HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Kiều Công Minh

ỨNG DỤNG REPRESENTATION LEARNING PHÁT HIỆN TẤN CÔNG BOTNET

LUẬN VĂN THẠC SỸ KỸ THUẬT

(Theo định hướng ứng dụng)

THÀNH PHỐ HỒ CHÍ MINH - 2023

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Kiều Công Minh

ỨNG DỤNG REPRESENTATION LEARNING PHÁT HIỆN TẤN CÔNG BOTNET

CHUYÊN NGÀNH: HỆ THỐNG THÔNG TIN

MÃ SỐ: 8.48.01.04

LUẬN VĂN THẠC SỸ KỸ THUẬT

(Theo định hướng ứng dụng)

NGƯỜI HƯỚNG DẪN KHOA HỌC:

TS. NGUYỄN HỒNG SƠN

THÀNH PHỐ HỒ CHÍ MINH - 2023

LỜI CAM ĐOAN

Tôi cam đoan rằng luận văn: *"Ứng dụng Representation Learning phát hiện tấn công Botnet"* là công trình nghiên cứu của chính tôi.

Tôi cam đoan các số liệu, kết quả nêu trong luận văn là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác.

Không có sản phẩm/nghiên cứu nào của người khác được sử dụng trong luận văn này mà không được trích dẫn theo đúng quy định.

TP. Hồ Chí Minh, ngày 28 tháng 02 năm 2023

Học viên thực hiện luận văn

Kiều Công Minh

LỜI CẢM ƠN

Trong suốt quá trình học tập và nghiên cứu thực hiện luận văn, ngoài nỗ lực của bản thân, tôi đã nhận được sự hướng dẫn nhiệt tình quý báu của quý Thầy Cô, cùng với sự động viên và ủng hộ của gia đình, bạn bè và đồng nghiệp. Với lòng kính trọng và biết ơn sâu sắc, tôi xin gửi lời cảm ơn chân thành tới: Ban Giám Đốc, Phòng đào tạo sau đại học và quý Thầy Cô đã tạo mọi điều kiện thuận lợi giúp tôi hoàn thành luận văn.

Tôi xin chân thành cảm ơn Thầy **TS. Nguyễn Hồng Sơn**, người thầy kính yêu đã hết lòng giúp đỡ, hướng dẫn, động viên, tạo điều kiện cho tôi trong suốt quá trình thực hiện và hoàn thành luận văn.

Tôi xin chân thành cảm ơn gia đình, bạn bè, đồng nghiệp trong cơ quan đã động viên, hỗ trợ tôi trong lúc khó khăn để tôi có thể học tập và hoàn thành luận văn. Mặc dù đã có nhiều cố gắng, nỗ lực, nhưng do thời gian và kinh nghiệm nghiên cứu khoa học còn hạn chế nên không thể tránh khỏi những thiếu sót. Tôi rất mong nhận được sự góp ý của quý Thầy Cô cùng bạn bè đồng nghiệp để kiến thức của tôi ngày một hoàn thiện hơn.

Xin chân thành cảm ơn!

TP. Hồ Chí Minh, ngày 28 tháng 02 năm 2023

Học viên thực hiện luận văn

Kiều Công Minh

DANH SÁCH HÌNH VẼ

DANH SÁCH BẢNG

DANH MỤC CÁC THUẬT NGỮ, CHỮ VIẾT TẮT

Viết tắt	Tiếng Anh	Tiếng Việt
DDoS	Distributed Denial of Service	Tấn công từ chối dịch vụ phân tán
APT	Advanced Persistent Threat	Tấn công có chủ đích
IRC	Internet Relay Chat	
RPC	Remote procedure call	Gọi hàm từ xa
C&C / C2	Command and control	Máy chủ điều khiển và kiểm soát
HTTP	Hypertext tranfer protocol	Giao thức truyền tải siêu văn bản
IDS	Intrusion Detection System	Hệ thống phát hiện xâm nhập
P2P	Peer to peer	Mạng ngang hàng
RL	Representation Learning	Học biểu diễn
CNN	Convolutional Neural Network	Mạng nơ ron tích chập
DL	Deep Learning	Học sâu

MỤC LỤC

MỞ ĐẦU

1. Lý do chọn đề tài

Trong cuộc sống hiện đại ngày nay, với việc mạng Internet ngày càng phát triển không ngừng, công nghệ thông tin được ứng dụng vào mọi mặt của đời sống, kinh tế, chính trị, xã hội đã giúp cho cá nhân, tổ chức, doanh nghiệp và các cơ quan hành chính nhà nước trên thế giới nói chung và Việt Nam nói riêng dễ dàng trao đổi thông tin và thực hiện các giao dịch được thuận lợi nhanh chóng.

Tuy nhiên, song song với quá trình phát triển đó là kèm theo những mối đe dọa về tấn công mạng cũng xuất hiện ngày càng nhiều, trong số đó là mối đe dọa về Botnet. Khai thác, phát tán mã độc, phát tán thư rác số lượng lớn, tấn công từ chối dịch vụ DDoS và đặc biệt là tấn công APT là những hành vi nguy hiểm thường thấy của Botnet, nó đã gây ra những thiệt hại không nhỏ về hệ thống mạng và sự mất mát dữ liệu của người dung, dẫn đến thiệt hại về kinh tế, xã hội của các cá nhân, tổ chức, doanh nghiệp và cơ quan hành chính nhà nước. Việc phát hiện và ngăn chặn Botnet là một nhiệm vụ khó khăn, dẫn đến các nạn nhân của Botnet không ngừng gia tăng và với số lượng ngày càng lớn.

Thông qua việc nghiên cứu các phương pháp kỹ thuật về phát hiện và xử lý Botnet, các tổ chức chuyên gia an ninh mạng đã tìm thấy và ngăn chặn nhiều đợt tấn công mạng Botnet trên Internet. Tuy nhiên, qua thời gian Botnet liên tục thay đổi, các nhà khai thác Bot ngày càng trở nên rất tinh vi và các biện pháp chống tấn công Botnet nội bộ thường tốn nhiều thời gian và không đạt hiệu quả cao.Vì vậy việc nghiên cứu các phương pháp phát hiện và xử lý Botnet mới luôn là một lĩnh vực nghiên cứu cấp thiết và ý nghĩa.

Với sự phát triển mạnh mẽ của trí tuệ nhân tạo trong những năm gần đây, đặc biệt là các kỹ thuật máy học, đã mở ra như một giải pháp rất có tiềm năng cho việc ứng dụng phát hiện các tấn công mạng Botnet với độ chính xác và đạt hiệu quả cao hơn các phương pháp trước đây. Trong đó mô hình dựa vào phương pháp

representatin learning có thể phát huy nhiều ưu điểm cho bài toán này. Từ những lý do trên luận văn này xin chọn đề tài nghiên cứu như sau:

"Ứng dụng phương pháp representation learning phát hiện tấn công botnet"

2. Tổng quan về vấn đề nghiên cứu

Botnet [3] thuật ngữ đầy đủ là "Bots network" dùng để chỉ một mạng lưới các máy tính bị chi phối bởi ai đó và bị điều khiển bởi một con máy tính khác từ xa. Botnet là một phần mềm độc hại, đa phần các máy tính đều bị nhiễm bởi một Bot nào đó mà chúng ta không thể nào phát hiện được. Các máy tính đang bị nhiễm Botnet nôm na đều gọi là các "Zombie". Máy tính bị nhiễm sẽ bị chi phối bởi một Botmaster ở trên và điều khiển mọi hoạt động của máy tính đang dính mã độc làm cản trở hoạt động, gián đoạn gây mất nhiều thời gian, giảm năng suất công việc của người dùng. Cách chúng ta trở thành nạn nhân của nó giống như việc bị lây nhiễm malware, và cách thức chiếm và sử dụng dữ liệu đánh cắp cũng chỉ với mục đích riêng của hacker.

Botnet là từ chỉ một tập hợp các robot phần mềm hoặc các con bot hoạt động một cách tự chủ. Từ này còn được dùng để chỉ một mạng các máy tính sử dụng phần mềm tính toán phân tán. Tuy từ "botnet" có thể dùng để chỉ một nhóm bot bất kỳ, chẳng hạn IRC bot, từ này thường được dùng để chỉ một tập hợp các máy tính đã bị tấn công và thỏa hiệp và đang chạy các chương trình độc hại, thường là sâu máy tính, trojan horse hay các cửa hậu, dưới cùng một hạ tầng cơ sở lệnh và điều khiển. Một chương trình chỉ huy botnet (botnet's originator hay bot herder) có thể điều khiển cả nhóm bot từ xa, thường là qua một phương tiện chẳng hạn như IRC, và thường là nhằm các mục đích bất chính. Mỗi con bot thường chạy ẩn và tuân theo chuẩn RFC 1459 (IRC). Thông thường, kẻ tạo botnet trước đó đã thỏa hiệp một loạt hệ thống bằng nhiều công cụ đa dạng (tràn bộ nhớ đệm,...). Các bot mới hơn có thể tự động quét môi trường của chúng và tự lan truyền bản thân bằng cách sử dụng các lỗ hổng an ninh và mật khẩu yếu. Nếu một con Bot có thể quét và tự lan truyền qua càng nhiều lỗ hổng an ninh, thì nó càng trở nên giá trị đối với một cộng đồng điều khiển botnet.

Các botnet đã trở nên một phần quan trọng của Internet, tuy chúng ngày càng ẩn kĩ. Do đa số các mạng IRC truyền thống thực hiện các biện pháp cấm truy nhập đối với các botnet đã từng ngụ tại đó, những người điều khiển botnet phải tự tìm các server cho mình. Một botnet thường bao gồm nhiều kết nối, chẳng hạn quay số, ADSL và cáp, và nhiều loại mạng máy tính, chẳng hạn mạng giáo dục, công ty, chính phủ và thậm chí quân sự. Đôi khi, một người điều khiển giấu một cài đặt IRC server trên một site công ty hoặc giáo dục, nơi các đường kết nối tốc độ cao có thể hỗ trợ một số lớn các bot khác. Chỉ đến gần đây, phương pháp sử dụng bot để chỉ huy các bot khác mới phát triển mạnh, do đa số hacker không chuyên (script kiddie) không đủ kiến thức để sử dụng phương pháp này.

Botnet [4] có thể bị trục xuất hoặc ngừng xâm nhập vào máy tính bằng cách sử dụng chương trình chống phần mềm độc hại, có thể phát hiện việc lây nhiễm trên ổ cứng hoặc lưu lượng mạng và xử lý chúng ngay lập tức. Mặt khác, cách tiếp cận hiệu quả nhất sẽ là tìm hiểu để nhận thức được toàn diện về cách chống lại botnet. Có rất nhiều biện pháp và cách thức ngăn chặn và phát hiện BotNet, tuy nhiên tất cả đều phải thông qua quan sát, giám sát của máy tính và hệ thống mạng. Ở cấp độ mạng, phát hiện BotNet không hề đơn giản, và phức tạp hơn khi các máy trong mạng cho phép các bots này như một ứng dụng chính thống. Chính vì thế việc phát hiện thông qua các công cụ thông thường phổ biến chưa thật sự hiệu quả. Việc nghiên cứu ứng dụng các phương pháp máy học dựa trên bộ dữ liệu mạng để phát hiện ra các tấn công là rất cần thiết và hiệu quả. Đã có rất nhiều nghiên cứu áp dụng học máy vào phát hiện tấn công, đặc biệt là các kỹ thuật học sâu.

Trong bài báo [5] các tác giả đã đưa ra mô hình phát hiện các loại DGA botnet và FF botnet dựa trên phân loại các tên miền độc hại sử dụng bởi botnet và các tên miền bình thường sử dụng một số kỹ thuật học máy có giám sát thông dụng. Trong đó có 2 giai đoạn: (1) giai đoạn huấn luyện và (2) là giai đoạn phát hiện. Ở giai đoạn huấn luyện, tác giả đã sử dụng các thuật toán của học máy trên các tập huấn luyện để đưa ra đánh giá hiệu quả của việc phân loại tên miền bình thường và tên miền độc hại sử dụng bởi botnet. Qua giai đoạn thực hiện để phát hiện botnet, kết quả cho thấy

thuật toán học máy Rừng ngẫu nhiên cho tỉ lệ thành công cao nhất so với các thuật toán còn lại.

Trong bài báo [6] các tác giả đã đề xuất bổ sung 4 đặc trưng: (1) tên miền dạng băm, (2) giá trị dự kiến cho mỗi tên miền, (3) số lượng từ có nghĩa trong mỗi tên miền, (4) độ dài tên miền. Trong đó, đặc trưng tên miền dạng băm giúp phân biệt nhóm tên miền DGA sử dụng giá trị băm, hoặc ký tự thập lục phân; đặc trưng giá trị dự kiến cho mỗi tên miền giúp tăng khả năng phân biệt dựa trên thống kê; số lượng từ có nghĩa trong mỗi tên miền lành tính thường cao hơn so với tên miền DGA; và độ dài tên miền nhỏ hơn 5 thường là tên miền lành tính. Cả 4 đặc trưng trên đều góp phần làm tăng khả năng phát hiện các botnet dựa trên tên miền đạt kết quả khá cao. Tuy nhiên, nó cũng có nhược điểm là làm tăng thời gian huấn luyện và phát hiện do tăng kích thước véctơ biểu diễn tên miền dù theo khảo sát của tác giả là không lớn.

Trong bài báo [7] các tác giả Hoàng Xuân Dậu, Nguyễn Trọng Hưng, Ninh Thị Thu Trang đã nghiên cứu phân tích ảnh hưởng các yếu tố hiệu quả từ phương pháp phát hiện DGA botnet dựa trên học máy sử dụng dữ liệu truy vấn DNS. Các tác giả đã đưa ra xem xét, phân tích các yếu tố bao gồm: (1) vấn đề sử dụng hoặc loại bỏ vấn đề sử dụng hoặc loại bỏ phần tên miền cấp cao nhất và (2) ảnh hưởng của các nhóm đặc trưng huấn luyện. Kết quả nghiên cứu cho thấy phần tên miền cấp cao nhất giúp tăng đáng kể hiệu quả phát hiện tấn công Botnet.

Trong bài báo [8] tác giả đã trình bày một phương pháp dựa trên máy chủ để phát hiện và phân biệt các loại lây nhiễm botnet khác nhau dựa trên các kiểu C&C của botnet như dựa trên IRC, dựa trên HTTP hoặc peer to peer (P2P). Thông qua các đặc tính lưu lượng mạng botnet C&C với lưu lạng mạng hợp lệ. Giải pháp của tác giả đạt hiệu quả có độ chính xác khá cao. Qua bài báo, ta có thể tham khảo giải pháp cho hướng nghiên cứu của đề tài.

Trong bài báo [9] các tác giả đã có những cái nhìn tổng quan và phân tích về học biểu diễn (RL) gồm những ưu điểm, nhược điểm và các phương pháp kỹ thuật của học biểu diễn, cũng như những hướng cần cải thiện về phương pháp học biểu diễn trong thời gian tới. Các nghiên cứu [10], [11] giúp ta có cái nhìn tổng quan và

có thể hiểu hơn về học biểu diễn, ứng dụng của máy học trong việc phát hiện ra các xâm nhập tấn công BotNet, và hiệu quả cao của phương pháp này. Từ những tài liệu này, cho thấy tiềm năng của các ứng dụng này.

Trong thập kỷ qua, những nỗ lực khoa học đáng kể đã được đầu tư vào việc phát triển các phương pháp có thể cung cấp khả năng phát hiện botnet hiệu quả và hiệu quả. Kết quả là, một loạt các phương pháp phát hiện dựa trên các nguyên tắc kỹ thuật đa dạng và nhắm mục tiêu các khía cạnh khác nhau của các hiện tượng botnet đã được xác định. Vì botnet dựa vào Internet để giao tiếp với kẻ tấn công cũng như để thực hiện các chiến dịch tấn công khác nhau, nên phân tích lưu lượng mạng là một trong những phương tiện chính để xác định sự tồn tại của chúng. Ngoài việc dựa vào phân tích lưu lượng để phát hiện botnet, nhiều phương pháp hiện đại sử dụng kỹ thuật học máy để xác định lưu lượng độc hại. Bài báo [12] trình bày một cuộc khảo sát về các phương pháp phát hiện mạng botnet hiện đại dựa trên học máy để xác định lưu lượng mạng botnet. Bài báo cung cấp một cái nhìn tổng quan toàn diện về các công trình khoa học hiện có, do đó góp phần hiểu rõ hơn về các khả năng, hạn chế và cơ hội của việc sử dụng học máy để xác định lưu lượng botnet. Hơn nữa, bài báo nêu ra các khả năng cho sự phát triển trong tương lai của các hệ thống phát hiện botnet dựa trên máy học.

Phân tích các chứng tích để lại từ botnet [13], hay còn gọi là phân tích pháp y botnet, giúp hiểu được bản chất của các cuộc tấn công và phương thức hoạt động mà những kẻ tấn công sử dụng. Các cuộc tấn công botnet rất khó theo dõi vì tốc độ nhanh, tính chất “dịch bệnh” và quy mô nhỏ hơn. Máy học hoạt động như một liều thuốc chữa bách bệnh cho các vấn đề liên quan đến tấn công mạng botnet. Nó không chỉ tạo điều kiện phát hiện mà còn giúp ngăn chặn sự tấn công của bot. Mô hình điều tra được đề xuất nỗ lực cải thiện chất lượng kết quả bằng cách phát hiện toàn diện mạng botnet và phân tích pháp y. Kịch bản này đã được áp dụng trong tám sự kết hợp khác nhau của kỹ thuật phân loại tổng hợp để phát hiện bằng chứng botnet. Nghiên cứu cũng được so sánh giữa các bộ phân loại dựa trên tập hợp với bộ phân loại duy nhất

sử dụng các tham số khác nhau. Các kết quả cho thấy rằng mô hình được đề xuất có thể cải thiện độ chính xác trên một bộ phân loại duy nhất.

Android là hệ điều hành di động phổ biến nhất hiện nay. Chính vì thế nên android đã trở thành mục tiêu của phần mềm nhiều độc hại. Các ứng dụng độc hại được thiết kế để biến thiết bị di động thành các bot có thể tạo thành một phần của mạng botnet lớn hơn đã trở nên khá phổ biến, do đó chúng gây ra mối đe dọa nghiêm trọng. Điều này yêu cầu các phương pháp hiệu quả hơn để phát hiện botnet trên nền tảng Android. Do đó, trong bài báo [14], các tác giả trình bày một cách tiếp cận học tập sâu để phát hiện botnet Android dựa trên Mạng thần kinh tích hợp (CNN). Hệ thống phát hiện botnet được đề xuất của các tác giả được triển khai dưới dạng mô hình dựa trên CNN được đào tạo về 342 tính năng ứng dụng tĩnh để phân biệt giữa ứng dụng botnet và ứng dụng bình thường. Mô hình phát hiện botnet được đào tạo được đánh giá trên một tập hợp 6.802 ứng dụng thực có chứa 1.929 botnet từ tập dữ liệu mạng botnet ISCX công khai. Kết quả cho thấy rằng phương pháp dựa trên CNN của họ có độ chính xác tổng thể cao nhất so với các bộ phân loại học máy phổ biến khác. Hơn nữa, kết quả hiệu suất được quan sát từ mô hình của họ tốt hơn so với kết quả được báo cáo trong các nghiên cứu trước đây về phát hiện botnet Android dựa trên máy học.

