Phương pháp phát hiện và phân loại DGA Botnet dựa trên Học máy

1 VẤN ĐỀ BÀI BÁO

Trong nghiên cứu này, chúng tôi nhắm đến việc giải quyết hai vấn đề quan trọng nhất trong việc phát hiện botnet, bao gồm:

- Vấn đề phân loại nhị phân: Với dữ liệu đầu vào của các truy vấn tên miền, vấn đề này sẽ phân loại các tên miền thành hai lớp, bao gồm tên miền vô hại và tên miền độc hại.
- Vấn đề phân loại đa lớp: Với dữ liệu đầu vào của các truy vấn tên miền đã được đánh dấu độc hại, vấn đề này sẽ xác định xem tên miền thuộc về gia đình botnet DGA nào.

Kết quả nhận diện đó có thể hỗ trợ quét và loại bỏ botnet từ các máy tính bị nhiễm bệnh một cách tốt hơn.

Bài báo này [TLT22] đề cập đến mối đe dọa đáng kể từ botnet đối với an ninh internet bằng cách đề xuất các giải pháp phát hiện và phân loại mới lạ nhằm mục tiêu cụ thể là botnet sử dụng thuật toán tạo tên miền (DGA). Bằng cách tận dụng mạng LSTM và các lớp Attention, các tác giả giới thiệu hai mô hình học sâu, LA_Bin07 và LA_Mul07, hiệu quả đối phó với thách thức phân loại nhị phân và đa lớp. Đánh giá trên tập dữ liệu AADR, bao gồm 8 gia đình botnet DGA, thể hiện độ chính xác cao của các mô hình trong việc xác định các miền độc hại và phân loại các gia đình botnet.

2 CƠ SỞ LÝ THUYẾT

2.1 Botnet

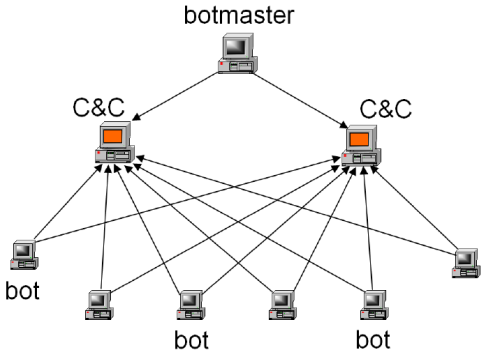
2.1.1 Khái niệm.

Botnet là một khái niệm đại diện cho một mạng máy tính bị tin tặc chiếm quyền trái phép, với quy mô có thể từ hàng ngàn máy tính trong một botnet cụ thể. Botnet được quản lý bởi Bot Master và có thể thực hiện các cuộc tấn công mạng.

2.1.2 Kiến trúc.

Botnet có thể được triển khai dựa trên hai mô hình: Peer- to-Peer và Client-Server. Một botnet thường bao gồm: BotMaster, máy chủ Command & Control (C&C) và mạng lưới bot được minh họa trong Hình 1.

- BotMaster: Tin tặc phân phối bot và điều khiển chúng thông qua máy chủ C&C.
- Máy chủ C&C: Các máy chủ đóng vai trò trung gian giữa BotMaster và các bot. Chúng có thể quản lý bot, nhận lệnh từ BotMaster để phát tán mã độc, chuyển tiếp lệnh để tiến hành tấn công, hoặc phân phối các bản cập nhật mới.
- Bots: Máy tính hoặc thiết bị thông minh có kết nối Internet. Các thiết bị này bị nhiễm mã độc. BotMaster quản lý chúng thông qua máy chủ C&C. Chúng thực hiện các hoạt động tiêu chuẩn như ẩn nấp trong hệ thống, nhận lệnh, hoặc nhận cập nhật từ máy chủ C&C.



Hình 1: Kiến trúc của Botnet

2.1.3 Hành vi của Botnet.

Botnet được sử dụng cho nhiều mục đích:

- Tấn công từ chối dịch vụ (DDoS): Tấn công các máy chủ hoặc cá nhân trên Internet, gây thiệt hại cho hệ thống.
- Gửi tin nhắn hoặc email spam: Gây lãng phí tài nguyên mạng và đe dọa thông tin cá nhân người dùng.
- Trộm cắp thông tin và gián điệp: Thu thập thông tin tình báo, ghi lại hành vi người dùng, đánh cắp thông tin cá nhân.
- Hành vi tự động trên Internet: Giả mạo người dùng để thực hiện các hành động tự động, như nhấp quảng cáo hoặc bình luận tự động.

Botnet gây nguy hiểm cho Internet và mang lại hàng triệu đô la mỗi tháng cho tin tặc. Phần mềm độc hại đã gây thiệt hại 13.2 tỷ USD cho kinh tế thế giới nuam 2006. Tin tặc thuê botnet để thực hiện hoạt động bất hợp pháp với chi phí 7 USD/giờ và cho thuê lại với giá 25 USD/giờ. Việc phát hiện và chặn botnet là ưu tiên hàng đầu của chuyên gia an ninh mạng.

2.2 DGA botnets

2.2.1 DGA (Domain Generation Algorithm - Thuật toán tạo miền) .

Là một kỹ thuật được sử dụng để sinh và đăng ký nhiều tên miền ngẫu nhiên khác nhau. Kỹ thuật này được áp dụng để tạo ra một lượng lớn tên miền giả định kỳ, mà sau đó các tên miền này được phân giải thành địa chỉ IP của máy chủ chỉ huy và điều khiển (C&C) của các hệ thống mạng botnet.

Mục đích chính của việc sử dụng DGA là để phức tạp hóa quá trình kiểm soát và thu hồi tên miền, từ đó giúp các hệ thống botnet tránh được sự phát hiện và ngăn chặn của các cơ quan chức năng và các hệ thống bảo mật.

