BỘ GIÁO DỤC
VÀ ĐÀO TẠO

VIỆN HÀN LÂM KHOA HỌC
VÀ CÔNG NGHỆ VIỆT NAM

HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

Tống Anh Tuấn

NGHIÊN CỨU CẢI TIẾN MỘT SỐ MÔ HÌNH HỌC MÁY VÀ HỌC SÂU ÁP DỤNG CHO BÀI TOÁN PHÂN LOẠI DGA BOTNET

LUẬN ÁN TIẾN SĨ HỆ THỐNG THÔNG TIN

Hà Nội - 2023

BỘ GIÁO DỤC
VÀ ĐÀO TẠO

VIỆN HÀN LÂM KHOA HỌC
VÀ CÔNG NGHỆ VIỆT NAM

HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

Tống Anh Tuấn

NGHIÊN CỨU CẢI TIẾN MỘT SỐ MÔ HÌNH HỌC MÁY VÀ HỌC SÂU ÁP DỤNG CHO BÀI TOÁN PHÂN LOẠI DGA BOTNET

LUẬN ÁN TIẾN SĨ HỆ THỐNG THÔNG TIN
Mã số: 9 48 01 04

Xác nhận của Học viện Khoa học và Công nghệ	Người hướng dẫn 1	Người hướng dẫn 2
KT. GIÁM ĐỐC PHÓ GIÁM ĐỐC	*(Ký, ghi rõ họ tên)*	*(Ký, ghi rõ họ tên)*
Nguyễn Thị Trung	Hoàng Việt Long	Nguyễn Việt Anh

Hà Nội - 2023

LỜI CAM ĐOAN

Tôi xin cam đoan đề tài nghiên cứu trong luận án này là công trình nghiên cứu của tôi dựa trên những tài liệu, số liệu do chính tôi tự tìm hiểu và nghiên cứu. Chính vì vậy, các kết quả nghiên cứu đảm bảo trung thực và khách quan nhất. Đồng thời, kết quả này chưa từng xuất hiện trong bất cứ một nghiên cứu nào. Các số liệu, kết quả nêu trong luận án này là trung thực, nếu sai tôi hoàn toàn chịu trách nhiệm trước pháp luật.

TÁC GIẢ LUẬN ÁN

Tống Anh Tuấn

LỜI CÁM ƠN

Để hoàn thành luận án tiến sĩ này, tôi đã nhận được rất nhiều sự chỉ dạy, giúp đỡ từ tập thể người hướng dẫn, đồng nghiệp và các nhà khoa học.

Trước tiên, tôi xin được gửi lời cám ơn chân thành tới thầy PGS. TS. Hoàng Việt Long - người hướng dẫn thứ nhất và là trưởng đơn vị, người đã định hướng, giúp đỡ tôi về mặt chuyên môn cũng như tạo điều kiện cho tôi trong công tác. Tôi xin gửi lời cám ơn chân thành tới thầy PGS. TS. Nguyễn Việt Anh - người hướng dẫn thứ hai, đã luôn quan tâm, hướng dẫn chuyên môn và ủng hộ tôi trong suốt quá trình học tập tại học viện.

Tôi xin gửi lời cám ơn tới thầy PGS. TS. Lê Hoàng Sơn, PGS. TS. Nguyễn Long Giang và các thầy cô, nhà khoa học của Viện Công nghệ thông tin và Học viện Khoa học và Công nghệ đã giảng dạy, truyền đạt kiến thức, kỹ năng nghiên cứu; tạo điều kiện cho tôi tham gia các hoạt động khoa học cùng các nhóm nghiên cứu; góp ý, hướng dẫn cho tôi trong quá trình học tập.

Tôi xin gửi lời cám ơn chân thành tới Ban Giám đốc, Phòng Đào tạo và các phòng chức năng của Học viện Khoa học và Công nghệ, Viện Hàn lâm Khoa học và Công nghệ Việt Nam đã luôn quan tâm, hướng dẫn, tạo điều kiện thuận lợi cho tôi trong quá trình học tập.

Tôi xin gửi lời cám ơn tới Ban Giám hiệu, Khoa Công nghệ thông tin và các đơn vị chức năng của Trường Đại học Kỹ thuật - Hậu cần CAND đã quan tâm, động viên, tạo điều kiện về công tác, thời gian để tôi có thể tập trung học tập, nghiên cứu.

Tôi xin gửi lời cám ơn tới Quỹ Đổi mới sáng tạo Vingroup - VinIF đã tài trợ học bổng thuộc chương trình Hỗ trợ đào tạo thạc sĩ/tiến sĩ trong nước và nhiệm vụ khoa học công nghệ cấp Nhà nước mã số ĐTĐL.CN-105/21-C đã hỗ trợ đào tạo.

Tôi xin chân thành cám ơn Hội đồng khoa học, Hội đồng đánh giá luận án tiến sĩ cấp Cơ sở, Hội đồng đánh giá luận án tiến sĩ cấp Học viện đã cho tôi những ý kiến đánh giá, góp ý và hướng dẫn quý giá để tôi chỉnh sửa, hoàn thiện luận án.

Cuối cùng, tôi chia sẻ niềm vui này với vợ Lê Thị Oanh và gia đình - những người đã động viên, giúp tôi chăm sóc con nhỏ để tôi có thời gian yên tâm học tập và hoàn thành luận án này.

Hà Nội, ngày 29 tháng 08 năm 2023

Tăng Anh Tuấn

MỤC LỤC

DANH MỤC CÁC KÝ HIỆU

STT	Ký hiệu	Ý nghĩa
1.	C	Tập các phụ âm
2.	V	Tập các nguyên âm
3.	N	Tập các chữ số
4.	S	Tập các ký tự đặc biệt
5.	T	Tập các ký tự thỏa mãn một điều kiện nhất định
6.	dom	Một tên miền
7.	TF	Tần suất xuất hiện của văn bản
8.	IDF	Nghịch đảo tần suất xuất hiện của văn bản
9.	$LCS(T, dom)$	Thuật toán tìm độ dài của chuỗi dài nhất
10.	$ACS(T, dom)$	Thuật toán tìm độ dài trung bình của các chuỗi
11.	$DCS(T, dom)$	Thuật toán tìm độ chênh lệch giữa chuỗi ký tự dài nhất và ngắn nhất
12.	$NoC(T, dom)$	Thuật toán tìm số lượng ký tự xuất hiện trong tên miền
13.	$RoC(T, dom)$	Thuật toán tìm tỉ lệ xuất hiện của ký tự trong tên miền
14.	i, j, k	Ký tự thể hiện số đếm trong vòng lặp
15.	TP	Số lượng mẫu tên miền nhãn 1 được phân loại là 1
16.	TN	Số lượng mẫu tên miền nhãn 0 được phân loại là 0
17.	FP	Số lượng mẫu tên miền nhãn 0 được phân loại là 1
18.	FN	Số lượng mẫu tên miền nhãn 1 được phân loại là 0
19.	Acc	Giá trị Accuracy của bộ phân lớp
20.	Pre	Giá trị Precision của một nhãn
21.	$A.Pre$	Giá trị Precision trung bình có trọng số của bộ phân lớp
22.	Re	Giá trị Recall của một nhãn
23.	$A.Re$	Giá trị Recall trung bình có trọng số của bộ phân lớp
24.	F_1	Giá trị F_1-score của một nhãn
25.	$A.F_1$	Giá trị F_1-score trung bình có trọng số của bộ phân lớp
26.	Sup	Giá trị Support của một nhãn

DANH MỤC CÁC CHỮ VIẾT TẮT

STT	Viết tắt	Viết đầy đủ tiếng nước ngoài	Viết đầy đủ Tiếng Việt
1.	AB	Adaptive Boosting	Thuật toán Tăng cường thích ứng
2.	CNN	Convolutional Neural Network	Thuật toán Mạng nơ-ron tích chập
3.	C&C	Command and Control	Câu lệnh và điều khiển
4.	DDoS	Distributed Denial of Service	Tấn công từ chối dịch vụ phân tán
5.	DGA	Domain Generation Algorithm	Thuật toán sinh tên miền tự động
6.	DNS	Domain Name Service	Dịch vụ phân giải tên miền
7.	DoS	Denial of Service	Tấn công từ chối dịch vụ
8.	DT	Decision Trees	Thuật toán Cây quyết định
9.	HEA	Hard Ensemble Algorithm	Thuật toán học kết hợp cứng
10.	HTTP	Hyper Text Transfer Protocol	Giao thức truyền tải siêu văn bản
11.	HTTPS	Hypertext Transfer Protocol Security	Giao thức bảo mật truyền tải siêu văn bản
12.	IDS	Intrusion Detection System	Hệ thống phát hiện xâm nhập
13.	IoT	Internet of Things	Internet vạn vật
14.	IRC	Internet Relay Chat	Trò chuyện qua Internet Relay
15.	k-NN	k-Nearest Neighbour	Thuật toán k-Láng giềng gần nhất
16.	LR	Logistic Regression	Thuật toán Hồi quy logic
17.	LSTM	Long Short-Term Memory	Bộ nhớ dài-ngắn hạn
18.	NB	Naive Bayes	Thuật toán Naive Bayes
19.	NCM	Neutrosophic C-Means	Thuật toán phân cụm mờ trên tập Neutrosophic Set
20.	NCS	PhD Student	Nghiên cứu sinh
21.	NN	Neural Networks	Thuật toán Mạng nơ-ron
22.	N/A	Not Available	Không có thông tin
23.	RF	Random Forests	Thuật toán Rừng ngẫu nhiên
24.	RNN	Recurrent Neural Network	Thuật toán Mạng nơ-ron hồi quy
25.	SVM	Support Vector Machines	Thuật toán Máy vector hỗ trợ
26.	TF-IDF	Term-Frequency – Inverse Document Frequency	Tần suất thuật ngữ - Tần suất nghịch đảo văn bản

27.	URL	Uniform Resource Locator	Địa chỉ tài nguyên đồng nhất
28.	VEA	Voting Ensemble Algorithm	Thuật toán học kết hợp dựa trên bình chọn

DANH MỤC CÁC BẢNG

Trang

DANH MỤC CÁC HÌNH VẼ, ĐỒ THỊ

Trang

MỞ ĐẦU

1. Đặt vấn đề

Botnet là tập hợp các máy tính bị mã độc xâm nhập, được điều khiển và quản trị từ xa thông qua các máy chủ điều khiển - Commond-and-Control Server (C&C Server) [1]. Một số nghiên cứu đã chỉ rõ ảnh hưởng và thiệt hại mà Botnet gây ra đối với mạng máy tính [2] [3] và [4], khẳng định rằng Botnet là một mối đe dọa lớn trên Internet. Từ đó, giải pháp phát hiện Botnet là câu hỏi luôn được các nhà khoa học đặt ra và quan tâm giải quyết.

Có hai hướng tiếp cận chính thường được sử dụng trong phát hiện Botnet, bao gồm [5]:

(1) Hướng tiếp cận dựa trên Honeynet (mạng bẫy).

(2) Hướng tiếp cận dựa trên hệ thống phát hiện xâm nhập, bao gồm:

- Kỹ thuật phát hiện Botnet dựa trên sự bất thường.
- Kỹ thuật phát hiện Botnet dựa trên chữ ký.
- Kỹ thuật phát hiện Botnet dựa trên tên miền.

Mỗi kỹ thuật đều có ưu nhược điểm riêng. Theo đó, hướng tiếp cận phát hiện Botnet dựa trên mạng bẫy Honeynet thường phụ thuộc vào hoạt động của Botnet, hạn chế khả năng mở rộng và thường phục vụ nghiên cứu. Kỹ thuật phát hiện Botnet dựa trên sự bất thường có thể bị nhiễu bởi các yếu tố gây bất thường khác như sự cố máy tính, xung đột phần cứng, phần mềm. Kỹ thuật phát hiện Botnet dựa trên chữ ký có độ đặc hiệu cao với các mẫu Botnet đã biết, nhưng hạn chế với các mẫu Botnet mới. Cuối cùng kỹ thuật phát hiện Botnet dựa trên tên miền có hiệu quả với những mẫu Botnet đã được biết và huấn luyện, cũng như có khả năng đưa ra phán đoán với các mẫu Botnet mới chưa biết dựa trên kỹ thuật học sâu. Đây là hướng tiếp cận chính mà NCS sử dụng để đề xuất giải pháp trong luận án này.

Trong phạm vi của luận án, NCS tập trung nghiên cứu vào DGA Botnet. Một số kết quả nghiên cứu chuyên sâu về bài toán DGA Botnet đã được công bố, cụ thể như sau: Các kỹ thuật dựa trên phân tích lưu lượng mạng của Alieyan và cộng sự [6], Kwon và cộng sự [7], Wang và cộng sự [8], Bisio và cộng sự [9], Trung và cộng sự [10]; Các kỹ thuật sử dụng học máy: Hiếu và cộng sự [11], Khan và cộng sự [12], Zago và cộng sự [13], Xuân và cộng sự [14]; Các kỹ thuật sử dụng học sâu: Đức và cộng sự [15], Curtin và cộng sự [16], Qiao và cộng sự [17], Vranken và cộng sự [18], Namgung và cộng sự [19], Vinayakumar và cộng sự [20].

Từ các vấn đề trên, NCS đặt ra các câu hỏi nghiên cứu cho luận án như sau: "Nghiên cứu cải tiến kỹ thuật như thế nào để tăng cường khả năng phân loại DGA Botnet dựa trên cách tiếp cận học máy, học sâu?".

2. Mục tiêu nghiên cứu

Đề tài đặt ra mục tiêu chính là nghiên cứu, cải tiến các mô hình học máy, học sâu để nâng cao độ chính xác của giải pháp phân loại DGA Botnet, với các mục tiêu cụ thể như sau:

- Trình bày nền tảng lý thuyết, các kỹ thuật, nghiên cứu liên quan, là cơ sở để phát triển các thuật toán phát hiện và phân loại DGA Botnet.

- Đề xuất kiến trúc lõi của mô hình học sâu, áp dụng xây dựng mô hình học sâu mới để nâng cao độ chính xác cho bài toán phát hiện và phân loại DGA Botnet. Trong đó, trọng tâm chính là bài toán phân loại DGA Botnet.

- Bổ sung hoàn thiện quy trình xây dựng bộ dữ liệu và đề xuất bộ dữ liệu mới công khai, tin cậy phục vụ đánh giá bài toán DGA Botnet.

3. Đối tượng và phạm vi nghiên cứu

Nghiên cứu tập trung vào các đối tượng như sau:

- Đặc điểm, cơ chế, hành vi của DGA Botnet; kỹ thuật phát hiện, phân loại Botnet dựa trên tên miền.

- Bài toán phân lớp nhị phân và phân lớp đa lớp, tương ứng với phát hiện và phân loại DGA Botnet.

- Các bộ dữ liệu công khai, tin cậy và cập nhật về DGA Botnet cùng quy trình xây dựng bộ dữ liệu mới.

Nghiên cứu của luận án tập trung vào bài toán phân loại DGA Botnet và các vấn đề liên quan trong phạm vi và thời gian thực hiện của luận án.

4. Nội dung và phương pháp nghiên cứu

a. Nôi dung nghiên cứu

Một số nội dung chi tiết mà NCS sẽ tập trung nghiên cứu như sau:

- Nghiên cứu đặc điểm, các kỹ thuật phát hiện và phân loại DGA Botnet;

- Nghiên cứu mạng LSTM, cơ chế Attention và các biến thể, trên cơ sở đó cải tiến, đề xuất mô hình học sâu mới để nâng cao hiệu quả phân loại DGA Botnet.

- Nghiên cứu về quy trình, tiêu chí, các bộ dữ liệu về DGA Botnet và áp dụng.

b. Phương pháp nghiên cứu

NCS sử dụng các phương pháp nghiên cứu bao gồm:

- Nghiên cứu lý thuyết: Thu thập, tổng hợp và đánh giá các kết quả nghiên cứu trước đó và đề xuất phương pháp lý thuyết.

- Tham khảo ý kiến chuyên gia: Tham khảo, xin ý kiến của tập thể giáo viên hướng dẫn, các thầy cô, nhà khoa học; Trao đổi, chia sẻ và tham khảo ý kiến các chuyên gia, các đơn vị chuyên về an toàn thông tin tại Học viện Khoa học và Công nghê, Trường Đại học Kỹ thuật - Hậu cần CAND, Học viện Kỹ thuật mật mã.

- Nghiên cứu thực nghiệm, đánh giá: Thực nghiệm các giải pháp phát hiện và phân loại DGA Botnet đề xuất trên các bộ dữ liệu như: Andrey Abakumov's DGA Repository, OSINT DGA feed, UMUDGA Dataset, 360NetLab Dataset.

5. Các đóng góp của luận án

Luận án có 02 đóng góp bao gồm:

Đóng góp 1: Đề xuất cải tiến kiến trúc lõi kết hợp BiLSTM với cơ chế Attention và sử dụng trong xây dựng mô hình LA_Bin07 để phát hiện và mô hình LA_Mul07 để phân loại DGA Botnet với độ chính xác được cải thiện.

Đóng góp 2: Hoàn thiện bổ sung quy trình xây dựng tập dữ liệu mẫu và đề xuất bộ dữ liệu UTL_DGA22 được mô tả và gán nhãn, phục vụ phân loại DGA Botnet.

6. Bố cục của luận án

Nội dung luận án được cấu trúc thành 04 chương, cụ thể như sau:

- *Chương 1: Cơ sở lý thuyết về DGA Botnet*

- *Chương 2: Phát hiện DGA Botnet sử dụng NCM và học máy*

- *Chương 3:Phát hiện và phân loại DGA Botnet sử dụng học sâu.*

- *Chương 4: Quy trình xây dựng và bộ dữ liệu mới UTL_DGA22 cho bài toán DGA Botnet.*

Các kết quả nghiên cứu của luận án được công bố tại 04 bài báo trên tạp chí khoa học chuyên ngành quốc tế thuộc danh mục SCIE/Scopus, 01 báo cáo tại hội thảo khoa học chuyên ngành quốc gia và 01 báo cáo tại hội thảo khoa học chuyên ngành quốc tế uy tín, được liệt kê trong phần “Danh mục các công trình công bố liên quan đến luận án” ở cuối của luận án này.

Chương 1. CƠ SỞ LÝ THUYẾT VỀ DGA BOTNET

Chương 1 trình bày cơ sở lý thuyết về Botnet nói chung và DGA Botnet nói riêng. NCS cũng trình bày hai bài toán trong DGA Botnet là phân lớp nhị phân và phân lớp đa lớp, tương ứng với bài toán phát hiện và phân loại DGA Botnet. Đây là vấn đề được NCS tập trung nghiên cứu, giải quyết và trình bày kết quả trong các chương tiếp theo của luận án này.

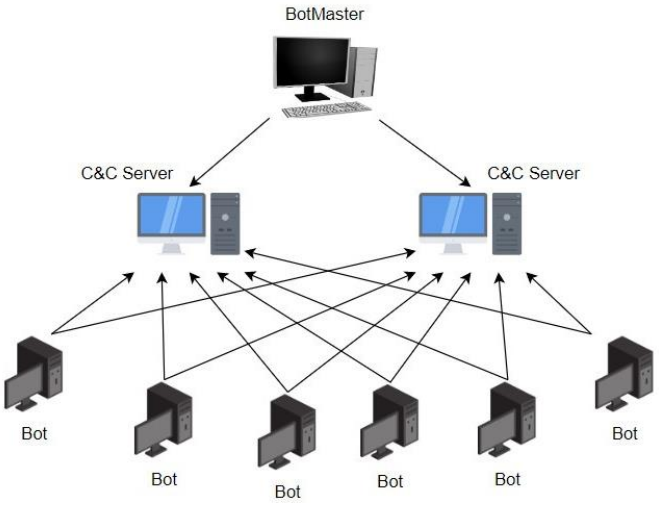
1.1. Tổng quan chung về Botnet

1.1.1. Khái niệm Botnet

Khái niệm Bot: Là một đoạn mã máy tính thực hiện một công việc hay nhiệm vụ nào đó một cách tự động, hoạt động trên máy tính cục bộ hoặc trên môi trường mạng. Các Bot thường thực hiện các nhiệm vụ đơn giản nhưng lặp lại với tốc độ cao.

Bot được phát triển lần đầu tiên trong hệ thống mạng Internet Relay Chat - IRC [21]. Giao thức IRC là một dạng truyền dữ liệu thời gian thực trên Internet, cho phép một nhóm người có thể trò chuyện với nhau thông qua một kênh truyền chung. IRC cũng hỗ trợ các cuộc trò chuyện riêng tư giữa hai máy khách và truyền tải dữ liệu. Các Bot đầu tiên đã được phát triển và sử dụng như một phương tiện để bảo vệ kênh IRC chống lại các hình thức tấn công từ chối dịch vụ (Denial of Service - DoS).

Khái niệm Botnet: Theo Provos & Holz [22], Botnet là một "mạng gồm rất nhiều máy tính bị xâm nhập và có thể bị kẻ tấn công điều khiển từ xa". Máy tính bị xâm nhập là máy tính đã bị lây nhiễm mã độc và chịu sự điều khiển bí mật của kẻ tấn công, gọi là BotMaster. Botnet tiếp nhận và thực thi lệnh từ máy chủ điều khiển C&C Server. Máy chủ này đóng vai trò là trung gian, gửi các lệnh từ kẻ tấn công hay người điều khiển tới Botnet.



Hình 1.1. Mô hình mạng Botnet với các C&C Server và BotMaster

Hình 1.1 thể hiện một mạng Botnet với BotMaster, C&C Server và các Bot. Các bước hoạt động của mạng Botnet này được mô tả như sau:

- Bước 1: Khi một Bot mới được tạo ra bằng cách lây nhiễm vào máy tính, nhiệm vụ đầu tiên của nó là tìm kiếm và báo cáo thông tin trở lại máy chủ C&C.

- Bước 2: Trong thời gian lây nhiễm, Bot sẽ ẩn mình để hạn chế tối đa khả năng phát hiện, chúng có thể trao đổi với C&C Server để báo cáo thông tin, cập nhật mã nguồn hoặc các hoạt động khác.

- Bước 3: Khi BotMaster gửi lệnh, các C&C Server sẽ chuyển tiếp và gửi chúng tới Bot để lên lịch cho hoạt động được yêu cầu.

- Bước 4: Vào thời điểm đã định, tất cả các Bot đã được nhận lệnh sẽ bắt đầu thực hiện các hành vi của mình nhắm tới mục tiêu. Các hành vi này có thể bao gồm: Gửi lưu lượng mạng độc hại, gửi tin nhắn/email rác hoặc tham gia vào cuộc tấn công từ chối dịch vụ phân tán (Distributed Denial of Service - DDoS).

- Bước 5: Bot báo cáo lại kết quả cho máy chủ C&C, chẳng hạn như đã hoàn thành nhiệm vụ và sẵn sàng cho các lệnh mới. Chúng cũng có thể bị ngắt kết nối để kết thúc vòng đời của mình hoặc bị phát hiện, bóc gỡ bởi người dùng.

Trong quá trình xây dựng và duy trì Botnet, BotMaster sẽ xây dựng cơ sở hạ tầng quản lý Bot với nhiều C&C Server được bố trí nhiều nơi khác nhau trên Internet. Các Bot sẽ lựa chọn các C&C Server theo sự chỉ định của BotMaster, hoặc theo vị trí của chúng so với C&C Server hay bất kỳ C&C Server nào khả dụng mà không bị các giải pháp bảo mật ngăn chặn.

Mỗi Bot cần kết nối tới ít nhất một C&C Server để báo cáo, nhận lệnh hay cập nhật mã nguồn mới. Một Bot cũng có thể kết nối đồng thời với nhiều C&C Server (nếu khả dụng) để tăng cường khả năng giữ liên lạc liên tục với BosMaster trong trường hợp một C&C Server bị chặn bởi các giải pháp an ninh.

Trong mô hình này, tin tặc có thể khá yên tâm rằng máy tính thực hiện các hành động tấn công không phải là máy tính của họ và C&C Server cũng không có nằm trên máy của họ, từ đó có thể hạn chế khả năng bị lộ danh tính. Để ngăn chặn Botnet, các chuyên gia, nhà quản trị mạng phải theo dõi ngược từ máy của nạn nhân đến C&C Server và cố gắng tìm ra BosMaster. Để tăng cường thêm cơ chế ẩn mình, BotMaster có thể thêm một lớp trung gian bằng cách gửi tất cả các lệnh thông qua một proxy, hoặc thông qua một loạt nhiều bước nhảy bằng cách sử dụng công cụ Tor. Thêm vào đó, một số loại Botnet còn bao gồm cả các lệnh xóa bằng chứng, lệnh mã hóa lưu lượng và các kỹ thuật ẩn mình đa hình.

Mục tiêu cuối cùng của một mạng Botnet là để thực hiện các hoạt động độc hại với quy mô lớn, tốc độ cao như phát tán tin nhắn rác, mã độc hoặc tấn công DDoS. Trước đây, các Bot được thiết kế để lây nhiễm trên các máy tính cá nhân và mạng Botnet cũng được hình thành từ những thiết bị này. Tuy nhiên hiện nay, với sự phát triển của các thiết bị Internet of Things - IoT, phạm vi của Botnet đã mở rộng hơn, không chỉ dừng lại ở việc lây nhiễm vào máy tính cá nhân, mà chúng còn có thể lây nhiễm vào các loại thiết bị IoT như tivi thông minh, tủ lạnh thông minh, camera an ninh, các cảm biến không dây. Từ đó, quy mô và sức ảnh hưởng của Botnet hiện đại cũng tăng lên nhiều lần so với mạng Botnet truyền thống.

1.1.2. Các bước phát triển về công nghệ Botnet

Bot ban đầu được thiết kế như một công cụ hữu ích để hỗ trợ cho con người. Chúng được phát triển dưới dạng một cá nhân ảo có thể đứng trên kênh IRC và làm việc tự động cho chủ sở hữu. Trải qua thời gian, Bot liên tục được phát triển, cải tiến về công nghệ, cụ thể như sau:

- Năm 1989: Greg Lindahl tạo ra GM, được xem là Bot đầu tiên [23]. GM có thể chơi trò “Hunt the Wumpus” với người dùng giao thức IRC.

- Năm 1999: Pretty Park được phát triển [23], chúng được xem là loại virus đầu tiên sử dụng hệ thống máy chủ IRC như một hệ thống điều khiển từ xa.

- Năm 1999: Phát hiện Subseven Trojan/Bot [23], là một trojan điều khiển từ xa có thêm quyền kiểm soát thông qua IRC.

- Năm 2000: GTBot, dựa trên mIRC để chạy các tập lệnh phản hồi sự kiện máy chủ IRC [24]. Bot này hỗ trợ cả giao thức TCP và UDP.

- Năm 2002: SDBot [25] được viết bằng C, dùng để khai thác dữ liệu cho cộng đồng tin tặc.

- Năm 2002: AgoBot lần đầu tiên được thiết kế dưới dạng module [23]. Gồm các module như tải xuống, ẩn mình và tấn công.

- Năm 2003: SpyBot [26] xuất hiện với khả năng tương tự với các phần mềm gián điệp.

- Năm 2003: Rbot được phát triển [26], chúng có thể vượt qua mật khẩu yếu, dễ dàng sửa đổi, sử dụng phần mềm đóng gói.

- Năm 2004: PolyBot là một biến thể của AgoBot với khả năng đa hình [23]. Chúng có thể thay đổi mã của nó trong mỗi lần lây nhiễm.

- Năm 2005: MYTOB My Doom [27], là loại sâu có khả năng gửi số lượng lớn các email rác.

- Năm 2016: Xuất hiện Botnet Mirai [28] có khả năng lây nhiễm trên các thiết bị IoT, được xem là IoT Botnet đầu tiên.

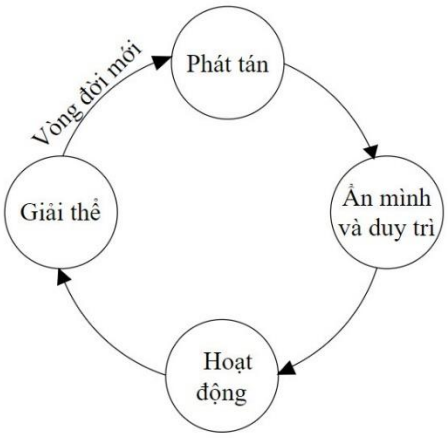
1.1.3. Một số đặc điểm của Botnet

Mạng Botnet là sự kết hợp của nhiều thành phần, thường bao gồm một vài máy máy chủ C&C và nhiều BotClient. Mạng Botnet với hàng trăm hoặc vài nghìn BotClient được coi là quy mô nhỏ. Các mạng Botnet quy mô lớn hơn có thể bao gồm vài chục hoặc vài trăm nghìn Bot.

Botnet có những đặc điểm đặc trưng về vòng đời, phương thức lây nhiễm và các hành vi độc hại, cụ thể như sau:

1.1.3.1. Vòng đời hoạt động

Vòng đời của một mạng Botnet thường trải qua các giai đoạn như Hình 1.2:



Hình 1.2. Các giai đoạn trong vòng đời của Botnet

- Phát tán: Các Bot được phát tán thông qua mã độc hoặc phần mềm chứa mã độc. Chúng cũng có thể lây nhiễm qua các kênh như tin nhắn, email, tệp tin tải về từ Internet hay giao tiếp USB.

- Ẩn mình và duy trì: Khi đã lây nhiễm vào máy tính, các Bot sẽ bí mật kết nối trở lại C&C Server để báo cáo. Chúng duy trì sự hiện diện bí mật của mình trên thiết bị của nạn nhân và cố gắng không tạo ra những dấu hiệu khác lạ. Trong giai đoạn này, chúng cũng có thể liên tục cập nhật mã nguồn mới hoặc gửi về các báo cáo mà chúng thu thập được.

- Hoạt động: Các Bot sẽ đồng loạt hoạt động vào một thời điểm và điều kiện được chỉ định bởi BotMaster thông qua C&C Server. Chúng có thể phát động một cuộc tấn công từ chối dịch vụ phân tán, phát tán tin nhắn rác hoặc thực hiện các hành vi gian lận. Trong giai đoạn này, người dùng có thể phần nào nhận ra được sự hoạt động của Bot trên thiết bị của mình.

- Giải thể: Thông thường, Bot có thể dễ bị phát hiện khi chúng đã tiến hành hoạt động cho một mục đích nào đó. Trong giai đoạn cuối cùng của vòng đời, Bot sẽ tự hủy hoạt động của nó, xóa các dấu vết trên thiết bị nạn nhân, cũng như lịch sử kết nối tới C&C Server. Điều này giúp kẻ tấn công có thể phòng ngừa việc bị lần ra dấu vết. Đồng thời cũng để giải phóng một mạng Botnet đã bị lộ. Một mạng Botnet mới sẽ được hình thành và sẽ lặp lại theo vòng đời ở trên.

1.1.3.2. Phương thức lây nhiễm

Botnet có thể sử dụng các phương thức khác nhau để lây nhiễm trên máy tính. Đồng thời, một số cũng có khả năng sao chép chính mình từ thiết bị này sang thiết bị khác. Jose Nazario đã liệt kê một số đặc trưng về phương thức lây nhiễm của Botnet, cụ thể như sau [29]:

- Thông qua email: Các email được gửi đính kèm các tệp tin chứa mã độc hại. Nếu người dùng mở những email này và thực thi các mã đó thì máy tính của họ sẽ bị lây nhiễm mã độc. Trước đây, hình thức này tương đối dễ bị phát hiện bởi những dấu hiệu như email đến từ địa chỉ không tin cậy, tệp tin đính kèm có nhận dạng là độc hại. Hiện nay, kỹ thuật phát tán ngày càng trở nên tinh vi hơn khi các email được giả mạo được gửi từ các công ty hay tổ chức uy tín. Đồng thời, khi một máy bị nhiễm thì danh sách địa chỉ liên hệ của người dùng trên máy đó sẽ bị đọc và các email độc hại này lại tiếp tục được gửi đến những địa chỉ liên hệ đó.

- URL độc hại: Là những URL điều hướng người sử dụng đến các website chứa mã độc hại. Tin tặc có thể gửi liên kết này thông qua email, mạng xã hội, tin nhắn hay sử dụng kỹ nghệ xã hội để thuyết phục người dùng rằng nó đáng tin cậy.

- Website giả mạo: Các trang web này thường được thiết lập để giả mạo các website nổi tiếng như Facebook, Youtube hoặc một trang web đáng tin cậy nào đó. Có hai kiểu tấn công chính là tấn công phía Client và khai thác tải về:

+ Trong hình thức tấn công phía Client, khi người dùng truy cập vào trang web, các mã độc hại sẽ được khởi động và cố gắng lợi dụng các lỗ hổng bảo mật trên trình duyệt để có thể truy cập vào máy tính. Nếu lỗ hổng này được lợi dụng thành công, máy tính của người dùng sẽ bị lây nhiễm và trở thành một Bot.

+ Dạng tấn công dựa trên khai thác tải về là khi truy cập vào một website, người dùng sẽ được nhắc nhở để tải về một tập tin. Nếu người dùng chấp nhận tải về và thực thi chúng, máy tính sẽ có khả năng bị lây nhiễm mã độc hại được đính kèm trong tập tin này.

1.1.3.3. Các hành vi độc hại

Một số hành vi độc hại của các Bot bao gồm:

- Gửi tin nhắn hoặc thư rác: Các mạng Botnet có thể gửi một lượng lớn các tin nhắn hoặc thư rác, gây tiêu tốn băng thông mạng, phiền toái và các nguy cơ lừa đảo, đánh cắp thông tin người dùng.

- Tấn công từ chối dịch vụ phân tán: Các mạng Botnet lớn thường được sử dụng để phát động một cuộc tấn công từ chối dịch vụ phân tán. Đây là một đặc điểm phổ biến của Botnet giúp phân biệt chúng với các phần mềm mã độc khác.

Một số vụ tấn công nổi bật như: Cuộc tấn công vào Amazon Web Services năm 2022 [30]; Tấn công nhắm vào Brian Krebs và OVH năm 2016 với tốc độ 620 Gbps [31]. Lúc đó, đây là cuộc tấn công DDoS lớn nhất từng được ghi nhận.

- Do thám người dùng: Các Bot khi nhiễm vào máy tính có thể bí mật thu thập thông tin của người dùng, như lịch sử gõ phím hoặc các tệp tin lưu trữ trên đó.

- Gian lận Click: Bot có thể đóng vai trò giả mạo con người để truy cập vào quảng cáo hoặc tương tự. Trong trường hợp này, việc thực hiện tự động trên giúp kẻ tấn công thu được lợi nhuận bất chính khi giả danh con người.

- Đánh cắp bản quyền: Một số họ Botnet có khả năng thu thập và đánh cắp bản quyền phần mềm trên máy tính mà nó lây nhiễm.

- Phát tán mã độc: Bot tự bản thân nó cũng có thể tiếp tục phát tán mã nguồn để lây nhiễm tới nhiều máy hơn, từ đó có thể mở rộng mạng lưới của chúng.

Một số giải pháp để phòng ngừa và ngăn chặn Botnet bao gồm: Phòng ngừa sự lây nhiễm của Botnet vào máy tính cá nhân và thiết bị IoT, cài đặt các phần mềm chống virus cho máy tính cá nhân, có phương án dự phòng về máy chủ, băng thông cho các hệ thống thông tin, ứng dụng các giải pháp phòng chống và điều tra tấn công DDoS từ các nhà cung cấp dịch vụ.

1.1.4. Phân loại Botnet

Botnet có thể được phân loại theo các tiêu chí như: Giao thức, thiết bị lây nhiễm hoặc kiến trúc.

1.1.4.1. Phân loại Botnet theo giao thức

Giao thức là phương thức gửi nhận giữa Bot và C&C Server. Botnet thường sử dụng các giao thức phổ biến là giao thức truyền tải siêu văn bản HTTP - HyperText Transfer Protocol, HTTPS - Hypertext Transfer Protocol Security và giao thức IRC.

- IRC Botnet: Đây là loại Botnet hoạt động trên giao thức IRC. Hầu hết mọi IRC Server đều cho phép truy cập miễn phí, không kể đối tượng sử dụng. Các IRC Botnet sử dụng kênh IRC để liên lạc, phát tán và thực hiện hành vi độc hại.

- HTTP Botnet: Botnet sử dụng giao thức HTTP cũng hoạt động theo mô hình Client-Server. Tuy nhiên, thay vì nhận lệnh thông qua kênh chat thì HTTP Botnet sẽ sử dụng giao thức HTTP để gửi các yêu cầu và nhận lệnh từ Bot Master qua C&C Server. HTTP Botnet thường không nhận lệnh theo thời gian thực mà chúng sẽ gửi yêu cầu liên tục hoặc qua một khoảng thời gian nào đó để cập nhật các dữ liệu mới. Các loại Botnet hiện đại hơn sử dụng giao thức HTTPS để tăng cường khả năng che dấu của mình.

1.1.4.2. Phân loại Botnet theo thiết bị lây nhiễm

Thiết bị lây nhiễm là mục tiêu mà Bot có khả năng hoặc được thiết kế để lây nhiễm. Các thiết bị này có điểm chung là có hệ điều hành, có kết nối Internet và tồn tại các lỗ hổng có thể bị khai thác bởi mã độc.

- Botnet truyền thống: Đây là các Bot được thiết kế để lây nhiễm trên máy tính cá nhân của người dùng, thông thường là chạy các hệ điều hành họ Windows.

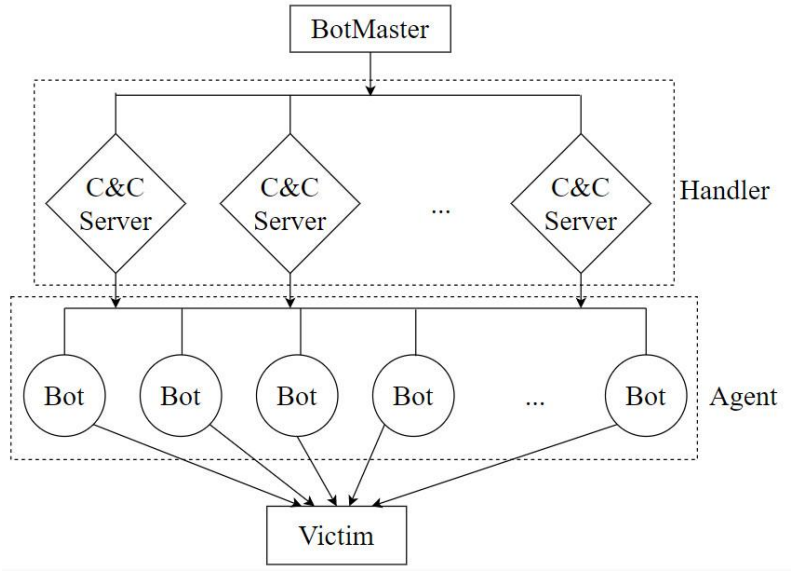
- Mobile Botnet: Là dạng Botnet được thiết kế để lây nhiễm trên các thiết bị di động thông minh như điện thoại thông minh, máy tính bảng thông qua Internet. Các Bot này thường đi kèm với các ứng dụng độc hại được cài lên thiết bị, thông thường là sử dụng hệ điều hành Android hoặc iOS.

- IoT Botnet: Đây là dạng Botnet lây nhiễm trên các thiết bị IoT, như các cảm biến, thiết bị gia đình thông minh, camera an ninh. Sự bùng nổ về số lượng của các thiết bị IoT cùng với năng lực xử lý ngày càng cao, thậm chí tiệm cận đến năng lực xử lý của máy tính cá nhân, cho phép các mạng IoT Botnet có thể phát triển với quy mô lớn hơn rất nhiều so với các mạng Botnet truyền thống, đồng nghĩa với việc tạo ra các cuộc tấn công có sức ảnh hưởng lớn hơn.

1.1.4.3. Phân loại theo kiến trúc

Có bốn loại kiến trúc của Botnet: Kiến trúc Agent-Handler, kiến trúc IRC-based, kiến trúc Peer-to-Peer, và các kiến trúc lai tiên tiến [32].

- Kiến trúc Agent-Handler (Hình 1.3): Trong đó, Agent là các Bot còn Handler là hệ thống chỉ huy và kiểm soát hay là C&C Server. Lớp Handler đóng vai trò trung gian giữa Bot và BotMaster.



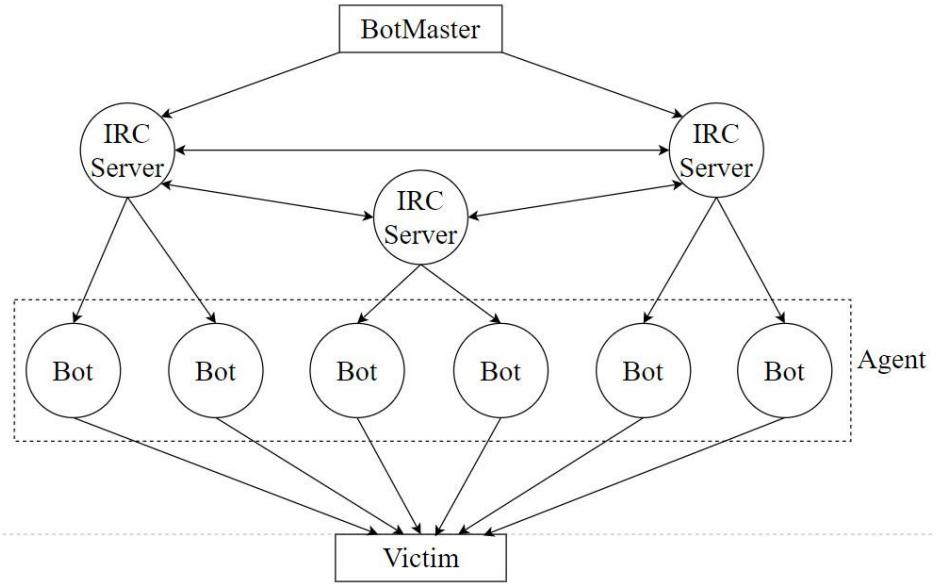
Hình 1.3. Kiến trúc Agent-Handler của Botnet

Kẻ tấn công giao tiếp với các C&C Server để thiết lập các cài đặt và kiểm soát hệ thống. Đây thường là một máy chủ mạnh mẽ với rất nhiều tài nguyên về băng thông, bộ nhớ và sức mạnh xử lý. Ngoài việc nhận lệnh từ kẻ tấn công, C&C Server còn có trách nhiệm theo dõi các Bot và gửi lệnh bao gồm cấu hình, cập nhật mới.

Chủ sở hữu của hệ thống máy tính bị xâm nhập thường không có biết rằng các phần mềm độc hại đã được cài đặt trong máy tính của họ hay họ là một phần của Botnet. Những kẻ tấn công sử dụng Bot như một bàn đạp để khởi động các cuộc tấn công chống lại mục tiêu.

Kiến trúc Agent-Handler có một hạn chế là kẻ tấn công phải có khả năng giao tiếp với các C&C Server cũng như C&C Server phải có khả năng liên lạc với các Bot. Nếu các kết nối trên bị gián đoạn thì kẻ tấn công có thể mất kiểm soát với một C&C Server, cũng như một C&C Server sẽ không kiểm soát được các Bot mà nó phụ trách. Điều này có thể dẫn đến việc không thể thiết lập cho các Bot để tấn công một mục tiêu mới.

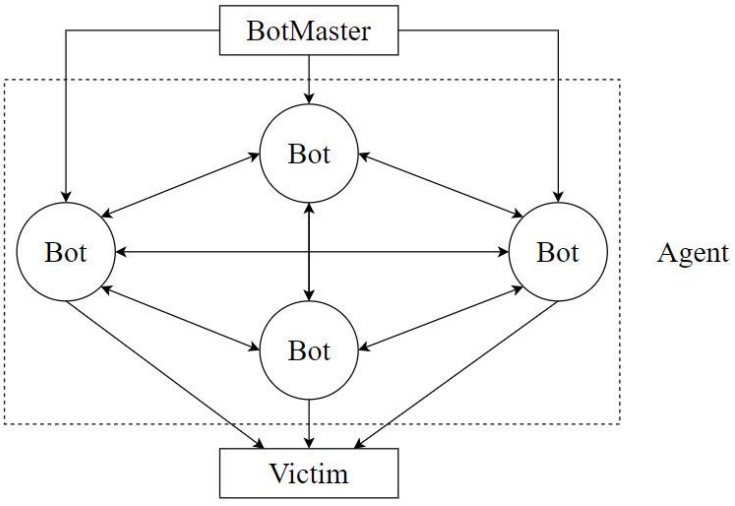
- Kiến trúc IRC-Based (Hình 1.4): Kiến trúc Botnet Internet Relay Chat giải quyết các giới hạn trong kiến trúc Agent-Handler. Botnet IRC-Based thay thế các Handlers bằng các Public IRC Server.



Hình 1.4. Kiến trúc IRC-Based của Botnet

Khi các Bot đã được triển khai, mỗi Bot sẽ kết nối đến một máy chủ IRC và chờ lệnh. Kẻ tấn công ra lệnh cho các Bot thông qua các kênh IRC. Nó cho phép mỗi Bot khởi đầu một hoặc cả hai hình thức tấn công. Nó cũng tạo nên một lớp phức tạp để che giấu các dấu vết của BotMaster. Thông tin liên lạc giữa BotMaster và các máy chủ IRC có thể được mã hóa. Khác biệt chính giữa các kiến trúc dựa trên IRC và kiến trúc Agent-Handler là cấu trúc điều khiển và thông tin liên lạc.

- Kiến trúc Peer-to-Peer (Hình 1.5): Không giống như kiến trúc Agent-Handler và IRC-based, cấu trúc Peer-to-Peer không có C&C Server riêng biệt. Lệnh được gửi đến các Bot bằng giao thức P2P. Mỗi Bot không chỉ chịu trách nhiệm cho việc chuyển tiếp lệnh tấn công mà còn là một phần của cơ cấu chỉ huy và kiểm soát để quản lý các Bot khác. Như vậy, kiến trúc P2P Botnet khó để đánh sập vì sự phân bố tự nhiên rất cao của nó.



Hình 1.5. Kiến trúc Peer-to-Peer của Botnet

Ngoài việc phân phối lệnh, các kênh truyền thông P2P cũng được sử dụng để phân phối các phiên bản mới của phần mềm Bot, tải về công cụ tấn công mới hoặc một danh sách các mục tiêu mới. Việc phát hiện các thông tin liên lạc này là khó hơn nhiều vì sự phân tán của chúng, đồng thời dữ liệu cũng được mã hóa.

- Kiến trúc lai Peer-to-Peer tiên tiến: Kiến trúc Botnet lai Peer-to-Peer hoạt động với vai trò như cả máy khách và máy chủ trong một hệ thống chia sẻ tệp tin P2P truyền thống. Kẻ tấn công có thể chèn câu lệnh của mình vào bất kỳ máy chủ nào của các Botnet này. Mỗi máy chủ định kỳ sẽ kết nối với các Bot để cập nhật thông tin mới. Khi một lệnh mới xuất hiện, máy chủ sẽ chuyển lệnh này cho tất cả các Bot gần đó. Kiến trúc như vậy có các ưu điểm là một Bot chỉ biết một danh sách hạn chế của các bot xung quanh. Do đó, ngay cả khi Bot này bị phát hiện, những người điều tra cuộc tấn công cũng chỉ có thể có được danh sách hạn chế của các Bot, mà không phải là danh sách đầy đủ của toàn bộ mạng lưới. Mô hình này cũng giúp kẻ tấn công dễ dàng quản lý hoặc huy động toàn bộ mạng Botnet của mình bằng cách phát đi một lệnh duy nhất.

1.2. Kỹ thuật phát hiện Botnet

Hiện nay, có kỹ thuật chính được sử dụng để phát hiện Botnet [5]:

- (1) Các kỹ thuật dựa trên honeynet (mạng bẫy tin tặc).

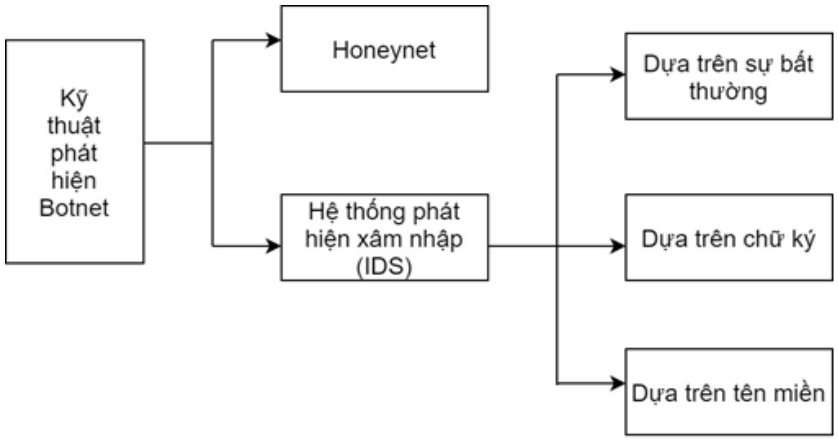
- (2) Các kỹ thuật dựa trên hệ thống phát hiện xâm nhập (Intrusion Detection System - IDS), bao gồm:

+ Phát hiện Botnet dựa trên sự bất thường.

