ĐẠI HỌC BÁCH KHOA HÀ NỘI

Trường Công nghệ thông tin và Truyền thông

BÁO CÁO PROJECT 3

Đề tài: Nghiên cứu và xây dựng hệ thống phát hiện xâm nhập mạng sử dụng kỹ thuật Machine Learning

Sinh viên: Nguyễn Trung Đức

MSSV: 20225288

GVHD: PGS. TS. Ngô Quỳnh Thu

Mã học phần: IT3943

Mã lớp: 755578

Mục lục

LỜI CẢM ƠN **2**

TÓM TẮT **3**

CHƯƠNG 1: MỞ ĐẦU **4**

1.1. Đặt vấn đề 4

1.2. Mục tiêu đề tài 4

1.3. Phạm vi nghiên cứu 5

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ CÔNG CỤ **6**

2.1. Tổng quan về IDS 6

2.2. Các thuật toán Machine Learning sử dụng 8

2.3. Các chỉ số đánh giá 12

CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG **14**

3.1. Quy trình tổng thể 14

3.2. Bộ dữ liệu CIC-IDS2017 16

3.3. Kĩ thuật Tiền xử lý dữ liệu 18

CHƯƠNG 4: CÀI ĐẶT VÀ THỰC NGHIỆM **21**

4.1. Môi trường cài đặt 21

4.2. Kịch bản thử nghiệm 22

4.3. Kết quả thực nghiệm và đánh giá 24

4.4. Thảo luận 27

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN **28**

5.1. Kết luận 28

5.2. Hạn chế và hướng phát triển 29

TÀI LIỆU THAM KHẢO ..... **Error! Bookmark not defined.**

LỜI CẢM ƠN

Trước hết, em xin gửi lời cảm ơn chân thành đến giảng viên hướng dẫn, cô Ngô Quỳnh Thu đã tận tình hướng dẫn, chỉ dẫn em trong suốt quá trình thực hiện đề tài *"Nghiên cứu và xây dựng hệ thống phát hiện xâm nhập mạng sử dụng kỹ thuật Machine Learning"*. Những kiến thức, góp ý và kinh nghiệm quý báu của thầy/cô đã giúp em định hướng nghiên cứu và hoàn thiện luận văn này.

Đặc biệt, em xin cảm ơn Canadian Institute for Cybersecurity đã công bố bộ dữ liệu CIC-IDS2017, một nguồn dữ liệu quan trọng giúp em huấn luyện và đánh giá hiệu quả các mô hình Machine Learning trong đề tài.

Mặc dù đã nỗ lực hết mình, nhưng luận văn vẫn còn nhiều hạn chế. Em rất mong nhận được những góp ý từ thầy/cô để có thể hoàn thiện hơn trong tương lai.

Xin chân thành cảm ơn!

TÓM TẮT

Trong bối cảnh các cuộc tấn công mạng ngày càng gia tăng về quy mô và mức độ tinh vi, các hệ thống tường lửa truyền thống không còn đủ khả năng để đảm bảo an toàn thông tin. Đồ án này tập trung nghiên cứu và ứng dụng các thuật toán Học máy (Machine Learning) để xây dựng một hệ thống phát hiện xâm nhập (IDS) có khả năng nhận diện các hành vi bất thường trong lưu lượng mạng.

Nghiên cứu sử dụng bộ dữ liệu chuẩn CIC-IDS2017, tích hợp ba tập dữ liệu con đại diện cho ba hình thái tấn công phổ biến nhất hiện nay là: Dò quét cổng (PortScan), Vét cạn mật khẩu (Brute Force) và Tấn công từ chối dịch vụ (DDoS). Dữ liệu sau khi được tiền xử lý và gộp lại đã được đưa vào huấn luyện và kiểm thử trên ba mô hình: Logistic Regression, Decision Tree và Random Forest.

Kết quả thực nghiệm cho thấy cả ba mô hình đều có khả năng phân loại tốt, trong đó Decision Tree và Random Forest đạt hiệu suất vượt trội. Đáng chú ý, thuật toán Decision Tree đã chứng minh là giải pháp tối ưu nhất cho bài toán này với độ nhạy (Recall) đạt 99.98%, chỉ bỏ sót 2 trường hợp tấn công trên tổng số hơn 12.000 mẫu thử nghiệm (thấp hơn so với 11 trường hợp của Random Forest).

Từ kết quả trên, đồ án đề xuất sử dụng Decision Tree làm mô hình lõi cho hệ thống IDS. Giải pháp này không chỉ đảm bảo độ an toàn cao nhờ khả năng "bắt dính" các dấu hiệu tấn công, mà còn tối ưu hóa tài nguyên hệ thống nhờ tốc độ xử lý thời gian thực và khả năng minh bạch hóa các quyết định phân loại.

CHƯƠNG 1: MỞ ĐẦU

1.1. Đặt vấn đề

Trong kỷ nguyên số hóa hiện nay, an ninh mạng đang trở thành thách thức sống còn đối với mọi tổ chức. Các phương thức tấn công mạng (Cyber Attacks) ngày càng tinh vi, đa dạng và biến đổi khôn lường, khiến các hệ thống phòng thủ truyền thống dựa trên luật (Rule-based) hoặc chữ ký (Signature-based) dần trở nên lạc hậu. Các hệ thống cũ này thường bất lực trước các dạng tấn công mới (Zero-day) hoặc các biến thể chưa từng được định nghĩa.

Điều này đặt ra nhu cầu cấp thiết về một hệ thống phát hiện xâm nhập (Intrusion Detection System - IDS) thông minh hơn, có khả năng "học" từ dữ liệu lịch sử để tự động nhận diện các hành vi bất thường mà không cần lập trình cứng nhắc.

1.2. Mục tiêu đề tài

1.2.1. Mục tiêu tổng quát

Nghiên cứu và xây dựng hệ thống phát hiện xâm nhập mạng (Intrusion Detection System - IDS) dựa trên kỹ thuật Học máy (Machine Learning), nhằm tự động hóa việc nhận diện và phân loại các hành vi tấn công mạng với độ chính xác cao, khắc phục nhược điểm của các hệ thống IDS truyền thống.

1.2.2. Mục tiêu cụ thể

Để đạt được mục tiêu tổng quát nêu trên, đồ án tập trung thực hiện các nhiệm vụ cụ thể sau:

- Nghiên cứu cơ sở lý thuyết:
    - Tìm hiểu về các phương thức tấn công mạng phổ biến (Brute Force, DDoS, Port Scan).
    - Nghiên cứu nguyên lý hoạt động của các thuật toán Học máy: Logistic Regression, Decision Tree và tập trung sâu vào Random Forest.
- Thu thập và Xử lý dữ liệu (Data Preprocessing):
    - Phân tích cấu trúc bộ dữ liệu chuẩn CIC-IDS2017.

  - Thực hiện quy trình làm sạch dữ liệu: Xử lý giá trị thiếu (Null), giá trị vô cực (Infinity) và loại bỏ dữ liệu trùng lặp.
  - Trích xuất đặc trưng và chuẩn hóa dữ liệu (Scaling) để tối ưu hóa đầu vào cho mô hình.
- Xây dựng và Huấn luyện mô hình:
  - Cài đặt và huấn luyện 3 mô hình: Logistic Regression, Decision Tree và Random Forest trên môi trường Python.
  - Tối ưu hóa tham số cho thuật toán Random Forest để đạt hiệu suất cao nhất.
- Thực nghiệm và Đánh giá hiệu quả:
  - So sánh hiệu năng giữa các thuật toán dựa trên các chỉ số: Accuracy (Độ chính xác), Precision (Độ chính xác dự báo), Recall (Độ nhạy) và F1-Score.
  - Phân tích Ma trận nhầm lẫn (Confusion Matrix) để đánh giá khả năng giảm thiểu tỷ lệ bỏ lọt tấn công (False Negative) - yếu tố quan trọng nhất trong an ninh mạng.

