ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN KHOA

MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

ĐỒ ÁN MÔN AN TOÀN MẠNG

< LỚP NT140.O11.ANTT NHÓM 7 >

ĐỀ TÀI:

< Enabling Dynamic Network Access Control with Anomaly-based IDS and SDN >

GIẢNG VIÊN HƯỚNG DẪN

< ThS. Nghi Hoàng Khoa >

DANH SÁCH THÀNH VIÊN NHÓM 7

1. Nguyễn Đạo Ga Đô – 21521955
2. Đào Võ Hữu Hiệp – 21522065
3. Hoàng Anh Khoa – 21522220
4. Trần Văn Bình – 21521879

TP. HỒ CHÍ MINH, NĂM 2023

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

……, ngày … tháng … năm 202…

Người nhận xét

(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Để hoàn thành đồ án này, chúng em xin gửi lời cảm ơn chân thành đến trường Đại học Công nghệ Thông tin cùng Khoa Mạng máy tính và Truyền thông đã tạo điều kiện cho chúng em được tìm hiểu và học môn An toàn mạng. Trong quá trình học tập đã giúp chúng em có được rất nhiều kiến thức và kinh nghiệm quý báu liên quan đến an toàn mạng máy tính và bảo mật hệ thống mạng. Điều này không chỉ giúp chúng em nắm bắt được những kiến thức cơ bản về bảo mật mạng mà còn mở rộng tầm nhìn về những thách thức và cơ hội đối với lĩnh vực này trong thực tế.

Đặc biệt, chúng em xin gửi lời cảm ơn và lòng biết ơn sâu sắc nhất đến thầy Nghi Hoàng Khoa là giảng viên lý thuyết và giảng viên Tô Trọng Nghĩa hướng dẫn thực hành môn An toàn mạng thuộc Khoa Mạng máy tính và Truyền thông, trường Đại học Công Nghệ Thông Tin đã nhiệt tình giảng dạy và tạo mọi điều kiện giúp đỡ chúng em trong quá trình học tập và nghiên cứu.

Với điều kiện thời gian cũng như kinh nghiệm còn hạn chế của mỗi thành viên trong nhóm nên bài báo cáo này không thể tránh được những thiếu sót. Chúng em rất mong nhận được sự chỉ bảo, đóng góp ý kiến của các thầy, cô để chúng em có điều kiện bổ sung, nâng cao ý thức của mình để phục vụ tốt hơn công tác thực tế sau này.

Lời cuối cùng, chúng em một lần nữa xin được chân thành cảm ơn.

TP. Hồ Chí Minh, ngày 06 tháng 01 năm 2024

Nhóm 7

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG

DANH MỤC TỪ VIẾT TẮT

Từ viết tắt	Ý nghĩa
SDN	Software-Defined Network Công nghệ mạng xác định bằng phần mềm
IDS	Intrusion Detection System Hệ thống phát hiện xâm nhập
RNN	Recurrent Neural Network Mạng thần kinh hồi quy
FNN	Feedforward Neural Network Mạng thần kinh truyền thẳng
LSTM	Long Short Term Memory Bộ nhớ ngắn-dài hạn
XAI	Explainable Artificial Intelligence Trí tuệ nhân tạo có khả năng giải thích
DoS	Denial of Service Tấn công từ chối dịch vụ
TCP	Transmission Control Protocol Giao thức điều khiển truyền tải
LEMNA	Local Explanation Method using Nonlinear Approximation Phương pháp giải thích cục bộ sử dụng xấp xỉ phi tuyến tính
LIME	Local Interpretable Model-Agnostic Explanations Mô hình giải thích cục bộ không phụ thuộc vào mô hình
GMM	Gaussian Mixture Model Mô hình hỗn hợp Gaussian

Chương 1. GIỚI THIỆU TỔNG QUAN

1.1. Giới thiệu tổng quan

Công nghệ mạng xác định bằng phần mềm (Software-defined network – SDN) là một cách tiếp cận theo hướng điện toán đám mây, tức là tập trung hóa việc quản trị thiết bị mạng, giảm thiểu công việc riêng lẻ, tốn nhiều thời gian trên từng thiết bị cụ thể, tạo điều kiện thuận lợi cho việc quản lý mạng và cho phép cấu hình mạng hiệu quả bằng các chương trình được lập trình để cải thiện hiệu suất và tăng cường khả năng giám sát mạng. SDN nhằm mục tiêu giải quyết vấn đề kiến trúc tĩnh, phi tập trung, phức tạp của các mạng truyền thống không phù hợp với yêu cầu về tính linh hoạt và xử lý sự cố dễ dàng của các hệ thống mạng hiện tại.

Các mô hình mạng SDN hiện nay mang lại tính linh hoạt đáng kể cho mạng, tuy nhiên chúng đòi hỏi một cách linh hoạt hơn trong việc thực thi các chính sách kiểm soát truy cập trên mạng (dynamic network access control). Thông thường, trong các hệ thống mạng truyền thống, các chính sách kiểm soát truy cập mạng được xác định trước dưới dạng các mục của bộ định tuyến hoặc các quy tắc trên tường lửa. SDN cho phép linh hoạt hơn bằng cách chủ động cài đặt lại các luồng quy tắc vào bộ chuyển mạch (switch) để đạt được khả năng kiểm soát truy cập trên mạng. Tuy nhiên, SDN lại bị hạn chế trong việc theo dõi các hoạt động bất thường của mạng, đây thường là dấu hiệu quan trọng của các mối đe dọa bảo mật. Hạn chế này về cơ bản là do bộ điều khiển SDN không được thiết kế để phân tích chuyên sâu lưu lượng mạng như IDS.

1.2. Thực trạng

Với hạn chế trong việc theo dõi các hoạt động bất thường của mạng, nhiều nghiên cứu đã đưa ra các hướng phát triển để đạt được khả năng kiểm soát truy cập trên mạng SDN như SDN firewall, virtual firewall,… Tuy nhiên, các giải pháp này giả định các chính sách kiểm soát truy cập được quản trị viên mạng xác định dựa trên kiến thức của họ về mạng. Các chính sách kiểm soát truy cập đó không xem xét đến bất kỳ sự bất thường nào mới xảy ra trong mạng, tức là các hoạt động trên mạng chưa từng được quan sát hoặc ghi nhận lại trước đây. Những

bất thường đó đặc biệt nguy hiểm vì chúng có khả năng liên quan đến các lỗ hổng chưa xác định trong mạng hoặc các mối đe dọa bảo mật zero-day.

Anomaly-based IDS là một giải pháp mạnh mẽ với khả năng phát hiện các mối đe dọa bảo mật mới. Tuy nhiên, giải pháp này có nhược điểm đó là thay vì xác định cụ thể những mối đe dọa nào được phát hiện, anomaly-based IDS chỉ đưa kết quả là mức độ cảnh báo về các hoạt động mạng. Kết quả này sẽ không chi tiết, từ đó khó để tạo ra các chính sách kiểm soát truy cập dựa trên các thông tin này. Để sử dụng đầy đủ kết quả của anomaly-based IDS cho việc kiểm soát truy cập mạng, chúng ta phải thu hẹp khoảng cách về mặt ngữ nghĩa giữa kết quả của anomaly-based IDS và các chính sách kiểm soát truy cập mà chúng ta muốn tạo.

Cốt lõi của anomaly-based IDS là mô hình hóa mô hình bình thường của mạng. Những mô hình đó thường được triển khai thông qua phương pháp học máy. Để có thể giải thích được kết quả của các mô hình học máy này nhằm có được các kết quả chi tiết của anomaly-based IDS, hai phương pháp được sử dụng đó là whitebox và blackbox. Cơ chế whitebox giả định các đặc điểm bên trong của một mô hình là có sẵn và tăng cường cho mô hình khả năng đưa ra lời giải thích cho từng dự đoán. Cơ chế blackbox xử lý mô hình như một "hộp đen" và giải thích kết quả thông qua việc xấp xỉ giá trị decision boundary xung quanh kết quả của mô hình. Bằng cách giải thích mô hình học máy, người ta có thể hiểu rõ hơn về toàn bộ mô hình hoặc có được kiến thức về lý do đằng sau quyết định của mô hình đối với một đầu vào cụ thể. Giải thích về mô hình học máy là một giải pháp tốt có thể giúp chúng ta tạo ra các chính sách kiểm soát truy cập mạng từ anomaly-based IDS.

Từ các vấn đề trên, các tác giả Hongda Li, Feng Wei, Hongxin Hu với bài báo "*Enabling Dynamic Network Access Control with Anomaly-based IDS and SDN*" đã đề xuất ra một mô hình sử dụng anomaly-based IDS được xây dựng bằng phương pháp blackbox Machine Learning có thể dự đoán liệu đầu vào là bình thường hay bất thường. Tiếp đó, một mô hình hồi quy kết hợp với fused lasso sẽ được sử dụng để giải thích chi tiết kết quả thu được, ở đây chính là giá trị local decision boundary từ mô hình anomaly-based IDS. Phần giải thích bao gồm các điểm số cho biết mức độ đóng góp của từng đặc trưng riêng lẻ vào việc dự đoán so với đầu vào đã cho. Các đặc trưng sau đó được sắp xếp dựa trên tầm quan trọng của chúng. Mô hình sẽ chọn ra k đặc trưng để hỗ trợ tạo chính sách kiểm soát truy cập mạng, trong đó k

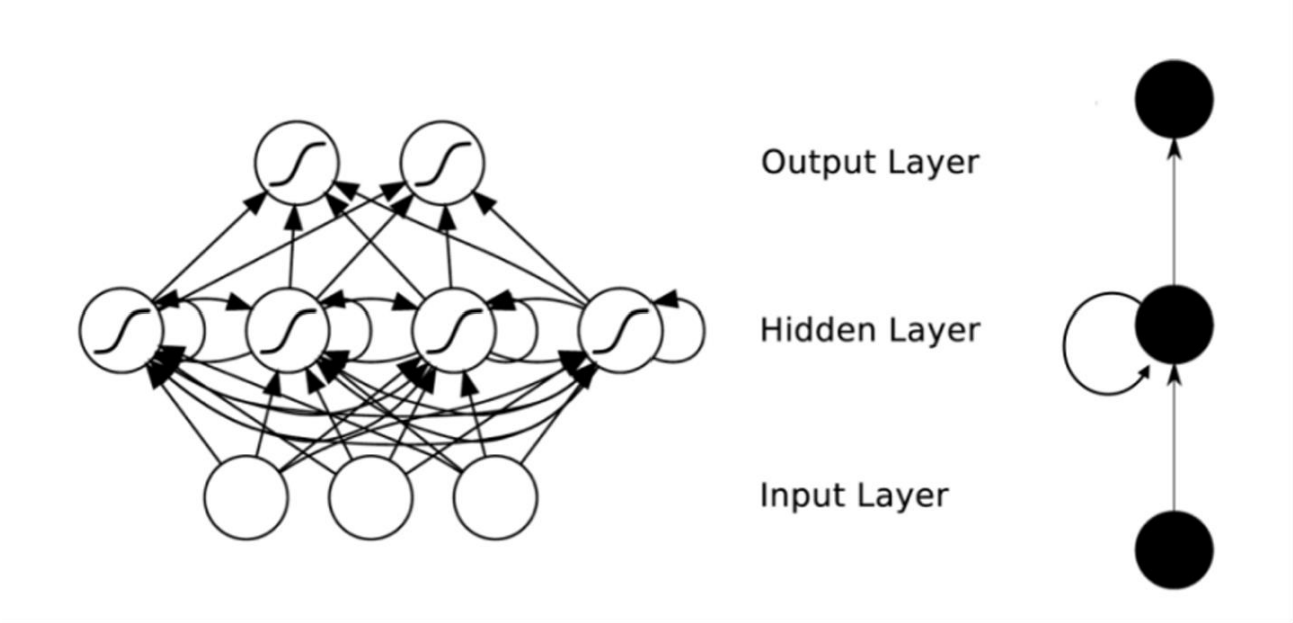
là siêu tham số. Cuối cùng, một chính sách kiểm soát truy cập mạng được tạo ra theo lời giải thích của kết quả đó.