+ Phát hiện Botnet dựa trên chữ ký.

+ Phát hiện Botnet dựa trên tên miền.

Khái quát các kỹ thuật trên được trình bày tại Hình 1.6:



Hình 1.6. Khái quát về các kỹ thuật phát hiện Botnet

1.2.1. Kỹ thuật phát hiện Botnet sử dụng HoneyNet

1.2.1.1. Khái niệm HoneyNet

Honeynet là một hệ thống thông tin được xây dựng với mục đích giả dạng đánh lừa những tin tặc và các hành vi xâm nhập không hợp pháp. Honeynet thu hút sự chú ý và bí mật truy tìm thông tin của mục tiêu, đồng thời ngăn không cho chúng tiếp xúc với hệ thống thật.

Honeynet có thể giả dạng bất cứ loại máy chủ tài nguyên nào như là Mail Server, Domain Name Server, Web Server… Honeypot là một điểm trong HoneyNet sẽ trực tiếp tương tác với tin tặc và tìm cách khai thác thông tin về chúng như hình thức tấn công, công cụ tấn công hay cách thức tiến hành cuộc tấn công đó.

Honeynet gồm hai loại chính là tương tác thấp và tương tác cao:

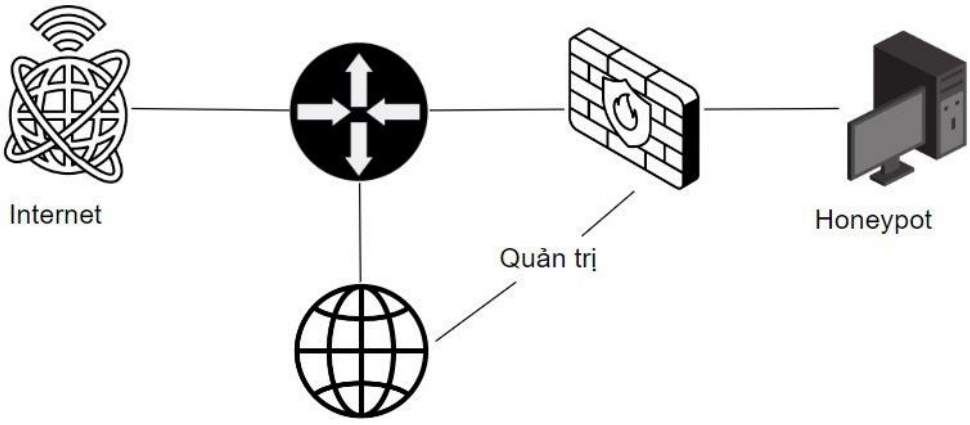
- Tương tác thấp (Low Interaction): Mô phỏng giả lập các dịch vụ, ứng dụng và hệ điều hành. Mức độ rủi ro thấp, dễ triển khai và bảo dưỡng nhưng bị giới hạn về dịch vụ.

- Tương tác cao (High Interaction): Là các dịch vụ, ứng dụng và hệ điều hành thực. Mức độ thông tin thu thập được cao. Nhưng rủi ro cũng cao tương ứng và tốn nhiều thời gian để vận hành, bảo dưỡng.

1.2.1.2. Mô hình phát hiện Botnet dựa trên Honeynet

Từ ý tưởng đánh lừa kẻ tấn công ở trên, mạng Honeynet phát hiện Botnet được xây dựng để thu thập thông tin chi tiết về Botnet, như nguồn gốc của máy chủ C&C, các thành viên trong mạng hay các hành vi tấn công của chúng.

Mô hình xây dựng hệ thống Honeynet để phát hiện Botnet bao gồm Honeywall và Windows Honeypot, được minh họa tại Hình 1.7:



Hình 1.7. Mô hình mạng HoneyNet để phát hiện Botnet

Trong đó:

- Windows Honeypot: Là một hoặc nhiều máy tính cá nhân, được cài đặt các phiên bản hệ điều hành cũ hoặc chưa được vá lỗi kịp thời, nhằm tạo cơ hội cho Botnet lây nhiễm. Đây là sẽ mục tiêu ưu thích của các mạng Botnet.

- Honeywall: Đóng vai trò tương tự như tường lửa, nhưng thay vì mục đích ngăn chặn các cuộc tấn công của Botnet, thiết bị này được cấu hình để cho phép các Botnet đi qua và lây nhiễm theo như cách mà chúng được thiết kế. Trong quá trình đó, Honeywall sẽ theo dõi và phân tích các hành vi, đặc trưng và chữ ký của Botnet, từ đó nhận được tri thức về Botnet để cập nhật cho Firewall trên các hệ thống thật sự.

1.2.2. Kỹ thuật phát hiện Botnet sử dụng hệ thống phát hiện xâm nhập

1.2.2.1. Phát hiện Botnet dựa trên sự bất thường

Kỹ thuật này thông qua việc phân tích lưu lượng mạng và hoạt động của máy tính để phát hiện sự cố bất thường như sự gia tăng đột ngột lưu lượng truy cập, lưu lượng đến cổng dịch vụ, độ trễ mạng cao, máy tính chịu tải cao và những dấu hiệu tương tự khác. Những bất thường này được sử dụng để so sánh với trạng thái bình thường của hệ thống, từ đó phát hiện ra những mối đe dọa đã làm thay đổi hiện trạng hệ thống.

Một số hoạt động bất thường bao gồm:

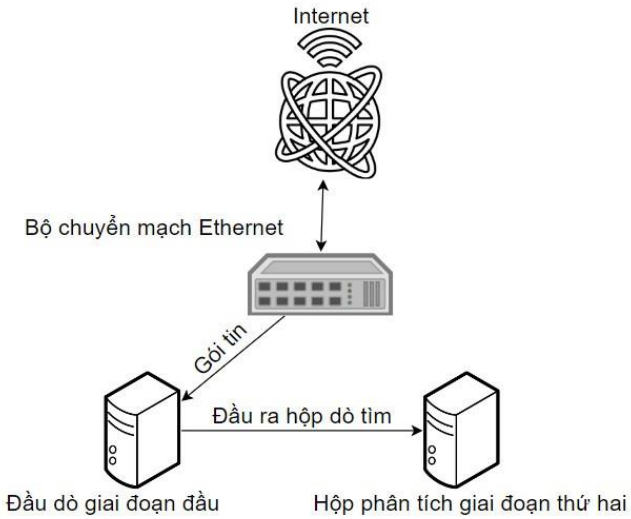
- Phân tích các luồng dữ liệu mạng để phát hiện các hoạt động không bình thường như: Các kết nối đến các cổng không thường xuyên hoặc kích thước gói tin lớn hơn mức bình thường.

- Trạng thái máy tính: Việc phát hiện Botnet cũng có thể được thực hiện bằng cách theo dõi trạng thái của các máy tính trong mạng, bao gồm tình trạng tài nguyên của máy tính như sử dụng bộ nhớ, bộ vi xử lý và tốc độ đọc/ghi đĩa.

- Hành vi người dùng: Nếu có một số lượng lớn các truy cập vào cùng một tài khoản trong một khoảng thời gian ngắn, thì có thể đây là một tấn công giả mạo được thực hiện bởi các Bot.

Hình 1.8 minh họa một mô hình đơn giản cho quá trình theo dõi sự bất thường. Với lưu lượng Internet đi vào, quản trị viên có thể gắn đầu dò giai đoạn đầu vào bộ chuyển mạch Ethernet để dò gói tin. Đầu ra của hộp dò tìm được đưa tới Hộp phân tích giai đoạn thứ hai để thực hiện các chức năng như: Ghi nhật ký dữ liệu, phân tích và mô hình hóa lưu lượng. Bằng cách so sánh các thông số hiện tại với thông số bình

thường, hệ thống có thể nhận biết các bất thường và xác định các nguy cơ tương ứng. Một số công cụ giám sát mạng thường được sử dụng như SNMP hay Netflow.



Hình 1.8. Minh họa cơ sở hạ tầng giám sát mạng

Hạn chế của kỹ thuật phát hiện Botnet dựa trên sự bất thường đó là nó có thể tạo ra nhiều cảnh báo giả. Hơn nữa, nó cũng không phát hiện được những Botnet có hoạt động rất giống như hoạt động bình thường của hệ thống mạng.

1.2.2.2. Phát hiện Botnet dựa trên chữ ký

Phát hiện Botnet dựa trên chữ ký là một cách tiếp cận truyền thống, có thể được áp dụng không chỉ với Botnet mà còn các loại virus, mã độc hay các phần mềm độc hại khác. Kỹ thuật này dựa trên cơ sở người quản trị đã có những tri thức về Botnet, họ thu thập và tổng hợp được một cơ sở dữ liệu các thông tin về Botnet và tạo nên khái niệm gọi là chữ ký của chúng, hay nói cách khác là những đặc trưng đã biết của Botnet. Chữ ký của Botnet có thể được thể hiện dưới nhiều dạng khác nhau. Chúng có thể đơn giản chỉ là một địa chỉ IP hay một chuỗi giá trị băm, hoặc cũng có thể phức tạp hơn như là số lượng byte NULL xuất hiện sau một chuỗi xác định khi sử dụng một giao thức nào đó.

Khi áp dụng trong thực tế, kỹ thuật này tiến hành đối sánh các mẫu thu thập được với các mẫu đã biết để từ đó quyết định xem mẫu thu thập được có phải là một dạng Botnet đã biết hay không.

Các cơ sở dữ liệu có thể chứa các chữ ký về Botnet, hay là chữ ký của các phần mềm, công cụ lành tính. Nói chung, mục tiêu của phương pháp này là nhận dạng nguy cơ dựa trên những tri thức đã biết. Hạn chế của nó là kém hiệu quả để phát hiện được các mối đe dọa mới, bởi vì cơ sở dữ liệu chữ ký luôn theo sau sự xuất hiện của các loại Botnet mới.

1.2.2.3. Phát hiện Botnet dựa trên tên miền

Nhiều họ Botnet khi lây nhiễm thành công vào máy tính của nạn nhân thì kết nối trở lại máy chủ thông qua những tên miền được sinh tự động, được gọi là DGA Botnet Những tên miền này có đặc điểm tương tự nhau và tuân theo cùng một thuật toán sinh đối với từng họ Botnet. Kỹ thuật phát hiện Botnet dựa trên tên miền vận dụng cơ chế trên để phát hiện DGA Botnet. Nhà quản trị sẽ thu thập, giám sát lưu lượng DNS, từ đó có khả năng phát hiện ra các họ DGA Botnet đã bị lây nhiễm ở trong mạng, đồng thời gián tiếp tìm được địa chỉ của C&C Server.

Đầu vào của các thuật phát hiện DGA Botnet là các tên miền, bao gồm nhãn độc hại và nhãn lành tính. Nhãn lành tính là các tên miền thông thường, còn nhãn độc hại là các tên miền được sinh ra bởi DGA Botnet. Có sự khác nhau cho phép phân biệt được giữa hai nhóm tên miền này. Trong một số trường hợp, các nhãn độc hại tiếp tục được phân loại nhỏ hơn để nhận diện từng họ Botnet nói riêng.

Một số kỹ thuật có thể được sử dụng để phát hiện DGA Botnet như sau:

- Kỹ thuật đối sánh tên miền: Kỹ thuật này đơn giản nhất và tương tự như việc phát hiện Botnet dựa trên chữ ký.

- Sử dụng các mô hình học máy, các thuật toán phân cụm, các kỹ thuật dựa trên xác suất thống kê… Thực tế cho thấy các mô hình này có độ chính xác khá tốt.

- Sử dụng học sâu: Sự phát triển trong năng lực của vi xử lý giúp cho các mô hình học sâu có thể chạy nhanh hơn, hiệu quả hơn, đưa công nghệ này trở nên khả thi hơn trong nhiều lĩnh vực, trong đó có phát hiện DGA Botnet.

Ngoài ra, bên cạnh yếu tố tên miền, các yếu tố khác của một họ Botnet như tần suất, số lượng tên miền/đơn vị thời gian cũng có thể sử dụng để nâng cao hiệu quả phân loại Botnet. Đây cũng là một hướng nghiên cứu hứa hẹn trong tương lai với các giải pháp phân loại sử dụng đa yếu tố.

1.3. Bài toán DGA Botnet

Hai bài toán thường được nghiên cứu trong vấn đề DGA Botnet là bài toán phát hiện (phân lớp nhị phân) và bài toán phân loại (phân lớp đa lớp) DGA Botnet.

1.3.1. Khái quát về DGA Botnet

1.3.1.1. Vấn đề truy vấn C&C Server

Trong vòng đời Botnet, hoạt động truyền thông giữa Bot với C&C Server đóng vai trò rất quan trọng. Hoạt động này giúp Bot gửi các báo cáo, cập nhật mã nguồn

hoặc nhận lệnh từ máy chủ C&C. Các Bot cần có địa chỉ IP để kết nối với máy chủ C&C. Dễ thấy rằng, việc thiết lập một địa chỉ IP cố định cho C&C Server trong mã nguồn của Bot là không thực tế. Bởi vì nhà quản trị có thể dễ dàng phát hiện và ngăn chặn chúng thông qua các giải pháp bảo mật như tường lửa, hệ thống phát hiện và ngăn chặn xâm nhập. Đồng thời, máy chủ C&C cũng dễ dàng bị lộ và có nguy cơ bị điều tra.

Để giải quyết vấn đề trên, Bot sử dụng truy vấn tên miền như là một giải pháp hiệu quả để tìm đến địa chỉ máy của C&C. Ở những phiên bản Bot đầu tiên, có hai kỹ thuật đơn giản thường được sử dụng là Domain Name và Multihoming [27]:

- Kỹ thuật Domain Name sử dụng một vài tên miền để cùng trỏ đến một địa chỉ IP đích. Ví dụ:

```
domain01.com pointing to 192.168.1.1
domain02.com pointing to 192.168.1.2
domain03.com pointing to 192.168.1.2
domain04.net pointing to 192.168.1.2
domain05.net pointing to 192.168.1.3
domain06.net pointing to 192.168.1.3
```

Ta thấy rằng, địa chỉ IP là 192.168.1.2 được trỏ đến bởi ba tên miền khác nhau, bao gồm domain02.com, domain04.com và domain04.net.

- Kỹ thuật Multihoming cho phép nhiều tên miền cùng trỏ đến một địa chỉ IP đích. Khi đó, nếu một trong các địa chỉ IP không còn khả dụng, tên miền vẫn tìm được địa chỉ IP khác đang hoạt động để kết nối. Đối với Botnet, các địa chỉ IP này là các C&C Server.

```
domain01.com pointing to 192.168.1.1
domain01.com pointing to 192.168.1.2
domain01.com pointing to 192.168.1.3
domain01.com pointing to 192.168.1.4
domain01.com pointing to 192.168.1.5
domain01.com pointing to 192.168.1.6
```

Trong ví dụ trên, ta thấy rằng tên miền domain01.com trỏ đến các địa chỉ IP gồm 192.168.1.1 đến 192.168.1.6. Giả sử kết nối đến C&C Server đặt tại địa chỉ 192.168.1.1 bị chặn, Bot vẫn có thể sử dụng tên miền trên để truy cập đến các địa chỉ IP khác như 192.168.1.2, 192.168.1.3..., nơi có các C&C Server khác dự phòng.

Nhìn chung, tuy có ưu điểm nhưng đây cũng chưa phải là một giải pháp hiệu quả trong thực tế bởi tính đơn giản và dễ bị nhận biết của nó.

1.3.1.2. Giải pháp truy vấn sử dụng DGA

Dịch vụ phân giải tên miền (DNS – Domain Name Service) là một dịch vụ phổ biến trên mạng Internet, cho phép phân giải tên máy, hoặc tên miền sang địa chỉ IP và ngược lại. Dịch vụ DNS được các Bot sử dụng để qua mặt hệ thống bảo vệ với kỹ thuật sinh tên miền tự động.

DGA - Domain Generation Algorithm là thuật toán sinh tên miền tự động. DGA Botnet là những Botnet sử dụng phương thức truy vấn tên miền được sinh tự động theo thuật toán để tìm địa chỉ IP của máy chủ điều khiển.Giải pháp truy vấn địa chỉ IP thông qua thuật toán sinh tên miền sinh tự động gọi là DGA được áp dụng.

Trong mô hình này, các máy chủ C&C và Bots sẽ cùng nhau thống nhất một thuật toán sinh tên miền. Vào mỗi thời điểm, các tên miền được sinh ra sẽ trỏ đến một địa chỉ IP được quy định bởi BotMaster. Các Bot và C&C Server sẽ tuân theo thuật toán này để thay đổi IP tương ứng, giúp chúng vẫn tìm thấy địa chỉ của nhau nhưng có thể qua mặt được các hệ thống kiểm soát. Giải pháp này cải tiến hơn nhiều so với các giải pháp trước đó, cho phép các Bot có thể truy vấn tới máy chủ với tỉ lệ bị phát hiện thấp hơn. Đồng thời, C&C Server cũng có thể dễ dàng ẩn mình hoặc thay đổi địa chỉ IP theo thời gian mà vẫn cho phép các Bot tìm thấy được chúng.

1.3.1.3. Khái niệm DGA Botnet

DGA Botnet là khái niệm chỉ một dạng Botnet được triển khai theo mô hình Client-Server. Trong đó, các Bot đóng vai trò là Client sẽ liên kết trở lại máy chủ C&C - đóng vai trò là Server - thông qua các tên miền DNS được sinh một cách tự động. Các tên miền này được sinh dựa trên quy tắc mà BotMaster quy định và khác với các tên miền lành tính. Việc phát hiện các tên miền này là khó hơn nhiều đối với các giải pháp an ninh.

Các thuật toán sinh tên miền được thiết kế để cố gắng sinh ra các tên miền có thể qua mặt được các hệ thống bảo mật. Thuật toán DGA có thể sử dụng các toán tử kết hợp với các biến luôn thay đổi chẳng hạn như năm, tháng, ngày, giờ, phút để sinh tên miền ngẫu nhiên. Ví dụ, một dạng của thuật toán DGA được thực hiện bởi một hàm có chứa 16 vòng lặp, mỗi vòng lặp sinh ngẫu nhiên một ký tự trong tên miền như sau [27]:

```
year = ((year ^ 8 * year) >> 11) ^ ((year & 0xFFFFFFF0) << 17)
month = ((month ^ 4 * month) >> 25) ^ 16 * (month & 0xFFFFFFF8)
```

```
day = ((day ^ (day << 13)) >> 19) ^ ((day & 0xFFFFFFFE) << 12)
domain += chr(((year ^ month ^ day) % 25) + 97)
```

Trong đó:

- Toán tử '^': Phép mũ.

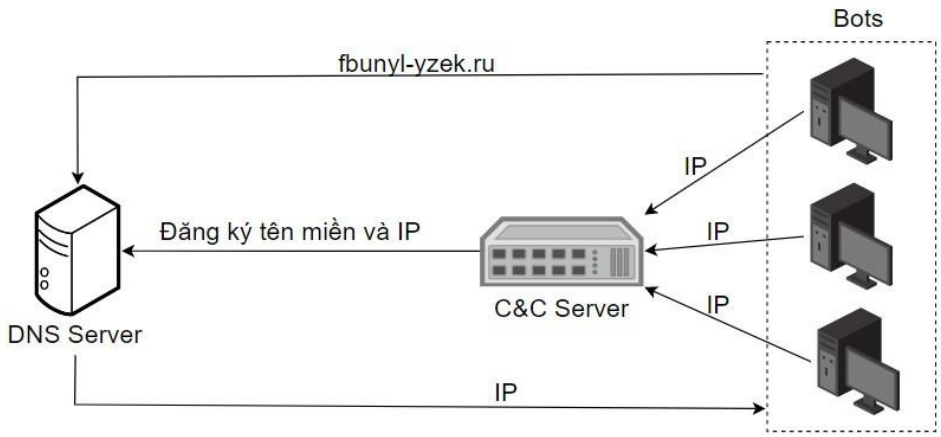
- Toán tử '*': Phép nhân.

- Toán tử '%': Phép chia lấy phần dư.

- Toán tử '>>': Phép dịch bit sang phải.

- Toán tử '<<': Phép dịch bit sang trái.

- Toán tử '&': Phép AND.



Hình 1.9. DGA Botnet sử dụng thuật toán sinh để sinh và đăng ký các tên miền cho máy chủ C&C

Hình 1.9 biểu diễn cơ chế Botnet sử dụng thuật toán DGA để tự động sinh và đăng ký các tên miền cho máy chủ C&C. Theo đó, máy chủ C&C và các Bot sử dụng cùng một thuật toán DGA nên chúng có thể sinh ra cùng một tập các tên miền. DGA Botnet thường sử dụng ngày giờ như là nhân để khởi tạo thuật toán sinh tên miền.

Để khởi tạo kết nối đến máy chủ C&C, một Bot trước hết cần thực thi thuật toán DGA để sinh một tên miền và tên miền này cũng có thể được sinh tự động bởi máy chủ C&C và chung một thời gian. Sau khi tạo được tên miền, Bot sử dụng hệ thống DNS để phân giải tên miền thành địa chỉ IP của máy chủ C&C. Nếu quá trình phân giải tên miền không thành công, Bot sử dụng DGA để sinh một tên miền mới và lặp lại yêu cầu. Nếu quá trình phân giải tên miền thành công, Bot sử dụng địa chỉ IP để kết nối đến máy chủ C&C để nhận các lệnh và điều khiển từ BotMaster.

Việc giám sát và phân tích dữ liệu truy vấn DNS, đặc biệt là các tên miền và kết quả truy vấn có thể tiết lộ sự tồn tại của các hành vi độc hại trong hệ thống mạng do

Botnet tạo ra. Do đó, nếu có thể phát hiện và ngăn chặn các truy vấn tên miền này, ta có thể gián tiếp xác định được địa chỉ IP của C&C Server, cô lập và hạn chế các hoạt động của Bot kể cả khi chúng đã lây nhiễm vào máy tính.

1.3.2. Bài toán phát hiện DGA Botnet

Bài toán phát hiện DGA Botnet - Phân lớp nhị phân: Là bài toán với mục tiêu phát hiện các tên miền được sinh ra bởi DGA Botnet trong số tất cả các tên miền được truy vấn trong mạng. Bài toán này được thực hiện trên bộ dữ liệu gồm có hai nhãn là 0 và 1. Trong đó, một tên miền lành tính sẽ được gán nhãn 0 và tên miền của Botnet sẽ được gán nhãn 1.

Minh họa về tên miền và nhãn tương ứng trong bài toán phát hiện DGA Botnet được trình bày tại Bảng 1.1:

Bảng 1.1. Minh họa dữ liệu và nhãn của bài toán phát hiện DGA Botnet

Tên miền	Nhãn	Giải thích
google.com.vn	0	google.com.vn là một tên miền lành tính
facebook.com	0	facebook.com là một tên miền lành tính
ofdhiydrrttpblp.com	1	ofdhiydrrttpblp.com là một tên miền độc hại được sinh ra bởi DGA Botnet
eimgukowkqeckykg.org	1	eimgukowkqeckykg.org là một tên miền độc hại được sinh ra bởi DGA Botnet

1.3.3. Bài toán phân loại DGA Botnet

Bài toán phân loại DGA Botnet - Phân lớp đa lớp: Là bài toán nhằm mục tiêu phát hiện họ của DGA Botnet. Bài toán này có đầu vào là các tên miền đã được gán nhãn là DGA Botnet. Công việc phân lớp đa lớp được thực hiện trên bộ dữ liệu gồm có n nhãn, tương ứng với n họ DGA Botnet được xem xét. Các nhãn được đánh số i có giá trị từ 1 đến n, với i tương ứng là họ của DGA Botnet.

Bảng 1.2 minh họa về tên miền và nhãn tương ứng của DGA Botnet trong bài toán phân loại DGA Botnet, với 03 nhãn được thể hiện.

Bảng 1.2. Minh họa dữ liệu và 03 nhãn trong bài toán phân loại DGA Botnet

Tên miền	Nhãn	Họ DGA Botnet	Giải thích
qkdchxxxzn.com meojytusps.com	1	alureon	Đây là tên miền DGA Botnet thuộc về họ alureon

ofdhiydrrttpblp.com aaifkidvjspwc.biz	2	cryptolocker	Đây là tên miền DGA Botnet thuộc về họ cryptolocker
eimgukowkqeckykg.org uowgugsysycoqeqs.org	3	ramdo	Đây là tên miền DGA Botnet thuộc về họ ramdo

1.3.4. Phân biệt với bài toán phát hiện URL giả mạo

Bài toán phát hiện DGA Botnet có những nét tương đồng với bài toán phát hiện các Uniform Resource Locator - URL giả mạo. Tuy nhiên chúng lại có những đặc điểm khác nhau trong cả dữ liệu vào và phương thức tiếp cận.

Corona và cộng sự [33] đã đề xuất một giải pháp là DeltaPhish để phát hiện các trang web giả mạo. Họ đánh giá rằng, các trang web giả mạo được tin tặc tạo ra để đánh cắp thông tin cá nhân, lừa đảo hoawjcc kiếm tiền. Việc bấm vào các URL độc hại sẽ đưa người dùng tới các trang web giả mạo, nơi mà họ có thể đối mặt với các nguy hiểm đến từ tội phạm mạng. Giải pháp của họ là thông qua việc phân tích sự khác biệt giữa các trang đã truy cập và trang tham chiếu được xác định từ trước, để đưa ra kết luận là trang web này có giả mạo hay không. Họ dựa trên mã nguồn HTML và sự xuất hiện trực quan của mỗi trang web để trích xuất các thuộc tính. Các thuộc tính này được đưa qua bộ phân loại Fussion để cho ra nhãn Phishing hoặc Legitimate. Bộ dữ liệu đánh giá gồm 5.6GB chứa cả URL và HTML code của một số webpage giả mạo. Độ chính xác đạt được cho phát hiện Website giả mạo là 99%.

Marchal và cộng sự cũng phát triển giải pháp là PhishStorm nhằm phát hiện các website giả mạo [34]. Khác với DeltaPhish khi xem xét cả mã nguồn HTML, giải pháp PhishStorm đánh giá một một trang web có giả mạo hay không chủ yếu dựa trên phân tích URL của nó. Họ sử dụng một số mô hình học máy để phân loại URL là giả mạo hay lành tính. Độ chính xác đạt được là 94,91%.

Nhìn chung, bài toán phát hiện trang web giả mạo và phát hiện DGA Botnet là khác nhau. NCS tóm tắt các điểm khác nhau chính ở Bảng 1.3 như sau:

Bảng 1.3. So sánh bài toán phát hiện Website giả mạo và bài toán phát hiện DGA Botnet

	Đầu vào	**Mục tiêu**	**Nhãn phân lớp nhị phân**	**Nhãn phân lớp đa lớp**	**Thuật toán DGA**
Phát hiện Website giả mạo	URLs / HTML Code	Trang Web giả mạo	0: Website lành tính 1: Website giả mạo	NULL	Không

Phát hiện và phân loại DGA Botnet	Tên miền	Địa chỉ IP của C&C Server	0: Tên miền lành tính 1: Tên miền độc hại	n nhãn tương ứng với n họ DGA Botnet	Có

- Đầu tiên, dữ liệu đầu vào của bài toán DGA Botnet chỉ là tên miền, trong khi đó đối với bài toán phát hiện URL giả mạo thì đó là các URL và HTML Code. Cần lưu ý rằng, khái niệm URL khác với khái niệm tên miền.

- Thứ hai, trong bài toán phát hiện DGA Botnet, các tên miền được trỏ đến địa chỉ IP của C&C Server. Ngược lại, các URL độc hại đưa người dùng đến các webpage với nội dung giả mạo, độc hại.

- Thứ ba, vấn đề DGA Botnet gồm hai bài toán phát hiện và phân loại. Trong khi đó, các URL chỉ được đánh giá với hai nhãn là giả mạo hoặc lành tính mà không có nhãn cho bài toán phân loại.

- Cuối cùng, các thuật toán sinh tên miền xuất hiện trong bài toán DGA Botnet, mà không xuất hiện trong bài toán Website giả mạo.

1.3.5. Bộ dữ liệu đánh giá cho bài toán DGA Botnet

Một số bộ dữ liệu cho đánh giá bài toán DGA Botnet đã được công bố. NCS lựa chọn 04 bộ dữ liệu được xem là phù hợp nhất cho các đánh giá thuật toán được trình bày ở các chương sau, bao gồm: Andrey Abakumov's DGA Repository [35], OSINT DGA feed [36], UMUDGA Dataset [13] và 360NetLab Dataset [37]:

- Andrey Abakymov’s DGA Repository (AADR): Đây là kho lưu trữ được tạo vào năm 2016 bởi Andrey Abakumov, bao gồm mã nguồn các thuật toán tạo tên miền tự động và các tên miền DGA độc hại. Top 1.000.000 tên miền phổ biến của Alexa cũng được sử dụng và gán nhãn lành tính. Bộ dữ liệu này đã được nhiều nghiên cứu trước đó sử dụng để đánh giá giải pháp của họ. Dữ liệu được công bố công khai trên GitHub.

- OSINT DGA feed (OSINT): Là một bộ dữ liệu đáng tin cậy, được tạo bởi Bambenek, với một khối lượng lớn các tên miền độc hại được thu thập và tổng hợp. Dữ liệu này được chia sẻ miễn phí cho cộng đồng khoa học và không được cho phép sử dụng vào các mục đích thương mại.

- UMUDGA Dataset (UMUDGA): Bộ dữ liệu được tổng hợp, xây dựng bởi Zago và cộng sự thuộc trường đại học University of Murcia. Đây là bộ dữ liệu có thể xem là đầy đủ nhất hiện nay, khi tổng hợp được 50 họ DGA Botnet. Mỗi họ có từ 10.000 đến 500.000 mẫu tên miền. Các dữ liệu được bố trí một cách khoa học, dưới

các dạng ARFF, CSV và TXT, phù hợp với các công cụ và ngôn ngữ lập trình khác nhau. Toàn bộ dữ liệu có thể dễ dàng tiếp cận trên Mendeley Data, đảm bảo tính tin cậy và công bố rộng rãi.

- 360NetLab Dataset (360NL): Là bộ dữ liệu gồm các tên miền độc hại thu thập từ thế giới thực, được xây dựng bởi nhóm nghiên cứu 360NetLab, Qihoo 360 Technology Co.,Ltd. Máy tìm kiếm sẽ liên tục tìm và phát hiện các mẫu DGA Botnet mới nhất để cập nhật vào cơ sở dữ liệu. Tuy nhiên, một vấn đề của bộ dữ liệu này là liên tục thay đổi theo thời gian thực.

Chi tiết về số lượng tên miền và tính chất của các bộ dữ liệu trên được cho bởi Bảng 1.4:

Bảng 1.4. Mô tả về 04 bộ dữ liệu DGA Botnet được sử dụng trong các đánh giá

	Phát hiện DGA Botnet	**Phân loại DGA Botnet**	**Số mẫu lành tính**	**Số mẫu DGA Botnet**	**Số họ DGA Botnet**
AADR	✓	✓	1.000.000	801.667	08
OSINT	✓	×	1.000.000	495.186	
UMUDGA	✓	✓	1.000.000	500.000	50
360NL	✓	×	1.000.000	1.513.524	

1.3.6. Thông số đánh giá thuật toán

1.3.6.1. Tham số đánh giá cho phân lớp nhị phân

Đối với bài toán phân lớp nhị phân, với nhãn 0 là tên miền lành tính, nhãn 1 là tên miền DGA Botnet, ta định nghĩa:

- TP: Số lượng mẫu tên miền độc hại được phân loại đúng là độc hại.

- TN: Số lượng mẫu tên miền là lành tính được phân loại đúng là lành tính.

- FP: Số lượng mẫu tên miền lành tính được phân loại sai thành độc hại.

- FN: Số lượng mẫu tên miền độc hại được phân loại sai thành lành tính.

NCS đánh giá hiệu quả của giải pháp phát hiện DGA Botnet thông qua các tham số gồm Accuracy, Precision, Recall và F_1-score, lần lượt được tính bằng các công thức (1.1), (1.2), (1.3) và (1.4) dưới đây:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (1.1)$$

Accuracy giúp đánh giá độ chính xác chung của mô hình, bao gồm việc phát hiện đúng tên miền là lành tính hoặc tên miền là độc hại.

$$Precision = \frac{TP}{TP + FP} \tag{1.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{1.3}$$

Precision và Recall đánh giá riêng về khả năng phát hiện đúng một tên miền lành tính hay một tên miền độc hại trên tổng số lượng mẫu của tên miền đó. Nhìn chung, việc phát hiện nhầm một tên miền lành tính thành độc hại sẽ ít gây nguy hiểm hơn là việc chấp nhận nhầm một tên miền độc hại là lành tính.

$$F_1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{1.4}$$

$F_1 - score$ là đại lượng dùng chung để đánh giá một cách tổng thể giữa Precison và Recall trong mô hình.

1.3.6.2. Tham số đánh giá cho phân lớp đa lớp

Đối với các bài toán phân lớp có nhiều lớp, NCS sử dụng phương pháp tính Weighted Averrage để đánh giá hiệu quả phân loại cho tất cả các lớp với các giá trị tương ứng là $Accuracy$, $Avg\ Precision$, $Avg\ Recall$ và $Avg\ F_1 - score$. Các thông số này được đánh giá dựa trên Precision, Recall và F_1-Score của từng lớp và số lượng mẫu phân loại tương ứng, lần lượt được tính bởi các công thức (1.5), (1.6), (1.7) và (1.8) như sau:

$$Accuracy = \frac{\sum_{i=1}^{n}(Support_i \times Accuracy_i)}{\sum_{i=1}^{n} Support_i} \tag{1.5}$$

$$Avg\ Precision = \frac{\sum_{i=1}^{n}(Support_i \times Precision_i)}{\sum_{i=1}^{n} Support_i} \tag{1.6}$$

$$Avg\ Recall = \frac{\sum_{i=1}^{n}(Support_i \times Recall_i)}{\sum_{i=1}^{n} Support_i} \tag{1.7}$$

$$Avg\ F_1 - score = \frac{\sum_{i=1}^{n}(Support_i \times F_1 - score_i)}{\sum_{i=1}^{n} Support_i} \tag{1.8}$$

Trong đó, $Support_i$ là số lượng được phân loại có nhãn i với $1 \leq i \leq n$.

Dựa trên yêu cầu của bài toán, NCS linh hoạt sử dụng các tham số phù hợp cho việc đánh giá kết quả phân loại. Đối với riêng từng nhãn, NCS sử dụng các giá trị Precision, Recall và F_1-score được tính bằng công thức (1.2), (1.3) và (1.4). Trong trường hợp đánh giá tổng thể cho nhiều nhãn, NCS sử dụng các giá trị Accuracy, A.Precision, A.Recall và A.F_1-score được tính bằng công thức (1.5), (1.6), (1.7) và (1.8) và linh hoạt trong ký hiệu với tiền tố "A." để thuận tiện cho việc trình bày sơ đồ, bảng biểu.

1.3.7. Ý nghĩa của bài toán DGA Botnet

Botnet ngày càng phát triển, các Bot được thiết kế không chỉ để lây nhiễm trên các thiết bị truyền thống như máy tính cá nhân mà còn có khả năng lây nhiễm trên các thiết bị IoT. Quy mô của mạng Botnet hiện đại có thể mở rộng hơn rất nhiều so với các mạng Botnet truyền thống, từ đó cho phép chúng có khả năng tấn công với quy mô, cường độ mạnh hơn, gây ra những thiệt hại lớn và lâu dài hơn cho nạn nhân.

Botnet có nhiều phương thức để lây nhiễm vào thiết bị, chúng cũng liên tục cải tiến mã nguồn để có thể lây nhiễm và ẩn mình hiệu quả hơn. Có một số phương án được đưa ra như: Phòng chống lây nhiễm, rà quét mã độc và ngăn chặn các cuộc tấn công khi nó đang diễn ra. Mỗi phương án đều có những ưu điểm và hạn chế riêng.

Vận dụng cơ chế hoạt động của thuật toán sinh tên miền có thể mang lại một giải pháp hiệu quả để phát hiện và ngăn chặn DGA Botnet. Các thuật toán này có đối tượng đầu vào là lưu lượng DNS, không đòi hỏi quá nhiều năng lực thu thập và xử lý của hệ thống. Việc phát hiện hoạt động của DGA Botnet diễn kể ra ngay cả khi bằng cách nào đó chúng đã lây nhiễm vào máy tính. Đồng thời, chúng cũng bị vô hiệu hóa một cách hiệu quả nếu hệ thống phát hiện và ngăn chặn kết nối của chúng ra bên ngoài. Một ưu điểm nữa là giải pháp cho bài toán DGA Botnet hoàn toàn có thể áp dụng cho việc phát hiện những phần mềm, mã độc có hành vi truy vấn DNS tương tự. Đặc biệt là các phần mềm, mã độc có hành vi nghe lén, tình báo vì chúng có xu hướng cần kết nối ra bên ngoài.

Các thuật toán, giải pháp đề xuất để giải quyết bài toán DGA Botnet có thể được áp dụng vào các giải pháp bảo mật như tường lửa, hệ thống phát hiện và ngăn chặn xâm nhập, mudule phát hiện mã độc trong thiết bị UTM hay các giải pháp đảm bảo an ninh thế hệ mới.

1.4. Một số nghiên cứu giải quyết bài toán DGA Botnet

Một số kỹ thuật đã được đề xuất để phát hiện Botnet nói chung và DGA Botnet nói riêng, nổi bật như: Các kỹ thuật phân tích DNS; thuật toán học máy và học sâu.

1.4.1. Hướng tiếp cận sử dụng các kỹ thuật phân tích DNS

Trong [6], Alieyan và cộng sự trình bày các kỹ thuật để khám phá mạng Botnet khác nhau thông qua phân tích lưu lượng DNS. Đây cũng là cuộc khảo sát để thảo luận các kỹ thuật phát hiện Botnet dựa trên DNS. Trong đó đưa ra và làm rõ các vấn đề, giải pháp hiện tại và hướng nghiên cứu trong tương lai.

Trong nghiên cứu [7], tác giả Kwon và cộng sự đã lập luận rằng, có thể trích xuất được nhiều thông tin quan trọng dựa trên lưu lượng DNS. Tuy nhiên, trong trường hợp lưu lượng các truy vấn DNS có mức độ lớn thì việc phân tích gặp nhiều khó khăn. Các nghiên cứu cũng bị hạn chế tiêu cực bởi số lượng lớn các truy vấn và tên miền. Nhóm nghiên cứu giới thiệu một hướng tiếp cận nhanh, được gọi là PsyBoG, dùng để phát hiện hành vi độc hại dựa trên phân tích một lượng lớn lưu lượng DNS. Giải pháp đề xuất sử dụng kỹ thuật xử lý tín hiệu, phân tích mật độ phổ năng lượng PSD để phát hiện các truy vấn DNS định kỳ của Botnet. Giải pháp PsyBoG có các ưu điểm bao gồm: (1) Phát hiện các mạng Botnet có khả năng ẩn mình tinh vi; (2) Cho phép xử lý các truy vấn DNS trên quy mô lớn và (3) Có khả năng phát hiện các nhóm máy chủ có hành vi độc hại.

Giải pháp đề xuất được đánh giá dựa trên 755 DNS Traces thu thập trong một mạng thực, đã bao gồm các lưu lượng độc hại. Kết hợp với 756 lưu lượng truy cập máy chủ DNS thực. Kết quả thử nghiệm cho thấy PsyBoG có khả năng phát hiện chính xác với tỉ lệ 95%, đồng thời chứng minh tính hiệu quả và khả năng ứng dụng trong thực tế khi có thể hoạt động với lưu lượng DNS lớn. Trong các lưu lượng DNS trên, nhóm nghiên cứu cũng phát hiện được 23 họ Botnet chưa biết, 26 họ Botnet đã biết với tỉ lệ dương tính giả chỉ 0,1%.

Wang và cộng sự phát hiện Botnet dựa trên phân tích tên miền [8]. Họ đề xuất một sơ đồ phát hiện dựa trên DGA, gọi là DBod. Giải pháp này dựa trên phân tích hành vi truy vấn DNS. Thử nghiệm thực tế trên các máy chủ và nhận thấy rằng hầu hết các truy vấn độc hại đều bị chặn bắt. Tính khả thi của giải pháp được thể hiện thông qua việc đánh giá trên bộ dữ liệu mạng thu được trong môi trường giáo dục trong thời gian 26 tháng, với quy mô khoảng 10.000 người dùng. Kết quả cho thấy giải pháp đề xuất có khả năng phát hiện Botnet với độ chính xác trên 99%. Tuy nhiên, hạn chế của nghiên cứu đó là số mẫu trong bộ dữ liệu đánh giá còn nhỏ, chỉ gồm 04 họ DGA Botnet được xem xét tới gồm Kraken, Conficker, Cycbot và Murofet, cũng như chưa có sự nghiên cứu rộng rãi trên các bộ dữ liệu khác.

Chowdhury và cộng sự nhận xét rằng, Botnet ngày càng trở nên tinh vi và có khả năng tấn công gây thiệt hại to lớn hơn [38]. Hầu hết các phương pháp phát hiện

Botnet dựa trên luật hay lưu lượng mạng tỏ ra không thật sự hoàn hảo. Do đó, việc xây dựng một phương pháp có khả năng phát hiện nhanh và mạnh mẽ có ý nghĩa quan trọng. Trong nghiên cứu của mình, nhóm tác giả đề xuất một phương pháp phát hiện Botnet mới dựa trên đặc điểm cấu trúc liên kết của các nút trong đồ thị, bao gồm: Bậc, bán bậc ra, bán bậc vào, trọng số của bán bậc ra và bán bậc vào, hệ số phân cụm... Phương pháp phân cụm dựa trên bản đồ tự tổ chức được áp dụng để thiết lập các cụm nút trong mạng. Phương pháp đề xuất có khả năng cô lập Bot trong các cụm có kích thước nhỏ, trong khi cũng chứa phần lớn các nút bình thường trong cùng một cụm lớn. Khi đó, các Bot có thể được phát hiện bằng cách tìm kiếm một số lượng hạn chế các node. Bên cạnh đó, một quy trình lọc cũng được đề xuất để nâng cao tính chính xác của thuật toán. Nghiên cứu được đánh giá trên bộ dữ liệu CTU-13 [39]. Kết quả cho thấy phương pháp đề xuất có thể phát hiện các Bot một cách hiệu quả, dù cho các hành vi của chúng là khác nhau.

Trong nghiên cứu [9], Bisio và cộng sự đã trình bày báo cáo về thuật toán phát hiện DGA Botnet dựa trên một Single Network Monitoring. Các giai đoạn của giải pháp đề xuất bao gồm: (1) Phát hiện một Bot đang tìm kiếm máy chủ C&C và các tên miền được sinh ra tự động có liên quan; (2) Phân tích các yêu cầu DNS đã được giải quyết trong cùng một khoảng thời gian. Sau đó, các đặc điểm về ngôn ngữ và ngữ nghĩa được phân lớp chúng vào một họ DGA cụ thể. Cuối cùng (3), các cụm được phân tích để giảm thiểu dương tính giả. Giải pháp đề xuất được đánh giá trên hai môi trường, bao gồm một môi trường mạng đặc biệt được xây dựng, với các họ DGA Botnet đã được gán nhãn. Môi trường thứ hai là mạng nội bộ LAN của một công ty. Trong thí nghiệm đầu tiên, tất cả các họ DGA Botnet đều được phát hiện. Với thí nghiệm thứ hai, thuật toán đã phát hiện ra một máy chủ bị nhiễm mã độc trong một quá trình đánh giá kéo dài 15 ngày. Bộ dữ liệu thử nghiệm bao gồm 40 họ DGA Botnet khác nhau, với khả năng phát hiện hầu hết các họ DGA Botnet với độ chính xác cao từ 92,67% trở lên, duy nhất một trường hợp đạt 88,85%.

Wang và cộng sự giới thiệu một cách tiếp cận bao gồm: (1) Phát hiện sự hiện diện của Botnet và (2) Xác định các nút bị lây nhiễm [40]. Trong giai đoạn đầu tiên, các bất thường sẽ được phát hiện bằng cách tính toán sự chênh lệch giữa trạng thái hiện tại và các trạng thái trước đó, gọi là phân phối theo kinh nghiệm. Ở giai đoạn thứ hai, các Bot được phát hiện dựa trên ý tưởng xây dựng cộng đồng mạng xã hội trong một biểu đồ, ghi lại mối tương quan và tương tác giữa các nút theo thời gian. Việc phát hiện cộng đồng được thực hiện bằng cách tối đa hóa thước đo module trong biểu đồ này. Nhóm nghiên cứu đánh giá phương pháp đề xuất bằng lưu lượng truy

cập Botnet trong thế giới thực và so sánh với các nghiên cứu khác. Tuy nhiên, các kết quả đánh giá chưa được thảo luận một cách kỹ lưỡng.

Trung và cộng sự [10] trình bày các nghiên cứu mở rộng về Botnet trên các thiết bị IoT. Chúng có các khả năng tương tự với Botnet truyền thống, tạo nên những cuộc tấn công từ chối dịch vụ quy mô lớn. Mối đe dọa này càng tăng khi mà các cơ chế bảo mật cho thiết bị IoT chưa thực sự đầy đủ. Các thiết bị IoT có những đặc điểm hạn chế riêng về hệ điều hành, khả năng xử lý cũng như nguồn năng lượng. Hầu hết các nghiên cứu trước đó chưa giải quyết vấn đề đa kiến trúc và cần tiết kiệm tài nguyên xử lý trên các thiết bị IoT, đòi hỏi một giải pháp cần ít tài nguyên xử lý hơn.

Trong nghiên cứu của mình, nhóm tác giả đề xuất một phương pháp phát hiện IoT Botnet, dựa trên việc trích xuất các thuộc tính từ đồ thị PSI. Giải pháp này có thể khắc phục được vấn đề đa kiến trúc trong thiết bị IoT, đồng thời giảm thiểu sự phức tạp tính toán. Kết quả thử nghiệm cho thấy, giải pháp đề xuất đạt độ chính xác 98,7%. Bộ dữ liệu thử nghiệm gồm 11.200 tệp ELF chứa 7.199 mẫu IoT Botnet và 4.001 mẫu lành tính. Dữ liệu được thu thập từ IoTPOT Team [41] và VirusShare [42]. So sánh với một số nghiên cứu khác cho thấy mô hình đề xuất cho kết quả tốt hơn. Mã nguồn của giải pháp cũng được công bố trên GitHub.

1.4.2. Hướng tiếp cận dựa trên học máy

Trong một nghiên cứu tiếp cận theo hướng sử dụng học máy, Hiếu và cộng sự đã đánh giá độ hiệu quả các các thuật toán học máy có giám sát trong việc phát hiện DGA Botnet [11]. Với dữ liệu đầu vào là các tiên miền, nhóm nghiên cứu xây dựng các mô hình bao gồm: Hidden Markov, C4.5 Decision Tree, Extreme Learing Machine. Đồng thời cũng thí nghiệm trên các mô hình SVM, Recurrent SVM, CNN kết hợp LSTM và Bidirectional LSTM. Các đánh giá được thực hiện trên bộ dữ liệu gồm 1.000.000 tên miền lành tính của Alexa được gán nhãn non-DGA [43], và 37 họ DGA Botnet được gán nhãn DGA. Các họ DGA Botnet này được tổng hợp từ OSINT DGA feed from Bambenek Consulting, với 81.490 tên miền. Kết quả thử nghiệm cho thấy, trong bài toán phát hiện DGA Botnet, các mô hình dựa trên SVM và LSTM cho kết quả với độ chính xác từ 99,55% trở lên, cao hơn đáng kể so với các mô hình học máy truyền thống. Đối với bài toán phân loại DGA Botnet, các kết quả thu được chưa thực sự khả quan, đặc biệt là có 08 họ DGA Botnet không phát hiện được bởi các mô hình học máy có giám sát.

Trong một nghiên cứu khác, Khan và cộng sự xem xét tới việc phát hiện các mạng Botnet ngang hàng (Peer-to-Peer) [12]. Họ lập luận rằng việc phát hiện các mô hình Botnet dạng này đặt ra các thách thức nhiều hơn so với Botnet sử dụng giao thức

IRC hay HTTP. Để giải quyết vấn đề trên, nhóm nghiên cứu đề xuất một phương pháp phân loại lưu lượng đa lớp bằng cách áp dụng các mô hình học máy. Đầu tiên, các thuộc tính phù hợp sẽ được lựa chọn, đồng nghĩa với việc loại bỏ những thuộc tính dư thừa dựa trên thuật toán cây quyết định. Các gói tin không phải là P2P cũng được loại bỏ để giảm thiểu lưu lượng qua các cổng mạng. Tiếp theo, lớp thứ hai thực hiện phân loại lưu lượng thành hai nhóm là P2P và không phải P2P. Ở lớp cuối cùng, nhóm nghiên cứu đề xuất sử dụng bộ phân loại cây quyết định để phát hiện mạng P2P Botnet. Độ chính xác trung bình đạt được là 98,7%. Các thí nghiệm được thực hiện trên bộ dữ liệu CTU-13 và ISOT Dataset.