1.3. Phạm vi nghiên cứu

Trong khuôn khổ của đồ án, đề tài tập trung nghiên cứu và xây dựng hệ thống phát hiện xâm nhập mạng dựa trên kĩ thuật Machine Learning với phạm vi cụ thể như sau:

- Đối tượng nghiên cứu: Hệ thống phát hiện xâm nhập mạng (Network-based Intrusion Detection System – IDS) dựa trên phương pháp phát hiện bất thường.
- Phạm vi dữ liệu: Đề tài sử dụng bộ dữ liệu CIC-ISD2017 (Canadian Institute for Cybersecurity Intrusion Detection System 2017) để huấn luyện và đánh giá mô hình. Dữ liệu phản ánh các loại lưu lượng mạng bao gồm lưu lượng bình thường và một số dạng tấn công phổ biến.
- Phạm vi phương pháp: Đề tài tập trung áp dụng các thuật toán Machine Learning có giám sát như Logistic Regression, Decision Tree và Random Forest để xây dựng mô hình phát hiện xâm nhập. Các phương

pháp học sâu (Deep Learning) không được xem xét trong phạm vi nghiên cứu này.

- Phạm vi chức năng hệ thống: Hệ thống tập trung giải quyết bài toán Phân loại nhị phân (Binary Classification), phân biệt lưu lượng mạng thành hai nhãn: Bình thường (Benign) và Tấn công (Attack). Chức năng của hệ thống dừng lại ở việc phân tích và đánh giá dữ liệu ngoại tuyến (Offline Analysis) dựa trên tập dữ liệu lịch sử. Các chức năng liên quan đến bắt gói tin (Packet Capturing) và cảnh báo thời gian thực (Real-time Alerting) nằm ngoài phạm vi của đề tài này.
- Phạm vi môi trường triển khai: Mô hình được xây dựng, huấn luyện và kiểm thử trên môi trường thực nghiệm Google Colab, sử dụng ngôn ngữ lập trình Python cùng các thư viện mã nguồn mở như Scikit-learn, Pandas. Đề tài được giới hạn ở mức độ Mô hình thử nghiệm (Prototype/Proof of Concept) để đánh giá hiệu quả thuật toán, chưa triển khai tích hợp trên hạ tầng mạng vật lý hoặc các thiết bị định tuyến thực tế.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ CÔNG CỤ

2.1. Tổng quan về IDS

2.1.1. Khái niệm

Theo định nghĩa từ IBM, Hệ thống phát hiện xâm nhập (Intrusion Detection System - IDS) là một công cụ an ninh mạng – bao gồm thiết bị phần cứng hoặc ứng dụng phần mềm – có nhiệm vụ giám sát lưu lượng mạng (network traffic) và các thiết bị nhằm phát hiện các hoạt động độc hại, hành vi đáng ngờ hoặc các vi phạm chính sách bảo mật của tổ chức.

Mục tiêu cốt lõi của IDS là cung cấp khả năng quan sát (visibility) cho quản trị viên. Khi phát hiện mối đe dọa, IDS sẽ thực hiện:

- Gửi cảnh báo (Alerting): Thông báo ngay lập tức cho đội ngũ bảo mật.
- Ghi nhận sự kiện: Chuyển thông tin đến một hệ thống quản lý sự kiện và thông tin bảo mật trung tâm (SIEM) để tổng hợp và phân tích sâu hơn.

2.1.2. Phân loại hệ thống

Hệ thống IDS được chia thành hai loại chính dựa trên vị trí triển khai và đối tượng giám sát:

1. Network Intrusion Detection System (NIDS):
    - *Vị trí:* Được đặt tại các điểm chiến lược trong mạng (thường là sau tường lửa hoặc tại các switch cốt lõi) để giám sát lưu lượng đi qua toàn bộ một phân đoạn mạng (subnet).
    - *Cơ chế*: Phân tích các gói tin (packet) di chuyển trên đường truyền để tìm kiếm dấu hiệu tấn công.
    - Đồ án này tập trung xây dựng một mô hình NIDS, sử dụng dữ liệu lưu lượng mạng (Network Traffic Flow) từ bộ dữ liệu CIC-IDS2017.
2. Host Intrusion Detection System (HIDS):
    - *Vị trí:* Được cài đặt trực tiếp trên từng thiết bị đầu cuối (máy chủ, máy trạm).
    - *Cơ chế:* Giám sát các hoạt động nội bộ của thiết bị như các tệp tin hệ thống, log file, và các tiến trình (processes) đang chạy.

2.1.3. Phương pháp phát hiện

IDS sử dụng hai phương pháp chính để xác định mối đe dọa:

1. Phát hiện dựa trên chữ ký (Signature-based Detection):
    - *Nguyên lý:* So sánh lưu lượng mạng với một cơ sở dữ liệu chứa các "dấu vân tay" (signatures) của các cuộc tấn công đã biết (ví dụ: một chuỗi byte cụ thể trong gói tin độc hại).
    - *Ưu điểm:* Độ chính xác rất cao và tốc độ nhanh đối với các loại tấn công cũ, đã biết.
    - *Hạn chế:* Không thể phát hiện các cuộc tấn công mới chưa từng được ghi nhận (Zero-day attacks) hoặc các biến thể đã được mã hóa.
2. Phát hiện dựa trên bất thường (Anomaly-based Detection):
    - *Nguyên lý:* Sử dụng Học máy (Machine Learning) để xây dựng một mô hình về trạng thái hoạt động "bình thường" của mạng (baseline).

Bất kỳ hành vi nào lệch chuẩn so với trạng thái này (ví dụ: lưu lượng tăng đột biến, truy cập vào cổng lạ) đều bị coi là nghi vấn.

- *Ưu điểm:* Có khả năng phát hiện các cuộc tấn công mới (Zero-day) mà không cần cập nhật chữ ký.
- *Hạn chế:* Có thể sinh ra báo động giả (False Positive) nếu hành vi hợp lệ của người dùng thay đổi đột ngột.
- Đồ án này áp dụng phương pháp này, sử dụng thuật toán Random Forest để học các đặc trưng của lưu lượng mạng và tự động phân loại hành vi bất thường.

2.1.4. Sự khác biệt giữa IDS và IPS

Cần phân biệt rõ IDS với Hệ thống ngăn chặn xâm nhập (Intrusion Prevention System - IPS):

- IDS (Detection): Chỉ tập trung vào giám sát và cảnh báo. Nó giống như một hệ thống camera quan sát thụ động.
- IPS (Prevention): Có khả năng tự động can thiệp (như ngắt kết nối, chặn IP) để ngăn chặn cuộc tấn công ngay khi nó xảy ra.

Trong phạm vi nghiên cứu này, hệ thống được thiết kế dưới dạng một công cụ phân tích và phát hiện (IDS), tập trung vào độ chính xác của việc phân loại hơn là cơ chế chặn tự động.

2.2. Các thuật toán Machine Learning sử dụng

Trong khuôn khổ đồ án, nghiên cứu tập trung vào ba thuật toán đại diện cho các hướng tiếp cận khác nhau: Logistic Regression (Mô hình tuyến tính cơ bản), Decision Tree (Mô hình cây đơn lẻ) và Random Forest (Mô hình tổ hợp). Việc lựa chọn này nhằm mục đích so sánh hiệu quả và tìm ra giải pháp tối ưu nhất cho bài toán phát hiện xâm nhập.

