.

Ngoài ra, khi áp dụng các mô hình vào phần mềm IDS-DDoS trong môi trường thực nghiệm, tôi nhận ra vấn đề về khả năng nhận diện tấn công của các mô hình trong khi huấn luyện và trên thực tế là khá khác nhau.

Để xấy dựng môi trường thực nghiệm, tôi thiết kế một mạng gồm hai máy ảo, một máy đóng vai trò là ”attacker” một máy đóng vai trò là ”victim”. Trên máy victim tôi cài đặt phần mềm IDS-DDoS để nhận diện tấn công. Sau đó, tôi sử dụng công cụ HOIC để thực hiện giả lập tấn công DDoS từ máy attacker sang máy victim.

Trong lúc thực nghiệm, mô hình Naive Bayes cho kết quả tệ nhất các luồng tấn công và luồng thông thường bị nhận diện nhầm rất nhiều. Tuy nhiên điều làm tôi ngạc nhiên là mặc dù có độ chính xác cao với kiến trúc đơn giản nhưng Decision Tree và Random Forest lại thể hiện khả năng nhận diện tấn công rất kém, hầu hết các luồng tấn công đều được nhận diện là luồng thông thường. Trong khi đó, mô hình LinearSVM và DNN tuy có các tiếp cận phức tạp hơn và độ chính xác không cao bằng, nhưng lại nhận diện rất tốt các luồng tấn công.

Chương 7

Tổng kết

Qua 6 chương trước đó tôi đã lần lượt trình bày quá trình tìm hiểu và cài đặt công cụ phát hiện và phòng thủ trước tấn công DoS/DDoS trong mạng SDN của mình. Chương này tổng kết lại những điểm được và chưa được của khóa luận này và đưa ra những hướng phát triển nếu có thể trong tương lai.

7.1 Các kết luận từ đề tài

Từ kết luận ở mục 4.2.4 và mục 6.3, tôi rút ra được các ý sau.

- Đối với dataset CICIDS2018, mô hình có cách tiếp cận đơn giản nhưng mang lại kết quả tốt nhất là Decision Tree.
- Khi áp dụng vào thực nghiệm, độ chính xác của mô hình khi huấn luyện lại khác xa so với thực tế. Ở đây, mô hình Decision Tree lại thể hiện khả năng nhận diện kém hơn LinearSVM.
- Các mô hình hiện tại sẽ bị vô hiệu trước một mẫu tấn công mới, hoàn toàn xa lạ. Điều này có thể rút ra từ phần kiểm thử với công cụ Hulk, khi mà công cụ tôi sử dụng hoạt động khác so với công cụ mà tác giả của dataset CICIDS2018 sử dụng.

– Công cụ tôi phát triển không nên sử dụng độc lập mà cần kết hợp với nhiều công cụ sử dụng các kỹ thuật khác để tăng độ chính xác và độ tin cậy.

7.2 Kết quả đạt được

Những điều làm được:

– Hiểu được các cách thức và nguy cơ của tấn công DoS/DDoS hiện nay.

– Hiểu và ứng dụng các mô hình học máy Linear SVM, Naïve Bayes, Decision Tree, Random Forest và mô hình học sâu để giải quyết vấn đề phân loại luồng gói tin.

– Huấn luyện được nhiều mô hình học máy và tìm được mô hình đơn giản mà hiệu quả là Decision Tree.

– Hoàn thành proof-of-concept công cụ phát hiện tấn công DoS/DDoS theo thời gian thực (IDS-DDoS).

– Hoàn thành proof-of-concept hệ thống mạng SDN mô phỏng quá trình tấn công và phòng thủ với hai kịch bản thường gặp.

Những điều chưa làm được:

– Chưa huấn luyện được một số mô hình học sâu do hạn chế về khả năng xử lý dữ liệu, do kích thước dữ liệu quá lớn.

– Chưa kiểm tra chéo mô hình trên các tập dữ liệu khác nhau do nhiều nguyên nhân như bất đồng về định dạng của các tập dữ liệu, kích thước tập dữ liệu quá lớn, vv.

7.3 Hướng phát triển

– Huấn luyện thêm một số mô hình học sâu.

– Kiểm tra chéo kết quả các mô hình trên các tập dữ liệu khác nhau.

- Áp dụng và đánh giá công cụ IDS-DDoS vào một số nền tảng mạng khác so với SDN.
- Tìm ra nguyên nhân vì sao độ chính xác khi huấn luyện lại khác so với thực nghiệm.
- Tìm một mô hình học máy hay học sâu tốt hơn để có thể nhận diện được công cụ tấn công không nằm trong dataset.
- Kết hợp công cụ IDS-DDoS với các công cụ khác để tăng độ tin cậy và độ chính xác.

7.4 Lời kết

Trong thời gian có hạn, tôi chỉ mới nghiên cứu được một phần trong lĩnh vực phòng chống tấn công DoS/DDoS cũng như công nghệ SDN.