Zago và cộng sự [13] trình bày một nghiên cứu về bộ dữ liệu mới cho phát hiện DGA Botnet. Nhóm nghiên cứu tổng hợp và xây dựng thành bộ dữ liệu với tên gọi là UMUDGA Dataset. Bộ dữ liệu này bao gồm 1.000.000 tên miền lành tính kết hợp với 50 họ DGA Botnet khác nhau, mỗi họ DGA Botnet chứa mã nguồn tự động sinh tên miền, các tên miền mẫu với số lượng từ 10.000 đến 500.000 đối với mỗi họ. Các đánh giá ban đầu của nhóm tác giả cho thấy, thuật toán SVM cho F_1-score cao nhất trong các mô hình học máy, đạt 98,9% cho bài toán phát hiện DGA Botnet [44]. Đối với bài toán phân loại DGA Botnet, các thuật toán này tỏ ra chưa hiệu quả khi F_1-Score đều ở dưới mức 80%.

Trong một công bố gần đây, Xuân và cộng sự [14] đã đề xuất những cải tiến cho mô hình học máy. Họ đề xuất sử dụng các thuộc tính mới và áp dụng trên thuật toán rừng ngẫu nhiên. Kết quả cho thấy thuật toán mới giúp giảm tỉ lệ cảnh báo nhầm cũng như tăng tỉ lệ phát hiện tên miền của DGA Botnet. Cụ thể, mô hình có thể lệ cảnh báo sai dưới 3,02% và điểm F_1-score đạt 97,03% trong các đánh giá.

Suryotrisongko và cộng sự [45] giới thiệu một hướng tiếp cận mới sử dụng các mô hình học máy lượng tử lai để phát hiện DGA Botnet. Họ phân tích hiệu suất của mô hình lai so với mô hình truyền thống để nghiên cứu hiệu quả của mạch lượng tử như một lớp trong mô hình học sâu. Nhóm nghiên cứu sử dụng bốn đặc trưng được trích xuất từ DGA Botnet bao gồm MinREBotnets, CharLength, TreeNewFeature và nGramReputation_Alexa. Mạch lượng tử của mô hình hybrid là sự kết hợp của các kỹ thuật nhúng và mạch lớp. Kỹ thuật gây nhiễu cũng được áp dụng để đánh giá khả năng thích ứng của mô hình đối với nhiễu. Kết quả thực nghiệm cho thấy, trong một số trường hợp, mô hình lai đạt hiệu suất cao (độ chính xác tối đa lên đến 94,7% với n=100 và đạt 93,9% với n=1.000). Cách tiếp cận trên mang lại độ chính xác cao, vượt trội hơn so với mô hình học sâu trong các thử nghiệm với n=100. Tuy nhiên, hiệu suất tổng thể vẫn kém hơn trong các trường hợp còn lại. Mô hình này gợi ý một hướng nghiên cứu tiềm năng trong tương lai để phát hiện DGA Botnet.

Zhao và cộng sự [46] đề xuất phương pháp DOmain Linguistic PHonIcs detectioN - DOLPHIN để phát hiện DGA Botnet. Dựa trên các mẫu DOLPHIN, một phương pháp so khớp mới được sử dụng để tái tạo tên miền với các thành phần của nguyên âm và phụ âm có độ dài biến đổi. Từ những tên miền đó, DOLPHIN trích xuất các đặc trưng dựa trên ngữ âm. Kết quả thực nghiệm cho thấy, so với FANCI và RF, DOLPHIN có thể đạt được độ chính xác phát hiện cao hơn 0,0265 với tỷ lệ dương tính sai và tỷ lệ phủ sóng sai thấp hơn mà không đòi hỏi quá nhiều chi phí tính toán. DOLPHIN cũng có khả năng tổng quát hóa cho các nguồn dữ liệu khác trong thế giới thực. Do đó, giải pháp này có thể phù hợp với hầu hết các đặc trưng ngôn ngữ và mang lại cải thiện về hiệu suất so với các phương pháp dựa trên đặc trưng ngôn ngữ hiện có.

Các lưu lượng mạng của Botnet luôn cố gắng ẩn mình trước các lưu lượng thông thường của người dùng, đặc biệt là các mạng Botnet được triển khai theo mô hình P2P. Trong một nghiên cứu của mình [47], Alauthman và cộng sự đề xuất một cơ chế giúp giảm thiểu các lưu lượng phức tạp, tích hợp với một kỹ thuật học tăng cường. Lưu lượng mạng được rút gọn giúp giảm khối lượng tính toán của thuật toán đề xuất. Kết quả thử nghiệm cho thấy phương pháp mới đạt tỉ lệ phát hiện đúng là 98,3%, tỉ lệ dương tính giả thấp với 0,012%. Nghiên cứu được đánh giá trên sự tổng hợp của ba bộ dữ liệu gồm ISOT Dataset [48], P2P Botnet [49] và Information Security Centre of Excellence Dataset [50].

1.4.3. Hướng tiếp cận dựa trên học sâu

Trong hướng tiếp cận dựa trên học sâu, Đức và cộng sự [15] đã sử dụng mạng bộ nhớ ngắn hạn dài Long Short-Term Memory Network - LSTM để giải quyết cả hai bài toán DGA Botnet. Nhóm nghiên cứu đề xuất một thuật toán mới với tên gọi là LSTM.MI, kết hợp cả hai mô hình phân loại, kế thừa ưu điểm của LSTM truyền thống và các cải tiến để tăng cường độ chính xác với nhiễu. Về dữ liệu thử nghiệm, các tên miền được nhóm tác giả thu thập từ thực tế, với 100.000 tên miền lành tính phổ biến nhất từ Alexa và 37 họ DGA Botnet được tổng hợp. Kết quả thử nghiệm cho thấy thuật toán đề xuất giúp nâng cao ít nhất 7% độ chính xác so với mô hình LSTM truyền thống. Nó cũng đạt độ chính xác cao trong bài toán phân lớp nhị phân với F_1-score đạt 98,49%, đồng thời có khả năng nhận ra 05 họ DGA Botnet bổ sung. Hạn chế của nghiên cứu là bộ dữ liệu chưa thực sự đầy đủ và một số họ DGA Botnet gần như không thể phát hiện được.

Curtin và cộng sự đã sử dụng mạng RNN để phát hiện và phân loại DGA Botnet [16]. Họ nhận ra rằng các tên miền được xây dựng dựa trên một không gian từ vừng,

có các nét đặc trưng khá giống với các tên miền lành tính, từ đó giúp tăng khả năng ẩn mình của các tên miền được sinh ra. Nhóm nghiên cứu đã đề xuất một khái niệm mới, gọi là thang điểm Smashword. Đây là thang đo lường mức độ giống nhau giữa tên miền của Botnet và các tên miền lành tính. Nhóm nghiên cứu áp dụng mô hình mạng Recurrent Neural Network và Side Information để áp dụng tiêu chuẩn đo lường trên. Bộ dữ liệu thử nghiệm với 1.000.000 tên miền Alexa, kết hợp với 41 họ DGA Botnet được nhóm nghiên cứu tổng hợp, với tổng cộng 2.300.000 tên miền cho cả hai nhãn. Thực nghiệm cho thấy mô hình mới có tiềm năng ứng dụng cao cải tiến được độ chính xác hơn so với các mô hình trước đó. Một số họ DGA Botnet phức tạp như matsnu, suppobiox hay rovnix cũng được phát hiện bởi giải pháp trên.

Qiao và cộng sự [17] đề xuất một kiến trúc học sâu phát triển dựa trên mạng LSTM và Attention. Kiến trúc mới có lõi gồm một lớp LSTM với một lớp Attention được bố trí tuần tự, kích thước đầu vào là 54 × 128. Nhóm tác giả đánh giá trên một bộ dữ liệu gồm 16 nhãn bao gồm cả nhãn lành tính. Kết quả thực nghiệm cho thấy mô hình đề xuất có F_1-score đạt 94,58% cho bài toán phân loại.

Namgung và cộng sự [19] đề xuất một cải tiến được phát triển từ mạng LSTM và mạng CNN. Họ đề xuất các mô hình kết hợp, trong đó nổi bật là hai mô hình BiLSMT_Attention và mô hình Ensemble dựa trên BiLSTM và CNN. Hai mô hình được đánh giá trên một bộ dữ liệu gồm 21 nhãn và có F_1-score lần lượt là 96,18% và 96,66% cho bài toán phân loại DGA Botnet.

Vinayakumar và nhóm cộng sự nghiên cứu bài toán phát hiện tên miền độc hại được sinh bởi Botnet hay thư điện tử, URL độc hại [20]. Một số kỹ thuật biểu diễn đặc trưng dựa trên n-gram đã được sử dụng để mô hình hóa bài toán. Bộ dữ liệu để đánh giá bao gồm các tên miền lành tính và độc hại được thu thập từ OpenDNS, Alexa và OSINT Feeds. Kết quả so sánh thấy mô hình CNN-LSTM là hiệu quả nhất với F_1-score đạt 96,3% cho bài toán phát hiện.

Liu và cộng sự [51] lập luận rằng biểu diễn đặc trưng bằng các giá trị vô hướng có thể dẫn đến mất mát thông tin. Nhóm tác giả đề xuất một Capsule Network tuần tự (mạng học sâu được cải tiến từ CNN) dựa trên thuật toán k-means, gọi là LSTM-CapsNet. Mô hình sử dụng một đơn vị LSTM hai chiều để trích xuất các thuộc tính cơ bản và sử dụng thuật toán k-mean để phân cụm thành hai nhãn độc hại và lành tính. Đánh giá được thực hiện trên hai bộ dữ liệu, bao gồm bộ dữ liệu tên miền DGA từ mạng thực tế và tên miền DGA thu được thông qua thuật toán tạo tên miền tự động. Kết quả thực nghiệm cho thấy mô hình đề xuất đạt độ chính xác lần lượt là 99,17% và 97,75% trên hai bộ dữ liệu. Mô hình này không chỉ cải thiện khả năng nhận dạng

tên miền DGA và nhận dạng họ tên miền DGA trên lý thuyết mà còn thể hiện hiệu quả trong bộ dữ liệu thực tế.

1.5. Kết luận Chương 1

Trong Chương 1, NCS trình bày tổng quan chung về Botnet nói chung, bài toán DGA Botnet nói riêng và khảo sát các nghiên cứu liên quan. Trong đó, xác phạm vi nghiên cứu của luận án là bài toán phân loại DGA Botnet. Chương 1 cũng trình bày sự khác nhau giữa bài toán phát hiện DGA Botnet với bài toán phát hiện URL độc hại và ý nghĩa của bài toán này.

Trong các chương tiếp theo, NCS trình bày các kết quả nghiên cứu, đánh giá và đề xuất giải pháp dựa trên học sâu để giải quyết bài toán phát hiện và phân loại DGA Botnet.

Một phần kết quả trình bày tại Chương 1 được công bố tại [CT2] [CT6] trong Danh mục các công trình công bố liên quan đến luận án.

Chương 2. PHÁT HIỆN DGA BOTNET SỬ DỤNG NCM VÀ HỌC MÁY

Trong chương 2, NCS tiếp cận theo hai hướng là thuật toán NCM và học máy để phát hiện DGA Botnet. NCS cũng đề xuất thử nghiệm hai mô hình học máy kết hợp là VEA và HEA để cải thiện độ chính xác so với các mô hình đơn. Mục tiêu của chương này là xem xét hiệu quả của hai hướng tiếp trên và làm cơ sở để tiếp cận học sâu tại Chương 3 của luận án.

2.1. Phát hiện DGA Botnet sử dụng NCM

Việc phân cụm dữ liệu có nhiều ứng dụng quan trọng trong khai phá dữ liệu, nhận dạng mẫu, truy hồi thông tin và học máy. Trong thực tế, để các dữ liệu thường phức tạp, thiếu hụt hoặc có tính chất mơ hồ, không chắc chắn. Trong phần này, NCS tiếp cận theo hướng áp dụng thuật toán NCM cho bài toán phát hiện DGA Botnet.

2.1.1. Thuật toán NCM

Tập mờ trung lập – Neutrosophic Set được Smarandache đề xuất [52], là một cải tiến của tập mờ truyền thống. Trên không gian X, một tập mờ trung lập A được định nghĩa như sau:

$$A = \{x, \big(T_A(x), I_A(x), F_A(x)\big) : x \in X\}$$

Trong đó, hàm $T_A(x), I_A(x), F_A(x)$ lần lượt thể hiện độ thuộc của phần tử x gồm thuộc về, trung lập và không thuộc về một tập xác định nào đó. Các giá trị $T_A(x), I_A(x), F_A(x) \in [0, 1]$ và thỏa mãn điều kiện:

$$0 \leq T_A(x) + I_A(x) + F_A(x) \leq 3$$

Neutrosophic C-Means - NCM là thuật toán phân cụm mờ trên tập mờ trung lập, được đề xuất bởi Gou và cộng sự [53], được phát biểu như sau: Cho X là một tập khác rỗng, với một phần tử của X ký hiệu là $x \in$ X, tập mờ trung lập A xác định trên không gian X được đặc trưng bởi ba hàm số:

- Hàm T_A(x) đo độ thuộc chỉ ra rằng sự kiện x sẽ xảy ra;

- Hàm I_A(x) đo độ trung lập, hay là x thuộc vào vùng ranh giới;

- Hàm F_A(x) đo độ không thuộc, tin rằng sự kiện x sẽ không xảy ra.

Áp dụng các kết quả của NCM trong bài toán DGA Botnet như sau: Ta xác định mỗi điểm dữ liệu là một tên miền đầu vào, được biểu diễn bằng một vector. Thuật toán NCM sẽ chia các điểm dữ liệu này thành 03 cụm tương ứng. Bằng phương pháp

xem xét nhãn đại diện cho các phần tử trong cụm, ta xác định được nhãn chung của toàn cụm. Các nhãn chung bao gồm: Nhãn DGA Botnet, nhãn lành tính và nhãn nhiễu.

Thuật toán NCM được trình bày như sau:

Thuật toán: $NCM(X, \varepsilon)$	
Dữ liệu vào	X, ε
Dữ liệu ra	k
Khởi tạo $T^{(0)}, I^{(0)}, F^{(0)}$	
Khởi tạo $C, m, \varepsilon, \delta,\ \omega_1,\ \omega_2,\ \omega_3$	
Lặp:	
	Tính $c_j^{(k)}$
	Tính $\bar{c}_{i_{max}}$
	Cập nhật $T^{(\mathrm{k+1})}$
	Cập nhật $I^{(\mathrm{k+1})}$
	Cập nhật $F^{(\mathrm{k+1})}$
Điều kiện lặp	$\left\|T_{ij}^{(k+1)} - T_{ij}^{(k)}\right\| > \varepsilon$
Gán mỗi dữ liệu vào lớp với giá trị $TM = [T, I, F]$ lớn nhất.	
$x_i \epsilon\ k^{th}$ nếu $k = argmax(TM_{ij})\ với\ j = 1, 2, \ldots, C+2$	

Biểu diễn k^{th} là vòng lặp thứ k.

Trong đó, C là số cụm, m và δ là hệ số mờ được lựa chọn cố định, w_1, w_2, w_3 là các trọng số tương ứng thỏa mãn điều kiện $0 < w_1, w_2, w_3 < 1$ và $(w_1 + w_2 + w_3 = 1)$. Giá trị ε là ngưỡng để đánh giá sự hội tụ của thuật toán. Bộ tham số được lựa chọn cho thuật toán NCM như sau: $C = 2; w_1 = 0{,}8;\ w_2 = 0{,}1; w_3 = 0{,}1; m = 2, \delta = 1{,}4; \varepsilon = 10^{-5}$.

Với N là số phần tử $(1 \le i \le N,\ 1 \le j \le C)$. Các tâm cụm được ký hiệu là c_j và được tính bởi công thức (2.1):

$$c_j = \frac{\sum_{i=1}^{N}(\omega_1 T_{ij})^m x_i}{\sum_{i=1}^{N}(\omega_1 T_{ij})^m} \tag{2.1}$$

Với mỗi điểm i, giá trị của $\bar{c}_{i_{max}}$ được xác định bằng giá trị của trung tâm cụm với giá trị lớn nhất và lớn thứ nhì của T_{ij} theo các công thức (2.2):

$$\bar{c}_{i_{max}} = \frac{c_{p_i} + c_{q_i}}{2} \quad (2.2)$$

Với:

$$p_i = argmax(T_{ij}) \; với \; j = 1, 2, \dots C \quad (2.3)$$

$$q_i = argmax(T_{ij}) \; với \; j \neq p_i \cap j = 1, 2, \dots C \quad (2.4)$$

Với T_{ij} là độ thuộc của phần tử i đối với cụm j; I_i và F_i thể hiện độ trung lập và độ nhiễu của phần tử i, thỏa mãn điều kiện sau:

$$\begin{cases} 0 < T_{ij}, I_i, F_i < 1 \\ \sum_{j=1}^{C} T_{ij} + F_i + I_i = 1 \end{cases}$$

Các giá trị T_{ij}, I_i, F_i lần lượt được tính và cập nhật bởi các công thức (2.5), (2.6) và (2.7):

$$T_{ij} = \frac{K}{\omega_1}(x_i - c_j)^{-\frac{2}{m-1}} \quad (2.5)$$

$$I_i = \frac{K}{\omega_2}(x_i - \bar{c}_{imax})^{-\frac{2}{m-1}} \quad (2.6)$$

$$F_i = \frac{K}{\omega_3}\delta^{-\frac{2}{m-1}} \quad (2.7)$$

Thuật toán dừng khi ta có $\left|T_{ij}^{(k+1)} - T_{ij}^{(k)}\right| < \varepsilon$ là ngưỡng hội tụ.

2.1.2. Áp dụng NCM để phát hiện DGA Botnet

NCS áp dụng thuật toán NCM qua hai bước như sau:

(1) Lựa chọn đặc trưng: NCS đề xuất và lựa chọn một số đặc trưng cơ bản của tên miền. Các đặc trưng được vector hóa phù hợp để làm đầu vào cho thuật toán NCM.

(2) Phân cụm và gán nhãn: Sử dụng thuật toán NCM để chia các điểm dữ liệu thành ba cụm, sau đó tiến hành gán nhãn đại diện cho các cụm tương ứng là DGA Botnet, Lành tính và Nhiễu.

2.1.2.1. Lựa chọn đặc trưng

Bảng 2.1 liệt kê 16 đặc trưng đề xuất được trích rút từ tên miền.

Bảng 2.1. Các đặc trưng được đề xuất cho thuật toán NCM

STT	Đặc trưng	Ý nghĩa
1.	DNL	Độ dài của tên miền
2.	NoS	Số lượng tên miền con
3.	SLM	Độ dài của tên miền phụ
4.	ND	Số lượng các ký tự xuất hiện ít nhất 1 lần trong tên miền
5.	HwP	Tên miền có chứa "www" hay không? 1: Có 0: Không
6.	HVTLD	Chứa tên miền root hợp lệ hay không? *(Căn cứ www.iana.org)* 1: Có 0: Không
7.	CTS	Có tên miền phụ nằm trong Root-zone database hay không? 1: Có 0: Không
8.	CIPA	Tên miền chứa địa chỉ IP hay không? 1: Có 0: Không
9.	CD	Có chứa chữ số hay không? 1: Có 0: Không
10.	UR	Tỉ lệ ký tự "_" trên tổng độ dài tên miền
11.	VR	Tỉ lệ ký tự nguyên âm trên tổng độ dài tên miền
12.	NR	Tỉ lệ ký tự chữ số trên tổng độ dài tên miền
13.	RRC	Giá trị lặp lại của một ký tự trong tên miền con
14.	RCC	Tỉ lệ phụ âm liên tiếp trong tên miền
15.	RCN	Tỉ lệ chữ số liên tiếp trong tên miền
16.	Entropy	Giá trị entropy của tên miền con

Với c là một ký tự trong dom, D là tập các ký tự có trong dom.

$V = \{'a', 'e', 'i', 'o', 'u'\}$.

$N = \{'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'\}$.

Ta tính một số giá trị trên như sau:

- UR của một tên miền được xác định bởi công thức (2.8):

$$VR_{dom} = \frac{count('\_', dom)}{length(dom)} \quad (2.8)$$

- VR của một tên miền được xác định bởi công thức (2.9):

$$VR_{dom} = \frac{count(V, dom)}{length(dom)} \quad (2.9)$$

- NR của một tên miền được xác định bởi công thức sau (2.10):

$$NR_{dom} = \frac{count(N, dom)}{length(dom)} \quad (2.10)$$

- RRC của một tên miền ký hiệu là RRC_{dom} được xác định bởi công thức (2.11):

$$RRC_{dom} = \frac{\sum_{c \in D} count(repeated_{dom})}{length(dom)} \quad (2.11)$$

Với $repeated_{dom}$ là số lần ký tự được lặp lại trong dom.

- RCC của một tên miền ký hiệu là RCC_{dom} được xác định bởi công thức (2.12):

$$RCC_{dom} = \frac{\sum_{c \in C} count(CC_{dom})}{length(dom)} \quad (2.12)$$

Với CC_{dom} là số lần phụ âm được được lặp lại liên tiếp trong dom.

- RCN của một tên miền ký hiệu là RCN_{dom} được xác định bởi công thức (2.13):

$$RCN_{dom} = \frac{\sum_{c \in N} count(CN_{dom})}{length(dom)} \quad (2.13)$$

Với CN_{dom} là số lần chữ số được được lặp lại liên tiếp trong dom.

- Entropy của một tên miền ký hiệu là $Entropy_{dom}$, để đo mức độ hỗn loạn của các ký tự trong tên miền, được xác định bởi công thức (2.14):

$$Entropy_{dom} = -\sum_{c \in D} \frac{count(c, dom)}{length(dom)} \times \log\left(\frac{count(c, dom)}{length(dom)}\right) \quad (2.14)$$

Với c là một ký tự trong dom, D là tập các ký tự có trong dom

Các thuộc tính trích xuất từ tên miền được vector hóa để đưa vào thuật toán NCM. Ở đây, NCS áp dụng thêm một giai đoạn là lựa chọn thuộc tính dựa vào đặc trưng Pearson. Mục tiêu của việc này nhằm lựa chọn các thuộc tính có ảnh hưởng

nhất làm đầu vào thuật toán, giúp giảm số chiều của vector, rút ngắn thời gian tính toán và cũng hạn chế các thuộc tính không có nhiều giá trị trong quá trình thực thi thuật toán.

Trong ma trận tương quan Pearson, giá trị tương quan của thuộc tính x và y là r_{xy} được tính bởi công thức (2.15):

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \quad (2.15)$$

Trong đó:

- n là số lượng giá trị mẫu;

- x_i, y_i: Là giá trị mẫu thứ i của x và y;

- $\bar{x}$: Giá trị trung bình các mẫu của x được xác định bởi công thức (2.16):

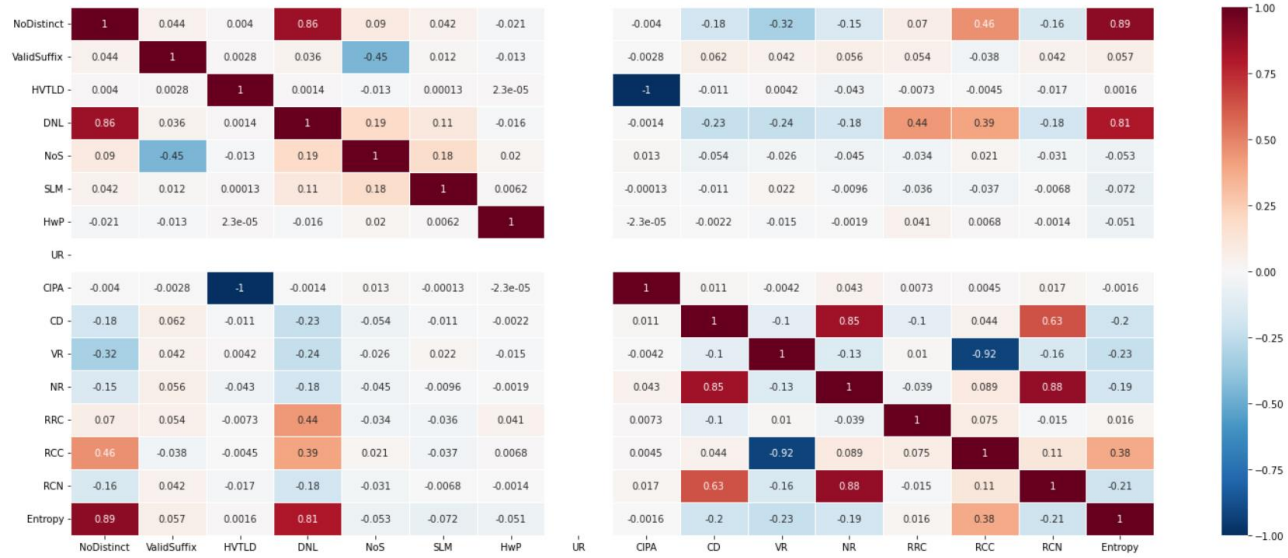
$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (2.16)$$

- $\bar{y}$: Giá trị trung bình các mẫu của y được xác định bởi công thức (2.17):

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \quad (2.17)$$

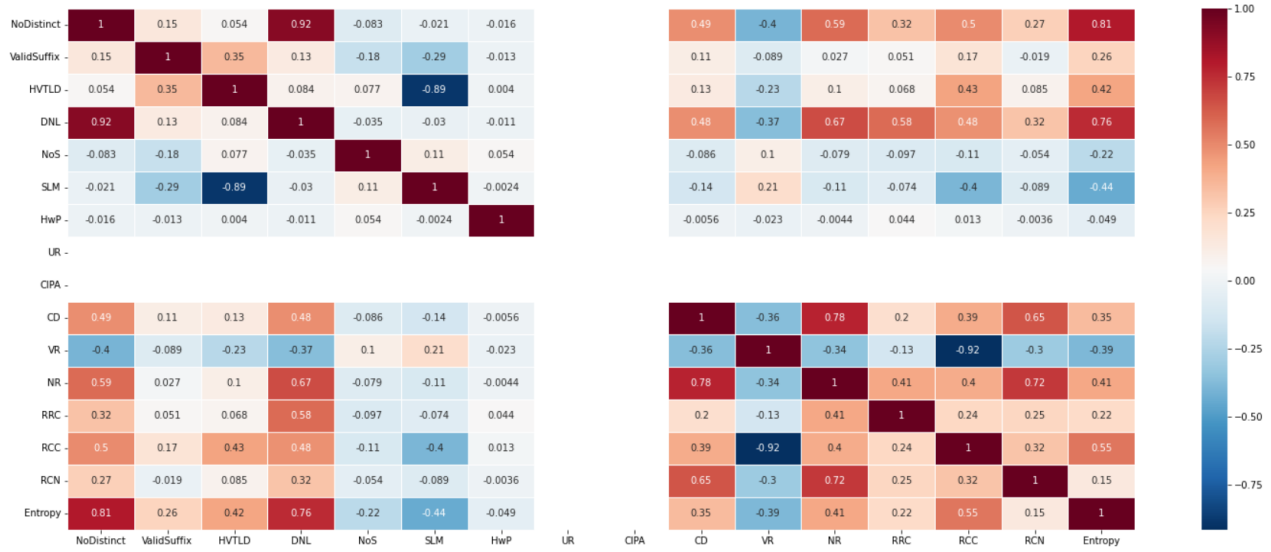
Hệ số tương quan có giá trị từ [-1, 1], với giá trị -1 hoặc 1 thể hiện mối liên quan tuyệt đối giữa hai biến số; ngược lại giá trị 0 thể hiện rằng không có bất kỳ mối quan hệ nào giữa hai biến.

Tính toán ma trận tương quan của các đặc trưng trên các bộ dữ liệu AADR, 360NL, UMUDGA và OSINT, cho kết quả lần lượt được thể hiện tại Hình 2.1, Hình 2.2, Hình 2.3 và Hình 2.4.

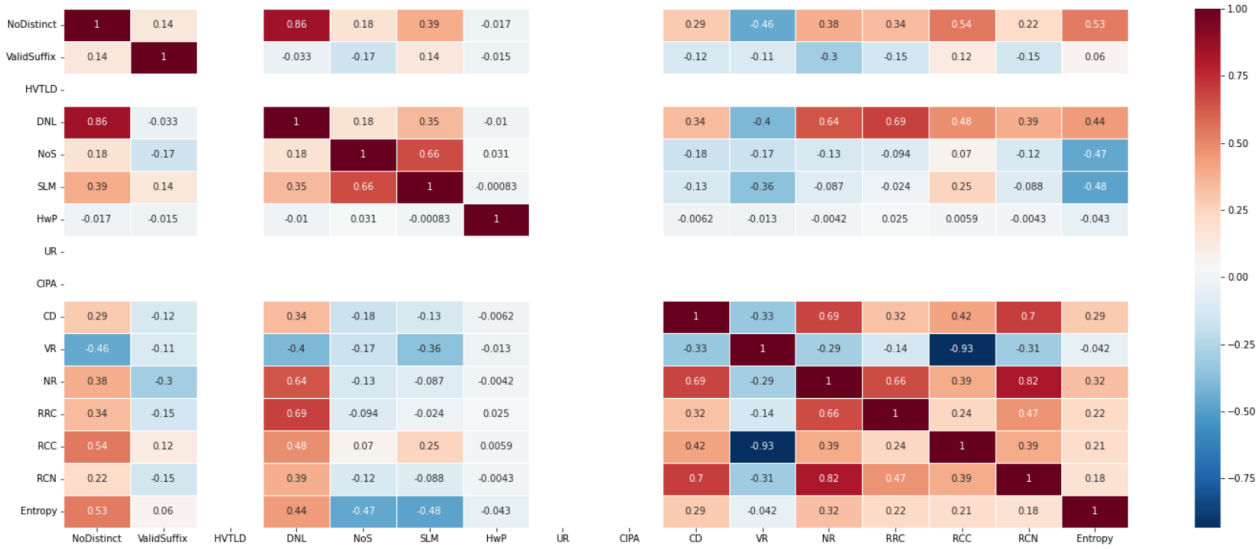


Hình 2.1. Ma trận tương quan các thuộc tính trên tập AADR

Hình 2.2. Ma trận tương quan các thuộc tính trên tập 360NL



Hình 2.3. Ma trận tương quan trên tập OSINT



Hình 2.4. Ma trận tương quan trên tập UMUDGA

Kết quả lựa chọn đặc trưng trên 04 bộ dữ liệu (trình bày tại 1.3.5), được liệt kê tại Bảng 2.2:

Bảng 2.2. Các đặc trưng được lựa chọn làm đầu vào cho thuật toán NCM

STT	AADR	360NetLab	OSINT	UMUDGA
1	CIPA	RCC	DNL	RCC
2	HVTLD	VR	ND	VR
3	VR	Entropy	VR	Entropy
4	RCC	ND	RCC	ND
5	ND	DNL	Entropy	DNL
6	Entropy	NR	NR	NR
7	RCN	CD	CD	CD

2.1.2.2. Thực nghiệm và đánh giá

Thuật toán NCM được thể hiện bằng ngôn ngữ lập trình Python, chạy trên Google CoLab, sử dụng CPU và 12 GB RAM. Hiệu quả của thuật toán được đánh giá thông qua các tham số A.Precision, A.Recall và A.F_1-Score.

Lần lượt đánh giá trên các bộ dữ liệu AADR, 360NL, OSINT và UMUDGA. Kết quả được thể hiện tại Bảng 2.3:

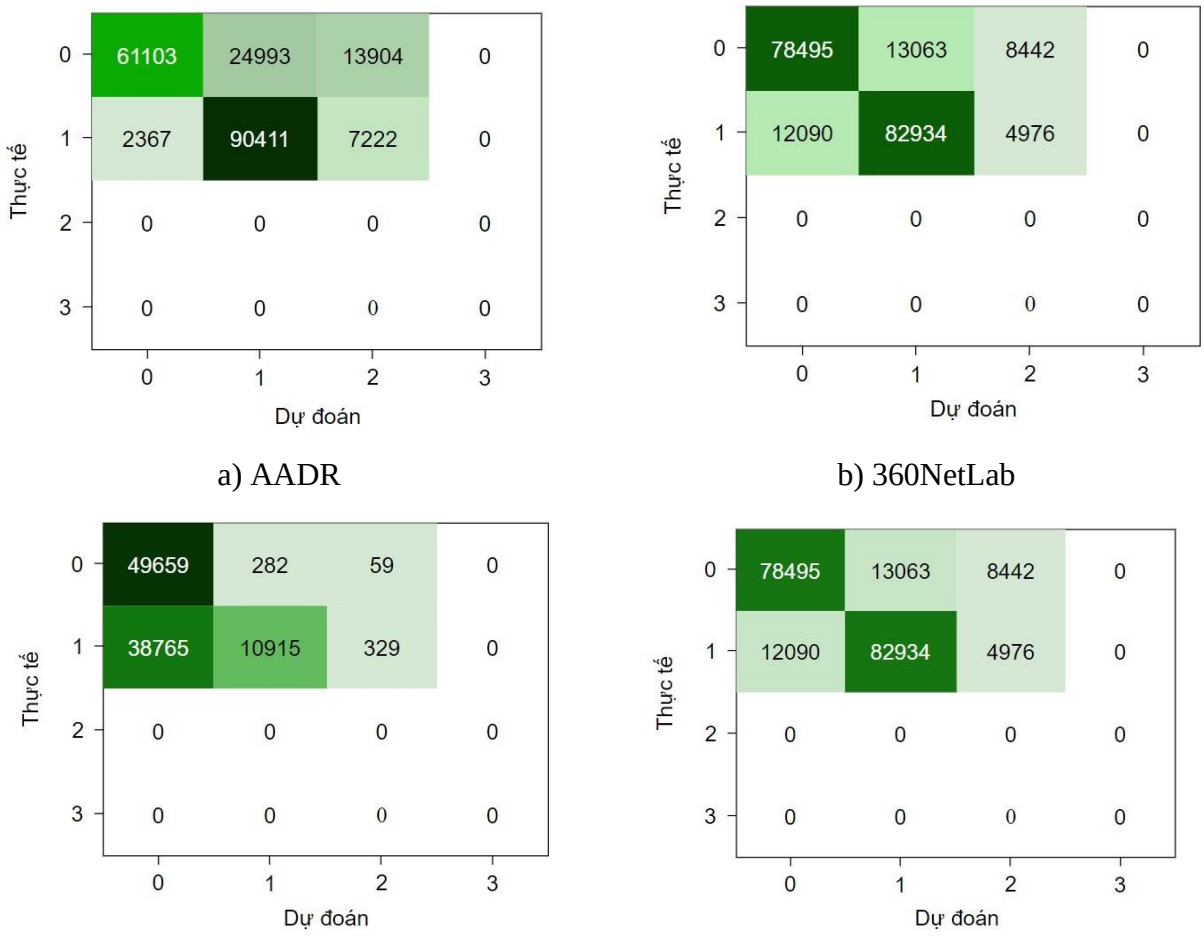
Bảng 2.3. Kết quả phát hiện DGA Botnet của thuật toán NCM trên 04 bộ dữ liệu

	A.Precision	A.Recall	A.F_1-Score
AADR	0,87	0,76	0,79

360NL	0,87	0,81	0,84
OSINT	0,77	0,61	0,54
UMUDGA	0,87	0,81	0,84

Về tổng thể, thuật toán NCM có thông số chung đại diện cho A.Precision và A.Recall là A.F_1-score trong khoảng từ 0,54 đến 0,84 trong 04 bộ dữ liệu đánh giá. Thuật toán hoạt động tốt nhất trên bộ dữ liệu 360NL và UMUDGA với A.F_1-score đều là 0,84; và kém hiệu quả nhất trên bộ dữ liệu OSINT với A.F_1-score đạt 0,54. Trong cả 04 trường hợp, chỉ số A.Precision thường cao hơn so với A.Recall. Điều này có ý nghĩa là giải pháp NCM có khả năng phát hiện đúng các tên miền độc hại hiệu quả hơn.

Các ma trận nhầm lẫn trên 04 bộ dữ liệu được cho tại Hình 2.5. Trong đó, nhãn 0 và 1 tương ứng là lành tính hoặc DGA Botnet đối với từng bộ dữ liệu. Nhãn 2 tương ứng với nhiễu và nhãn 3 là không phân loại.



a) AADR b) 360NetLab

c) OSINT d) UMUDGA

Hình 2.5. Ma trận nhầm lẫn của thuật toán NCM trên 04 bộ dữ liệu

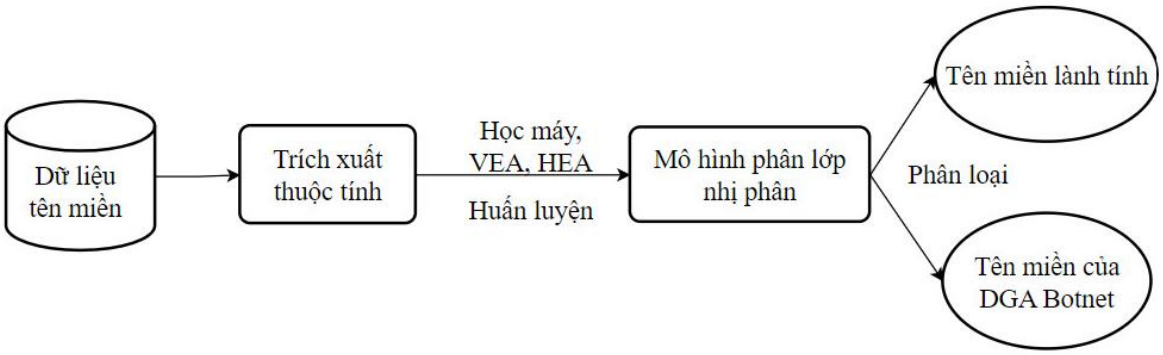
Hình 2.5 cho thấy, thuật toán NCM có khả năng phân loại chưa được tốt giữa tên miền lành tính và tên miền độc hại. Khả năng phát hiện đúng các tên miền độc hại là tốt hơn so với khả năng xác định đúng các tên miền lành tính. Tuy nhiên, nó cũng chưa thể phân loại một số lượng đáng kể các tên miền, chúng được xếp vào dạng nhiễu, nghĩa là chưa đủ căn cứ để xếp vào một lớp cụ thể. Mức độ nhiễu thể hiện rõ nhất trên bộ dữ liệu AADR và thấp nhất trên bộ dữ liệu OSINT, được thể hiện bằng số lượng tên miền được phân loại là nhãn 2 và nhãn 3 trong ma trận. Ngoài ra, thuật toán NCM cũng hoạt động không ổn định, khi có sự chênh lệch lớn khi đánh giá trên các bộ dữ liệu khác nhau, với A.F_1-score dao động trong khoảng 0,54 đến 0,84. Đây cũng là một hạn chế của NCM so với các thuật toán khác.

2.2. Phát hiện DGA Botnet sử dụng học máy

2.2.1. Mô hình đánh giá thuật toán học máy

2.2.1.1. Mô hình chung

Các giai đoạn trong quá trình sử dụng học máy để phát hiện DGA Botnet được thể hiện ở Hình 2.6.



Hình 2.6. Sơ đồ mô hình huấn luyện, đánh giá

Theo đó, với dữ liệu đầu vào là các tên miền, bao gồm cả lành tính và độc hại đã được gán nhãn, sử dụng n-gram để tách các tên miền và kỹ thuật TF-IDF để biểu diễn các đặc trưng.

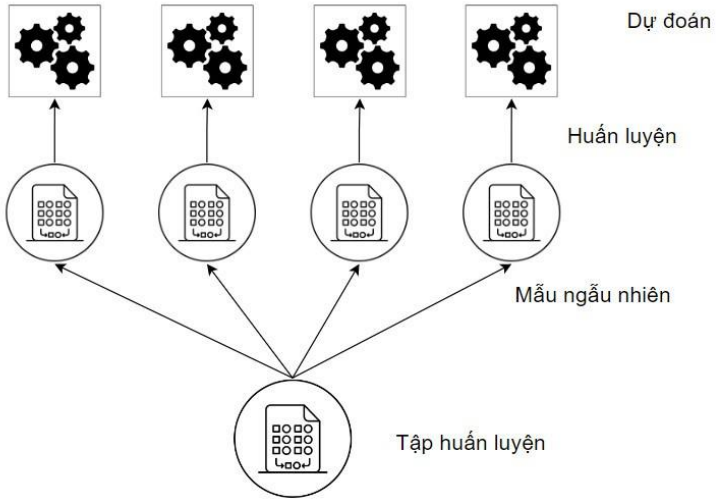
- Đối với học máy, NCS sử dụng các thuật toán sau: Support Vector Machines - SVM, Logistic Regression - LR, Naive Bayes - NB, Neural Networks - NN, Decision Trees - DT, Random Forests - RF, k-Nearest Neighbour -k-NN và Adaptive Boosting - AB.

- Đối với mô hình học kết hợp dựa trên bình chọn, NCS đề xuất hai mô hình VEA và HEA.

2.2.1.2. Học kết hợp dựa trên bình chọn

Gọi các mô hình đơn lẻ là các bộ phân loại yếu - Weak Classifier, ta có thể cải thiện hơn nữa độ chính xác của mô hình này bằng việc kết hợp các bộ phân loại yếu thành bộ phân loại mạnh hơn. NCS đề xuất hai mô hình gọi là VEA và HEA dựa trên cơ chế Ensemble Learning. Mục tiêu của bộ phân loại mới là dựa trên kết quả phân loại của mô hình đơn để bình chọn và tạo nên mô hình mới có độ chính xác cao hơn.

Mô hình VEA và HEA sử dụng phương thức Bagging & Pasting, được thể hiện tại Hình 2.7:



Hình 2.7. Phương thức Bagging & Pasting trong mô hình VEA và HEA

Trong phương thức này, dữ liệu từ tập huấn luyện sẽ được chia thành các mẫu ngẫu nhiên (hay các tập huấn luyện con). Các mẫu ngẫu nhiên này có phân phối các nhãn tương tự như tập huấn luyện gốc, được sử dụng cho việc huấn luyện các mô hình đơn. Dưới dạng toán học, l mẫu Bootstrap tương ứng với l mẫu ngẫu nhiên có kích thước b.

Ta có biểu diễn:

$$\{z_1^1, z_2^1, \dots, z_b^1\}, \{z_1^2, z_2^2, \dots, z_b^2\}, \dots, \{z_1^l, z_2^l, \dots, z_b^l\}$$

với z_i^j là phần tử thứ i của mẫu Bootstrap thứ j với $(1 \le i \le b; 1 \le j \le l)$

Tương ứng với l mẫu Bootstrap ta có l model đơn lẻ (hay model yếu). Quá trình huấn luyện các mô hình này được thực hiện song song.

$$m_1(.), m_2(.), \dots, m_l(.)$$

Trong các bài toán phân lớp, với l mô hình đơn, ta nhận được l kết quả phân loại tương ứng là nhãn của dữ liệu đầu vào. Ta có n nhãn từ $1 - n$. Kết quả cuối cùng được lựa chọn dựa trên kỹ thuật Voting như sau:

- Voting Ensemble Algorithm (VEA): Đưa ra kết quả cuối cùng dựa trên việc kết hợp các xác suất dự đoán của các bộ phân loại đơn. Sau khi tính trung bình xác suất lựa chọn theo từng lớp của các nhãn, nhãn được lựa chọn là nhãn t có tổng thể xác xuất cao nhất, thỏa mãn:

$$\sum_{i=1}^{l} v_i^t \geq \sum_{i=1}^{l} v_i^k \ với i \ \forall k \in (1, n)$$

Trong đó, v_i^k là xác suất phần tử thuộc về lớp k theo kết quả phân loại của mô hình m_i. Với $1 \leq k \leq n$.

Mô hình VEA bao gồm các bộ phân loại sau:

$$VEA \begin{cases} w_1 : LG \\ w_2 : NN \\ w_3 : SVM \end{cases}$$

- Hard Ensemble Algorithm (HEA): Tương tự như VEA, nhưng mô hình HEA sẽ lựa chọn nhãn dựa trên tổng số lượng bầu lớn nhất. Nhãn cuối cùng t sẽ được chọn khi đó là nhãn mà được dự đoán bởi nhiều bộ phân loại nhất. Khi đó, nhãn t cần thỏa mãn điều kiện sau:

$$\sum_{i=1}^{l} h_i^t \geq \sum_{i=1}^{l} h_i^k \ với i \ \forall k \in (1, n)$$

Trong đó: h_i^t có giá trị là 1 nếu mô hình m_i dự đoán t là nhãn điểm cần phân loại, và có giá trị là 0 trong trường hợp ngược lại.

Mô hình HEA gồm các bộ phân loại đơn sau:

$$HEA \begin{cases} w_1 : LG \\ w_2 : NN \\ w_3 : AB \end{cases}$$

2.2.1.3. Trích xuất thuộc tính

NCS sử dụng TF-IDF (Term-Frequency – Inverse Document Frequency) để trích xuất thuộc tính của tên miền [54]. Kỹ thuật này tính toán tần suất xuất hiện của một ký tự hoặc nhóm ký tự trong văn bản, thường là một từ nếu đặt trong ngữ cảnh

là một câu văn. Kỹ thuật này được sử dụng để hỗ trợ đánh giá tầm quan trọng của một từ trong một câu văn hoặc một đoạn văn bản.

Trong đó:

- TF: Tần suất xuất hiện: Là tần suất xuất hiện của một từ trong văn bản, được tính tỉ lệ theo số lần xuất hiện của từ có số lần xuất hiện nhiều nhất trong văn bản d. Giá trị này cho biết một từ xuất hiện nhiều hay ít, được tính bằng công thức (2.18):

$$TF(t,d) = \frac{f(t,d)}{max\{f(w,d): w \in d\}} \quad (2.18)$$

Trong đó:

- $TF(t,d)$ là tần suất xuất hiện của từ t trong văn bản d;
- $f(t,d)$ là số lần xuất hiện của từ t trong văn bản d;
- max $\{f(w,d): w \in d\}$ là số lần xuất hiện của từ có số lần xuất hiện nhiều nhất trong văn bản d.

- IDF Nghịch đảo tần suất của văn bản: Khái niệm này cũng để chỉ tần suất xuất hiện của một từ nhưng được dành cho những từ không mang nhiều ý nghĩa. Giá trị này giúp giảm bớt sự ảnh hưởng của những từ xuất hiện nhiều nhưng không mang nhiều ý nghĩa vào kết quả đánh giá chung.

Giá trị IDF được tính bằng công thức (2.19):

$$IDF(t,D) = log\frac{|D|}{|\{d \in D: t \in d\}|} \quad (2.19)$$

Trong đó:

- $IDF(t,D)$ là giá trị IDF của từ t trong tập văn bản D;
- $|D|$ là tổng số văn bản trong tập D;
- $|\{d \in D: t \in d\}|$ là số văn bản trong tập D có chứa từ t.

Cơ số logarit trong trường hợp này không nhằm thay đổi giá trị IDF mà nhằm mục đích giới hạn giá trị nhận được trong một khoảng xác định. Ngoài ra, ta có mối quan hệ giữa TF và IDF như công thức (2.20):

$$TF - IDF(t,d,D) = TF(t,d) \times IDF(t,D) \quad (2.20)$$

Quá trình trích xuất thuộc tính như sau: Đầu tiên, sử dụng đặc trưng n-gram để tách tên miền thành các nhóm ký tự n-gram. Tiếp đến, sử dụng kỹ thuật TF-IDF để biểu diễn các đặc trưng rút trích thành các vector biểu diễn tần suất của các n-gam đại diện cho tên miền. Các vector đặc trưng n-gram này được sử dụng làm đầu vào cho các mô hình học máy.

2.2.1.4. Bộ dữ liệu đánh giá

Đánh giá được thực hiện trên bộ dữ liệu UMUDGA. NCS trích xuất 1.000.000 tên miền lành tính làm nhãn 0 và 20.000 mẫu tên miền của từng họ DGA Botnet để tổng hợp làm nhãn 1. Tỉ lệ số lượng mẫu giữa hai nhãn 0 và 1 là 50%-50%, đảm bảo sự cân bằng dữ liệu. Chi tiết về số lượng mẫu được cho ở Bảng 2.4:

Bảng 2.4. Số lượng mẫu dành cho phát hiện DGA Botnet sử dụng học máy

	Nhãn 0 (Lành tính)	**Nhãn 1 (DGA Botnet)**
Số lượng mẫu cho mỗi họ tên miền	1.000.000	20.000
Số lượng họ tên miền	1	50
Tổng số lượng mẫu	1.000.000	1.000.000
Tỉ lệ của nhãn trong toàn bộ tập dữ liệu	50%	50%

2.2.2. Kết quả phát hiện DGA Botnet của các mô hình học máy

Bảng 2.5 thể hiện kết quả các độ đo Precision, Recall và F_1-score khi sử dụng các thuật toán học máy để phát hiện DGA Botnet.