Chương 2. PHƯƠNG PHÁP ĐỀ XUẤT

2.1. Kỹ thuật sử dụng

2.1.1. Recurrent Neural Network (RNN)

Mạng thần kinh hồi quy (Recurrent Neural Networks - RNN) bao gồm 3 layer là input layer, hidden layer và output layer, trong đó hidden layer đóng vai trò quan trọng nhất. Mô hình RNN về cơ bản có luồng thông tin một chiều từ input layer đến hidden. Ta có thể coi các nút trong hidden layer là nơi lưu trữ của toàn bộ mạng, ghi nhớ thông tin từ đầu đến cuối. Khi triển khai RNN, chúng ta có thể thấy rằng nó thể hiện khả năng học sâu. Cách tiếp cận RNN có thể được sử dụng để học phân loại có giám sát.



Hình 1. Mô hình RNN

RNN cho phép một vòng lặp định hướng có thể ghi nhớ thông tin trước đó và áp dụng nó cho đầu ra hiện tại, đây là điểm khác biệt cơ bản so với Feed-forward Neural Networks (FNNs) truyền thống. Đầu ra trước đó cũng liên quan đến đầu ra hiện tại của một chuỗi kết quả và các nút giữa các hidden layer đã có sự kết nối với nhau. Không chỉ đầu ra của input layer mà đầu ra của hidden layer cuối cùng cũng tác động lên đầu vào của hidden layer.

2.1.2. Explainable Machine Learning (XAI)

Trong quá trình phát triển của học máy, các mô hình phức tạp như mạng nơ-ron sâu (deep neural networks) thường cho ra kết quả chính xác, nhưng cách thức hoạt động của chúng thường khá khó hiểu đối với con người. Điều này gây ra một thách thức lớn trong việc áp dụng học máy vào các lĩnh vực như y tế, tài chính, an ninh mạng, nơi mà sự minh bạch và khả năng giải thích đóng vai trò quan trọng.

Explainable Machine Learning, hay còn gọi là Explainable Artificial Intelligence (XAI) trong học máy là một lĩnh vực nghiên cứu nhằm tạo ra các mô hình học máy có khả năng giải thích và hiểu được quyết định hoặc dự đoán của chúng. XAI nhằm giải quyết vấn đề "black box" trong học máy, nơi mà ngay cả những nhà thiết kế AI cũng không thể giải thích tại sao một quyết định cụ thể được đưa ra.

XAI nhằm cung cấp khả năng giải thích cho người dùng về quá trình ra quyết định của mô hình học máy. Điều này giúp người dùng hiểu rõ hơn về cách mà mô hình đưa ra dự đoán và quyết định, từ đó tăng cường sự tin tưởng và sử dụng hiệu quả hơn các hệ thống dựa trên AI.

Tất cả các mô hình học máy XAI không tuân theo cùng một cách tiếp cận hay phương pháp cụ thể. Tuy nhiên, có ba nguyên tắc chính mà các mô hình XAI thường hướng đến: tính minh bạch, khả năng giải thích và tính hiểu được.

Một mô hình được coi là minh bạch (transparency) nếu quá trình trích xuất các thông số mô hình từ dữ liệu huấn luyện và tạo ra các nhãn từ dữ liệu kiểm tra có thể được mô tả và lý giải bởi người thiết kế phương pháp. Điều này đảm bảo rằng các quá trình và phương pháp mà mô hình sử dụng để đưa ra quyết định có thể được hiểu và theo dõi.

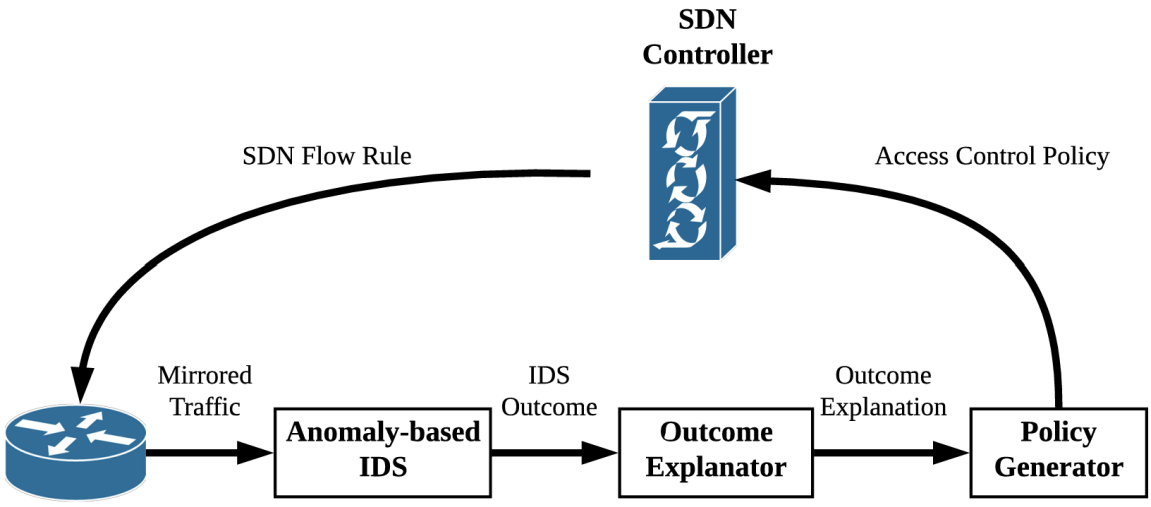
Tính hiểu được (interpretability) mô tả khả năng hiểu và giải thích được mô hình học máy. Điều này đảm bảo rằng mô hình có khả năng trình bày cơ sở cốt lõi dưới hình thức quyết định, sao cho con người có thể hiểu được. Tính hiểu được giúp người dùng có thể nắm bắt và đánh giá các quyết định của mô hình, đồng thời tạo niềm tin và sự chấp nhận.

Tính giải thích (explainability) là một khái niệm quan trọng trong XAI, nhưng chưa có một định nghĩa chung được chấp nhận. Có một lời giải thích về khái niệm này có thể chấp nhận được đó là "tập hợp các đặc trưng của miền có thể giải thích, đã đóng góp vào việc đưa ra

quyết định (ví dụ: phân loại hoặc hồi quy) cho một ví dụ cụ thể". Tính giải thích giúp người dùng hiểu được những yếu tố nào trong dữ liệu đã đóng góp vào quyết định của mô hình, giúp kiểm tra và cải thiện mô hình, đồng thời khám phá các thông tin mới.

2.2. Mô hình tổng quan

Mặc dù SDN cung cấp khả năng lập trình cho mạng nhưng nó vẫn bị hạn chế trong việc kiểm soát sự bất thường trong mạng. Hạn chế này về cơ bản là do bộ điều khiển SDN không được thiết kế để phân tích chuyên sâu lưu lượng mạng như IDS.



Hình 2.Mô hình tổng quan

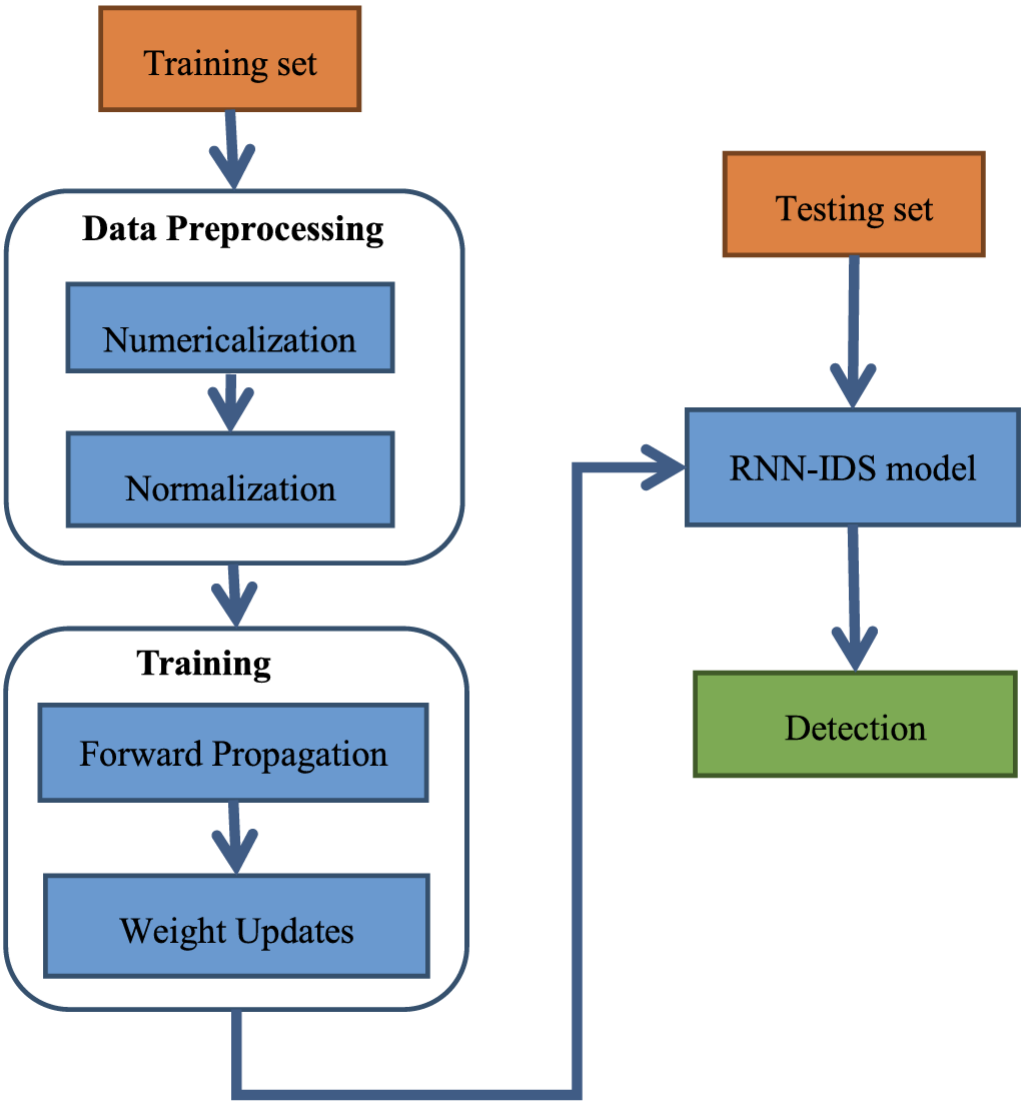
Để khắc phục nhược điểm này, anomaly-based IDS được sử dụng nhằm phân tích trực tiếp lưu lượng truy cập một cách chi tiết và báo cáo bất kỳ sự bất thường nào của mạng để hỗ trợ bộ điều khiển SDN kiểm soát mạng. Khả năng kiểm soát truy cập mạng động đạt được thông qua sự kết hợp giữa SDN và anomaly-based IDS. Hình 2 mô tả ý tưởng tổng quan về cách tiếp cận tổng thể của bài báo. Các tác giả sao chép một bản sao lưu lượng truy cập mạng từ SDN switch sang *Anomaly-based IDS*. Nếu kết quả của anomaly-based IDS cho thấy sự bất thường, nó sẽ được gửi đến bộ giải thích kết quả - *Outcome Explanator* để được giải thích. Sau đó, phần giải thích về kết quả sẽ được trình tạo chính sách – *Policy Generator* xử lý để tạo ra chính sách kiểm soát truy cập. Chính sách kiểm soát truy cập được tạo có thể được gửi đến bộ

điều khiển SDN thông qua một framework kiểm soát mạng. Bộ điều khiển SDN cuối cùng sẽ cài đặt các luồng quy tắc SDN vào SDN swtich theo chính sách kiểm soát truy cập.