2.2.1. Logistic Regression (Hồi quy Logistic)

a. Khái niệm:

Logistic Regression là một thuật toán học máy có giám sát được sử dụng cho bài toán phân loại (Classification). Mặc dù tên gọi là "Hồi quy", nhưng

nó được dùng để dự đoán xác suất một mẫu dữ liệu thuộc về một lớp nhất định (trong đồ án này là lớp 0: Bình thường hoặc lớp 1: Tấn công).

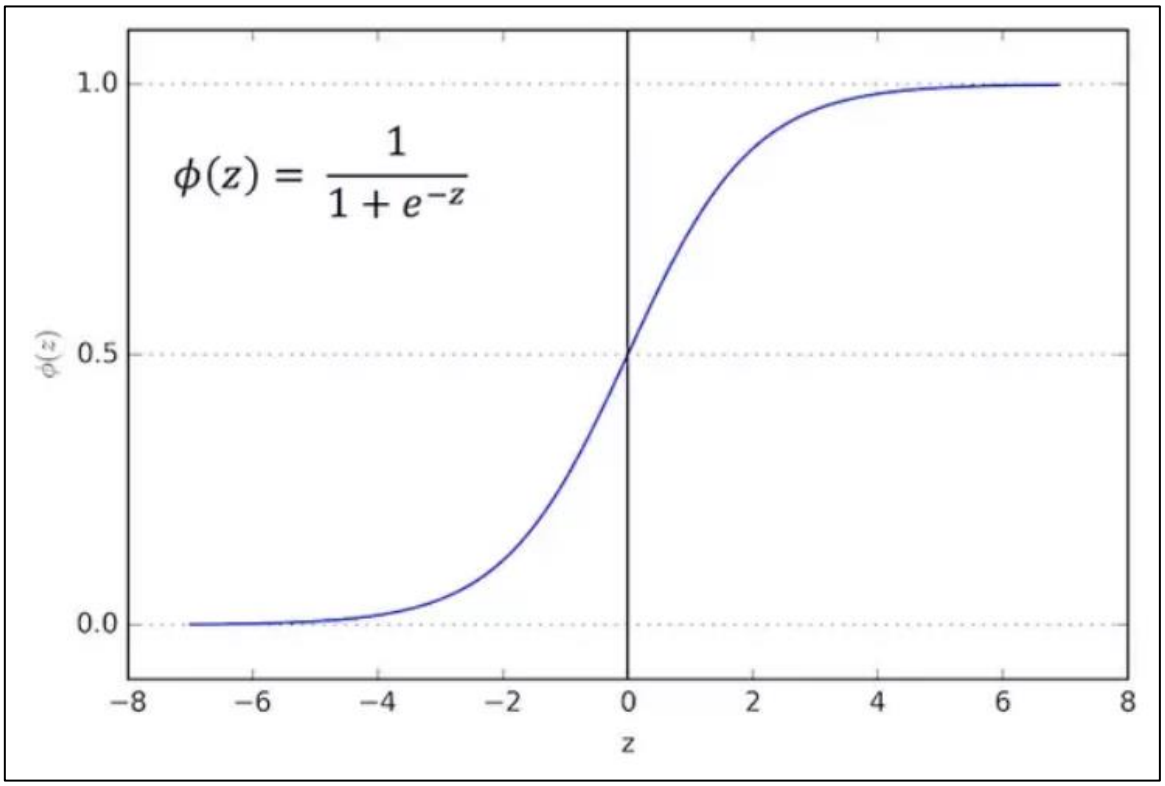
b. Nguyên lý hoạt động:

Thuật toán sử dụng hàm Sigmoid để chuyển đổi đầu ra của phương trình tuyến tính về giá trị xác suất trong khoảng (0, 1). Dựa trên ngưỡng (threshold) – thường là 0.5 – mô hình sẽ quyết định nhãn của dữ liệu.

- Nếu $P(y = 1|x) \geq 0.5$: Dự đoán là Tấn công.
- Nếu $P(y = 1|x) < 0.5$: Dự đoán là Bình thường.

c. Lý do lựa chọn:

Trong đồ án này, Logistic Regression đóng vai trò là mô hình cơ sở (Baseline Model). Vì đây là thuật toán đơn giản, tốc độ huấn luyện nhanh và dễ hiểu, kết quả của nó được dùng làm mốc chuẩn để đánh giá xem các mô hình phức tạp hơn (như Random Forest) có thực sự mang lại hiệu quả vượt trội hay không.



Ảnh 1: Hàm Sigmoid

2.2.2. Decision Tree (Cây quyết định)

a. Khái niệm:

Decision Tree là một thuật toán mô phỏng quá trình ra quyết định của con người dưới dạng sơ đồ cây. Cấu trúc cây bao gồm:

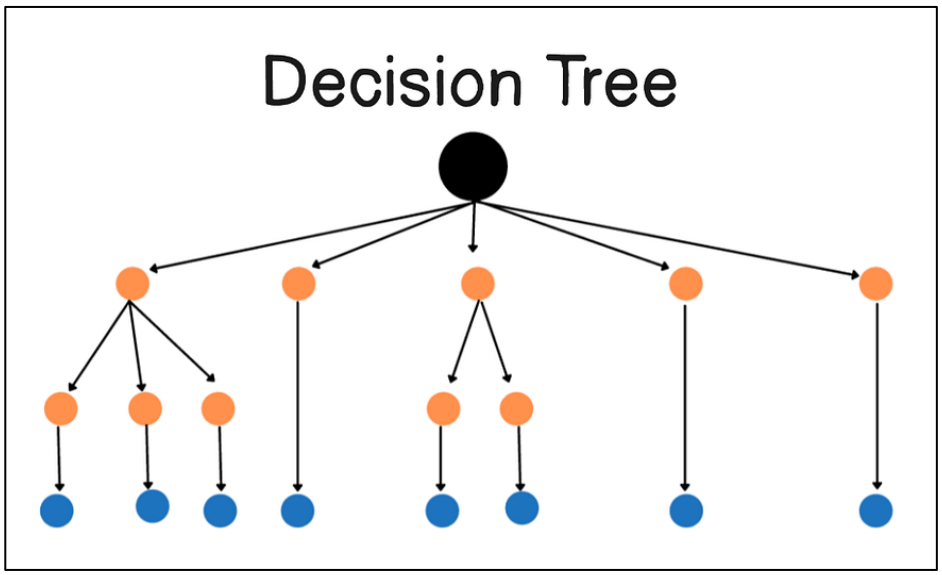
- Nút gốc (Root Node): Chứa toàn bộ dữ liệu.
- Nút quyết định (Decision Node): Nơi thực hiện kiểm tra một điều kiện trên một đặc trưng (ví dụ: "Thời gian gói tin > 5ms?").
- Nút lá (Leaf Node): Chứa kết quả dự đoán cuối cùng (nhãn).

b. Nguyên lý hoạt động:

Quá trình huấn luyện thực chất là quá trình phân chia dữ liệu liên tục dựa trên các thuộc tính sao cho độ tinh khiết (purity) của các nhóm con là cao nhất. Các tiêu chuẩn phân chia phổ biến bao gồm Gini Impurity hoặc Information Gain (Entropy).

c. Ưu và nhược điểm:

- Ưu điểm: Trực quan, dễ giải thích (có thể vẽ ra cây để xem tại sao mô hình lại đưa ra quyết định đó), không yêu cầu quá nhiều bước tiền xử lý dữ liệu.
- Nhược điểm: Rất dễ gặp vấn đề Overfitting (Quá khớp). Cây có thể học quá chi tiết các nhiễu (noise) trong tập huấn luyện, dẫn đến việc hoạt động kém hiệu quả trên dữ liệu thực tế mới.