Botnet [15] là mối đe dọa chính đối với an ninh trên Internet. Khả năng phân biệt chính xác lưu lượng truy cập botnet với lưu lượng không phải botnet có thể giúp giảm thiểu đáng kể các botnet độc hại. Các tác giả trình bày một cách tiếp cận mới để phát hiện botnet áp dụng học sâu trên các luồng gói tin TCP / UDP / IP. Trong kết quả thử nghiệm của họ với một tập dữ liệu lớn, nhóm tác giả đã thu được độ chính xác 99,7% để phân loại lưu lượng truy cập P2P-botnet. Điều này có thể nói lên rằng nghiên cứu của họ có độ chính xác bằng hoặc tốt hơn so với các phương pháp phát hiện botnet thông thường, đồng thời giảm thiểu nỗ lực về kỹ thuật tính năng và lựa chọn tính năng.

Botnet [16] hiện là nguyên nhân chính cho nhiều cuộc tấn công mạng, chẳng hạn như tấn công DDoS và thư rác. Tuy nhiên, hầu hết các phương pháp phát hiện

truyền thống chủ yếu dựa vào các tiêu chí phát hiện gồm nhiều giai đoạn và được thiết kế chủ yếu là dựa trên kinh nghiệm. Trong bài báo này, các tác giả xem xét những thách thức về thiết kế mạng nơ-ron khi sử dụng các kỹ thuật học sâu hiện đại để tìm hiểu các chính sách phát hiện botnet tự động. Để tạo dữ liệu đào tạo, các tác giả đã tổng hợp các kết nối mạng botnet với các mẫu giao tiếp cơ bản khác nhau được phủ trên các mạng thực quy mô lớn dưới dạng tập dữ liệu. Để nắm bắt cấu trúc phân cấp quan trọng của các mạng botnet tập trung và cấu trúc trộn nhanh cho các mạng botnet phi tập trung, họ điều chỉnh biểu đồ mạng thần kinh (GNN) để phát hiện các thuộc tính của các cấu trúc này. Kết quả thử nghiệm cho thấy các GNN có khả năng nắm bắt cấu trúc mạng botnet tốt hơn so với các phương pháp không học trước khi được đào tạo với dữ liệu thích hợp và các GNN sâu hơn rất quan trọng để học các cấu trúc mạng botnet khó. Các tác giả tin rằng dữ liệu và nghiên cứu của họ có thể hữu ích cho cả cộng đồng học tập đồ thị và an ninh mạng.

Botnet [17] là các vector mà qua đó tin tặc có thể chiếm quyền kiểm soát nhiều hệ thống và tiến hành các hoạt động độc hại. Các nhà nghiên cứu đã đề xuất nhiều giải pháp để phát hiện và xác định các mạng botnet trong thời gian thực. Tuy nhiên, những giải pháp được đề xuất này có những điểm khó khăn khi bắt kịp với sự phát triển nhanh chóng của mạng botnet. Bài báo này đề xuất một mô hình phát hiện botnet bằng cách sử dụng học sâu để xác định các cuộc tấn công botnet zero-day trong thời gian thực. Mô hình đề xuất được đào tạo và đánh giá trên tập dữ liệu CTU-13 với nhiều thiết kế mạng nơ-ron và các lớp ẩn. Kết quả chứng minh rằng mô hình mạng thần kinh nghệ thuật học sâu có thể xác định chính xác và tiện lợi các botnet.

Những tiến bộ của Internet đã cho phép kết nối nhiều thiết bị hơn vào công nghệ này mỗi ngày. Sự xuất hiện của Internet of Things đã tổng hợp sự tăng trưởng này. Thiếu bảo mật trong thế giới IoT khiến các thiết bị này trở thành mục tiêu nóng cho bọn tội phạm mạng thực hiện các cuộc tấn công động độc hại của chúng. Một trong những hành động này là cuộc tấn công Botnet, một trong những mối đe dọa phá hoại chính đã phát triển từ năm 2003 thành nhiều dạng khác nhau. Cuộc tấn công này là một mối đe dọa nghiêm trọng đối với an ninh và quyền riêng tư của thông tin. Khả

năng mở rộng, cấu trúc, sức mạnh và chiến lược của nó cũng đang được phát triển liên tục và nó đã tồn tại trong nhiều thập kỷ. Bot [18] được định nghĩa là một ứng dụng phần mềm thực hiện một số tác vụ tự động (đơn giản nhưng có cấu trúc lặp đi lặp lại) qua Internet. Một số bot tạo ra một mạng botnet lây nhiễm sang một số thiết bị và giao tiếp với bộ điều khiển của chúng được gọi là botmaster để nhận hướng dẫn của chúng. Một mạng botnet thực thi các nhiệm vụ với tốc độ không thể được thực hiện bởi một con người. Ngày nay, các hoạt động của bot được che giấu giữa các luồng web bình thường và chiếm hơn một nửa lưu lượng truy cập web. Việc sử dụng bot nhiều nhất là trong việc khai thác web (trình thu thập thông tin web), trong đó tập lệnh tự động tìm nạp, phân tích và tệp thông tin từ các máy chủ web. Chúng cũng góp phần vào các cuộc tấn công khác, chẳng hạn như DDoS, SPAM, đánh cắp danh tính, lừa đảo và gián điệp. Một số kỹ thuật phát hiện botnet đã được đề xuất, chẳng hạn như dựa trên honeynet và Hệ thống phát hiện xâm nhập (IDS). Các kỹ thuật này không còn hiệu quả nữa do sự cập nhật liên tục của các bot và cơ chế trốn tránh của chúng. Gần đây, các kỹ thuật phát hiện botnet dựa trên máy học / học sâu đã được đề xuất có nhiều khả năng hơn so với các kỹ thuật đã đề cập trước đây. Trong nghiên cứu này, các tác giả đề xuất một công cụ dựa trên học tập sâu để phát hiện botnet sẽ được sử dụng trong IoT và các thiết bị đeo được. Trong hệ thống này, dữ liệu lưu lượng mạng bình thường và mạng botnet được chuyển đổi thành hình ảnh trước khi được đưa vào một mạng nơ-ron phức hợp sâu, được đặt tên là DenseNet có và không tính đến quá trình học chuyển giao. Hệ thống được triển khai bằng ngôn ngữ lập trình Python và Bộ dữ liệu CTU-13 được sử dụng để đánh giá trong một nghiên cứu. Theo kết quả mô phỏng của các tác giả, sử dụng học chuyển giao có thể cải thiện độ chính xác từ 33,41% lên đến 99,98%. Ngoài ra, hai bộ phân loại khác của Máy vectơ hỗ trợ (SVM) và hồi quy hậu cần đã được sử dụng. Họ cho thấy độ chính xác lần lượt là 83,15% và 78,56%. Trong một nghiên cứu khác, các tác giả đánh giá hệ thống của họ bằng một tập dữ liệu bình thường trực tiếp nội bộ và một tập dữ liệu botnet duy nhất. Tương tự, hệ thống thực hiện rất tốt việc phân loại dữ liệu trong các nghiên cứu này. Để kiểm tra khả năng ứng dụng thời gian thực của hệ thống, các tác giả đo thời gian

đào tạo và kiểm tra hệ thống. Theo kiểm tra của họ, phải mất 0,004868 mili giây để xử lý mỗi gói từ dữ liệu lưu lượng mạng trong quá trình thử nghiệm.

Nhận xét đánh giá: qua các bài báo nhìn chung có nhiều nghiên cứu đưa ra mô hình phân tích đánh giá dựa trên các đặc trưng ứng dụng các thuật toán trong máy học để phát hiện các loại botnet. Kết quả nghiên cứu khá đa dạng và đều cho kết quả khả quan. Tuy nhiên, việc nâng cao hiệu quả hơn nữa để phát hiện các tấn công botnet với mức độ phát triển công nghệ vũ bảo như ngày nay là một nhu cầu thiết yếu. Trong đó việc ứng dụng máy học với kỹ thuật representation learning để phát hiện botnet cũng được xem như là một giải pháp cần thiết.

3. Mục đích nghiên cứu

Mục tiêu chính của nghiên cứu: "Xây dựng mô hình máy học sử dụng phương pháp representation learning để phát hiện tấn công botnet nhằm nâng cao độ chính xác của phát hiện".

Từ mục tiêu trên, luận văn sẽ có những mục tiêu cụ thể như sau:

Nghiên cứu cơ sở lý thuyết về tấn công Botnet, các kỹ thuật phát hiện ra tấn công botnet.

Nghiên cứu về thuật toán máy học Representation Learning, các ưu điểm nhược điểm và đặc tính của phương pháp này.

Nghiên cứu và thu thập bộ dữ liệu liên quan tới tấn công botnet, malware… để nhằm phát hiện ra botnet. Từ đó xây dựng mô hình dự báo / cảnh báo tấn công botnet thông qua dữ liệu huấn luyện.

Nghiên cứu xây dựng ứng dụng phát hiện tấn công botnet thông qua mô hình dự báo với representation learning.

4. Đối tượng và phạm vi nghiên cứu

4.1. Đối tượng nghiên cứu

Đối tượng nghiên cứu là hệ thống phát hiện tấn công bot net bằng máy học thông qua phương pháp representation learning, cụ thể là:

Tìm hiểu tổng quan về Bot, Botnet.

Tìm hiểu các đặc trưng của Botnet.

Tìm hiểu tổng quan các kỹ thuật phát hiện Botnet.

Nghiên cứu về representation learning trong học máy.

Nghiên cứu chính là phát hiện tấn công Botnet thông qua máy học sử dụng phương pháp representation learning.

Nghiên cứu các kỹ thuật máy học phổ biến như R, MatLab, Python… để xây dựng mô hình phát hiện tấn công bằng phương pháp máy học.

4.2. Phạm vi nghiên cứu

Mô phỏng thực hiện trong mạng LAN nhỏ (từ 3~5 máy) có một số máy chủ ảo có kết nối Internet nhiễm mã độc Botnet.

Hoặc thử nghiệm tại phòng lab trung tâm tích hợp dữ liệu của tỉnh Tây Ninh.

5. Phương pháp nghiên cứu

Phương pháp luận: Dựa trên cơ sở là các lý thuyết về tấn công mạng Botnet, phương pháp representation learning trong máy học.

Dự kiến ứng dụng RL để xây dựng một trong các mô hình sau: representation learning có giám sát, không giám sát và học sâu. Dự kiến ứng dụng dựa trên tập dữ liệu phân loại các dấu vết các lưu lượng truy cập mạng hợp lệ và lưu lượng truy cập mạng của Botnet hoặc sử dụng tập dữ liệu botnet CTU-13 tại https://mcfp.felk.cvut.cz/publicDatasets/CTU-13-Dataset/ để nghiên cứu, ứng dụng.

Phương pháp đánh giá dựa trên cơ sở toán học: Trên cơ sở lý thuyết của representation learning, đưa ra đề xuất thuật toán thông qua việc xây dựng mô hình detect botnet, có khả năng phát hiện tấn công Botnet với độ chính xác cao. Từ đó đánh giá hiệu quả của mô hình.

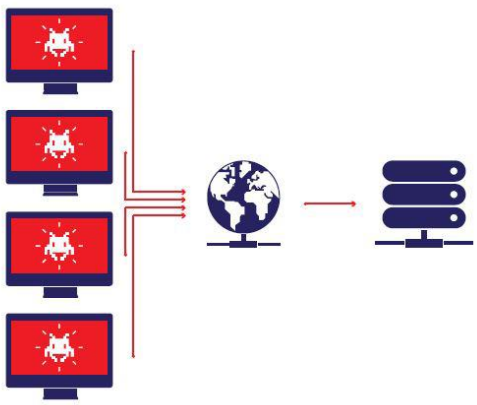
Phương pháp đánh giá bằng mô phỏng thực nghiệm: Xây dựng mô hình mô phỏng và đưa vào thực nghiệm.

CHƯƠNG 1. TỔNG QUAN TẤN CÔNG BOTNET VÀ REPRESENTATION LEARNING

1.1. Tổng quan về tấn công Botnet

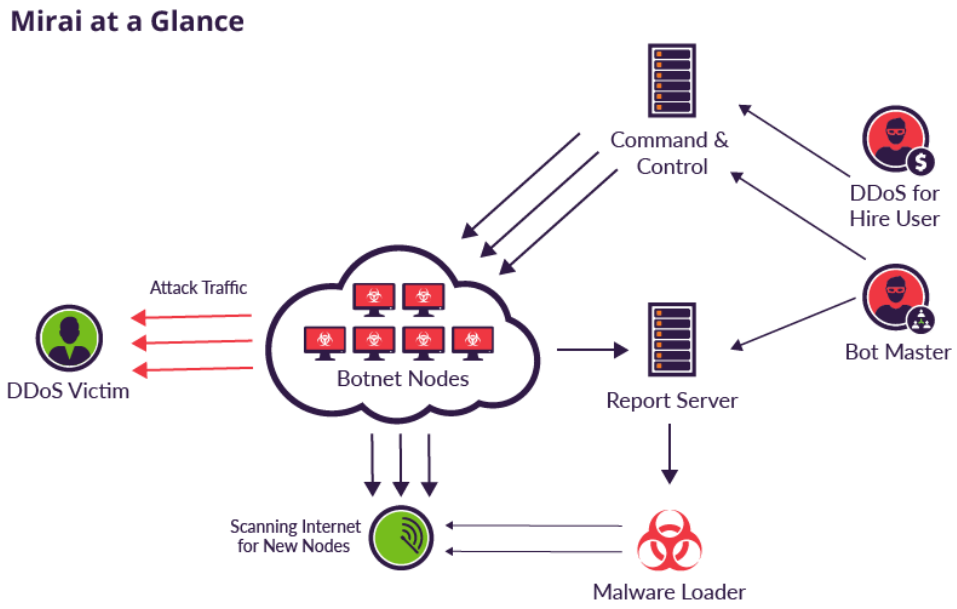
1.1.1. Botnet là gì?

Botnet [19] đang nổi lên như một mối đe dọa đáng kể nhất mà hệ sinh thái trực tuyến và tài sản máy tính phải đối mặt (Hình 1.1). Mạng botnet độc hại là các nền tảng máy tính phân tán chủ yếu được sử dụng cho các hoạt động bất hợp pháp như khởi chạy các cuộc tấn công DDoS, gửi thư rác, trojan và email lừa đảo, phân phối bất hợp pháp phần mềm và phương tiện vi phạm bản quyền, đánh cắp thông tin và tài nguyên máy tính, tống tiền hệ thống kinh doanh điện tử, thực hiện gian lận nhấp chuột và đánh cắp danh tính [3, 4]. Ưu điểm của botnet là khả năng cung cấp tính năng ẩn danh thông qua việc sử dụng kiến trúc điều khiển và lệnh nhiều tầng (C&C). Hơn nữa, các bot riêng lẻ không thuộc sở hữu thực của quản trị viên bot và có thể được đặt ở một số địa điểm trên toàn cầu. Sự khác biệt về múi giờ, ngôn ngữ và luật pháp khiến việc theo dõi các hoạt động botnet độc hại trên khắp các biên giới quốc tế trở nên khó khăn [2, 5]. Đặc điểm này làm cho botnet trở thành một công cụ hấp dẫn đối với tội phạm mạng và trên thực tế là một mối đe dọa lớn đối với an ninh mạng.



Hình 1.1: Ví dụ về botnet [3]

Mạng botnet [2] là một mạng lưới các máy tính bị xâm nhập, được điều khiển từ xa, được sử dụng cho các mục đích xấu. Các hệ thống hoặc máy chủ bị nhiễm trong một mạng botnet được gọi là Bots và bộ điều khiển của một mạng botnet được gọi là bot-master. Là mạng lớp phủ khổng lồ, botnet là nền tảng hỗ trợ phức tạp để tấn công người khác bằng cách duy trì quyền kiểm soát một số lượng lớn máy tính và các thiết bị khác. Vòng đời của botnet bao gồm một số bước bắt đầu từ khi bot-master lây nhiễm cho nạn nhân qua phần mềm độc hại, v.v… Bot bị nhiễm kết nối với các C&C bằng cách sử dụng HTTP, IRC hoặc bất kỳ giao thức khác. Bot-master gửi lệnh đến bot bằng máy chủ C&C và từ từ tạo ra một đội quân bot (Hình 1.2).



Hình 1.2: Sơ đồ cách thức tấn công của Botnet [3]

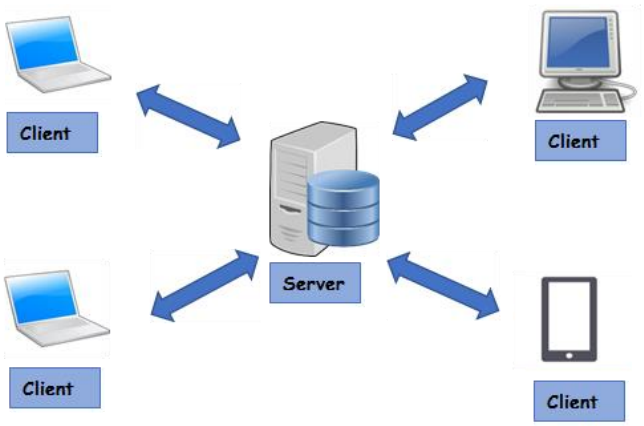
Mạng botnet [19] là một mạng lưới các máy tính bị xâm nhập được gọi là "Bots" dưới sự điều khiển từ xa của người điều hành con người được gọi là "Botmaster". Thuật ngữ "Bot" có nguồn gốc từ từ "Robot"; và tương tự như robot, bot được thiết kế để thực hiện một số chức năng được xác định trước theo cách tự động. Nói cách khác, các bot riêng lẻ là các chương trình phần mềm chạy trên máy tính chủ cho phép quản trị viên bot điều khiển các hành động của máy chủ từ xa [19]. Các botnet đặt ra một mối đe dọa đáng kể và ngày càng gia tăng đối với an ninh mạng vì chúng cung cấp một nền tảng phân tán cho nhiều hành vi không gian mạng.

1.1.2. Cấu trúc của Botnet

Một Botnet thường có một trong hai dạng như sau:

Mô hình client - server (mô hình khách - chủ)

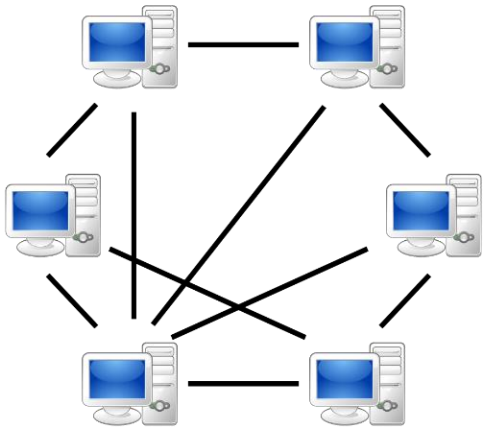
Trong cấu trúc của mô hình client – server (Hình 1.3), một mạng sẽ được thiết lập kết nối với một máy chủ từ đó hoạt động như botmaster. Botmaster này sẽ kiểm soát việc truyền tải thông tin từ máy khách để thiết lập các lệnh và điều khiển các thiết bị khách khác. Mô hình hoạt động bởi sự trợ giúp của các phần mềm đặc biệt và nó cho phép botmaster giữ quyền kiểm soát. Tuy nhiên, mô hình này tồn tại một vài nhược điểm như chỉ có một điểm kiểm soát và có thể định vị một cách dễ dàng. Ngoài ra, với mô hình client – server nếu máy chủ bị phá hủy thì botnet sẽ "chết".