Nếu một botnet sử dụng một tên miền tĩnh cho máy chủ C&C của nó, việc kiểm soát và thu hồi tên miền có thể được thực hiện một cách dễ dàng thông qua việc phối hợp với bên

quản lý tên miền gốc để chỉnh sửa các bản ghi tên miền trên máy chủ DNS. Tuy nhiên, khi DGA được sử dụng để sinh các tên miền động, việc kiểm soát và thu hồi các tên miền sẽ trở nên rất khó khăn. Điều này là do các bot thường xuyên thay đổi tên miền, sử dụng các tên miền mới được tạo ra sau một khoảng thời gian nhất định để kết nối đến máy chủ C&C, khiến cho việc kiểm soát các tên miền đã hết hạn sử dụng trở nên không có ý nghĩa.

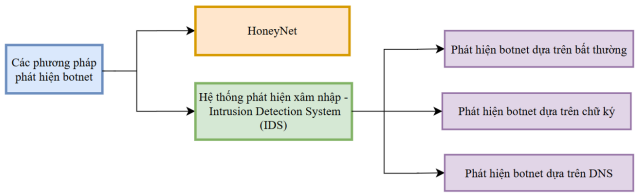
2.2.2 DGA botnet.

Hay còn gọi là DGA-based botnet, là các họ botnet sử dụng kỹ thuật DGA để sinh và đăng ký nhiều tên miền ngẫu nhiên khác nhau cho máy chủ chỉ huy và điều khiển C&C của chúng.

Mục tiêu chính của các botnet dạng này là chống lại việc bị kiểm soát và đưa vào danh sách đen. Các botnet này sử dụng thuật toán DGA để định kỳ sinh và đăng ký một lượng lớn tên miền giả ngẫu nhiên. Việc này giúp các botnet duy trì sự ẩn danh và khả năng hoạt động liên tục mà không bị gián đoạn. Bằng cách thay đổi liên tục tên miền kết nối đến máy chủ C&C, các DGA botnet làm cho việc phát hiện và ngăn chặn trở nên khó khăn hơn rất nhiều. Điều này đặc biệt quan trọng trong bối cảnh các biện pháp bảo mật ngày càng trở nên tiên tiến và chặt chẽ hơn.

Khi các tên miền được tạo ra một cách động và thay đổi thường xuyên, việc phát hiện và ngăn chặn các botnet này đòi hỏi các giải pháp bảo mật phải có khả năng dự đoán và phản ứng nhanh chóng trước những thay đổi liên tục này. Do đó, DGA botnet có thể duy trì hoạt động lâu dài và gây ra nhiều thiệt hại trước khi bị phát hiện và ngăn chặn hoàn toàn.

2.3 Phương pháp phát hiện botnet



Hình 2: Phương pháp phát hiện botnet

Phát hiện botnet dựa trên Honeynet là phương pháp sử dụng Honeynet để thu hút và phát hiện botnet. Honeynet là một mạng giả được thiết kế để mô phỏng các hệ thống mạng thực tế. Khi botnet quét internet để tìm kiếm các mục tiêu tiềm năng, nó có thể bị thu hút bởi Honeynet. Khi botnet tấn công Honeynet, nó sẽ bị phát hiện và có thể được nghiên cứu để tìm hiểu thêm về hoạt động của nó.

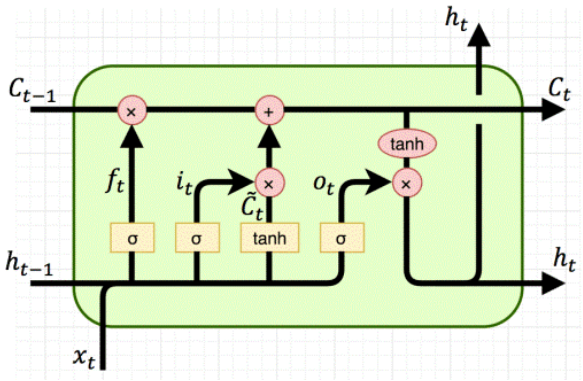
IDS là một hệ thống giám sát mạng để phát hiện các hoạt động bất thường có thể là dấu hiệu của tấn công mạng. IDS có thể được sử dụng để phát hiện botnet bằng cách theo dõi các mẫu lưu lượng mạng và hành vi của botnet.

- Phát hiện botnet dựa trên bất thường là phương pháp sử dụng các kỹ thuật thống kê và học máy để xác định các mẫu lưu lượng mạng bất thường có thể là dấu hiệu của botnet.
- Phát hiện botnet dựa trên chữ ký là phương pháp sử dụng các chữ ký để xác định các botnet đã được biết đến. Chữ ký có thể bao gồm các mẫu lưu lượng mạng, mã độc hại hoặc hành vi của botnet.
- Phát hiện botnet dựa trên DNS là phương pháp sử dụng các bản ghi DNS để xác định các botnet. Botnet thường sử dụng các tên miền đã được biết đến để giao tiếp với nhau.

Trong nghiên cứu này, nhóm nghiên cứu áp dụng phương pháp phát hiện botnet dựa trên phân tích DNS nhằm phát hiện và phân loại các kết nối DNS độc hại. Phương pháp này có ưu điểm là có thể ngăn chặn botnet giao tiếp với máy chủ điều khiển và chỉ huy (C&C). Từ đó, có thể vô hiệu hóa hiệu quả các botnet, ngay cả khi chúng đã lây nhiễm vào máy tính. Bên cạnh đó, phân tích DNS yêu cầu ít tài nguyên tính toán hơn so với các giải pháp khác.

2.4 Mạng Long-Short Term Memory (LSTM)

Mạng Long-Short Term Memory (LSTM) được phát triển từ mạng RNN, với khả năng học các phụ thuộc xa hơn so với RNN (Du và Swamy, 2014) [DS14]. LSTM lần đầu tiên được đề xuất vào năm 1997 bởi Hochreiter (1997) [HS97] và đã liên tục được cải tiến kể từ đó. Thuật toán này hoạt động hiệu quả trên nhiều vấn đề khác nhau, đặc biệt là các vấn đề chuỗi.