Ảnh 2: Mô hình Decision Tree

2.2.3. Random Forest (Rừng ngẫu nhiên) - Thuật toán đề xuất

a. Khái niệm:

Random Forest là một thuật toán thuộc nhóm Học máy tổ hợp (Ensemble Learning), cụ thể là phương pháp Bagging (Bootstrap Aggregating). Thay vì chỉ sử dụng một cây quyết định duy nhất, Random Forest xây dựng một "khu rừng" gồm nhiều cây quyết định (ví dụ: 100 cây) hoạt động song song.

b. Nguyên lý hoạt động:

Quá trình dự đoán của Random Forest dựa trên cơ chế "Bầu chọn số đông" (Majority Voting):

1. Mỗi cây trong rừng sẽ đưa ra một dự đoán riêng biệt cho mẫu dữ liệu đầu vào.
2. Mô hình sẽ tổng hợp các kết quả này.
3. Kết quả cuối cùng được chọn là nhãn nhận được nhiều phiếu bầu nhất từ các cây.

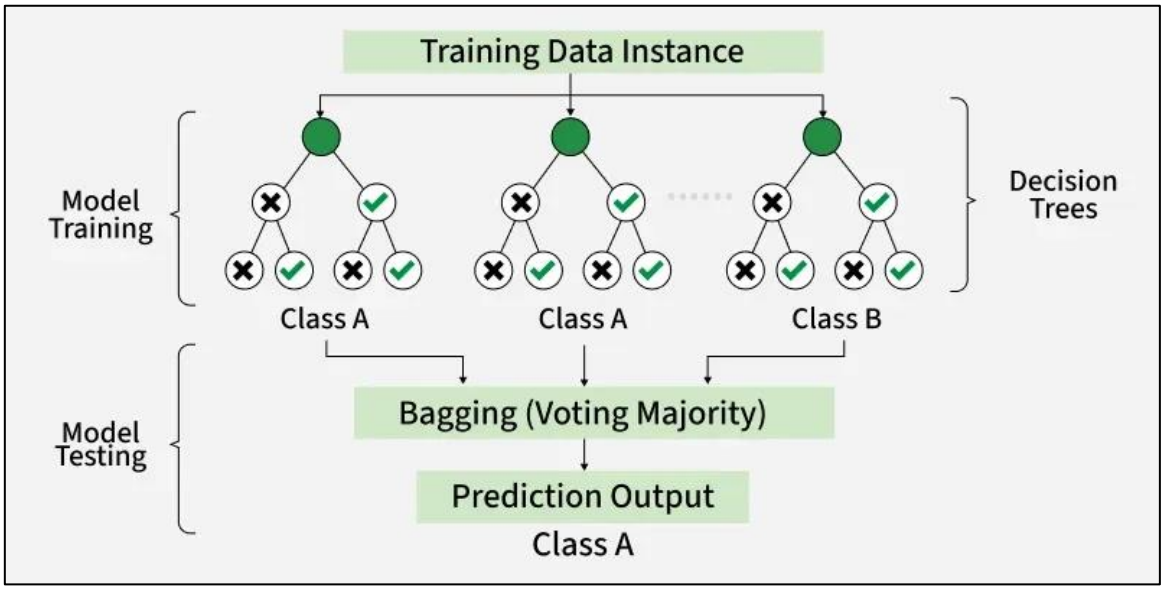
Để đảm bảo các cây không giống hệt nhau, Random Forest sử dụng hai yếu tố ngẫu nhiên:

- Lấy mẫu ngẫu nhiên (Bootstrapping): Mỗi cây được huấn luyện trên một tập con ngẫu nhiên của dữ liệu gốc.
- Chọn đặc trưng ngẫu nhiên: Tại mỗi nút phân chia, cây chỉ được xem xét một tập hợp con ngẫu nhiên các đặc trưng.

c. Lý do lựa chọn:

Đối với bài toán an ninh mạng có dữ liệu lớn và phức tạp như CIC-IDS2017, Random Forest khắc phục được nhược điểm của Decision Tree:

- Chống Overfitting: Nhờ việc kết hợp nhiều cây, sai số của từng cây đơn lẻ sẽ được triệt tiêu, giúp mô hình tổng quát hóa tốt hơn.
- Độ chính xác cao: Thường cho kết quả tốt hơn các mô hình đơn lẻ trên các bộ dữ liệu nhiều chiều.
- Khả năng chịu nhiễu tốt: Ít bị ảnh hưởng bởi các dữ liệu ngoại lai (outliers).



Ảnh 3: Mô hình thuật toán Random Forest

2.3. Các chỉ số đánh giá

Để đánh giá hiệu quả của các mô hình phân loại, đồ án sử dụng tập hợp các chỉ số chuẩn dựa trên Ma trận nhầm lẫn (Confusion Matrix). Đặc biệt đối với bài toán phát hiện xâm nhập, nơi mà hậu quả của việc bỏ sót tấn công là rất nghiêm trọng, các chỉ số như Recall đóng vai trò quan trọng hơn độ chính xác đơn thuần.

2.3.1. Ma trận nhầm lẫn (Confusion Matrix)

Ma trận nhầm lẫn là bảng đối chiếu giữa kết quả dự đoán của mô hình và nhãn thực tế của dữ liệu. Đối với bài toán phân loại nhị phân (0: Bình thường, 1: Tấn công), ma trận này bao gồm 4 thành phần:

- True Positive (TP - Dương tính thật): Số lượng mẫu Tấn công được mô hình dự đoán chính xác là Tấn công. → *Kết quả mong muốn.*
- True Negative (TN - Âm tính thật): Số lượng mẫu Bình thường được mô hình dự đoán chính xác là Bình thường. → *Kết quả mong muốn.*
- False Positive (FP - Dương tính giả): Số lượng mẫu Bình thường bị mô hình nhầm lẫn là Tấn công.
    - *Trong thực tế:* Đây là "Báo động giả". Nó gây phiền nhiễu cho quản trị viên mạng nhưng không gây hại trực tiếp đến hệ thống.

- False Negative (FN - Âm tính giả): Số lượng mẫu Tấn công bị mô hình nhầm lẫn là Bình thường.
    - *Trong thực tế:* Đây là "Bỏ lọt tội phạm". Đây là sai lầm nguy hiểm nhất trong an ninh mạng vì kẻ tấn công có thể xâm nhập hệ thống mà không bị phát hiện.

2.3.2. Các chỉ số định lượng

Từ 4 thành phần trên, các chỉ số sau được tính toán:

a. Độ chính xác (Accuracy)

Là tỷ lệ tổng số mẫu được dự đoán đúng trên tổng số mẫu thực nghiệm.

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

Ý nghĩa: Cho biết cái nhìn tổng quan về hiệu suất. Tuy nhiên, trong bài toán IDS (nơi dữ liệu thường mất cân bằng, số lượng gói tin bình thường nhiều hơn tấn công gấp nhiều lần), chỉ số này có thể gây hiểu lầm.

b. Độ chính xác dự báo (Precision)

Là tỷ lệ số mẫu dự đoán là Tấn công chính xác trên tổng số mẫu được dự đoán là Tấn công.

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

Ý nghĩa: Trả lời câu hỏi: *"Khi hệ thống cảnh báo tấn công, bao nhiêu phần trăm là tấn công thật?"*. Precision cao giúp giảm thiểu tình trạng "Spam cảnh báo" (Alert Fatigue).

c. Độ nhạy (Recall) - Chỉ số quan trọng nhất

Là tỷ lệ số mẫu Tấn công được phát hiện chính xác trên tổng số mẫu Tấn công thực tế.

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

Ý nghĩa: Trả lời câu hỏi: *"Hệ thống đã phát hiện được bao nhiêu phần trăm tổng số cuộc tấn công?"*.