Hình 1.3: Mô hình client – server [3]

Mô hình peer-to-peer (ngang hàng)

Với mục tiêu khắc phục nhược điểm của việc chỉ dựa vào một máy chủ tập trung thì các botnet đã phát triển và kết nối với nhau dưới dạng cấu trúc ngang hàng (Hình 1.4). Trong mô hình ngang hàng, mỗi một thiết bị được kết nối sẽ hoạt động một cách độc lập như một client và server, chúng phối hợp với nhau để cập nhật và truyền tải thông tin qua lại. Cấu trúc botnet sử dụng mô hình ngang hàng mạnh hơn do không có nơi điều khiển tập trung duy nhất.



Hình 1.4: Mô hình peer-to-peer [3]

1.1.3. Các loại tấn công Botnet

Distributed Denial of Operations Service (DDoS)

Các cuộc tấn công DDoS có thể sử dụng Botnet nhằm mục đích phá hủy kết nối và dịch vụ mạng bằng cách làm quá tải tài nguyên tính toán hoặc làm tiêu tốn băng thông của mục tiêu. Những cuộc tấn công TCP SYN và UDP flood là những cuộc tấn công được thực hiện phổ biến nhất. Ngoài ra, không chỉ giới hạn ở phạm vi máy chủ web, các cuộc tấn công DDoS còn có thể nhắm mục tiêu vào bất kỳ dịch vụ nào kết nối với Internet. Bằng cách dùng HTTP flood lên trang web của nạn nhân, mức độ nghiêm trọng của các cuộc tấn công còn có thể được tăng lên (hình thức này gọi là spidering – được tiến hành nhằm tăng tải hiệu quả).

Phát tán thư rác và giám sát lưu lượng

Mỗi bot có thể được dùng để xác định sự xuất hiện của dữ liệu nhạy cảm trong các máy tính bị nhiễm; xác định vị trí các botnet phía đối thủ nếu chúng được cài đặt trong cùng một máy. Một số bot có thể mở giao thức proxy SOCKS v4/v5, một khi proxy SOCKS được kích hoạt trên máy tính bị nhiễm thì nó có thể được dùng cho nhiều mục đích khác nhau, chẳng hạn như phát tán thư rác (hay còn gọi là spamming). Bằng cách sử dụng packet sniffer, bot theo dõi thông tin hay dữ liệu được truyền bởi máy bị xâm nhập, từ đó sniffer có thể truy xuất các thông tin nhạy cảm của người dùng như người đăng nhập và mật khẩu.

Keylogging

Với sự hỗ trợ của keylogger, việc botmaster đánh cắp dữ liệu và lấy thông tin nhạy cảm đã trở nên dễ dàng hơn. Khi sử dụng chương trình này kẻ tấn công có thể thu thập các phím được nhập vào trong PayPal, Yahoo, v.v... OSX/XSLCmd – một loại phần mềm gián điệp có khả năng chuyển từ Windows sang OS X, có thể thực hiện keylogging và chụp màn hình.

Đánh cắp danh tính hàng loạt

Các loại bot khác nhau có thể kết hợp lại để triển khai hành vi trộm cắp danh tính với quy mô lớn, đây cũng là một trong những hành vi phạm tội có tốc độ phát triển nhanh nhất. Bot gửi email spam để chuyển hướng lưu lượng truy cập đến các website giả mạo mang chức năng đại diện cho bot nhằm thu thập dữ liệu cá nhân. Thêm vào đó, bot có thể được sử dụng với danh phận là công ty hợp pháp và yêu cầu người dùng gửi các thông tin cá nhân của họ như: tài khoản ngân hàng, mật khẩu, chi tiết thuế, chi tiết thẻ tín dụng, v.v... Hành vi đánh cắp danh tính hàng loạt được triển khai bằng cách sử dụng các email phishing lừa nạn nhân nhập thông tin đăng nhập của họ trên các trang web như Amazon, eBay, hoặc thậm chí ngay cả ngân hàng.

Lạm dụng việc trả tiền cho mỗi lần nhấp

Ứng dụng mang tên AdSense (Google) cho phép các website hiển thị quảng cáo Google để hưởng lợi nhuận từ chúng. Chủ sở hữu trang web sẽ được Google trả tiền dựa trên số lần nhấp của người dùng vào quảng cáo của họ. Máy bị nhiễm botnet sẽ được sử dụng để tự động nhấp vào quảng cáo từ đó làm tăng số lần nhấp được gửi đến công ty.

Lây lan botnet

Botnet có khả năng lây lan từ botnet này sang botnet khác bằng cách thuyết phục người dùng tải về chương trình hay ứng dụng nào đó thông qua HTTP, email, hay FTP. Thông thường, virus sử dụng botnet dạng này sẽ được phát tán qua chính các email này.

Phần mềm quảng cáo

Phần mềm quảng cáo được dùng với mục đích thu hút người dùng bằng các quảng cáo trên các website hoặc ứng dụng. Các quảng cáo này xuất hiện mà không cần sự cho phép từ người dùng, trong đó quảng cáo gốc đã bị thay thế bằng phần mềm quảng cáo lừa đảo, chúng sẽ tiến hành lây nhiễm vào hệ thống khi có bất kỳ người dùng vào nhấp vào. Phần mềm quảng cáo nhìn qua có vẻ vô hại nhưng bản chất nó tương tự như phần mềm gián điệp thu thập dữ liệu từ trình duyệt người dùng. Sử dụng phần mềm chặn quảng cáo có thể giúp ta thoát khỏi phần mềm quảng cáo này. Hiện nay, tuy có rất nhiều phần mềm chặn quảng cáo (có trả phí hay miễn phí) nhưng tốt nhất ta nên tin dùng phần mềm có giấy phép, uy tín.

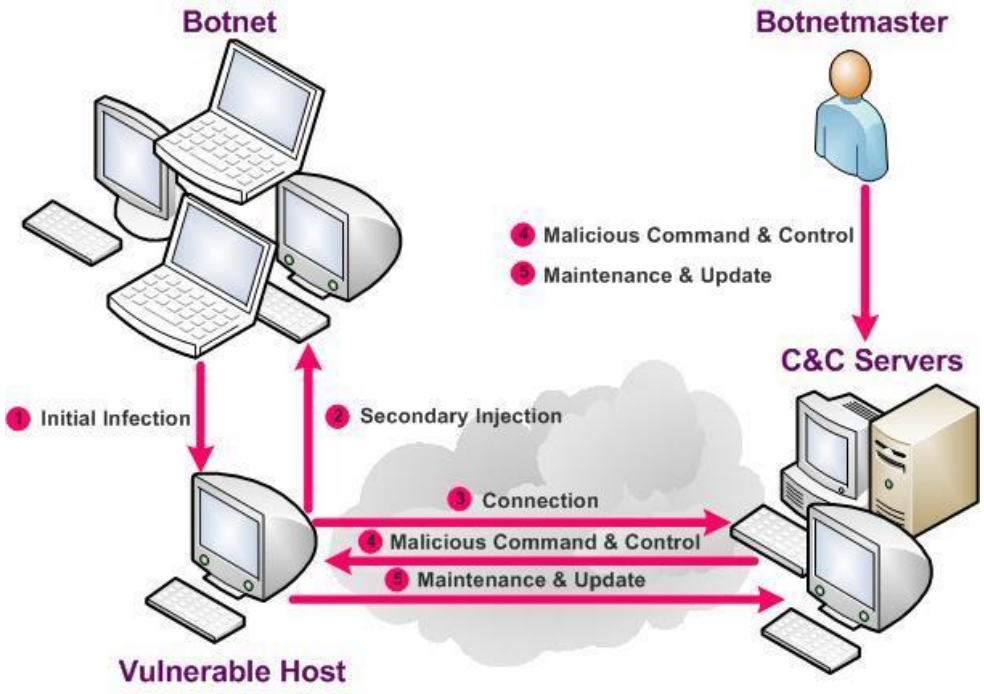
1.2. Các đặc trưng của Botnet

Giống như các thế hệ vi rút và sâu trước đây, bot là một ứng dụng tự lan truyền để lây nhiễm cho các máy chủ dễ bị tấn công thông qua các hoạt động khai thác để mở rộng phạm vi tiếp cận của chúng. Các phương pháp lây nhiễm bot tương tự như các lớp phần mềm độc hại khác tuyển dụng các hệ thống dễ bị tấn công bằng cách khai thác các lỗ hổng phần mềm, chèn trojan, cũng như các kỹ thuật xã hội dẫn đến tải xuống mã bot độc hại. Theo các nghiên cứu đo lường trong các chương trình hiện đại được trang bị một số vectơ khai thác để cải thiện cơ hội khai thác. Tuy nhiên, trong số các loại phần mềm độc hại khác, đặc điểm xác định của mạng botnet là việc sử dụng các kênh ra lệnh và điều khiển (C&C) mà qua đó chúng có thể được cập nhật và chỉ đạo. Kiến trúc C&C nhiều tầng của botnet cung cấp tính ẩn danh cho botmaster. Các kênh C&C có thể hoạt động trên một loạt các cấu trúc liên kết mạng logic và sử dụng các giao thức truyền thông khác nhau.

Botnet thường được phân loại theo kiến trúc chỉ huy và điều khiển của chúng. Theo kiến trúc lệnh và điều khiển của chúng, các botnet có thể được phân loại thành botnet dựa trên IRC, dựa trên HTTP, dựa trên DNS hoặc ngang hàng (P2P). Các botnet P2P sử dụng giao thức P2P gần đây để tránh một điểm lỗi duy nhất. Hơn nữa, các botnet P2P khó xác định vị trí, tắt máy, giám sát và chiếm quyền điều khiển hơn. Tuy nhiên, theo phân tích trong các mạng botnet phổ biến nhất dựa trên IRC với cơ

chế điều khiển và ra lệnh tập trung. Giao thức IRC ban đầu được thiết kế cho các phòng trò chuyện xã hội lớn để cho phép một số hình thức giao tiếp và phổ biến dữ liệu giữa một số lượng lớn các máy chủ cuối cùng. Sự phổ biến lớn của mạng botnet dựa trên IRC là do tính linh hoạt và khả năng mở rộng vốn có của giao thức này. Hơn nữa, có một số triển khai mã nguồn mở cho phép quản trị viên thực vật có thể mở rộng chúng theo yêu cầu của họ.

Một mạng botnet điển hình có thể được tạo và duy trì trong năm giai đoạn bao gồm: lây nhiễm ban đầu, tiêm thứ cấp, kết nối, ra lệnh và kiểm soát độc hại, cập nhật và bảo trì. Vòng đời này được mô tả trong Hình 1.5.



Hình 1.5: Vòng đời của Botnet [19]

Trong giai đoạn lây nhiễm ban đầu, kẻ tấn công sẽ quét một mạng con mục tiêu để tìm lỗ hổng đã biết và lây nhiễm vào máy nạn nhân thông qua các phương pháp khai thác khác nhau. Sau khi lây nhiễm ban đầu, trong giai đoạn tiêm thứ cấp, các máy chủ bị nhiễm thực thi một tập lệnh được gọi là mã trình bao. Mã shell tìm nạp hình ảnh của nhị phân bot thực tế từ vị trí cụ thể thông qua FTP, HTTP hoặc P2P. Hệ nhị phân bot tự cài đặt trên máy mục tiêu. Sau khi chương trình bot được cài đặt,

máy tính nạn nhân sẽ chuyển sang dạng “Zombie” và chạy mã độc. Ứng dụng bot tự động khởi động mỗi khi khởi động lại zombie. Trong giai đoạn kết nối, chương trình bot thiết lập kênh chỉ huy và điều khiển (C&C), đồng thời kết nối thây ma với máy chủ chỉ huy và điều khiển (C&C). Khi kênh C&C được thành lập, thây ma trở thành một phần của đội quân botnet của kẻ tấn công. Sau giai đoạn kết nối, các hoạt động điều khiển và chỉ huy botnet thực tế sẽ được bắt đầu. Quản trị viên bot sử dụng kênh C&C để phổ biến lệnh cho đội quân bot của mình. Các chương trình bot nhận và thực hiện các lệnh được gửi bởi botmaster. Kênh C&C cho phép quản trị viên bot điều khiển từ xa hành động của một số lượng lớn bot để tiến hành các hoạt động bất hợp pháp khác nhau.

Giai đoạn cuối là duy trì bot sống động và cập nhật. Trong giai đoạn này, các bot được lệnh tải xuống một tệp nhị phân được cập nhật. Bộ điều khiển bot có thể cần cập nhật mạng botnet của họ vì một số lý do. Ví dụ: họ có thể cần cập nhật nhị phân bot để tránh các kỹ thuật phát hiện hoặc họ có thể có ý định thêm chức năng mới vào đội quân bot của mình. Hơn nữa, đôi khi nhị phân được cập nhật di chuyển các bot sang một máy chủ C&C khác. Quá trình này được gọi là quá trình di chuyển máy chủ và nó rất hữu ích cho các quản trị viên bot để giữ cho mạng botnet của họ tồn tại. Các nhà quản trị thực vật cố gắng giữ cho các mạng botnet của họ vô hình và di động bằng cách sử dụng Dynamic DNS (DDNS), là một dịch vụ phân giải tạo điều kiện cho các cập nhật và thay đổi thường xuyên ở các vị trí máy chủ. Trong trường hợp cơ quan chức năng làm gián đoạn máy chủ C&C tại một địa chỉ IP nhất định, quản trị viên bot có thể dễ dàng thiết lập một phiên bản máy chủ C&C khác có cùng tên tại một địa chỉ IP khác. Các thay đổi về địa chỉ IP trong các máy chủ C&C truyền gần như ngay lập tức đến các bot do giá trị thời gian tồn tại ngắn (TTL) đối với các tên miền do nhà cung cấp DDNS đặt. Do đó, các bot sẽ di chuyển đến vị trí máy chủ C&C mới và vẫn tồn tại.

1.3. Tổng quan các kỹ thuật phát hiện và cơ chế phòng vệ Botnet

1.3.1. Phát hiện dựa trên chữ ký - Signature-based Detection

Kiến thức về các chữ ký hữu ích và hành vi của các mạng botnet hiện có rất hữu ích cho việc phát hiện botnet. Ví dụ, Snort [24] là một hệ thống phát hiện xâm nhập mã nguồn mở (IDS) theo dõi lưu lượng mạng để tìm ra các dấu hiệu xâm nhập. Giống như hầu hết các hệ thống IDS, Snort được cấu hình với một tập hợp các quy tắc hoặc chữ ký để ghi lại lưu lượng truy cập được cho là đáng ngờ [24]. Tuy nhiên, các kỹ thuật phát hiện dựa trên chữ ký có thể được sử dụng để phát hiện các mạng botnet đã biết. Vì vậy, giải pháp này không hữu ích cho các bot không xác định.

1.3.2. Phát hiện dựa trên điểm bất thường - Aomaly-based Detection

Các kỹ thuật phát hiện dựa trên sự bất thường cố gắng phát hiện các botnet dựa trên một số điểm bất thường về lưu lượng mạng như độ trễ mạng cao, lưu lượng lớn, lưu lượng truy cập trên các cổng bất thường và hành vi hệ thống bất thường có thể cho thấy sự hiện diện của bot độc hại trong mạng. Mặc dù các kỹ thuật phát hiện dị thường giải quyết được vấn đề phát hiện các mạng botnet không xác định, nhưng các vấn đề với phát hiện bất thường có thể bao gồm việc phát hiện mạng IRC có thể là một mạng botnet nhưng chưa được sử dụng để tấn công, do đó không có bất thường. Để giải quyết vấn đề này, Binkley và Singh đã đề xuất thuật toán không hiệu quả kết hợp phát hiện bất thường dựa trên TCP với mã thông báo IRC và thống kê tin nhắn IRC để tạo ra một hệ thống có thể phát hiện rõ ràng các botnet của khách hàng. Thuật toán này cũng có thể tiết lộ các máy chủ bot. Tuy nhiên, cách tiếp cận của Binkley có thể dễ dàng bị đánh bại chỉ bằng cách sử dụng một mật mã tầm thường để mã hóa các lệnh IRC. Năm 2007, Karasaridis và cộng sự [20] đã trình bày một thuật toán để phát hiện và mô tả đặc điểm của botnet bằng cách sử dụng phân tích thụ động dựa trên dữ liệu luồng trong lớp truyền tải. Thuật toán này có thể phát hiện các giao tiếp botnet được mã hóa. Nó giúp định lượng kích thước của mạng botnet, xác định và mô tả đặc điểm hoạt động của chúng mà không cần tham gia mạng botnet. Gần đây, Gu et al. đã đề xuất Botsniffer sử dụng tính năng phát hiện bất thường dựa trên mạng để xác

định các kênh botnet C&C trong mạng cục bộ. Botsniffer dựa trên quan sát rằng các bot trong cùng một mạng botnet có thể sẽ thể hiện sự đồng bộ hóa rất mạnh trong các phản hồi và hoạt động của chúng. Do đó, nó sử dụng một số thuật toán phân tích tương quan để phát hiện mối tương quan không gian-thời gian trong lưu lượng mạng với tỷ lệ dương tính giả rất thấp.

1.4. Tổng quan các ứng dụng học máy về phát hiện tấn công Botnet

Dựa theo nghiên cứu của Mustafa Alshamkhany và các công sự [21] được công bố vào năm 2020, chúng ta sẽ tìm hiểu về một số thuật toán machine learning dùng để phát hiện botnet.

Một số thuật toán machine learning hiện nay thường được sử dụng để phát hiện tấn công botnet là gồm các thuật toán: Naïve Bayes (NB), K-Nearest Neighbor (k-NN), Support Vector Machine (SVM), Decision Tree (DT)

1) Naive Bayes (NB) with Gaussian probabilities- một bộ phân loại xác suất sử dụng định lý Bayes và giả định sự độc lập có điều kiện giữa các đặc điểm khác nhau của tập dữ liệu. NB ước tính xác suất của lớp dựa trên tập huấn luyện.

2) K-Nearest Neighbor (k-NN) - một thuật toán không tham số được sử dụng để phân loại và hồi quy. Để dự đoán lớp, mô hình chỉ định lớp của mẫu thử dựa trên phần lớn k lân cận gần nhất của mẫu thử đã cho đó.