2.3. Mô hình chi tiết

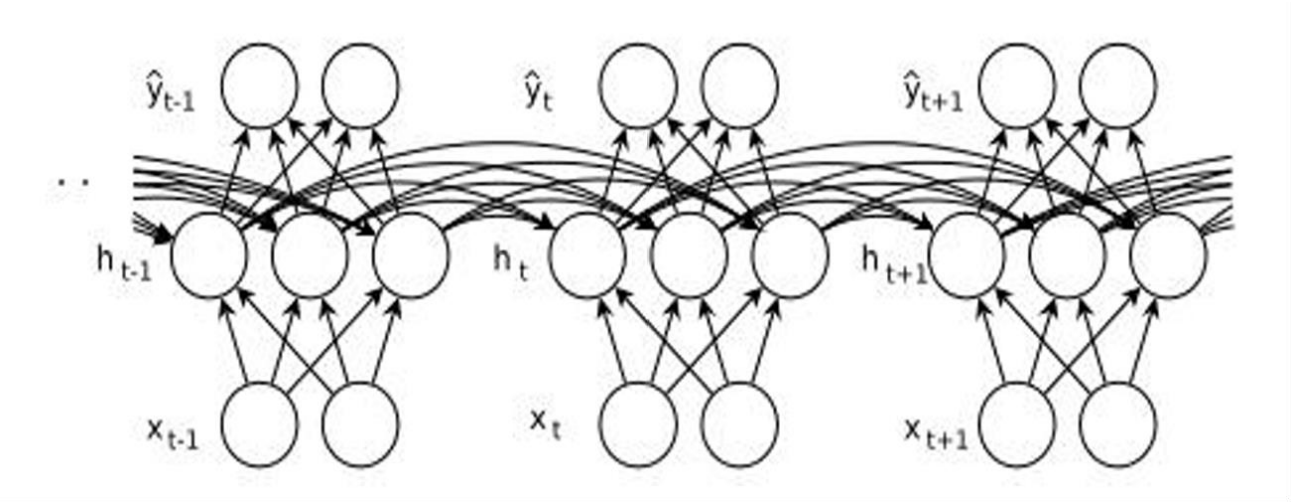
2.3.1. Anomaly-based IDS

Bởi vì Deep Learning có khả năng trích xuất các biểu diễn tốt hơn từ dữ liệu để tạo ra các mô hình tốt hơn. Từ đó, hệ thống abnomaly-based IDS sẽ được xây dựng dựa trên phương pháp Deep Learning, mà phương pháp cụ thể được sử dụng đó là mạng thần kinh hồi quy - RNN, từ đó xây dựng một mô hình RNN-IDS hoàn thiện. Các tác giả đề xuất sử dụng mô hình này trong [1].



Hình 3. Mô hình RNN-IDS

Việc huấn luyện mô hình RNN-IDS bao gồm hai phần, đó là Forward Propagation và Back Propagation. Forward Propagation có trách nhiệm tính toán các giá trị đầu ra và Back Propagation có trách nhiệm chuyển các phần dư đã được tích lũy để cập nhật các trọng số, về cơ bản RNN không khác biệt so với đào tạo mạng thần kinh bình thường. Phương pháp tổng quát để xây dựng mô hình RNN-IDS được mô tả trong Hình 3.



Hình 4. Chi tiết RNN

Theo Hình 1, một mạng lưới thần kinh hồi quy được biểu diễn chi tiết trong Hình 4. Mô hình RNN sẽ như sau: Với các mẫu đào tạo x_i $(i = 1,2,..,m)$, chuỗi các trạng thái ẩn h_i $(i = 1,2,..,m)$ và một chuỗi dự đoán $\hat{y}_i$ $(i = 1,2,..,m)$. $\boldsymbol{W}_{hx}$ là ma trận trọng số input-to-hidden, $\boldsymbol{W}_{hh}$ là ma trận trọng số hidden-to-hidden, $\boldsymbol{W}_{yh}$ là ma trận trọng số hidden-to-output, và các vector b_h và b_y là các bias. Hàm kích hoạt (activation function) $\boldsymbol{e}$ là $sigmoid$ và hàm phân loại là $SoftMax$.

Từ đó ta có được thuật toán cho bước Forward Propagation và Weights Update lần lượt như sau:

Algorithm 1 Forward Propagation Algorithm

Input x_i(i = 1, 2, . . ., m)
Output $\hat{y}_i$
1: for *i* from *1* to *m* do
2: $ti = \mathbf{W}_{hxxi} + \mathbf{W}_{hhhi-1+bh}$
3: $hi = \text{sigmoid}\ (ti)$
4: $si = \mathbf{W}_{yhhi+by}$
5: $\hat{y}i = \text{SoftMax}\ (si)$
6: end for

Hình 5: Thuật toán 1 - Forward Propagation

Algorithm 2 Weights Update Algorithm

Input $\langle y_i, \hat{y}_i \rangle (i = 1, 2, \ldots, \mathrm{m})$

Initialization $\theta = \{\mathbf{W}_{hx}, \mathbf{W}_{hh}, \mathbf{W}_{yh}, b_h, b_y\}$

Output $\theta = \{\mathbf{W}_{hx}, \mathbf{W}_{hh}, \mathbf{W}_{yh}, b_h, b_y\}$

1: for i from k downto 1 do

2: Calculate the cross entropy between the output value and the label value: $\mathrm{L}(y_i: \hat{y}_i) \leftarrow -\sum_{\mathrm{i}} \sum_j \mathrm{y}_{ij} \log\ (\hat{y}_{ij}) + (1 - y_{ij}) \log(1 - \hat{y}_{ij})$

3: Compute the partial derivative with respect to θ_i : $\delta_i \leftarrow dL/d\theta_i$

4: Weight update: $\theta_i \leftarrow \theta_i \eta + \delta_i$

5: end for

Hình 6: Thuật toán 2 - Weights Update

Hàm mục tiêu liên quan đến RNN cho một cặp huấn luyện (x_i, y_i) được định nghĩa là $f(\theta) = \boldsymbol{L}(y_i: \hat{y}_i)$, trong đó $\boldsymbol{L}$ là hàm khoảng cách đo độ lệch của dự đoán $\hat{y}_i$ so với nhãn thực tế y_i. Gọi n là tốc độ học (learning rate) và k là số lần lặp hiện tại. Cho một dãy các nhãn y_i $(i = 1,2,..,m)$.

2.3.2. Outcome Explanation

Các tác giả đã sử dụng LEMNA để xây dựng một mô hình có thể hiểu được (interpretable) và thu nhận được những giải thích để minh hoạ một điều rằng tại sao mỗi bản ghi (record) lại được xem là bình thường hoặc bất thường.

Việc giải thích kết quả của hệ thống Anomaly-based IDS bao gồm hai bước chính. Ở bước đầu tiên, hệ thống cần huấn luyện một mô hình để xấp xỉ giá trị local decision boundary của mô hình dự đoán mục tiêu. Điều này có nghĩa là mô hình sẽ xác định một ranh giới quyết định để phân biệt các điểm dữ liệu bình thường và bất thường trong tập dữ liệu huấn luyện. Sau đó, ở bước thứ hai, hệ thống sẽ sử dụng mô hình đã được huấn luyện và đầu vào được cung cấp để suy luận dựa trên một số logic giải thích. Mục đích của bước này là giải thích kết quả dự đoán của mô hình bằng cách sử dụng mô hình đã huấn luyện và đầu vào cụ thể. Có thể sử dụng các phương pháp giải thích khác nhau để hiểu logic hoạt động của mô hình và lý giải kết quả dự đoán.

Tại bước đầu tiên, tổng hợp các mẫu dữ liệu xung quanh một điểm dữ liệu x và sử dụng mô hình dự đoán mục tiêu để dự đoán nhãn cho mỗi mẫu dữ liệu. Tiếp theo, có thể sử dụng những mẫu dữ liệu tổng hợp này để huấn luyện một mô hình hồi quy tuyến tính được định nghĩa như sau:

$$l(\mathbf{x}) = \boldsymbol{\alpha}\mathbf{x} + \epsilon$$

trong đó ϵ là sai số đề cập đến sự sai lệch trong đường hồi quy, x là mẫu dữ liệu tổng hợp chứa $(x_1, x_2, \ldots, x_M)^T$ chứa M đặc trưng, vector $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_M)$ chứa các hệ số của mô hình tuyến tính.

Mô hình hồi quy tuyến tính tiêu chuẩn không thể xấp xỉ đúng đường ranh giới quyết định cục bộ cho hệ thống anomaly-based IDS. Điều này xảy ra vì hệ thống anomaly-based IDS đưa ra quyết định dựa trên sự phụ thuộc giữa nhiều đặc trưng. Để giải quyết vấn đề trên, các tác giả bài báo đã sử dụng Fused Lasso, một thuật ngữ có thể giúp mô hình hồi quy tuyến tính ước tính sự phụ thuộc giữa các đặc trưng. Fused Lasso giúp mô hình hồi quy tuyến tính xấp xỉ đúng sự phụ thuộc giữa các đặc trưng. Dưới đây là hàm mất mát L của mô hình hồi quy tuyến tính được định nghĩa với Fused Lasso:

$$L(l(\mathbf{x}), y) = \sum_{i=1}^{N} \| \boldsymbol{\alpha}\mathbf{x_i} - y_i \|$$

$$\text{subject to} \quad \sum_{j=2}^{M} \| \alpha_j - \alpha_{j-1} \| \leq S$$

trong đó $\mathbf{x_i}$ là mẫu dữ liệu tổng hợp thứ i biểu diễn dưới dạng vector đặc trưng, N là số lượng mẫu dữ liệu tổng hợp, là y_i ăn của , $\mathbf{x_i}$ ‖ là khoảng cách chuẩn L2 giữa dự đoán của mô hình và nhãn thực tế. S là một siêu tham số điều khiển ngưỡng khác biệt của các hệ số được gán cho các đặc trưng liền kề. Thông qua tối thiểu hóa một hàm mất mát vừa được định nghĩa, thu được một mô hình hồi quy tuyến tính tốt hơn để xấp xỉ đường ranh giới quyết định cục bộ.

Sau khi phát triển một mô hình hồi quy tuyến tính xấp xỉ đường ranh giới quyết định cục bộ của hệ thống anomaly-based IDS, chúng ta có thể sử dụng các hệ số của mô hình tuyến tính để tạo ra lời giải thích cho kết quả dự đoán của mô hình. Khi huấn luyện mô hình hồi quy tuyến tính này, các hệ số được gán cho các đặc trưng. Gán hệ số lớn hơn cho một đặc trưng trong quá trình huấn luyện mô hình tuyến tính có nghĩa là đặc trưng này góp phần lớn vào dự đoán của mô hình tuyến tính, do đó có tác động lớn đến quyết định của hệ thống. Do đó, tầm quan trọng của mỗi đặc trưng cung cấp một giải thích cho kết quả của hệ thống

Trong thực nghiệm của bài báo, các tác giả đã sử dụng các trường hợp của các cuộc tấn công Neptune Attack làm trường hợp thử nghiệm. Tấn công Neptune là một trong các kiểu tấn công thuộc loại tấn công từ chối dịch vụ (Denial of Service - DoS). Mục tiêu chính của tấn công Neptune là làm cho tài nguyên của máy chủ TCP mục tiêu cạn kiệt và không còn khả dụng cho người dùng thông thường khác. Tấn công Neptune thường tạo ra một lượng lớn yêu cầu không hợp lệ hoặc gây quá tải cho hệ thống mạng, dẫn đến sự chậm trễ hoặc sự cố về hiệu suất của hệ thống. Các bản ghi Neptune được sử dụng làm đầu vào trong thử nghiệm được biểu thị dưới dạng một mảng gồm 41 các đặc tính (feature) của từng gói tin TCP ví dụ như:

Record1: (0, tcp,private, S0, ..., 255, 20, 0.08, 0.07, 0, 0, 1, 1, 0, 0)

Record2: (0, tcp, imap4, REJ, ..., 255, 17, 0.07, 0.07, 0, 0, 0, 0, 1, 1)

Trong thử nghiệm này, có hai bước chính cần thực hiện. Đầu tiên, tạo ra một tập hợp các mẫu bản ghi nhất định, sau đó sử dụng các bản ghi đó và nhãn phân loại của chúng để huấn luyện ra một mô hình phân loại mang tính hiểu được. Sau khi huấn luyện thành công mô hình có thể hiểu được trên, mô hình này sẽ gán điểm quan trọng khác nhau cho tất cả 41 đặc trưng (features) dựa trên hệ số của chúng. Điểm quan trọng này cho biết mức độ ảnh hưởng của từng đặc trưng đến quyết định của mô hình, trong trường hợp thực nghiệm này là quyết định xem gói tin TCP này là một gói tin bình thường hay một gói tin được sinh ra bởi một cuộc tấn công Neptune. Điểm quan trọng cao hơn cho thấy đặc trưng đó có ảnh hưởng lớn đến quyết định, trong khi điểm quan trọng thấp hơn cho thấy đặc trưng đó có ảnh hưởng ít đến quyết định. Các tác giả đã chọn ra 4 đặc trưng quan trọng nhất để biểu diễn cho lời giải thích của từng kết quả. (dst_host_serror_rate) và (dst_host_srv_serror_rate) đại diện cho tỉ lệ phần trăm các kết nối có lỗi syn, (dst_host_rerror_rate) và (dst_host_srv_rerror_rate) đại diện cho tỉ lệ phần trăm các kết nối có lỗi reject. Đây là hai loại lỗi thường dùng để nhận dạng các cuộc tấn công Neptune

Feature	dur.	proto.	service	flag	...	dsthost_serror_rate*	dsthost_srv_serror_rate*	dsthost_rerror_rate*	dsthost_srv_rerror_rate*
Record1	0	tcp	private	S0	...	1	1	0	0
Record2	0	tcp	imap4	REJ	...	0	0	1	1

2.3.3. Policy Generation

Một trong những thách thức của Anomaly-based IDS là cách thức xác định các hoạt động mạng bất thường. Một cách tiếp cận phổ biến là sử dụng mô hình phân loại để phân loại các hoạt động mạng thành các loại bình thường và bất thường. Tuy nhiên, cách tiếp cận này có thể dẫn đến các kết quả False Positive, tức là các hoạt động mạng bình thường bị nhầm là bất thường.