Hình 3: Kiến trúc 4 tầng của mô-đun LSTM

Các thành phần của mô hình LSTM

- Cell State (Trạng thái ô nhớ) – C_t: Đây là thành phần chính của LSTM, giữ thông tin qua các bước thời gian. Nó có thể được cập nhật hoặc giữ nguyên qua các bước thời gian thông qua các cổng khác nhau.
- Hidden State (Trạng thái ẩn) - h_t: Trạng thái ẩn được sử dụng để truyền thông tin từ bước thời gian hiện tại sang bước thời gian tiếp theo và ra ngoài mạng LSTM.
- Input Gate (Cổng đầu vào) - h_t: Cổng này quyết định thông tin nào từ đầu vào hiện tại x_t và trạng thái ẩn trước đó h_{t-1} sẽ được sử dụng để cập nhật trạng thái ô nhớ. Cổng này sử dụng một hàm kích hoạt sigmoid σ.

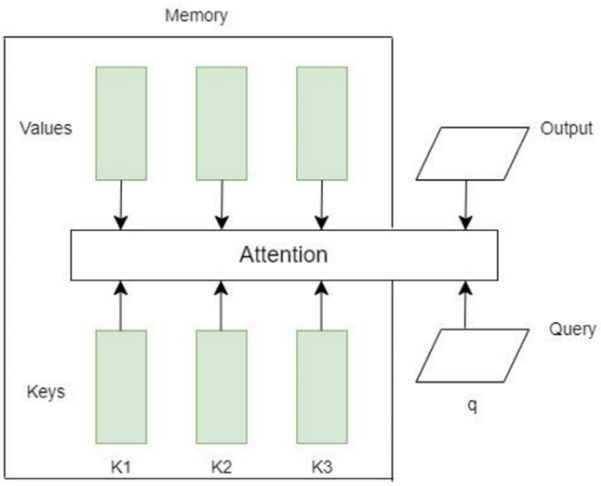
- Forget Gate (Cổng quên) - f_t: Cổng này quyết định thông tin nào từ trạng thái ô nhớ trước đó C_{t-1} sẽ bị loại bỏ. Cổng này cũng sử dụng một hàm kích hoạt sigmoid σ.
- Output Gate (Cổng đầu ra) - o_t: Cổng này quyết định thông tin nào từ trạng thái ô nhớ hiện tại sẽ được sử dụng để tạo trạng thái ẩn h_t. Cổng này sử dụng một hàm kích hoạt sigmoid σ.
- Candidate Cell State (Trạng thái ô nhớ ứng viên) –$\tilde{C}_t$: Trạng thái này được tạo ra bởi một hàm kích hoạt tanh và được sử dụng để cập nhật trạng thái ô nhớ.

2.5 Attention layer

Attention Layer: là một cơ chế được sử dụng trong các mô hình học sâu, đặc biệt là trong các mô hình xử lý ngôn ngữ tự nhiên (Neuro-Linguistic Programming – NLP) như Transformer, để cải thiện hiệu suất và khả năng xử lý thông tin. Lớp chú ý cho phép mô hình tập trung vào các phần quan trọng của đầu vào, giúp cải thiện khả năng hiểu và xử lý các dữ liệu có ngữ cảnh dài hoặc phức tạp.

2.5.1 Các thành phần chính của Attention Layer.

Attention Layer thường được mô tả bởi ba thành phần chính như trong hình 4:



Hình 4: Kiến trúc của Attention Layer

- Query (Truy vấn): Đây là một vector đại diện cho yếu tố mà mô hình đang cố gắng tìm kiếm thông tin liên quan từ các vector đầu vào khác.
- Key (Khóa): Đây là các vector đại diện cho các đặc trưng của các phần tử trong đầu vào. Các vector này sẽ được so sánh với vector truy vấn để xác định mức độ liên quan.
- Value (Giá trị): Đây là các vector chứa thông tin thực tế của các phần tử trong đầu vào mà mô hình cần tập trung vào. Các giá trị này sẽ được trọng số hóa dựa trên mức độ liên quan giữa truy vấn và khóa.

2.5.2 Cách thức hoạt động của Attention Layer.

Quy trình của Attention Layer bao gồm các bước sau:

- Tính điểm tương đồng: Tính điểm tương đồng giữa vector truy vấn và mỗi vector khóa để xác định mức độ liên quan của từng phần tử trong đầu vào. Phép tính này thường sử dụng sản phẩm vô hướng (dot product).
- Tính trọng số chú ý: Áp dụng hàm softmax lên các điểm tương đồng để chuyển đổi chúng thành các trọng số chú ý. Các trọng số này biểu thị xác suất mà mỗi phần tử trong đầu vào có liên quan đến truy vấn.
- Tính giá trị chú ý: Trọng số hóa các vector giá trị bằng các trọng số chú ý đã tính được, sau đó tính tổng để tạo ra vector đầu ra cuối cùng.

2.5.3 Lợi ích của Attention Layer.

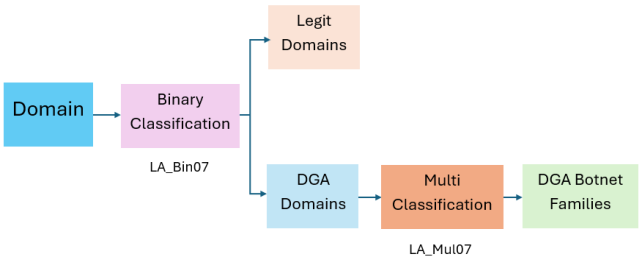
- Khả năng xử lý ngữ cảnh dài: Attention Layer cho phép mô hình tập trung vào các phần quan trọng của đầu vào bất kể khoảng cách vị trí, giúp mô hình xử lý tốt hơn các ngữ cảnh dài.
- Tăng cường hiệu suất: Attention Layer giúp mô hình học cách tập trung vào các phần tử quan trọng, cải thiện hiệu suất của các tác vụ như dịch máy, tóm tắt văn bản, và trả lời câu hỏi.
- Linh hoạt và mở rộng: Cơ chế chú ý có thể dễ dàng tích hợp vào nhiều kiến trúc mạng khác nhau như RNN, LSTM, và đặc biệt là Transformer, mang lại sự linh hoạt và khả năng mở rộng cho các ứng dụng khác nhau.