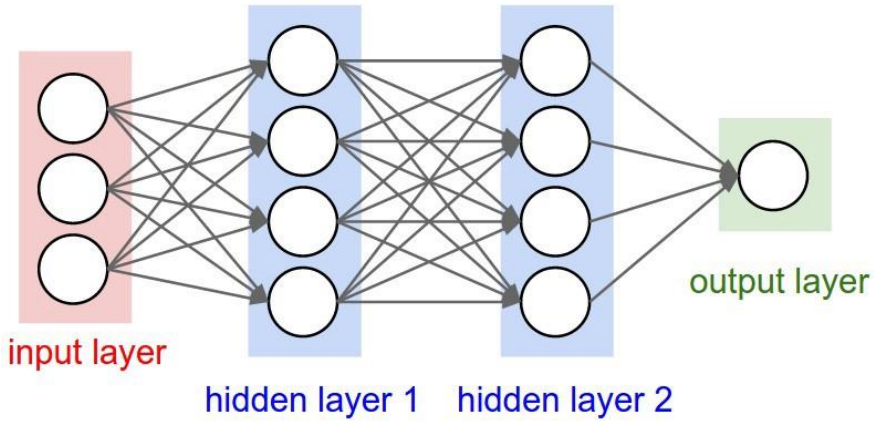
3) Support Vector Machine (SVM) với nhân phi tuyến, cụ thể là hàm cơ sở xuyên tâm (RBF) - tạo ra một ranh giới quyết định dựa trên các mẫu của các lớp khác nhau. Hình dạng của ranh giới quyết định được tạo dựa trên chức năng hạt nhân được sử dụng và các siêu tham số chính như C điều khiển sự cân bằng giữa độ trơn của ranh giới quyết định và tính đúng đắn của phân loại và gamma xác định ảnh hưởng của các điểm dữ liệu 'phân bố trên hình dạng của ranh giới quyết định.

4) Decision Tree (DT) - một mô hình phân loại dạng cây trong đó mỗi nút trong cây chỉ định một bài kiểm tra trên một đối tượng địa lý và mỗi nhánh giảm dần từ nút đó tương ứng với một trong các giá trị có thể có cho đối tượng địa lý đó.

1.5. Mạng nơ-ron và Deep Learning

1.5.1. Mạng nơ-ron

Neural network [22] hay còn gọi là mạng nơ-ron được xây dựng dựa trên mạng nơ-ron sinh học. Nó là một mạng lưới gồm các nút được kết nối với nhau – gọi là nơ-ron và các cạnh nối chúng lại với nhau. Nhiệm vụ chính của mạng nơ-ron là nhận vào một tập đầu vào, thực hiện các phép tính toán phức tạp và sau đó sử dụng đầu ra để giải quyết vấn đề. Mạng nơ-ron là một mạng có cấu trúc và nhiều lớp (layer). Một mạng nơ-ron có 3 lớp chính là: input, hidden và output.



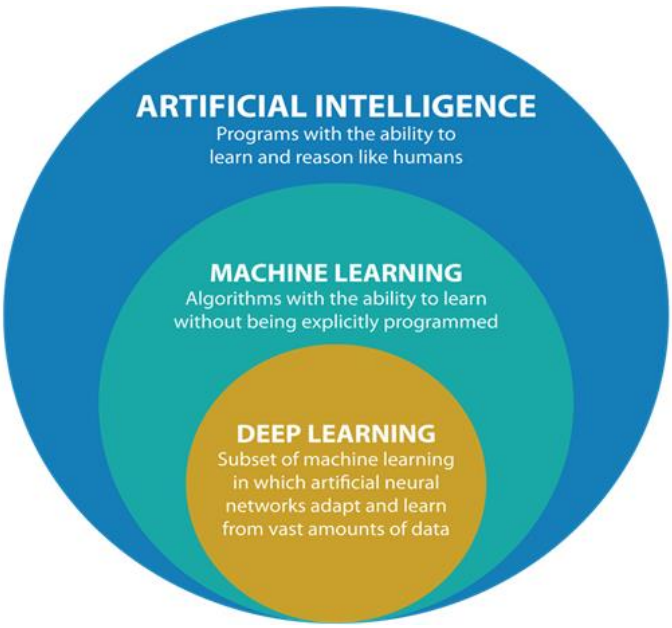
Hình 1.6: Mạng nơ-ron với hai lớp hidden [22]

Mỗi kết nối giống như khớp thần kinh trong não sinh học, có thể truyền tín hiệu đến các tế bào thần kinh khác. Một nơ-ron nhận một tín hiệu sau đó xử lý nó và có thể phát tín hiệu cho các nơ-ron khác kết nối với nó. "Tín hiệu" tại một kết nối (hay cạnh) là một số thực và đầu ra của mỗi nơ-ron được tính bằng một số hàm phi tuyến tính (còn được gọi là **hàm kích hoạt** hay **activation funtion**) của tổng các đầu vào của nó. Các nơ-ron và các cạnh thường có trọng số được điều chỉnh khi quá trình học tập diễn ra (trọng số này được gọi là **bias** đối với các nơ-ron và **weight** đối với các cạnh). Trọng số làm tăng hoặc giảm cường độ của tín hiệu tại một cạnh. Các nơ-ron có thể có ngưỡng sao cho tín hiệu được gửi đi chỉ khi tín hiệu tổng hợp vượt qua ngưỡng đó, khi đó các output của lớp này sẽ là input của lớp phía sau. Thông qua việc

lặp lại các bước trên, mạng nơron học thông qua nhiều lớp và các nơ-ron rồi sau đó kết hợp lại ở lớp cuối cùng để cho ra một dự đoán.

1.5.2. Deep Learning

Khi mà khả năng tính toán của các máy tính ngày càng được nâng lên một tầm cao mới và lượng dữ liệu ngày càng khổng lồ, Machine Learning (ML) đã tiến thêm một bước dài và một lĩnh vực mới được ra đời gọi là Deep Learning. ML [23] là một tập hợp con của AI cung cấp cho máy tính khả năng tự động học hỏi dựa trên dữ liệu đưa vào mà không cần phải được lập trình cụ thể. Do đó, học máy là một phương pháp giúp máy móc giải quyết vấn đề bằng cách đạt được khả năng suy nghĩ. Trong khi đó Deep Learning [1] lại tập trung giải quyết các vấn đề liên quan đến mạng thần kinh nhân tạo nhằm nâng cấp các công nghệ như nhận diện giọng nói, thị giác máy tính và xử lý ngôn ngữ tự nhiên. Chỉ trong thời gian ngắn, Deep Learning đã giúp máy tính làm được rất nhiều công việc phức tạp như: chỉnh màu cho ảnh đen trắng, thêm âm thanh vào phim câm, dịch máy tự động, phân loại các đối tượng trong ảnh, tạo chữ viết tay tự động, tạo phụ đề cho hình ảnh,... Ngoài ra còn rất nhiều lĩnh vực khác đã được đơn giản hóa và nâng cao hiệu quả hoạt động với sự trợ giúp của Deep Learning.



Hình 1.7: Mối liên hệ giữa AI, Machine Learning và Deep Learning [1]

1.6. Tổng quan về Representation Learning

Nhiều tác vụ xử lý thông tin có thể rất dễ hoặc rất khó tùy thuộc vào cách thông tin được biểu diễn. Đây là nguyên tắc chung áp dụng cho cuộc sống hàng ngày, khoa học máy tính nói chung và học máy. Ví dụ, một người sẽ đơn giản là chia 210 cho 6 bằng cách sử dụng phép chia dài. Nhiệm vụ trở nên kém đơn giản hơn đáng kể nếu thay vào đó nó được đặt ra bằng cách sử dụng biểu diễn số La Mã của các con số. Hầu hết những người hiện đại được yêu cầu chia CCX cho VI sẽ bắt đầu bằng cách chuyển đổi các số sang biểu diễn chữ số Ả Rập, cho phép các quy trình phân chia dài sử dụng hệ thống giá trị vị trí. Cụ thể hơn, chúng ta có thể định lượng thời gian chạy tiệm cận của các hoạt động khác nhau bằng cách sử dụng các biểu diễn phù hợp hoặc không phù hợp. Ví dụ: chèn một số vào vị trí chính xác trong danh sách các số đã được sắp xếp là một phép toán O (n) nếu danh sách được biểu thị dưới dạng danh sách liên kết, nhưng chỉ O (log n) nếu danh sách được biểu thị dưới dạng màu đỏ-đen cây.

Trong bối cảnh của học máy, một biểu diễn tốt là một biểu diễn giúp cho nhiệm vụ học tập tiếp theo trở nên dễ dàng hơn. Việc lựa chọn biểu diễn thường sẽ phụ thuộc vào sự lựa chọn của nhiệm vụ học tập tiếp theo. Hầu hết các vấn đề về học biểu diễn đều phải đối mặt với sự cân bằng giữa việc lưu giữ càng nhiều thông tin về đầu vào càng tốt và đạt được các đặc tính tốt đẹp (chẳng hạn như tính độc lập).

Vậy học đại diện hay học biểu diễn (Representation Learning) [24] *là một tập hợp các kỹ thuật cho phép hệ thống tự động khám phá các biểu diễn cần thiết để phát hiện hoặc phân loại tính năng từ dữ liệu thô.*

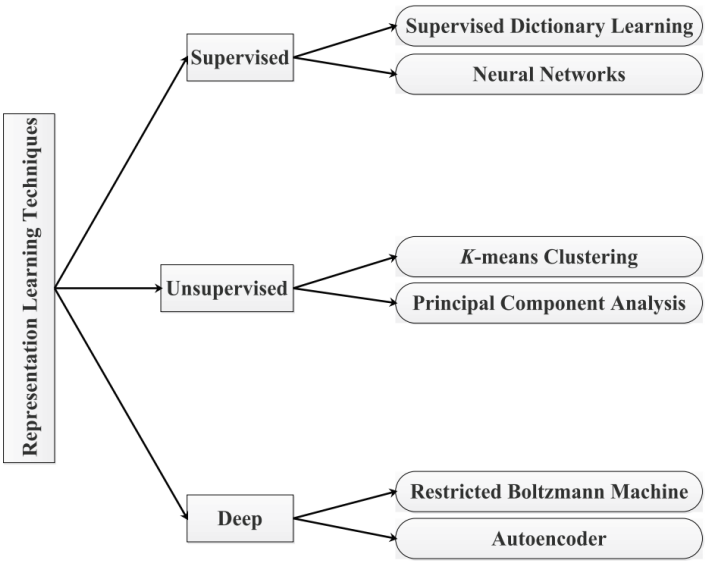
Học đại diện đặc biệt thú vị vì nó cung cấp một cách để thực hiện học không giám sát và bán giám sát. Thường có một lượng rất lớn dữ liệu đào tạo không được gắn nhãn và tương đối ít dữ liệu đào tạo được gắn nhãn. Việc đào tạo với các kỹ thuật học tập có giám sát trên tập hợp con được dán nhãn thường dẫn đến tình trạng quá mức nghiêm trọng.

Học tập bán giám sát mang lại cơ hội giải quyết vấn đề quá tải này bằng cách học hỏi từ dữ liệu không được gắn nhãn. Cụ thể, chúng ta có thể tìm hiểu các cách

biểu diễn tốt cho dữ liệu chưa được gắn nhãn và sau đó sử dụng các cách biểu diễn này để giải quyết nhiệm vụ học tập có giám sát.

1.7. Các kỹ thuật Representation Learning

Tài liệu [9] trình bày về các kỹ thuật khác nhau từ các tài liệu gần đây. Các kỹ thuật này dựa trên các thuật toán RL và được thiết kế cho các ứng dụng an ninh mạng khác nhau. Các kỹ thuật này được phân bổ thành ba loại lớn bao gồm: kiến trúc có giám sát, không được giám sát và kiến trúc sâu, như thể hiện trong Hình 1.8. Trong mỗi loại, các kỹ thuật được thảo luận chi tiết với ưu và nhược điểm trong các phần phụ sau:



Hình 1.8: Các kỹ thuật Representation Learning

Supervised Dictionary Learning: Trong những năm gần đây, các kỹ thuật dựa trên việc học đại diện có giám sát được sử dụng để điều tra các vấn đề liên quan đến bảo mật trong các hệ thống an ninh mạng. Đánh giá về những tiến bộ gần đây trong kỹ thuật học máy cùng với các phương pháp tính toán mềm để bảo mật các hệ thống dựa trên Công nghệ Thông tin và Truyền thông (ICT) được trình bày trong Tài liệu tham khảo [17]. Trong bài đánh giá này, các vấn đề bảo mật khác nhau liên quan đến Internet và các dịch vụ của nó đối với các hệ thống CNTT-TT được phân tích và các xu hướng của việc học từ điển có giám sát cùng với các phương pháp tính toán mềm

được nghiên cứu để đảm bảo an toàn cho các hệ thống CNTT-TT. Một cuộc khảo sát khác về các phương pháp học máy để bảo vệ các hệ thống sản xuất không gian mạng chống lại các cuộc tấn công mạng đã được trình bày trong tài liệu [17]. Các hệ thống sản xuất không gian mạng mô tả khái niệm kết hợp các hệ thống CNTT-TT khác nhau, ví dụ: IoT, điện toán đám mây, mạng cảm biến và nền tảng máy học. Tuy nhiên, các hệ thống này phải đối mặt với nhiều cuộc tấn công mạng mạnh mẽ khác nhau, bao gồm cả Stuxnet và các cuộc tấn công xay xát được điều khiển bằng số máy tính. Trong cuộc khảo sát này, thông qua các thí nghiệm gợi ý rằng các thuật toán học biểu diễn, đặc biệt là thuật toán học từ điển có giám sát, có thể phát hiện các cuộc tấn công mạng với tỷ lệ chính xác cao nếu dữ liệu vật lý được nghiên cứu đúng cách. Một cuộc khảo sát về các hệ thống vật lý mạng công nghiệp cùng với các cơ chế phát hiện tấn công và kiểm soát an ninh của chúng đã được trình bày trong Tài liệu tham khảo [31]. Cuộc khảo sát này thảo luận về các cuộc tấn công DoS, lừa dối và phát lại cùng với các điểm yếu của chúng. Hơn nữa, các phát triển khác nhau về phát hiện tấn công được xem xét từ góc độ phát hiện. Những phát triển này có thể được sử dụng để phát triển các thuật toán học từ điển có giám sát nhằm bảo vệ các hệ thống an ninh mạng công nghiệp.

Artificial Neural Networks (ANN) được lấy cảm hứng từ mạng nơron sinh học để thực hiện các nhiệm vụ khác nhau. Các nhiệm vụ được thực hiện thông qua việc học dựa trên các ví dụ được cung cấp. Trong quá trình học, không có quy định cụ thể nào. ANN bao gồm các đơn vị nhỏ khác nhau được gọi là tế bào thần kinh nhân tạo, nơi mỗi tế bào thần kinh có thể giao tiếp với các tế bào thần kinh được kết nối khác. Các kết nối giữa các nơ-ron được gọi là các cạnh. Các tế bào thần kinh và các cạnh có trọng lượng cụ thể tăng hoặc giảm khi quá trình học tập diễn ra. Nói chung, các tế bào thần kinh được phân phối thành nhiều lớp để thực hiện các hoạt động khác nhau trên dữ liệu đầu vào. Ngày nay, ANN là một ví dụ tiêu chuẩn và phổ biến trong các lĩnh vực xử lý dữ liệu khác nhau, ví dụ: thị giác máy tính, nhận dạng mẫu, xử lý giọng nói, mạng xã hội, phân loại dữ liệu, trò chơi điện tử, theo dõi đối tượng, phân tích dữ liệu lớn và an ninh mạng.

K-mean clustering là một kỹ thuật để phân tích và phân vùng dữ liệu đầu vào (tức là các quan sát) thành k-cluster. Phân cụm k-mean được coi là một bài toán phức tạp về mặt tính toán; tuy nhiên, các thuật toán heuristic hiệu quả có thể được triển khai để chuyển đổi nhanh chóng. Các cụm được hình thành có thể có hình dạng và kích thước khác nhau. Kỹ thuật này khác với bài toán k-gần nhất trong miền học máy. Một thuật toán lân cận n có thể được áp dụng trên một cụm được hình thành từ kỹ thuật phân cụm k-mean để phân loại dữ liệu mới được nhập trong các cụm hiện có. Phân cụm k-means có thể được áp dụng trong các ứng dụng khác nhau bao gồm thị giác máy tính, thiên văn học, tiếp thị và nông nghiệp.

Principal Component Analysis (PCA) là một thủ tục để chuyển đổi một tập hợp các quan sát thành một tập hợp các biến không tương quan tuyến tính được gọi là các thành phần chính. Trong các thành phần chính được chuyển đổi, phương sai lớn nhất có thể được giữ bởi thành phần đầu tiên và mỗi thành phần tiếp theo giữ giá trị phương sai cao nhất. PCA là hình thức đơn giản nhất của phân tích dựa trên vectơ eigen và được sử dụng như một công cụ để phân tích dữ liệu và đưa ra các mô hình dự đoán. PCA khá phổ biến trong lĩnh vực khoa học thần kinh và tài chính định lượng.

Restricted Boltzmann Machine (RBM) dựa trên ANN tổng hợp ngẫu nhiên. RBM có thể hoạt động ở cả chế độ có giám sát và không được giám sát. Trong RBM, các nút từ mỗi nhóm đơn vị có thể chia sẻ một kết nối đối xứng. Tuy nhiên, không có kết nối nào giữa các nút trong cùng một nhóm đơn vị. Các RBM có thể được sử dụng để tạo mạng học sâu bằng cách xếp chồng nhiều RBM. Các mạng học sâu được hình thành sau này có thể được điều chỉnh bằng cách sử dụng các mạng lan truyền ngược và giảm độ dốc. Các RBM có thể được sử dụng trong các ứng dụng thực tế khác nhau, ví dụ: học tính năng, phân loại dữ liệu và giảm kích thước của dữ liệu được thu thập.

Autoencoder dựa trên ANN và hoạt động theo cách không được giám sát để học dữ liệu mã hóa hiệu quả. Nó được sử dụng để học các biểu diễn của dữ liệu đầu vào cho mục đích nén. Nén được áp dụng bằng cách giảm kích thước của dữ liệu được cung cấp. Dữ liệu đầu vào được chuyển đổi thành một đoạn mã ngắn mà sau này không được nén để khớp với dữ liệu đầu vào ban đầu. Các mã tự động có thể

được sử dụng ở dạng xếp chồng lên nhau trong các ứng dụng nhất định, ví dụ: nhận dạng hình ảnh. Ở định dạng xếp chồng lên nhau, các lớp dưới học và mã hóa các tính năng dễ dàng trong khi các lớp trên phân tích kết quả đầu ra của các lớp trước và mã hóa các tính năng bị thiếu, khó hoặc ẩn. Quá trình này tiếp tục cho đến khi toàn bộ dữ liệu được mã hóa.

1.8. Các trình nghiên cứu liên quan.

1.8.1. Các công trình nghiên cứu trong nước

Năm 2018, Nguyễn Trọng Hưng và các cộng sự [5] đã công bố nghiên cứu *"Phát hiện botnet dựa trên phân loại tên miền sử dụng các kỹ thuật học máy"*. Trong những năm gần đây, các botnet đã trở thành một trong các nguy cơ gây mất an toàn thông tin hàng đầu do chúng không ngừng phát triển về cả quy mô và mức độ tinh vi. Nhiều giải pháp phát hiện botnet, như dựa trên honeynet, hoặc IDS đã được đề xuất. Tuy nhiên, các giải pháp dựa trên IDS sử dụng chữ ký tỏ ra kém hiệu quả do botnet được trang bị khả năng tự cập nhật mã và nhiều kỹ thuật lẩn tránh tinh vi. Kết quả nhiều nghiên cứu cho thấy các phương pháp phát hiện botnet dựa trên bất thường tỏ ra hiệu quả hơn. Bài báo này giới thiệu mô hình và khảo sát hiệu quả phát hiện botnet dựa trên phân loại các tên miền độc hại của botnet và các tên miền bình thường sử dụng các kỹ thuật học máy. Kết quả thử nghiệm cho thấy các thuật toán học máy có giám sát có thể sử dụng hiệu quả trong phát hiện botnet dựa trên phân loại tên miền và thuật toán học máy Rừng ngẫu nhiên cho độ chính xác phát hiện cao nhất.