Bảng 2.5. Kết quả phát hiện DGA Botnet sử dụng học máy trên bộ dữ liệu UMUDGA

Mô hình học máy	**A.Precision**	**A.Recall**	**A.F_1-score**
LR	0,97	0,97	0,97
NB	0,93	0,89	0,91
DT	0,93	0,95	0,94
NN	0,97	0,97	0,97
SVM	0,97	0,96	0,97
RF	0,74	0,82	0,77
k-NN	0,97	0,66	0,78
AB	0,83	0,85	0,84

Kết quả trên cho thấy, hầu hết các thuật toán học máy đạt được độ chính xác cao trong bài toán phát hiện DGA Botnet. Mô hình LG, NN và SVM, với A.F_1-score đạt từ 0,97 trở lên đạt kết quả tổng thể cao nhất. Mô hình có kết quả thấp nhất là RF với A.Precision, A.Recall và A.F_1-score lần lượt đạt 0,74, 0,82 và 0,77.

Các mô hình LG, NN, SVM, AB và DT có sự cân bằng giữa A.Precision và A.Recall. Tuy nhiên, các mô hình như NB, RF và k-NN lại có sự chênh lệch đáng kể. Trong đó, NB và k-NN có A.Precision đạt lần lượt là 0,93 và 0,97, thì A.Recall lại chỉ đạt lần lượt là 0,89 và 0,66. Điều này chứng tỏ hai mô hình có khả năng phát hiện chính xác các tên miền độc hại, nhưng tỉ lệ phát hiện nhầm các tên miền lành tính thành độc hại cũng cao. Mô hình RF ngược lại với A.Precision thấp hơn nhiều so với A.Recall, lần lượt đạt 0,74 và 0,82.

Một nhận xét nữa đó là mô hình AB và RF cho độ chính xác thấp hơn so với DT, mặc dù chúng cùng dựa trên cơ chế Voting với bộ phân loại yếu là DT. Điều này được lý giải bởi việc sử dụng tham số mặc định của mô hình.

2.2.3. Kết quả phát hiện DGA Botnet của mô hình học kết hợp

Trong phần này, NCS tiến hành sử dụng hai mô hình học kết hợp để phát hiện DGA Botnet. NCS cũng bổ sung các nội dung về kết quả đánh giá trung bình của các mô hình học máy đơn, mô hình mạnh nhất là NN và mô hình yếu nhất là RF. Kết quả đánh giá được tổng hợp và thể hiện tại Bảng 2.6.

Bảng 2.6. Kết quả phát hiện DGA Botnet của mô hình VEA và HEA trên bộ dữ liệu UMUDGA

Thuật toán	**A.Precision**	**A.Recall**	**A.F_1-score**
Trung bình các mô hình đơn	*0,92*	*0,88*	*0,89*
Neural Network	0,97	0,97	0,97
Random Forest	0,74	0,82	0,77
VEA	0,98	0,99	0,98
HEA	0,97	0,97	0,97

Bảng 2.6 cho thấy, cả hai mô hình VEA và HEA đều có độ chính xác cao hơn so với giá trị trung bình của các mô hình đơn, với A.F_1-score lần lượt là 0,98 và 0,97, coa hơn so với trung bình là 0,89.

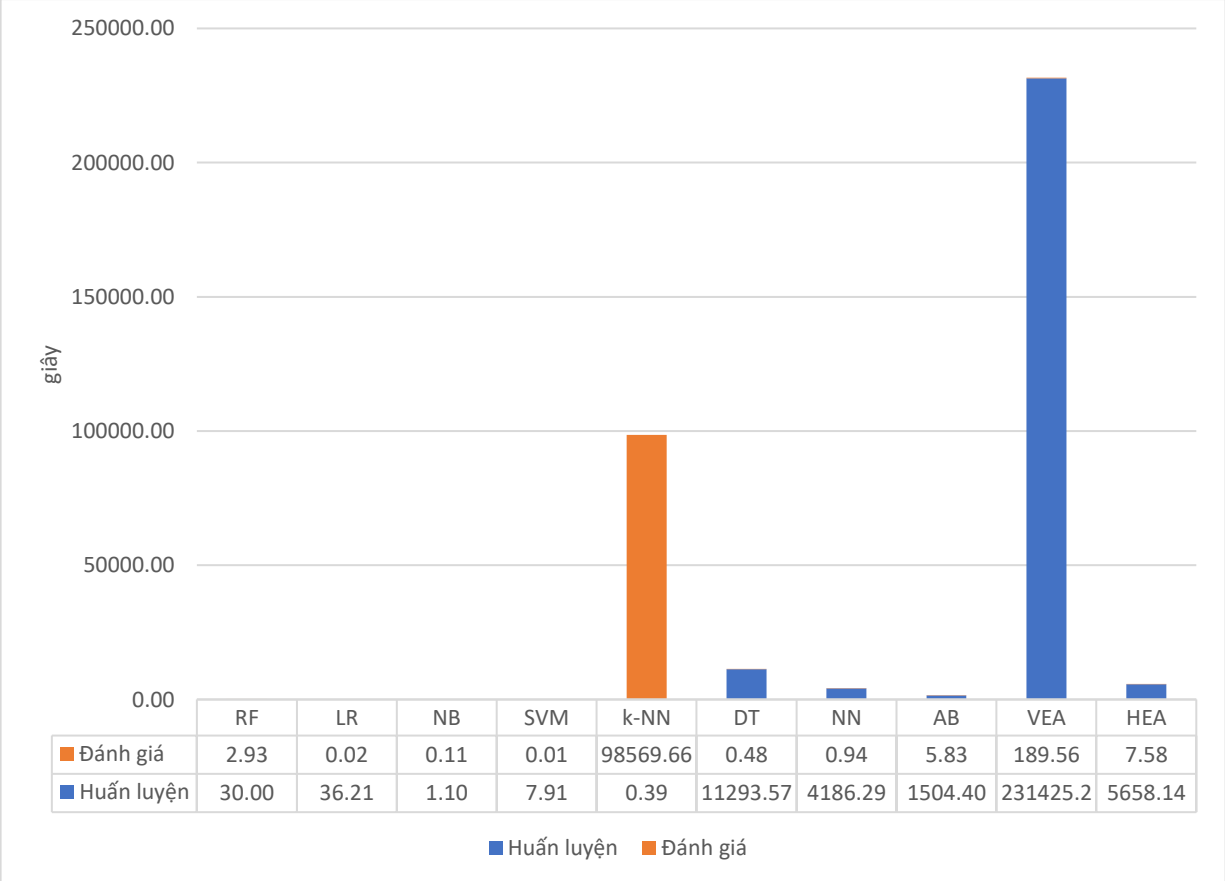
So sánh với mô hình đơn yếu nhất là RF có A.F_1-score là 0,77, ta thấy rằng VEA và HEA hiệu quả hơn rõ rệt về độ chính xác của mô hình. Đối với mô hình mạnh nhất là NN có A.F_1-score là 0,97, mô hình VEA cải thiện độ chính xác tăng thêm 0,01, với

A.F_1-score tăng từ 0,97 lên 0,98. Điều này được giải thích bởi VEA được kết hợp của cả ba mô hình đơn đã cho kết quả tốt là LG, NN và SVM. Trong trường hợp ngược lại, HEA tuy cải thiện rõ rệt độ chính xác so với mức trung bình, nhưng không có cải thiện so với mô hình đơn tốt nhất là NN. Điều này giải thích bởi việc ta đã đưa vào một bộ phân loại yếu là AB và cơ chế bình chọn không dựa trên trọng số của HEA.

Các kết quả trên cũng cho thấy rằng, cơ chế học dựa trên bình chọn này hiệu quả trong việc kết hợp ưu điểm của các mô hình yếu, tạo nên một mô hình mạnh hơn. Đồng thời, cả hai mô hình VEA và HEA đều có một sự cân bằng trong A.Precision và A.Recall, thể hiện rằng khả năng phát hiện không bị thiên vị về một phía nào.

2.2.4. Thời gian huấn luyện và đánh giá của các mô hình học máy

Thời gian huấn luyện và đánh giá của các mô hình cũng là vấn đề được quan tâm. Hình 2.8 thể hiện thời gian dành cho việc huấn luyện và đánh giá của các mô hình học máy đơn và hai mô hình học kết hợp VEA, HEA.



	RF	LR	NB	SVM	k-NN	DT	NN	AB	VEA	HEA
Đánh giá	2.93	0.02	0.11	0.01	98569.66	0.48	0.94	5.83	189.56	7.58
Huấn luyện	30.00	36.21	1.10	7.91	0.39	11293.57	4186.29	1504.40	231425.2	5658.14

Hình 2.8. Thời gian huấn luyện và thực thi của VEA, HEA và các mô hình học máy trên bộ dữ liệu UMUDGA

Về thời gian huấn luyện, các mô hình DT, RF, AB và NN, VEA và HEA có thời gian huấn luyện khá lâu, đều từ 1.504,40 giây trở lên. Đặc biệt, mô hình VEA có thời

gian huấn luyện rất lâu là 231.425,23 giây. Ở chiều ngược lại, các mô hình như NB, SVM, RF, LR lại có thời gian nhanh hơn rất nhiều với lần lượt 1,10, 7,91, 30,00 và 36,21 giây mà vẫn đảm bảo độ chính xác ở mức cao như đã phân tích ở trên. Cần lưu ý rằng, thông thường các thuật toán học máy sẽ được thực thi trên CPU với tốc độ và khả năng tính toán hạn chế. Điều này có thể được cải thiện bởi các mô hình học sâu có thể chạy trên GPU giúp tiết kiệm nhiều thời gian cho giai đoạn huấn luyện.

Về thời gian thực thi, ngoại trừ k-NN, các mô hình còn lại đều có thời gian thực thi chiếm một tỉ lệ thấp so với thời gian huấn luyện, với các giá trị từ 0,02 giây của LR đến cao nhất là 189,56 giây của VEA. Một điều cần lưu ý là mô hình k-NN thực chất không cần thời gian huấn luyện (0,39 giây) mà sẽ tiến hành phân lớp luôn với thời gian là 98.569,66 giây.

2.3. Kết luận Chương 2

Trong chương 2, NCS trình bày các kết quả đánh giá về hướng tiếp cận sử dụng NCM và học máy để bài toán phát hiện DGA Botnet.

- Với hướng tiếp cận sử dụng NCM. NCS đã đưa vào các điều chỉnh từ thuật toán gốc để phù hợp với bài toán phát hiện DGA Botnet. Đánh giá trên 04 bộ dữ liệu cho thấy, mặc dù có tốc độ thực thi nhanh, hạn chế của NCM là độ chính xác thấp hơn đáng kể so với các thuật toán học máy.

- Với hướng tiếp cận sử dụng học máy, các đánh giá trên 04 bộ dữ liệu cho thấy, các mô hình học máy mang lại độ chính xác cao hơn rất đáng kể so với NCM, với cao nhất là thuật toán NN với A.F_1-score đạt 0,97. Độ chính xác này được cải thiện khi sử dụng các mô hình kết hợp dựa trên cơ chế bình chọn là VEA và HEA, tuy nhiên lại đòi hỏi nhiều chi phí về mặt thời gian.

Cả hai hướng tiếp cận trên đều có những ưu điểm và hạn chế về mặt thời gian hoặc độ chính xác. Đồng thời, việc áp dụng vào bài toán phân loại DGA Botnet là còn hạn chế. Cả hai yếu tố này được giải quyết bằng học sâu mà NCS đề xuất và trình bày ở Chương 3 của luận án.

Một phần kết quả trình bày tại Chương 2 được công bố tại [CT1] [CT3] trong Danh mục các công trình công bố liên quan đến luận án.

Chương 3. PHÁT HIỆN VÀ PHÂN LOẠI DGA BOTNET SỬ DỤNG HỌC SÂU

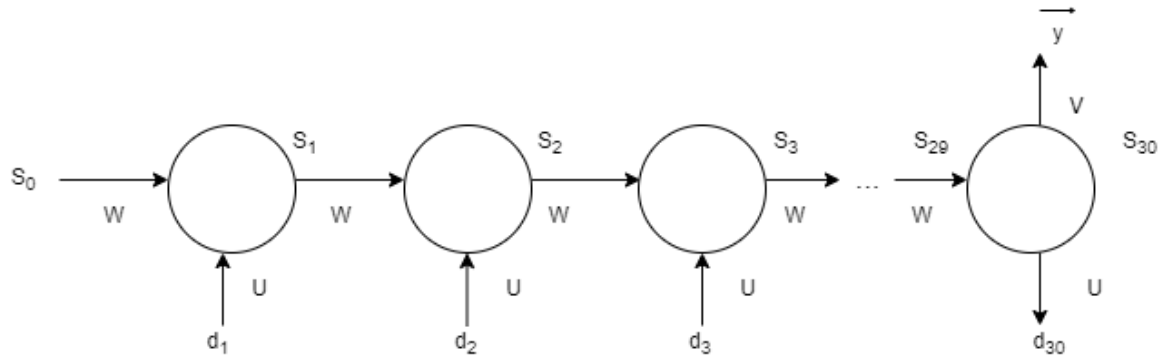
Dựa trên các đánh giá trước đó, NCS tiếp cận theo hướng sử dụng học sâu để nâng cao độ chính xác của bài toán phát hiện và phân loại DGA Botnet. NCS đề xuất mỗi lõi kiến trúc và xây dựng hai mô hình học sâu mới là LA_Bin07 và LA_Mul07 để phát hiện và phân loại DGA Botnet. Các đánh giá cho thấy, mô hình đề xuất có độ chính xác được cải thiện so với các nghiên cứu trước đó.

3.1. Nền tảng kỹ thuật học sâu cho bài toán DGA Botnet

Trong bài toán phát hiện và phân loại Botnet, dữ liệu DNS trong thực tế có tính chất tuần tự, liên tục. Nếu có hoạt động của các Bot, chúng liên tục gửi các truy vấn DNS được lặp lại sau một khoảng thời gian. Các tên miền này có tính chất giống nhau bởi cùng được sinh ra từ một thuật toán và nếu mô hình có thể phát hiện được mối quan hệ giữa một tên miền ở thời điểm hiện tại, với các tên miền ghi nhận được trong quãng thời gian trước đó, chúng có cơ sở để phát hiện và phân loại chính xác những tên miền này.

3.1.1. Mạng Recurrent Neural Network

Recurrent Neural Network - RNN hay mạng nơ-ron hồi quy, được thiết kế để huấn luyện với các dữ liệu đầu vào có dạng chuỗi (sequence/ time-series). Mạng RNN có cấu trúc như Hình 3.1:



Hình 3.1. Cấu trúc mạng RNN trong bài toán DGA Botnet

Cấu trúc RNN này bao gồm:

- $D = \{d_0, d_1, \dots, d_n\}$: Là các tên miền được sử dụng làm đầu vào. Trong trường hợp này, ta có n tên miền tương ứng với n đầu vào tuần tự.

- Mỗi hình tròn là một $State$, mỗi $State$ bao gồm: Input, Output và Activation Function. Với $State_i$ ta có:

+ Input: d_i và s_{i-1}, với s_{i-1} là Output của $State$ trước đó. Riêng trong $State$ đầu tiên thì s_0 được sử dụng để kích hoạt chuỗi.

+ Output: s_i được tính bằng công thức (3.1):

$$s_i = f(U \times d_i + W \times s_{i-1}) \tag{3.1}$$

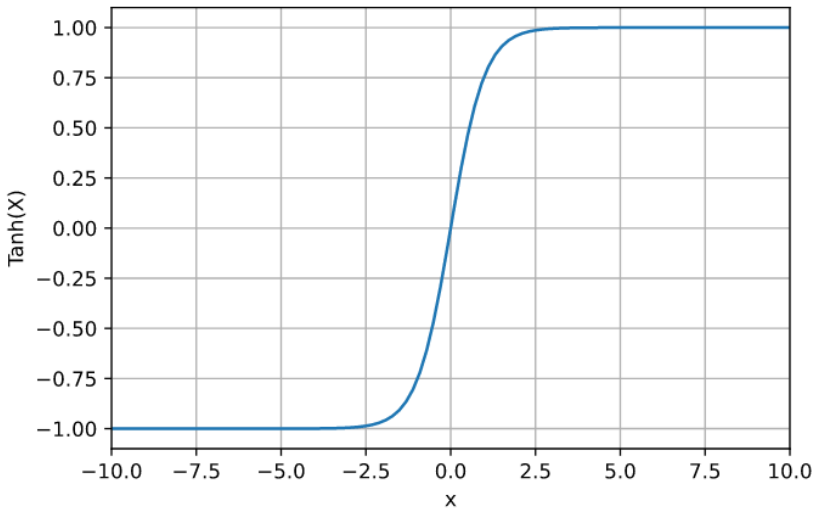
Trong đó, W là ma trận trọng số của mô hình, U là ma trận để biến đổi dữ liệu đầu vào.

+ Acctivation: Hàm f là hàm kích hoạt, thường sử dụng là hàm $Tanh$ hoặc hàm $ReLU$ cho bởi công thức (3.2) và (3.3):

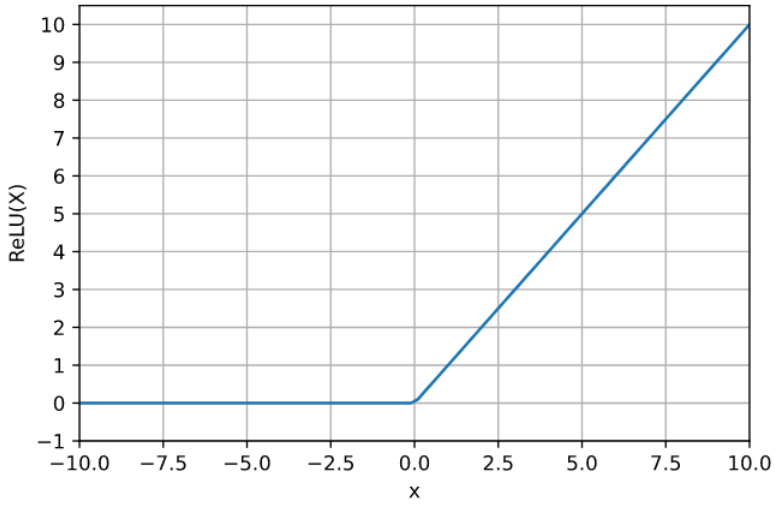
$$Tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.2}$$

$$ReLU(x) = \max(0, x) \tag{3.3}$$

Đồ thị hàm $Tanh$ và hàm $ReLU$ lần lượt được cho tại Hình 3.2 và Hình 3.3. Nhìn vào đồ thị ta thấy, cả hai hàm kích hoạt đều có sự phân hóa rõ ràng ở hai phía của điểm $x = 0$. Tính chất này giúp nó được sử dụng để kích hoạt tìm nhãn cho các mô hình phân lớp.



Hình 3.2. Đồ thị hàm $Tanh$



Hình 3.3. Đồ thị hàm $ReLU$

Cấu trúc RNN trong Hình 3.1 cho thấy, s_i được tính toán dựa trên thông tin từ s_{i-1}, và s_{i-1} lại mang thông tin từ s_{i-2}. Điều này có nghĩa là $State$ sau được tính toán một phần dựa vào $State$ trước đó, ta gọi khả năng này là "nhớ" các đặc điểm trước đó của chuỗi. Lưu ý rằng, s_{i-1} và s_{i-2} được tính toán dựa trên d_{i-1} và d_{i-2}. Như vậy, khả năng "nhớ" này về cơ bản chính là các giá trị trước đó của chuỗi dữ liệu đầu vào.

Giá trị s_0 được xem là kích hoạt, biểu thị rằng ban đầu trạng thái bộ nhớ rỗng. Giá trị nhãn sẽ được tính toán sau $State$ cuối cùng ($State_n$). Ở trạng thái này, mô hình đã học được hết các dữ liệu tên miền đầu vào. Giá trị dự đoán $\hat{y}$ được tính bằng công thức (3.4):

$$\hat{y} = g(V \times s_n) \tag{3.4}$$

Trong đó, g là một hàm kích hoạt phù hợp với bài toán.

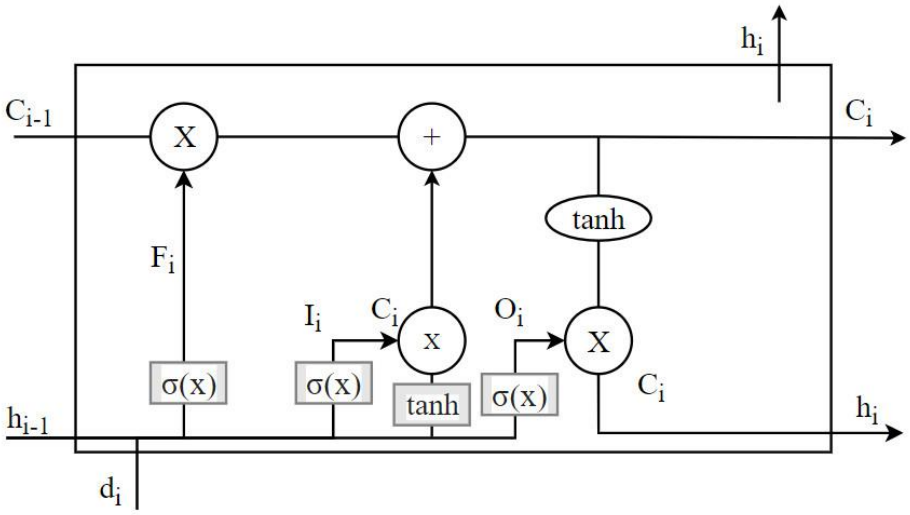
3.1.2. Mạng Long-Short Term Memory và biến thể

Mạng RNN tuy có khả năng nhớ các trạng thái trước đó, nhưng khả năng nhớ này chưa thực sự tốt và một trong các lý do là sự biến mất đạo hàm. Mô hình RNN không thể học các phụ thuộc xa trong quá khứ và dẫn đến khả năng nhớ của mạng RNN bị hạn chế. Nghĩa là, các $State$ hiện tại có xu hướng nhớ các $State$ gần nó nhiều hơn là các $State$ xa. Điều này khiến cho mô hình bị phụ thuộc nhiều vào các $State$ ở cuối. Đồng thời, hiện tượng này cũng khiến cho quá trình suy giảm độ dốc trở nên kém hiệu quả và sự hội tụ trở nên chậm chạp.

Mạng LSTM được phát triển từ mạng RNN, với khả năng học được các phụ thuộc xa hơn. LSTM lần đầu tiên được đề xuất vào năm 1996 bởi Hochreiter [55] và

liên tục được cải tiến sau đó. Mô hình này hoạt động hiệu quả trên nhiều bài toán khác nhau, đặc biệt là các bài toán mà dữ liệu có dạng chuỗi.

Cấu trúc của mạng LSTM cũng dựa trên kiến trúc của RNN, nhưng mỗi $State$ trong chuỗi được cải tiến. Thay vì chỉ có một tầng mạng NN như RNN, LSTM có 04 tầng trong mỗi $State$ và chúng có thể tương tác với nhau. Kiến của của một $State$ trong LSTM được thể hiện tại Hình 3.4.



Hình 3.4. Kiến trúc 04 tầng của một $State$ trong LSTM

Trong đó, tại $State_i$ ta có:

- Các mũi tên chỉ đường đi của các giá trị trong $State_i$, thể hiện tuần tự các bước thực hiện.

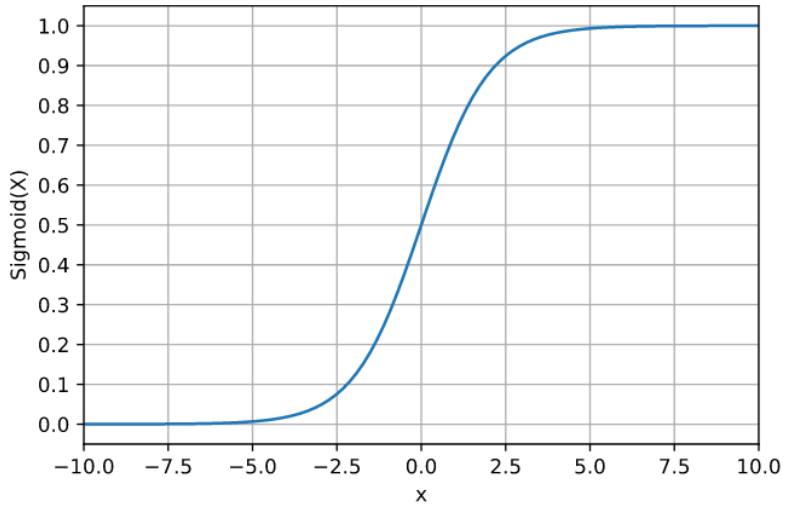
- Output: Gồm c_i và h_i, trong đó c_i là trạng thái của $State$ hiện tại, h_i là trạng thái ẩn của $State$ hiện tại.

- Input: Gồm c_{i-1}, h_{i-1} và d_i, trong đó c_{i-1} và h_{i-1} là giá trị của $State$ trước đó $State_{i-1}$, được dùng để tính toán cho $State$ hiện tại, đảm bảo khả năng "nhớ" của mạng LSTM; d_i là giá trị tên miền đầu vào.

Trong một $State$ của LSTM, ta sử dụng thêm một hàm kích hoạt là $Sigmoid$, được tính bằng công thức (3.5):

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.5}$$

Hình 3.5 thể hiện đồ thị của hàm $Sigmoid$.



Hình 3.5. Đồ thị hàm $Sigmoid$

Bên trong một $State$ của LSTM, được thiết kế ba cổng gồm cổng Forget Gate F, Input Gate I và Output Gate O. Ở $State_i$, được cho bởi công thức (3.6), (3.7) và (3.8):

$$F_i = \sigma(U_F * d_i + W_F * h_{i-1} + b_F) \tag{3.6}$$

$$I_i = \sigma(U_I * d_i + W_I * h_{i-1} + b_I) \tag{3.7}$$

$$O_i = \sigma(U_O * d_i + W_O * h_{i-1} + b_O) \tag{3.8}$$

Ta có $0 < F_i, I_i, O_i < 1$ vì là kết quả của hàm $Sigmoid$; b_F, b_I và b_O là các sai số để điều chỉnh. Phép $*$ là phép nhân Element-Wise;

Xem xét kỹ hơn tới các Gate, ta có:

- Input Gate I: Cổng này có chức năng lựa chọn bao nhiêu lượng thông tin đầu vào sẽ được dùng cho $State$ mới. Kết quả của I là giá trị của hàm $Sigmoid$, nằm trong khoảng [0,1]. Một vector khi đi qua hàm này có thể từ không giữ lại được thông tin nào (khi $\sigma = 0$) cho tới được giữ lại thông tin hoàn toàn (khi $\sigma = 1$).

- Forget Gate F: Cổng này sẽ thực hiện việc chọn bao nhiêu thông tin để loại bỏ. Việc thực hiện cũng thông qua hàm $Sigmoid$ tương tự cổng I.

- Output Gate O: Cổng này có chức năng điều chỉnh lượng thông tin có thể ra ngoài, cũng như lượng thông tin truyền tới $State$ tiếp theo.

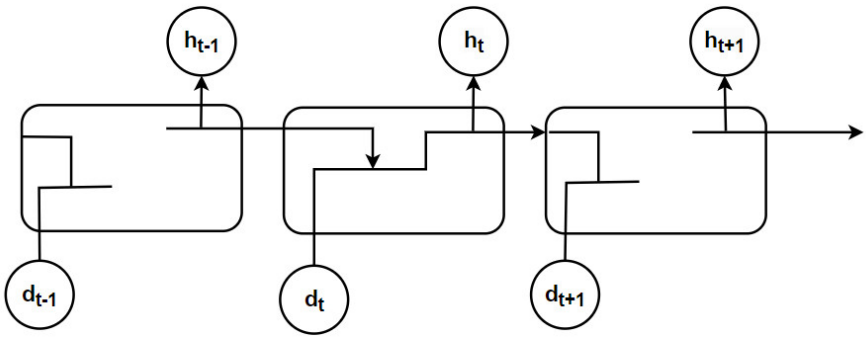
Cuối cùng, các giá trị đầu ra của $State_i$ được tính bằng công thức (3.9), (3.10) và (3.11):

$$\tilde{c}_i = \tanh\left(U_c * d_i + W_c * h_{i-1} + b_c\right) \quad (3.9)$$

$$c_i = F_i * c_{i-1} + I_i * \tilde{c}_i \quad (3.10)$$

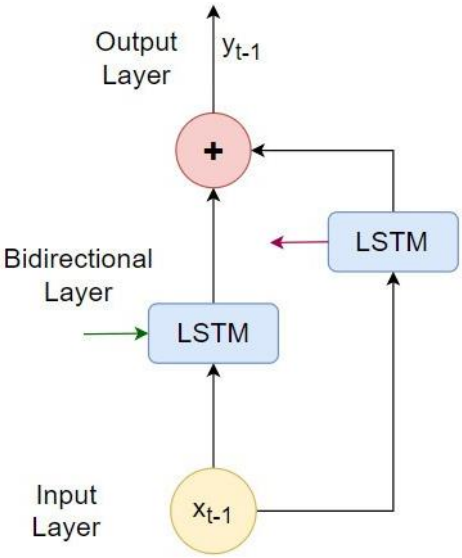
$$h_i = O_i * \tanh\left(c_i\right) \quad (3.11)$$

Với sự có mặt của c_i, mô hình LSTM cải tiến hơn so với RNN, giúp lựa chọn và đưa thông tin đi xa hơn trong chuỗi, được minh họa tại Hình 3.6.



Hình 3.6. Minh họa chuỗi các lớp LSTM

Mạng LSTM hai chiều (Bidirectional Long Short-Term Memory - BiLSTM) là một biến thể cải tiến của mạng LSTM với kiến trúc được mô tả tại Hình 3.7.



Hình 3.7. Kiến trúc mạng BiLSTM

Theo đó, cấu trúc của BiLSTM được thêm một lớp LSTM nữa, giúp làm đảo chiều luồng thông tin đầu vào. Điều này tạo cho chuỗi đầu vào di chuyển ngược lại trong lớp LSTM bổ sung này, từ đó cho phép quá trình huấn luyện tham số được thực hiện theo cả hai hướng là lan truyền ngược và lan truyền tiến.

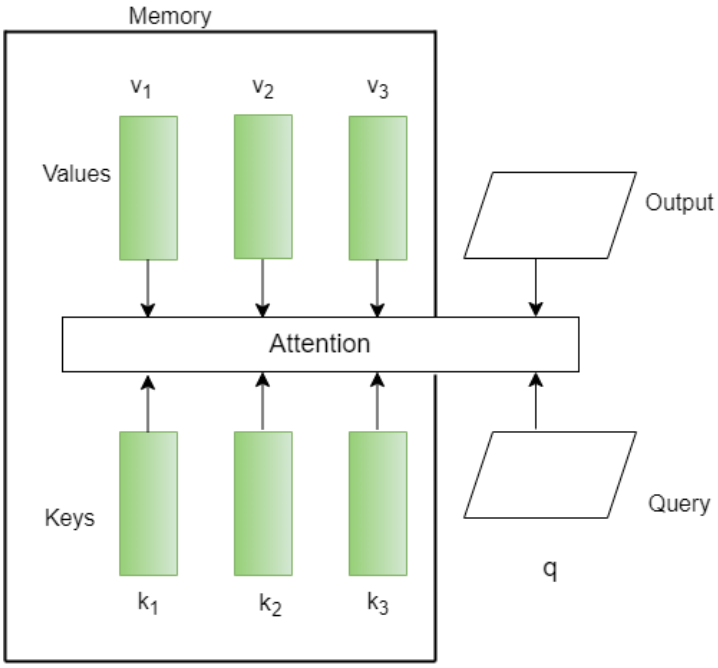
Kiến trúc BiLSTM này có nhiều ưu điểm trong các vấn đề thực tế. Lý do chính của điều này là mọi thành phần của chuỗi đầu vào mang thông tin từ cả quá khứ và

hiện tại, giúp BiLSTM có thể tạo ra một đầu ra có ý nghĩa hơn. Tuy nhiên, hạn chế của BiLSTM là chúng chậm hơn và đòi hỏi nhiều thời gian để huấn luyện hơn so với LSTM gốc.

3.1.3. Cơ chế Attention và biến thể

Mạng LSTM đã khắc phục được hạn chế của RNN về vấn đề nhớ các phụ thuộc ở xa. Tuy nhiên, một vấn đề khác cần quan tâm đó là trọng số của các thuộc tính. Về cơ bản thì mỗi thuộc tính thường có một vai trò khác nhau ảnh hưởng đến nhãn phân loại, có thuộc tính chiếm tỉ trọng nhiều và ngược lại. Cơ chế Attention có thể hỗ trợ giải quyết vấn đề trên và được trình bày tại [56].

Attention là một thành phần gồm ba nhân tố: Query, Key và Value. Chúng được đề xuất để giúp mô hình huấn luyện tập trung hơn vào một đặc điểm nào đó của dữ liệu, hay nói cách khác là tập trung hơn và các thông tin quan trọng. Các thành phần của Attention được thể hiện tại Hình 3.8:



Hình 3.8. Kiến trúc của một lớp Attention

Theo đó, Query q sẽ lấy thông tin tiếp theo cần xử lý. Chúng được chia thành hai phần tương ứng là Key và Value, thông tin thứ i ký hiệu là k_i và v_i

Với mỗi q đầu vào, a_i được gọi là mức độ ảnh hưởng của thông tin thứ i lên q, được tính theo công thức (3.12):

$$a_i = \alpha(q, k_i) \tag{3.12}$$

Các giá trị a_i sau đó được chuẩn hóa để có được b_i. Hàm chuẩn hóa thường được sử dụng là $softmax$ [57] như công thức (3.13):

$$b_i = \frac{exp(a_i)}{\sum_i exp(a_i)}, b = [b_1, b_2, \ldots, b_n]^T \quad (3.13)$$

Các giá trị v_i được tính lại dựa trên b_i. Việc chuẩn hóa giúp cho dữ liệu không bị thay đổi kích cỡ so với mô hình đang huấn luyện. Cuối cùng, đầu ra o là tổng trọng số của các giá trị được tính bằng công thức (3.14):

$$o = \sum_{i=1}^{n} b_i v_i \quad (3.14)$$

RNN và cải tiến của nó là LSTM đã thể hiện hiệu quả của mình với khả năng ghi nhận được sự phụ thuộc thời gian của các từ trong câu hay các thành phần trong tên miền đối với bài toán DGA Botnet. Tuy nhiên, các nghiên cứu mới đã chỉ ra rằng với cơ chế Attention (mà không cần sử dụng RNN) ta có thể cải thiện được kết quả của các tác vụ học.

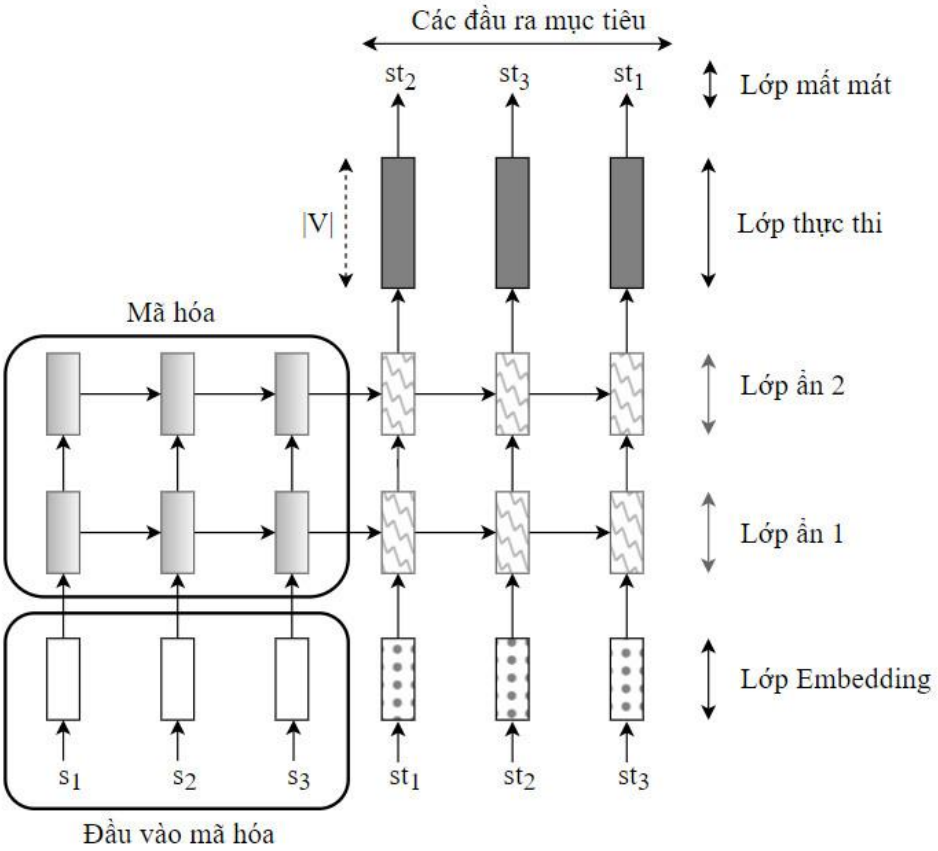
Một dạng của Attention là Self-attention, là cơ chế tự chú ý. Điểm khác biệt của Self-Attention là nó chú ý trọng tâm vào câu hay lớp hiện tại mà nó được thiết kế. Nó xác định mức độ quan trọng của mỗi thành phần trong chuỗi đầu vào đối với từng thành phần khác bằng cách tính toán một trọng số (attention weight) cho mỗi cặp thành phần trong chuỗi. Các trọng số này sau đó được sử dụng để tạo ra một biểu diễn mới cho mỗi thành phần, kết hợp thông tin từ tất cả các thành phần khác trong chuỗi.

Cơ chế này đã chứng minh được sự hiệu quả trong các ứng dụng tóm tắt văn bản, tạo tiêu đề cho hình ảnh. Self-attention đã đóng vai trò quan trọng trong sự phát triển của các mô hình như Transformer, Chat GPT và BERT [58].

3.1.4. Mạng LSTM tích hợp Attention

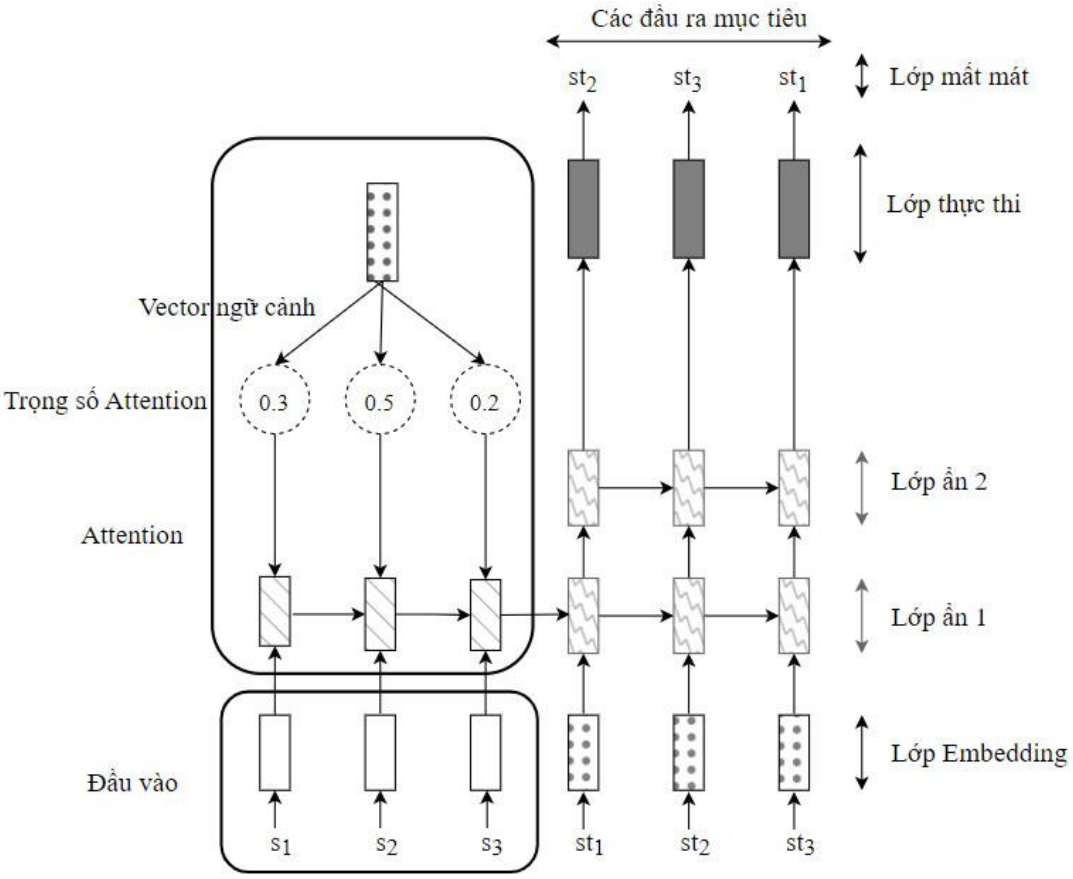
Mạng LSTM có thể nâng cao khả năng học tập qua quá trình huấn luyện với sự có mặt của Attetion, điều này đã được thể hiện ở một số nghiên cứu trước đó và lập luận, thí nghiệm của NCS. Lớp Attention khi được bổ sung vào xác định vai trò giữa các thuộc tính, giúp LSTM điều chỉnh sự tập trung lớn hơn tới các thuộc tính có giá trị hơn.

Hình 3.9 minh họa mô hình LSTM không có Attention, dữ liệu đầu vào s_1, s_2, s_3 sau khi qua bộ mã hóa sẽ được đưa tới các Lớp ẩn 1, Lớp ẩn 2 của mô hình để huấn luyện. Tại các lớp này, vai trò của s_1, s_2, s_3 được mô hình coi là như nhau.



Hình 3.9. Mô hình LSTM truyền thống không có Attention

Hình 3.10 mô tả ví dụ về vai trò của Attention trong mạng LSTM. Sự có mặt của Attention giúp tính toán trọng số cho từng phần tử đầu vào là s_1, s_2, s_3 được gán với bộ trọng số tương ứng $(0{,}3, 0{,}5, 0{,}2)$. Trọng số này giúp các Lớp ẩn 1, Lớp ẩn 2 biết những nội dung nào quan trọng hơn để tập trung hơn.



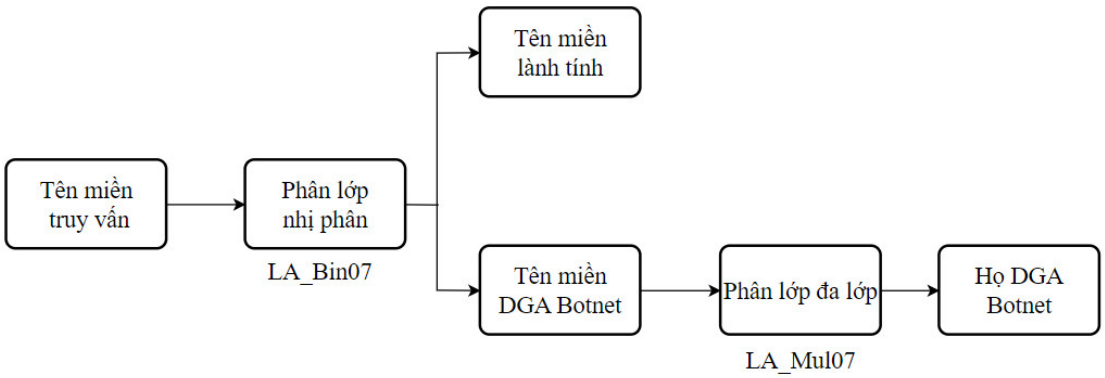
Hình 3.10. Mô hình LSTM cải tiến với Attention

3.2. Đề xuất kiến trúc lõi và hai mô hình học sâu mới

Trong phần này, NCS đề xuất kiến trúc lõi BiLSTM_SelfA_Double và kế thừa để đề xuất hai mô hình LA_Bin07, LA_Mul07 để phát hiện và phân loại DGA Botnet.

3.2.1. Quy trình thực hiện bài toán DGA Botnet

Giải pháp phát hiện và phân loại DGA Botnet với hai mô hình LA_Bin07 và LA_Mul07 được mô tả tại Hình 3.11:



Hình 3.11. Giải pháp phát hiện và phân loại DGA Botnet với hai mô hình học sâu mới LA_Bin07 và LA_Mul07

Đầu tiên, các tên miền truy vấn sẽ được đưa vào mô hình LA_Bin07, mô hình sẽ thực hiện việc gán nhãn đây là tên miền độc hại hay lành tính.

Tiếp theo:

- Đối với các tên miền lành tính, việc truy cập được cho phép diễn ra bình thường.

- Đối với các tên miền độc hại, chúng tiếp tục được đi qua bộ phân loại LA_Mul07 để xác định họ của tên miền DGA Botnet đó.

Việc chia thành hai mô hình LA_Bin07 và LA_Mul07 cho phát hiện và phân loại mang lại các ưu điểm thay vì gộp chung thành mô hình phân lớp đa lớp, thể hiện ở các yếu tố sau:

- Tính linh hoạt: Việc chia hai mô hình phát hiện và phân loại giúp giải pháp linh hoạt hơn trong đánh giá học thuật cũng như trong áp dụng thực tiễn. Ví dụ một hệ thống chỉ cần phát hiện sớm những tên miền độc hại trước để ngăn chặn kịp thời, còn việc phân tích tên miền đó sẽ được thực hiện sau. Phần lớn các nghiên cứu tiếp cận theo hướng này cả hai bài toán này hoặc chỉ tiếp cận ở bài toán phát hiện DGA Botnet.

- Phân loại theo sự tương đồng: Tên miền lành tính và tên miền độc hại của DGA Botnet có điểm khác nhau lớn đó là tên miền lành tính không sinh theo thuật toán còn tên miền độc hại thì sinh theo thuật toán với những quy luật, đặc trưng riêng. Do đó, việc chia hai mô hình sẽ có ý nghĩa: Mô hình phát hiện (nhị phân) sẽ phân loại giữa tên miền sinh theo thuật toán và tên miền không sinh theo thuật toán; Và mô hình phân loại (đa lớp) sẽ phân loại giữa các tên miền cùng sinh theo thuật toán.

- Số lượng mẫu: Các tên miền lành tính được thu thập từ thực tế, có số lượng mẫu không biến đổi nhiều và khoảng 1.000.000 mẫu theo thống kê của Alexa. Trong khi đó, các tên miền DGA Botnet được sinh ra theo thuật toán và thông thường có số lượng mẫu rất lớn, biến thiên trong khoảng rộng, tùy theo thuật toán và không gian ký tự. Do đó việc đưa mẫu lành tính vào để phân lớp cùng mẫu DGA Botnet sẽ có thể dẫn đến sự mất cân bằng mẫu.

3.2.2. Đề xuất kiến trúc lõi của mô hình học sâu

Cơ sở ý tưởng: Một số nghiên cứu liên quan trước đó đã đưa đến những kết quả như sau: Đức và cộng sự [15] cho thấy mạng LSTM hiệu quả hơn RNN trong phát hiện và phân loại DGA Botnet, cả trong trường hợp bộ dữ liệu mất cân bằng; Vinayakumar và cộng sự [20] đưa ra ý tưởng về một mô hình phức hợp là CNN-LSTM hay cho hiệu quả cao hơn một mô hình đơn. Qiao và cộng sự [17] với mạng

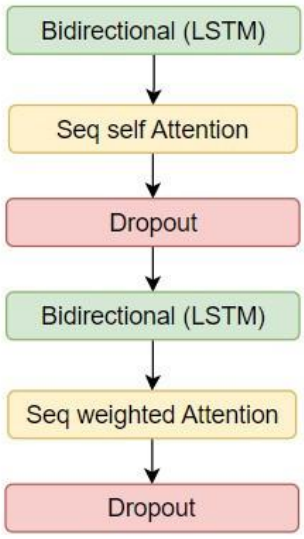
LSTM_AM cho thấy cơ chế chú ý giúp tiếp tục cải thiện độ chính xác đạt được. Namgung và cộng sự [19] thiết kế một mạng phức hợp CNN-BiLSTM tương tự Vinayakumar, nhưng cho thấy BiLSTM là hiệu quả hơn so với mạng LSTM gốc.

Từ đó, NCS đề xuất kiến trúc lõi gọi là BiLSTM_SelfA_Double cho mô hình học sâu phát hiện và phân loại DGA Botnet với thiết kế được thể hiện tại Hình 3.12, với các điểm mới trong thiết kế so với các kết quả trước đó như sau:

(1) Sử dụng mạng BiLSTM thay cho LSTM để nâng cao hiệu quả từ cơ chế hoạt động huấn luyện hai chiều;

(2) Bổ sung cơ chế Selt Attention được biến thể từ Attention, với hai lớp Seq Self Attetion và Seq weighted Attention, giúp tăng cường hiệu quả "chú ý".