Một cách tiếp cận khác là sử dụng mô hình hồi quy để dự đoán khả năng xảy ra tấn công dựa trên các tính năng mạng bất thường. Cách tiếp cận này có thể giúp giảm thiểu các kết quả sai dương, nhưng nó có thể dẫn đến các kết quả False Negative, tức là các hoạt động mạng bất thường bị bỏ qua.

Một cách tiếp cận khác là sử dụng "policy generation" để tạo ra các chính sách kiểm soát truy cập mạng (access control policy) để chặn các hoạt động mạng bất thường. Từ đó, dựa trên các

giải thích từ mô hình Outcome Explanation, ta có thể tạo ra được các chính sách phù hợp để gởi đến cho SDN controller, sau đó bộ điều khiển này tạo và cài đặt các quy tắc SDN tương ứng vào SDN switch để đạt được kiểm soát truy cập trên mạng.

Policy generation hoạt động theo các bước sau:

- Bước 1: IDS xác định các hoạt động mạng bất thường.
- Bước 2: IDS sử dụng các thuật toán học máy để phân tích các hoạt động mạng bất thường và xác định các tính năng bất thường.
- Bước 3: IDS sử dụng các tính năng bất thường để tạo ra một chính sách kiểm soát truy cập mạng.

Chính sách kiểm soát truy cập mạng sẽ được gửi đến bộ điều khiển mạng (network controller), bộ điều khiển này sẽ cài đặt chính sách trong các thiết bị mạng để chặn các hoạt động mạng bất thường.

Một chính sách kiểm soát truy cập trên mạng gồm 2 thành phần là *filter* và *action,* và có thể biểu diễn dưới dạng <*filter*, *action*>. Các *filter* sẽ chịu trách nhiệm chọn các thực thể mạng như host, flow, connection và packet mà các chính sách kiểm soát truy cập sẽ được áp dụng lên. Các *action* xác định các hành động sẽ được thực hiện đối với thực thể được các bộ lọc chọn.

Ưu điểm của “policy generation”:

- Có thể giúp giảm thiểu các kết quả sai dương và sai âm.
- Có thể được sử dụng để chặn các loại tấn công khác nhau.
- Có thể được tích hợp với các hệ thống AIDS khác.

Nhược điểm của “policy generation”:

- Có thể yêu cầu nhiều dữ liệu đào tạo hơn các phương pháp khác.
- Có thể khó giải thích.
-

Policy generation là một phương pháp tiềm năng để cải thiện hiệu quả của các hệ thống abnomaly-based IDS. Phương pháp này có thể giúp giảm thiểu các kết quả sai dương và sai âm, cũng như có thể được sử dụng để chặn các loại tấn công khác nhau.

Chương 3. TRIỂN KHAI THỰC NGHIỆM

3.1. Anomaly-based IDS

3.1.1. Dataset

Bộ dataset được sử dụng là NSL-KDD. So với bộ dữ liệu KDD gốc, NSL-KDD đã giải quyết được hai vấn đề lớn. Vấn đề đầu tiên là số lượng bản ghi dư thừa quá lớn khiến cho bộ phân loại ưu tiên các bản ghi thường xuất hiện hơn và bỏ qua các bản ghi không thường xuất hiện. Vấn đề thứ hai là NSL-KDD dán nhãn các bản ghi trong tập dữ liệu KDD với mức độ khó khác nhau. Mỗi bản ghi có 41 đặc điểm được mô tả trong Bảng 1.

No.	**Features**	**Types**	No.	**Features**	**Types**
1	duration	cont.	22	is_guest_login	cont.
2	protocol_type	symb.	23	count	cont.
3	service	symb.	24	srv_count	cont.
4	flag	symb.	25	serror_rate	cont.
5	src_bytes	cont.	26	srv_serror_rate	cont.
6	dst_bytes	cont.	27	rerror_rate	cont.
7	land	cont.	28	srv_rerror_rate	cont.
8	wrong_fragment	cont.	29	same_srv_rate	cont.
9	urgent	cont.	30	diff_srv_rate	cont.
10	hot	cont.	31	srv_diff_host_rate	cont.
11	num_failed_logins	cont.	32	dst_host_count	cont.
12	logged_in	cont.	33	dst_host_srv_count	cont.
13	num_compromised	cont.	34	dst_host_same_srv_count	cont.
14	root_shell	cont.	35	dst_host_diff_srv_rate	cont.
15	su_attempted	cont.	36	dst_host_same_src_port_rate	cont.
16	num_root	cont.	37	dst_host_srv_diff_host_rate	cont.
17	num_file_creations	cont.	38	dst_host_serror_rate	cont.
18	num_shells	cont.	39	dst_host_srv_serror_rate	cont.
19	num_access_files	cont.	40	dst_host_rerror_rate	cont.
20	num_outbound_cmds	cont.	41	dst_host_srv_rerror_rate	cont.
21	is_host_login	cont.			

Bảng 1: Các đặc trưng của NSL-KDD dataset

3.1.2. Tiền xử lý dữ liệu

Có 38 đặc trưng numeric và 3 đặc trưng nonumeric trong bộ dữ liệu NSL-KDD. Vì giá trị đầu vào của RNN-IDS phải là ma trận numeric nên chúng ta phải chuyển đổi một số đặc trưng nonumeric như ‘protocol_type’, ‘service’ và ‘flag’ thành dạng numeric. Ví dụ: đặc trưng ‘protocol_type’ có ba loại thuộc tính là ‘tcp’, ‘udp’ và ‘icmp’ và các giá trị số của nó được mã hóa dưới dạng vector nhị phân (1,0,0), (0,1,0) và (0,0,1). Tương tự, đối tượng ‘service’ có 70 loại thuộc tính và đối tượng 'flag' có 11 loại thuộc tính. Tiếp tục theo cách này, các 41 đặc trưng ban đầu sẽ được chuyển thành 121 đặc trưng sau quá trình chuyển đổi.

Tiếp theo, ta sẽ cần chuẩn hoá một số đặc trưng, ví dụ như ‘duration[0,58329]’, ‘src_bytes[0,1.3 × 109]’ và ‘dst_bytes[0,1.3 × 109]’, ta thấy rằng sự chênh lệch giữa giá trị lớn nhất và nhỏ nhất có phạm vi rất lớn, vì vậy ta cần áp dụng phương pháp logarithmic scaling để thu được các phạm vi của ‘duration[0,4,77]’, ‘src_bytes[0,9,11]’ và ‘dst_bytes[0,9,11]’. Thứ hai, giá trị của mọi đặc trưng được ánh xạ tuyến tính tới phạm vi [0,1].

3.1.3. Xây dựng và huấn luyện mô hình

Nhóm sẽ xây dựng một mô hình RNN-IDS cho việc phân loại nhị phân (hai trạng thái là ‘normal’ và ‘abnomal’), mô hình sẽ được xây dựng và huấn luyện trên Google Colab. Nhóm sử dụng Keras để xây dựng model RNN, LSTM để xây dựng các hidden layer và một Dense layer cho mô hình.

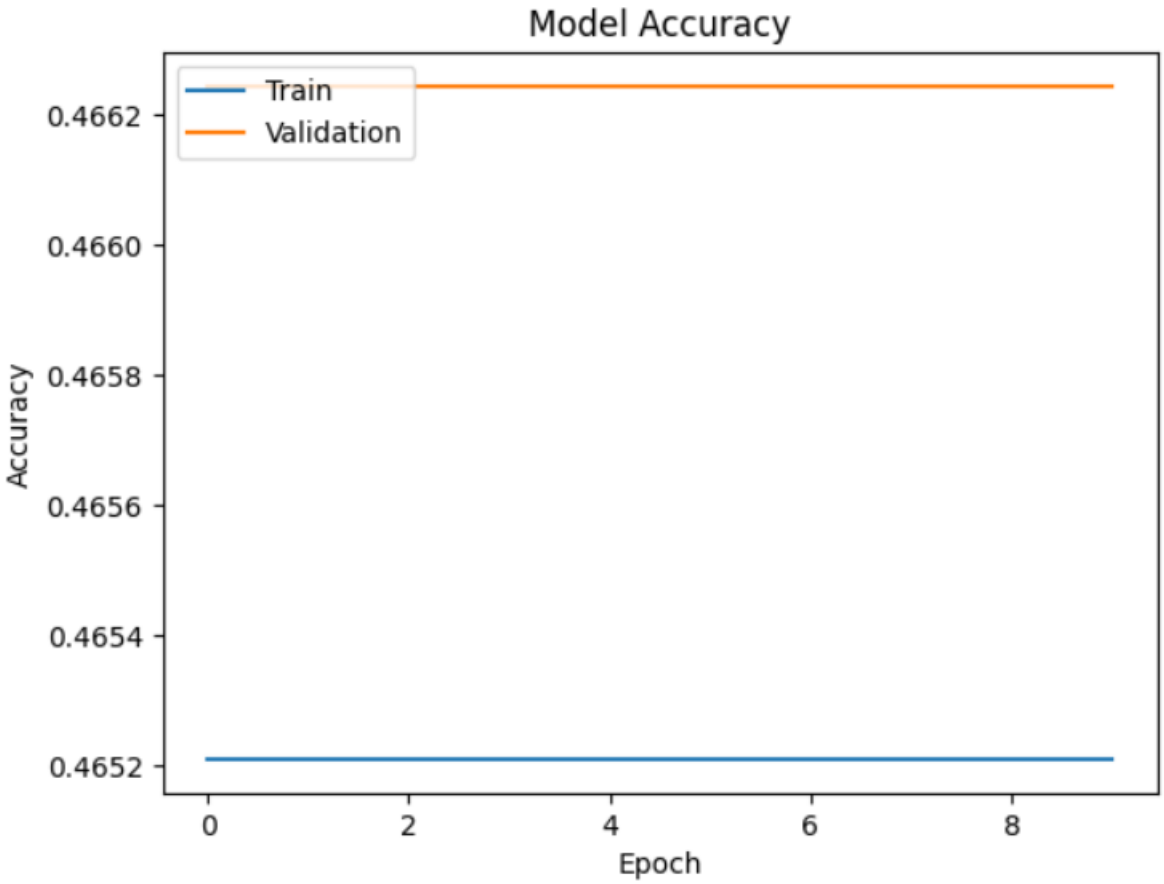
Cấu hình chi tiết của máy trên Google Colab là 12.7GB RAM, TPU là Python 3 Google Computer Engine backend, 107.7GB SSD.