Vai trò trong đồ án: Đây là chỉ số ưu tiên hàng đầu. Một hệ thống IDS tốt phải có Recall tiệm cận 100% để đảm bảo không bỏ lọt mối nguy hiểm (giảm thiểu False Negative).

d. Điểm F1 (F1-Score)

Là trung bình điều hòa giữa Precision và Recall.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Ý nghĩa: F1-Score được sử dụng để tìm kiếm sự cân bằng giữa Precision và Recall. Nó là thước đo tin cậy hơn Accuracy trong trường hợp dữ liệu bị mất cân bằng (Imbalanced Data).

Trong bối cảnh an ninh mạng, chi phí cho một lần báo động giả (FP) thường thấp hơn rất nhiều so với thiệt hại của một cuộc tấn công bị bỏ lọt (FN). Do đó, nghiên cứu này sẽ tập trung tối ưu hóa mô hình để đạt chỉ số Recall cao nhất có thể, chấp nhận đánh đổi một lượng nhỏ Precision nếu cần thiết.

CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

3.1. Quy trình tổng thể

Để xây dựng hệ thống phát hiện xâm nhập hiệu quả, hệ thống được xây dựng theo quy trình học máy tiêu chuẩn (Standard Machine Learning Pipeline) bao gồm 5 giai đoạn chính. Quy trình này đảm bảo dữ liệu đầu vào được xử lý sạch sẽ, nhất quán trước khi đưa vào huấn luyện mô hình.

Sơ đồ tổng quát của hệ thống được mô tả như sau:

Giai đoạn 1: Thu thập dữ liệu (Data Collection)

- Đầu vào: Tập dữ liệu gốc CIC-IDS2017 định dạng .csv (cụ thể là các file Tuesday-WorkingHours.csv, Friday-WorkingHours.csv...).
- Đặc điểm: Dữ liệu chứa các thông tin về luồng mạng (Network Flows) như địa chỉ IP nguồn/đích, cổng, giao thức, độ dài gói tin, thời gian duy trì luồng...

Giai đoạn 2: Làm sạch dữ liệu (Data Cleaning)

Dữ liệu mạng thực tế thường chứa nhiều nhiễu và sai sót. Giai đoạn này thực hiện các bước xử lý thiết yếu:

1. Xử lý tên cột: Loại bỏ các khoảng trắng thừa trong tên cột

   (ví dụ: ‘ Label’ → ‘Label’) để tránh lỗi truy xuất.

2. Xử lý giá trị lỗi: Tìm và xử lý các giá trị vô cực (Infinity) và giá trị rỗng (NaN) phát sinh trong quá trình trích xuất gói tin. Giải pháp áp dụng là loại bỏ các dòng chứa giá trị này để đảm bảo tính toàn vẹn.
3. Khử trùng lặp: Loại bỏ các dòng dữ liệu trùng lặp hoàn toàn (duplicate rows) để tránh việc mô hình bị thiên lệch (bias) do học đi học lại một mẫu quá nhiều lần.

Giai đoạn 3: Tiền xử lý & Trích chọn đặc trưng (Preprocessing & Feature Engineering)

Đây là giai đoạn quan trọng nhất để chuyển đổi dữ liệu thô thành dạng mà máy tính có thể hiểu được:

1. Mã hóa nhãn (Label Encoding): Chuyển đổi cột mục tiêu Label từ dạng văn bản sang dạng số để phục vụ bài toán phân loại nhị phân:
   - BENIGN (Bình thường) → Gán nhãn 0.
   - Các loại tấn công (FTP-Patator, SSH-Patator, DDoS...) → Gán nhãn 1.
2. Chuẩn hóa dữ liệu (Feature Scaling): Sử dụng kỹ thuật StandardScaler để đưa các đặc trưng số về cùng một phân phối chuẩn (mean = 0, std = 1).
   - *Mục đích:* Giúp các đặc trưng có đơn vị lớn (như Flow Duration - hàng triệu micro giây) không lấn át các đặc trưng có đơn vị nhỏ (như Total Fwd Packets - hàng chục gói tin), giúp thuật toán hội tụ nhanh và chính xác hơn.

Giai đoạn 4: Phân chia dữ liệu (Data Splitting)

Tập dữ liệu sau khi xử lý được chia thành 2 phần độc lập theo tỷ lệ 80/20:

- Tập huấn luyện (Training Set - 80%): Dùng để dạy mô hình học các đặc trưng của tấn công và bình thường.
- Tập kiểm thử (Test Set - 20%): Dùng để đánh giá khách quan hiệu suất của mô hình. Mô hình hoàn toàn không được "nhìn thấy" dữ liệu này trong quá trình học.

Giai đoạn 5: Huấn luyện và Đánh giá (Modeling & Evaluation)

- Huấn luyện: Áp dụng các thuật toán (Logistic Regression, Decision Tree, Random Forest) lên tập Training Set.
- Đánh giá: Sử dụng mô hình đã huấn luyện để dự đoán trên tập Test Set. Kết quả được so sánh với nhãn thực tế thông qua Ma trận nhầm lẫn và các chỉ số Precision, Recall, F1-Score.

3.2. Bộ dữ liệu CIC-IDS2017

3.2.1. Tổng quan về bộ dữ liệu

Bộ dữ liệu CIC-IDS2017 (Canadian Institute for Cybersecurity Intrusion Detection System 2017) được phát triển bởi Viện An ninh mạng Canada thuộc Đại học New Brunswick (UNB). Đây được xem là một trong những bộ dữ liệu chuẩn mực và cập nhật nhất hiện nay dành cho việc nghiên cứu phát hiện xâm nhập mạng.

Khác với các bộ dữ liệu cũ (như KDD99 hay NSL-KDD) thường thiếu các mẫu tấn công hiện đại hoặc cấu trúc mạng không thực tế, CIC-IDS2017 mô phỏng đầy đủ các hành vi của người dùng thực và bao gồm các kịch bản tấn công phổ biến nhất hiện nay như: Brute Force, DoS, DDoS, Web Attack, Infiltration, Botnet và Port Scan.

3.2.2. Cấu trúc và Đặc trưng (Features)

Dữ liệu gốc được thu thập dưới dạng PCAP (Packet Capture), sau đó được trích xuất thành định dạng CSV bằng công cụ CICFlowMeter. Mỗi dòng trong bộ dữ liệu đại diện cho một luồng mạng (Network Flow) và bao gồm hơn 80 đặc trưng (features) thống kê.

Một số đặc trưng quan trọng được sử dụng trong mô hình bao gồm:

- Đặc trưng về thời gian: Flow Duration (Tổng thời gian luồng), Flow IAT (Khoảng thời gian giữa các gói tin).
- Đặc trưng về kích thước: Total Fwd Packets (Tổng số gói tin chiều đi), Total Backward Packets (Tổng số gói tin chiều về), Total Length of Fwd Packets.
- Đặc trưng về cờ (Flags): FIN, SYN, RST, PUSH, ACK (Các cờ trong tiêu đề TCP giúp nhận diện hành vi quét cổng hoặc tấn công từ chối dịch vụ).
- Nhãn (Target Label): Cột Label xác định luồng đó là BENIGN (Bình thường) hay tên cụ thể của cuộc tấn công (ví dụ: FTP-Patator).

3.2.3. Dữ liệu thực nghiệm (Experimental Dataset)

Để xây dựng một hệ thống IDS có khả năng phát hiện đa lớp (Multi-class Detection) nhưng vẫn đảm bảo tối ưu tài nguyên tính toán, nghiên cứu sinh đã lựa chọn chiến lược tích hợp 3 tập dữ liệu con tiêu biểu nhất từ bộ CIC-IDS2017.