(3) Thiết kế mạng hai vòng lặp BiLSTM_SelfA tương ứng với hai lớp Attention để một lớp thực hiện tự chú ý và một lớp có trọng số tương ứng với mạng BiLSTM của vòng đó.



Hình 3.12. Kiến trúc lõi BiLSTM_SelfA_Double đề xuất

So sánh với các kết quả trước đó, kiến trúc lõi đề xuất có những điểm cải tiến được tổng hợp tại Bảng 3.1. Cụ thể bao gồm: Sử dụng mạng BiLSTM để nâng cao hiệu quả quá trình học so với LSTM; bổ sung SelfAttention để tự học chú ý trên lớp hiện tại; kiến trúc Double với hai vòng lặp đạt được sự cân bằng tối ưu giữa độ chính xác đạt được và chi phí thời gian, năng lực tính toán.

Bảng 3.1. So sánh kiến trúc lõi đề xuất với các mô hình liên quan

	Đức và cộng sự [15]	**Qiao và cộng sự [17]**	**Namgung và cộng sự [19]**	**Vinayakumar và cộng sự [20]**	***Mô hình NCS đề xuất***

Thành phần lõi	Cost-sensitive LSTM	LSTM-Attention	BiLSTM và CNN-BiLSTM-Ensemble	LSTM	BiLSTM-SelftAttention
Xử lý đầu vào	Embedding	Embedding (54x128)	Embedding (73x32)	Embedding (None, 91, 128)	Embedding (None, 100, 128)
Cơ chế bổ sung			Ensemble		Double

Kiến trúc lõi BiLSTM_SelfA_Double trên được kế thừa làm lõi của hai mô hình học sâu LA_Bin07 và LA_Mul07 đề xuất.

3.2.3. Xử lý dữ liệu đầu vào

Với dữ liệu đầu vào là các tên miền, NCS xử lý để đưa vào huấn luyện qua hai bước gồm Encoding và Embedding.

(1) Quá trình Encoding: Dữ liệu tên miền sau khi được làm sạch sau quá trình đọc (loại bỏ chuỗi "/n", chuẩn hóa chữ cái hoa thành chữ cái thường) sẽ được mã hóa tương ứng với một số hạng từ 0 đến 39 để có thể tính toán được.

(2) Quá trình Embedding: Dữ liệu sau khi Encoding được vector hóa bằng kỹ thuật Word Embedding được mô tả tại [59], là kỹ thuật được sử dụng phổ biến trong xử lý ngôn ngữ tự nhiên.

- Đầu tiên, các tên miền được làm đầy để đạt chiều dài tối đa maxlen là 100 ký tự, giá trị để làm đầy là pad_value = 40.

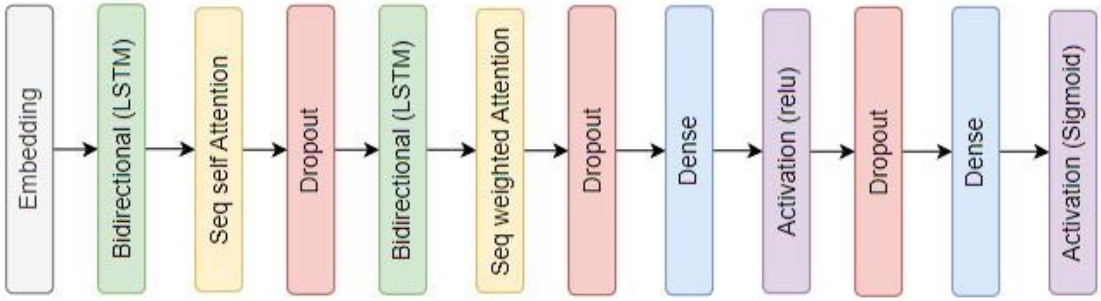
- Số lượng thuộc tính tối đa đưa vào là input_dim = 5000, các thuộc tính này được xử lý nhúng và trở thành vector có số chiều là output_dim = 128.

- Vector với số chiều là 128 sẽ được sử dụng làm đầu vào cho mô hình học sâu.

Word Embedding là một kỹ thuật thể hiện tính hiệu quả cao trong xử lý ngôn ngữ tự nhiên, chúng có ưu điểm so với kỹ thuật Bag of Words bởi giúp giảm số lượng các thuộc tính không có giá trị, từ đó giảm số chiều của vector thuộc tính mà vẫn giữ được đặc trưng của tên miền.

3.2.4. Mô hình LA_Bin07 cho phát hiện DGA Botnet

Mô hình LA_Bin07 đóng vai trò phát hiện DGA Botnet, được thiết kế theo dạng Sequence-to-Sequence kế thừa kiến trúc lõi đề xuất, với cấu trúc chi tiết được thể hiện ở Hình 3.13.



Hình 3.13. Kiến trúc của mô hình LA_Bin07

Trong mô hình LA_Bin07, lớp đầu tiên là Embedding, có chức năng xử lý dữ liệu đầu vào. Tiếp theo là kiến trúc lõi BiLSTM_SelfA_Double, trong đó lớp Dropout với xác suất $p = 0{,}5$, có chức năng giảm số lượng tham số được sinh ra và được áp dụng như sau:

$$h' = \begin{cases} 0 \\ \dfrac{h}{1-p} \end{cases}$$

Theo đó, $h' = 0$ xảy ra với xác suất là p. Đầu ra của hàm kích hoạt trung gian h được thay thế bởi một biến ngẫu nhiên h' có cùng kỳ vọng.

Mô hình được tiếp đến với lớp Dense, giúp kết nối dữ liệu trước đó duỗi dữ liệu ra phù hợp với số nhãn cần phân loại. Hàm ReLU được sử dụng cho kích hoạt những giá trị này. Một lớp Droput kế tiếp để giảm tiếp trọng số. Lớp Dense cuối cùng để kết nối các dữ liệu trước khi được kích hoạt bằng hàm Sigmoid để xác định nhãn.

Chi tiết về mô hình được thể hiện ở Bảng 3.2.

Bảng 3.2. Chi tiết về kích thước đầu và số lượng tham số của mô hình LA_Bin07

STT	Lớp	Kích thước đầu ra	Số tham số
1	Embedding	(None, 100, 128)	640.000
2	Bidirectional LSTM	(None, 100, 256)	263.168
3	Seq Self Attention	(None, None, 256)	16.449
4	Dropout	(None, None, 256)	0
5	Bidirectional LSTM	(None, None, 256)	394.240
6	Seq Weighted Attention	(None, 256)	257
7	Dropout	(None, 256)	0
8	Dense	(None, 64)	16.448
9	Activation (ReLU)	(None, 64)	0

10	Dropout	(None, 64)	0
11	Dense	(None, 1)	65
12	Activation (Sigmoid)	(None, 1)	0
Tổng số tham số			1.330.627

Về chi tiết, kiến trúc mô hình LA_Bin07 được mô tả bao gồm các lớp theo thứ tự như sau:

1. Lớp nhúng (Embedding): Lớp này chuyển đổi đầu vào thành một không gian nhúng có kích thước (None, 100, 128). Tổng số tham số của lớp nhúng là 640.000.

2. Lớp Bidirectional LSTM: Lớp này sử dụng mạng LSTM có kết nối hai chiều, giúp mô hình có khả năng học từ cả hai phía của dữ liệu chuỗi. Kích thước đầu ra của lớp này là (None, 100, 256) và số tham số là 263.168.

3. Lớp Seq Self Attention: Lớp này áp dụng cơ chế Self Attention để tạo ra một ma trận trọng số Attention dựa trên đầu ra của lớp trước đó. Kích thước đầu ra của lớp này là (None, None, 256) và số tham số là 16.449.

3. Lớp Dropout: Lớp này áp dụng kỹ thuật Dropout để ngẫu nhiên loại bỏ một số lượng các đơn vị trong quá trình huấn luyện, nhằm tránh hiện tượng quá khớp. Kích thước đầu ra và số tham số của lớp này giống với lớp Seq Self Attention.

4. Lớp Bidirectional LSTM: Tương tự như lớp thứ 2, lớp này sử dụng mạng BiLSTM với kích thước đầu ra là (None, None, 256) và số tham số là 394.240.

5. Lớp Seq Weighted Attention: Lớp này áp dụng cơ chế Attention dựa trên trọng số cho chuỗi đầu vào, tạo ra một đầu ra có kích thước (None, 256). Số tham số của lớp này là 257.

6. Lớp Dropout: Lớp này có kích thước đầu ra là (None, 256) và số tham số là 0, tương tự như các lớp Dropout trước đó.

7. Lớp Dense: Lớp này áp dụng phép biến đổi tuyến tính và có kích thước đầu ra là (None, 64). Số tham số của lớp này là 16.448.

8. Lớp Activation (ReLU): Lớp này áp dụng hàm kích hoạt ReLU lên đầu ra của lớp trước đó.

9. Lớp Dropout: Tương tự như các lớp Dropout trước đó.

10. Lớp Dense: Lớp này có kích thước đầu ra là (None, 1), nhằm thực hiện phân loại nhị phân. Số tham số của lớp này là 65.

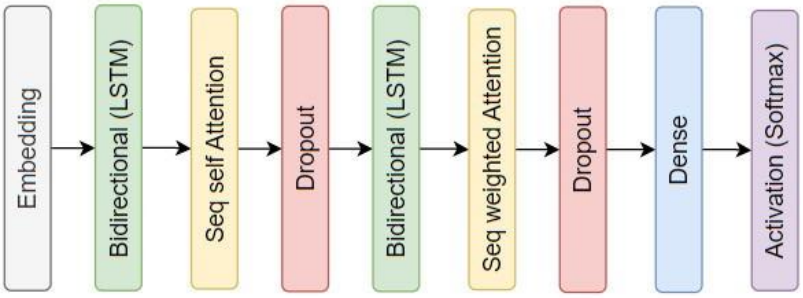
11. Lớp Activation (Sigmoid): Lớp này áp dụng hàm kích hoạt Sigmoid lên đầu ra của lớp trước đó, để đưa ra dự đoán cuối cùng về xác suất tên miền là độc hại hay không.

Tổng cộng, mô hình này có số tham số là 1.330.627.

3.2.5. Mô hình LA_Mul07 cho phân loại DGA Botnet

Đối với bài toán phân loại DGA Botnet, mô hình LA_Mul07 được thiết kế tinh giản hơn cho phù hợp với yêu cầu xác định nhiều nhãn dữ liệu. Một số lớp Dense và Dropout được loại bỏ, hàm kích hoạt được sử dụng là $Softmax$.

Kiến trúc các lớp trong mô hình LA_Mul07 được thể hiện ở Hình 3.14.



Hình 3.14. Cấu trúc đề xuất của mô hình LA_Mul07

Trong mô hình LA_Mul07, lớp đầu tiên là Embedding để đưa dữ liệu vào. Lõi BiLSTM_SelfA_Double được kết thừa tương tự LA_Bin07. Lớp Dense giúp duỗi các giá trị tương ứng và mô hình sẽ xác định nhãn thông qua hàm kích hoạt $Softmax$ ở lớp cuối cùng.

Trong bài toán phân lớp đa lớp, số nhãn đưa vào thường khá nhiều, số tham số cần tính toán có thể tăng nhanh nên một mô hình không quá phức tạp sẽ giảm bớt khối lượng tính toán. Chi tiết về mô hình LA_Mul07 được cho ở Bảng 3.3:

Bảng 3.3. Chi tiết về kích thước đầu và số lượng tham số của mô hình LA_Mul07

STT	Lớp	Kích thước đầu ra	Số tham số
1	Embedding	(None, 100, 128)	640.000
2	Bidirectional LSTM	(None, 100, 256)	263.168
3	Seq Self Attention	(None, None, 256)	16.449
4	Dropout	(None, None, 256)	0
5	Bidirectional LSTM	(None, None, 256)	394.240
6	Seq Weighted Attention	(None, 256)	257
7	Dropout	(None, 256)	0

8	Dense	(None, n)	P_n
9	Activation (Softmax)	(None, n)	0
Tổng số tham số			1.314.114+P_n

Trong đó, P_n là số lớp cần phân loại, giá trị này càng tăng khi mà số lượng lớp càng nhiều.

Về chi tiết, kiến trúc mô hình LA_Mul07 được mô tả bao gồm các lớp theo thứ tự như sau:

1. Lớp nhúng (Embedding): Lớp này chuyển đổi đầu vào thành một không gian nhúng có kích thước (None, 100, 128). Tổng số tham số của lớp nhúng là 640.000.

2. Lớp Bidirectional LSTM: Lớp này sử dụng mạng BiLSTM, giúp mô hình có khả năng học từ cả hai phía của dữ liệu chuỗi. Kích thước đầu ra của lớp này là (None, 100, 256). Số tham số của lớp này là 263.168.

3. Lớp Seq Self Attention: Lớp này áp dụng cơ chế Self Attention để tạo ra một ma trận trọng số attention dựa trên đầu ra của lớp trước đó. Kích thước đầu ra của lớp này là (None, None, 256). Số tham số của lớp này là 16.449.

4. Lớp Dropout: Lớp này áp dụng kỹ thuật Dropout để ngẫu nhiên loại bỏ một số lượng các đơn vị trong quá trình huấn luyện, nhằm tránh hiện tượng quá khớp. Kích thước đầu ra và số tham số của lớp này giống với lớp Seq Self Attention.

5. Lớp Bidirectional LSTM: Tương tự như lớp thứ 2, lớp này sử dụng mạng BiLSTM với kích thước đầu ra là (None, None, 256) và số tham số là 394.240.

6. Lớp Seq Weighted Attention: Lớp này áp dụng cơ chế Attention dựa trên trọng số cho chuỗi đầu vào, tạo ra một đầu ra có kích thước (None, 256). Số tham số của lớp này là 257.

7. Lớp Dropout: Lớp này có kích thước đầu ra là (None, 256) và số tham số là 0, tương tự như các lớp Dropout trước đó.

8. Lớp Dense: Lớp này có kích thước đầu ra là (None, n), với n là số lượng lớp đầu ra mong muốn. Số tham số của lớp này là P_n.

9. Lớp Activation (Softmax): Lớp này áp dụng hàm kích hoạt Softmax lên đầu ra của lớp Dense. Softmax chuyển đổi giá trị đầu ra thành một phân phối xác suất, trong đó tổng các xác suất bằng 1. Kích thước đầu ra của lớp này là (None, n), tương ứng với số lượng lớp đầu ra mong muốn.

Tổng cộng, mô hình này có số tham số là 1.314.114 + P_n.

3.3. Đánh giá hai mô hình học sâu đề xuất

3.3.1. Bộ dữ liệu và môi trường đánh giá

Trong đánh giá của mình, NCS sử dụng 04 bộ dữ liệu cho bài toán phát hiện và 02 bộ dữ liệu gồm AADR và UMUDGA cho bài toán phân loại. Chi tiết về các bộ dữ liệu đã được trình bày tại mục 1.3.5. Các bộ dữ liệu được chia theo tỉ lệ 80%-20% tương ứng với tập huấn luyện và tập đánh giá. Để thuận tiện, các đánh giá được ký hiệu như tại Bảng 3.4.

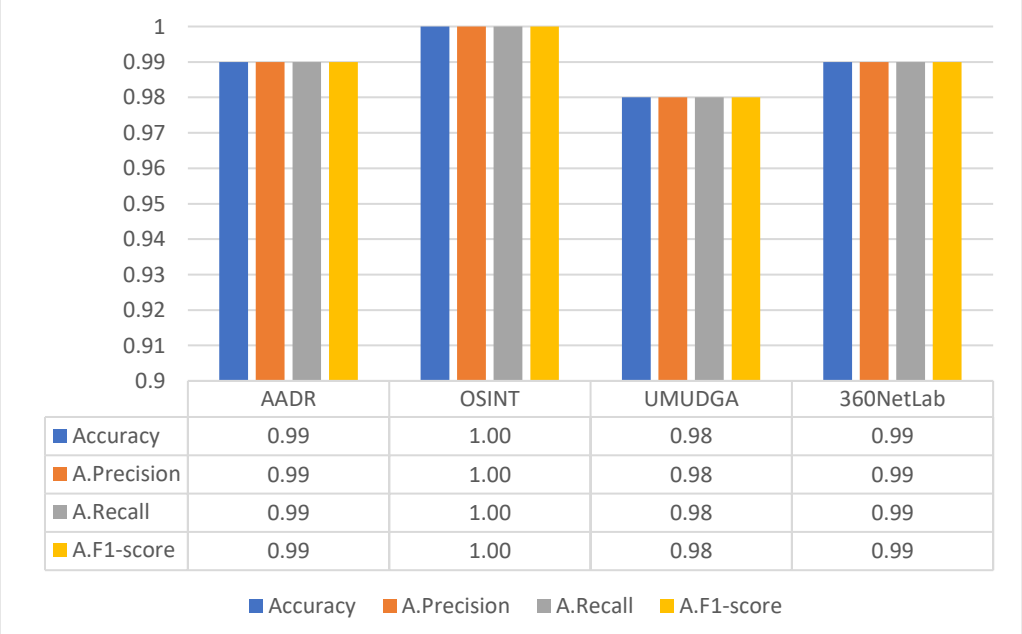
Bảng 3.4. Ký hiệu các đánh giá hai mô hình học sâu đề xuất

	LA_Bin07	**LA_Mul07**
AADR	Đánh giá B1	Đánh giá M1
OSINT	Đánh giá B2	
UMUDGA	Đánh giá B3	Đánh giá M2
360NetLab	Đánh giá B4	

Mô hình được huấn luyện và đánh giá trên công cụ Google Colab Pro, môi trường Linux, sử dụng GPU Tesla K80, RAM 24GB, thư viện hỗ trợ Keras, batch_size = 64.

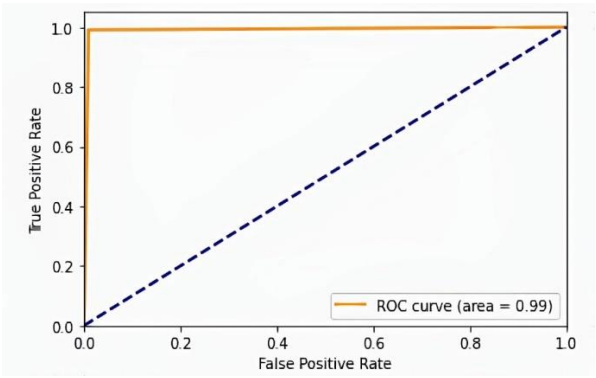
3.3.2. Đánh giá mô hình LA_Bin07 cho bài toán phát hiện DGA Botnet

Đối với bài toán phân lớp nhị phân, kết quả đánh giá mô hình LA_Bin07 qua các tham số Accuracy, A.Precision, A.Recall và A.F_1-Score được cho ở Hình 3.15:

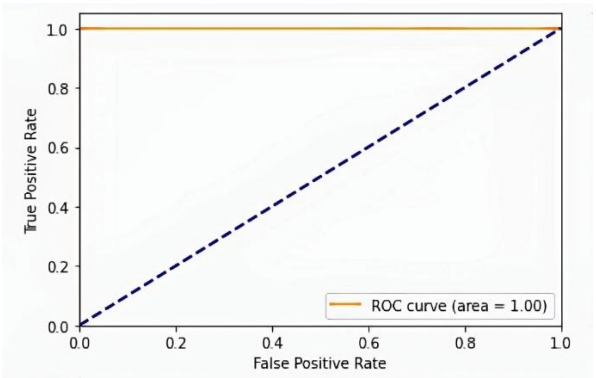


Hình 3.15. Kết quả các đánh giá B1, B2, B3 và B4

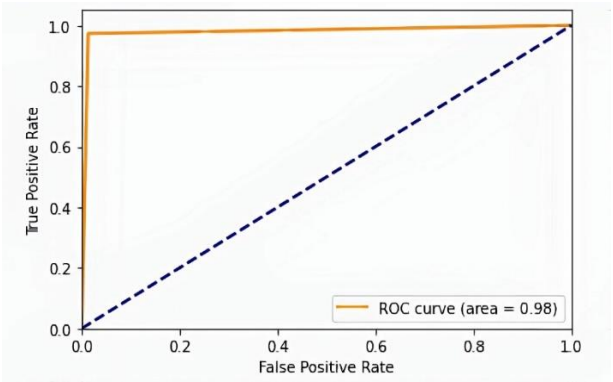
Biểu đồ ROC Curve và AUC tương ứng đối với 04 bộ dữ liệu AADR, OSINT, UMUDGA và 360NetLab được cho tại Hình 3.16:



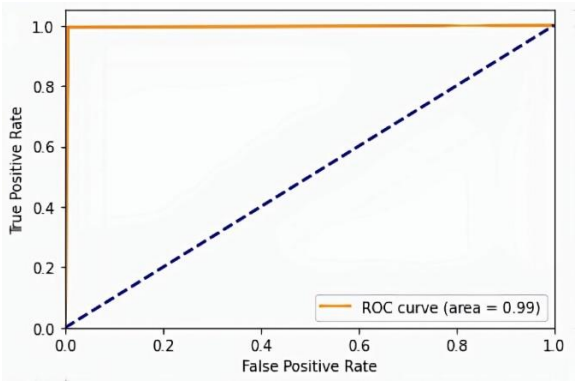
a) AADR



b) OSINT



c) UMUDGA



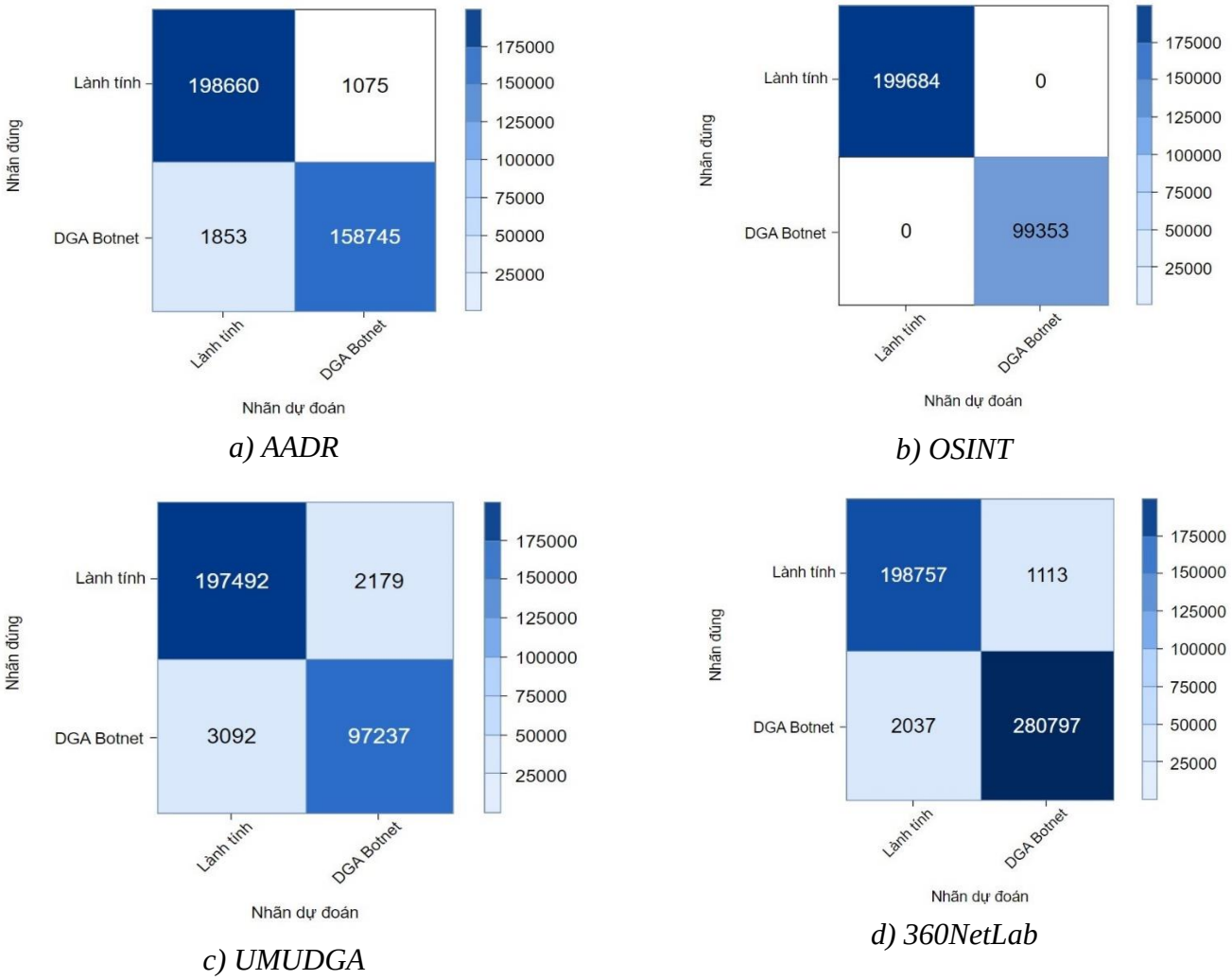
d) 360NetLab

Hình 3.16. ROC Curve và AUC của LA_Bin07 tương ứng với các đánh giá B1, B2, B3 và B4

Các kết quả trên cho thấy, mô hình LA_Bin07 có độ chính xác cao trong việc phân loại các tên miền lành tính và độc hại, với Accuracy đều đạt từ 0,98 trở lên trên cả 04 đánh giá. Điều này cũng thể hiện thông qua ROC Curve và AUC, với ROC Curve đạt trạng thái gần tối ưu và AUC cũng đạt từ 0,98 trở lên.

Trong đó, mô hình LA_Bin07 dự đoán chính xác với tỉ lệ là 1,00 trên bộ dữ liệu OSINT. Kết quả thử nghiệm trên bộ dữ liệu UMUDGA thấp hơn một chút ở mức 0,98 có thể được giải thích bởi số lượng mẫu DGA Botnet nhiều hơn và đa dạng hơn, trong đó có một số họ tên miền khá giống với tên miền lành tính về cả độ dài, không gian ký tự và từ khóa.

Chi tiết hơn về khả năng phát hiện của mô hình LA_Bin07 được thể hiện bởi ma trận nhầm lẫn và ma trận nhầm lẫn chuẩn hóa. Các ma trận này được thể hiện ở Hình 3.17, là kết quả thử nghiệm của mô hình LA_Bin07 lần lượt trên các bộ dữ liệu AADR, OSINT, UMUDGA và 360NetLab.



a) AADR *b) OSINT*

c) UMUDGA *d) 360NetLab*

Hình 3.17. Ma trận nhầm lẫn của mô hình LA_Bin07 tương ứng với các đánh giá B1, B2, B3 và B4

Mô hình LA_Bin07 có kết quả tốt nhất trên bộ dữ liệu OSINT, với số lượng mẫu chấp nhận nhầm và phát hiện nhầm là 0 (tỉ lệ gần 0,00). Ở hướng ngược lại, mô hình LA_Bin07 có số lượng mẫu chấp nhận nhầm và phát hiện nhầm lần lượt là 3.092 và 2.179 mẫu, tỉ lệ là 0,03 trên bộ dữ liệu UMUDGA. Điều này có thể giải thích bởi bộ UMUDGA có cập nhật thêm một số họ DGA Botnet mới với các đặc trưng giống tên miền lành tính đã phân tích ở trên. Một số mẫu đặc trưng dạng này như gozi_gpl, gozi_lutherm gozi_nase và gozi_rfc4343.

Trong vấn đề phòng chống DGA Botnet, việc phát hiện nhầm một tên miền lành tính thành độc hại có thể được xem là ít nguy hiểm hơn so với việc bỏ sót một tên miền độc hại được gán nhãn thành lành tính. Các ma trận nhầm lẫn ở Hình 3.17 cho thấy mô hình đề xuất có tỉ lệ bỏ sót các tên miền của DGA Botnet là rất thấp, đều dưới 0,03.

Hình 3.18 thể hiện thời gian huấn luyện và đánh giá của mô hình LA_Bin07 trên 04 bộ dữ liệu.

Hình 3.18. Thời gian huấn luyện và đánh giá của mô hình LA_Bin07 tương ứng với các đánh giá B1, B2, B3 và B4

Mô hình LA_Bin07 có thời gian luyện dao động từ 3.106 giây đến 6.883 giây cho mỗi đánh giá. Thời gian huấn luyện này tăng dần theo thứ tự OSINT, UMUDGA, AADR và 360NetLab với lần lượt là 3.106 giây, 4.424 giây, 5.083 giây và 6.883 giây. Nguyên nhân chính là do số lượng mẫu tên miền của các bộ dữ liệu trên cũng tăng theo thứ tự trên.

Trong cả bốn trường hợp, mô hình được huấn luyện với epoch = 10. Các ghi nhận thực tế cho thấy rằng mô hình bắt đầu hội tụ từ epoch = 6 trở đi. Hơn nữa, việc tăng số lượng epoch huấn luyện có thể cải thiện không đáng kể độ chính xác, nhưng lại tốn kém nhiều hơn về mặt thời gian.

Cuối cùng, có thể thấy rằng thời gian để huấn luyện, đánh giá bằng học sâu tận dụng được năng lực tính toán của GPU. Các mô hình học máy có thời gian huấn luyện nhanh hơn như LR, SVM lại không đạt được độ chính xác tương đương. Cần lưu ý rằng, việc huấn luyện mô hình trên CPU cho tốc độ chậm hơn rất nhiều so với huấn luyện trên GPU.

3.3.3. Đánh giá mô hình LA_Mul07 cho bài toán phân loại DGA Botnet

Mô hình LA_Mul07 áp dụng cho bài toán phân loại DGA Botnet được đánh giá lần lượt trên hai bộ dữ liệu là AADR và UMUDGA, bởi các bộ dữ liệu này đã được gán nhãn các họ DGA Botnet.

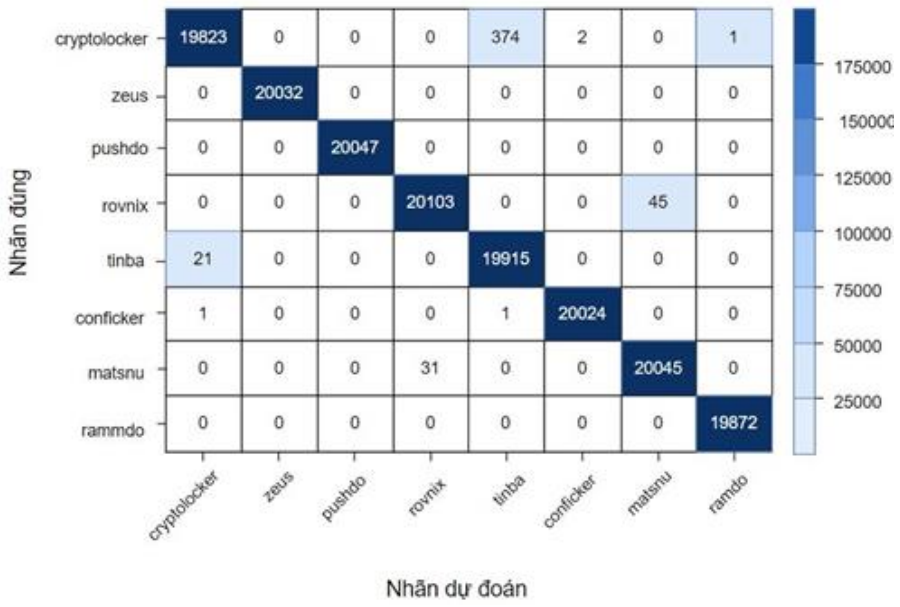
3.3.3.1. Đánh giá trên bộ dữ liệu AADR

Kết quả đánh giá mô hình LA_Mul07 trên bộ AADR bao gồm các tham số đánh giá, ma trận nhầm lẫn và ROC Curve lần lượt được thể hiện tại Bảng 3.5, Hình 3.19 và Hình 3.20 như sau:

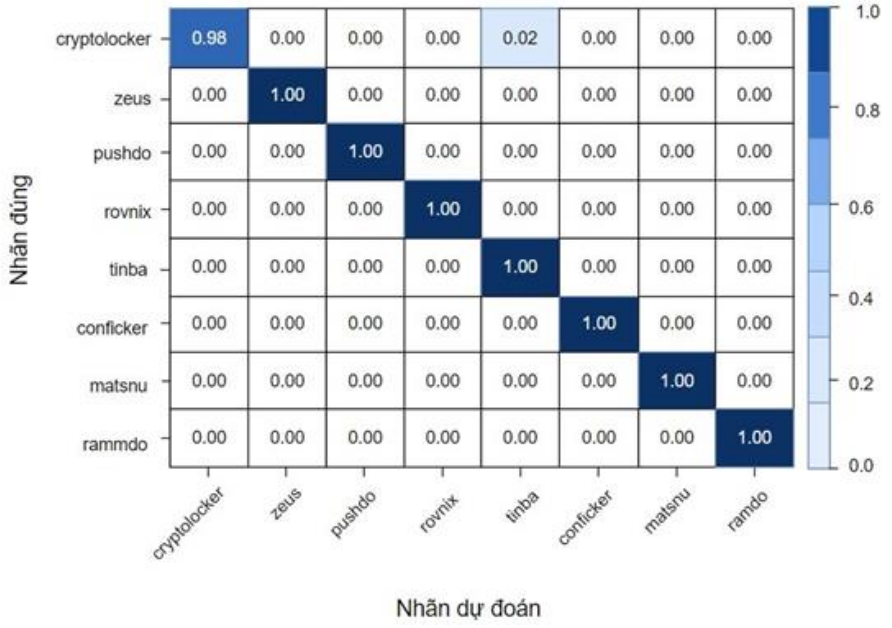
Bảng 3.5. Kết quả đánh giá M1 của mô hình LA_Mul07 trên bộ dữ liệu AADR

STT	**DGA Botnet**	**Precision**	**Recall**	**F_1-Score**
1	cryptolocker	1,00	0,98	0,99
2	zeus	1,00	1,00	1,00
3	pushdo	1,00	1,00	1,00
4	rovnix	1,00	1,00	1,00
5	tinba	0,98	1,00	0,99
6	conficker	1,00	1,00	1,00
7	matsnu	1,00	1,00	1,00
8	ramdo	1,00	1,00	1,00
Accuracy		***1,00***		

Mô hình LA_Mul07 có độ chính xác phân loại rất cao trên bộ dữ liệu AADR, với Accuracy đạt 1,00. Trong 08 họ DGA Botnet được xem xét tới thì có 06 họ DGA Botnet bao gồm: cryptolocker, zeus, pushdo, rovnix, conficker, matsnu và ramdo được phát hiện gần như chính xác hoàn toàn với Precision, Recall, F_1-Score đều đạt tới 1,00.



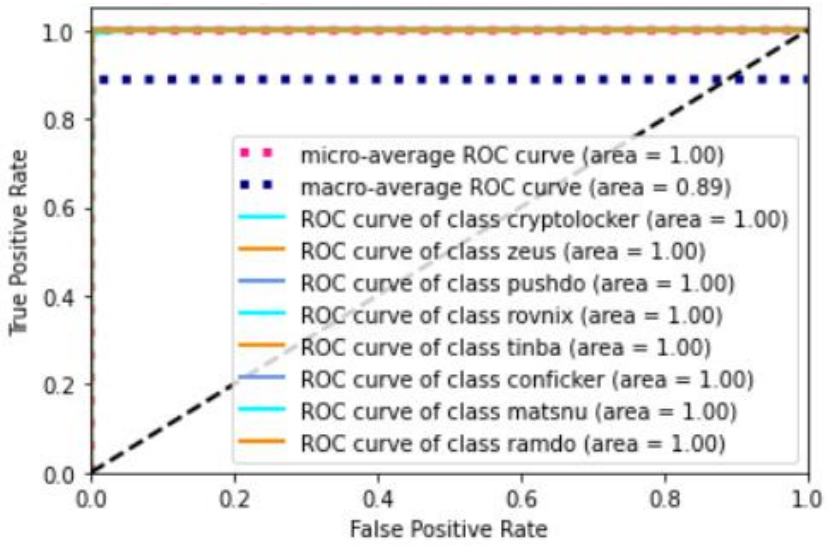
a) Ma trận nhầm lẫn



b) Ma trận nhầm lẫn chuẩn hóa

Hình 3.19. Ma trận nhầm lẫn và ma trận nhầm lẫn chuẩn hóa trong đánh giá M1

Ma trận nhầm lẫn ở Hình 3.19 cho thấy các tên miền lành tính và độc hại được phân loại một cách gần như chính xác. Trong ma trận nhầm lẫn chuẩn hóa, điều này thể hiện bằng giá trị 1,00 ở các ô nằm trên đường chéo chính của ma trận. Riêng họ cryptolocker có tỉ lệ 0,02 các mẫu bị phân loại sai thành họ tinba.



Hình 3.20. Biểu diễn ROC Curve và AUC trong đánh giá M1

Hình 3.20 cho thấy rõ hơn hiệu quả của mô hình LA_Mul07, với các đường ROC Curve và Area, giá trị đạt cao.

3.3.3.2. Đánh giá trên bộ dữ liệu UMUDGA

Kết quả đánh giá mô hình LA_Mul07 trên bộ dữ liệu UMUDGA gồm các thông số và ma trận nhầm lẫn lần lượt được cho tại Bảng 3.6 và Hình 3.21.

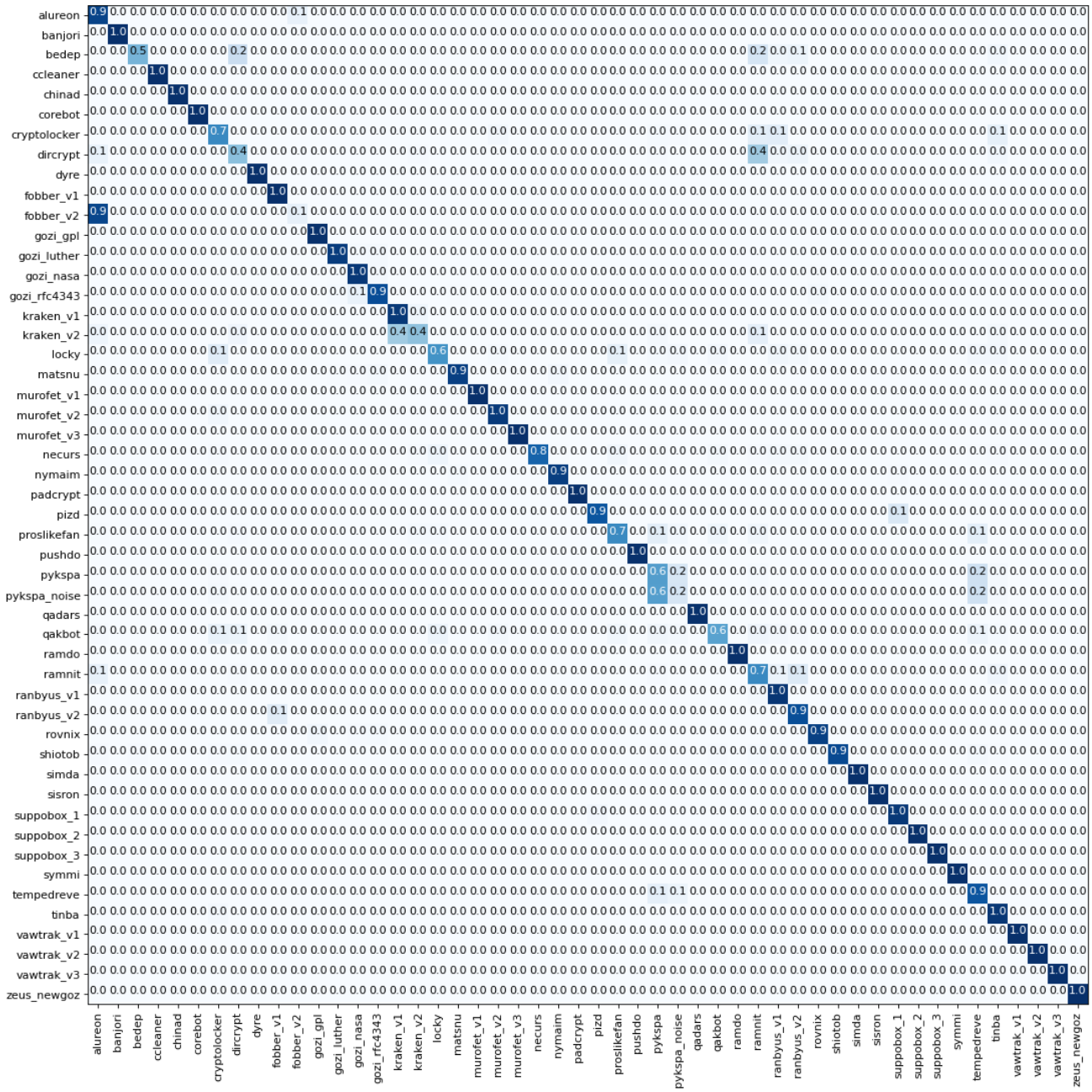
Bảng 3.6. Kết quả đánh giá M2 của mô hình LA_Mul07 trên bộ dữ liệu UMUDGA

STT	DGA Botnet	Pre	Re	F_1	STT	DGA Botnet	Pre	Re	F_1
1	alureon	0,45	0,92	0,60	26	pizd	0,97	0,86	0,91
2	banjori	0,99	1,00	1,00	27	proslikefan	0,82	0,65	0,73
3	bedep	0,96	0,47	0,63	28	pushdo	0,99	0,99	0,99
4	ccleaner	1,00	1,00	1,00	29	pykspa	0,39	0,57	0,47
5	china	1,00	0,99	1,00	30	pykspa_noise	0,35	0,16	0,22
6	corebot	1,00	1,00	1,00	31	qadars	0,99	0,99	0,99
7	cryptoloker	0,70	0,66	0,68	32	qakbot	0,84	0,55	0,67
8	dircrypt	0,52	0,42	0,47	33	ramdo	1,00	1,00	1,00
9	dyre	1,00	1,00	1,00	34	ramnit	0,44	0,66	0,52
10	fobber_v1	0,88	1,00	0,93	35	ranbyus_v1	0,76	0,98	0,86
11	fobber_v2	0,48	0,08	0,14	36	ranbyus_v2	0,76	0,88	0,82
12	gozi_gpl	0,96	0,99	0,98	37	rovnix	0,97	0,94	0,95
13	gozi_luther	0,97	0,95	0,96	38	shiotob	1,00	0,90	0,95
14	gozi_nase	0,89	0,97	0,93	39	simda	1,00	1,00	1,00

15	gozi_rfc4343	0,91	0,98	0,90	40	aaron	1,00	1,00	1,00
16	kraken_v1	0,72	0,96	0,83	41	suppobox_1	0,87	0,97	0,92
17	kraken_v2	0,82	0,41	0,55	42	suppobox_2	0,98	1,00	0,99
18	locky	0,84	0,62	0,71	43	suppobox_3	0,99	1,00	1,00
19	matsnu	0,98	0,94	0,96	44	symmi	1,00	1,00	1,00
20	murofet_v1	0,99	1,00	1,00	45	tempedreve	0,58	0,86	0,69
21	murofet_v2	0,94	0,96	0,95	46	tinba	0,77	0,97	0,86
22	murofet_v3	1,00	1,00	1,00	47	vawtrak_v1	1,00	1,00	1,00
23	necurs	0,99	0,80	0,89	48	vawtrak_v2	0,99	1,00	1,00
24	maim	0,95	0,94	0,95	49	vawtrak_v3	1,00	1,00	1,00
25	padcrypt	1,00	1,00	1,00	50	zeus_newgoz	1,00	1,00	1,00
Accuracy		***0,86***							

Bảng 3.6 cho thấy, mô hình LA_Mul07 có độ chính xác đạt cao trong phân loại các họ DGA Botnet, kể cả trong trường hợp số lượng họ DGA Botnet cần phân loại nhiều như ở bộ dữ liệu UMUDGA, với Accuracy đạt 0,86. Hầu hết các họ DGA Botnet đều được phân loại có độ chính xác cao từ 0,90 trở lên, trừ một số họ có tỉ lệ khá thấp như alureon, pikspa, pikspa_noise với F_1-score lần lượt là 0,60, 0,47, 0,22. Nhìn chung, với nhiệm vụ phân loại cùng lớp 50 lớp của bộ dữ liệu UMUDGA thì mô hình LA_Mul07 cho kết quả cao hơn so với các nghiên cứu trước đó.

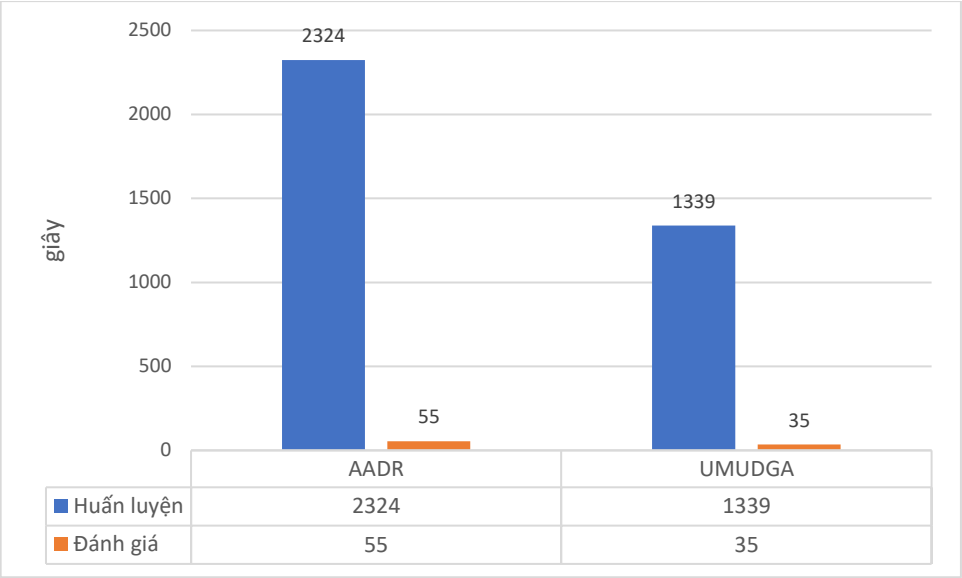
Hình 3.21 cho ta thấy ma trận nhầm lẫn của mô hình LA_Mul07 có đường chéo chính đậm nét, thể hiện rằng phần lớn các lớp được phân loại đúng. Mỗi ô có giá trị trong đoạn [0, 1] được chuẩn hóa bởi tỉ lệ so với số lượng mẫu. Các vấn đề còn tồn tại bao gồm: Họ fobber_v2 bị phân loại nhầm thành alureon với tỉ lệ 0,90; họ kraken_v2 bị phân loại nhầm thành kreken_v1 với tỉ lệ 0,40; Tỉ lệ gần 0,60 họ pykspa_noise bị phân loại nhầm thành pykspa. Điều này có thể giải thích bởi 2/3 họ này là các biến thể của nhau, có những đặc trưng khá giống nhau, nên có thể gây nhầm lẫn cho bộ phân loại. Đánh giá trên những họ DGA Botnet còn lại, bộ phân loại LA_Mul07 cho độ chính xác cao từ 0,90 trở lên.



Hình 3.21. Ma trận nhầm lẫn chuẩn hóa trong đánh giá M2 của mô hình LA_Mul07

3.3.3.3. Thời gian huấn luyện và đánh giá

Hình 3.24 thể hiện thời gian huấn luyện và đánh giá của mô hình LA_Mul07 đối với bài toán phân loại DGA Botnet trên bộ dữ liệu AADR và UMUDGA.



Hình 3.22. Thời gian huấn luyện và đánh giá của mô hình LA_Mul07 trong hai đánh giá M1 và M2

Kết quả cho thấy, mô hình LA_Mul07 có thời gian huấn luyện nhanh, với khoảng 2.324 giây trên bộ dữ liệu AADR và 1.339 giây trên bộ dữ liệu UMUDGA. Thời gian đánh giá lần lượt là 55 giây và 35 giây theo thứ tự trên. Nhận xét rằng, thời gian đánh giá chiếm từ 2.3% đến 2.6% so với thời gian huấn luyện mô hình.