3 PHÂN TÍCH THIẾT KẾ HỆ THỐNG

Hai mô hình LA_Bin07 và LA_Mul07 được áp dụng vào vấn đề phân loại nhị phân và phân loại đa lớp, tương ứng, với các bước được minh họa trong Hình 5.

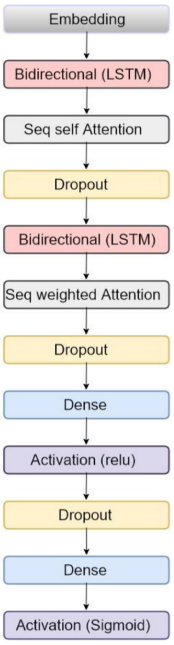
Trước hết, các tên miền truy vấn sẽ được đưa vào một mô hình phân loại nhị phân đã được huấn luyện, gán nhãn là độc hại hoặc vô hại. Đối với các tên miền vô hại, việc truy cập được phép thường diễn ra trong thực tế.

Các tên miền được gán nhãn là độc hại sau đó được thông qua mô hình phân loại đa lớp để xác định chính xác họ tên của các botnet DGA của chúng.



Hình 5: Các bước trong việc phát hiện DGA botnet

3.1 LA_Bin07



Hình 6: Mô hình kiến trúc của LA_Bin07

Mô hình được thiết kế ở trên với hai khối LSTM kết hợp với lớp Attention, hoạt động trong một mô hình tuần tự. Theo sơ đồ được cung cấp trong hình 6, mô hình kiến trúc của LA_Bin07 bao gồm các lớp sau:

(1) Embedding
Lớp này có nhiệm vụ chuyển đổi các từ đầu vào thành các vector biểu diễn. Các vector này được gọi là embedding của các từ. Embedding giúp mô hình học được mối quan hệ ngữ nghĩa giữa các từ.

(2) Các lớp Bidirectional LSTM
Lớp này bao gồm hai mạng LSTM chạy ngược chiều nhau. Mạng LSTM đầu tiên xử lý chuỗi đầu vào theo thứ tự từ đầu đến cuối, trong khi mạng LSTM thứ hai xử lý chuỗi đầu vào theo thứ tự từ cuối đến đầu. Cả hai mạng LSTM đều học được các mối quan hệ phụ thuộc giữa các từ trong chuỗi.

(3) Seq Self Attention
Lớp này sử dụng cơ chế tự học của attention để tập trung vào các phần quan trọng của chuỗi bằng cách tính toán trọng số chú ý cho từng phần tử trong chuỗi dựa trên chính chuỗi đó.

(4) Các lớp Dropout
Ba lớp dropout giúp mô hình tránh overfitting bằng cách ngẫu nhiên loại bỏ một số phần tử trong đầu ra của các lớp trước đó. Điều này giúp giảm thời gian huấn luyện và đồng thời giảm nguy cơ mô hình học thuộc và phụ thuộc quá mức vào dữ liệu.

(5) Seq Weighted Attention
Lớp này tính toán trọng số cho mỗi phần tử trong chuỗi dựa trên các đặc trưng cụ thể của chuỗi. Trọng số này được sử dụng để tạo ra một tổ hợp tuyến tính của các phần tử trong chuỗi, tập trung vào các phần tử quan trọng nhất.
SeqSelfAttention nắm bắt mối quan hệ phức tạp và ngữ cảnh dài hạn, trong khi SeqWeightedAttention tập trung hiệu quả vào các phần quan trọng nhất của chuỗi và giảm chi phí tính toán. Sự kết hợp này giúp mô hình hiểu ngữ cảnh sâu sắc và tập trung vào các đặc trưng quan trọng, tối ưu hóa quá trình học và dự đoán.

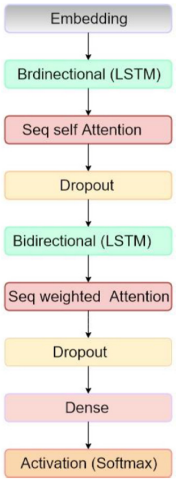
(6) Dense (thứ nhất)
Lớp này chuyển đổi đầu ra của lớp Seq Weighted Attention thành một vector cố định kích thước phù hợp cho việc phân loại.

(7) Activation (relu)
Sử dụng hàm kích hoạt ReLU để thực hiện các phép biến đổi phi tuyến tính, giúp mô hình học các quan hệ phức tạp trong dữ liệu.

(8) Dense (thứ hai)
Lớp này tạo ra đầu ra phân loại cuối cùng. Lớp Dense cuối cùng với một nơ-ron và hàm kích hoạt Sigmoid để đưa ra xác suất dự đoán cho bài toán phân loại nhị phân.

(9) Activation (Sigmoid)
Áp dụng hàm kích hoạt sigmoid vào đầu ra của lớp Dense để đưa ra xác suất dự đoán cho bài toán phân loại nhị phân.

3.2 LA_Mul07



Hình 7: Mô hình kiến trúc của LA_Mul07

Dưới đây là giải thích chi tiết từng lớp trong kiến trúc của mô hình LA_Mul07 được thể hiện trong hình 7:

(1) Embedding
Lớp nhúng (embedding) chuyển đổi các từ trong chuỗi đầu vào thành các vectơ thực. Mỗi từ trong từ vựng sẽ được gán cho một vectơ duy nhất đại diện cho ý nghĩa của từ đó.