Sự lựa chọn này không ngẫu nhiên mà dựa trên mô hình Cyber Kill Chain (Chuỗi tiêu diệt), bao gồm ba giai đoạn tấn công phổ biến nhất:

1. Giai đoạn Do thám (Reconnaissance): Sử dụng tập dữ liệu Friday-PortScan. Đây là hành vi kẻ tấn công dò quét các cổng mở để tìm điểm yếu.
2. Giai đoạn Thâm nhập (Delivery/Exploitation): Sử dụng tập dữ liệu Tuesday-WorkingHours (Brute Force). Đây là hành vi kẻ tấn công cố gắng bẻ khóa mật khẩu SSH/FTP để giành quyền truy cập hệ thống.
3. Giai đoạn Phá hoại (Action on Objectives): Sử dụng tập dữ liệu Friday-DDoS. Đây là hành vi tấn công từ chối dịch vụ nhằm làm tê liệt hệ thống mục tiêu.

Tổng hợp dữ liệu: Sau khi gộp 3 file và loại bỏ các giá trị lỗi, bộ dữ liệu bao gồm đầy đủ hai nhóm nhãn:

- Nhãn Bình thường (0): Được trích xuất từ phần lưu lượng sạch (Benign) có sẵn trong cả 3 file.

- Nhãn Tấn công (1): Bao gồm tổng hợp của PortScan, Brute Force và DDoS.

Việc kết hợp này giúp các mô hình học được các đặc trưng đa dạng của tấn công thay vì chỉ học vẹt một kiểu tấn công duy nhất.

3.3. Kĩ thuật Tiền xử lý dữ liệu

Dữ liệu thô từ bộ CIC-IDS2017 tuy chất lượng cao nhưng vẫn chứa nhiều "sạn" (nhiễu) và chưa phù hợp để đưa trực tiếp vào các thuật toán học máy. Do đó, quá trình tiền xử lý được thực hiện qua 4 bước nghiêm ngặt sau:

3.3.1. Vệ sinh dữ liệu (Data Cleaning)

Đây là bước đầu tiên để đảm bảo tính toàn vẹn của dữ liệu đầu vào.

- Xử lý tên cột: Trong file CSV gốc, tên cột chứa các khoảng trắng thừa ở đầu (ví dụ: ' Label' thay vì 'Label'). Hệ thống thực hiện chuẩn hóa tên cột bằng cách loại bỏ các khoảng trắng này để tránh lỗi truy xuất.
- Xử lý giá trị vô cực và rỗng (Infinity & NaN): Trong quá trình trích xuất đặc trưng luồng mạng (Flow), một số phép tính (như chia cho 0) tạo ra giá trị vô cực (Infinity) hoặc không xác định (NaN). Các thuật toán của thư viện Scikit-learn không thể xử lý các giá trị này.
    - *Giải pháp:* Hệ thống quét toàn bộ dữ liệu và loại bỏ các dòng chứa giá trị này.
- Loại bỏ dữ liệu trùng lặp: Dữ liệu mạng thường có các gói tin lặp lại giống hệt nhau. Việc giữ lại chúng sẽ khiến mô hình bị thiên lệch (bias), học thuộc lòng các mẫu này thay vì học quy luật chung.
    - *Giải pháp:* Sử dụng hàm drop_duplicates() để giữ lại các mẫu duy nhất.

3.3.2. Mã hóa nhãn (Label Encoding)

Các thuật toán học máy yêu cầu dữ liệu đầu vào và đầu ra phải ở dạng số (Numerical). Tuy nhiên, cột nhãn mục tiêu (Label) trong bộ dữ liệu lại ở dạng văn bản (Categorical).

- Phương pháp: Áp dụng kỹ thuật mã hóa nhãn để chuyển đổi bài toán về dạng Phân loại nhị phân (Binary Classification):

- Nhãn BENIGN → Mã hóa thành 0.
- Các nhãn tấn công (FTP-Patator, SSH-Patator) → Mã hóa thành 1.

```
# 3. Mã hóa nhãn (Label)
df_temp['is_attack'] = df_temp['Label'].apply(lambda x: 0 if x == 'BENIGN' else 1)
df_temp.drop('Label', axis=1, inplace=True)
```

Ảnh 4: Mã nguồn xử lý "Mã hóa nhãn"

3.3.3. Chuẩn hóa dữ liệu (Feature Scaling)

Bộ dữ liệu CIC-IDS2017 có sự chênh lệch rất lớn về thang đo giữa các đặc trưng.

- Ví dụ: Đặc trưng Flow Duration (Thời gian luồng) có thể lên tới hàng triệu (micro giây), trong khi đặc trưng Total Fwd Packets (Số gói tin) chỉ khoảng vài chục.
- Vấn đề: Nếu không chuẩn hóa, mô hình sẽ mặc định coi Flow Duration quan trọng hơn chỉ vì nó có giá trị lớn hơn, dẫn đến sai lệch trong quá trình học (đặc biệt với Logistic Regression).
- Giải pháp: Sử dụng kỹ thuật StandardScaler (Chuẩn hóa Z-score) để đưa tất cả các đặc trưng về cùng một phân phối chuẩn với trung bình $\mu = 0$ và độ lệch chuẩn $\sigma = 1$.

Công thức chuẩn hóa cho từng giá trị x:

$$z = \frac{x - \mu}{\sigma}$$

```
# Import thư viện StandardScaler
from sklearn.preprocessing import StandardScaler

if 'df' in locals():
    # 3a. Tách Đặc trưng (X) và Mục tiêu (y)
    y = df['is_attack']
    X = df.drop('is_attack', axis=1)

    print(f"Đã tách X (Kích thước: {X.shape}) và y (Kích thước: {y.shape}).")

    # 3b. Chuẩn hóa (Scaling) Đặc trưng X
    # Tạo một đối tượng StandardScaler
    scaler = StandardScaler()

    # "Học" (fit) và "biến đổi" (transform) X
    X_scaled_array = scaler.fit_transform(X)

    # Chuyển mảng NumPy trở lại thành DataFrame
    X_scaled = pd.DataFrame(X_scaled_array, columns=X.columns)

    print("Chuẩn hóa X hoàn tất!")
    print("\n--- 5 hàng đầu của X ---")
    print(X_scaled.head())
else:
    print("LỖI")
```

Ảnh 5: Mã nguồn xử lý "Chuẩn hóa dữ liệu"

3.3.4. Phân chia tập dữ liệu (Train/Test Split)

Để đánh giá khách quan khả năng tổng quát hóa của mô hình, dữ liệu được chia ngẫu nhiên thành 2 tập độc lập:

- Tập huấn luyện (Training Set - 80%): Dùng để huấn luyện mô hình.
- Tập kiểm thử (Test Set - 20%): Dùng để kiểm tra độ chính xác.
    - Việc phân chia được thực hiện ngẫu nhiên (random_state=42) để đảm bảo tính tái lập (reproducibility) của thí nghiệm.

```python
# Import thư viện train_test_split
from sklearn.model_selection import train_test_split

if 'X_scaled' in locals() and 'y' in locals():
    # test_size=0.2 (20% test)
    X_train, X_test, y_train, y_test = train_test_split(
        X_scaled,
        y,
        test_size=0.2,
        random_state=42
    )

    print("Chia dữ liệu hoàn tất!")
    print(f"X_train (đặc trưng huấn luyện): {X_train.shape}")
    print(f"y_train (mục tiêu huấn luyện): {y_train.shape}")
    print(f"X_test (đặc trưng kiểm tra):  {X_test.shape}")
    print(f"y_test (mục tiêu kiểm tra):  {y_test.shape}")
else:
    print("LỖI")
```

Ảnh 6: Mã nguồn xử lý "Phân chia dữ liệu"

CHƯƠNG 4: CÀI ĐẶT VÀ THỰC NGHIỆM

4.1. Môi trường cài đặt

Để đảm bảo tính linh hoạt và khả năng tái lập của thực nghiệm, đồ án được xây dựng và triển khai hoàn toàn trên nền tảng đám mây, sử dụng bộ công cụ mã nguồn mở phổ biến nhất trong lĩnh vực Khoa học dữ liệu (Data Science).