Sau khi huấn luyện trên tập dữ liệu huấn luyện và thử nghiệm trên tập dữ liệu thử nghiệm theo phương pháp trong phần 3.1, nhóm đã hoàn thiện được mô hình, tuy nhiên độ chính xác không được cao:

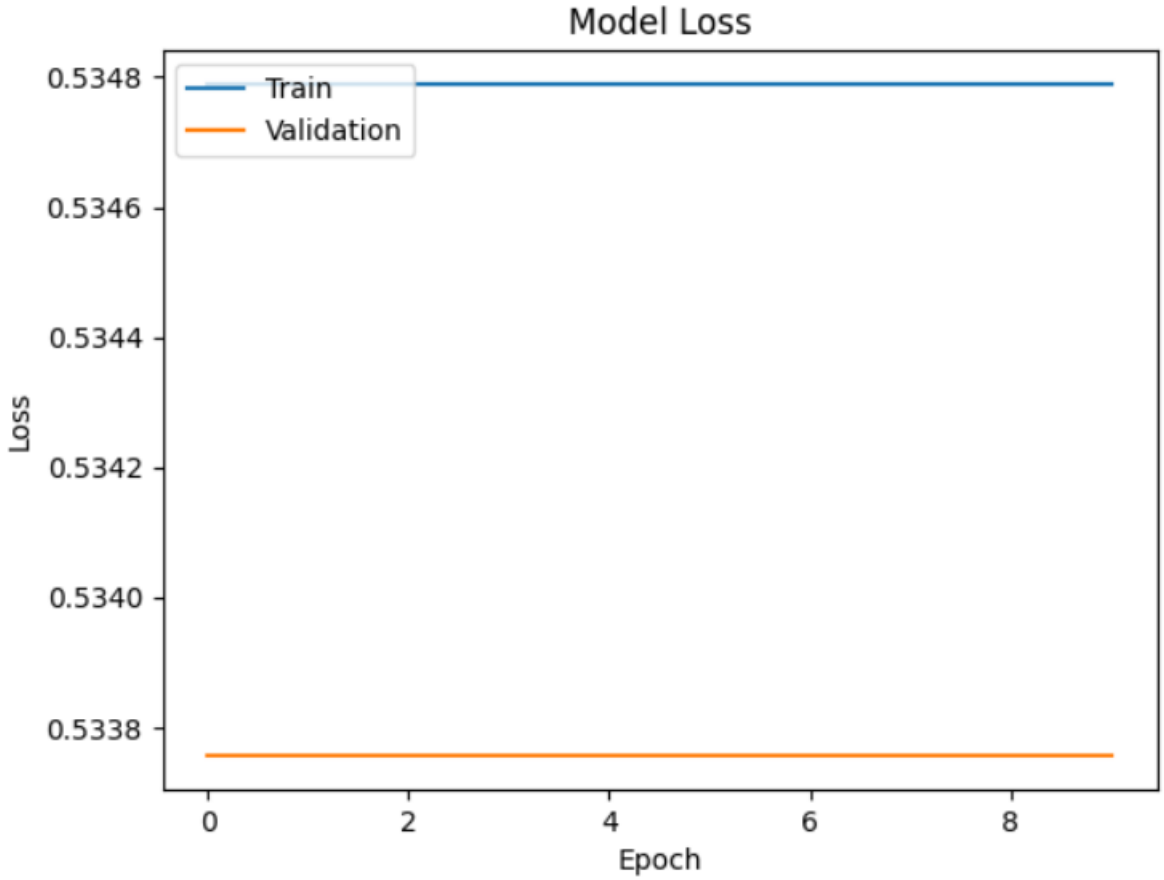
```
[ ]  Model = keras.Sequential([

            keras.layers.LSTM(80,input_shape=(features,x_train.shape[2]),
                              activation='sigmoid',recurrent_activation='hard_sigmoid'),
            keras.layers.Dense(1,activation="softmax")
        ])

    Model.compile(optimizer='rmsprop',loss='mse', metrics=['accuracy'])

    #Training the model

    Model.fit(x_train, y, epochs=10, batch_size= 32)
    Model.summary()

    # Final evaluation of the model
    scores = Model.evaluate(x_test, ytest, verbose=0)
    print('/n')
    print("Accuracy: %.2f%%" % (scores[1]*100))
```

Hình 7: Xây dựng mô hình RNN-IDS với "softmax"

Kết quả:



Hình 8: Biểu đồ Accuracy sau khi huấn luyện mô hình RNN-IDS với "softmax"



Hình 9: Biểu đồ Loss sau khi huấn luyện mô hình RNN-IDS với "softmax"

```
Epoch 1/10
3150/3150 [==============================] - 442s 139ms/step - loss: 0.5348 - accuracy: 0.4652 - val_loss: 0.5338 - val_accuracy: 0.4662
Epoch 2/10
3150/3150 [==============================] - 414s 131ms/step - loss: 0.5348 - accuracy: 0.4652 - val_loss: 0.5338 - val_accuracy: 0.4662
Epoch 3/10
3150/3150 [==============================] - 405s 129ms/step - loss: 0.5348 - accuracy: 0.4652 - val_loss: 0.5338 - val_accuracy: 0.4662
Epoch 4/10
3150/3150 [==============================] - 406s 129ms/step - loss: 0.5348 - accuracy: 0.4652 - val_loss: 0.5338 - val_accuracy: 0.4662
Epoch 5/10
3150/3150 [==============================] - 409s 130ms/step - loss: 0.5348 - accuracy: 0.4652 - val_loss: 0.5338 - val_accuracy: 0.4662
Epoch 6/10
3150/3150 [==============================] - 408s 129ms/step - loss: 0.5348 - accuracy: 0.4652 - val_loss: 0.5338 - val_accuracy: 0.4662
Epoch 7/10
3150/3150 [==============================] - 407s 129ms/step - loss: 0.5348 - accuracy: 0.4652 - val_loss: 0.5338 - val_accuracy: 0.4662
Epoch 8/10
3150/3150 [==============================] - 406s 129ms/step - loss: 0.5348 - accuracy: 0.4652 - val_loss: 0.5338 - val_accuracy: 0.4662
Epoch 9/10
3150/3150 [==============================] - 409s 130ms/step - loss: 0.5348 - accuracy: 0.4652 - val_loss: 0.5338 - val_accuracy: 0.4662
Epoch 10/10
3150/3150 [==============================] - 406s 129ms/step - loss: 0.5348 - accuracy: 0.4652 - val_loss: 0.5338 - val_accuracy: 0.4662
/n
Accuracy: 56.92%
```

Hình 10: Kết quả sau khi huấn luyện mô hình RNN-IDS với "softmax"

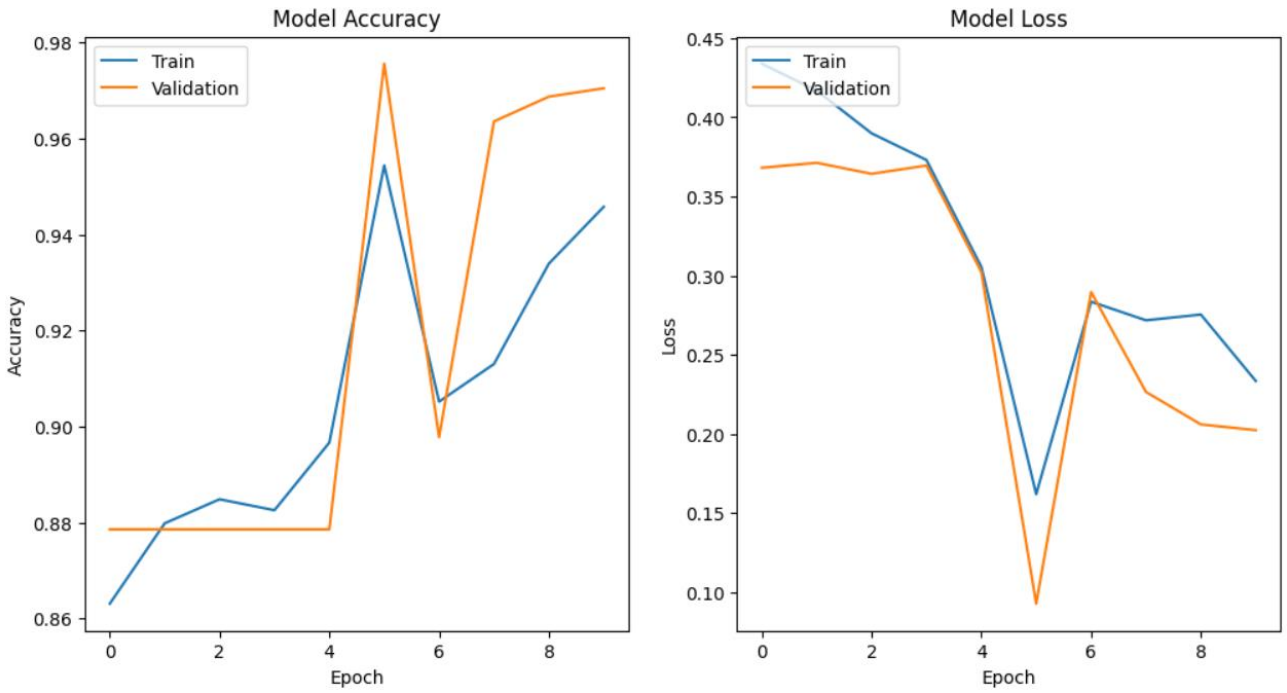
Độ chính xác trên tập dữ liệu huấn luyện khoảng 46.52% và trên tập dữ liệu kiểm tra là 56.92%, ta có thể thấy rằng độ chính xác của mô hình này khá thấp.

Sau khi xem xét lại phương pháp trong phần 3.1, nhóm nhận thấy rằng kết quả của mô hình ta cần đó là phân loại nhị phân, vì vậy nếu sử dụng classification function là SoftMax ở đây không phù hợp, có thể dẫn đến mô hình giảm độ chính xác, vì vậy nhóm đã có một số điều chỉnh trong mô hình như sau:

```
[ ]  # Using tanh and sigmoid as activation functions

     Model = keras.Sequential([

             keras.layers.LSTM(80,input_shape=(features,x_train.shape[2]),
                               activation='tanh',recurrent_activation='hard_sigmoid'),
             keras.layers.Dense(1,activation="tanh")
         ])

     Model.compile(optimizer='adam',loss='binary_crossentropy', metrics=['accuracy'])

     #Training the model
     Model.fit(x_train, y, epochs=10, batch_size= 32)
     Model.summary()

     # Final evaluation of the model
     scores = Model.evaluate(x_test, ytest, verbose=0)
     print("/n")
     print("Accuracy: %.2f%%" % (scores[1]*100))
```

Hình 11: Xây dựng mô hình RNN-IDS với "tanh"

Kết quả:



Hình 12: Đồ thi Accuracy và Loss sau khi huấn luyện mô hình RNN-IDS với "tanh"

```
Epoch 1/10
3150/3150 [==============================] - 415s 131ms/step - loss: 0.4339 - accuracy: 0.8631 - val_loss: 0.3682 - val_accuracy: 0.8786
Epoch 2/10
3150/3150 [==============================] - 406s 129ms/step - loss: 0.4169 - accuracy: 0.8798 - val_loss: 0.3713 - val_accuracy: 0.8786
Epoch 3/10
3150/3150 [==============================] - 406s 129ms/step - loss: 0.3899 - accuracy: 0.8848 - val_loss: 0.3644 - val_accuracy: 0.8786
Epoch 4/10
3150/3150 [==============================] - 414s 132ms/step - loss: 0.3731 - accuracy: 0.8826 - val_loss: 0.3696 - val_accuracy: 0.8786
Epoch 5/10
3150/3150 [==============================] - 409s 130ms/step - loss: 0.3056 - accuracy: 0.8967 - val_loss: 0.3023 - val_accuracy: 0.8786
Epoch 6/10
3150/3150 [==============================] - 407s 129ms/step - loss: 0.1620 - accuracy: 0.9544 - val_loss: 0.0928 - val_accuracy: 0.9756
Epoch 7/10
3150/3150 [==============================] - 405s 129ms/step - loss: 0.2836 - accuracy: 0.9052 - val_loss: 0.2897 - val_accuracy: 0.8978
Epoch 8/10
3150/3150 [==============================] - 409s 130ms/step - loss: 0.2718 - accuracy: 0.9130 - val_loss: 0.2266 - val_accuracy: 0.9636
Epoch 9/10
3150/3150 [==============================] - 405s 129ms/step - loss: 0.2755 - accuracy: 0.9339 - val_loss: 0.2061 - val_accuracy: 0.9687
Epoch 10/10
3150/3150 [==============================] - 408s 129ms/step - loss: 0.2335 - accuracy: 0.9458 - val_loss: 0.2024 - val_accuracy: 0.9705
/n
Accuracy: 71.15%
```

Hình 13: Kết quả sau khi huấn luyện mô hình RNN-IDS với "tanh"

Độ chính xác trên tập dữ liệu huấn luyện lúc này đã đạt rất cao, khoảng 86-95% ở các epoch, và trên tập dữ liệu kiểm tra là khoảng 71.15%, như vậy là nhóm đã xây dựng được một mô hình RNN-IDS với tỉ lệ chính xác cao.