(2) Bi-directional LSTM (LSTM hai chiều)
Lớp LSTM hai chiều (Bi-directional LSTM) có thể học được các mối tương quan phụ thuộc dài hạn trong chuỗi. Lớp này sẽ xử lý chuỗi đầu vào theo hai hướng, để thu thập thông tin từ cả ngữ cảnh trước và sau của mỗi từ.

(3) Seq Self Attention
Lớp này giúp mô hình tập trung vào các phần quan trọng của chuỗi đầu vào, dựa trên cơ chế tự chú ý. Điều này giúp cải thiện khả năng hiểu ngữ cảnh của mô hình.

(4) Các lớp Dropout
Hai lớp dropout này ngẫu nhiên loại bỏ một số kết nối trong mạng nơ-ron trong quá trình đào tạo. Điều này giúp mô hình học được các tính năng chung của dữ liệu thay vì học thuộc các chi tiết cụ thể, giúp ngăn chặn hiện tượng quá khớp (overfitting).

(5) Seq Weighted Attention
Tương tự như lớp chú ý trước đó, nhưng tập trung vào chuỗi đầu ra thay vì chuỗi đầu vào. Lớp này tính toán trọng số cho mỗi vectơ đầu ra từ LSTM hai chiều thứ hai dựa trên mức độ liên quan của nó đến vectơ đầu ra mong muốn.

(6) Dense
Lớp dense chuyển đổi vectơ đầu ra từ LSTM hai chiều thứ hai thành vectơ đầu ra mong muốn. Lớp này thực hiện một phép biến đổi tuyến tính trên vectơ đầu vào để tạo ra vectơ đầu ra.

(7) Activation (Softmax)
Lớp này thực hiện phân loại đa lớp. Hàm softmax tính toán xác suất cho mỗi lớp, sao cho tổng xác suất của tất cả các lớp là 1. Điều này giúp xác định lớp nào có khả năng cao nhất đối với đầu vào hiện tại.

3.3 Hiệu chỉnh tham số

Variables	Values
Output Dimension	64
Units	64
Max_sequence_length	73
Attention_activation	Sigmoid
Dropout rate	0.5
Dense activation	Relu, sigmoid
Optimizer	Adam
Learning_rate	0.001
Loss	binary_crossentropy
Metrics	accuracy
Epochs	10
Batch_size	32

Hình 8: Hiệu chỉnh tham số LA_Bin07

Variables	Values
Output Dimension	64
Units	64
Max_sequence_length	40
Attention_activation	Sigmoid
Dropout rate	0.5
Dense activation	Relu, softmax
Optimizer	Adam
Learning_rate	0.001
Loss	sparse_categorical_crossentropy
Metrics	accuracy
Epochs	10
Batch_size	128

Hình 9: Hiệu chỉnh tham số LA_Mul07

4 PHƯƠNG PHÁP THỰC HIỆN

Mô hình được đánh giá trên Google Colab, môi trường Linux, sử dụng GPU NVIDIA Tesla T4, những thư viện được tích hợp sẵn trên môi trường Google Colab và tải thêm thư viện keras-self-attention.

4.1 Chuẩn bị dataset

Thu thập tập dữ liệu UMUDGA, AADR, đóng vai trò là dữ liệu huấn luyện cho các mô hình. Tập dữ liệu này nên bao gồm các ví dụ đã được gán nhãn của tên miền cùng với phân loại tương ứng (benign hoặc thuộc về một gia đình botnet cụ thể).

4.2 Tiền xử lý dữ liệu

Chuẩn bị dữ liệu cho mô hình học máy phân loại tên miền bằng cách mã hóa nhãn lớp: Vấn đề phân loại nhị phân có hai nhãn, 0 (cho 'legit') và 1 (cho các lớp DGA khác). Vấn đề phân loại đa lớp có n nhãn được đánh số, tương ứng với các nhãn của các họ botnet được phát hiện.

Sử dụng Tokenizer để chuyển các tên miền thành các chuỗi số dựa trên ký tự; lấp đầy các chuỗi để có độ dài đồng nhất; chia dữ liệu thành tập huấn luyện và tập kiểm tra với tỷ lệ 80-20, đảm bảo tính khách quan trong các đánh giá.; và xác định kích thước từ vựng dựa trên số lượng ký tự duy nhất trong dữ liệu. Bước này đảm bảo dữ liệu có định dạng phù hợp để huấn luyện mô hình.

4.3 Thiết kế kiến trúc mô hình

Xác định kiến trúc của các mô hình LA_Bin07 và LA_Mul07. Các mô hình này nên kết hợp các lớp LSTM và lớp Attention. Các lớp LSTM giúp mô hình nắm bắt tính tuần tự của dữ liệu, trong khi lớp Attention giúp mô hình tập trung vào các phần quan trọng của đầu vào.

4.4 Huấn luyện mô hình

Huấn luyện các mô hình LA_Bin07 và LA_Mul07 bằng cách sử dụng tập dữ liệu đã được chuẩn bị và tiền xử lý. Trong quá trình huấn luyện, các mô hình học cách nhận ra các mẫu và đưa ra dự đoán chính xác dựa trên các tên miền đầu vào và phân loại tương ứng.

4.5 Đánh giá hiệu suất mô hình

Đánh giá hiệu suất của các mô hình đã được huấn luyện bằng cách sử dụng các độ đo đánh giá phù hợp, chẳng hạn như độ chính xác, độ precision, độ recall hoặc điểm F1. Bước này giúp xác định mức độ mô hình tổng quát hóa với dữ liệu chưa được nhìn thấy trước, kết quả mô hình phát hiện và phân loại botnet DGA một cách hiệu quả.

4.6 Xây dựng web

Xây dựng một giao diện web từ 2 mô hình LA_Bin07 và LA_Mul07 cho phép người dùng nhập một tên miền và nhận kết quả dự đoán về tính hợp lệ và họ botnet (nếu có).