4.1.1. Nền tảng triển khai (Platform)

Hệ thống được phát triển trên môi trường Google Colab (Colaboratory).

- Mô tả: Đây là dịch vụ sổ tay Jupyter (Jupyter Notebook) được lưu trữ trên đám mây của Google.
- Lý do lựa chọn:
    - Cung cấp môi trường Python được cấu hình sẵn, không cần cài đặt phức tạp trên máy cá nhân.
    - Cung cấp tài nguyên phần cứng mạnh mẽ (RAM ~12GB, CPU ảo hóa) giúp xử lý nhanh các tập dữ liệu lớn như CIC-IDS2017 mà không gây treo máy tính cá nhân.
    - Dễ dàng chia sẻ mã nguồn và kết quả thực nghiệm.

4.1.2. Ngôn ngữ lập trình

- Ngôn ngữ: Python (Phiên bản 3.10+).
- Đặc điểm: Python là ngôn ngữ lập trình số một thế giới về Trí tuệ nhân tạo và Học máy nhờ cú pháp đơn giản, cộng đồng hỗ trợ lớn và hệ sinh thái thư viện phong phú.

4.1.3. Các thư viện hỗ trợ (Libraries)

Hệ thống sử dụng các thư viện mã nguồn mở tiêu chuẩn sau:

Thư viện	Vai trò trong đồ án
Pandas	Đọc file CSV, xử lý dữ liệu dạng bảng (DataFrame), làm sạch dữ liệu (xử lý NaN, Inf).
NumPy	Xử lý các phép toán đại số tuyến tính và thao tác trên mảng (Array) hiệu năng cao.
Scikit-learn (sklearn)	Thư viện lõi (Core) cung cấp các thuật toán Machine Learning (Random Forest, Decision Tree, Logistic Regression) và các công cụ tiền xử lý (StandardScaler, train_test_split), đánh giá (classification_report, confusion_matrix).
Matplotlib / Seaborn	Trực quan hóa dữ liệu, vẽ biểu đồ so sánh và hiển thị Ma trận nhầm lẫn (Confusion Matrix).
Joblib	Lưu trữ và tải lại mô hình đã huấn luyện.

4.2. Kịch bản thử nghiệm

Để đảm bảo tính khách quan và công bằng trong việc so sánh, tất cả các mô hình đều được huấn luyện và kiểm thử trên cùng một tập dữ liệu đã qua xử lý, tuân theo quy trình thống nhất sau:

4.2.1. Thiết lập dữ liệu (Data Setup)

Phương pháp phân chia: Sử dụng phương pháp Hold-out. Dữ liệu được chia thành hai tập độc lập:

- Tập huấn luyện (Train set): 80% (Dùng để học tham số).
- Tập kiểm thử (Test set): 20% (Dùng để đánh giá độc lập).

Cơ chế ngẫu nhiên: Sử dụng tham số random_state=42 trong quá trình chia dữ liệu.

Mục đích: Đảm bảo tính tái lập (reproducibility). Bất kỳ ai chạy lại code này cũng sẽ nhận được cách chia dữ liệu y hệt nhau, giúp kết quả thực nghiệm ổn định, không bị thay đổi ngẫu nhiên mỗi lần chạy.

4.2.2. Cấu hình tham số mô hình (Hyperparameters)

Các thuật toán được thiết lập với các tham số tối ưu hóa như sau:

Mô hình (Model)	**Tham số (Parameter)**	**Giá trị thiết lập**	**Giải thích ý nghĩa**
Logistic Regression	max_iter	1000	Tăng số vòng lặp tối đa để đảm bảo thuật toán hội tụ (tìm ra nghiệm tối ưu) thay vì dừng sớm (mặc định chỉ là 100).
	random_state	42	Đảm bảo kết quả nhất quán.
Decision Tree	criterion	gini	(Mặc định) Sử dụng chỉ số Gini để đo độ tinh khiết khi phân chia nút.
	random_state	42	Đảm bảo cây được xây dựng giống nhau mỗi lần chạy.
Random Forest	n_estimators	100	Số lượng cây quyết định trong rừng. 100 cây là con số cân bằng giữa hiệu suất và tốc độ.
	n_jobs	-1	Sử dụng tất cả các nhân CPU có sẵn của hệ thống để huấn luyện

			song song, giúp tăng tốc độ xử lý dữ liệu lớn.
	random_state	42	Đảm bảo tính ổn định của mô hình tổ hợp.

```
# Cấu hình các mô hình
model_lr = LogisticRegression(random_state=42, max_iter=1000)
model_dt = DecisionTreeClassifier(random_state=42)
model_rf = RandomForestClassifier(n_estimators=100, random_state=42, n_jobs=-1)
```

Ảnh 7: Thiết lập các mô hình

4.2.3. Quy trình thực hiện

Quy trình thực nghiệm diễn ra theo 3 bước:

1. *Bước 1 - Huấn luyện (Training):* Lần lượt đưa tập *Train set* vào 3 mô hình để học các đặc trưng phân loại.
2. *Bước 2 - Dự báo (Prediction):* Sử dụng các mô hình đã học để dự đoán nhãn cho tập *Test set* (dữ liệu mô hình chưa từng gặp).
3. *Bước 3 - So sánh (Comparison):* Đối chiếu kết quả dự báo với nhãn thực tế (Ground Truth) để tính toán các chỉ số Precision, Recall và lập bảng xếp hạng hiệu năng.

4.3. Kết quả thực nghiệm và đánh giá

Sau quá trình huấn luyện và kiểm thử trên tập dữ liệu Tuesday-WorkingHours.csv (đã qua trích chọn đặc trưng), kết quả thu được từ ba mô hình như sau:

4.3.1. So sánh hiệu năng tổng quát (Performance Comparison)

Bảng dưới đây tổng hợp các chỉ số đánh giá của ba thuật toán trên cùng tập dữ liệu kiểm thử (Test Set):

Thuật toán (Model)	Accuracy	Precision	Recall	F1-Score	Xếp hạng
Decision Tree	~99.98%	~99.94%	99.98%	~99.96%	1
Random Forest	~99.97%	100.00%	~99.90%	~99.5%	2
Logistic Regression	~97.7%	~97.45%	~95.13%	~96.28%	3

Nhận xét:

- Logistic Regression cho kết quả thấp nhất, đặc biệt là chỉ số Recall. Điều này dễ hiểu vì đây là mô hình tuyến tính, khó có khả năng phân tách dữ liệu phức tạp và nhiều chiều của lưu lượng mạng.
- Decision Tree cho kết quả khá tốt, tiệm cận với Random Forest. Tuy nhiên, mô hình đơn lẻ này thường có độ ổn định kém hơn.
- Random Forest đạt hiệu suất vượt trội và ổn định nhất trên cả 4 chỉ số. Đây là mô hình được lựa chọn làm giải pháp cuối cùng cho hệ thống.