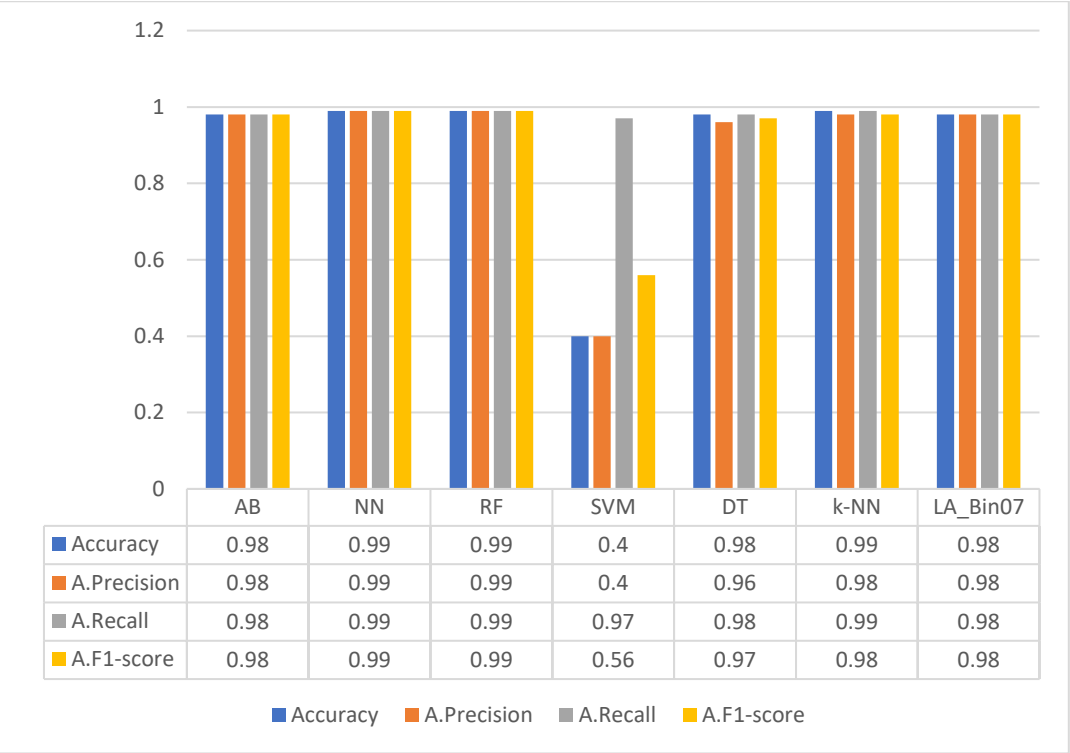
3.4. Đánh giá với các nghiên cứu liên quan

3.4.1. Đánh giá hai mô hình đề xuất trên bộ dữ liệu UMUDGA

Trong công bố về bộ dữ liệu UMUDGA, Zago và cộng sự [44] sử dụng các mô hình học máy bao gồm AB, NN, RF, SVM, DT, k-NN để áp dụng phát hiện và phân loại DGA Botnet. NCS tiến hành so sánh, đánh giá mô hình LA_Bin07 và LA_Mul07 với các mô hình đã đề cập trong nghiên cứu ở trên. Môi trường thử nghiệm và số lượng mẫu cho mỗi đánh giá được thiết lập là 10.000 mẫu cho mỗi họ DGA Botnet.

3.4.1.1. Đối với mô hình LA_Bin07

Đối với bài toán phân lớp nhị phân, kết quả so sánh giữa LA_Bin07 và các kết quả của Zago được thể hiện tại Hình 3.23.



Hình 3.23. So sánh mô hình LA_Bin07 với các thuật toán học máy của Zago trên bộ dữ liệu UMUDGA

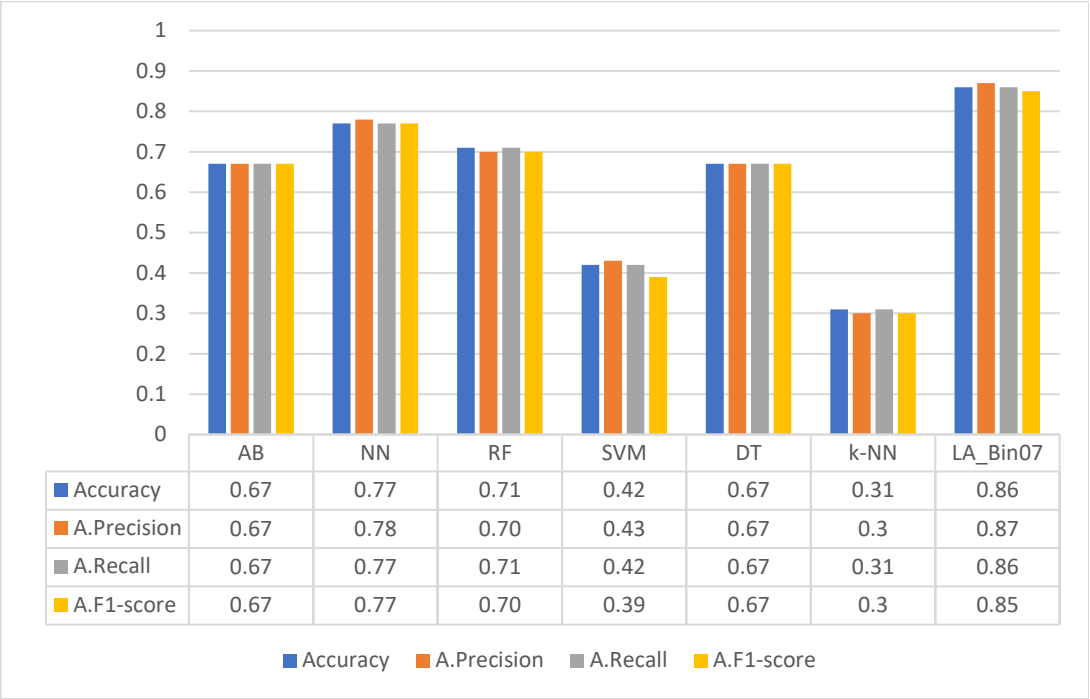
Hình 3.23 cho thấy, mô hình LA_Bin07 có độ chính xác tốt hơn rất nhiều so với mô hình SVM. Đồng thời, cho kết quả gần như tương đương với các mô hình AB, DT và thấp hơn 0,01 so với NN, RF và k-NN, với Accuray đạt từ 0,98 đến 0,99.

Ta cũng thấy rằng, SVM có độ chính xác thấp hơn hẳn so với các thuật toán được đối sánh. Điều này được giải thích bởi việc thuật toán SVM nhạy cảm với nhiễu, trong khi các họ DGA Botnet khác nhau tuy có đặc điểm khác nhau nhưng được đặt chung nhãn độc hại nên ảnh hưởng đến khả năng phân loại của SVM. Ngoài ra, việc trích xuất nhiều thuộc tính từ tên miễn cũng khiến cho việc tìm kiếm một mặt siêu phẳng phân tách hai lớp của SVM trở nên kém khả thi hơn.

Thuật toán k-NN tỏ ra hiệu quả trong trường hợp này nhưng lại hạn chế của k-NN là rất tốn kém thời gian đánh giá. Việc chênh lệch độ chính xác giữa NN, k-NN và LA_Bin07 trong thực tế là không đến 1% do làm tròn số, cách xử lý dữ liệu đầu vào khác nhau.

3.4.1.2. Đối với mô hình LA_Mul07

Đối với nhiệm vụ phân loại DGA Botnet, kết quả so sánh mô hình LA_Mul07 với các kết quả của Zago được tổng hợp và thể hiện tại Hình 3.24.



	AB	NN	RF	SVM	DT	k-NN	LA_Bin07
Accuracy	0.67	0.77	0.71	0.42	0.67	0.31	0.86
A.Precision	0.67	0.78	0.70	0.43	0.67	0.3	0.87
A.Recall	0.67	0.77	0.71	0.42	0.67	0.31	0.86
A.F1-score	0.67	0.77	0.70	0.39	0.67	0.3	0.85

Hình 3.24. So sánh mô hình LA_Mul07 với các thuật toán học máy của Zago trên bộ dữ liệu UMUDGA

Mô hình LA_Mul07 cho Accuracy cao hơn rõ rệt so với các mô hình học máy còn lại. Với Accuracy, A.Precision, A.Recall và A.F_1-score lần lượt đạt 0,86, 0,87, 0,86 và 0,85. Trong khi đó, mô hình tốt thứ hai là NN chỉ đạt lần lượt là 0,77, 0,78, 0,77 và 0,77. Ở chiều ngược lại, mặc dù đạt độ chính xác rất cao trong bài toán phân lớp nhị phân, mô hình k-NN lại tỏ ra rất thiếu chính xác trong bài toán phân lớp đa lớp khi chỉ đạt lần lượt là 0,31, 0,30, 0,31 và 0,30. Có thể kết luận rằng, mô hình LA_Mul07 cho kết quả phân lớp đa lớp chính xác và toàn diện hơn nhiều so với các mô hình còn lại.

3.4.2. Đánh giá hai mô hình với một số kiến trúc học sâu khác

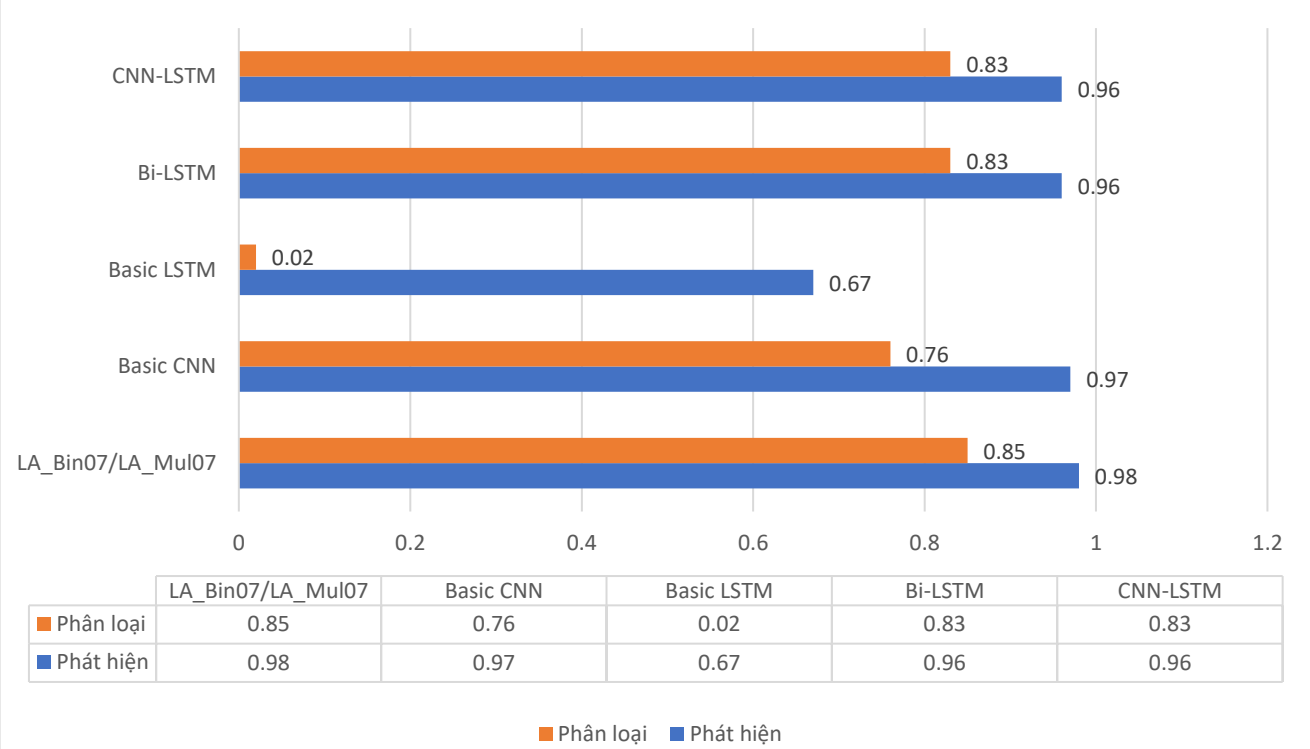
NCS cũng đồng thời đánh giá với một số kiến trúc học sâu khác mà NCS xây dựng trên cơ sở CNN và LSTM bao gồm: Basic CNN, Basic LSTM, Bi-LSTM và CNN-LSTM, được mô tả chi tiết tại Bảng 3.7:

Bảng 3.7. Một số kiến trúc học sâu khác cho bài toán DGA Botnet

STT	**Basic CNN**	**Basic LSTM**	**Bi-LSTM**	**CNN LSTM**
1	Embedding	Embedding	Embedding	Embedding
2	Dropout(0,2)	LSTM(128)	Bidirectional (LSTM(128))	Dropout(0,25)
3	Conv1D	Dropout(0,5)	Dropout(0,5)	Convo1D

4	MaxPooling1D	Dense(class_num)	Dense(class_num)	MaxPooling1D
5	Dense(250)	Activation (sigmoid)	Activation (sigmoid)	LSTM(128)
6	Dropout(0,2)			Dense(class_num)
7	Activation(relu)			Activation (sigmoid)
8	Dense(class_num)			
9	Activation (sigmoid)			

Kết quả Accuracy của các mô hình trên khi so sánh với LA_Bin07 (phát hiện) và LA_Mul07 (phân loại) trên bộ dữ liệu UMUDGA, được thể hiện tại Hình 3.25:



Hình 3.25. Thử nghiệm mô hình LA_Bin07 và LA_Mul07 với một số kiến trúc học sâu dựa trên CNN và LSTM trên bộ dữ liệu UMUDGA

Kết quả trên cho thấy, mô hình LA_Bin07 và LA_Mul07 đạt được độ Accuracy cao nhất trong lần lượt hai bài toán phát hiện và phân loại. Mô hình Basic CNN và Basic LSTM cho kết quả lần lượt là 0,76 và 0,02, thể hiện rằng các mô hình truyền thống này không đạt hiệu quả trong phân loại. Ngược lại, việc cải tiến các mô hình truyền thống này thể hiện qua Bi-LSTM hoặc CNN-LSTM cải thiện độ chính xác hơn rất nhiều và gần đạt được độ chính xác tốt nhất, với lần lượt là 0,86 cho phân lớp nhị phân và 0,83 cho phân lớp đa lớp, nhưng vẫn thấp hơn so với mô hình LA_Bin07 và

LA_Mul07. Cuối cùng, việc áp dụng kiến trúc lõi BiLSTM_SelfA_Double như mô hình LA_Bin07 và LA_Mul07 đề xuất cho kết quả tốt nhất.

3.4.3. Đánh giá mô hình phân loại LA_Mul07 với một số mô hình liên quan

Trong phần này, NCS sử dụng mô hình LA_Mul07 để đánh giá với mô hình của Qiao và Namgung với cùng hướng tiếp cận LSTM kết hợp Attention. Các đánh giá được thực hiện trên cùng bộ dữ liệu mà các tác giả mô tả và công bố.

3.4.3.1. Đánh giá với kết quả của Qiao và cộng sự

Bộ dữ liệu đánh giá của Qiao và cộng sự [17] được mô tả tại Bảng 3.8 như sau:

Bảng 3.8. Mô tả bộ dữ liệu đánh giá của Qiao và cộng sự với 16 nhãn

STT	Nhãn	Số mẫu
1	banjori	439.223
2	Post	66.000
3	tinba	65.603
4	ramnit	47.510
5	nears	32.768
6	qakbot	20.000
7	murofet	14.260
8	pykspa	14.215
9	ranbyus	13.960
10	simda	13.681
11	shiotob *(urlzone/bebloh)*	12.521
12	dyre	7.998
13	Cryptolocker	6.000
14	nymaim	6.000
15	locky	5.352
16	Alex	910.313

NCS chuẩn bị dữ liệu bằng cách kế thừa và bổ sung theo mô tả của tác giả, cụ thể như sau:

- Có 08 họ đầy đủ 10.000 mẫu bao gồm: murofet, necurs, Post, qakbot, ramnit, ranbyus, shiotob, tinba lấy từ nguồn tác giả sử dụng.

- Có 03 họ chưa đủ 10.000 mẫu được NCS bổ sung mẫu từ UMUDGA gồm: cryptolocker, dyre, locky.

- Có 04 họ không có từ [24] và được NCS lấy từ UMUDGA và UTL_DGA22 gồm: banjori, nymain, pykspa (prenecurs), simda.

- Có 01 họ Alex là các tên miền lành tính của Alexa.

Dữ liệu được chia 90% cho huấn luyện và 10% cho đánh giá, kết quả được thể hiện tại Bảng 3.9:

Bảng 3.9. Đánh giá kết quả của LA_Mul07 với mô hình của Qiao và cộng sự trên bộ dữ liệu của Qiao

	LSTM_AM (Qiao và cộng sự)			**LA_Mul07 (Tuấn và cộng sự)**		
	Pre	**Re**	**F_1**	**Pre**	**Re**	**F_1**
nymaim	0,40	0,11	0,17	0,74	0,79	0,77
ranbyus	0,47	0,85	0,60	0,87	0,91	0,89
murofet	0,76	0,72	0,74	0,86	0,84	0,85
pykspa	0,90	0,72	0,80	0,97	1,00	0,98
locky	0,00	0,00	0,00	0,91	0,45	0,60
shiotob	0,98	0,93	0,95	0,99	0,94	0,96
banjori	1,00	1,00	1,00	1,00	1,00	1,00
necurs	0,67	0,17	0,27	0,95	0,87	0,91
Cryptolocker	0,10	0,00	0,00	0,65	0,56	0,60
simda	0,93	0,97	0,95	0,99	0,94	0,96
dyre	1,00	1,00	1,00	1,00	1,00	1,00
Post	0,99	1,00	1,00	1,00	1,00	1,00
tinba	0,93	0,99	0,96	0,95	1,00	0,97
qakbot	0,79	0,50	0,61	0,81	0,75	0,78
ramnit	0,47	0,75	0,58	0,85	0,94	0,89
Alex	0,99	1,00	0,99	1,00	1,00	1,00
Avg	**0,95**	**0,95**	**0,95**	**0,98**	**0,98**	**0,98**

Bảng trên cho thấy, mô hình LA_Mul07 có A.F_1-score được cải thiện 3% so với mô hình LSTM_AM của Qiao và cộng sự. Đồng thời, có ưu điểm là có khả năng nhận diện đúng các họ DGA Botnet (thông qua chỉ số Precision, Recall của từng nhãn) đồng đều hơn so với của Qiao. Một số họ DGA Botnet trong mô hình của Qiao gần như không phát hiện được như locky hay cryptolocker đều cùng 0,00.

3.4.3.2. Đánh giá với kết quả của Namgung và cộng sự

Bộ dữ liệu đánh giá của Namgung và cộng sự được mô tả tại Bảng 3.10. Nhóm tác giả chia sẻ bộ dữ liệu này trên GitHub [60] và được NCS sử dụng cho việc so sánh, đánh giá của mình.

Bảng 3.10 mô tả 21 nhãn trong bộ dữ liệu đánh giá của Namgung và cộng sự. Dữ liệu được NCS chia 90% cho huấn luyện và 10% cho đánh giá.

Bảng 3.10. Mô tả bộ dữ liệu đánh giá của Namgung và cộng sự với 21 nhãn

STT	Nhãn	Số mẫu
1	Non-DGA	603.387
2	Banjori	439.223
3	Tmba	66.789
4	Post	66.000
5	Ramnit	64.605
6	Qakbot	40.000
7	Necurs	32.768
8	Murofet	28.520
9	Shiotob *(Urlzone/Bebloh)*	15.021
10	Simda	14.755
11	Ranbyus	13.200
12	Pykspa	10.019
13	Dyre	7.998
14	Kraken	7.878
15	Cryptolocker	6.000
16	Nymaim	6.000
17	Locky	4.014
18	Vawtrak	3.150
19	Shifu	2.331
20	Ramdo	2.000
21	P2P	2.000

Kết quả đánh giá được thể hiện ở Bảng 3.11 như sau:

Bảng 3.11. Đánh giá kết quả của LA_Mul07 với mô hình của Namgung và cộng sựu trên bộ dữ liệu của Namgung

	BiLSTM_Attention			**CNN-BiLSTM_Ensemble**			**LA_Mul07**		
	Pre	**Re**	**F_1**	**Pre**	**Re**	**F_1**	**Pre**	**Re**	**F_1**
Non-DGA	0,99	0,99	0,99	0,99	1,00	0,99	0,99	0,99	0,99
Banjori	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Tinba	0,93	0,99	0,96	0,93	1,00	0,96	0,95	0,98	0,97
Post	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Ramnit	0,83	0,90	0,86	0,85	0,90	0,87	0,84	0,93	0,88
Qakbot	0,77	0,78	0,77	0,78	0,80	0,79	0,82	0,80	0,81
Necurs	0,95	0,80	0,86	0,95	0,82	0,88	0,97	0,81	0,88
Murofet	0,82	0,83	0,83	0,85	0,82	0,83	0,83	0,89	0,85
Shiotob	0,99	0,91	0,94	0,99	0,91	0,95	0,99	0,91	0,95

Simda	0,96	0,99	0,97	0,97	0,99	0,98	0,97	1,00	0,99
Ranbyus	0,87	0,83	0,85	0,85	0,85	0,85	0,85	0,84	0,85
Pykspa	0,93	0,99	0,96	0,98	0,99	0,98	0,98	1,00	0,99
Dyre	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Kraken	0,93	0,86	0,89	0,94	0,85	0,90	0,90	0,88	0,89
Cryptolocker	0,52	0,45	0,48	0,63	0,32	0,43	0,72	0,35	0,47
Nymaim	0,54	0,28	0,37	0,68	0,25	0,37	0,66	0,72	0,69
Locky	0,87	0,32	0,47	0,98	0,30	0,46	0,66	0,45	0,54
Vawtrak	0,71	0,27	0,39	0,95	0,75	0,84	0,89	0,93	0,91
Shifu	0,86	0,98	0,91	0,88	0,97	0,92	0,85	0,98	0,91
Ramdo	0,99	1,00	0,99	0,99	1,00	0,99	0,99	1,00	1,00
P2P	0,40	0,37	0,38	0,46	0,73	0,56	0,96	0,92	0,94
Avg	***0,96***	***0,96***	***0,96***	***0,97***	***0,97***	***0,97***	***0,97***	***0,97***	***0,97***
Accuracy	**0,9620**			**0,9684**			**0,9722**		

Kết quả trên cho thấy, mô hình LA_Mul07 đã cải thiện độ chính xác Accuracy là 1,03% so với mô hình BiLSTM_Attention và 0,38% so với mô hình CNN-BiLSTM_Ensemble.

Một ưu điểm khác của LA_Mul07 là khả năng phân loại các họ DGA Botnet đồng đều hơn, với các họ DGA Botnet được phát hiện với Precision, Recall hầu hết đạt từ cao từ 0.85 trở lên và không có mẫu nào quá thấp. Ở chiều ngược lại, mô hình BiLSTM_Attention của Namgung có khả năng phát hiện không đồng đều và hạn chế ở một số họ DGA Botnet như Vawtrak, Nymaim hay P2P.

3.5. Kết luận Chương 3

Trong Chương 3, NCS đề xuất mô hình lõi BiLSTM_SelfA_Double và kết thừa để xây dựng hai mô hình học sâu mới là LA_Bin07 và LA_Mul07 lần lượt giải quyết bài toán phát hiện và phân loại DGA Botnet.

Các đánh giá cho thấy, mô hình LA_Bin07 có độ chính xác rất cao, đạt từ 0,98 trên bộ UMUDGA cho đến 1,00 trên bộ dữ liệu OSINT. Mô hình LA_Mul07 cho khả năng phân loại các họ DGA Botnet cao với Accuracy lần lượt đạt 1,00 và 0,86 trên hai bộ dữ liệu là AADR và UMUDGA, đồng thời nâng cao độ chính xác so với các mô hình liên quan.

Việc giải quyết bài toán DGA Botnet dựa trên học sâu mang lại nhiều ý nghĩa trong vấn đề đảm bảo an ninh mạng, đặc biệt là bài toán phân loại DGA Botnet. Thứ nhất, hướng tiếp cận này có thể nhanh chóng đưa ra các cảnh báo về DGA Botnet với độ chính xác cao. Thứ hai, giải pháp này đòi hỏi ít tài nguyên tính toán hơn so với

các giải pháp phân tích gói tin truyền thống và tận dụng được năng lực tính toán của GPU. Thứ ba, giải pháp có thể mở rộng áp dụng cho các loại mã độc, phần mềm độc hại, phần mềm gián điệp có cơ chế truy vấn tên miền tương tự. Cuối cùng, module phát hiện và phân loại tên miền độc hại hoàn toàn có thể được tích hợp trên các giải pháp bảo mật hiện đại ngày nay.

Một phần kết quả trình bày tại Chương 3 được công bố tại [CT4] trong Danh mục các công trình công bố liên quan đến luận án.

Chương 4. QUY TRÌNH XÂY DỰNG VÀ BỘ DỮ LIỆU MỚI UTL_DGA22 CHO BÀI TOÁN DGA BOTNET

Trong chương 4, NCS đề xuất bổ sung hoàn thiện một quy trình xây dựng bộ dữ liệu mới và áp dụng xây dựng bộ dữ liệu mới là UTL_DGA22. Bộ dữ liệu này kế thừa kết quả của những nghiên cứu trước đó, đồng thời bổ sung thêm các cải tiến, cập nhật mới đáp ứng các tiêu chí đặt ra. NCS mong muốn đóng góp bộ dữ liệu UTL_DGA22 để phục vụ cho các nghiên cứu cùng lĩnh vực trong thời gian tới.

4.1. Đặt vấn đề bộ dữ liệu DGA Botnet

4.1.1. Khái quát vấn đề

Một số nghiên cứu được công bố trước đó đề xuất các thuật toán hoặc mô hình mới để phát hiện DGA Botnet. Điểm tương đồng giữa các nghiên cứu này đó là việc chúng đều giải quyết ít nhất một trong hai bài toán phát hiện và phân loại DGA Botnet. Một số thuật toán có độ chính xác rất cao. Tuy nhiên, việc thực hiện các đối sánh giữa những nghiên cứu đó là chưa thuận lợi, bởi các đánh giá được chạy trên các bộ dữ liệu cơ bản khác nhau, phần lớn là được tổng hợp bởi chính nhóm nghiên cứu hoặc thu thập trực tuyến trên Internet. Việc công bố và mô tả chi tiết bộ dữ liệu, số lượng mẫu trong từng bộ dữ liệu cũng còn hạn chế.

Bảng 4.1 mô tả chi tiết về đặc điểm của một số bộ dữ liệu được sử dụng cho đánh giá bởi các nghiên cứu trước đó về DGA Botnet.

Bảng 4.1. Một số nghiên cứu và bộ dữ liệu để đánh giá trong bài toán DGA Botnet

STT	Tác giả/Nhóm nghiên cứu	Năm công bố	Thuật toán đề xuất	Mô tả bộ dữ liệu
1	Antonakakis và cộng sự [61]	2012	Cây quyết định, mô hình Makov ẩn	Bộ dữ liệu được thu thập trong 15 tháng từ 11/2010 đến 01/2012
2	Zhou và cộng sự [62]	2013	DNS NXDomain Traffic	Bộ dữ liệu Alexa top 1.000.000 Domain và NXDomain Traffic Capture.
3	Bilge và cộng sự [63]	2014	EXPOSURE	Được thu thập sử dụng SIE DNS feeds theo thời gian thực trong 2,5 tháng, bao gồm 100 tỉ truy vấn DNS

4	Nguyen và cộng sự [64]	2015	Lọc cộng tác và phân cụm	Được thu thập từ những bản ghi DNS từ 18.000 người dùng từ 01/04/2015 đến 15/04/2015
5	Sharifnya và cộng sự [65]	2015	DFBotKiller	Dữ liệu thu thập từ lưu lượng DNS thực tế
6	Bottazzi và cộng sự [66]	2015	Fast Mining	Dữ liệu được thu thập từ mạng của một công ty trong tháng 06/2014, với gồm hơn 60.000 máy trạm và 100.000 người dùng ở Italia
7	Kwon và cộng sự [7]	2016	PsyBoG	Dữ liệu thu thập từ lưu lượng DNS thực tế, bao gồm cả lưu lượng của mã độc và lưu lượng từ máy chủ DNS
8	Erquiaga và cộng sự [67]	2016	Markov Models	Dữ liệu thu thập từ lưu lượng truy cập của máy tính cá nhân, mạng quy mô nhỏ và mạng trong khuôn viên trường đại học. Bộ dữ liệu được công khai là một phần của Malware Capture Facility Project [68]
9	Mạc và cộng sự [11]	2017	Học máy	Dữ liệu thu thập từ thực tế, bao gồm 168.900 mẫu DNS của 38 họ DGA Botnet.
10	Wang và cộng sự [8]	2017	DBoD	Dữ liệu DNS được thu thập từ một mạng dành cho giáo dục, trong thời gian 26 tháng
11	Bisio và cộng sự [9]	2017	Phân cụm	Đánh giá trên 2 bộ dữ liệu: - Dataset 01: Gồm 40 họ DGA Botnet Dataset 02: Thu thập từ mạng LAN của một công ty trong vòng 15 ngày
12	Trần và cộng sự [15]	2018	LSTM	Dữ liệu lấy từ Alexa top 1.000.000 sites và 37 họ DGA Botnet từ OSINT DGA feed.
13	Curtin và cộng sự [16]	2019	RNN và Side Information	Bộ dữ liệu gồm 41 họ DGA Botnet và những tên miền lành tính, bao gồm 2,3 triệu tên miền, trong đó có 1,01 tên miền lành

				tính và 1,28 triệu tên miền của DGA Botnet.
14	Ashiq và cộng sự [69]	2019	Feedforward NN	Dữ liệu được mô tả tại tài liệu [70]
15	Alieyan và cộng sự [71]	2019	DNS rule-based schema	Dữ liệu DNS được lọc từ bộ dữ liệu ISOT Dataset [48]
16	Căn và cộng sự [CT1]	2020	NCM	Sử dụng dữ liệu DNS từ Alexa Top 1.000.000 Domain, Bambenek Consulting feed và 360 NetLab
17	Yun và cộng sự [72]	2020	Khaos	Tên miền lành tính: LD1 và LD2 lần lượt được tổng hợp từ Reverse DNS và Alexa Top 1M Domains. Tên miền độc hại được thu thập từ Research DGAs [73][70] và dữ liệu tổng hợp thực tế
18	Qiao và cộng sự [17]	2020	LSTM_AM	Bộ dữ liệu gồm 16 nhãn bao gồm DGA Botnet và lành tính, được lấy từ 360NetLab và Alexa top 1M Domain.
19	Vinayakumar và cộng sự [20]	2020	Học sâu	Bộ dữ liệu DS1 và DS2 (AmritaDGA), là một phần trích ra từ DMD 2018 Dataset [74]
20	Zago [13] [44]	2020	Học máy	Sử dụng bộ dữ liệu UMUDGA Dataset được nhóm nghiên cứu xây dựng và công bố
21	Pei và cộng sự [75]	2020	Capsule Networks và Sliced RNN	Gồm: Alexa Top 1M Domains, OSINT, 360NetLab, và Andrey Abakumov’s repository.
22	Namgung và cộng sự [19]	2021	CNN-BiLSTM	Bộ dữ liệu gồm 21 nhãn bao gồm DGA Botnet và lành tính, được lấy từ OSINT DGA feed from Bambenek Consulting và Alexa top 1M Domain.

Nhận xét chung, các giải pháp đề xuất thường được đánh giá trên những bộ dữ liệu do nhóm nghiên cứu thu thập vào những thời điểm khác nhau, số lượng mẫu không đồng đều, tính công bố rộng rãi không cao và thường không thuận tiện cho việc đối sánh.

4.1.2. Bộ dữ liệu về Botnet nói chung

Trong phần này, NCS đề cập một cách khái quát đến các bộ dữ liệu về Botnet nói chung, để làm căn cứ phân biệt với các bộ dữ liệu về DGA Botnet.

- CTU-13 [76] là một bộ dữ liệu ghi lại lưu lượng mạng của Botnet. Bộ dữ liệu này bao gồm các lưu lượng mạng được thu thập tại CTU University, Czech Republic vào năm 2011. Nhóm tác giả đã xây dựng 13 kịch bản của các loại Botnet khác nhau. Mỗi kịch bản được ghi lại với đầy đủ lưu lượng mạng Botnet, các lưu lượng bình thường và các lưu lượng nền. Dung lượng dữ liệu của mỗi kịch bản trên từ 5,2 GB cho đến 123 GB. Một bộ dữ liệu nhỏ hơn được trích xuất chỉ gồm các lưu lượng của Botnet có dung lượng là 1,9 GB. Các trường thuộc tính trong mỗi mẫu dữ liệu này bao gồm: Địa chỉ IP, cổng của nguồn và đích, giao thức, dung lượng gói tin, các cờ của gói tin. Bộ dữ liệu đã gán nhãn lưu lượng của một số loại Botnet như Neris, Rbot, Virut, Menti, NSIS.ay, Sogou hay Murlo. Bộ dữ liệu CTU-13 tương thích với các giải pháp dựa trên việc phân tích gói tin.

- UGR16 là bộ dữ liệu về phát hiện xâm nhập mạng được tạo ra bởi Maciá-Fernández và cộng sự [77]. Bộ dữ liệu này cũng thu thập lại tất cả các lưu lượng trong một mạng thực tế. Họ xây dựng một mạng và đặt các cảm biến tại nút mạng để thu thập dữ liệu. Những dữ liệu này chứa trong đó các lưu lượng của một cuộc tấn công Low-rate DoS, Port Scanning hay Bot Traffic. Tác giả đã gán nhãn của các cuộc tấn công, bao gồm Signitue-based Labeling và Anomoly-based Labelling. Các tài liệu mô tả về bộ dữ liệu cũng được nhóm tác giả công bố đầy đủ. Cần lưu ý rằng, bộ dữ liệu UGR16 không chỉ tập trung vào việc phát hiện Botnet, mà bao gồm cả những kỹ thuật tấn công khác như DoS hay Port Scanning.

- DreLAB là bộ dữ liệu được công bố bởi Andrea Venturi và cộng sự [78]. Mục tiêu của bộ dữ liệu này là hướng tới việc đánh giá sự hiệu quả của các thuật toán trong Botnet nói chung. Họ sử dụng kỹ thuật học sâu tăng cường để tạo ra các mẫu dữ liệu. Họ cũng tạo các mẫu Botnet đối nghịch giúp tăng cường khả năng trốn tránh sự phát hiện. Toàn bộ dữ liệu mạng được thu thập dưới dạng tệp tin *pcap, sau đó chuyển thành *CSV để công bố. Bộ dữ liệu cung cấp những hiểu biết sâu hơn về hoạt động của Botnet nói chung và đánh giá khả năng phát hiện của các thuật toán học máy nói riêng.

UNSW-NB15 cũng là một bộ dữ liệu mạng về Botnet, được tạo ra bởi The IXIA PerfectStorm Tool tại the Cyber Range Lab of UNSW Canberra [79]. Họ sử dụng công cụ TCPDump đặt tại một nút mạng để thu thập lưu lượng của các truy cập mạng. Họ ghi lại được 100 GB lưu lượng truy cập dưới dạng pcap. Trong số này, ngoài các

lưu lượng mạng thông thường, còn bao gồm 09 dạng tấn công mạng khác nhau, gồm Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode và Worms. Tiếp theo, họ sử dụng công cụ Argus, Bro-IDS để trích xuất 49 thuộc tính và gán nhãn tương ứng. Các thuộc tính và nhãn được cung cấp dưới dạng tệp tin CSV. Nhìn chung, tương tự như UGR16, bộ dữ liệu UNSW-NB15 không phải là bộ dữ liệu chuyên tập trung vào Botnet mà còn cả những dạng tấn công khác.

Bộ dữ liệu ISCX-Bot-2014 được xây dựng bởi Viện An ninh mạng Canada, bao gồm 16 loại Botnet khác nhau, bao gồm: Neris, Rbot, Menti, Sogou, Murlo, Virut, NSIS, Zeus, SMTP Spam, UDP Storm, Tbot, Zero Access, Weasel, Smoke Bot, Zeus Control và ISCX IRC Bot [33]. Bộ dữ liệu được chia thành hai nhóm gồm TrainSet có khối lượng 5,3 GB và TestSet có dung lượng 8,5 GB. Chúng là sự tổng hợp của 03 bộ dữ liệu thành phần, bao gồm: ISOT Dataset [80], ISCX 2012 IDS Dataset [50] và Malware Capture Facility Project [81]. Mỗi bộ đều bao gồm các lưu lượng mạng độc hại và lưu lượng mạng bình thường. Đối với các thuật toán học máy, học sâu thì việc phân chia dữ liệu có ý nghĩa phần TrainSet được dùng cho quá trình huấn luyện, còn tập TestSet được dùng cho đánh giá. Tương tự, bộ dữ liệu ISCX-Bot-2014 phù hợp để đánh giá các thuật toán phát hiện Botnet nói chung.

Bảng 4.2 trình bày tóm tắt tính chất của các bộ dữ liệu về Botnet đã trình bày ở trên, cụ thể như sau:

Bảng 4.2. Đặc điểm của các bộ dữ liệu về chung về Botnet

STT	**Bộ dữ liệu**	**Ký hiệu**	**Lưu lượng mạng**	**Nhãn**	**Lưu lượng độc hại khác**	**Định dạng**	**Năm công bố**
1	CTU-13 [76]	CTU	Thực tế	Botnet	Không	PCAP & Flow	2014
2	UGR16 [77]	UGR	Thực tế	Low-rate DoS, Port Scanning, Botnet	Có	Flow & CSV	2018
3	DreLAB [78]	DLAB	Phòng thí nghiệm	Botnet	Không	CSV	2021
4	UNSW-NB15 [79]	UNSW	Phòng thí nghiệm	Botnet và Network Attack	Có	CSV	2015
5	ISCX-Bot-2014 [33]	ISCX	Phòng thí nghiệm	Botnet và mã độc	Có	PCAP & CSV	2014

Nhận xét rằng, các bộ dữ liệu trên đều được thu thập từ lưu lượng mạng bằng các công cụ như TCPDump hay Wireshark dưới dạng PCAP. Một số được trích xuất ra dạng CSV để giảm bớt dung lượng và trích chọn thuộc tính. Ngoài hai bộ dữ liệu CTU-13 và DreLab chỉ tập trung vào bài toán Botnet, thì ba bộ dữ liệu còn lại được mở rộng cho việc phát hiện mã độc và các dạng tấn công mạng khác. Cuối cùng, cả 05 bộ dữ liệu ở trên đều không được thiết kế để đánh giá cho bài toán DGA Botnet bởi chúng thiếu các mẫu tên miền DGA Botnet và nhãn tương ứng. Đây là cơ sở quan trọng để phát triển các bộ dữ liệu cho bài toán DGA Botnet.

4.1.3. Bộ dữ liệu về DGA Botnet

Một số nghiên cứu trước đó đã có đề cập đến các bộ dữ liệu cho đánh giá bài toán DGA Botnet, cụ thể như sau:

Trong nghiên cứu [82], Suryotrisongko và cộng sự chia sẻ bộ dữ liệu "Botnet DGA Dataset" đi kèm với các kết quả nghiên cứu của họ. Bộ dữ liệu này có dung lượng 205 MB, được lưu trữ dưới dạng tệp tin *csv, bao gồm 1.000.000 tên miền từ Alexa Top 1 Milion Domains [43] được gán nhãn lành tính và 10 họ DGA Botnet với 1.803.333 tên miền được gán nhãn độc hại. Mỗi mẫu dữ liệu có 07 trường thuộc tính đã được trích xuất từ tên miền gốc bao gồm: CharLength, TreeNewFeature, nGramReputation_Alexa, REAlexa, MinREBotnets, Entropy và InformationRadius. Thuộc tính thứ 08 là Class bao gồm hai giá trị là 0 và 1, tương ứng với nhãn của tên miền. Bộ dữ liệu Botnet DGA Dataset được công bố trên IEEE Dataport [83]. Hạn chế của bộ dữ liệu này là không chứa các tên miền gốc và số lượng mẫu của các họ DGA Botnet còn ít.

Andrey Abakumov công bố một bộ dữ liệu về DGA Botnet trên kho lưu trữ GibHub tại địa chỉ [35]. Bộ dữ liệu này bao gồm 1.000.000 tên miền lành tính từ Alexa, kết hợp với tên miền của 08 họ DGA Botnet bao gồm: cryptolocker, zeus, pushdo, rovnix, tinba, conficker, matsnu và ramdo. Các tên miền được cung cấp dưới dạng tệp tin *txt và chưa được rút trích đặc trưng. Nhãn 0 được gán cho tên miền lành tính và 08 họ DGA Botnet lần lượt được gán nhãn từ 1 đến 8. Bộ dữ liệu này cho phép thực hiện đánh giá cả bài toán phát hiện và phân loại. Bên cạnh đó, họ cũng cung cấp mã nguồn sinh tên miền tự động cho các họ DGA Botnet được đề cập.

Nhóm nghiên cứu của Zago và cộng sự từ University of Murcia đã xây dựng một bộ dữ liệu được xem là đầy đủ và chi tiết nhất về DGA Botnet tính đến thời điểm công bố, gọi là UMUDGA Dataset vào năm 2020 [44] [13]. Bộ dữ liệu này bao gồm 1.000.000 tên miền lành tính và 50 họ DGA Botnet. Trong đó, mỗi họ DGA Botnet có từ 10.000 đến 1.000.000 mẫu tên miền được lưu dưới dạng *ARFF, *CSV và

*TXT. Mã nguồn sinh tên miền cũng được nhóm tác giả kèm theo bộ dữ liệu này và chia sẻ công khai trên IEEE Dataport.

Ngoài việc cung cấp tên miền nguyên gốc, Zago và các cộng sự cũng đã đề xuất một số đặc trưng của tên miền, gồm các đặc trưng cơ bản, các đặc trưng n-gram và đặc trưng dựa trên xử lý ngôn ngữ tự nhiên. Kết quả thử nghiệm cho thấy các đặc trưng này là phù hợp để làm đầu vào các mô hình học máy. Bộ dữ liệu cho phép đánh giá với cả hai bài toán phát hiện và phân loại DGA Botnet.

DGArchive là một dịch vụ được cung cấp bởi Fraunhofer FKIE và được quản trị bởi Daniel Plohmann. Dịch vụ này tổng hợp và cung cấp dữ liệu về DGA Botnet với 86 họ DGA Botnet khác nhau [84]. Bộ dữ liệu DGArchive bao gồm các họ tên miền của DGA Botnet được thể hiện dưới dạng rõ. Mặc dù số lượng họ DGA Botnet được tổng hợp là rất nhiều, nhưng số lượng tên miền trong mỗi họ DGA Botnet là không đồng đều, có một số họ rất ít dữ liệu trong khi một số họ khác lại nhiều hơn rất đáng kể. Bộ dữ liệu này được cung cấp dưới dạng tệp tin *CSV và không được phép sử dụng cho các mục đích thương mại. Daniel Plohmann và các cộng sự cũng đã trình bày những đánh giá của họ về DGA Botnet trên bộ dữ liệu này và được công bố tại [85].

Alexa công bố 1.000.000 tên miền được xếp hạng dựa trên độ phổ biến của chúng đối với người dùng Internet từ hơn 70 quốc gia [43]. Những tên miền này được các nhóm nghiên cứu gán nhãn là lành tính trong các bài đánh giá của họ. Chúng ta dễ dàng tìm thấy những tên miền phổ biến được xếp hạng cao như google.com. facebook.com, youtube.com. Những tên miền được xếp hạng thấp hơn thì ít phổ biến hơn và trong một số trường hợp, nó trông khá giống những tên miền được sinh ra bởi DGA Botnet.

OSINT DGA feed [36] là một bộ dữ liệu đáng tin cậy về DGA Botnet, được tạo bởi Bambenek, bao gồm một khối lượng lớn các tên miền độc hại được thu thập và tổng hợp. Dữ liệu này được tác giả chia sẻ miễn phí cho cộng đồng khoa học và không được phép sử dụng vào mục đích thương mại. Cần lưu ý rằng dữ liệu này không phải luôn sẵn sàng để tải về mà người sử dụng cần gửi thư điện tử tới chủ sở hữu để xin cấp quyền. Hạn chế của bộ dữ liệu này là không gán nhãn cho các họ DGA Botnet, do đó chỉ có thể sử dụng được cho bài toán phát hiện.

360NetLab Dataset [37] là bộ dữ liệu về DNS được thu thập theo thời gian thực, bao gồm các tên miền độc hại đang truy vấn trên thế giới, được xây dựng và duy trì bởi nhóm nghiên cứu 360NetLab, Qihoo 360 Technology Co., Ltd. Website của nhóm cung cấp một công cụ cho phép dò quét và tổng hợp các tên miền độc hại theo

thời gian thực, sau đó bổ sung vào cơ sở dữ liệu đang có. Hạn chế của việc này là dữ liệu tên miền Botnet liên tục thay đổi do có sự cập nhật thường xuyên, khiến cho việc khó tạo thành một tiêu chuẩn chung cho đánh giá. Hơn nữa, số lượng họ DGA Botnet tại một thời điểm được lưu trữ trong bộ dữ liệu là không phong phú và ổn định như các bộ dữ liệu khác.

Johannes Bader công bố một bộ dữ liệu về DGA Botnets trên Github, với 48 họ DGA Botnets được chia sẻ [86]. Bộ dữ liệu này được công bố bắt đầu từ năm 2018, và liên tục được cập nhật các họ DGA Botnets mới. Mặc dù có một số họ DGA Botnets trùng lặp với những bộ dữ liệu trước đó, bộ dữ liệu của Bader cũng bao gồm một số họ DGA Botnet đặc trưng mà các bộ dữ liệu khác không có. Mỗi họ DGA Botnet thường bao gồm mã nguồn sinh tên miền tự động và các mẫu tên miền của họ đó. Tác giả công bố bộ dữ liệu với giấy phép GPL-2.0 License [87]. Một điểm rất đáng chú ý là tác giả cũng cung cấp các phân tích chi tiết về các họ DGA Botnets trên blog cá nhân của mình [88], giúp các nhà nghiên cứu có thể tìm hiểu rõ hơn về từng họ DGA Botnet cụ thể.

Bên cạnh Alexa top 1M domain, một bộ dữ liệu khác về các tên miền lành tính là The Majestic Million [89], được cung cấp bởi Majestic và công bố trên Website của họ. Bên cạnh việc cung cấp 1.000.000 tên miền lành tính, bộ dữ liệu này còn sắp xếp chúng theo thứ tự mức độ phổ biến trong thực tế. Danh sách các tên miền và mức độ phổ biến của chúng cũng được cập nhật liên tục từ hệ thống máy quét tự động. Do đó, mang lại sự yên tâm cho các nhà nghiên cứu khi dữ liệu luôn được cập nhật mới. Các tệp tin được cho phép tải về dưới dạng *CSV và sử dụng dưới giấy phép Creative Commons Attribution 3.0 Unported License [90].

Bảng 4.3 tóm tắt các đặc điểm chính của các bộ dữ liệu phổ biến về DGA Botnet đã nêu ở trên.

Bảng 4.3. Đặc điểm chính của các bộ dữ liệu phổ biến hiện nay về DGA Botnet

STT	**Bộ dữ liệu**	**Viết tắt**	**Số lượng tên miền lành tính**	**Số lượng tên miền của DGA Botnet**	**Sọ họ DGA Botnet**	**Định dạng**	**Năm công bố**
1	Andrey Abakumov’s DGA Repository [35]	AADR	1.000.000	801.667	08	txt	2016
2	Johannes Bader’s Domain Generation	JBR	Null	N/A	48	txt	2018

	Algorithms Repository [86]						
3	Alexa Top 1 milion domains [43]	AT1D	1.000.000	0	0	csv	2019
4	Botnet DGA Dataset [83]	BDD	1.000.000	1.803.333	10	csv	2020
5	UMUDGA Dataset [13]	UMU	1.000.000	Trên 30.000.000	50	arff, csv, txt	2020
6	DGArchive by Fraunhofer FKIE [84]	DFF	Null	N/A	86	csv	2020
7	OSINT DGA feed [36]	OSINT	1.000.000	495.186	0	txt	2021
8	360NetLab Dataset [37]	360NL	0	Không cố định	Không cố định	txt	2021
9	The Majestic Million [89]	TMM	1.000.000	0	0	csv	2021

4.1.4. Đặt vấn đề nghiên cứu

Nhìn chung, có sự khác nhau về cấu trúc và mục đích giữa các bộ dữ liệu cho Botnet nói chung và bộ dữ liệu cho DGA Botnet nói riêng. NCS phân nhóm và thể hiện chi tiết tại Bảng 4.4:

Bảng 4.4. Đánh giá về đặc điểm các nhóm bộ dữ liệu cho Botnet

Bộ dữ liệu	**Nhóm**	**Phát hiện Botnet**	**Phát hiện DGA Botnet**	**Phát hiện tấn công lưu**	**Lưu lượng mạng**	**Định dạng**		
						PCAP	**SCV**	**TXT**
CTU	Botnet	✓	×	×	✓	✓	✓	×
UGR	Botnet/IDS	✓	×	✓	✓	✓	✓	×
DLAB	Botnet	✓	×	×	✓	✓	✓	×
UNSW	Botnet/IDS	✓	×	✓	✓	✓	✓	×
ISCX	Botnet/IDS	✓	×	✓	✓	✓	✓	×
AADR	DGA Botnet	✓	✓	×	×	×	✓	✓
JBR	DGA Botnet	✓	✓	×	×	×	✓	✓

AT1D	DGA Botnet	✓	✓	×	×	×	✓	✓
BDD	DGA Botnet	✓	✓	×	×	×	✓	✓
UMU	DGA Botnet	✓	✓	×	×	×	✓	✓
DFF	DGA Botnet	✓	✓	×	×	×	✓	✓
OSINT	DGA Botnet	✓	✓	×	×	×	✓	✓
360NL	DGA Botnet	✓	✓	×	×	×	✓	✓
TMM	DGA Botnet	✓	✓	×	×	×	✓	✓

Bảng 4.4 thể hiện sự khác nhau giữa hai nhóm bộ dữ liệu Botnet/IDS với DGA Botnet. Nhìn chung, các bộ dữ liệu CTU, UGR, DLAB, UNSW và ISCX thuộc nhóm Botnet/IDS, chứa các gói tin là lưu lượng mạng, thường được lưu dưới dạng PCAP hoặc trích xuất ra CSV. Ngược lại, các bộ dữ liệu thuộc nhóm DGA Botnet như AADR, JBR, AT1D, BDD, UMU, DFF, OSINT, 360NL và TMM chỉ chứa các tên miền được lưu dưới dạng TXT hoặc CSV.