5 TẬP DATASET VÀ KẾT QUẢ

5.1 Tiêu chí đánh giá

5.1.1 Binary Classification.

Đối với bài toán phân loại nhị phân, với nhãn 0 đại diện cho các tên miền hợp pháp và nhãn 1 đại diện cho các tên miền DGA, chúng tôi đánh giá Accuracy, Precision, Recall, và F_1-score, được tính toán lần lượt bằng Công thức 1, Công thức 2, Công thức 3 và Công thức 4.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

Trong đó:

- TP: Số lượng mẫu của các tên miền hợp pháp được phân loại đúng là hợp pháp.
- TN: Số lượng mẫu của các tên miền độc hại được phân loại đúng là độc hại.
- FP: Số lượng mẫu tên miền độc hại bị phân loại nhầm thành hợp pháp.
- FN: Số lượng mẫu tên miền hợp pháp bị phân loại nhầm thành độc hại.

5.1.2 Multi Classification.

Trong bài toán phân loại đa lớp, chúng tôi tiếp tục đánh giá Micro-average và Macro-average của Precision và Recall.

Micro-average tính toán tổng số lần dự đoán chính xác trên tất cả các lớp và chia cho tổng số dự đoán trên tất cả các lớp, trong khi Macro-average tính trung bình của Precision và Recall cho mỗi lớp riêng lẻ. Micro-average giúp hiển thị hiệu suất tổng thể của mô hình, trong khi Macro-average giúp đánh giá khả năng dự đoán của mô hình trên mỗi lớp một cách công bằng.

Điều này giúp nhìn nhận được hiệu suất tổng thể của mô hình thay vì chỉ nhìn vào từng lớp cụ thể. Các giá trị này được tính bằng các công thức sau:

Tính toán trung bình micro:

$$\text{Micro average Precision} = \frac{\sum_{c=1}^{C} TP_c}{\sum_{c=1}^{C} (TP_c + FP_c)} \tag{5}$$

$$\text{Micro average Recall} = \frac{\sum_{c=1}^{C} TP_c}{\sum_{c=1}^{C} (TP_c + FN_c)} \tag{6}$$

Tính toán trung bình macro:

$$\text{Macro average Precision} = \frac{\sum_{c=1}^{C} \text{Precision}_c}{C} \tag{7}$$

$$\text{Macro average Recall} = \frac{\sum_{c=1}^{C} \text{Recall}_c}{C} \tag{8}$$

trong đó:

- C là số lớp cần phân loại, c là số thứ tự của lớp đang xét.
- Dựa trên Precision và Recall, Micro-average F1-Score và Macro-average F1-Score được tính tương tự như F1-score, tương ứng.

5.2 Dataset

Để có kết quả đánh giá khách quan và toàn diện nhất, chúng tôi đã tiến hành các thử nghiệm trên 4 bộ dữ liệu, bao gồm Andrey Abakumov's DGA Repository [Aba16], và UMUDGA dataset [ZPP20] được mô tả trong bảng 1, cụ thể như sau:

- Andrey Abakumov's DGA Repository (viết tắt là AADR): Kho lưu trữ này được tạo ra vào 2016 bởi Andrey Abakumov, bao gồm mã nguồn cho các thuật toán tạo tên miền tự động và các tên miền DGA độc hại. Bộ dữ liệu này phù hợp để xây dựng mạng nơ-ron phát hiện botnet DGA. Nhiều nghiên cứu trước đây đã sử dụng nó để đánh giá giải pháp của họ. Bộ dữ liệu này được công khai trên GitHub [Aba16].
- UMUDGA dataset (viết tắt là UMUDGA): Là một bộ dữ liệu được tổng hợp và xây dựng bởi một nhóm nghiên cứu tại Đại học Murcia. Đây là bộ dữ liệu đầy đủ nhất hiện nay khi 50 họ tên miền DGA được tổng hợp. Nhóm nghiên cứu cũng cung cấp từ 10.000 đến 500.000 mẫu tên miền độc hại cho mỗi họ tên miền DGA này. Dữ liệu được sắp xếp khoa học, ở các định dạng ARFF, CSV, và tài liệu, phù hợp với các công cụ và ngôn ngữ lập trình khác nhau. Tất cả dữ liệu đều dễ dàng truy cập trên Mendeley Data [ZPP20], đảm bảo độ tin cậy và công khai của dữ liệu.

Dataset	Legit domains	DGA domains	DGA families
AADR	1.000.000	800.000	08
UMUDGA	1.000.000	500.000	50

Bảng 1: Bảng mô tả tập dataset được sử dụng

Sử dụng 02 bộ dữ liệu trên để đánh giá vì chúng đáp ứng đầy đủ các tính chất sau:

- Tính nguyên bản: Các bộ dữ liệu này bao gồm các miền lành tính và độc hại ở dạng ban đầu và dữ liệu cũng đã được dán nhãn, phù hợp làm đầu vào cho các vấn đề nghiên cứu.
- Dễ so sánh: Bốn bộ dữ liệu đã được các nhà nghiên cứu khác sử dụng để đánh giá trong các nghiên cứu trước đây của họ. Thật thuận tiện cho việc so sánh giữa kết quả của chúng tôi và kết quả trước đó.
- Tính công khai: Các bộ dữ liệu này được cung cấp công khai trên Internet, dễ dàng truy cập và thuận tiện cho các nhà nghiên cứu.

5.3 Kết quả của Binary classification

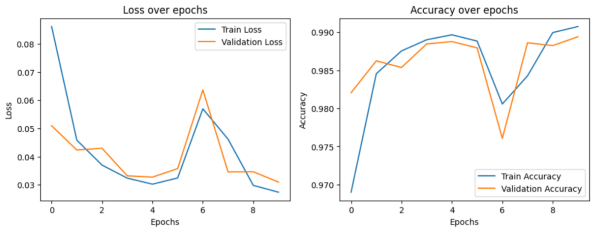
Mô hình được đánh giá thông qua các chỉ số như độ chính xác, precision, recall, F1-Score trên các tập dữ liệu thực tế. Kết quả thử nghiệm cho thấy mô hình đạt hiệu suất tốt trong việc phân loại botnets DGA.