Năm 2019, Vũ Xuân Hạnh và Hoàng Xuân Dậu [6] đã công bố nghiên cứu về *"Phát hiện dga botnet sử dụng kết hợp nhiều nhóm đặc trưng phân loại tên miền"*. Trong những năm gần đây, các botnet đã trở thành một trong các nguy cơ gây mất an toàn thông tin hàng đầu do chúng không ngừng phát triển về cả quy mô và mức độ tinh vi. Nhiều dạng botnet sử dụng kỹ thuật DGA để sinh và đăng ký nhiều tên miền ngẫu nhiên khác nhau cho máy chủ lệnh và điều khiển (C&C) của chúng nhằm chống lại việc bị kiểm soát. Việc phân tích phát hiện các tên miền truy vấn hệ thống DNS có thể giúp phát hiện các hoạt động của botnet do các bot tồn tại trong hệ thống mạng cũng liên tục sử dụng kỹ thuật DGA để sinh tên miền và truy vấn hệ thống DNS để

tìm địa chỉ IP của các máy chủ C&C. Các mô hình phát hiện DGA botnet dựa trên phân phân loại tên miền do botnet sinh tự động với tên miền bình thường đã được nghiên cứu, đề xuất. Bài báo này đề xuất bổ sung một nhóm gồm 4 đặc trưng phân loại tên miền mới kết hợp với 3 nhóm gồm 18 đặc trưng đã có nhằm cải thiện hiệu quả phát hiện của mô hình phát hiện DGA botnet dựa trên học máy. Các kết quả thử nghiệm cho thấy, nhóm đặc trưng phân loại mới giúp tăng đáng kể độ chính xác phát hiện và giảm tỷ lệ phát hiện nhầm. Trong bài báo này đề xuất bổ sung 4 đặc trưng phân loại mới kết hợp với 18 đặc trưng đã có để hình thành 4 nhóm đặc trưng phân loại tên miền nhằm nâng cao độ chính xác phát hiện và giảm tỷ lệ phát hiện sai. Các kết quả thử nghiệm trên tập dữ liệu mở rộng khẳng định, các đặc trưng bổ sung giúp cải thiện đáng kể hiệu quả phát hiện. Đặc biệt, trong trường hợp bổ sung cả 4 đặc trưng tên miền mới, độ chính xác phân loại chung ACC tăng 1.16%, độ chính xác dương Pre tăng 1.79% và tỷ lệ âm tính giả giảm 1.78%.

Năm 2019, Hoàng Xuân Dậu và các cộng sự [7] đã công bố nghiên cứu *"Phát hiện botnet dựa trên học máy sử dụng dữ liệu truy vấn DNS: Phân tích ảnh hưởng của các đặc trưng huấn luyện".* Bài báo đưa ra mô hình phát hiện botnet dựa trên học máy sử dụng dữ liệu tên miền trích xuất từ dữ liệu truy vấn DNS cho kết quả phát hiện DGA botnet khả quan. Mô hình này loại bỏ phần tên miền cấp cao nhất trong mỗi tên miền truy vấn và sử dụng 18 đặc trưng để vector hoá mỗi tên miền cho khâu huấn luyện và phát hiện. Trong bài báo này, các tác giả tập trung phân tích ảnh hưởng của một số yếu tố lên hiệu quả của mô hình phát hiện. Các yếu tố được xem xét phân tích bao gồm (1) vấn đề loại bỏ hoặc giữ nguyên phần tên miền cấp cao nhất và (2) ảnh hưởng của các nhóm đặc trưng huấn luyện. Bài báo này khảo sát ảnh hưởng của hai yếu tố đến độ chính xác của mô hình phát hiện DGA botnet dựa trên học máy sử dụng dữ liệu truy vấn DNS. Các yếu tố được khảo sát bao gồm (1) việc sử dụng hay loại bỏ tên miền cấp cao nhất và (2) ảnh hưởng của các nhóm đặc trưng huấn luyện lên độ chính xác phát hiện. Các kết quả trên 4 kịch bản thử nghiệm cho thấy việc sử dụng tên miền cấp cao nhất làm tăng đáng kể độ chính xác phân loại chung (tăng 1.21%). Trong 3 nhóm đặc trưng, nhóm 8 đặc trưng 2-gram có ảnh hưởng lớn nhất

đến độ chính xác phân loại chung, xếp sau là nhóm 3-gram và cuối cùng là nhóm các đặc trưng nguyên âm. Kết quả thử nghiệm cũng cho thấy từng đặc trưng 2-gram có ảnh hưởng không đáng kể đến độ chính xác phân loại chung. Dù vậy, tập hợp 16 đặc trưng 2-gram và 3-gram có ảnh hưởng quyết định đến độ chính xác phân loại chung.

1.8.2. Các công trình nghiên cứu trên thế giới

Năm 2009, Maryam Feily và các công sự [19] đã công bố nghiên cứu *"A Survey of Botnet and Botnet Detection",* bài báo này là một cuộc khảo sát về những tiến bộ gần đây trong nghiên cứu mạng botnet. Cuộc khảo sát phân loại nghiên cứu botnet thành ba lĩnh vực: hiểu botnet, phát hiện và theo dõi botnet, cuối cùng là bảo vệ chống lại botnet. Mặc dù mạng botnet phổ biến rộng rãi, nhưng nghiên cứu và các giải pháp cho mạng botnet vẫn còn sơ khai. Bài báo cũng tóm tắt các nghiên cứu hiện có và đề xuất trong tương lai hướng nghiên cứu mạng botnet. Bài báo này được sử dụng như cơ sở lý thuyết về botnet cho luận văn này.

Năm 2011, Gregory Fedynyshyn và các cộng sự [8] đã công bố nghiên cứu *"Detection and Classification of Different Botnet C&C Channels"*. Không giống như các loại phần mềm độc hại khác, botnet được đặc trưng bởi các kênh chỉ huy và kiểm soát (C&C), qua đó cơ quan quản lý trung tâm, quản trị viên bot, có thể sử dụng máy tính bị nhiễm để thực hiện các hoạt động độc hại. Do các mạng botnet có khả năng gây ra thiệt hại, việc phát hiện và giảm thiểu các mối đe dọa từ botnet là bắt buộc. Trong bài báo này, nhóm tác giả trình bày một phương pháp dựa trên máy chủ để phát hiện và phân biệt các loại lây nhiễm botnet khác nhau dựa trên kiểu C&C của chúng, ví dụ: dựa trên IRCbased, dựa trên HTTP hoặc peer-to-peer (P2P). Khả năng phát hiện và phân loại các kênh C&C của mạng botnet cho thấy có sự giống nhau cố hữu trong cấu trúc C&C cho các loại bot khác nhau và đặc tính mạng của lưu lượng botnet C&C vốn dĩ khác với lưu lượng mạng hợp pháp. Hiệu suất tốt nhất của hệ thống phát hiện của chúng tôi có độ chính xác tổng thể là 0,929 và tỷ lệ dương tính giả là 0,078.

Năm 2019, Riaz Ullah Khan và các cộng sự [25] đã công bố nghiên cứu về *"An Adaptive Multi-Layer Botnet Detection Technique Using Machine Learning*

Classifiers”. Bài báo này trình bày một kỹ thuật nhiều lớp để phân loại (lưu lượng mạng botnet P2P và lưu lượng không phải P2P) và xác định các botnet bằng cách áp dụng bộ phân loại học máy. Lưu lượng mạng được lọc ra bằng cách khai thác các tính năng mạng như lọc cổng, truy vấn DNS và đếm luồng. Hơn nữa, cơ chế xác thực chéo 10 lần đã được triển khai để so sánh và phân tích hiệu suất của phiên không có bộ lọc và tính năng phiên được lọc sau khi phân loại. Giá trị ngưỡng được sử dụng để phân tích và so sánh kết quả. Trên thực tế, ngưỡng là giá trị tối thiểu hoặc tối đa (được phát triển cho một đặc tính, tính năng hoặc biến) dùng làm điểm chuẩn để so sánh hoặc hướng dẫn và có thể yêu cầu xem xét toàn bộ hệ thống. Tốc độ nhận dạng của lưu lượng truy cập internet có liên quan đến tập hợp các ngưỡng, đếm luồng và nhận dạng cổng. Tỷ lệ phát hiện của botnet là 99%, 98,9%, 98,7%, v.v. Tỷ lệ báo động giả của lưu lượng truy cập bình thường lần lượt là 9%, 7,6%, 3% cho các ngưỡng 1, 2 và 3. Tỷ lệ báo động sai của lưu lượng truy cập bình thường cho các ngưỡng 4, 5, 6, 7 và 8 là 1%. Trong thử nghiệm này, các tác giả sử dụng các ngưỡng khác nhau (1, 2, 3, 4, 5, 6, 7, 8) để xác định lưu lượng truy cập internet. Kết quả được thể hiện trong Hình 16, tức là khi ngưỡng đạt đến 3, tỷ lệ phát hiện giảm xuống nhanh chóng. Với sự gia tăng của giá trị ngưỡng lớn hơn 3 và tỷ lệ báo động sai không thay đổi. Tuy nhiên, khi ngưỡng quá nhỏ trong trường hợp của chúng tôi là 1, do đó tỷ lệ báo động sai sẽ tăng lên và tỷ lệ phát hiện không thay đổi nhiều. Do đó, theo kết quả kiểm tra, chúng tôi đặt ngưỡng 3 trong toàn bộ quá trình phát hiện lưu lượng P2P. Kết quả cho thấy việc phân loại cổng nổi tiếng, DNS và lọc đếm luồng không chỉ tốt hơn P2P mà không cần lọc, mà tỷ lệ báo động sai của lưu lượng thông thường cũng giảm hơn nữa. Hơn nữa, các thử nghiệm của chúng tôi cũng chứng minh rằng độ chính xác của khuôn khổ được đề xuất đã được cải thiện lên đến 99%, tuy nhiên, với cái giá là báo cáo sai về các tệp lành tính như botnet cũng như báo cáo sai về botnet là lành tính. Xem xét yếu tố liệu các tệp lành tính có gửi yêu cầu tìm kiếm một cách nhất quán để các tệp lành tính có thể bị báo cáo là botnet hay không. Ngoài ra, người ta cảm thấy thất vọng rằng độ chính xác có thể được cải thiện bằng cách tăng kỷ nguyên của các thuật toán học sâu với chi phí thực hiện nhiều hơn. Dựa trên các đặc điểm của phiên,

bài báo này sử dụng thuật toán phân loại Cây quyết định để phân loại lưu lượng truy cập mạng botnet và P2P thông thường. Độ sâu của thuật toán Cây quyết định được đặt thành 8 và cây phân loại được đặt thành 100. Tỷ lệ phát hiện Cây quyết định là 98,7% đối với ngưỡng 3 của tỷ lệ dương tính giả.

Năm 2020, Ankit Bansal và các cộng sự [2] đã công bố nghiên cứu *"A Comparative Analysis of Machine Learning Techniques for Botnet Detection",* Trong bài báo này, các tác giả so sánh ba kỹ thuật khác nhau để phát hiện botnet với mỗi kỹ thuật có các trường hợp sử dụng riêng. Kết quả phát hiện các phương pháp đã được xác minh bằng cách sử dụng Bộ dữ liệu phát hiện xâm nhập ISCX và Bộ dữ liệu CTU-13. Các kỹ thuật được đề cập đến trong bài báo bao gồm các kỹ thuật Clustering Method, Neural Network Method, Recurrent Neural Network Method. Phương pháp Neural Network vượt trội hơn tất cả các phương pháp phát hiện khác với hiệu suất của nó như được hiển thị trong Hình 1.9 bên dưới.

Method	F1 Score	Sensitivity	Specificity	Precision	Accuracy
Clustering	0.8545	0.8447	0.9922	0.8645	0.9839
NN	0.8897	0.8570	0.9305	0.9250	0.8938
RNN	0.8044	0.6953	0.9666	0.9541	0.8309

Hình 1.9: Kết quả so sánh hai phương pháp với các độ đo [2]

Phương pháp này yêu cầu một số lượng lớn các điểm dữ liệu và cũng cần sự phân bổ đồng đều của cả luồng độc hại và luồng bình thường để huấn luyện hiệu quả. Trong các tình huống thực tế, không thể có được số lượng luồng độc hại và luồng bình thường bằng nhau.

Method	Dataset Size	Training Time	Feature Extraction
Clustering	Moderate	Low	Manual
NN	Large	Moderate	Manual
RNN	Very Large	High	Automatic

Hình 1.10: So sánh các phương pháp phát hiện [2]

Hình 1.10 bên trên tóm tắt các yêu cầu của từng phương pháp phát hiện. Phương pháp Clustering không yêu cầu một tập dữ liệu khổng lồ và phân phối đồng đều các luồng trong cả hai danh mục mặc dù nó có hiệu suất thấp hơn. RNN có điểm đáng khen là không yêu cầu kỹ thuật tính năng thủ công nhưng để đào tạo mô hình, nó yêu cầu một tập dữ liệu lớn hơn. Các CTU Dataset đã không thể cung cấp đủ ví dụ huấn luyện để đào tạo mô hình này một cách hiệu quả. Hơn nữa, hiệu suất của RNN có thể được cải thiện với một tập dữ liệu lớn hơn. Bài báo này tập trung chủ yếu vào các phương pháp phát hiện botnet, đây cũng là một trong những cơ sở nghiên cứu cho đề tài.

CHƯƠNG 2. XÂY DỰNG MÔ HÌNH PHÁT HIỆN TẤN CÔNG BOTNET

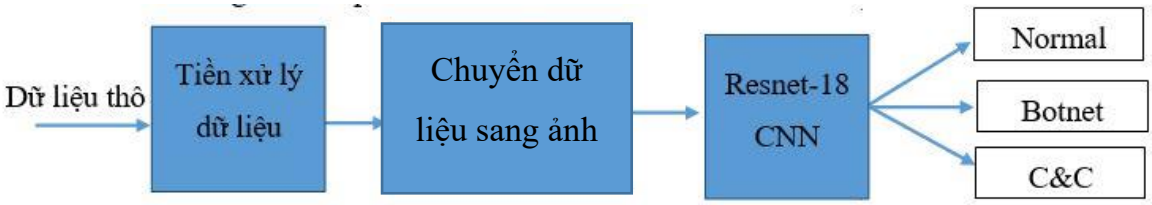
2.1. Thiết kế mô hình

Mô hình được xây dựng và làm việc như bộ phân loại. Mô hình nhận dữ liệu từ mạng bao gồm dữ liệu của các ứng dụng thông thường và có dữ liệu tấn công botnet. Mô hình chỉ ra chính xác dữ liệu tấn công botnet.

Hiện nay các mô hình học RL sử dụng mạng neural học sâu đã cho kết quả rất khả quan trong lĩnh vực thị giác máy tính, áp dụng thành công trong phân loại ảnh và video. Áp dụng các mô hình này vào lĩnh vực an toàn thông tin mà cụ thể là phát hiện tấn công botnet để cải thiện khả năng phát hiện cho các hệ thống phát hiện xâm nhập. Tuy nhiên dữ liệu tấn công mạng khác với dữ liệu multimedia trên các mô hình thị giác máy tính. Cần có phương pháp tạo sự tương thích để có thể áp dụng được.

Mô hình được xây dựng sử dụng công nghệ DL và dùng bộ phân loại CNN học có giám sát dựa vào nhãn để phân loại phát hiện nhận dạng dữ liệu mạng nào là bình thường, dữ liệu nào là tấn công Botnet, dữ liệu nào là tấn công C&C

Mô hình gồm có 3 phần chính:



Hình 2.1: Thiết kế chi tiết mô hình

2.2. Bộ dữ liệu

CTU – 13 [32] là tập dữ liệu về lưu lượng tấn công mạng botnet đã được thu thập tại trường đại học CTU, Cộng Hòa Séc vào năm 2011. Tập dữ liệu bao gồm 13 bản chụp các mẫu botnet khác nhau được chụp lại trong môi trường mạng thực tế và đã được gán nhãn. Trong mỗi bản chụp thu thập bao gồm một lượng lớn lưu lượng bị tấn công botnet, lưu lượng thông thường (normal) và lưu lượng nền (background). Ở

mỗi bản chụp, tác giả đã thực thi một phần mềm độc hại cụ thể, các phần mềm này sử dụng một số giao thức và thực hiện các hành động khác nhau để phân tích đặc điểm của từng bản chụp. Bảng 2.1 cho thấy những đặc điểm của các kịch bản mạng botnet trong bộ dữ liệu.

Bảng 2.1: Đặc điểm các kịch bản mạng Botnet trong bộ dữ liệu CTU-13

Id	IRC	SPAM	CF	PS	DdoS	FF	P2P	US	HTTP	Ghi chú
1	✓	✓	✓							
2	✓	✓	✓							
3	✓			✓				✓		
4	✓				✓			✓		UDP và ICMP DdoS
5		✓		✓					✓	Scan web proxies.
6				✓						Proprietary C&C. RDP.
7									✓	Chinese hosts
8				✓						Proprietary C&C. Net-BIOS, STUN.
9	✓	✓	✓	✓						
10	✓				✓			✓		UDP DdoS
11	✓				✓			✓		ICMP DdoS
12							✓			Synchronization
13		✓		✓					✓	Captcha. Web mail.

Mỗi bản chụp được ghi lại trong một tệp pcap (Packet Capture Data: dùng để thu thập và ghi dữ liệu gói từ mạng) chứa tất cả các gói (package) của ba loại lưu lượng truy cập. Các tệp pcap này được xử lý để thu thập các loại thông tin khác nhau, chẳng hạn như NetFlows, WebLogs, v.v...

Mối quan hệ giữa thời lượng bản chụp, số lượng gói tin, số lượng NetFlows và kích thước của các tệp pcap, định danh của các Bot được hiển thị trong Bảng 2.3.

Bảng 2.2: Lượng dữ liệu trên mỗi bản chụp mạng botnet

Id	Thời lượng (giờ)	#Packets	#NetFlows	Size	Bot	#Bots
1	6.15	71,971,482	2,824,637	52GB	Neris	1
2	4.21	71,851,300	1,808,123	60GB	Neris	1
3	66.85	167,730,395	4,710,639	121GB	Rbot	1
4	4.21	62,089,135	1,127,077	53GB	Rbot	1
5	11.63	4,481,167	129,833	37.6GB	Virut	1

6	2.18	38,764,357	558,920	30GB	Menti	1
7	0.38	7,467,139	114,078	5.8GB	Sogou	1
8	19.5	155,207,799	2,954,231	123GB	Murlo	1
9	5.18	115,415,321	2,753,885	94GB	Neris	10
10	4.75	90,389,782	1,309,792	73GB	Rbot	10
11	0.26	6,337,202	107,252	5.2GB	Rbot	3
12	1.21	13,212,268	325,472	8.3GB	NSIS.ay	3
13	16.36	50,888,256	1,925,150	34GB	Virut	1

Tập dữ liệu CTU-13 được các tác giả phân tích và gán nhãn cho từng tình bản chụp theo cách thủ công. Quá trình gán nhãn đã được thực hiện bên trong các tệp. Bảng 2.4 cho thấy mối quan hệ giữa số lượng nhãn cho Background, Botnet, C&C và Normal trên mỗi bản.