Các bộ dữ liệu Botnet/IDS có thể dùng cho đánh giá các kỹ thuật tấn công mạng khác nhưng không phù hợp để đánh giá cho các giải pháp DGA Botnet bởi chúng không được đánh nhãn tương ứng và số lượng mẫu tên miền cũng rất hạn chế. Trong khi bộ dữ liệu DGA Botnet chỉ dành cho đánh giá các giải pháp phát hiện DGA Botnet và được đánh nhãn tương ứng. Các bộ dữ liệu về Botnet chung bao gồm cả các lưu lượng mạng thô, nên có dung lượng lớn hơn rất nhiều (tính bằng GB) so với các bộ dữ liệu dành cho DGA Botnet đã được trích xuất các thông tin cần thiết (tính bằng MB). Cuối cùng, số lượng mẫu tên miền và số họ DGA Botnet trong các bộ dữ liệu chung về Botnet là rất ít so với các bộ dữ liệu về DGA Botnet chuyên dụng. Điều này tạo nên các hạn chế cho các đánh giá thuật toán, đặc biệt là các thuật toán học máy, học sâu cần một số lượng đủ dữ liệu để huấn luyện. Các vấn đề trên cho thấy sự cần thiết của một số dữ liệu phù hợp cho DGA Botnet.

4.1.5. Tiêu chí xây dựng bộ dữ liệu DGA Botnet

NCS đề xuất nhóm gồm 06 tiêu chí cơ bản đối với một bộ dữ liệu về DGA Botnet, bao gồm:

(1) Nhãn nhị phân: Bộ dữ liệu được gán 02 nhãn tương ứng nhãn 0 (tên miền lành tính) và nhãn 1 (tên miền của DGA Botnet). Bộ dữ liệu thỏa mãn tiêu chí này có thể được sử dụng để đánh giá bài toán phát hiện DGA Botnet.

(2) Nhãn đa lớp: Bộ dữ liệu được gán nhãn riêng biệt cho từng họ DGA Botnet, số lượng tùy theo quy mô của bộ dữ liệu. Bộ dữ liệu thỏa mãn tiêu chí này có thể được sử dụng để đánh giá bài toán phân loại DGA Botnet.

(3) Tên miền gốc: Bộ dữ liệu bao gồm tên miền gốc. Bộ dữ liệu thỏa mãn tiêu chí này cho phép các nhà khoa học có thể áp dụng các kỹ thuật xử lý dữ liệu đầu vào với các thuộc tính khác mà không bị phụ thuộc vào các thuộc tính đã trích xuất.

(4) Trích xuất thuộc tính: Bộ dữ liệu bao gồm các thuộc tính thông dụng đã được trích xuất và lưu trữ sẵn dưới các định dạng phổ biến và tiện dụng. Bộ dữ liệu thỏa mãn tính chất này cho phép các nhà khoa học thuận tiện áp dụng thuật toán của họ mà không cần tốn thời gian, tài nguyên tính toán để trích xuất thuộc tính. Đồng thời, cũng cho phép các phần mềm phổ biến đọc được dữ liệu một cách thuận lợi như phần mềm Weka, các thư viện tiền xử lý dữ liệu trong Keras Framework.

(5) Công khai: Bộ dữ liệu được công khai rộng rãi và dễ dàng tiếp cận. Bộ dữ liệu thỏa mãn tiêu chí này giúp các nhà khoa học có thể tải về và sử dụng bộ dữ liệu một cách dễ dàng với yêu cầu là không sử dụng cho mục đích thương mại.

(6) Tài liệu: Bộ dữ liệu được mô tả một cách chi tiết trong các tài liệu chính thống. Bộ dữ liệu thỏa mãn tiêu chí này giúp các nhà khoa học hiểu rõ về bộ dữ liệu, là cơ sở để kế thừa, bổ sung, phát triển trong tương lai.

Bảng 4.5 trình bày đánh giá về các tiêu chí ở trên đối với các bộ dữ liệu về DGA Botnet hiện nay. Trong bảng này, NCS bổ sung riêng hàng cuối là bộ dữ liệu UTL_DGA22 (ký hiệu *UTL*), để làm rõ tính đáp ứng tiêu chí của bộ dữ liệu mới đề xuất (trình bày ở phần tiếp theo) so với các bộ dữ liệu hiện có.

Bảng 4.5. Khái quát ưu điểm và hạn chế của các bộ dữ liệu DGA Botnet hiện có và bộ dữ liệu UTL_DGA22 đề xuất

	Nhãn nhị phân	**Nhãn đa lớp**	**Tên miền gốc**	**Trích xuất thuộc tính**	**Công khai**	**Tài liệu**
AADR	✓	✓	✓	×	✓	N/A
JBR	✓	✓	✓	×	✓	✓ [88]
AT1D	✓	×	✓	×	✓	N/A
BDD	✓	×	×	✓	✓	✓ [83]

UMU	✓	✓	✓	✓	✓	✓[44] [13]
DFF	✓	✓	✓	×	✓	✓ [84]
OSINT	✓	×	✓	×	✓*	N/A
360NL	✓	✓	✓	×	✓	N/A
TMM	✓	×	✓	×	✓	N/A
UTL	✓	✓	✓	✓	✓	✓*[CT5]*

**: Người dùng phải liên hệ với chủ sở hữu qua email để xin cấp phép truy cập*

N/A: Tài liệu chưa đầy đủ hoặc chưa có tính công bố

Bảng 4.5 cho thấy rằng, hầu hết các bộ dữ liệu về DGA Botnets hiện nay đều có thể áp dụng cho bài toán phát hiện, nhưng chỉ một số trong đó có thể áp dụng cho bài toán phân loại (như BDD, AT1D, OSINT hay TMM). Điều này là do bộ dữ liệu chỉ bao gồm các tên miền lành tính, hoặc có chứa các tên miền độc hại nhưng không được phân loại rõ ràng thuộc họ DGA Botnet nào. Hầu hết các bộ dữ liệu được lưu dưới dạng tên miền gốc mà chưa được trích xuất sẵn các thuộc tính, trừ bộ dữ liệu BDD chứa các thuộc tính đã được trích xuất. Bộ dữ liệu UMUDGA Dataset được công bố năm 2020, là một bộ dữ liệu mới với 50 họ DGA Botnets, chứa cả dữ liệu gốc và dữ liệu thuộc tính đã trích xuất, có thể xem là khá toàn diện. Cuối cùng, đa phần các bộ dữ liệu trên đều được công bố trên Internet. Tuy nhiên, việc công bố tài liệu cho từng bộ dữ liệu cũng chưa hoàn toàn đầy đủ

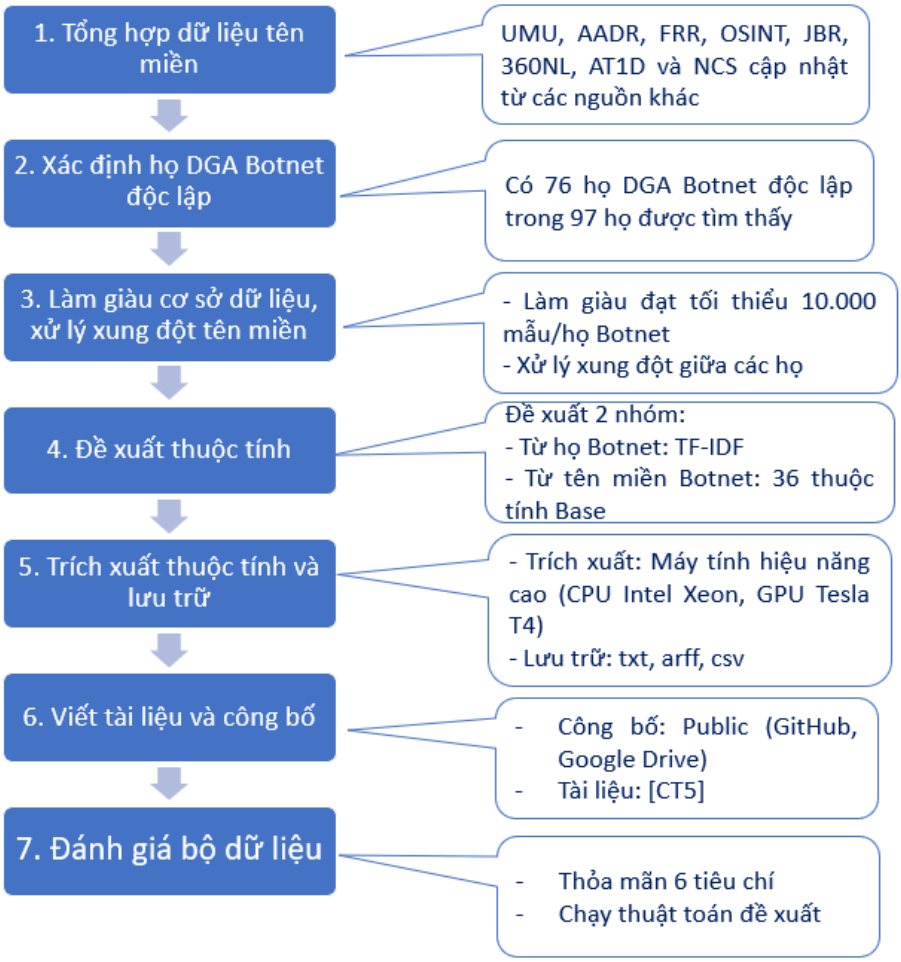
Mục tiêu của bộ dữ liệu đề xuất là kế thừa các kết quả của tất cả các bộ dữ liệu trên đáp ứng các tiêu chí đặt ra đồng thời cập nhật, bổ sung thêm các kết quả mới.

4.2. Bộ dữ liệu UTL_DGA22 đề xuất

Trong phần này, NCS đề xuất bổ sung hoàn thiện một quy trình xây dựng bộ dữ liệu và áp dụng để xây dựng một bộ dữ liệu mới về DGA Botnet là UTL_DGA22 [CT5]. Bộ dữ liệu này kế thừa các kết quả trước đó, đồng thời cập nhật, bổ sung và cải tiến để phù hợp với các tiêu chí đã đặt ra cho một bộ dữ liệu về DGA Botnets. UTL_DGA22 chỉ được sử dụng cho học tập, nghiên cứu, không được sử dụng cho các mục đích thương mại.

4.2.1. Quy trình xây dựng bộ dữ liệu

Để đạt được 06 tiêu chí ở trên, NCS đề xuất quy trình xây dựng bộ dữ liệu gồm 07 bước và kết quả tương ứng được tóm tắt tại Hình 4.1:



Hình 4.1. Quy trình 07 bước xây dựng bộ dữ liệu DGA Botnet và tóm tắt kết quả đạt được theo từng bước

Mô tả chi tiết các bước như sau:

- Bước 1. Tổng hợp dữ liệu tên miền: NCS tổng hợp dữ liệu tên miền lành tính và tên miền DGA Botnet từ các nguồn đáng tin cậy, bao gồm: UMUDGA Dataset [44] [13], DGArchive by Fraunhofer FKIE [84], Andrey Abakumov's GitHub Repository [35], The OSINT DGA feed [36], 360NetLab Dataset [37], Johannes Bader's GitHub Repository [86] và một số nguồn khác được NCS cập nhật. Dữ liệu được thể hiện dưới dạng tên miền gốc và lưu trữ trong tệp tin định dạng *txt.

- Bước 2. Xác định họ DGA Botnet độc lập: NCS phân tích và đánh giá, xác định các họ DGA Botnet độc lập, loại bỏ các trường hợp cùng một họ DGA Botnet nhưng khác tên gọi, hoặc các phiên bản khác nhau (được tính là các họ khác nhau) nhưng lại được tính chung là một họ. Kết quả có 76 họ DGA Botnets thỏa mãn các tiêu chí đặt ra.

- Bước 3. Làm giàu cơ sở dữ liệu, xử lý xung đột tên miền: NCS tiến hành làm giàu cơ sở dữ liệu tên miền, mục tiêu đạt 10.000 mẫu tên miền/họ DGA Botnet. Có

03 phương án được sử dụng để làm giàu dữ liệu, được thực hiện theo thứ tự ưu tiên như sau: (1) Kế thừa mẫu tên miền cùng họ từ các nguồn dữ liệu tin cậy; (2) Tái hiện và chạy mã nguồn thuật toán sinh tên miền (nếu được công bố) để tạo ra các tên miền cùng họ; (3) Thu thập trực tuyến thời gian thực từ nguồn 360NetLab cho đến khi đạt được số lượng mẫu yêu cầu của một họ DGA Botnet. Các mẫu tên miền sau khi thu thập đủ sẽ tiếp tục được xử lý xung đột để phát hiện các trường hợp nhầm lẫn, trùng lặp giữa các mẫu tên miền trong một họ hoặc giữa hai họ khác nhau. Dữ liệu cuối cùng sẽ được gán nhãn với họ DGA Botnet tương ứng.

- Bước 4: Đề xuất thuộc tính: NCS đề xuất 02 nhóm thuộc tính, bao gồm nhóm Base-Features (trích xuất từ một tên miền) và TF-IDF Features (trích xuất từ một họ tên miền). Những thuộc tính đề xuất được chứng minh sự hiệu quả trong quá trình tiền xử lý dữ liệu trong các nghiên cứu trước đó. Một số đánh giá bằng học máy với thông số cấu hình mặc định cho độ chính xác khả quan, điều này chứng tỏ các thuộc tính này hoàn toàn phù hợp và có thể ứng dụng trong giai đoạn Feature Selection cho các giải pháp đề xuất mới sau này.

- Bước 5. Trích xuất thuộc tính và lưu trữ: Bộ dữ liệu tên miền được lưu trữ dưới dạng tệp tin *TXT. NCS sử dụng máy tính hiệu năng cao với CPU Intel Xeon, GPU Tesla T4, RAM 24GB để trích xuất thuộc tính và lưu trữ dưới định dạng *ARFF và *CSV. Các định dạng này phù hợp để làm đầu vào cho các hệ thống như Weka, các thuật toán chạy trên Google CoLab hay Jupyter Notebook.

- Bước 6. Viết tài liệu và công bố: NCS viết tài liệu đặc tả cho bộ dữ liệu, đặt tên là UTL_DGA22 và thực hiện công bố tại [CT6]. NCS cũng đưa bộ dữ liệu lên các nền tảng chia sẻ công khai như Google Drive (gửi email đến tác giả) hoặc GitHub thông qua câu lệnh:

```
git clone https://github.com/tongtuan92/UTL_DGA22-Dataset.git
```

- Bước 7. Đánh giá bộ dữ liệu: NCS đánh giá lại bộ dữ liệu UTL_DGA22 với các tiêu chí đề xuất trước đó và khẳng định sự thỏa mãn 06 tiêu chí đã đặt ra bao gồm: (1) Nhãn nhị phân, (2) Nhãn đa lớp, (3) Tên miền gốc, (4) Trích xuất thuộc tính, (5) Công khai, (6) Tài liệu. Bên cạnh đó, NCS cũng đặt bộ dữ liệu UTL_DGA22 vào so sánh với 09 tiêu chí được Zago và cộng sự trình bày để có thêm các luận cứ về tính chất của bộ dữ liệu đề xuất.

4.2.2. Danh sách các họ DGA Botnet trong bộ dữ liệu UTL_DGA22

Bộ dữ liệu UTL_DGA22 bao gồm 76 họ DGA Botnet riêng biệt tương ứng với 76 nhãn. Danh sách các họ DGA Botnet này được liệt kê ở Bảng 4.6:

Bảng 4.6. Danh sách 76 họ DGA Botnet trong bộ dữ liệu UTL_DGA22

STT	**Tên (*Tên gọi khác*)**	**STT**	**Tên (*Tên gọi khác*)**
1	banjori (*MultiBanker 2 / BankPatch / BackPatcher*)	39	qsnatch
2	bazarbackdoor (*BazarLoader / Team9Backdoor*)	40	ramnit
3	bazarbackdoor_v2 (*BazarLoader / Team9Backdoor*)	41	ranbyus_v1
4	bazarbackdoor_v3 (*BazarLoader / Team9Backdoor*)	42	ranbyus_v2
5	chinad	43	reconyc
6	corebot	44	shiotob (*Urlzone / Bebloh*)
7	dircrypt	45	simda (*Shiz*)
8	dnschanger (*Alureon*)	46	sisron (*Tomb / win32_agent.wrq / trojan.scar*)
9	fobber_v1 (*Tinba_v3*)	47	suppobox_1
10	fobber_v2 (*Tinba_v3*)	48	suppobox_2
11	gozi_rfc4343	49	suppobox_3
12	gozi_nasa	50	symmi
13	gozi_luther	51	tempedreve
14	gozi_gpl	52	tinba [91] (*Tinybanker*)
15	kraken_v1 (*Oderoor / Bobax*)	53	vawtrak_v1
16	kraken_v2 (*Oderoor / Bobax*)	54	vawtrak_v2
17	locky	55	vawtrak_v3
18	monerodownloader	56	zloader
19	murofet_v1	57	cryptolocker
20	murofet_v2	58	rovnix
21	murofet_v3	59	matsnu
22	mydoom (*Novarg / Mimail.r / Shimgapi*)	60	ramdo
23	necurs	61	bigviktor

24	newgoz (*Gameover Zeus / Peer-to-Peer Zeus*)	62	ccleaner
25	nymaim	63	enviserv
26	nymaim2	64	vidro
27	padcrypt	65	dyre
28	pitou	66	beautiful baby
29	pizza	67	bamital
30	proslikefan	68	emotet
31	pushdo	69	infy
32	pykspa_improved_useful	70	murofetweekly
33	pykspa_improved_noise	71	oderoor
34	pykspa_precursor	72	pandabanker
35	qadars	73	sphinx
36	qakbot	74	szribi
37	virus	75	tinynuke
38	wd	76	torpig

4.2.3. Mô tả về các thuộc tính đề xuất

4.2.3.1. Nhóm thuộc tính dựa trên tên miền

Nhóm BaseFeatures gồm 36 thuộc tính, mỗi thuộc tính được trích xuất từ một tên miền gốc. Chi tiết các thuộc tính đề xuất và ký hiệu của chúng được trình bày tại Bảng 4.7:

Bảng 4.7. Các thuộc tính đề xuất thuộc nhóm BaseFeatures

STT	**Ký hiệu**	**Diễn giải**	**Công thức tính *(nếu có)***
1	dom_{TLD}	Top Level Domain	
2	dom_{SLD}	Second Level Domain	
3	L_{dom}	Độ dài của tên miền	$L_{dom} = length(dom)$
4	L_{TLD}	Độ dài của Top Level Domain	$L_{TLD} = length(dom_{TLD})$
5	L_{SLD}	Độ dài của Second Level Domain	$L_{TLD} = length(dom_{SLD})$

6	L_{OLD}	Độ dài của Other Level Domain	$L_{OLD} = L_{dom} - (L_{TLD} + L_{SLD} + 2)$
7	D_{dom}	Bậc của tên miền (tên miền cấu tạo gồm bao nhiêu phần)	$D_{dom} = \|parts(dom)\|$
8	HwP	Tên miền có bao gồm www hay không?	$HwP = \begin{cases} 1: Yes \\ 0: No \end{cases}$
9	HIP	Tên miền có bao gồm địa chỉ IP hay không	$HIP = \begin{cases} 1: Yes \\ 0: No \end{cases}$
10	LC_c	Độ dài của chuỗi phụ âm dài nhất	$LC_c = LCS(C, dom)$
11	LC_v	Độ dài của chuỗi nguyên âm dài nhất	$LC_v = LCS(V, dom)$
12	LC_l	Độ dài của chuỗi ký tự dài nhất	$LC_v = LCS(C \cup V, dom)$
13	LC_n	Độ dài của chuỗi số dài nhất	$LC_n = LCS(N, dom)$
14	AC_c	Độ dài trung bình của các chuỗi phụ âm	$AC_c = ACS(C, dom)$
15	AC_v	Độ dài trung bình của các chuỗi nguyên âm	$AC_v = ACS(V, dom)$
16	AC_l	Độ dài trung bình của các chuỗi ký tự	$AC_l = ACS(C \cup V, dom)$
17	AC_n	Độ dài trung bình của các chuỗi chữ số	$AC_n = ACS(N, dom)$
18	DC_c	Chênh lệch giữa chuỗi phụ âm dài nhất và ngắn nhất	$DC_c = DCS(C, dom)$
19	DC_v	Chênh lệch giữa chuỗi nguyên âm dài nhất và ngắn nhất	$DC_v = DCS(V, dom)$
20	DC_l	Chênh lệch giữa chuỗi ký tự dài nhất và ngắn nhất	$DC_l = DCS(C \cup V, dom)$
21	DC_n	Chênh lệch giữa chuỗi chữ số dài nhất và ngắn nhất	$DC_n = DCS(N, dom)$
22	NC_c	Số lượng các phụ âm khác nhau trong tên miền	$NC_c = NoC(C, dom)$
23	NC_v	Số lượng các nguyên âm khác nhau trong tên miền	$NC_v = NoC(V, dom)$

24	NC_l	Số lượng các chữ cái khác nhau trong tên miền	$NC_l = NC_c + NC_v$
25	NC_n	Số lượng các chữ số khác nhau trong tên miền	$NC_n = NoC(N, dom)$
26	NC_s	Số lượng các ký tự đặc biệt khác nhau trong tên miền	$NC_n = NoC(S, dom)$
27	RA_c	Tỉ lệ các ký tự phụ âm xuất hiện trong tên miền	$RA_c = \frac{NC_c}{21}$
28	RA_v	Tỉ lệ các ký tự nguyên âm xuất hiện trong tên miền	$RA_v = \frac{NC_v}{5}$
29	RA_l	Tỉ lệ các ký tự chữ cái xuất hiện trong tên miền	$RA_l = \frac{NC_l}{26}$
30	RA_n	Tỉ lệ các ký tự chữ số xuất hiện trong tên miền	$RA_n = \frac{NC_n}{10}$
31	RA_s	Tỉ lệ các ký tự đặc biệt xuất hiện trong tên miền	$RA_s = \frac{NC_s}{2}$
32	R_c	Tỉ lệ ký tự phụ âm trong tên miền	$R_c = RoC(C, dom)$
33	R_v	Tỉ lệ ký tự nguyên âm trong tên miền	$R_c = RoC(V, dom)$
34	R_l	Tỉ lệ ký tự chữ cái trong tên miền	$R_l = RoC(C \cup V, dom)$
35	R_n	Tỉ lệ ký tự chữ số trong tên miền	$R_n = RoC(N, dom)$
36	R_s	Tỉ lệ ký tự đặc biệt trong tên miền	$R_s = RoC(S, dom)$

Trong đó:

- C là tập các phụ âm:

$$C = \{b, c, d, f, g, h, j, k, l, m, n, p, q, r, s, t, v, w, x, y, z\}$$

- V là tập các nguyên âm:

$$V = \{a, e, i, o, u\}$$

- N là tập các ký tự chữ số:

$$N = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

- S là tập các ký tự đặc biệt:

$$S = \{".", "-"\}$$

- $LCS(T, dom)$: Là thuật toán tìm độ dài của chuỗi dài nhất chứa các ký tự thuộc tập T trong tên miền dom (Thuật toán 1).

Thuật toán 1: $LCS(T, dom)$

Input $T \neq \emptyset;\ |dom| > 0; T, dom_j \in C \cup V \cup N \cup S$

Output $l = LCS(T, dom)$

```
temp = 0
l = 0
for j in range(length(dom)):
        if dom_j ∈ T:
                temp = temp + 1
        else:
                l = max(l, temp)
                temp = 0
l = max(l, temp)
return l
```

- $ACS(T, dom)$: Là thuật toán tìm độ dài trung bình của các chuỗi được tạo bởi các ký tự thuộc tập T trong tên miền dom (Thuật toán 2).

Thuật toán 2: $ACS(T, dom)$

Input $T \neq \emptyset; |dom| > 0; T, dom_j \in C \cup V \cup N \cup S$

Output $a = ACS(T, dom)$

```
c = 0
st = 0
mark = true
for j in range(length(dom)):
        if dom_j ∈ T:
                c = c + 1
                if mark = true:
                        st = st + 1
                        mark = false
        else:
                mark = true
```

$a = c/st$

return a

- $DCS(T, dom)$: Là thuật toán tìm độ chênh lệch giữa chuỗi ký tự dài nhất và chuỗi ký tự ngắn nhất, các chuỗi được tạo bởi các ký tự thuộc tập T trong tên miền dom (Thuật toán 3).

Thuật toán 3: $DCS(T, dom)$

Input $T \neq \emptyset;\ |dom| > 0; T, dom_j \in C \cup V \cup N \cup S$

Output $d = DCS(T, dom)$

$temp = 0$

$minlength = +\infty$

$maxlength = -\infty$

for j in $range(length(dom))$:

    if $dom_j \in T$:

        $temp = temp + 1$

    else:

        $minlength = \min(minlength, temp)$

        $maxlength = \max(maxlength, temp)$

        $temp = 0$

$minlength = \min(minlength, temp)$

$maxlength = \max(maxlength, temp)$

$d = maxlength - minlength$

return d

- $NoC(T, dom)$: Là thuật toán tìm số lượng ký tự thuộc tập T xuất hiện ít nhất một lần trong tên miền dom (Thuật toán 4).

Thuật toán 4: $NoC(T, dom)$

Input $T \neq \emptyset;\ |dom| > 0; T, dom_j \in C \cup V \cup N \cup S$

Output $n = NoC(T, dom)$

$E = \emptyset$

$n = 0$

for j in $range(length(dom))$:

    if $dom_j \in T$ and $dom_j \notin E$

        $n = n + 1$

$E = E \cup \{dom_j\}$

return n

- $RoC(T, dom)$: Là thuật toán tìm tỉ lệ ký tự thuộc tập T trên số lượng ký tự của tên miền dom (Thuật toán 5).

Thuật toán 5: $RoC(T, dom)$

Input $T \neq \emptyset;\ |dom| > 0; T, dom_j \in C \cup V \cup N \cup S$

Output $c = RoC(T, dom)$

$count = 0$

for j in $range(length(dom))$:

    if $dom_j \in T$:

        $count = count + 1$

$c = count/length(dom)$

return c

Các thuộc tính đề xuất thể hiện vai trò, tính chất riêng của chúng trong từng tên miền hoặc họ tên miền. Điều này giúp chúng có ý nghĩa phân biệt trong việc phân loại. Vai trò của các thuộc tính được diễn giải tại Bảng 4.8:

Bảng 4.8. Vai trò của các thuộc tính đề xuất thuộc nhóm BaseFeatures

STT	Ký hiệu	Vai trò trong phân loại DGA Botnet
1	dom_{TLD}	Top Level Domain: TLD của mỗi tên miền trong cùng một họ DGA Botnet thường là giống nhau. Mỗi họ DGA Botnet khác nhau có TLD đặc trưng riêng.
2	dom_{SLD}	Second Level Domain: SLD của mỗi tên miền trong cùng một họ DGA Botnet thường là giống nhau. Mỗi họ DGA Botnet khác nhau có thể có SLD đặc trưng riêng.
3	L_{dom}	Độ dài của tên miền: Phần lớn các tên miền trong cùng một họ DGA Botnet có độ dài cố định. Độ dài cố định này thường là khác nhau giữa các họ DGA Botnet khác nhau.
4	L_{TLD}	Độ dài của Top Level Domain: Phân biệt độ dài TLD của tên miền giữa các họ DGA Botnet, thường có các giá trị là 2, 3 hoặc 4.
5	L_{SLD}	Độ dài của Second Level Domain: Phân biệt độ dài SLD của tên miền giữa các họ DGA Bonet.

6	L_{OLD}	Độ dài của Other Level Domain: Phân biệt độ dài OLD của tên miền giữa các họ DGA Botnet. Độ dài này thường khác nhau rõ ràng giữa các họ DGA Botnet.	
7	D_{dom}	Bậc của tên miền: Mỗi họ DGA Botnet có số bậc đặc trưng riêng.	
8	HwP	Tên miền có bao gồm www hay không: Phân biệt giữa các họ DGA Botnet có chứa và không chứa www trong tên miền	
9	HIP	Tên miền có bô gồm địa chỉ IP hay không: Phân biệt giữa các họ DGA Botnet có chứa và không chứa địa chỉ IP trong tên miền	
10	LC_c	Độ dài của chuỗi phụ âm dài nhất	Tỉ lệ các nguyên âm, phụ âm, chữ số trong bộ từ khóa, ký tự riêng của tên miền và thuật toán sinh tên miền có liên hệ mật thiết đến độ dài dài nhất của chuỗi các loại ký tự, góp phần tạo nên đặc trưng của họ DGA Botnet đó
11	LC_v	Độ dài của chuỗi nguyên âm dài nhất	
12	LC_l	Độ dài của chuỗi ký tự dài nhất	
13	LC_n	Độ dài của chuỗi số dài nhất	
14	AC_c	Độ dài trung bình của các chuỗi phụ âm	Tỉ lệ các nguyên âm, phụ âm, chữ số trong bộ từ khóa, ký tự riêng của tên miền và thuật toán sinh tên miền có liên hệ mật thiết đến độ dài trung bình của chuỗi các loại ký tự, góp phần tạo nên đặc trưng của họ DGA Botnet đó
15	AC_v	Độ dài trung bình của các chuỗi nguyên âm	
16	AC_l	Độ dài trung bình của các chuỗi ký tự	
17	AC_n	Độ dài trung bình của các chuỗi chữ số	
18	DC_c	Chênh lệch giữa chuỗi phụ âm dài nhất và ngắn nhất	Tỉ lệ các nguyên âm, phụ âm, chữ số trong bộ từ khóa, ký tự riêng của tên miền và thuật toán sinh tên miền có liên hệ mật thiết đến độ dài của chuỗi các loại ký tự. Sự chênh lệch giữa chuỗi dài nhất và chuỗi ngắn nhất cũng là một đặc trưng riêng của họ DGA Botnet.
19	DC_v	Chênh lệch giữa chuỗi nguyên âm dài nhất và ngắn nhất	
20	DC_l	Chênh lệch giữa chuỗi ký tự dài nhất và ngắn nhất	
21	DC_n	Chênh lệch giữa chuỗi chữ số dài nhất và ngắn nhất	
22	NC_c	Số lượng các phụ âm khác nhau trong tên miền	Do tỉ lệ các nguyên âm, phụ âm, chữ số trong bộ từ khóa, ký tự riêng của tên miền và thuật toán sinh tên miền là khác nhau giữa các họ DGA Botnet. Nên sự phân bố số lượng của từng loại
23	NC_v	Số lượng các nguyên âm khác nhau trong tên miền	
24	NC_l	Số lượng các chữ cái khác nhau trong tên miền	

25	NC_n	Số lượng các chữ số khác nhau trong tên miền	ký tự trong tên miền cũng có đặc trưng riêng cho từng họ.
26	NC_s	Số lượng các ký tự đặc biệt khác nhau trong tên miền	
27	RA_c	Tỉ lệ các ký tự phụ âm xuất hiện trong tên miền	Do tỉ lệ các nguyên âm, phụ âm, chữ số trong bộ từ khóa, ký tự riêng của tên miền và thuật toán sinh tên miền là khác nhau giữa các họ DGA Botnet. Nên tỉ lệ xuất hiện của từng loại ký tự trong tên miền cũng có đặc trưng riêng cho từng họ.
28	RA_v	Tỉ lệ các ký tự nguyên âm xuất hiện trong tên miền	
29	RA_l	Tỉ lệ các ký tự chữ cái xuất hiện trong tên miền	
30	RA_n	Tỉ lệ các ký tự chữ số xuất hiện trong tên miền	
31	RA_s	Tỉ lệ các ký tự đặc biệt xuất hiện trong tên miền	
32	R_c	Tỉ lệ ký tự phụ âm trong tên miền	Do tỉ lệ các nguyên âm, phụ âm, chữ số trong bộ từ khóa, ký tự riêng của tên miền. Nên sự tỉ lệ xuất hiện của từng loại ký tự trong tên miền cũng có đặc trưng riêng cho từng họ. Trong tỉ lệ này, các ký tự trùng nhau được tính duy nhất một lần.
33	R_v	Tỉ lệ ký tự nguyên âm trong tên miền	
34	R_l	Tỉ lệ ký tự chữ cái trong tên miền	
35	R_n	Tỉ lệ ký tự chữ số trong tên miền	
36	R_s	Tỉ lệ ký tự đặc biệt trong tên miền	

Bảng 4.9 minh họa ví dụ giá trị của các thuộc tính đề xuất thuộc nhóm Base Features, được áp dụng trên một tên miền lành tính và một tên miền của DGA Botnet.

- Tên miền lành tính - Nhãn 0: google.com.vn.

- Tên miền DGA Botnet - Nhãn 1: ojhpiwdmmyxneo88.com.

Bảng 4.9. Minh họa giá trị của 36 thuộc tính nhóm Base-Features

STT	**Thuộc tính**	***Nhãn 0***	***Nhãn 1***	**STT**	**Thuộc tính**	***Nhãn 0***	***Nhãn 1***
1	dom_{TLD}	vn	com	19	DC_v	1	1
2	dom_{SLD}	com	ojhpiwdmmyxneo88	20	DC_l	4	11
3	L_{dom}	13	20	21	DC_n	0	2
4	L_{TLD}	2	3	22	NC_c	6	10
5	L_{SLD}	3	0	23	NC_v	2	3
6	L_{OLD}	6	16	24	NC_l	8	13

7	D_{dom}	3	2	25	NC_n	0	1
8	HwP	0	0	26	NC_s	1	1
9	HIP	0	0	27	RA_c	0,29	0,48
10	LC_c	2	7	28	RA_v	0,40	0,60
11	LC_v	2	2	29	RA_l	0,31	0,50
12	LC_l	6	14	30	RA_n	0,00	0,10
13	LC_n	0	2	31	RA_s	0,50	0,50
14	AC_c	1,4	3,5	32	R_c	0,54	0,60
15	AC_v	1,33	1,25	33	R_v	0,31	0,25
16	AC_l	3,66	8,5	34	R_l	0,85	0,85
17	AC_n	0	2	35	R_n	0,00	0,10
18	DC_c	1	6	36	R_s	0,15	0,.05

4.2.3.2. Nhóm thuộc tính dựa trên họ tên miền

Nhận xét rằng, mỗi họ tên miền có các đặc trưng khác nhau về không gian ký tự, bộ từ khóa, thứ tự các từ khóa trong cấu tạo tên miền và sự phụ thuộc vào nhân sinh tên miền. Các đặc trưng TF-IDF có khả năng thể hiện được những đặc trưng trên đối với mỗi họ tên miền, khi mà số lượng mẫu tên miền trong một họ là đủ lớn. NCS đề xuất sử dụng các thuộc tính này và gọi là nhóm thuộc tính TI-IDF Features.

Khái niệm về TF-IDF được trình bày tại Mục 2.2.1.3. Để tính toán đặc trưng TF-IDF, NCS trích xuất các n-gram và được lưu trữ như một thành phần của bộ dữ liệu đề xuất.

4.2.4. Cấu trúc lưu trữ của bộ dữ liệu UTL_DGA22

Bộ dữ liệu DGA_UTL22 được cấu trúc gồm hai phần tương ứng với hai thư mục, gồm DGA_Botnets_Domains và DGA_Botnets_Features_Extraction.

- Thư mục DGA_Botnets_Domains bao gồm 76 tệp tin TXT, mỗi tệp tin chứa 10.000 tên miền gốc của họ DGA Botnets tương ứng. Thư mục cũng bao gồm tệp tin AT1M.txt lưu trữ 1.000.000 tên miền lành tính được cung cấp bởi Alexa. Danh sách các họ DGA Botnets có trong bộ dữ liệu được trình bày tại Mục 4.2.2.

- Thư mục DGA_Botnets_Features_Extraction, chứa dữ liệu được trích xuất từ 10.000 tên miền cho mỗi họ DGA Botnet và nhóm các tên miền lành tính, gồm hai thư mục con như sau:

+ Thư mục Base_Features: Chứa 76 tệp tin CSV và 76 tệp tin ARFF, lưu trữ các thuộc tính được trích xuất cho 76 họ DGA Botnet và 01 tệp tin CSV lưu họ tên miền lành tính.

+ Thư mục TF-IDF_Features: Chứa 308 tệp tin thuộc tính được trích xuất từ 76 họ DGA Botnet và một họ các tên miền lành tính. Mỗi họ gồm 4 tệp tin như sau: Tệp tin thuộc tính 2-gram, 3-gram tương ứng được lưu dưới dạng CSV hoặc ARFF. Tệp tin còn lại là ALLS.pickle chứa thuộc tính được trích xuất trên toàn bộ các tên miền.

4.2.5. Đánh giá với tiêu chí của Zago và cộng sự

Trong nghiên cứu của mình với bộ dữ liệu UMUDGA, Zago và cộng sự đề xuất 09 tiêu chí cho một bộ dữ liệu về DGA Botnet [44]. Trong phần này, NCS đánh giá bộ dữ liệu UTL_DGA22 theo quan điểm 09 tiêu chí trên, được thể hiện ở Bảng 4.10:

Bảng 4.10. Đánh giá tính đáp ứng của bộ dữ liệu UTL_DGA22 với các tiêu chí của Zago và cộng sự

Tiêu chí	**Mô tả của Zago**	**Đánh giá**	**Luận giải của NCS**
Tính tổng hợp (Def 2.1. SYNT)	Dữ liệu tạo ra bằng hai cách: Tạo các mẫu hoặc kết hợp nhiều nguồn với nhau	Đáp ứng	Các mẫu tên miền trong bộ dữ liệu UTL_DGA22 có được từ kế thừa, kết hợp nhiều nguồn dữ liệu tin cậy trước đó, tạo ra bằng thuật toán hoặc tổng hợp từ 360NetLab (mô tả tại Bước 1, Bước 3 mục 4.2.1)
Tính phổ biến (Def 2.2. GNRL)	Dữ liệu bao gồm nhiều họ mã độc để gần giống thực tế thay vì chỉ chứa một số lượng ít họ nào đó	Đáp ứng	Bộ dữ liệu gồm 76 họ DGA Botnet phổ biến (mô tả tại mục 4.2.2), bao quát phạm vi rộng thay vì chỉ một số ít mẫu họ DGA Botnet. Kết quả gần nhất của Zago và cộng sự có 50 họ DGA Botnet.
Tính đại diện (Def 2.3. RPST)	Bộ dữ liệu có đủ số lượng mẫu cho mỗi nhãn để phản ánh được chính xác các đặc tính của nhãn đó	Đáp ứng	Trong bộ dữ liệu UTL_DGA22, mỗi họ DGA Botnet chứa 10.000 mẫu, đủ để phản ánh chính xác đặc tính của loại DGA Botnet đó.
Tính cân bằng (Def 2.4. BLNC)	Bộ dữ liệu chứa một số lượng mẫu tương đương cho mỗi nhãn, không có lớp nào nhiều mẫu hơn so với lớp khác	Đáp ứng	Trong bộ dữ liệu UTL_DGA22, các họ DGA Botnet đều đạt cân bằng mẫu với 10.000 mẫu cho mỗi họ, đảm bảo sự cân bằng dữ liệu (mô tả tại Bước 3 mục 4.2.1)
Tính mở rộng (Def 2.5. EXTS)	Bộ dữ liệu được cung cấp công khai và tài liệu đầy đủ, cho phép	Đáp ứng	Bộ dữ liệu UTL_DGA22 được công bố công khai trên GitHub và Google Drive. Bộ dữ liệu có tài liệu mô tả tại [CT5].

	cộng đồng nghiên cứu mở rộng hoặc kết hợp với các nguồn dữ liệu khác nhằm tái sử dụng lại nó		Các nhà khoa học có thể sử dụng, kế thừa, phát triển bộ dữ liệu cho nghiên cứu khoa học, không nhằm mục đích thương mại.
Tính xác minh (Def 2.6. VRFB)	Dữ liệu có trong bộ dữ liệu được cung cấp đủ phương tiện để cộng đồng nghiên cứu tìm hiểu, xác định tính nhất quán. Lý tưởng nhất là bộ dữ liệu có thể được tái tạo đầy đủ thông qua tài liệu.	Đáp ứng	Các họ DGA Botnet được nghiên cứu, xác minh, phát hiện và xử lý các xung đột để đảm bảo tính chính xác (mô tả tại Bước 2 mục 4.2.1) và được trích dẫn nguồn cho từng họ DGA Botnet tại Table 11 của [CT6]. Bên cạnh đó, các mô tả tại [CT6] là đầy đủ cho phép tái tạo lại bộ dữ liệu theo quy trình NCS đề xuất trong luận án.
Tính định hướng riêng tư (Def 2.7. PROR)	Bộ dữ liệu được thiết kế để không bao gồm bất kỳ nội dung nào có thể gây hại đến quyền riêng tư và không yêu cầu cộng đồng nghiên cứu vi phạm quyền riêng tư của họ khi sử dụng dữ liệu.	Đáp ứng	Bộ dữ liệu được xây dựng dựa trên các dữ liệu được chia sẻ công khai, giấy phép hợp lệ với mọi quyền sử dụng, ngoại trừ việc thương mại. Dữ liệu được trích xuất dưới các dạng tệp tin dữ liệu, không phải tệp tin thực thi và không đính kèm các virus, mã độc nguy hiểm (mô tả tại mục 4.2.4). Cộng đồng nghiên cứu có thể tiếp cận sử dụng mà không bị đòi hỏi bất kỳ yêu cầu nào vi phạm tính riêng tư của họ ví dụ như danh tính, chức danh, vị trí làm việc.
Tính sẵn sàng cho học máy (Def 2.8. MLRD)	Bộ dữ liệu với các mẫu được quản lý cẩn thận, không có giá trị bị thiếu hoặc ký tự không mong muốn, định dạng dữ liệu đồng nhất trên tất cả các mẫu và thích hợp để sử dụng với các công cụ hàng đầu trong lĩnh vực	Đáp ứng	Bộ dữ liệu với các mẫu được gán nhãn, định dạng, lưu trữ một cách cẩn thận và khoa học (mô tả tại mục 4.2.4). Dữ liệu có thể được áp dụng thuận tiện để chạy các thuật toán sử dụng các công cụ và nền tảng hiện đại như Weka, Keras Framwork, Anaconda trên Google Colab, Jupyter Notebook. NCS cũng đã thử nghiệm chạy các thuật toán NCM, học máy, học sâu trên bộ dữ liệu trên (mô tả tại mục 4.3.2)
Tính gán nhãn (Def 2.9. LABL)	Mỗi mẫu được xác định cẩn thận tương ứng với một hoặc nhiều lớp (nhãn), có	Đáp ứng	Dữ liệu được gán nhãn cẩn thận và chính xác, trong đó: Nhãn lành tính được quy ước là 0 và lưu các mẫu tại tệp tin legit-1000000.txt. Các nhãn của 76 họ DGA

	thể cung cấp một mức độ chi tiết khác nhau của các nhãn.		Botnet được quy ước từ 01 đến 76 và các mẫu thuộc một nhãn được lưu riêng vào một tệp tin txt rất thuận tiện cho việc đọc và điều chỉnh quy ước nhãn sang tên của họ DGA Botnet (mô tả tại mục 4.2.4)

Các đánh giá trên cho thấy, bộ dữ liệu UTL_DGA22 đáp ứng đầy đủ 09 tiêu chí theo quan điểm của Zago và cộng sự.

4.3. Thử nghiệm một số thuật toán trên bộ dữ liệu đề xuất

4.3.1. Thử nghiệm áp dụng thuộc tính đề xuất

NCS sử dụng các thuộc tính đề xuất làm đầu vào của các thuật toán học máy cơ bản để phát hiện và phân loại DGA Botnet. Các thuật toán được sử dụng bao gồm: RF, LR, NB, SVM, k-NN, DT, NN và AB. Các mô hình được cài đặt thông số mặc định. Bộ dữ liệu đánh giá được lấy từ bộ DGA_UTL22, cụ thể:

- Tên miền lành tính: Gồm 1.000.000 tên miền lành tính của Alexa.

- Tên miền độc hại: Gồm 76 họ DGA Botnet, mỗi họ gồm 10.000 tên miền. Tổng số lượng tên miền độc hại là 760.000 tên miền.

Các tham số đánh giá bao gồm: Accuray, Precision, Recall và F_1-Score. Giá trị các tham số cài đặt cho các mô hình học máy được cho tại Bảng 4.11.