3.2. Outcome Explanation

Bộ dataset đã được xử lí ở trên sẽ là đầu vào cho model đánh giá sau. Model được xây dựng từ 2 phần theo bài báo LEMNA:

Phần 1: Xây dựng và train model bằng RNN

Ở bài này thì model RNN được cấu hình nhau sau:

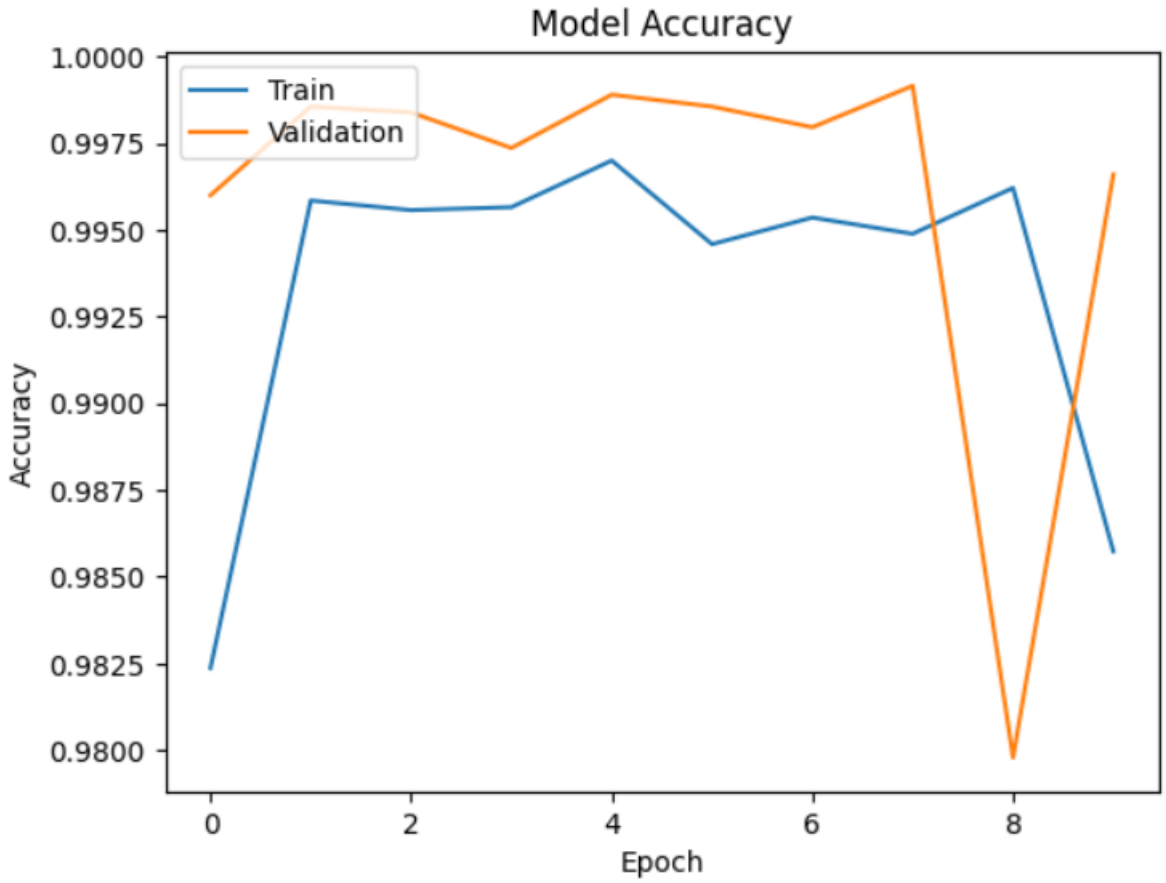
```
# Cấu hình mô hình
model = Sequential()
model.add(Dense(1024, input_dim=122, activation='relu'))
model.add(Dense(1024, activation='relu'))
model.add(Dense(1024, activation='sigmoid'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Fit mô hình và lưu lịch sử huấn luyện
history = model.fit(x, y, epochs=10, batch_size=10, validation_split=0.2)

# Vẽ đồ thị cho độ chính xác
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()

# Vẽ đồ thị cho giá trị mất mát
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()

y_pred = model.predict(xtest)
y_pred = list(map(lambda x: x>=0.5, y_pred))
accuracy_score(ytest, y_pred)
pickle.dump(model, open('classifier.sav', 'wb'))

# Final evaluation of the model
model.summary()
scores = model.evaluate(xtest, ytest, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

Nhóm em sau khi thực hiện xây dựng model thì có lưu nó vào file classifier.sav như bài báo LEMNA.

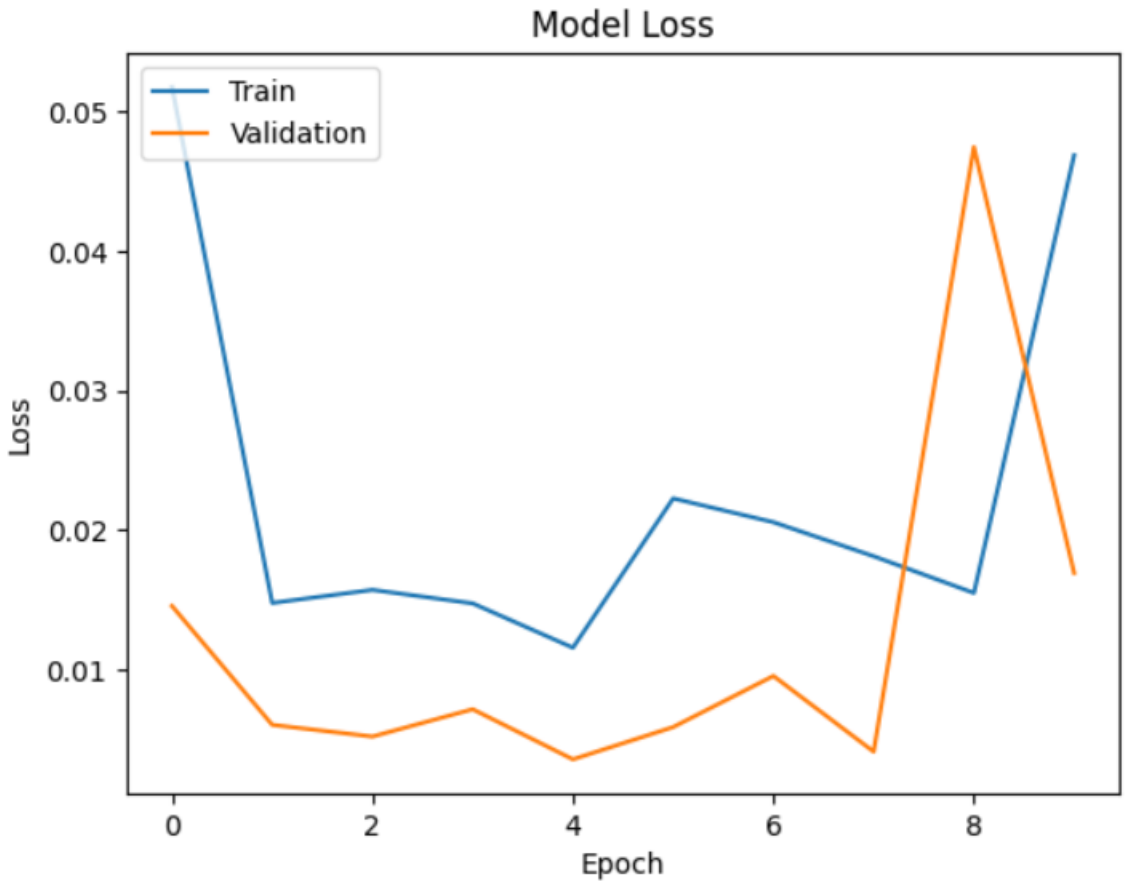
Kết quả sau khi train thì nhóm em thấy tỉ lệ khá tốt:

```
Epoch 1/10
4691/4691 [==============================] - 52s 11ms/step - loss: 0.0518 - accuracy: 0.9824 - val_loss: 0.0145 - val_accuracy: 0.9960
Epoch 2/10
4691/4691 [==============================] - 40s 8ms/step - loss: 0.0148 - accuracy: 0.9958 - val_loss: 0.0060 - val_accuracy: 0.9986
Epoch 3/10
4691/4691 [==============================] - 40s 9ms/step - loss: 0.0157 - accuracy: 0.9956 - val_loss: 0.0052 - val_accuracy: 0.9984
Epoch 4/10
4691/4691 [==============================] - 39s 8ms/step - loss: 0.0147 - accuracy: 0.9957 - val_loss: 0.0071 - val_accuracy: 0.9974
Epoch 5/10
4691/4691 [==============================] - 40s 9ms/step - loss: 0.0116 - accuracy: 0.9970 - val_loss: 0.0035 - val_accuracy: 0.9989
Epoch 6/10
4691/4691 [==============================] - 40s 8ms/step - loss: 0.0223 - accuracy: 0.9946 - val_loss: 0.0059 - val_accuracy: 0.9986
Epoch 7/10
4691/4691 [==============================] - 39s 8ms/step - loss: 0.0206 - accuracy: 0.9954 - val_loss: 0.0095 - val_accuracy: 0.9980
Epoch 8/10
4691/4691 [==============================] - 40s 8ms/step - loss: 0.0181 - accuracy: 0.9949 - val_loss: 0.0041 - val_accuracy: 0.9991
Epoch 9/10
4691/4691 [==============================] - 39s 8ms/step - loss: 0.0155 - accuracy: 0.9962 - val_loss: 0.0475 - val_accuracy: 0.9798
Epoch 10/10
4691/4691 [==============================] - 39s 8ms/step - loss: 0.0469 - accuracy: 0.9857 - val_loss: 0.0169 - val_accuracy: 0.9966
```

Hình 14: Kết quả huấn luyện mô hình RNN của Outcome Explanation



Hình 15: Biểu đồ Accuracy của Outcome Explanation – Train



Hình 16: Biểu đồ Loss của Outcome Explanation – Train

```
402/402 [==============================] - 1s 4ms/step
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_12 (Dense)            (None, 1024)              125952

 dense_13 (Dense)            (None, 1024)              1049600

 dense_14 (Dense)            (None, 1024)              1049600

 dense_15 (Dense)            (None, 1)                 1025

=================================================================
Total params: 2226177 (8.49 MB)
Trainable params: 2226177 (8.49 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Accuracy: 98.63%
```

Hình 17: Thông số khi lưu model và kết quả Accuracy của Outcome Explanation - Test

Sau đó để chuẩn bị cho việc đánh giá model nhóm sẽ trích xuất những thông tin như feature_names, feature_maxs và feature_means. Những biến trên sẽ là tham số cho model đánh giá bên dưới.

```
feature_means = []
feature_maxs = []
feature_names = []


def categorical_helper_fun(x):
    if isinstance(x, int) or isinstance(x,float):
        return int(x)
    else:
        return 0

for i in newdf:
    if True in newdf[i]:
        newdf[i] = list(map(lambda x: categorical_helper_fun(x), newdf[i]))
    feature_means.append(np.mean(newdf[i]))
    feature_maxs.append(max(newdf[i]))

for i in newdf:
    feature_names.append(i)
```

Chuẩn bị thêm hàm data_inverse để tạo ra 1 neighborhood xung quanh 1 dự đoán cụ thể. Nó tạo ra 1 dữ liệu giải lập để nhìn trực quan hơn xung quanh 1 điểm của dữ liệu. Hàm này phục vụ cho LIME 1 kĩ thuật giải thích được sử dụng trong LEMNA.

```
def data_inverse(l, data_row, num_samples):
    """Generates a neighborhood around a prediction.

    For numerical features, perturb them by sampling from a Normal(0,1) and
    doing the inverse operation of mean-centering and scaling, according to
    the means and stds in the training data. For categorical features,
    perturb by sampling according to the training distribution, and making
    a binary feature that is 1 when the value is the same as the instance
    being explained.