	AADR	UMUDGA
Accuracy	0.9894	0.9788
Precision	0.9875	0.9788
Recall	0.9887	1.0
F1-score	0.9881	0.9893

Hình 10: Bảng so sánh kết quả của LA_Bin07 trên AADR, UMUDGA

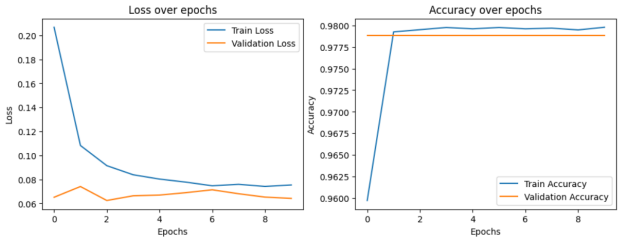
Bảng so sánh kết quả của LA_Bin07 trên AADR và UMIDGA cho thấy cả hai phương pháp đều có độ chính xác (accuracy), độ chính xác (precision), độ thu hồi (recall) và điểm F1 (F1-score) cao. Tuy nhiên, UMIDGA có độ thu hồi cao hơn (1.0) so với AADR (0.9887).

5.3.1 Đánh giá trên tập dataset AADR.



Hình 11: Sự thay đổi Accuracy và Loss của AADR trên LA_Bin07 theo thời gian

5.3.2 Đánh giá trên tập dataset UMUDGA.



Hình 12: Sự thay đổi Accuracy và Loss của UMUDGA trên LA_Bin07 theo thời gian

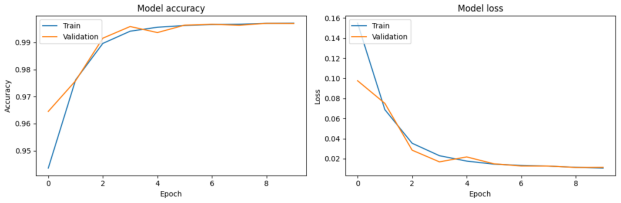
5.4 Kết quả của Multi classification

Kết quả thử nghiệm cho thấy mô hình đạt hiệu suất tốt trong giải quyết vấn đề phân loại nhị phân, với các chỉ số như độ chính xác, precision, recall được đánh giá chi tiết. Đánh giá mô hình thông qua các tham số này giúp xác định kết quả phân loại và hiệu suất tổng thể của mô hình.

	AADR	UMUDGA
Accuracy	0.997	0.8263
Micro average Precision	0.997	0.8263
Macro average Precision	0.9971	0.8422
Micro average Recall	0.997	0.8263
Macro average Recall	0.997	0.8258
Micro average F1-score	0.997	0.8263
Macro average F1-score	0.997	0.8226

Hình 13: Bảng so sánh kết quả của LA_Mul07 trên AADR, UMUDGA

Bảng kết quả cho thấy AADR và UMUDGA đều là những mô hình phân loại hiệu quả. Tuy nhiên, AADR có độ chính xác cao hơn UMUDGA đạt 99,7% so với 82,63%. Điều này cho thấy AADR có khả nuang phân loại chính xác các trường hợp hơn UMUDGA.



Hình 14: Sự thay đổi Accuracy và Loss của AADR trên LA_Mul07 theo thời gian

5.4.1 Đánh giá trên tập dataset AADR.

Mô hình đề xuất LA_Mul07 có kết quả thí nghiệm rất cao trên tập dữ liệu AADR, tiệm cận mức độ hoàn toàn chính xác với tỷ lệ chính xác tổng thể đạt 99,70%. Trong số tám gia đình botnet DGA được xem xét, 06 gia đình botnet DGA đã được phát hiện chính xác với Precision, Recall và F1-score đều đạt 1.00.

```
Classification Report:
              precision    recall  f1-score   support

    conficker       1.00      1.00      1.00     20088
cryptolocker       1.00      0.98      0.99     19986
      matsnu       1.00      1.00      1.00     20048
      pushdo       1.00      1.00      1.00     19824
       ramdo       1.00      1.00      1.00     19977
      rovnix       0.98      1.00      0.99     20042
       tinba       1.00      1.00      1.00     20180
        zeus       1.00      1.00      1.00     19855

    accuracy                           1.00    160000
   macro avg       1.00      1.00      1.00    160000
weighted avg       1.00      1.00      1.00    160000

Micro average Precision: 0.9969875
Micro average Recall: 0.9969875
Micro average F1-Score: 0.9969875
Macro average Precision: 0.9970396089338469
Macro average Recall: 0.99698795889685
Macro average F1-Score: 0.996990993324902
Accuracy: 0.996987521648407
```

Hình 15: Kết quả phân loại trên AADR của LA_Mul07

5.4.2 Đánh giá trên tập dataset UMUDGA.