Bảng 2.3: Phân phối nhãn trong NetFlows cho mỗi trường hợp trong tập dữ liệu

Scen.	Total Flows	Botnet Flows	Normal Flows	C&C Flows	Background Flows
1	2,824,636	39,933(1.41%)	30,387(1.07%)	1,026(0.03%)	2,753,290(97.47%)
2	1,808,122	18,839(1.04%)	9,120(0.5%)	2,102(0.11%)	1,778,061(98.33%)
3	4,710,638	26,759(0.56%)	116,887(2.48%)	63(0.001%)	4,566,929(96.94%)
4	1,121,076	1,719(0.15%)	25,268(2.25%)	49(0.004%)	1,094,040(97.58%)
5	129,832	695(0.53%)	4,679(3.6%)	206(1.15%)	124,252(95.7%)
6	558,919	4,431(0.79%)	7,494(1.34%)	199(0.03%)	546,795(97.83%)
7	114,077	37(0.03%)	1,677(1.47%)	26(0.02%)	112,337(98.47%)
8	2,954,230	5,052(0.17%)	72,822(2.46%)	1,074(2.4%)	2,875,282(97.32%)
9	2,753,884	179,880(6.5%)	43,340(1.57%)	5,099(0.18%)	2,525,565(91.7%)
10	1,309,791	106,315(8.11%)	15,847(1.2%)	37(0.002%)	1,187,592(90.67%)
11	107,251	8,161(7.6%)	2,718(2.53%)	3(0.002%)	96,369(89.85%)
12	325,471	2,143(0.65%)	7,628(2.34%)	25(0.007%)	315,675(96.99%)
13	1,925,149	38,791(2.01%)	31,939(1.65%)	1,202(0.06%)	1,853,217(96.26%)

Trong tập dữ liệu này, trong mỗi bản chụp ngoài tệp *.pcap chúng ta còn có một tập tin *.binetflow, tập tin này cũng chứa thông tin của các flow và mỗi flow đã được gán nhãn sẵn nên ta sẽ sử dụng để tiến hành huấn luyện và đánh giá mô hình.

Cụ thể trong nhãn của mỗi loại lưu lượng sẽ như sau:

- Flow normal: nhãn không chứa chuỗi "Botnet"
- Flow botnet: nhãn có chứa chuỗi "Botnet" và không chứa chuỗi "CC"
- Flow C&C: nhãn chứa cả chuỗi "Botnet" và "CC"

Chúng ta sẽ dựa vào những đặc điểm này để tiến hành xây dựng mô hình cho đề tài này.

2.3. Hiện thực mô hình

2.3.1. Chuẩn bị và xử lý dữ liệu

Đầu tiên, ta tiến hành tạo nơi để lưu trữ các bản chụp của bộ dữ liệu CTU-13, với tổng số 13 bản chụp ta khởi tạo các thư mục với đường dẫn từ Google Drive bằng các câu lệnh sau:

```
[1]  # Tạo các thư mục để chứa dữ liệu
     import os
     def create_directory(path):
       try:
         os.mkdir(path)
       except:
         pass

     main_dir = '/content/CTU-13-Dataset/'
     create_directory(main_dir)
     for i in range(1, 14):
       path = main_dir + str(i)
       create_directory(path)
```

Hình 2.2: Khởi tạo thư mục làm việc và lưu trữ dữ liệu

Tiếp theo ta tiến hành tải xuống 13 bản chụp này vào các thư mục tương ứng vừa được tạo ra, đường dẫn đến các tệp được công khai trong [26]

```
!wget -P /content/CTU-13-Dataset/1 https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-42/detailed-bidirectional-flow-labels/capture20110810.binetflow
!wget -P /content/CTU-13-Dataset/2 https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-43/detailed-bidirectional-flow-labels/capture20110811.binetflow
!wget -P /content/CTU-13-Dataset/3 https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-44/detailed-bidirectional-flow-labels/capture20110812.binetflow
!wget -P /content/CTU-13-Dataset/4 https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-45/detailed-bidirectional-flow-labels/capture20110815.binetflow
!wget -P /content/CTU-13-Dataset/5 https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-46/detailed-bidirectional-flow-labels/capture20110815-2.binetflow
!wget -P /content/CTU-13-Dataset/6 https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-47/detailed-bidirectional-flow-labels/capture20110816.binetflow
!wget -P /content/CTU-13-Dataset/7 https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-48/detailed-bidirectional-flow-labels/capture20110816-2.binetflow
!wget -P /content/CTU-13-Dataset/8 https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-49/detailed-bidirectional-flow-labels/capture20110816-3.binetflow
!wget -P /content/CTU-13-Dataset/9 https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-50/detailed-bidirectional-flow-labels/capture20110817.binetflow
!wget -P /content/CTU-13-Dataset/10 https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-51/detailed-bidirectional-flow-labels/capture20110818.binetflow
!wget -P /content/CTU-13-Dataset/11 https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-52/detailed-bidirectional-flow-labels/capture20110818-2.binetflow
!wget -P /content/CTU-13-Dataset/12 https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-53/detailed-bidirectional-flow-labels/capture20110819.binetflow
!wget -P /content/CTU-13-Dataset/13 https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-54/detailed-bidirectional-flow-labels/capture20110815-3.binetflow

--2022-10-15 04:18:42--  https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-42/detailed-bidirectional-flow-labels/capture20110810.binetflow
Resolving mcfp.felk.cvut.cz (mcfp.felk.cvut.cz)... 147.32.82.194
Connecting to mcfp.felk.cvut.cz (mcfp.felk.cvut.cz)|147.32.82.194|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 386629271 (369M)
Saving to: '/content/CTU-13-Dataset/1/capture20110810.binetflow'

capture20110810.bin 100%[===================>] 368.72M  9.65MB/s    in 33s

2022-10-15 04:19:16 (11.2 MB/s) - '/content/CTU-13-Dataset/1/capture20110810.binetflow' saved [386629271/386629271]
```

Hình 2.3: Tải xuống các bản chụp của bộ dữ liệu CTU-13

Cài đặt các thư viện cần thiết, cài đặt GPU cho notebook để tăng tốc độ xử lý (Google Colab mặc định chạy trên CPU)

```
# import các thư viện cần thiết
import numpy as np
from glob import glob
from tqdm import tqdm
import torch
import keras as k
import torchvision.models as models
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import Conv2D, MaxPooling2D, GlobalAveragePooling2D
from keras.layers import Dropout, Flatten, Dense, Activation
from keras.models import Sequential
from tensorflow.keras.layers import BatchNormalization
from torchvision import datasets, models, transforms
import torch.nn as nn
import torch.optim as optim
from tensorflow.keras.optimizers import Adam, Adamax
import gc
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
import math
from sklearn.preprocessing import MinMaxScaler
from PIL import Image
from sklearn.model_selection import train_test_split
import time

gc.collect()

torch.cuda.empty_cache()
```

Hình 2.4: Import các thư viện cần thiết và kiểm tra runtime

Tiếp theo ta sẽ chuyển các bản chụp (binetflow) sang tệp csv để dễ dàng đọc và xử lý dữ liệu.

```
def rename(path_file, new_name):
  dir = os.path.dirname(path_file)
  path_new_name = os.path.join(dir, new_name)
  os.rename(path_file, path_new_name)
```

```
dir = '/content/CTU-13-Dataset'
listDir = os.listdir(dir)
listCSV = []
for subDir in listDir:
  path_subDir = os.path.join(dir, subDir)
  binetflow_file = os.path.join(path_subDir, os.listdir(path_subDir)[0])
  new_name = subDir + '.csv'
  rename(binetflow_file, new_name)
  listCSV.append(os.path.join(path_subDir, new_name))
print(listCSV)
```

```
['/content/CTU-13-Dataset/11/11.csv', '/content/CTU-13-Dataset/10/10.csv', '/content/CTU-13-Dataset/7/7.csv',
```

Hình 2.5: Chuyển dữ liệu dạng binetflow sang csv và lưu trữ vào thư mục đã tạo trước đó

Ở bước xử lý dữ liệu này, ta sẽ chọn 1 trong 13 bản chụp để làm dataset cho bài toán, vì số lượng dữ liệu rất lớn nên sẽ tốn kém nhiều thời gian để huấn luyện. Cụ thể, ta sẽ chọn bản chụp thứ 8

```
df = pd.read_csv('/content/CTU-13-Dataset/8/8.csv')
df
```

	StartTime	Dur	Proto	SrcAddr	Sport	Dir	DstAddr	Dport	State	sTos	dTos	TotPkts	TotBytes	SrcBytes	Label
0	2011/08/16 14:18:55.889839	3599.725830	tcp	88.176.79.163	49375	<?>	147.32.84.172	46696	A_PA	0.0	0.0	274708	271837036	6887036	flow=Background
1	2011/08/16 14:18:55.890497	3532.512939	tcp	134.2.99.108	12106	<?>	147.32.84.59	51472	PA_PA	0.0	0.0	513	42640	21205	flow=Background-Established-cmpgw-CVUT
2	2011/08/16 14:18:55.892530	3599.922852	udp	109.80.124.147	10227	<->	147.32.86.77	43332	CON	0.0	0.0	69490	14362350	11870607	flow=Background-UDP-Established
3	2011/08/16 14:18:55.892624	82.457962	tcp	147.32.84.59	34394	<?>	74.125.39.117	80	FPA_FPA	0.0	0.0	12526	8472421	8118714	flow=Background-Established-cmpgw-CVUT
4	2011/08/16 14:18:55.894121	3599.954346	tcp	88.176.79.163	49410	<?>	147.32.84.172	52935	A_PA	0.0	0.0	285606	281932488	7230814	flow=Background
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2954225	2011/08/17 09:47:11.104593	0.177685	udp	109.87.182.34	7116	<->	147.32.85.84	13716	CON	0.0	0.0	4	256	120	flow=Background-UDP-Established
2954226	2011/08/17 09:47:11.124201	0.000630	udp	72.225.154.183	35027	<->	147.32.84.229	13363	CON	0.0	0.0	2	659	80	flow=Background-UDP-Established
2954227	2011/08/17 09:47:11.165013	0.047071	udp	147.32.84.229	13363	<->	86.188.207.200	35910	CON	0.0	0.0	2	551	77	flow=Background-UDP-Established
2954228	2011/08/17 09:47:11.206812	0.000491	udp	82.3.163.36	36771	<->	147.32.84.229	13363	CON	0.0	0.0	2	137	77	flow=Background-UDP-Established
2954229	2011/08/17 09:47:11.231725	0.000463	udp	76.21.111.166	63026	<->	147.32.84.229	13363	CON	0.0	0.0	2	133	73	flow=Background-UDP-Established

2954230 rows × 15 columns

Hình 2.6: Những thuộc tính của bản chụp 8 và các dòng dữ liệu đầu tiên

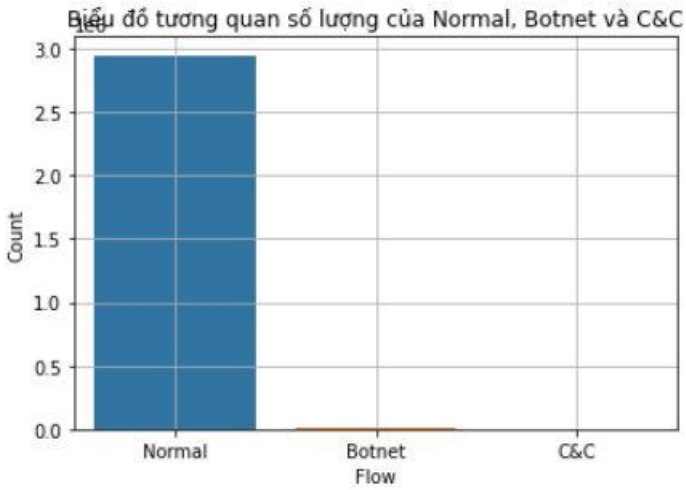
Bản chụp thứ 8 này có 2954230 dòng dữ liệu với 15 thuộc tính, trong đó có 3 thuộc tính với kiểu dữ liệu số thực, 3 thuộc tính kiểu số nguyên và 9 thuộc tính kiểu object. Chi tiết các thuộc tính, tên gọi, kiểu dữ liệu, ý nghĩa sẽ được mô tả cụ thể trong bảng dưới đây.

Bảng 2.4: Chi tiết các thuộc tính trong bộ dữ liệu

STT	Tên thuộc tính	SL dữ liệu không null	Kiểu dữ liệu	Ý nghĩa
1	StartTime	2954230	object	Thời gian bắt đầu ghi
2	Dur	2954230	float64	Tổng thời gian ghi
3	Proto	2954230	object	Giao thức giao dịch
4	SrcAddr	2954230	object	Địa chỉ IP nguồn
5	Sport	2923545	object	Mã port nguồn
6	Dir	2954230	object	Hướng giao dịch
7	DstAddr	2954230	object	Địa chỉ IP đích
8	Dport	2939943	object	Mã port đích
9	State	2954227	object	Source packet trên mỗi giây
10	sTos	2921292	float64	Giá trị byte TOS nguồn
11	dTos	2799837	float64	Giá trị byte TOS đích

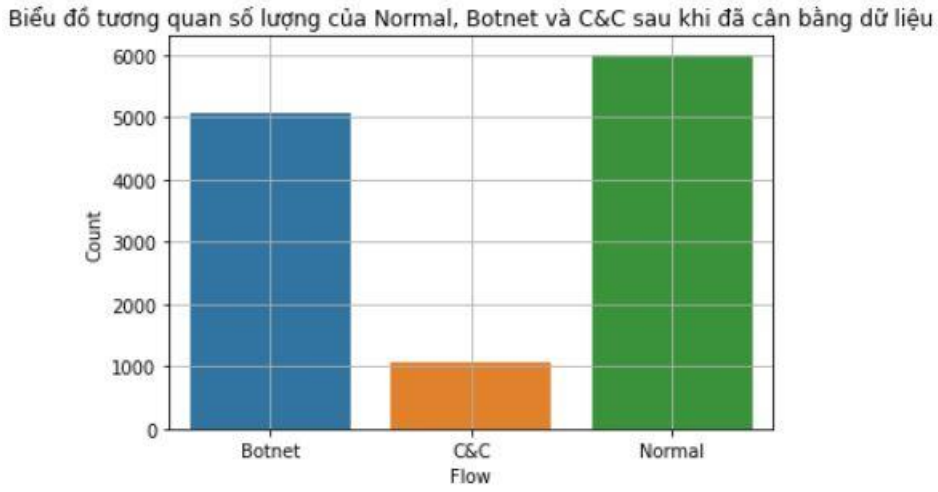
12	TotPkts	2954230	int64	Tổng số gói giao dịch
13	TotBytes	2954230	int64	Tổng số byte giao dịch
14	SrcBytes	2954230	int64	Số byte từ giao dịch nguồn đến đích
15	Label	2954230	object	Nhãn

Bước tiếp theo là chuẩn hóa cột Label – chứa nhãn của dữ liệu, cột Label có 4 giá trị là Normal, Botnet, C&C và Background. Mục đích của ta là phân loại lưu lượng có phải là Botnet, C&C hay không, vì thế ta sẽ bỏ qua lưu lượng có label Background và tính nó như một lưu lượng bình thường. Kết quả sau khi xử lý ta thu được cột Label với 3 loại nhãn Normal, C&C và Botnet có tỉ lệ như sau:



Hình 2.7: Biểu đồ tương quan số lượng giữa 3 nhãn trong bộ dữ liệu

Lúc này ta thấy rõ sự chênh lệch dữ liệu giữa 3 nhãn, điều này là không tốt vì có quá ít dữ liệu cho nhãn Botnet và C&C dẫn đến mô hình sẽ khó có thể học được chính xác khi bộ dữ liệu mất cân bằng. Chính vì thế ta sẽ giới hạn lại bộ dữ liệu sao cho tỉ lệ giữa 3 nhãn là cân đối. Ta thực hiện giảm dòng dữ liệu cho nhãn Normal. Sau khi giảm dòng Normal, ta có được bộ dữ liệu được cân bằng như hình 2.8.



Hình 2.8: Biểu đồ tương quan số lượng giữa 3 nhãn trong bộ dữ liệu sau khi cân bằng

Tiếp theo, ta sẽ tiến hành biến đổi và mã hóa dữ liệu của các thuộc tính có kiểu dữ liệu object. Những thuộc tính này có thể chia thành 2 nhóm, nhóm không phân loại bao gồm: thời gian bắt đầu record, địa chỉ nguồn và đích của IP và port (StartTime, SrcAddr, Sport, DstAddr, Dport). Nhóm có thể phân loại bao gồm Proto, Dir và State. Vì đặc trưng của dữ liệu nên ta sẽ chia 2 cách mã hóa khác nhau. Cụ thể sử dụng fit_transform của LabelEncoder để biến đổi nhóm đầu tiên gồm 5 thuộc tính và OneHotEncoder để biến đổi nhóm còn lại.

	StartTime	Dur	SrcAddr	Sport	DstAddr	Dport	sTos	dTos	TotPkts	TotBytes	...	State_51	State_52	State_53	State_54	State_55	State_56	State_57	State_58	State_59	State_60
0	1798	0.0	180	3197	221	56	0.0	NaN	1	62	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	3644	0.0	180	1787	327	56	0.0	NaN	1	62	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	395	6.758171	180	2041	599	348	0.0	0.0	18	6592	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	3321	0.0	180	4620	551	56	0.0	NaN	1	62	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	261	0.0	180	4179	318	56	0.0	NaN	1	62	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
12122	4619	0.000286	178	4464	139	279	0.0	0.0	2	214	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12123	6540	0.000013	232	8471	434	0	NaN	0.0	2	148	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12124	6534	0.000377	759	6958	351	54	0.0	0.0	2	231	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12125	6535	0.000475	726	6107	351	54	0.0	0.0	2	559	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12126	6537	0.000241	207	7785	139	279	0.0	0.0	2	208	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

12127 rows × 85 columns

Hình 2.9: Dữ liệu sau khi được mã hóa

Kết quả từ hình 2.9 cho thấy kết quả sau khi thực hiện quá trình biến đổi các thuộc tính trên, ta thấy kích thước của bộ dữ liệu đã tăng lên từ 15 lên 85 cột dữ liệu. Lý do là vì với các cột sử dụng OneHotEncoder sẽ tạo ra những cột dữ liệu mới có số lượng bằng với số lượng phân loại dữ liệu của mỗi cột đấy. Từ 15 cột ban đầu, bỏ đi 3 cột Proto, Dir và State và thay thế bởi số lượng phân loại cột tương ứng ta sẽ đạt được tổng số cột là 85.