    Args:
        data_row: 1d numpy array, corresponding to a row
        num_samples: size of the neighborhood to learn the linear model

    Returns:
        A tuple (data, inverse), where:
            data: dense num_samples * K matrix, where categorical features
            are encoded with either 0 (not equal to the corresponding value
            in data_row) or 1. The first row is the original instance.
            inverse: same as data, except the categorical features are not
            binary, but categorical (as the original data)
    """
    data = np.zeros((num_samples, data_row.shape[0]))
    categorical_features = range(data_row.shape[0])
    if l.discretizer is None:
        data = l.random_state.normal(
            0, 1, num_samples * data_row.shape[0]).reshape(
            num_samples, data_row.shape[0])
        if l.sample_around_instance:
            data = data * l.scaler.scale_ + data_row
        else:
            data = data * l.scaler.scale_ + l.scaler.mean_
        categorical_features = l.categorical_features
        first_row = data_row
    else:
        first_row = l.discretizer.discretize(data_row)
    data[0] = data_row.copy()
    inverse = data.copy()
    for column in categorical_features:
        values = l.feature_values[column]
```

```
        freqs = l.feature_frequencies[column]
        inverse_column = l.random_state.choice(values, size=num_samples,
                                               replace=True, p=freqs)
        binary_column = np.array([1 if x == first_row[column]
                                  else 0 for x in inverse_column])
        binary_column[0] = 1
        inverse_column[0] = data[0, column]
        data[:, column] = binary_column
        inverse[:, column] = inverse_column
    if l.discretizer is not None:
        inverse[1:] = l.discretizer.undiscretize(inverse[1:])
    inverse[0] = data_row
    return data, inverse
```

Phần 2: Đánh giá model vừa train

Về phần này mô hình LEMNA sử dụng 2 phương pháp và mô hình là LIME (Local Interpretable Model-agnostic Explanations) và GMM (Gaussian Mixture Model):

- LIME là một phương pháp để giải thích các dự đoán của mô hình học máy một cách dễ hiểu. LIME tập trung vào việc cung cấp giải thích cục bộ cho các dự đoán riêng lẻ thay vì cố gắng giải thích toàn bộ mô hình.

- GMM là một mô hình thống kê dựa trên việc kết hợp các phân phối Gaussian (phân phối chuẩn) khác nhau theo cách có trọng số để mô hình hóa sự phân bố của dữ liệu. Mỗi Gaussian trong hỗn hợp (mixture) đại diện cho một "nhóm" hoặc "cụm" trong dữ liệu.

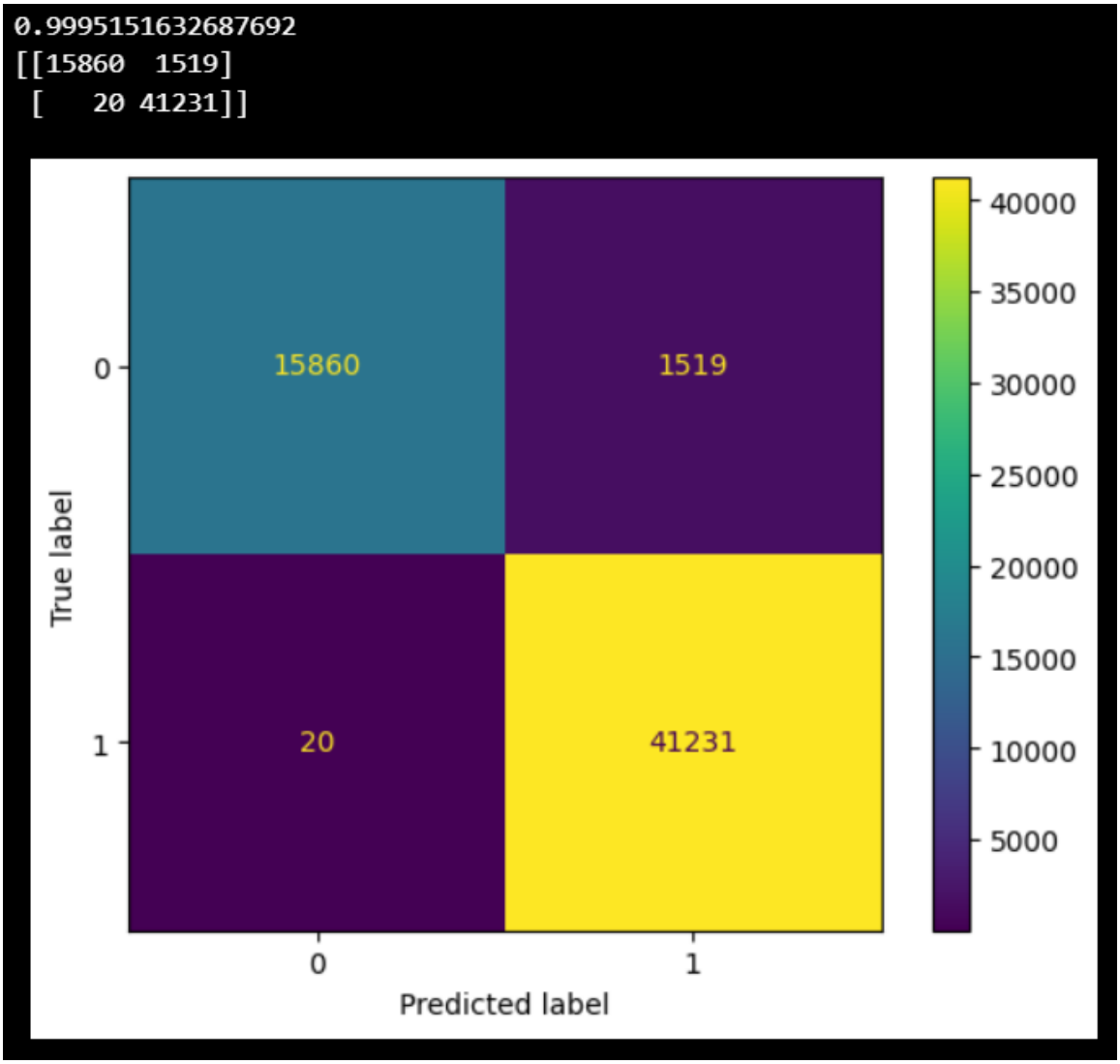
```
l = lime.lime_tabular.LimeTabularExplainer(np.array(x), feature_names =
feature_names, class_names=['True', 'False'], discretize_continuous=True)
a = data_inverse(l, x.iloc[0], 1000)

gm = GaussianMixture(n_components=2, covariance_type = 'diag')

p_data_train = a[1]

p_data_test = list(map(lambda x: x>=0.5, model.predict(p_data_train)))

gm.fit(p_data_train, p_data_test)

components = gm.precisions_
```

Kết quả:

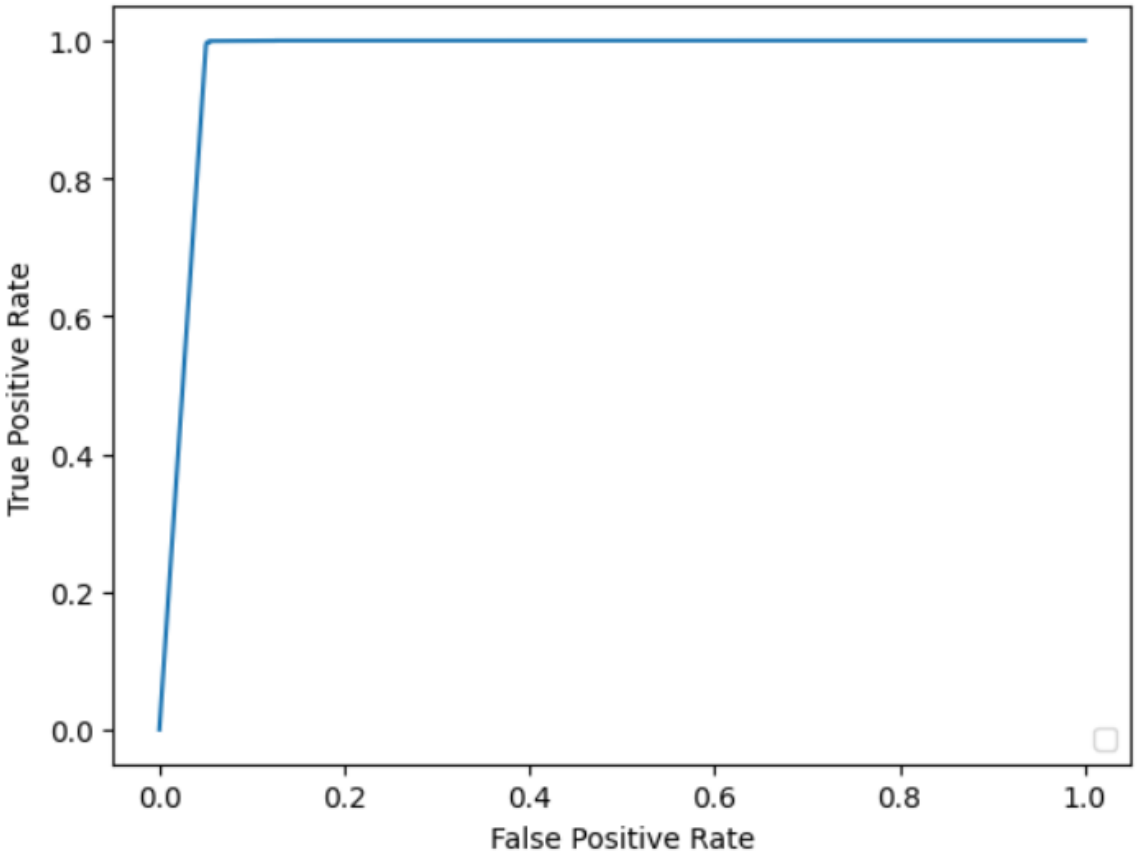
Đánh giá tập dữ liệu train:

```
print(recall(o1,n1))
print(confusionMatrix(o1, n1))

for i in range(len(o1)):
    if o1[i]==True:
        o1[i]=1
    else:
        o1[i]=0