Hình 16 cho thấy rằng mô hình LA_Mul07 đề xuất có độ chính xác cao trong việc phân loại các họ botnet DGA, ngay cả trong trường hợp có một số lượng lớn các họ botnet DGA, với độ chính xác trung bình là 83%. Hầu hết các lớp có thể phân loại chính xác, ngoại trừ một số lớp có tỷ lệ tương đối thấp, như alureon. Vấn đề này có thể được giải quyết bằng cách tách riêng và huấn luyện các lớp này riêng biệt. Nói chung, với việc phân loại 50 lớp của tập dữ liệu UMUDGA, mô hình đề xuất cho kết quả rất khả quan.

```
Classification Report:
              precision    recall  f1-score   support

      alureon       0.39      0.65      0.49       195
      banjori       1.00      1.00      1.00       175
        bedep       0.77      0.63      0.69       202
     ccleaner       0.94      0.98      0.96       203
       chinad       0.53      0.08      0.14       195
      corebot       0.45      0.71      0.55       213
cryptolocker       0.96      0.94      0.95       199
     dircrypt       0.91      0.53      0.67       188
         dyre       1.00      1.00      1.00       190
   fobber_v1       0.33      0.60      0.42       169
   fobber_v2       0.83      0.81      0.82       209
    gozi_gpl       0.91      0.71      0.80       208
 gozi_luther       0.91      0.63      0.74       205
    gozi_nasa       0.92      0.91      0.92       208
gozi_rfc4343       0.95      1.00      0.97       206
   kraken_v1       1.00      1.00      1.00       206
   kraken_v2       0.97      1.00      0.99       195
       locky       1.00      1.00      1.00       200
      matsnu       0.75      0.98      0.85       198
  murofet_v1       0.65      0.97      0.78       206
  murofet_v2       1.00      0.99      0.99       194
  murofet_v3       0.98      1.00      0.99       207
       necurs       0.98      1.00      0.99       201
      nymaim       1.00      1.00      1.00       205

    accuracy                           0.83      9980
   macro avg       0.84      0.83      0.82      9980
weighted avg       0.84      0.83      0.82      9980

Micro average Precision: 0.82625250501002
Micro average Recall: 0.82625250501002
Micro average F1-Score: 0.82625250501002
Macro average Precision: 0.842170146291474
Macro average Recall: 0.8258123063191859
Macro average F1-Score: 0.8225615218024079
```

```
              precision    recall  f1-score   support

     padcrypt       1.00      1.00      1.00       223
         pizd       0.97      0.89      0.93       197
proslikefan       0.92      0.74      0.82       189
      pushdo       0.97      0.49      0.65       197
      pykspa       1.00      1.00      1.00       184
pykspa_noise       1.00      0.94      0.97       192
      qadars       1.00      1.00      1.00       197
      qakbot       0.59      0.59      0.59       200
       ramdo       0.41      0.48      0.44       193
       ramnit       1.00      1.00      1.00       231
   ranbyus_v1       0.88      0.99      0.93       186
   ranbyus_v2       0.40      0.25      0.31       217
       rovnix       0.79      0.98      0.88       194
     shiotob       0.84      0.75      0.79       190
       simda       0.81      0.79      0.80       187
      sisron       0.68      0.61      0.64       220
  suppobox_1       0.72      0.59      0.65       209
  suppobox_2       0.99      0.99      0.99       202
  suppobox_3       0.92      0.92      0.92       213
       symmi       1.00      1.00      1.00       214
  tempedreve       0.96      0.97      0.97       184
       tinba       0.97      0.92      0.94       216
   vawtrak_v1       0.69      0.58      0.63       191
   vawtrak_v2       0.56      0.78      0.65       191
   vawtrak_v3       0.92      0.99      0.96       195
zeus-newgoz       0.98      0.93      0.95       191
```

Hình 16: Kết quả phân loại trên UMUDGA của LA_Mul07

6 KẾT LUẬN

Mô hình đề xuất có sự tin cậy trong phát hiện và phân loại botnets DGA. Hiệu suất cao và thời gian thử nghiệm nhanh của mô hình đề xuất đáp ứng tốt yêu cầu xử lý thời gian thực trong thực tế.

7 PHƯƠNG HƯỚNG PHÁT TRIỂN TRONG TƯƠNG LAI

Trong tương lai, dự án sẽ tiếp tục nghiên cứu và phát triển các phương pháp mới để cải thiện hiệu suất phát hiện botnets DGA. Đồng thời, tập trung vào việc mở rộng dữ liệu và cải thiện mô hình để xử lý các trường hợp phức tạp hơn, đồng thời tối ưu hóa thời gian xử lý và nâng cao hiệu quả phân loại.

8 PHÂN CÔNG CÔNG VIỆC

Phân công	Công việc
Bùi Hoàng Trúc Anh	Xử lý dataset AADR
Nguyễn Ngọc Trà My	Xử lý dataset UMUDGA
Bùi Hoàng Trúc Anh	Xây dựng model la_bin07_AADR
Lê Hoàng Oanh	Xây dựng model la_mul07_AADR
Bùi Hoàng Trúc Anh	Huấn luyện model la_bin07_AADR
Lê Hoàng Oanh	Huấn luyện model la_mul07_AADR
Bùi Hoàng Trúc Anh	Xây dựng model la_bin07_UMUDGA
Nguyễn Ngọc Trà My	Xây dựng model la_mul07_UMUDGA
Bùi Hoàng Trúc Anh	Huấn luyện model la_bin07_UMUDGA
Nguyễn Ngọc Trà My	Huấn luyện model la_mul07_UMUDGA
Nguyễn Ngọc Trà My	Hiện thực model thành website
Lê Hoàng Oanh	Tìm hiểu lý thuyết và nghiên cứu bài báo
Bùi Hoàng Trúc Anh, Lê Hoàng Oanh	Viết báo cáo, poster
Nguyễn Ngọc Trà My	Slide, thuyết trình

Hình 17: Bảng phân công công việc

TÀI LIỆU

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. inNeural computation: 9.8 (1997), pages 1735–1780.

[DS14] Ke-Lin Du and M.N.s Swamy. “Recurrent Neural Networks”. indecember 2014: pages 337–353. ISBN: 978-1-4471-5570-6. DOI: 10.1007/978-1-4471-5571-3_11.

[Aba16] Andrey Abakumov. DGA Repository. https://github.com/andrewaeva/DGA. Accessed: 2024-05-27. 2016.

[ZPP20] Mattia Zago, Manuel Gil Pérez and Gregorio Martínez Pérez. “UMUDGA-University of Murcia Domain Generation Algorithm Dataset”. inMendeley Data: (2020).

[TLT22] Tong Anh Tuan, Hoang Viet Long and David Taniar. “On detecting and classifying DGA botnets and their families”. inComputers & Security: 113 (2022), page 102549.