2.3.2. Chuyển đổi và phân chia dữ liệu hình ảnh

Đến lúc này, các thuộc tính đều đã được chuẩn hóa, tuy nhiên chúng lại có độ dài ngắn khác nhau bởi đặc trưng dữ liệu và phương pháp mã hóa vì thế ta cần đưa chúng về cùng một độ dài. Ở đây ta sẽ sử dụng MinMaxScaler để đưa dữ liệu sang miền giá trị của các pixels (0, 255) trước khi chuyển đổi chúng sang dữ liệu dạng hình ảnh. Mỗi ảnh sẽ mang kích thước 192x192 và được lưu trữ trong thư mục Images_Data/Botnet đối với hình ảnh có nhãn Botnet và Images_Data/Normal cho nhãn Normal, tương tự cho nhãn C&C. Ta sử dụng thư viện PIL để chuyển sang hình ảnh.

```
# Convert sang ảnh
scaler = MinMaxScaler(feature_range=(0, 255)) # Chuyển sang vùng giá trị của các điểm ảnh (0, 255)
botnet = 1
cc = 1
normal = 1
for i in range(len(X)):
  pixels = np.resize(X[i], (HEIGHT, WIDTH))
  pixels = scaler.fit_transform(pixels)
  pixels = np.rint(pixels).astype(np.uint8)
  image = Image.fromarray(pixels)
  if y[i] == 'Botnet':
    name = 'Botnet_' + str(botnet) + '.jpg'
    image.save('/content/Images_Data/Botnet/' + name)
    botnet += 1
  elif y[i] == 'C&C':
    name = 'C&C_' + str(cc) + '.jpg'
    image.save('/content/Images_Data/C&C/' + name)
    cc += 1
  else:
    name = 'Normal_' + str(normal) + '.jpg'
    image.save('/content/Images_Data/Normal/' + name)
    normal += 1
```

Hình 2.10: Chuyển dữ liệu về dạng hình ảnh

Sau khi đã hoàn tất các công đoạn trên ta tiến hành bước cuối cùng, chia dữ liệu thành tập train, test với tỉ lệ 80:20; Sau đó, từ tập train vừa chia xong ở trên ta chia tiếp train, validation theo tỉ lệ 75:25 để kiểm tra và đánh giá độ chính xác của model trong quá trình training, như vậy ta có 3 tập dữ liệu là train, test, validation được chia theo tỉ lệ 60:20:20. Do tỉ lệ 60:20:20 là tỉ lệ được sử dụng phổ biến và dữ liệu không quá lớn nên mô hình lựa chọn tỉ lệ này để luyện. Tiếp theo, chuyển các tập dữ liệu vừa chia xong vào các thư mục lưu trữ tương ứng như sau:

```
# Tạo ba thư mục chứa tập ảnh train, test và validation
create_directory('/content/image_dataset')

# Thư mục train
create_directory('/content/image_dataset/train_data')
create_directory('/content/image_dataset/train_data/Botnet')
create_directory('/content/image_dataset/train_data/Normal')
create_directory('/content/image_dataset/train_data/C&C')

# Thư mục test
create_directory('/content/image_dataset/test_data')
create_directory('/content/image_dataset/test_data/Botnet')
create_directory('/content/image_dataset/test_data/Normal')
create_directory('/content/image_dataset/test_data/C&C')

# Thư mục validation
create_directory('/content/image_dataset/validation_data')
create_directory('/content/image_dataset/validation_data/Botnet')
create_directory('/content/image_dataset/validation_data/Normal')
create_directory('/content/image_dataset/validation_data/C&C')
```

Hình 2.11: Tạo thư mục lưu trữ tương ứng cho mỗi loại Normal, Botnet và C&C

```
def move_image(path, dir):
  for img in path:
    if img.find('Botnet') != -1:
      os.rename(img, img.replace('Images_Data/Botnet', 'image_dataset/' + dir))
    elif img.find('C&C') != -1:
      os.rename(img, img.replace('Images_Data/C&C', 'image_dataset/' + dir))
    else:
      os.rename(img, img.replace('Images_Data/Normal', 'image_dataset/' + dir))


dir_nor = '/content/Images_Data/Normal'
dir_bot = '/content/Images_Data/Botnet'
dir_cc = '/content/Images_Data/C&C'
list_botnet = []
list_cc = []
list_normal = []
for img in os.listdir(dir_nor):
  list_normal.append(os.path.join(dir_nor, img))
for img in os.listdir(dir_bot):
  list_botnet.append(os.path.join(dir_bot, img))
for img in os.listdir(dir_cc):
  list_cc.append(os.path.join(dir_cc, img))
```

Hình 2.12: Định nghĩa nơi lưu trữ dữ liệu của từng loại

```
# Chia train, test từ tập dữ liệu ảnh đã convert
train_botnet, test_botnet = train_test_split(list_botnet, test_size=0.2, random_state=42)
train_cc, test_cc = train_test_split(list_cc, test_size=0.2, random_state=42)
train_normal, test_normal = train_test_split(list_normal, test_size=0.2, random_state=42)

# Chia train, validation từ tập dữ liệu train
train_botnet, val_botnet = train_test_split(train_botnet, test_size=0.25, random_state=42)
train_cc, val_cc = train_test_split(train_cc, test_size=0.25, random_state=42)
train_normal, val_normal = train_test_split(train_normal, test_size=0.25, random_state=42)

move_image(train_botnet, 'train_data/Botnet')
move_image(train_cc, 'train_data/C&C')
move_image(train_normal, 'train_data/Normal')

move_image(test_botnet, 'test_data/Botnet')
move_image(test_cc, 'test_data/C&C')
move_image(test_normal, 'test_data/Normal')

move_image(val_botnet, 'validation_data/Botnet')
move_image(val_cc, 'validation_data/C&C')
move_image(val_normal, 'validation_data/Normal')

os.rmdir(dir_nor)
os.rmdir(dir_bot)
os.rmdir(dir_cc)
os.rmdir('/content/Images_Data')
```

Hình 2.13: Chia dữ liệu và di chuyển vào nơi lưu trữ tương ứng mỗi loại

2.3.3. Xây dựng mô hình phân loại

Mô hình phân loại được xây dựng bằng việc sử dụng Resnet-18. Resnet-18 là mạng CNN phổ biến được sử dụng nhiều trong lĩnh vực học sâu Deep Learning, được sử dụng cho những hệ thống phân loại và dự đoán với độ chính xác cao.

Đầu tiên ta khai báo đường dẫn đến các thư mục train, test và validation của bộ dữ liệu.

```
[ ]  train_data_dir = '/content/image_dataset/train_data'
     test_data_dir = '/content/image_dataset/test_data'
     val_data_dir = '/content/image_dataset/validation_data'
```

Hình 2.14: Định nghĩa đường dẫn chứa các tập đã chia

Bước tiếp theo, ta thực hiện khởi tạo Resnet-18 CNN và thực hiện kiểm tra số lượng feature của vector đầu vào với số lớp đầu ra tương ứng, ở đây là 512 feature

cho một vector đặc trưng và 1000 lớp. Kết quả trên không phù hợp với bài toán hiện đang thực hiện nên ta cần phải khởi tạo lại số lớp đầu ra cho mô hình

```
resnet = models.resnet18(pretrained=True)
print(resnet)
print(resnet.fc.in_features)
print(resnet.fc.out_features)
# check if CUDA is available
use_cuda = torch.cuda.is_available()
```

Hình 2.15: Thực hiện khai báo mạng Resnet-18 CNN và kiểm tra feature

Kế tiếp, ta thực hiện transforms và load dữ liệu từ các tập train, test và validation**.** Ta phải transform dữ liệu sang kiểu tensor để phù hợp với mô hình của thư viện tensorflow

```
transform = transforms.Compose([transforms.ToTensor()])
train_data = datasets.ImageFolder('/content/image_dataset/train_data', transform=transform)
test_data = datasets.ImageFolder('/content/image_dataset/test_data', transform=transform)
val_data = datasets.ImageFolder('/content/image_dataset/validation_data', transform=transform)
print('Num training images: ', len(train_data))
print('Num validation images: ', len(val_data))
print('Num test images: ', len(test_data))
```

Hình 2.16: Transform dữ liệu sang kiểu tensor để phù hợp với mô hình

Ở bài toán này chúng ta cần phân loại ra 3 lớp là Normal, Botnet và C&C nên ta sẽ khởi tạo lại số output đầu ra cho mô hình là 3.

```
n_inputs = resnet.fc.in_features
last_layer = nn.Linear(n_inputs, 3)
resnet.fc = last_layer
print(resnet.fc.out_features)
```

Hình 2.17: Khởi tạo giá trị output đầu ra

Trích xuất vector đặc trưng và nhãn từ ảnh của của các tập train, test và validation để đưa vào mô hình thực hiện train

```
# define dataloader parameters
batch_size = 124
num_workers = 0

# prepare data loaders
train_loader = torch.utils.data.DataLoader(train_data, batch_size=batch_size,
                                           num_workers=num_workers, shuffle=True)
valid_loader = torch.utils.data.DataLoader(val_data, batch_size=batch_size,
                                           num_workers=num_workers, shuffle=True)
test_loader = torch.utils.data.DataLoader(test_data, batch_size=batch_size,
                                           num_workers=num_workers, shuffle=True)
loaders = {
    'train': train_loader,
    'valid': valid_loader,
    'test': test_loader
}
```

Hình 2.18: Trích xuất vector đặc trưng cho mỗi tập

Tiếp theo ta thực hiện xây dựng hàm train để thực hiện train mô hình, sau đó trả về mô hình đã được luyện.

```
# specify loss function (categorical cross-entropy)
criterion = nn.CrossEntropyLoss()

# specify optimizer (stochastic gradient descent) and learning rate = 0.001
optimizer = optim.SGD(resnet.fc.parameters(), lr=0.001)
```

```
def train(n_epochs, loaders, model, optimizer, criterion, use_cuda, save_path, last_validation_loss=None):
    """returns trained model"""
    # initialize tracker for minimum validation loss
    valid_loss_min = np.Inf
    train_loss_ls = []
    valid_loss_ls = []

    for epoch in range(1, n_epochs+1):
        # initialize variables to monitor training and validation loss
        train_loss = 0.0
        valid_loss = 0.0
        trainLoss = 0.0
        train_batch = 0
        validLoss = 0.0
        valid_batch = 0
```

```
###################
# train the model #
###################
model.train()
for batch_idx, (data, target) in enumerate(loaders['train']):

    # move to GPU
    if use_cuda:
        model = model.to('cuda')
        data, target = data.to('cuda'), target.to('cuda')
    ## find the loss and update the model parameters accordingly
    ## record the average training loss, using something like
    ## train_loss = train_loss + ((1 / (batch_idx + 1)) * (loss.data - train_loss))
    optimizer.zero_grad()
    # forward pass: compute predicted outputs by passing inputs to the model
    output = model(data)
    # calculate the batch loss
    loss = criterion(output, target)
    # backward pass: compute gradient of the loss with respect to model parameters
    loss.backward()
    # perform a single optimization step (parameter update)
    optimizer.step()
    # update training loss
    train_loss = train_loss + ((1 / (batch_idx + 1)) * (loss.data - train_loss))
    trainLoss += train_loss
    train_batch += 1
```

```
######################
# validate the model #
######################
model.eval()
for batch_idx, (data, target) in enumerate(loaders['valid']):

    # move to GPU
    if use_cuda:
        model = model.to('cuda')
        data, target = data.to('cuda'), target.to('cuda')
    ## update the average validation loss
    output = model(data)
    # calculate the batch loss
    loss = criterion(output, target)
    valid_loss = valid_loss + ((1 / (batch_idx + 1)) * (loss.data - valid_loss))
    validLoss += valid_loss
    valid_batch += 1

trainLoss = trainLoss.cpu().numpy()
validLoss = validLoss.cpu().numpy()
```

```
    # print training/validation statistics
    print('Epoch: {} \tTraining Loss: {:.6f} \tValidation Loss: {:.6f}'.format(
        epoch,
        train_loss,
        valid_loss
        ))
    valid_loss_ls.append(validLoss/valid_batch)
    train_loss_ls.append(trainLoss/train_batch)
    ## TODO: save the model if validation loss has decreased
    if valid_loss <= valid_loss_min:
        print('Validation loss decreased ({:.6f} --> {:.6f}).  Saving model ...'.format(
        valid_loss_min,
        valid_loss))
        torch.save(model.state_dict(), 'model.pt')
        valid_loss_min = valid_loss
    if len(valid_loss_ls) > 1 and abs(valid_loss_min - valid_loss) <= stopping_threshold:
      break
# return trained model
plt.plot(train_loss_ls,'-o')
plt.plot(valid_loss_ls,'-o')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train','Valid'])
plt.title('Train vs Valid Loss')
plt.show()
return model
```

Hình 2.19: Xây dựng hàm train và trả về mô hình đã train

CHƯƠNG 3. THÍ NGHIỆM VÀ ĐÁNH GIÁ

3.1. Các trường hợp thí nghiệm

Luận văn đã hiện thực mô hình với bản chụp thứ 8 của bộ dữ liệu CTU-13 với kích thước hình ảnh đầu vào từ dữ liệu đã qua xử lý là 192x192. Tuy nhiên, thực nghiệm chỉ một kích thước ảnh không đảm bảo rằng việc xử lý dữ liệu và áp dụng nó vào mô hình đề xuất có thật sự hiệu quả, đồng thời ta cũng chưa cân nhắc đến việc thay đổi kích thước sẽ ảnh hưởng đến độ chính xác của mô hình theo chiều hướng nào và đâu là kích thước phù hợp nhất cho mô hình phân loại.

Với mục tiêu khám phá độ phù hợp cũng như chất lượng của mô hình sẽ tăng giảm như thế nào khi điều chỉnh kích thước của hình ảnh đầu vào, luận văn đề xuất việc áp dụng ba trường hợp với kích thước ảnh lần lượt là: 192x192, 200x200, 224x244. Thí nghiệm sẽ thực hiện từng trường hợp và quan sát độ chính xác, độ mất mát cũng như thời gian thực thi của mô hình, từ đó hiểu được sự ảnh hưởng của kích thước ảnh với mô hình phân loại

3.2. Luyện và kiểm thử mô hình

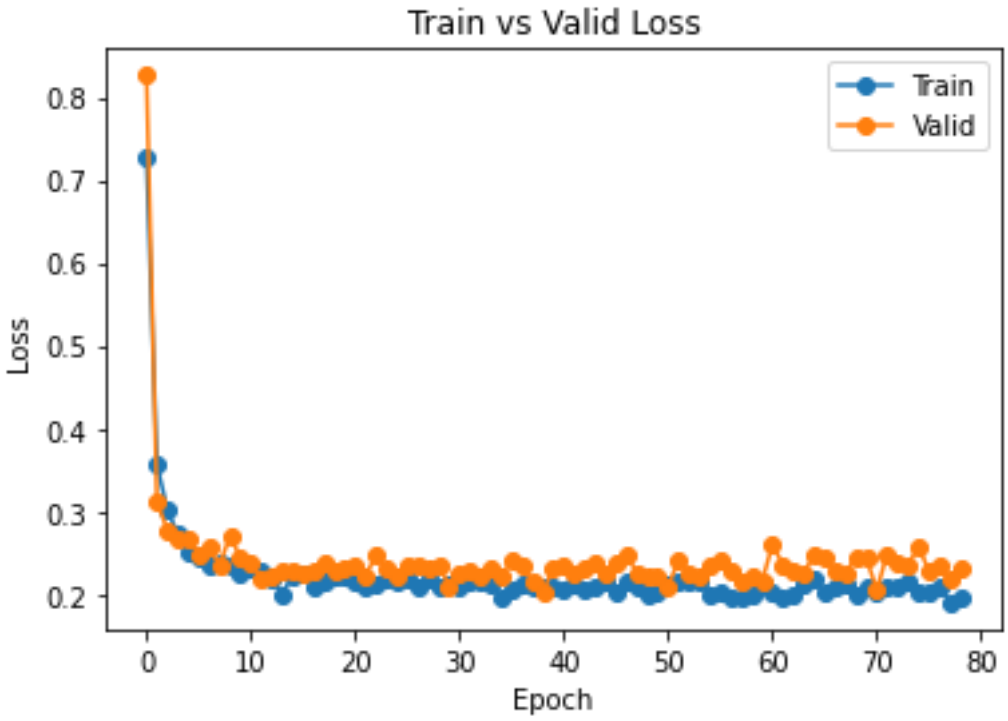
Để thực nghiệm mô hình đã xây dựng, luận văn đã sử dụng tính năng GPU của Google Colab để cải thiện tốc độ tính toán. Hiện nay, Google Colab có hỗ trợ nhiều loại GPU, thường là Nvidia K80s, T4s, P4s and P100s, tuy nhiên ta không thể tự do chọn loại GPU trong Colab vì chúng thay đổi theo thời gian. 1 GPU cho phép xử lý nhiều phép tính song song với rất nhiều core sẽ nhanh hơn nhiều so với CPU.

Sau khi chạy training mô hình với các kích thước như trên, ta thu được kết quả được kết quả về bộ biến thiên của hàm mất mát như trong hình 3.1-3.3. Ngoài ra, thông tin chi tiết về quá trình huấn luyện cũng đã được thống kê lại trong bảng dưới đây, bảng trình bày cụ thể giá trị của hàm mất mát, thời gian thực thi và số lượng epoch dùng để huấn luyện mô hình phân loại của cả ba trường hợp đã đề xuất:

Kích thước ảnh đầu vào	Epoch dừng	Độ mất mát	Thời gian thực thi
192x192	93	17.75%	28 phút 33 giây
200x200	46	12.33%	17 phút 02 giây
224x224	79	22.59%	30 phút 31 giây

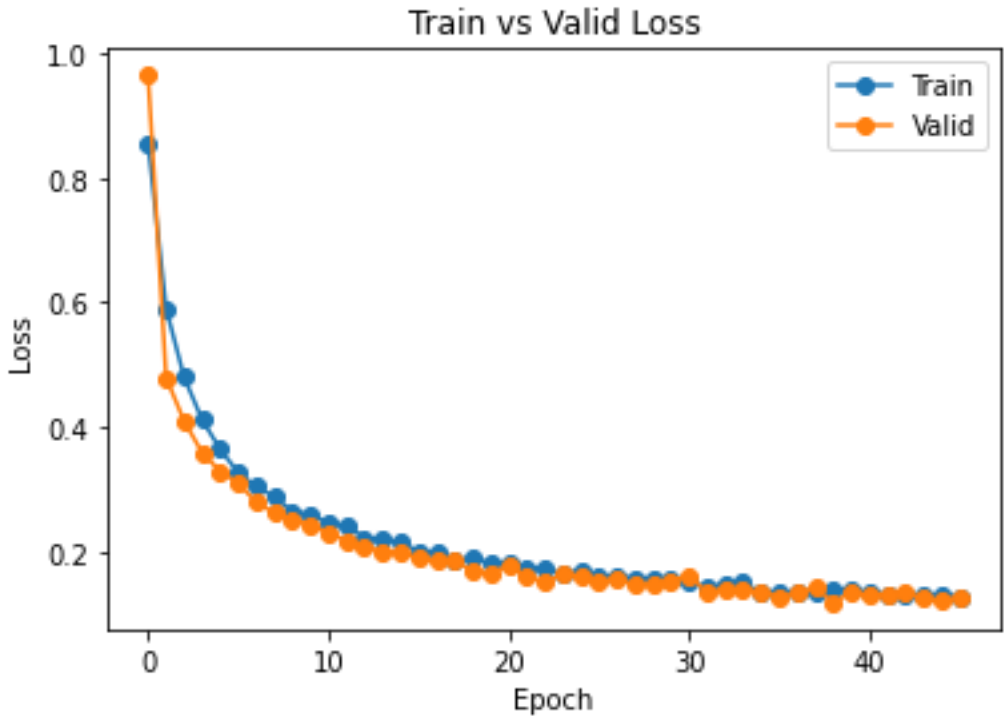
Bảng 3.1: Kết quả huấn luyện 3 kích thước ảnh

Tương ứng với số lượng epoch sử dụng để training, kích thước ảnh 192 và 224 tiêu tốn khá nhiều thời gian hơn so với trường hợp 200 mặc dù độ mất mát lại thua kém đáng kể. Điều này đã phần nào nói lên sự phù hợp của mô hình với các kích thước ảnh khác nhau. Tiếp theo, ta sẽ theo dõi sự biến thiên của mỗi mô hình trong quá trình huấn luyện. Mỗi biểu đồ dưới đây sẽ thể hiện sự biến thiên về giá trị của hàm mất mát, trong đó đường xanh tượng trưng cho tập train và cam đại diện cho tập validation. Từng mô hình đều có độ mất mát giảm dần từ 100% đến khoảng 20% theo chiều tăng dần của lượng epoch.