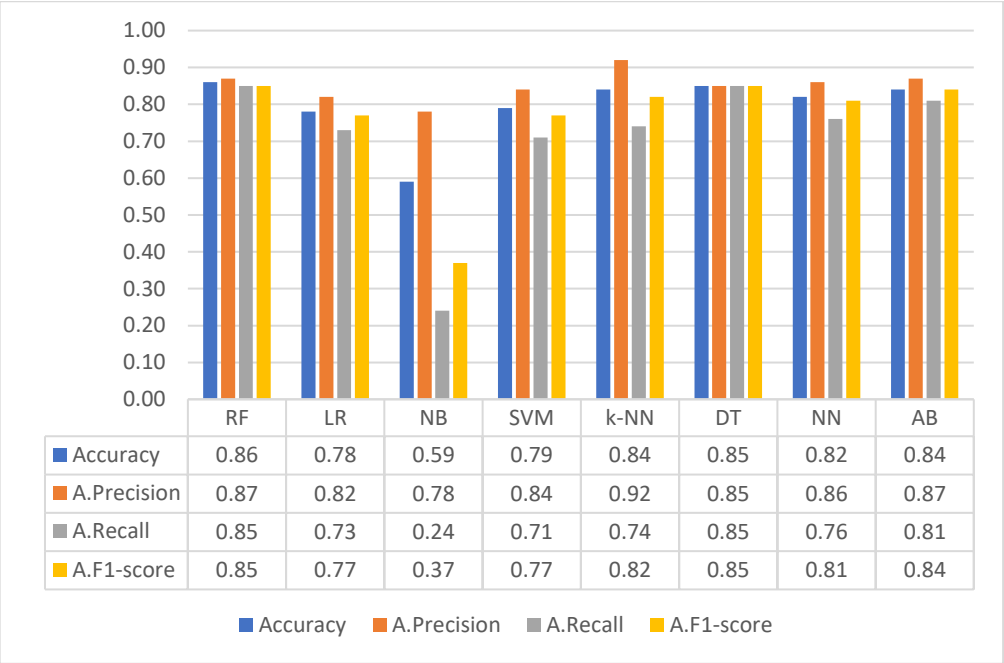
Bảng 4.11. Giá trị các tham số cài đặt cho các mô hình học máy khi đánh giá trên bộ dữ liệu UTL_DGA22

STT	Mô hình	Thư viện	Cấu hình tham số
1	Logistic Regression	Logistic Regression (sklearn.linear_model)	penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=42, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None
2	Naive Bayes	BernoulliNB (sklearn.naive_bayes)	*, alpha=1.0, binarize=0.0, fit_prior=True, class_prior=None
3	Decision Tree	DecisionTree Classifier (sklearn.tree)	*, criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=42, max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0)

4	Neural Network	MLPClassifier (sklearn.neural_ network)	hidden_layer_sizes=(100,), activation='relu', *, solver='lbfgs', alpha=0.00001, batch_size='auto', learning_rate='constant', learning_rate_init=0.001, power_t=0.5, max_iter=200, shuffle=True, random_state=42, tol=0.0001, verbose=False, warm_start=False, momentum=0.9, nesterovs_momentum=True, early_stopping=False, validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08, n_iter_no_change=10, max_fun=15000
5	Support Vector Machine	LinearSVC (sklearn.svm)	penalty='l2', loss='squared_hinge', *, dual=True, tol=0.0001, C=0.05, multi_class='ovr', fit_intercept=True, intercept_scaling=1, class_weight=None, verbose=0, random_state=None, max_iter=1000
6	Random Forest	RandomForestC lassifier (sklearn. ensemble)	n_estimators=10, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=42, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None
7	K-Nearest Neighbor	Kneighbors Classifier (sklearn. neighbors)	n_neighbors=3, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None
8	Adaptive Boosting	AdaBoost Classifier (sklearn. ensemble)	base_estimator=None, *, n_estimators=100, learning_rate=1.0, algorithm='SAMME.R', random_state=42

4.3.1.2. Thử nghiệm đối với bài toán phát hiện DGA Botnet

Hình 4.2 thể hiện kết quả phát hiện khi sử dụng các thuộc tính Base-Features làm đầu vào:



	RF	LR	NB	SVM	k-NN	DT	NN	AB
Accuracy	0.86	0.78	0.59	0.79	0.84	0.85	0.82	0.84
A.Precision	0.87	0.82	0.78	0.84	0.92	0.85	0.86	0.87
A.Recall	0.85	0.73	0.24	0.71	0.74	0.85	0.76	0.81
A.F1-score	0.85	0.77	0.37	0.77	0.82	0.85	0.81	0.84

Hình 4.2. Kết quả phát hiện sử dụng Base Features làm đầu vào trên bộ dữ liệu UTL_DGA22

Hình 4.3 thể hiện kết quả phát hiện khi sử dụng các thuộc tính TF-IDF-Features làm đầu vào:

	RF	LR	NB	SVM	k-NN	DT	NN	AB
Accuracy	0.86	0.78	0.59	0.79	0.84	0.85	0.82	0.84
A.Precision	0.87	0.82	0.78	0.84	0.92	0.85	0.86	0.87
A.Recall	0.85	0.73	0.24	0.71	0.74	0.85	0.76	0.81
A.F1-score	0.85	0.77	0.37	0.77	0.82	0.85	0.81	0.84

Hình 4.3. Kết quả phát hiện sử dụng TF-IDF Features làm đầu vào trên bộ dữ liệu UTL_DGA22

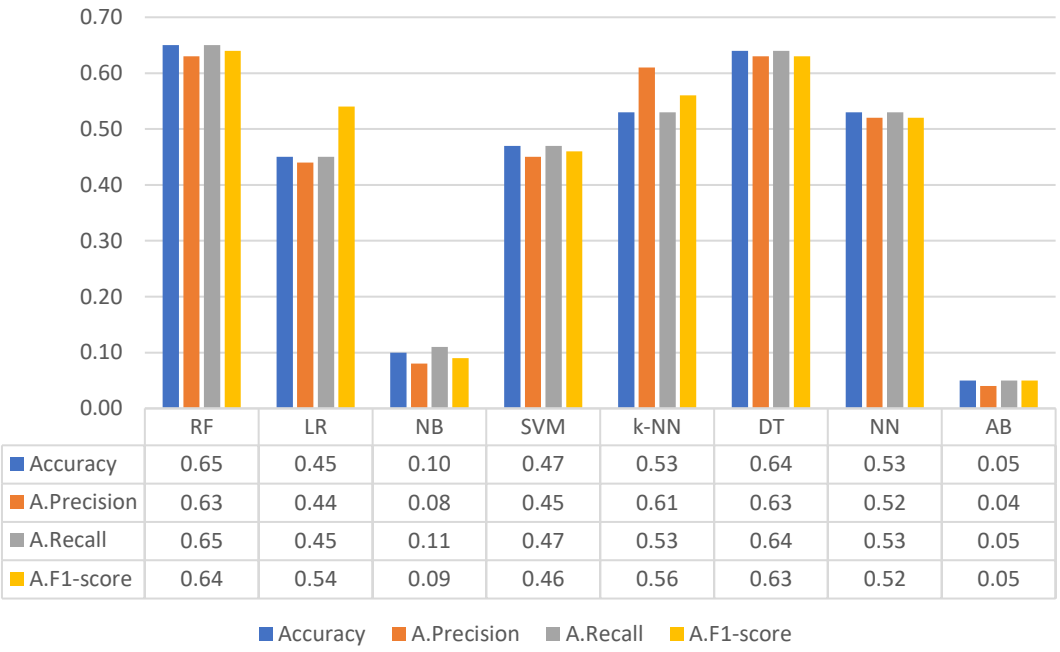
Nhận xét rằng, trong bài toán phát hiện, cả hai bộ thuộc tính Base và TF-IDF đều chứng tỏ sự phù hợp khi làm đầu vào cho các thuật toán học máy. Điều này được

thể hiện ở việc phần lớn các mô hình đều có Accuracy, A.Precision, A.Recall và A.F_1-Score đạt được giá trị cao, trong khoảng 0,82 đến 0,94 đối với A.F_1-score. Ví dụ, mô hình RF có Accuracy đạt 0,86 khi sử dụng các thuộc tính Base. Hay mô hình LR, NN cho Accuracy đạt 0,94 khi sử dụng các thuộc tính TF-IDF. Các mô hình còn lại cũng cho kết quả tốt, chứng tỏ rằng các nhóm thuộc tính đề xuất là hoàn toàn phù hợp, mang những đặc trưng riêng của từng họ tên miền, từ đó giúp cho các mô hình phân loại hoạt động hiệu quả.

Từ Hình 4.2 và Hình 4.3 cũng có thể thấy rằng, các thuật toán học máy có hiệu quả tốt hơn khi sử dụng nhóm các thuộc tính TF-IDF làm đầu vào so với nhóm các thuộc tính Base. Việc TF-IDF là một kỹ thuật trích xuất thuộc tính hiệu quả trong bài toán DGA Botnet cũng được chứng mình bởi các nghiên cứu trước đó.

4.3.1.3. Thử nghiệm đối với bài toán phân loại DGA Botnet

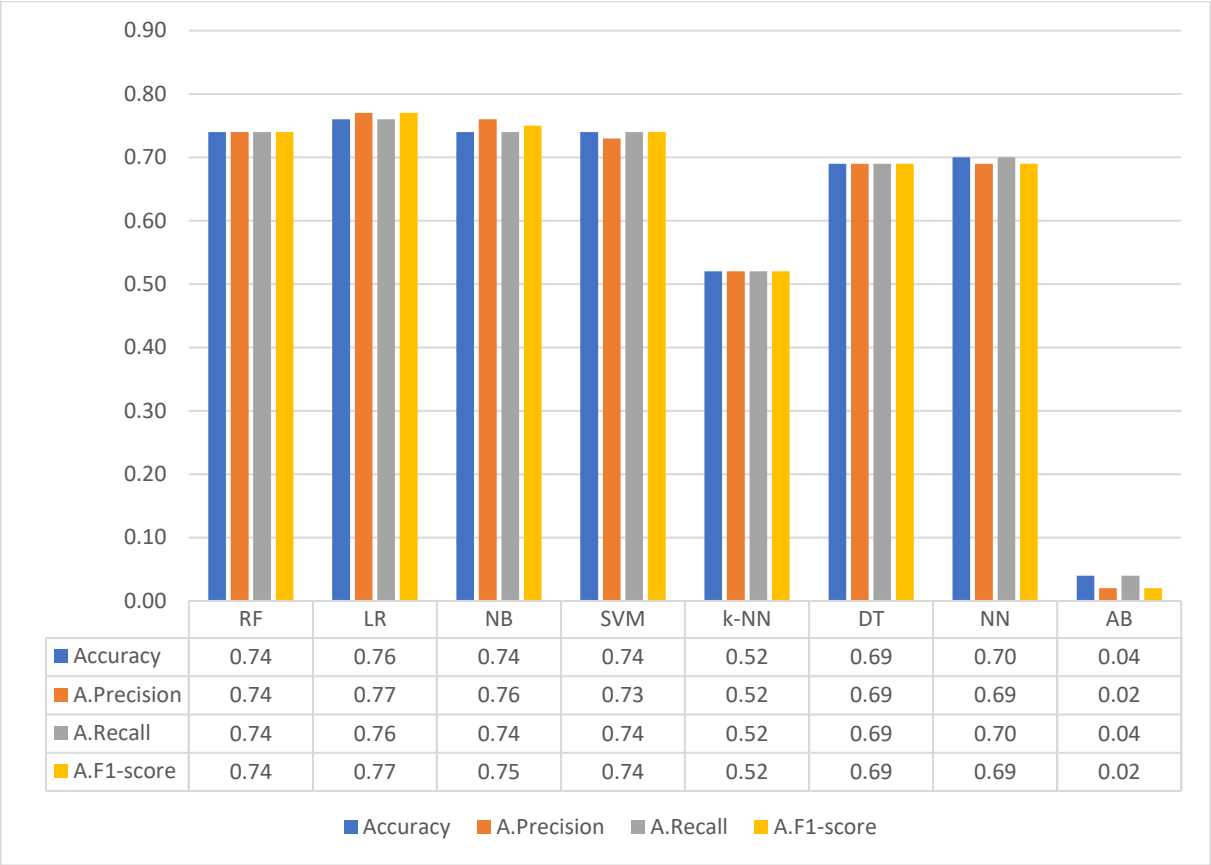
Trong trường hợp sử dụng các bộ thuộc tính Base và TF-IDF cho bài toán phân loại, ta có các kết quả lần lượt được cho ở Hình 4.4 và Hình 4.5.



	RF	LR	NB	SVM	k-NN	DT	NN	AB
Accuracy	0.65	0.45	0.10	0.47	0.53	0.64	0.53	0.05
A.Precision	0.63	0.44	0.08	0.45	0.61	0.63	0.52	0.04
A.Recall	0.65	0.45	0.11	0.47	0.53	0.64	0.53	0.05
A.F1-score	0.64	0.54	0.09	0.46	0.56	0.63	0.52	0.05

Hình 4.4. Kết quả phân loại của các mô hình học máy sử dụng Base Features trên bộ dữ liệu UTL_DGA22

Hình 4.4 cho thấy, có 6/8 thuật toán học máy cho kết quả phân loại ở mức trên 0,52. Trong đó, thuật toán RF đạt Accuracy, A.Precision, A.Recall và A.F_1-Score mức khoảng 0,65; thuật toán DT đạt Accuracy, A.Precision, A.Recall và A.F_1-Score lần lượt là 0,66, 0,64, 0,66 và 0,65. Chỉ hai thuật toán đạt kết quả thấp là NB và AB trong thử nghiệm này với Accuracy lần lượt là 0,10 và 0,05.



	RF	LR	NB	SVM	k-NN	DT	NN	AB
Accuracy	0.74	0.76	0.74	0.74	0.52	0.69	0.70	0.04
A.Precision	0.74	0.77	0.76	0.73	0.52	0.69	0.69	0.02
A.Recall	0.74	0.76	0.74	0.74	0.52	0.69	0.70	0.04
A.F1-score	0.74	0.77	0.75	0.74	0.52	0.69	0.69	0.02

Hình 4.5. Kết quả phân loại của các mô hình học máy sử dụng TF-IDF Featues trên bộ dữ liệu UTL_DGA22

Hình 4.5 cho thấy, khi sử dụng các thuộc tính TF-IDF làm đầu vào, các thuật toán học máy cho hiệu năng tốt với Accuracy hầu hết đạt từ 0,70 trở lên (RF, LR, SVM, NN. Mô hình AB và k-NN hoạt động kém hiệu quả trong trường hợp này với Accuracy lần lượt đạt 0,04 và 0,52.

Tổng kết lại, kết quả thực nghiệm trên cho thấy các tập thuộc tính được đề xuất là hoàn toàn phù hợp làm đầu vào cho các bộ phân loại, thể hiện bằng việc chúng đạt độ chính xác tốt khi chỉ áp dụng các mô hình học máy cơ bản. Việc lựa chọn các thuộc tính phụ thuộc vào từng phương pháp đề xuất mới để tối ưu hóa độ chính xác và thời gian huấn luyện. Sự kết hợp của nhóm thuộc tính Base và thuộc tính TF-IDF được khuyến nghị để có thể đạt được độ chính xác tối ưu.

4.3.2. Thử nghiệm áp dụng một số thuật toán

Trong phần này, NCS tiến hành chạy một số thuật toán đã trình bày tại Chương 2, Chương 3 trên bộ dữ liệu mới UTL_DGA22, bao gồm: Thuật toán NCM, mô hình VEA và HEA, hai mô hình học sâu LA_Bin07 và LA_Mul07.

4.3.2.1. Thử nghiệm với thuật toán phân cụm NCM

Thuật toán phân cụm NCM áp dụng cho bài toán phát hiện DGA Botnet, được mô tả tại Mục 2.1 của luận án này. Trong thử nghiệm này, NCS sử dụng 100.000 tên miền lành tính và 100.000 tên miền của DGA Botnet được lựa chọn ngẫu nhiên và phân phối đều cho 76 họ DGA Bonet. Kết quả được cho tại Bảng 4.12:

Bảng 4.12. Kết quả đánh giá thuật toán NCM trên bộ dữ liệu UTL_DGA22

Nhãn	**Precision**	**Recall**	**F_1-Score**
0	0,66	0,93	0,77
1	0,91	0,48	0,62
Avg	***0,79***	***0,70***	***0,70***

Nhận xét rằng, thuật toán NCM có A.Precision, A.Recall và A.F_1-score lần lượt đạt 0,79, 0,70 và 0,70 trên bộ dữ liệu UTL_DGA22. So sánh với kết quả trên 04 bộ dữ liệu AADR, 360NetLab, OSINT và UMUDGA (tại Bảng 2.3), ta thấy rằng ngoại trừ bộ dữ liệu OSINT, thì kết quả đánh giá của NCM trên UTL_DGA22 thấp hơn so với 03 bộ dữ liệu còn lại, tương ứng A.F_1-score là 0,70 so với lần lượt 0,79, 0,84 và 0,84. Điều này thể hiện rằng, bộ dữ liệu UTL_DGA22 là khó để phân loại hơn so với các bộ dữ liệu trước đó.

4.3.2.2. Thử nghiệm với các thuật toán học máy

Các mô hình học máy và VEA, HEA áp dụng cho bài toán phát hiện, được đề xuất và mô tả tại Mục 2.2 của luận án này. NCS sử dụng 1.000.000 tên miền lành tính và 76.000 tên miền của DGA Botnet tương ứng với 76 họ DGA Bonet trong bộ dữ liệu UTL_DGA22, mỗi họ có chứa đúng 10.000 tên miền. Kết quả đánh giá được thể hiện ở Bảng 4.13:

Bảng 4.13. Kết quả đánh giá các thuật toán học máy đề xuất trên bộ dữ liệu UTL_DGA22

Mô hình	**Acc**	**A.Pre**	**A.Re**	**A.F_1.**	**Thời gian huấn luyện (s)**	**Thời gian đánh giá (s)**
LR	0,96	0,97	0,95	0,96	75,59	0,04
NB	0,90	0,91	0,86	0,89	1,34	0,14
DT	0,90	0,90	0,88	0,89	16.004,87	0,71
NN	0,97	0,97	0,96	0,96	4.777,21	0,96
SVM	0,96	0,97	0,94	0,96	11,04	0,03

RF	0,60	1,00	0,07	0,14	62,57	4,06
k-NN	0,80	0,97	0,56	0,71	0,47	19.462,80
AB	0,84	0,84	0,79	0,81	2.917,49	12,22
VEA	0,97	0,97	0,96	0,96	16.803,67	18,45
HEA	0,97	0,97	0,95	0,96	7.425,03	10,10

Kết quả trên cho thấy, các thuật toán học máy có kết quả tốt trên bộ dữ liệu mới nhưng thấp hơn so với kết quả đánh giá trên bộ dữ liệu UMUDGA, trong đó mô hình đạt kết quả thấp nhất là RF với A.F_1-score là 0,14, mô hình đạt kết quả cao nhất là NN, VEA và HEA với A.F_1-score đạt 0,96. Hầu hết các mô hình đều có sự cân bằng giữa A.Precision và A.Recall, trừ hai mô hình RF và k-NN cho sự khác biệt khác lớn giữa A.Precision và A.Recall.

Xem xét về thời gian, hầu hết các mô hình đều có thời gian huấn luyện chiếm tỉ trọng lớn trong toàn bộ thời gian thử nghiệm, ngoại trừ k-NN. Mô hình NB cho tổng thời gian huấn luyện và đánh giá nhanh nhất với 1,48 giây, và chậm nhất là mô hình k-NN với 19.463,27 giây cho việc đánh giá. Lưu ý rằng, việc huấn luyện trên CPU đòi hỏi rất nhiều thời gian và không tận dụng được năng lực của GPU, đây cũng là một hạn chế so với các mô hình học sâu.

4.3.2.3. Thử nghiệm với hai mô hình LA_Bin07 và LA_Mul07

Hai mô hình học sâu LA_Bin07 và LA_Mul07 lần lượt được áp dụng cho bài toán phát hiện và phân loại, được đề xuất và trình bày tại Chương 3 của luận án này. NCS sử dụng 1.000.000 tên miền lành tính và 76.000 tên miền của DGA Botnet tương ứng với 76 họ DGA Bonet, mỗi họ có 10.000 tên miền..

- Kết quả đánh giá độ chính xác của mô hình LA_Bin07 trên bộ dữ liệu UTL_DGA22 cho bài toán phát hiện được thể hiện tại Bảng 4.14:

Bảng 4.14. Kết quả đánh giá mô hình LA_Bin07 trên bộ dữ liệu UTL_DGA22

Nhãn	**Precision**	**Recall**	**F_1-Score**
Lành tính	0,98	0,98	0,98
DGA Botnet	0,98	0,97	0,97
Accuracy	***0,98***		

Bảng trên cho thấy, mô hình LA_Bin07 có Accuracy cao đạt 0,98. Khả năng xác định các nhãn cũng đồng đều, không bị lệch về phía nào. Độ chính xác này cũng cao hơn so với hai giải pháp trước đó là NCM và học máy, chứng tỏ mô hình

LA_Bin07 vẫn đảm bảo tính hiệu quả và cải tiến độ chính xác như các đánh giá ở Chương 3.

- Kết quả thử nghiệm độ chính xác của mô hình LA_Mul07 trên bộ dữ liệu UTL_DGA22 trong bài toán phân loại được thể hiện tại Bảng 4.15:

Bảng 4.15. Kết quả đánh giá mô hình LA_Mul07 trên bộ dữ liệu UTL_DGA22

STT	DGA Botnet	Pre	Re	F_1	STT	DGA Botnet	Pre	Re	F_1
1	bamital	1,00	1,00	1,00	39	padcrypt	1,00	1,00	1,00
2	banjori	1,00	1,00	1,00	40	pandabanker	1,00	1,00	1,00
3	bazarbackdoor	1,00	1,00	1,00	41	pitou	1,00	1,00	1,00
4	bazarbackdoor_v2	1,00	1,00	1,00	42	pizd	0,92	0,97	0,95
5	bazarbackdoor_v3	1,00	1,00	1,00	43	proslikefan	0,79	0,62	0,70
6	bedep	0,71	0,67	0,69	44	pushdo	1,00	0,99	1,00
7	bigviktor	0,97	0,96	0,97	45	pykspa_ improved_noise	0,44	0,19	0,26
8	ccleaner	1,00	1,00	1,00	46	pykspa_ improved_useful	0,36	0,37	0,37
9	chinad	1,00	1,00	1,00	47	pykspa_precursor	0,99	0,99	0,99
10	corebot	1,00	1,00	1,00	48	qadars	1,00	0,99	1,00
11	cryptolocker	0,69	0,65	0,67	49	qakbot	0,83	0,48	0,61
12	dircrypt	0,50	0,13	0,20	50	qsnatch	1,00	1,00	1,00
13	dnschanger	0,41	0,84	0,55	51	ramdo	1,00	1,00	1,00
14	dyre	1,00	1,00	1,00	52	ramnit	0,37	0,56	0,44
15	emotet	1.00	1.00	1.00	53	ranbyus_v1	0,75	0,98	0,85
16	enviserv	1,00	1,00	1,00	54	ranbyus_v2	0,83	0,87	0,85
17	fobber_v1	0,88	1,00	0,94	55	reconyc	1,00	1,00	1,00
18	fobber_v2	0,44	0,17	0,25	56	rovnix	0,98	0,95	0,96
19	gozi_gpl	0,97	0,99	0,98	57	shiotob	1,00	0,91	0,95
20	gozi_luther	0,98	0,97	0,97	58	simda	1,00	1,00	1,00
21	gozi_nasa	0,94	0,97	0,95	59	sisron	1,00	1,00	1,00
22	gozi_rfc4343	0,93	0,95	0,94	60	sphinx	0,82	0,96	0,89
23	infy	1,00	1,00	1,00	61	suppobox_1	0,97	0,92	0,94
24	kraken_v1	0,90	0,44	0,59	62	suppobox_2	0,99	1,00	0,99
25	kraken_v2	0,58	0,66	0,62	63	suppobox_3	1,00	1,00	1,00

26	locky	0,85	0,63	0,72	64	symmi	1,00	1,00	1,00
27	matsnu	0,96	0,98	0,97	65	szribi	0,99	1,00	0,99
28	monerodownloader	1,00	1,00	1,00	66	tempedreve	0,59	0,75	0,66
29	murofetweekly	0,50	0,70	0,59	67	tinba	0,73	0,98	0,84
30	murofet_v1	0,95	0,96	0,95	68	tinynuke	1,00	1,00	1,00
31	murofet_v2	0,74	0,84	0,79	69	torpig	0,98	1,00	0,99
32	murofet_v3	0,48	0,29	0,36	70	vawtrak_v1	1,00	1,00	1,00
33	mydoom	1,00	1,00	1,00	71	vawtrak_v2	1,00	1,00	1,00
34	necurs	0,99	0,80	0,89	72	vawtrak_v3	1,00	1,00	1,00
35	newgoz	1,00	1,00	1,00	73	vidro	0,33	0,48	0,39
36	nymaim	0,50	0,84	0,63	74	virut	0,90	1,00	0,95
37	nymaim2	0,99	0,94	0,96	75	wd	1,00	1,00	1,00
38	oderoor	0,43	0,17	0,24	76	zloader	0,95	1,00	0,97
Accuracy		***0,86***							

Kết quả trên cho thấy, mô hình LA_Mul07 có Accuracy đạt 0,86 khi đánh giá trên bộ dữ liệu mới UTL_DGA22. Có 32 họ DGA Botnet được xác định nhãn đúng gần như tuyệt đối với F_1-score đạt 1,00. Một số họ DGA Botnet có khả năng phát hiện kém với F_1-score thấp như dircrypt (0,20), fobber_v2 (0,25), murofet_v3 (0,36), oderoor (0,24), pykspa_improved_noise (0,26), pykspa_improved_useful (0,37), ramnit (0,44), vidro (0,39). Các họ DGA Botnet còn lại nhìn chung cho kết quả phân loại khá tốt.

4.4. Kết luận Chương 4

Trong Chương 4, NCS trình bày đề xuất bổ sung hoàn thiện quy trình gồm 07 bước để xây dựng bộ dữ liệu và áp dụng xây dựng một bộ dữ liệu mới UTL_DGA22, trên cơ sở kế thừa các thành quả trước đó, bổ sung những cải tiến, dữ liệu và thuộc tính mới. Bộ dữ liệu mới đáp ứng 06 tiêu chí do NCS đặt ra và 09 tiêu chí theo quan điểm của Zago và cộng sự.

NCS mong muốn đóng góp bộ dữ liệu UTL_DGA22 tới các nhà khoa học để đánh giá hiệu năng các giải pháp đề xuất mới một cách thuận lợi, khách quan và dễ dàng đối sánh, tham chiếu.

Một phần kết quả trình bày tại Chương 4 được công bố tại [CT5] trong Danh mục các công trình công bố liên quan đến luận án.

KẾT LUẬN

Luận án "*Nghiên cứu cải tiến một số mô hình học máy và học sâu áp dụng cho bài toán phân loại DGA Botnet*" được hoàn thành tại Học viện Khoa học và Công nghệ, Viện Hàn lâm Khoa học và Công nghệ Việt Nam, với hai đóng góp bao gồm:

1. Đề xuất cải tiến kiến trúc lõi kết hợp BiLSTM với cơ chế Attention và sử dụng trong xây dựng mô hình LA_Bin07 để phát hiện và mô hình LA_Mul07 để phân loại DGA Botnet với độ chính xác được cải thiện.

2. Hoàn thiện bổ sung quy trình xây dựng tập dữ liệu mẫu và đề xuất bộ dữ liệu chuyên dùng UTL_DGA22 được mô tả và gán nhãn, phục vụ phân loại DGA Botnet.

Trả lời câu hỏi nghiên cứu đặt ra ban đầu: Kiến trúc lõi BiLSTM_SelfA_Double đã cải tiến được so với các kiến trúc trước đó, thể hiện qua độ chính xác được nâng cao của mô hình LA_Mul07 trong bài toán phân loại DGA Botnet.

Bên cạnh những kết quả đạt được, NCS dự kiến một số hướng phát triển trong thời gian tới, cụ thể như sau:

- Áp dụng mạng TCN để đề xuất kiến trúc học sâu mới đạt độ chính xác cao hơn trong phân loại.

- Xây dựng cơ chế huấn luyện dành riêng cho các họ DGA Botnet có sự tương đồng cao hoặc là các phiên bản kế tiếp của nhau.

Giải pháp đề xuất có đóng vai trò như một module phát hiện và phân loại DGA Botnet, có thể được tích hợp vào các giải pháp đảm bảo an ninh mạng như Tường lửa hoặc Giải pháp an ninh hợp nhất.

DANH MỤC CÁC CÔNG TRÌNH CÔNG BỐ LIÊN QUAN ĐẾN LUẬN ÁN

[CT1] Can, N.V., Tu, D. N., **Tuan, T. A.**, Long, H. V., Son, L. H., & Son, N. T. K. (2020). A new method to classify malicious domain name using Neutrosophic sets in DGA Botnet detection. *Journal of Intelligent & Fuzzy Systems*, *38*(4), 4223-4236. ***(ISI Q2, IF = 1.737)***

[CT2] **Tuan, T. A.**, Long, H. V., Son, L. H., Kumar, R., Priyadarshini, I., & Son, N. T. K. (2020). Performance evaluation of Botnet DDoS attack detection using machine learning. *Evolutionary Intelligence*, *13*(2), 283-294. ***(SCOPUS, ESCI Q2)***

[CT3] **Tuan, T. A.**, Anh, N. V., & Long, H. V. (2021, December). Assessment of Machine Learning Models in Detecting DGA Botnet in Characteristics by TF-IDF. In *2021 IEEE International Conference on Machine Learning and Applied Network Technologies (ICMLANT)* (pp. 1-5). IEEE. ***(SCOPUS)***

[CT4] **Tuan, T. A.**, Long, H. V., & Taniar, D. (2022). On Detecting and Classifying DGA Botnets and their Families. *Computers & Security*, *113*, 102549. ***(ISI Q1, IF = 5.105)***

[CT5] **Tuan, T. A.,** Anh, N. V., Luong, T. T., & Long, H. V. (2023). UTL_DGA22-a dataset for DGA botnet detection and classification. *Computer Networks*, 221, 109508. ***(ISI Q1, IF = 5.493)***

[CT6] **Tống Anh Tuấn**, Nguyễn Ngọc Cương, Nguyễn Việt Anh, Hoàng Việt Long. (2022). Đề xuất ứng dụng giải pháp phân lớp nhị phân trong bài toán DGA Botnet cho phát hiện địa chỉ IP độc hại. Hội thảo Quốc gia lần thứ XXV *"Một số vấn đề chọn lọc của Công nghệ thông tin và Truyền thông*" (VNICT 2022), trang 55-60.

TÀI LIỆU THAM KHẢO

[1] X. Li, C., Jiang, W., & Zou, "Botnet: Survey and case study. In innovative computing, information and control (icicic)," pp. 1187–1187, 2009.

[2] I. Ghafir, M. Hammoudeh, and V. Prenosil, "Botnet Command and Control Traffic Detection Challenges A Correlation based Solution," pp. 1–5, 2016, doi: 10.15224/978-1-63248-113-9-01.

[3] R. Kishore Kumar, G. Poonkuzhali, and P. Sudhakar, "Comparative study on email spam classifier using data mining techniques," *Lect. Notes Eng. Comput. Sci.*, vol. 2195, pp. 539–544, 2012.

[4] M. Roopak, G. Y. Tian, and J. Chambers, "Multi-objective-based feature selection for DDoS attack detection in IoT networks," *IET Networks*, vol. 9, no. 3, pp. 120–127, 2020, doi: 10.1049/iet-net.2018.5206.

[5] A. Karim, R. Bin Salleh, M. Shiraz, S. A. A. Shah, I. Awan, and N. B. Anuar, "Botnet detection techniques: review, future trends, and issues," *J. Zhejiang Univ. Sci. C*, vol. 15, no. 11, pp. 943–983, 2014, doi: 10.1631/jzus.C1300242.

[6] K. Alieyan, A. Almomani, A. Manasrah, and M. M. Kadhum, "A survey of botnet detection based on DNS," *Neural Comput. Appl.*, vol. 28, no. 7, pp. 1541–1558, 2017, doi: 10.1007/s00521-015-2128-0.

[7] J. Kwon, J. Lee, H. Lee, and A. Perrig, "PsyBoG: A scalable botnet detection method for large-scale DNS traffic," *Comput. Networks*, vol. 97, pp. 48–73, 2016, doi: 10.1016/j.comnet.2015.12.008.

[8] T. S. Wang, H. T. Lin, W. T. Cheng, and C. Y. Chen, "DBod: Clustering and detecting DGA-based botnets using DNS traffic analysis," *Comput. Secur.*, vol. 64, pp. 1–15, 2017, doi: 10.1016/j.cose.2016.10.001.

[9] F. Bisio, S. Saeli, P. Lombardo, D. Bernardi, A. Perotti, and D. Massa, "Real-time behavioral DGA detection through machine learning," *Proc. - Int. Carnahan Conf. Secur. Technol.*, vol. 2017-Octob, pp. 1–6, 2017, doi: 10.1109/CCST.2017.8167790.

[10] H. T. Nguyen, Q. D. Ngo, and V. H. Le, "A novel graph-based approach for IoT botnet detection," *Int. J. Inf. Secur.*, vol. 19, no. 5, pp. 567–577, 2020, doi: 10.1007/s10207-019-00475-6.

[11] H. Mac, D. Tran, V. Tong, L. G. Nguyen, and H. A. Tran, "DGA botnet detection using supervised learning methods," *ACM Int. Conf. Proceeding Ser.*, vol. 2017-Decem, pp. 211–218, 2017, doi: 10.1145/3155133.3155166.

[12] R. U. Khan, X. Zhang, R. Kumar, A. Sharif, N. A. Golilarz, and M. Alazab, "An adaptive multi-layer botnet detection technique using machine learning classifiers," *Appl. Sci.*, vol. 9, no. 11, 2019, doi: 10.3390/app9112375.

[13] M. Zago, M. Gil Pérez, and G. Martínez Pérez, "UMUDGA: A dataset for profiling algorithmically generated domain names in botnet detection," *Data Br.*, vol. 30, p. 105400, 2020, doi: 10.1016/j.dib.2020.105400.

[14] X. D. Hoang and X. H. Vu, "An improved model for detecting DGA botnets using random forest algorithm," *Inf. Secur. J.*, vol. 31, no. 4, pp. 441–450, 2022, doi: 10.1080/19393555.2021.1934198.

[15] D. Tran, H. Mac, V. Tong, H. A. Tran, and L. G. Nguyen, "A LSTM based framework for handling multiclass imbalance in DGA botnet detection," *Neurocomputing*, vol.

275, pp. 2401–2413, 2018, doi: 10.1016/j.neucom.2017.11.018.

[16] R. R. Curtin, A. B. Gardner, S. Grzonkowski, A. Kleymenov, and A. Mosquera, "Detecting DGA domains with recurrent neural networks and side information," *ACM Int. Conf. Proceeding Ser.*, 2019, doi: 10.1145/3339252.3339258.

[17] Y. Qiao, B. Zhang, W. Zhang, A. K. Sangaiah, and H. Wu, "DGA domain name classification method based on Long Short-Term Memory with attention mechanism," *Appl. Sci.*, vol. 9, no. 20, 2019, doi: 10.3390/app9204205.

[18] H. Vranken and H. Alizadeh, "Detection of DGA-Generated Domain Names with TF-IDF," *Electron.*, vol. 11, no. 3, pp. 1–28, 2022, doi: 10.3390/electronics11030414.

[19] J. Namgung, S. Son, and Y. S. Moon, "Efficient Deep Learning Models for DGA Domain Detection," *Secur. Commun. Networks*, vol. 2021, 2021, doi: 10.1155/2021/8887881.

[20] R. Vinayakumar, M. Alazab, S. Srinivasan, Q. V. Pham, S. K. Padannayil, and K. Simran, "A Visualized Botnet Detection System Based Deep Learning for the Internet of Things Networks of Smart Cities," *IEEE Trans. Ind. Appl.*, vol. 56, no. 4, pp. 4436–4456, 2020, doi: 10.1109/TIA.2020.2971952.

[21] T. Holz, "A short visit to the bot zoo [malicious bots software]," *IEEE Secur. Priv.*, vol. 3, no. 3, pp. 76–79, 2005.

[22] N. Provos and T. Holz, "Virtual honeypots: from botnet tracking to intrusion detection," p. 440, 2007, [Online]. Available: http://books.google.com/books?id=QuHnPgAACAAJ&pgis=1

[23] A. Kurniawan and A. Fitriansyah, "A Literature Review of Historical and Detection Analysis of Botnets Forensics," *Int. J. Comput. Commun. Eng.*, vol. 7, no. 4, pp. 128–135, 2018, doi: 10.17706/ijcce.2018.7.4.128-135.

[24] A. C. Atluri and V. Tran, "Botnets threat analysis and detection," *Inf. Secur. Pract. Emerg. Threat. Perspect.*, pp. 7–28, 2017, doi: 10.1007/978-3-319-48947-6_2.

[25] C. Li, W. Jiang, and X. Zou, "Botnet: Survey and case study," *2009 4th Int. Conf. Innov. Comput. Inf. Control. ICICIC 2009*, pp. 1184–1187, 2009, doi: 10.1109/ICICIC.2009.127.

[26] N. Manzoor, M. Saleem, and M. Aslam, "Role of Machine Learning Techniques in Digital Forensic Investigation of Botnet Attacks," *Int. J. Manag.*, 2021.

[27] C. A. Schiller *et al.*, "Botnets: The Killer Web Applications," *Botnets Kill. Web Appl.*, pp. 1–464, 2007, doi: 10.1016/B978-1-59749-135-8.X5000-8.

[28] K. Vengatesan, A. Kumar, M. Parthibhan, A. Singhal, and R. Rajesh, "Analysis of Mirai Botnet Malware Issues and Its Prediction Methods in Internet of Things," *Lect. Notes Data Eng. Commun. Technol.*, vol. 31, pp. 120–126, 2020, doi: 10.1007/978-3-030-24643-3_13.

[29] J. Nazario, "Bot and Botnet Taxonomy," *Comput. Secur. Institute. Comput. Secur. Inst. Secur. Exch.*, p. 52, 2008, [Online]. Available: https://www.monkey.org/~jose/presentations/csisx2008.d/CSI SX 2008 Nazario Botnet Taxonomy.pdf

[30] N. V. Patil, C. Rama Krishna, and K. Kumar, "Distributed frameworks for detecting distributed denial of service attacks: A comprehensive review, challenges and future directions," *Concurr. Comput.* , 2021, doi: 10.1002/cpe.6197.

[31] L. Barry, D., & Bol, "Who's Hacking Who?," 2016.

[32] D. S. & S. S. A. B. Tickle, E. Ahmed, S. M. Bhaskar, G. Mohay, S. Panichprecha, S. V. Raghavan, B. Ravindran, “Chapter 2: Background,” in *An Investigation into the Detection and Mitigation of Denial of Service (DoS) Attacks*, 2011. doi: 10.1007/978-81-322-0277-6.

[33] E. B. Beigi, H. H. Jazi, N. Stakhanova, and A. A. Ghorbani, “Towards effective feature selection in machine learning-based botnet detection approaches,” *2014 IEEE Conf. Commun. Netw. Secur. CNS 2014*, pp. 247–255, 2014, doi: 10.1109/CNS.2014.6997492.

[34] S. Marchal, J. Francois, R. State, and T. Engel, “Phish storm: Detecting phishing with streaming analytics,” *IEEE Trans. Netw. Serv. Manag.*, vol. 11, no. 4, pp. 458–471, 2014, doi: 10.1109/TNSM.2014.2377295.

[35] A. Abakumov, “DGA Repository,” *GitHub*, 2016. https://github.com/andrewaeva/DGA (accessed Jun. 08, 2021).

[36] OSINT, “Feeds from Bambenek Consulting,” 2021. https://osint.bambenekconsulting.com/feeds/ (accessed Mar. 15, 2021).

[37] 360NetLab, “DGA - Netlab OpenData Project,” *Qihoo 360 Technology*, 2022. https://data.netlab.360.com/dga/ (accessed Mar. 09, 2021).

[38] S. Chowdhury *et al.*, “Botnet detection using graph-based feature clustering,” *J. Big Data*, vol. 4, no. 1, 2017, doi: 10.1186/s40537-017-0074-7.

[39] O. Yavanoglu and M. Aydos, “A review on cyber security datasets for machine learning algorithms,” *Proc. - 2017 IEEE Int. Conf. Big Data, Big Data 2017*, vol. 2018-Janua, pp. 2186–2193, 2017, doi: 10.1109/BigData.2017.8258167.

[40] J. Wang and I. C. Paschalidis, “Botnet Detection Based on Anomaly and Community Detection,” *IEEE Trans. Control Netw. Syst.*, vol. 4, no. 2, pp. 392–404, 2017, doi: 10.1109/TCNS.2016.2532804.

[41] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, “IoTPOT: A novel honeypot for revealing current IoT threats,” *J. Inf. Process.*, vol. 24, no. 3, pp. 522–533, 2016, doi: 10.2197/ipsjjip.24.522.

[42] “VirusShare.com - Because Sharing is Caring.” https://virusshare.com/ (accessed Mar. 15, 2021).

[43] Alexa Internet Inc., “Alexa top 1 million sites,” *Kaggle Datasets*, 2019. http://s3.amazonaws.com/alexa-static/top-1m.csv.zip (accessed Mar. 10, 2021).

[44] M. Zago, M. Gil Pérez, and G. Martínez Pérez, “UMUDGA: A dataset for profiling DGA-based botnet,” *Comput. Secur.*, vol. 92, 2020, doi: 10.1016/j.cose.2020.101719.

[45] H. Suryotrisongko and Y. Musashi, “Evaluating hybrid quantum-classical deep learning for cybersecurity botnet DGA detection,” *Procedia Comput. Sci.*, vol. 197, pp. 223–229, 2021, doi: 10.1016/j.procs.2021.12.135.

[46] D. Zhao, H. Li, X. Sun, and Y. Tang, “Detecting DGA-based botnets through effective phonics-based features,” *Futur. Gener. Comput. Syst.*, vol. 143, pp. 105–117, 2023, doi: 10.1016/j.future.2023.01.027.

[47] M. Alauthman, N. Aslam, M. Al-kasassbeh, S. Khan, A. Al-Qerem, and K. K. Raymond Choo, “An efficient reinforcement learning-based Botnet detection approach,” *J. Netw. Comput. Appl.*, vol. 150, p. 102479, 2020, doi: 10.1016/j.jnca.2019.102479.

[48] S. Saad *et al.*, “Detecting P2P botnets through network behavior analysis and machine

learning," *2011 9th Annu. Int. Conf. Privacy, Secur. Trust. PST 2011*, pp. 174–180, 2011, doi: 10.1109/PST.2011.5971980.

[49] B. Rahbarinia, R. Perdisci, A. Lanzi, and K. Li, "PeerRush: Mining for unwanted P2P traffic," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7967 LNCS, pp. 62–82, 2013, doi: 10.1007/978-3-642-39235-1_4.

[50] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, 2012, doi: 10.1016/j.cose.2011.12.012.

[51] X. Liu and J. Liu, "DGA botnet detection method based on capsule network and k-means routing," *Neural Comput. Appl.*, vol. 34, no. 11, pp. 8803–8821, 2022, doi: 10.1007/s00521-022-06904-3.

[52] S. Broumi *et al.*, "Neutrosophic Sets: An Overview," *New Trends Neutrosophic Theory Appl.*, vol. II, p. 32, 2018, [Online]. Available: http://fs.gallup.unm.edu/nss.

[53] Y. Guo and A. Sengur, "NCM: Neutrosophic c-means clustering algorithm," *Pattern Recognit.*, vol. 48, no. 8, pp. 2710–2724, 2015, doi: 10.1016/j.patcog.2015.02.018.

[54] K. N. S. S. V Prasad, S. K. Saritha, and D. Saxena, "A Survey Paper on Concept Mining in Text Documents," *Int. J. Comput. Appl.*, vol. 166, no. 11, pp. 7–10, 2017, doi: 10.5120/ijca2017914143.

[55] S. Hochreiter, "Lstm Can Solve Hard Long Time Lag Problems," *Adv. Neural Inf. Process. Syst.*, 1996.

[56] A. Vaswani *et al.*, "Attention is all you need," *Adv. Neural Inf. Process. Syst.*, vol. 2017-Decem, no. Nips, pp. 5999–6009, 2017, [Online]. Available: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

[57] B. Gao and L. Pavel, "On the properties of the softmax function with application in game theory and reinforcement learning," *arXiv*, 2017.

[58] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv*, 2018.

[59] R. Mehrotra, "A Detailed Guide to understand the Word Embeddings and Embedding Layer in Keras," *Kaggle*. https://www.kaggle.com/code/rajmehra03/a-detailed-explanation-of-keras-embedding-layer (accessed Aug. 10, 2023).

[60] Juhong-Namgung, "Malicious-URL-and-DGA-Domain-Detection-using-Deep-Learning." https://github.com/Juhong-Namgung/Malicious-URL-and-DGA-Domain-Detection-using-Deep-Learning (accessed Jul. 06, 2023).

[61] M. Antonakakis *et al.*, "From throw-away traffic to bots: Detecting the rise of DGA-based malware," *Proc. 21st USENIX Secur. Symp.*, pp. 491–506, 2012.

[62] Y.-L. Zhou, Q.-S. Li, Q. Miao, and K. Yim, "DGA-Based Botnet Detection Using DNS Traffic," *J. Internet Serv. Inf. ...*, vol. 3, no. 11, pp. 116–123, 2013, [Online]. Available: http://isyou.info/jisis/vol3/no34/jisis-2013-vol3-no34-11.pdf

[63] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel, "EXPOSURE: A passive DNS analysis service to detect and report malicious domains," *ACM Trans. Inf. Syst. Secur.*, vol. 16, no. 4, 2014, doi: 10.1145/2584679.

[64] T. D. Nguyen, T. D. Cao, and L. G. Nguyen, "DGA botnet detection using collaborative filtering and density-based clustering," *ACM Int. Conf. Proceeding Ser.*, vol. 03-04-Dece, pp. 203–209, 2015, doi: 10.1145/2833258.2833310.

[65] R. Sharifnya and M. Abadi, “DFBotKiller: Domain-flux botnet detection based on the history of group activities and failures in DNS traffic,” *Digit. Investig.*, vol. 12, pp. 15–26, 2015, doi: 10.1016/j.diin.2014.11.001.

[66] G. Bottazzi and G. F. Italiano, “Fast mining of large-scale logs for Botnet detection: A field study,” *Proc. - 15th IEEE Int. Conf. Comput. Inf. Technol. CIT 2015, 14th IEEE Int. Conf. Ubiquitous Comput. Commun. IUCC 2015, 13th IEEE Int. Conf. Dependable, Auton. Se*, pp. 1989–1996, 2015, doi: 10.1109/CIT/IUCC/DASC/PICOM.2015.295.

[67] M. J. Erquiaga, C. Catania, and S. García, “Detecting DGA malware traffic through behavioral models,” *2016 IEEE Bienn. Congr. Argentina, ARGENCON 2016*, 2016, doi: 10.1109/ARGENCON.2016.7585238.

[68] S. Garcia, “Stratoshpere Project,” *Https://Stratosphereips.Org*, 2015. https://stratosphereips.org

[69] M. I. Ashiq, P. Bhowmick, M. S. Hossain, and H. S. Narman, “Domain Flux-based DGA Botnet Detection Using Feedforward Neural Network,” *Proc. - IEEE Mil. Commun. Conf. MILCOM*, vol. 2019-Novem, pp. 1–6, 2019, doi: 10.1109/MILCOM47813.2019.9020730.

[70] Y. Fu *et al.*, “Stealthy Domain Generation Algorithms,” *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 6, pp. 1430–1443, 2017, doi: 10.1109/TIFS.2017.2668361.

[71] K. Alieyan, A. Almomani, M. Anbar, M. Alauthman, R. Abdullah, and B. B. Gupta, “DNS rule-based schema to botnet detection,” *Enterp. Inf. Syst.*, vol. 00, no. 00, pp. 1–20, 2019, doi: 10.1080/17517575.2019.1644673.

[72] X. Yun, J. Huang, Y. Wang, T. Zang, Y. Zhou, and Y. Zhang, “Khaos: An Adversarial Neural Network DGA with High Anti-Detection Ability,” *IEEE Trans. Inf. Forensics Secur.*, vol. 15, no. c, pp. 2225–2240, 2020, doi: 10.1109/TIFS.2019.2960647.

[73] H. S. Anderson, J. Woodbridge, and B. Filar, “DeepDGA: Adversarially-tuned domain generation and detection,” *AISec 2016 - Proc. 2016 ACM Work. Artif. Intell. Secur. co-located with CCS 2016*, pp. 13–21, 2016, doi: 10.1145/2996758.2996767.

[74] R. Rajalakshmi, S. Ramraj, and R. Ramesh Kannan, “Transfer learning approach for identification of malicious domain names,” *Commun. Comput. Inf. Sci.*, vol. 969, pp. 656–666, 2019, doi: 10.1007/978-981-13-5826-5_51.

[75] X. Pei, S. Tian, L. Yu, H. Wang, and Y. Peng, “A Two-Stream Network Based on Capsule Networks and Sliced Recurrent Neural Networks for DGA Botnet Detection,” *J. Netw. Syst. Manag.*, vol. 28, no. 4, pp. 1694–1721, 2020, doi: 10.1007/s10922-020-09554-9.

[76] S. García, M. Grill, J. Stiborek, and A. Zunino, “An empirical comparison of botnet detection methods,” *Comput. Secur.*, vol. 45, pp. 100–123, 2014, doi: 10.1016/j.cose.2014.05.011.

[77] G. Maciá-Fernández, J. Camacho, R. Magán-Carrión, P. García-Teodoro, and R. Therón, “UGR‘16: A new dataset for the evaluation of cyclostationarity-based network IDSs,” *Comput. Secur.*, vol. 73, pp. 411–424, 2018, doi: 10.1016/j.cose.2017.11.004.

[78] A. Venturi, G. Apruzzese, M. Andreolini, M. Colajanni, and M. Marchetti, “DReLAB - Deep REinforcement Learning Adversarial Botnet: A benchmark dataset for adversarial attacks against botnet Intrusion Detection Systems,” *Data Br.*, vol. 34, 2021, doi: 10.1016/j.dib.2020.106631.

[79] N. Moustafa and J. Slay, “UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set),” *2015 Mil. Commun. Inf. Syst. Conf. MilCIS 2015 - Proc.*, 2015, doi: 10.1109/MilCIS.2015.7348942.

[80] D. Zhao *et al.*, “Botnet detection based on traffic behavior analysis and flow intervals,” *Comput. Secur.*, vol. 39, no. PARTA, pp. 2–16, 2013, doi: 10.1016/j.cose.2013.04.007.

[81] S. Garcia, “Malware Capture Facility Project,” 2013. https://mcfp.felk.cvut.cz/ (accessed Jul. 25, 2022).

[82] H. Suryotrisongko, “Computable CTI: Sharing AI Model for the Next Level of Actionable Cyber Threat Intelligence. Case Study of Botnet Detection,” *IEEE Open Access J.*, 2020.

[83] H. Suryotrisongko, “Botnet DGA Dataset,” *IEEE Dataport*, 2020. https://ieee-dataport.org/open-access/botnet-dga-dataset (accessed Jun. 08, 2021).

[84] F. FKIE, “DGArchive - Fraunhofer FKIE,” 2020. https://dgarchive.caad.fkie.fraunhofer.de/welcome/ (accessed Jun. 08, 2021).

[85] J. Spooren, D. Preuveneers, L. Desmet, P. Janssen, and W. Joosen, “Detection of algorithmically generated domain names used by botnets: A dual arms race,” *Proc. ACM Symp. Appl. Comput.*, vol. Part F1477, pp. 1916–1923, 2019, doi: 10.1145/3297280.3297467.

[86] J. Bader, “Domain_Generation_Algorithms Repository,” *GitHub*, 2018. https://github.com/baderj/domain_generation_algorithms (accessed Aug. 16, 2021).

[87] N. Brown, “GNU GPL 2.0 and 3.0: obligations to include license text, and provide source code,” *Int. Free Open Source Softw. Law Rev.*, vol. 2, no. 1, 2010, doi: 10.5033/ifosslr.v2i1.31.

[88] J. Bader, “Johannes Bader’s Blog.” https://johannesbader.ch/blog/ (accessed Aug. 16, 2021).

[89] Majestic, “Majestic Million - Majestic,” *Majestic Website*, 2019. https://majestic.com/reports/majestic-million

[90] L. Lessig, “Creative Commons - attribution 3.0 unported license,” 2001.

[91] “Tinba’s DGA Adds Other Top Level Domains,” *Johannes Bader’s Blog*, 2015. https://bin.re/blog/new-top-level-domains-for-tinbas-dga/ (accessed Oct. 11, 2021).