roc(o1, n_prob)
```

```
0.9995151632687692
[[15860  1519]
 [   20 41231]]
```

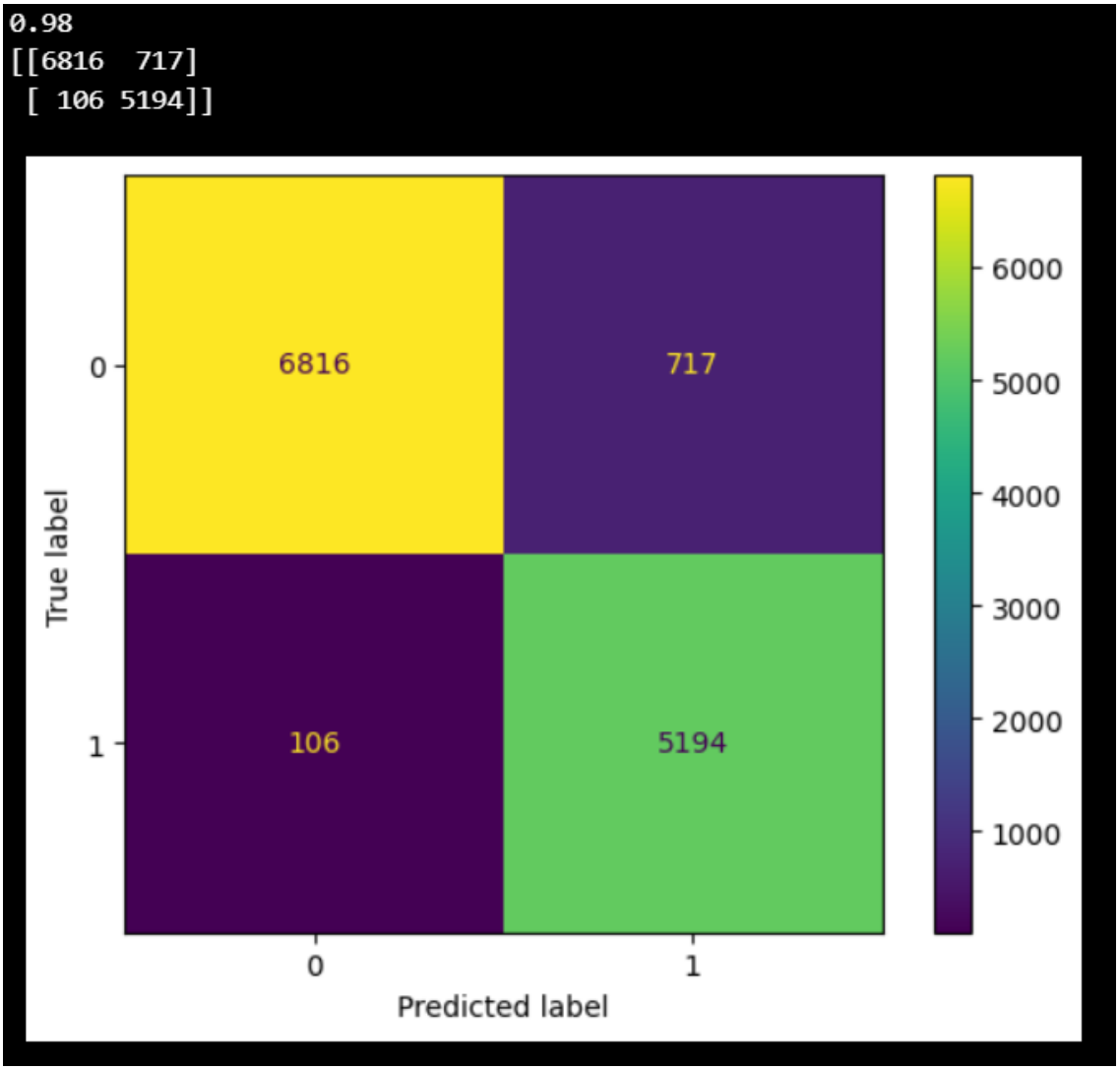


Hình 18: Accuracy và Confusion Matrix

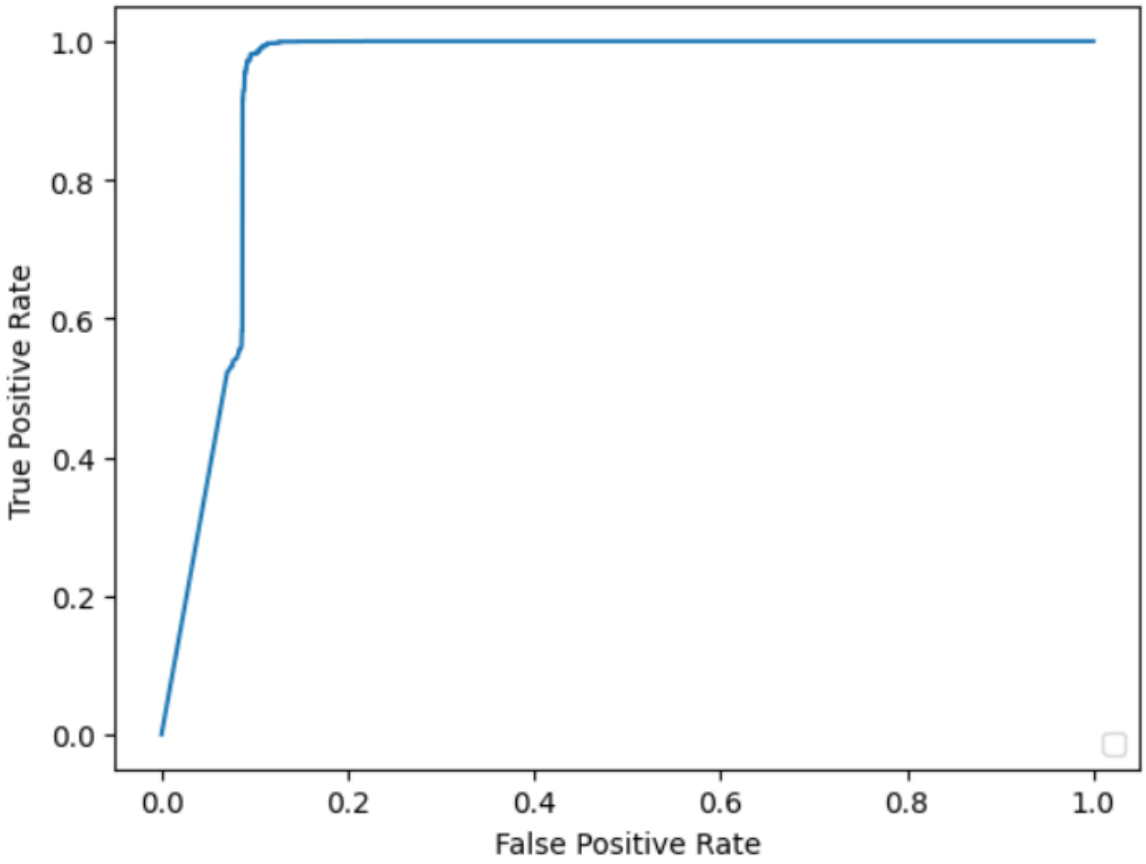


Hình 19: Đồ thị ROC

Dùng để đánh giá bộ dữ liệu test thì kết quả như sau:



Hình 20: Accuracy và Confusion Matrix



Hình 21: Biểu đồ ROC

LIME dự đoán:

```
y_pred=model.predict(p_data_test)

gm_pred1=[]
gm_pred2=[]
def LocalApproxAccuracy():
    for i in range(len(y_pred)):
        gm_pred1.append(gm.predict_proba(p_data_test)[i][1])
        gm_pred2.append(gm.predict_proba(p_data_test)[i][0])
    rmse1=mean_squared_error(y_pred,gm_pred1)
    rmse2=mean_squared_error(y_pred,gm_pred2)
    return(min(rmse1,rmse2))

#LIME Prediction
pred1=[]
for i in tqdm(range(len(y_pred))):
    pred1.append(l.explain_instance(p_data_test[i],gm.predict_proba,
num_features=5).predict_proba[0])
rmse=mean_squared_error(y_pred,pred1)
```

```
LocalApproxAccuracy()
```

Kết quả là mean_squared_error khá cao:

```
32/32 [==============================] - 0s 2ms/step
100%|██████████| 1000/1000 [19:17<00:00,  1.16s/it]

0.3992988372330793
```

Hình 22: Kết quả MSE của LIME predict

3.3. Policy Generation

Giả sử IDS xác định một hoạt động mạng bất thường liên quan đến trường SYN trong gói TCP đã được giải thích ở bước Outcome Explanation. Hoạt động mạng bất thường này được giả định là dấu hiệu của cuộc tấn công Neptune.

Từ các giải thích có được, các quản trị viên của hệ thống mạng sẽ phân tích và xác định được kết quả sẽ là dấu hiệu của cuộc tấn công Neptune, từ đó xác định các thông tin sau để có thể xây dựng chính sách truy cập:

- Địa chỉ IP nguồn là 192.168.1.2.
- Địa chỉ IP đích là 192.168.1.3.
- Loại giao thức là TCP.
- Cờ TCP có giá trị 0x02 (SYN flag set).

Từ đó, chính sách truy cập được tạo ra sẽ như sau:

<filters=(src_ip=192.168.1.2, dst_ip=192.168.1.3, ip_proto=6, tcp_flags=0x02), actions=(drop)>

Chính sách này sẽ chặn tất cả các gói TCP có địa chỉ IP nguồn là 192.168.1.2, địa chỉ IP đích là 192.168.1.3 và cờ TCP có giá trị 0x02.

Các trường của filters

Trường	Mô tả
field	Tên của tính năng mạng.
value	Giá trị của tính năng mạng.

operator	Toán tử so sánh giữa giá trị và ngưỡng.
threshold	Ngưỡng của tính năng mạng.
direction	Hướng của lưu lượng truy cập mạng.
source	Địa chỉ IP nguồn.
destination	Địa chỉ IP đích.
port	Cổng.
protocol	Giao thức.
source_port	Cổng nguồn.
destination_port	Cổng đích.
vlan	ID VLAN.
mac_address	Địa chỉ MAC.
interface	Tên giao diện.
application	Tên ứng dụng.
session_id	ID phiên.
priority	Mức độ ưu tiên.

Các trường của actions

Trường	**Mô tả**
action	Hành động cần thực hiện.

parameters	Các tham số của hành động.
reason	Lý do thực hiện hành động.
log	Nội dung nhật ký.
next	Đường dẫn đến chính sách tiếp theo.
timeout	Thời gian chờ cho hành động.

Lưu ý:

- Các trường bổ sung của filters và actions có thể được hỗ trợ hoặc không tùy thuộc vào nền tảng SDN mà ta đang sử dụng.
- Ta nên tham khảo tài liệu của nền tảng SDN đang sử dụng để biết thêm thông tin về các trường bổ sung này.

Một số ví dụ về cách sử dụng các trường bổ sung của filters và actions:

- Sử dụng trường "direction" để chỉ định hướng của lưu lượng truy cập mạng. Ví dụ: "direction": "inbound" sẽ chỉ định chỉ áp dụng chính sách cho lưu lượng truy cập mạng đến.
- Sử dụng trường "source" để chỉ định địa chỉ IP nguồn của lưu lượng truy cập mạng. Ví dụ: "source": "192.168.1.2" sẽ chỉ định chỉ áp dụng chính sách cho lưu lượng truy cập mạng đến từ địa chỉ IP 192.168.1.2.
- Sử dụng trường "destination" để chỉ định địa chỉ IP đích của lưu lượng truy cập mạng. Ví dụ: "destination": "192.168.1.3" sẽ chỉ định chỉ áp dụng chính sách cho lưu lượng truy cập mạng đi đến địa chỉ IP 192.168.1.3.
- Sử dụng trường "port" để chỉ định cổng của lưu lượng truy cập mạng. Ví dụ: "port": "8080" sẽ chỉ định chỉ áp dụng chính sách cho lưu lượng truy cập mạng đến cổng 8080.
- Sử dụng trường "protocol" để chỉ định giao thức của lưu lượng truy cập mạng. Ví dụ: "protocol": "tcp" sẽ chỉ định chỉ áp dụng chính sách cho lưu lượng truy cập mạng sử dụng giao thức TCP.

Ta có thể sử dụng các trường bổ sung của filters và actions để tạo ra các chính sách kiểm soát truy cập mạng linh hoạt và chính xác hơn.

Chương 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Bài báo đã giới thiệu một phương pháp giải thích kết quả của mô hình abnomaly-based IDS thông qua một mô hình có thể diễn giải được và tạo ra các chính sách kiểm soát truy cập mạng dựa trên các kết quả giải thích đó. Nhóm đã triển khai xây dựng thành công mô hình abnomaly-based IDS bằng Recurrent Neural Network và đạt được tỉ lệ chính xác cao. Với mô hình diễn giải kết quả của hệ thống IDS (Outcome Explanation), tuy nhóm không triển khai được mô hình với kiến trúc và cơ sở lý thuyết như bài báo đề cập, tuy nhiên nhóm vẫn xây dựng được một mô hình Outcome Explanation khác với chức năng tương đương để có thể giải thích được kết quả của mô hình IDS, từ đó có thể xây dựng được các chính sách truy cập mạng dựa trên kết quả giải thích của mô hình Outcome Explanation.

Sau khi tìm hiểu, xây dựng mô hình theo nội dung bài báo và đạt dược các kết quả, nhóm nhận thấy rằng đây là một mô hình đầy tiềm năng, mang lại khả năng xây dựng được các chính sách truy cập mạng hoàn chỉnh và đầy đủ cho hệ thống mạng nói chung và mạng mô hình mạng SDN nói riêng, mô hình của các tác giả đã giải quyết được những điểm yếu, thiếu sót trong hệ thống mạng SDN truyền thống, giúp các hệ thống này có thể phát hiện được các hành vi bất thường trong mạng, đặc biệt có thể có khả năng phát hiện được các lỗi bảo mật zero-day. Tuy nhiên, hệ thống mà các tác giả đề xuất và xây dựng hiện tại chỉ mới là hệ thống thử nghiệm, các thành phần trong hệ thống còn đang được xây dựng một cách riêng lẻ, bộ dữ liệu dùng để xây dựng, huấn luyện và kiểm tra chưa giống với thực tế. Bên cạnh đó các tác giả cũng chưa đề cập đến việc đưa mô hình trong bài báo lên mạng SDN thực tế. Chiến lược giải thích kết quả của mô hình IDS hiện nay trong Outcome Explanation cần được cải thiện để có thể giải thích kết quả rõ ràng hơn. Mô hình hiện tại cũng chỉ giải thích cho từng record cố định, vì vậy có thể bỏ qua các hành vi hay các cuộc tấn công mà phải phân tích luồng nhiều gói tin để xác định.

Từ các nhận xét mà nhóm rút ra được, mô hình trong bài báo có thể cải thiện hơn để đạt được hiệu quả tốt hơn. Mô hình nên được tự động hoá quá trình Policy Generation, từ đó làm cho toàn bộ mô hình trở nên tự động hoá, không cần đến sự can thiệp của người quản trị mạng để xây dựng các chính sách truy cập. Sau khi hoàn thiện, mô hình sẽ có thể áp dụng lên mạng SDN thực tế.

TÀI LIỆU THAM KHẢO

(2017). *A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks.*

(2019). *Enabling Dynamic Network Access Control with Anomaly-based IDS and SDN.*

(2018). *LEMNA: Explaining Deep Learning based Security Applications.*