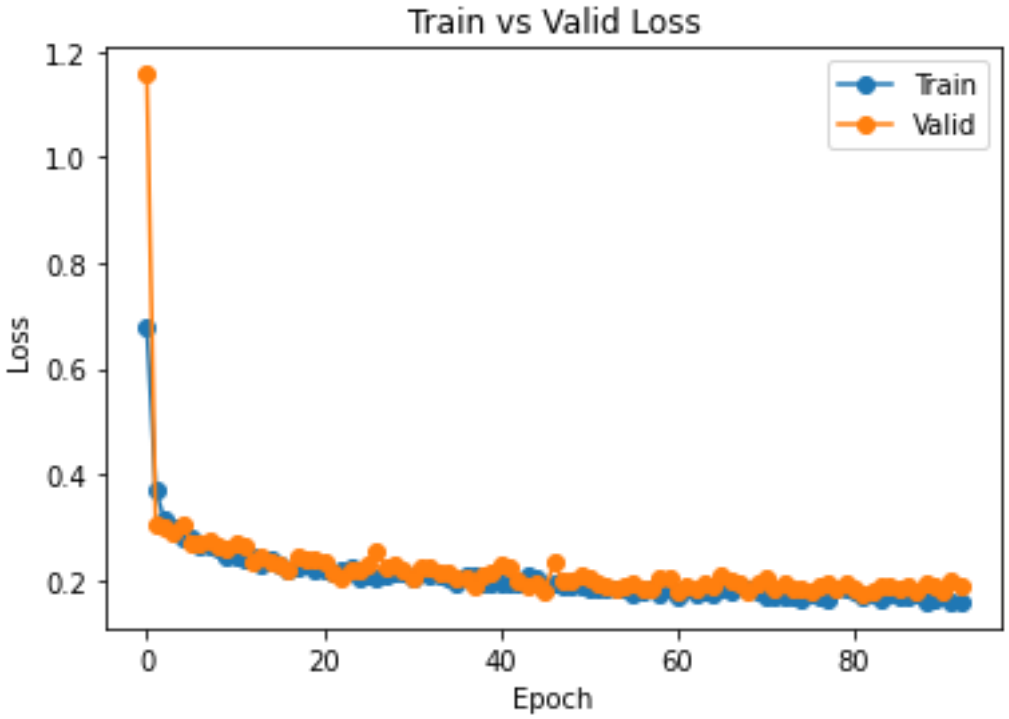
Hình 3.1: Độ biến thiên của hàm mất mát trong trường hợp kích thước 192x192

Với trường hợp kích thước 192x192 ta nhận thấy đường cam và xanh bắt đầu từ vị trí 80% và 70%, chúng giảm ngay về khoảng 35% với epoch đầu tiên và đồng thời giảm dần khi epoch tăng dần. Tuy nhiên ta thấy rằng hai đường train và validation không thật sự fit với nhau, cụ thể từ epoch thứ 40-80 các điểm dữ liệu giữa tập train và validation dần xa nhau hơn. Bên cạnh đó, các điểm trên đường validation tăng giảm khá nhiều và chưa có sự mượt mà, giá trị của hàm mất mát cũng không có xu hướng giảm thêm nhiều khi epoch ở giá trị từ 20. Điều này chứng tỏ rằng trường hợp kích thước đầu vào 192x192 này không phù hợp với mô hình phân loại của bài toán.



Hình 3.2: Độ biến thiên của hàm mất mát trong trường hợp kích thước 200x200

Ngược lại với Hình 3.1, Hình 3.2 ta thấy rõ sự phù hợp của cả hai đường train và validation. Các điểm của tập validation nằm vừa vặn với tập train, độ dốc của hàm mất mát cũng giảm dần một cách tự nhiên rất mượt mà. Số lượng epoch được sử dụng trong quá trình huấn luyện chưa đến 50, có nghĩa là mô hình dừng lại ở vòng lặp mà validation đạt giá trị nhỏ nhất. Chứng tỏ kích thước ảnh đầu vào này rất phù hợp với mô hình phân loại.



Hình 3.3: Độ biến thiên của hàm mất mát trong trường hợp kích thước 224x224

Trường hợp 224x224 không mang lại kết quả khả quan hơn so với trường hợp 200x200. Biểu đồ đã thể hiện sự thay đổi đột ngột về giá trị của hàm mất mát ngay tại epoch đầu tiên. Cả hai đường validation và train tuy fit với nhau nhưng ko mượt mà bằng trường hợp trước đó, các điểm giá trị trên đường validation xê dịch lên xuống không mấy ổn định. Ngoài ra, giá trị của hàm mất mát cũng không thay đổi quá nhiều so với những epoch đầu tiên.

Bước cuối cùng của quá trình thí nghiệm là chạy kiểm thử, ta tiến hành kiểm thử mô hình với đoạn code sau đây:

```python
def test(loaders, model, criterion, use_cuda):

    # monitor test loss and accuracy
    test_loss = 0.
    correct = 0.
    total = 0.

    model.eval()
    for batch_idx, (data, target) in enumerate(loaders['test']):
        # move to GPU
        if use_cuda:
            model = model.to('cuda')
            data, target = data.cuda(), target.cuda()
        # forward pass: compute predicted outputs by passing inputs to the model
        output = model(data)
        # calculate the loss
        loss = criterion(output, target)
        # update average test loss
        test_loss = test_loss + ((1 / (batch_idx + 1)) * (loss.data - test_loss))
        # convert output probabilities to predicted class
        pred = output.data.max(1, keepdim=True)[1]
        # compare predictions to true label
        correct += np.sum(np.squeeze(pred.eq(target.data.view_as(pred))).cpu().numpy())
        total += data.size(0)

    print('Test Loss: {:.6f}\n'.format(test_loss))

    print(f'Test Accuracy: {round(100. * correct / total, 2)}% ({int(correct)}/{int(total)})')

# call test function
test(loaders, resnet, criterion, use_cuda)
```

Hình 3.4: Kiểm thử mô hình

3.3. Kết quả và nhận xét

Sau khi hoàn thành quá trình kiểm thử, kết quả của cả ba trường hợp được tổng hợp trong bảng dưới đây:

Kích thước ảnh đầu vào	Độ chính xác	Số điểm phân loại chính xác
192x192	96.29%	2336/2426
200x200	97.16%	2357/2426
224x224	93.16%	2260/2426

Bảng 3.2: Kết quả thực nghiệm với tập test của 3 kích thước ảnh

Với ba trường hợp thí nghiệm, ta đã nhận được kết quả cao nhất về độ chính xác là 97.16% từ kích thước ảnh 200x200, mô hình đã phân loại được 2357 điểm chính xác trên tổng số 2426 điểm dữ liệu. Xếp thứ hai chính là trường hợp 192x192 với độ chính xác 96.29% và cuối cùng là 224x224 với độ chính xác 93.16%.

Qua các trường hợp thí nghiệm, luận văn đã tìm hiểu và xây dựng được mô hình phân loại sử dụng Representation Learning cụ thể là gray scale image cũng như khám phá ra sự ảnh hưởng của kích thước ảnh đến chất lượng của mô hình.

Tuy nhiên, do giới hạn về phần cứng khi chạy thử mô hình và số lượng dữ liệu kiểm nghiệm chưa nhiều mặc dù mô hình phân loại đạt kết quả khá cao nhưng cũng cần phải cải thiện thêm, vì thực tế nếu ta bỏ sót một vài trường hợp cũng đủ để các hacker tấn công và làm hại đến hệ thống.

KẾT LUẬN

1. Kết quả đạt được

1.1. Về mặt lý thuyết

Nắm được nguyên lý các kỹ thuật tấn công cơ bản, cách thức của tấn công Botnet.

Tìm hiểu về Trí tuệ nhân tạo (AI), các kỹ thuật Representation Learning và ứng dụng vào để phân tích dữ liệu.

Các loại kiến trúc mạng CNN của công nghệ Deep Learning.

1.2. Về mặt thực tiễn

Luận văn đã đưa ra được giải pháp cảnh báo tấn công dựa vào phân tích logs, có thể cho người quản trị biết được mối nguy hiểm trước khi có xảy ra tấn công.

Đưa ra giải pháp phân tích logs dựa vào ứng dụng Trí tuệ nhân tạo (AI), các kỹ thuật Representation Learning.

Xây dựng thành công phần mềm có thể dựa vào phân tích các pha ban đầu của tấn công để cảnh báo đến người quản trị một cách sớm nhất trước khi xảy ra cuộc tấn công thực sự nói chung và Botnet nói riêng.

2. Hạn chế

Tập dữ liệu ứng dụng nghiên cứu đã cũ cộng với phần cứng giới hạn nên độ chính xác có thể không như mong muốn. Kết quả đạt được chưa bao quát được hết các trường hợp, dữ liệu mẫu cần được training và mở rộng môi trường áp dụng.

3. Hướng phát triển

Tập trung nghiên cứu rút trích các đặc trưng phù hợp hơn cho quá trình phân tích, tăng độ chính xác trong việc nhận dạng các hành động trinh sát .

Nghiên cứu các mô hình tấn công mạng, các phương pháp trinh sát mới nhằm phát hiện và cảnh báo tốt hơn.

Mô hình cần được cải thiện và thực nghiệm với tập dữ liệu mới hơn.

DANH MỤC TÀI LIỆU THAM KHẢO

[1] M. HARGRAVE, "Deep Learning," Investopedia, 6 April 2021. [Online]. Available: https://www.investopedia.com/terms/d/deep-learning.asp.

[2] Bansal, Ankit; Mahapatra, Sudipta;, "A Comparative Analysis of Machine Learning Techniques for Botnet Detection," in *2020 21st International Arab Conference on Information Technology (ACIT)*, Giza, Egypt, 2020.

[3] V. Distribution, "Botnet là gì? Giải pháp phòng vệ Botnet FortiGuard.," 2021. [Online]. Available: https://vietnetco.vn/botnet-la-gi-giai-phap-phong-ve-botnet-fortiguard/4999.html. [Accessed 31 May 2021].

[4] Quantrimang.com, "Botnet là gì? Cấu trúc và cách botnet hoạt động như thế nào?," 2021. [Online]. Available: <https://quantrimang.com/botnet-hoat-dong-nhu-the-nao-36773. [Accessed 31 May 2021].

[5] N. T. Hưng, H. X. Dậu, V. X. Hạnh, "Phát hiện botnet dựa trên phân loại tên miền sử dụng các kỹ thuật học máy," in *Hội thảo lần thứ III: Một số vấn đề chọn lọc về an toàn an ninh thông tin*, Da Nang, 2018.

[6] V. X. Hạnh, H. X. Dậu, "Phát hiện dga botnet sử dụng kết hợp nhiều nhóm đặc trưng phân loại tên miền," in *Kỷ yếu Hội nghị KHCN Quốc gia lần thứ XII về Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR)*, Hue, 2019.

[7] H. X. Diệu, N. T. Hưng, N. T. T. Trang, "Phát hiện botnet dựa trên học máy sử dụng dữ liệu truy vấn DNS: Phân tích ảnh hưởng của các đặc trưng huấn luyện," in *Hội thảo quốc gia lần thứ XXII: Một số vấn đề chọn lọc của Công nghệ thông tin và truyền thông*, Thai Binh, 2019.

[8] G. Fedynyshyn, M. C. Chuah, G. Tan, "Detection and Classification of Different Botnet C&C Channels," in *International Conference on Autonomic and Trusted Computing*, 2011.

[9] M. Usman, M. A. Jan, X. He, J. Chen, "A Survey on Representation Learning Efforts in Cybersecurity Domain," pp. 1-28, 16 October 2019.

[10] Stevanovic, Matija; Pedersen, Jens Myrup;, "On the Use of Machine Learning for Identifying Botnet Network.," *Journal of Cyber Security and Mobility*, vol. 4, no. 2&3, p. 32, 2016.

[11] Miller, Sean; Busby-Earle, Curtis;, "The role of machine learning in botnet detection," in *2016 11th International Conference for Internet Technology and Secured Transactions (ICITST)*, Barcelona, Spain, 2016.

[12] Stevanovic, Matija; Pedersen, Jens Myrup;, "On the Use of Machine Learning for Identifying Botnet Network Traffic," *Journal of Cyber Security and Mobility,* vol. 4, no. 2&3, p. 32, 2016.

[13] A. Bijalwan, "Botnet Forensic Analysis Using Machine Learning," *Security and Communication Networks*, 2020.

[14] Suleiman Y. Yerima; Mohammed K. Alzaylaee, "Mobile Botnet Detection: A Deep Learning Approach Using Convolutional Neural Networks," in *2020 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, Dublin, Ireland, 2020.

[15] Jos Van Roosmalen; Harald P E Vranken; M. C. J. D. Van Eekelen, "Applying deep learning on packet flows for botnet detection," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, 2018.

[16] Jiawei Zhou; Zhiying Xu; Alexander M. Rush; Minlan Yu, "Automating Botnet Detection with Graph Neural Networks," in *AutoML for Networking and Systems Workshop of MLSys 2020 Conference*, 2020 .

[17] Abdulghani Ali Ahmed; Waheb A. Jabbar; Ali Safaa Sadiq; Hiran Patel , "Deep learning-based classification model for botnet attack detection," *Journal of Ambient Intelligence and Humanized Computing (2020)*, 2020.

[18] Taheri; Shayan; Salem, Milad; Yuan, Jiann-Shiun;, "Leveraging image representation of network traffic data and transfer learning in botnet detection," *Big Data and Cognitive Computing 2*, 2018.

[19] M. Feily, A. Shahrestani and S. Ramadass, "A Survey of Botnet and Botnet Detection," in *2009 Third International Conference on Emerging Security Information, Systems and Technologies*, Athens, Greece, 2009.

[20] A. Karasaridis, B. Rexroad, and D. Hoeflin, "Wide-scale Botnet Detection and Characterization," *HotBots*, 2007.

[21] Alshamkhany, Mustafa; Alshamkhany, Wisam; Mansour, Mohamed; Mueez Khan, Salam Dhou; Aloul, Fadi;, "Botnet Attack Detection using Machine Learning," in *14th International Conference on Innovations in Information Technology (IIT)*, Al Ain, United Arab Emirates, 2020.

[22] Wikipedia contributors, "Artificial neural network," Wikipedia, The Free Encyclopedia, [Online]. Available: https://en.wikipedia.org/wiki/Artificial_neural_network. [Accessed 24 May 2021].

[23] V. H. Tiệp, "Machine Learning cơ bản," 26 December 2016. [Online]. Available: https://machinelearningcoban.com/2016/12/26/introduce/.

[24] Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron;, "Representation Learning," in *Deep Learning*, London, MIT Press, 2016, p. 526.

[25] Khan, Riaz Ullah; Zhang, Xiaosong; Kumar, Rajesh; Sharif, Abubakar; Golilarz, Noorbakhsh Amiri; Alazab, Mamoun;, "An Adaptive Multi-Layer Botnet Detection Technique Using Machine Learning Classifiers," *MDPI*, vol. 9, no. 11, p. 2375, 2019.

[26] M. G. J. S. a. A. Z. Sebastian Garcia, "An empirical comparison of botnet detection methods," *Computers and Security Journal, Elsevier*, vol. 45, pp. 100-123, 2014.

[27] "Honeynet," [Online]. Available: https://www.honeynet.org/. [Accessed 15 May 2019].

[28] I. T. B. S. W. L. S. S. A. G. D. G. D. Zhao, "Botnet detection based on traffic behavior analysis and flow intervals," *Comput. Secur.*, pp. 2-16, 2013.

[29] D. O. S. M. I. S. G. Szabó, "On the validation of traffic classification algorithms," *In Proceedings of the International Conference on Passive and Active Network Measurement, Berlin, Germany*, pp. 72-81, 26-27 March 2018.

[30] Lawrence Berkeley National Laboratory and icsi, lbnl/icsi Enterprise Tracing Project, "lbnl Enterprise Trace Repository," 2005. [Online]. Available: http://www.icir.org/enterprise-tracing/. [Accessed 15 May 2019].

[31] H. S. M. T. A. G. A. Shiravi, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur*, pp. 357-374, 2012.

[32] "Malware Capture Facility Project," [Online]. Available: https://mcfp.weebly.com/. [Accessed 15 May 2019].

[33] "Classification: Precision and Recall," Machine Learning Creash Course, [Online]. Available: https://developers.google.com/machine-

learning/crash-course/classification/precision-and-recall. [Accessed 10 February 2020].

[34] Tensorflow, "tf.keras.Sequential," [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/Sequential. [Accessed 14 May 2021].

[35] Tensorflow, "Module: tf.keras.optimizers," [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/optimizers. [Accessed 14 May 2021].

[36] Tensorflow, "Module: tf.keras.losses," [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/losses. [Accessed 12 September 2020].

[37] Tensorflow, "Module: tf.keras.metrics," [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/metrics. [Accessed 20 May 2021].

[38] "Keras," [Online]. Available: https://keras.io/about/.

BẢN CAM ĐOAN

Tôi cam đoan đã thực hiện việc kiểm tra mức độ tương đồng nội dung luận văn/luận án qua phần mềm DoIT một cách trung thực và đạt kết quả mức độ tương đồng 12 % toàn bộ nội dung luận văn. Bản luận văn kiểm tra qua phần mềm là bản cứng luận văn đã nộp để bảo vệ trước hội đồng. Nếu sai tôi xin chịu các hình thức kỷ luật theo quy định hiện hành của Học viện.

Tp.HCM, ngày 28 tháng 02 năm 2023

HỌC VIÊN THỰC HIỆN LUẬN VĂN

Kiều Công Minh