Có thể những kiến thức tôi thu nhận được trong thời gian qua chưa đủ nhiều. Tuy nhiên tôi nhận thấy mình đã phát triển được những kỹ năng như: kỹ năng tìm kiếm thông tin, kỹ năng phân tích và giải quyết vấn đề, khả năng thích nghi với môi trường và công nghệ mới, vv. Với những kỹ năng thu được, tôi hi vọng sẽ giúp tôi phát triển hơn trong tương lai.

Mặc dù đã cố gắng trong quá trình thực hiện, chắc chắn khóa luận không tránh khỏi những thiếu sót. Rất mong nhận được sự góp ý và chỉ bảo tận tình của quý thầy cô và các bạn.

Tài liệu tham khảo

[1] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[2] "What is open vswitch?." `http://docs.openvswitch.org/en/latest/intro/what-is-ovs/`, [Accessed 02-Apr-2020].

[3] A. Dey, "Machine learning algorithms : A review," in *International Journal of Computer Science and Information Technologies (IJCSIT)*, vol. 7, pp. 1174–1179, 2016.

[4] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE Access*, vol. 7, pp. 53040–53065, 2019.

[5] X. Yuan, C. Li, and X. Li, "Deepdefense: Identifying ddos attack via deep learning," in *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, pp. 1–8, 2017.

[6] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martínez-del-Rincón, and D. Siracusa, "Lucid: A practical, lightweight deep learning solution for ddos attack detection," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 876–889, 2020.

[7] "Understanding denial-of-service attacks us-cert. 6 february 2013." `https://www.us-cert.gov/ncas/tips/ST04-015`, [Accessed 02-Apr-2020].

[8] "Distributed denial of service attack (ddos) definition." https://www.imperva.com/learn/application-security/ddos-attacks/, [Accessed 02-Apr-2020].

[9] Nam-Seok Ko, Sung-Kee Noh, Jong-Dae Park, Soon-Seok Lee, and Hong-Shik Park, "An efficient anti-ddos mechanism using flow-based forwarding technology," in *Digest of the 9th International Conference on Optical Internet (COIN 2010)*, pp. 1–3, 2010.

[10] M. Fallah and N. Kahani, "Tdpf: A traceback-based distributed packet filter to mitigate spoofed ddos attacks," *Security and Communication Networks*, vol. 7, pp. 254–264, 02 2014.

[11] M. Yoon, "Using whitelisting to mitigate ddos attacks on critical internet sites," *IEEE Communications Magazine*, vol. 48, no. 7, pp. 110–115, 2010.

[12] "What we know about friday's massive east coast internet outage." https://www.wired.com/2016/10/internet-outage-ddos-dns-dyn/, [Accessed 02-Apr-2020].

[13] "Github survived the biggest ddos attack ever recorded." https://www.wired.com/story/github-ddos-memcached/, [Accessed 02-Apr-2020].

[14] "Web attack knocks bbc websites offline." https://www.bbc.com/news/technology-35204915, [Accessed 02-Apr-2020].

[15] "Record-breaking ddos attack strikes cloudflare's network." https://www.cnbc.com/2014/02/11/record-breaking-ddos-attack-strikes-cloudflares-network.html, [Accessed 02-Apr-2020].

[16] "The ddos that knocked spamhaus offline (and how we mitigated it)." https://blog.cloudflare.com/the-ddos-that-knocked-spamhaus-offline-and-ho/, [Ac-

cessed 02-Apr-2020].

[17] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, p. 69–74, Mar. 2008.

[18] A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using openflow: A survey," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 493–512, 2014.

[19] S. Asadollahi, B. Goswami, and M. Sameer, "Ryu controller's scalability experiment on software defined networks," in *2018 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC)*, pp. 1–5, 2018.

[20] M. Welling and D. Bren in *A First Encounter with Machine Learning*, University of California Irvine, 2010.

[21] M. T. Bowles, *Machine Learning in Python: Essential Techniques for Predictive Analysis*. "John Wiley & Sons Inc", 2015.

[22] S. B. Kotsiantis, "Supervised machine learning: A review of classification techniques," in *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in EHealth, HCI, Information Retrieval and Pervasive Technologies*, (NLD), p. 3–24, IOS Press, 2007.

[23] A. Parmar, R. Katariya, and V. Patel, "A review on random forest: An ensemble classifier," in *International Conference on Intelligent Data Communication Technologies and Internet of Things (ICICI) 2018* (J. Hemanth, X. Fernando, P. Lafata, and Z. Baig, eds.), (Cham), pp. 758–763, Springer International Publishing, 2019.

[24] X. Du, Y. Cai, S. Wang, and L. Zhang, "Overview of deep learning," in *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pp. 159–164, 2016.

[25] W. S. McCulloch and W. Pitts, *A Logical Calculus of the Ideas Immanent in Nervous Activity*, p. 15–27. Cambridge, MA, USA: MIT Press, 1988.

[26] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation Functions: Comparison of trends in Practice and Research for Deep Learning," *arXiv e-prints*, p. arXiv:1811.03378, Nov. 2018.

[27] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, "Statistical approaches to ddos attack detection and response," in *Proceedings DARPA Information Survivability Conference and Exposition*, vol. 1, pp. 303–314 vol.1, 2003.

[28] P. Bojović, I. Bašičević, S. Ocovaj, and M. Popović, "A practical approach to detection of distributed denial-of-service attacks using a hybrid detection method," *Computers & Electrical Engineering*, vol. 73, pp. 84–96, 2019.

[29] P. Kumar, M. Tripathi, A. Nehra, M. Conti, and C. Lal, "Safety: Early detection and mitigation of tcp syn flood utilizing entropy in sdn," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1545–1559, 2018.

[30] M. Ring, S. Wunderlich, D. Scheuring, D. Land es, and A. Hotho, "A Survey of Network-based Intrusion Detection Data Sets," *arXiv e-prints*, p. arXiv:1903.02460, Mar. 2019.

[31] J. J. Santanna, R. van Rijswijk-Deij, R. Hofstede, A. Sperotto, M. Wierbosch, L. Z. Granville, and A. Pras, "Booters — an analysis of ddos-as-a-service attacks," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 243–251, 2015.

[32] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, p. 357–374, May 2012.

[33] I. Sharafaldin., A. H. Lashkari., and A. A. Ghorbani., "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy*

- *Volume 1: ICISSP,*, pp. 108–116, INSTICC, SciTePress, 2018.

[34] C. I. for Cybersecurity, "A realistic cyber defense dataset (cse-cic-ids2018)." `https://registry.opendata.aws/cse-cic-ids2018/`, [Accessed 02-Apr-2020].

[35] C. I. for Cybersecurity, "Cicflowmeter." `https://www.unb.ca/cic/research/applications.html`, [Accessed 02-Apr-2020].

[36] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 darpa off-line intrusion detection evaluation," *Comput. Netw.*, vol. 34, p. 579–595, Oct. 2000.

[37] S. Stolfo, "Kdd cup 99 dataset." `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`, [Accessed 02-Apr-2020].

[38] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–6, 2009.

[39] Z. He, T. Zhang, and R. B. Lee, "Machine learning based ddos attack detection from source side in cloud," in *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, pp. 114–120, 2017.

[40] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices," in *2018 IEEE Security and Privacy Workshops (SPW)*, pp. 29–35, 2018.

[41] A. Koay, A. Chen, I. Welch, and W. K. G. Seah, "A new multi classifier system using entropy-based features in ddos attack detection," in *2018 International Conference on Information Networking (ICOIN)*, pp. 162–167, 2018.

[42] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[43] E. Min, J. Long, Q. Liu, J. Cui, and W. Chen, "Tr-ids: Anomaly-based intrusion detection through text-convolutional neural network and random forest," *Security and Communication Networks*, vol. 2018, pp. 1–9, 07 2018.

[44] K. Wu, Z. Chen, and W. Li, "A novel intrusion detection model for a massive network using convolutional neural networks," *IEEE Access*, vol. 6, pp. 50850–50859, 2018.

[45] C. J. L. W. Ram B. Basnet, Riad Shash and T. Doleck, "Towards detecting and classifying network intrusion traffic using deep learning frameworks," *Journal of Internet Services and Information Security (JISIS)*, vol. 9, pp. 1–17, nov 2019.

[46] P. Fonseca, R. Bennesby, E. Mota, and A. Passito, "A replication component for resilient openflow-based networking," in *2012 IEEE Network Operations and Management Symposium*, pp. 933–939, 2012.

[47] Myung-Sup Kim, Hun-Jeong Kong, Seong-Cheol Hong, Seung-Hwa Chung, and J. W. Hong, "A flow-based method for abnormal network traffic detection," in *2004 IEEE/IFIP Network Operations and Management Symposium (IEEE Cat. No.04CH37507)*, vol. 1, pp. 599–612 Vol.1, 2004.

[48] J. S. C. Manso, Pedro; Moura, "Sdn-based intrusion detection system for early detection and mitigation of ddos attacks," *MDPI Information journal*, vol. 10, p. 106, mar 2019.

[49] Q. Niyaz, W. Sun, and A. Y. Javaid, "A Deep Learning Based DDoS Detection System in Software-Defined Networking (SDN)," *arXiv e-prints*, p. arXiv:1611.07400, Nov. 2016.

[50] C. Li, Y. Wu, X. Yuan, Z. Sun, W. Wang, X. Li, and L. Gong, "Detection and defense of ddos attack–based on deep learning in openflow-based sdn," *International Journal of Communication Systems*, vol. 31, no. 5, p. e3497, 2018. e3497 IJCS-17-0848.R1.

[51] A. Patle and D. S. Chouhan, "Svm kernel functions for classification," in *2013*

International Conference on Advances in Technology and Engineering (ICATE), pp. 1–9, 2013.

[52] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, p. 273–297, Sept. 1995.

[53] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, pp. 41–46, 2001.

[54] M. Somvanshi, P. Chavan, S. Tambade, and S. V. Shinde, "A review of machine learning techniques using decision tree and support vector machine," in *2016 International Conference on Computing Communication Control and automation (ICCUBEA)*, pp. 1–7, 2016.

[55] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv e-prints*, p. arXiv:1412.6980, Dec. 2014.