4.3.2. Phân tích chi tiết mô hình đề xuất

Dựa trên kết quả thực nghiệm, Decision Tree được lựa chọn là mô hình tối ưu nhất cho hệ thống. Để minh chứng cho độ tin cậy của mô hình này, chúng ta đi sâu phân tích Ma trận nhầm lẫn (Confusion Matrix) thực tế:

```
--- Ma trận nhầm lẫn (Confusion Matrix) ---
          (Dự đoán 0)   (Dự đoán 1)
Thực tế 0: 26224        7            (True Negative / False Positive)
Thực tế 1: 2            12065        (False Negative / True Positive)
```

Ảnh 8: Ma trận nhầm lẫn của Decision Tree

- Khả năng phát hiện Tấn công (Sensitivity/Recall):
  - Trong tổng số 12,067 mẫu tấn công thực tế, mô hình đã phát hiện chính xác 12,065 trường hợp.
  - Số lượng mẫu bị bỏ lọt (False Negative) chỉ là 2. Đây là chỉ số quan trọng nhất, cho thấy mô hình có độ nhạy cực cao, giảm thiểu tối đa rủi ro hệ thống bị xâm nhập mà không hay biết.
- Tỷ lệ báo động giả (False Alarm Rate):
  - Trong hơn 26,000 mẫu bình thường, mô hình chỉ nhầm lẫn 7 mẫu là tấn công (False Positive).

- Mặc dù con số này không phải là tuyệt đối (bằng 0), nhưng tỷ lệ này là không đáng kể so với tổng lưu lượng mạng, hoàn toàn nằm trong ngưỡng chấp nhận được của các hệ thống giám sát.

4.3.3. So sánh đối chứng với Random Forest

Một điểm thú vị trong thực nghiệm này là sự so sánh trực tiếp với mô hình Random Forest. Dưới đây là bảng đối chiếu nhanh dựa trên ma trận nhầm lẫn của hai thuật toán:

Chỉ số đánh giá	Decision Tree	Random Forest	Đánh giá
Độ chính xác (Accuracy)	99.98%	99.97%	Decision Tree nhỉnh hơn.
Báo động giả (False Positive)	7	0	Random Forest sạch hơn (Không ồn).
Bỏ lọt tấn công (False Negative)	2	11	Decision Tree an toàn hơn rất nhiều.
Độ nhạy (Recall)	99.98%	99.91%	Decision Tree bắt lỗi tốt hơn.

Đánh giá:

- Random Forest đạt được sự "hoàn hảo" về độ chính xác dự báo (Precision 100%) khi chỉ số FP = 0. Điều này có nghĩa là mọi cảnh báo nó đưa ra đều chắc chắn là tấn công.
- Tuy nhiên, Decision Tree lại chiến thắng ở chỉ số Recall. Nó chỉ bỏ sót 2 cuộc tấn công, trong khi Random Forest bỏ sót tới 11.
- Kết luận: Trong bài toán an ninh mạng, một cuộc tấn công bị bỏ lọt (FN) có thể gây hậu quả nghiêm trọng hơn nhiều so với một vài cảnh báo giả (FP). Do đó, việc Decision Tree bắt được nhiều hơn (dù chỉ là 1 mẫu) cộng với lợi thế vượt trội về tốc độ xử lý, giúp nó trở thành lựa chọn hợp lý hơn.

4.4. Thảo luận

Kết quả thực nghiệm trên tập dữ liệu mở rộng (3 loại tấn công) đã củng cố mạnh mẽ luận điểm lựa chọn Decision Tree của đồ án. Dưới đây là các thảo luận chuyên sâu:

4.4.1. Sự đánh đổi giữa "Yên tĩnh" và "An toàn"

Chúng ta đứng trước hai lựa chọn:

- Lựa chọn Random Forest (Sự yên tĩnh): Hệ thống tuyệt đối không làm phiền quản trị viên (0 báo động giả), nhưng hacker có 11 cơ hội để xâm nhập hệ thống thành công.
- Lựa chọn Decision Tree (Sự an toàn): Hệ thống có thể báo nhầm 7 lần (việc kiểm tra lại 7 gói tin này chỉ mất vài giây), nhưng đổi lại hacker chỉ còn 2 cơ hội mong manh để lọt lưới.

Trong bối cảnh an ninh mạng, mục tiêu tối thượng là Zero Trust (Không bỏ sót). Do đó, sự đánh đổi của Decision Tree là hoàn toàn xứng đáng và cần thiết. Việc để lọt 9 cuộc tấn công chênh lệch (11 so với 2) có thể dẫn đến hậu quả nghiêm trọng hơn rất nhiều so với chi phí xử lý 7 cảnh báo giả.

4.4.2. Khả năng phân loại đa lớp (Multi-class Classification)

Kết quả này bác bỏ giả thuyết cho rằng Decision Tree chỉ hoạt động tốt trên các bài toán đơn giản. Ngay cả khi dữ liệu bao gồm hỗn hợp 3 loại tấn công (PortScan, Brute Force, DDoS) với các đặc trưng khác nhau, Decision Tree vẫn xây dựng được các biên quyết định (Decision Boundaries) cực kỳ sắc bén.

Điều này cho thấy các loại tấn công mạng này có tính phân tách tuyến tính (Linear Separability) rất cao.

Cấu trúc cây đơn lẻ đã tận dụng triệt để đặc tính này để cắt gọt không gian dữ liệu chính xác hơn cả phương pháp bỏ phiếu (Voting) của Random Forest, vốn có thể bị "pha loãng" quyết định bởi một số cây con hoạt động kém hiệu quả.

4.4.3. Kết luận

Với chỉ số Recall 99.98% và khả năng xử lý tốc độ cao, Decision Tree là mô hình phù hợp nhất để được lựa chọn làm mô hình lõi cho hệ thống. Nó chứng minh rằng một giải pháp đơn giản nhưng phù hợp với đặc trưng dữ liệu sẽ mang lại hiệu quả vượt trội hơn các giải pháp phức tạp.

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết luận

Đồ án đã nghiên cứu và xây dựng thành công Hệ thống phát hiện xâm nhập mạng (IDS) dựa trên kỹ thuật Học máy, sử dụng bộ dữ liệu chuẩn CIC-IDS2017. Thông qua việc tích hợp ba tập dữ liệu đại diện cho ba giai đoạn tấn công phổ biến (Dò quét - PortScan, Thâm nhập - Brute Force, Phá hoại - DDoS), và thực nghiệm so sánh giữa các thuật toán, đồ án đã rút ra những kết luận quan trọng sau:

1. Hiệu quả của việc tích hợp dữ liệu đa lớp: Việc kết hợp 3 nguồn dữ liệu khác nhau đã giúp mô hình không chỉ nhận diện được một loại tấn công đơn lẻ mà có khả năng bao quát các mối đe dọa đa dạng. Kết quả thực nghiệm cho thấy các đặc trưng hành vi của các cuộc tấn công này có tính phân tách rất cao so với lưu lượng bình thường.
2. Sự vượt trội của Decision Tree: Kết quả thực nghiệm đã chứng minh Decision Tree là thuật toán tối ưu nhất cho kịch bản này.
   - Với chỉ số Recall đạt 99.98%, mô hình chỉ bỏ lọt 2 trường hợp tấn công trên tổng số hơn 12,000 mẫu thử nghiệm (thấp hơn nhiều so với 11 trường hợp của Random Forest).
   - Decision Tree cũng thể hiện ưu thế tuyệt đối về tốc độ xử lý và khả năng giải thích (Explainability), cho phép trích xuất các luật nhận diện tấn công minh bạch.
3. Lựa chọn giải pháp: Dựa trên nguyên lý "Lưỡi dao Occam" (Occam's Razor), đồ án đề xuất sử dụng Decision Tree làm mô hình lõi. Đây là giải pháp cân bằng tốt nhất giữa độ an toàn (Recall cao nhất) và hiệu năng hệ thống (Chi phí tính toán thấp nhất), phù hợp để triển khai trên các thiết bị mạng có tài nguyên giới hạn.

5.2. Hạn chế và hướng phát triển

Bên cạnh các kết quả đạt được, nhóm nghiên cứu nhìn nhận khách quan những hạn chế còn tồn tại cần được khắc phục:

1. Nguy cơ quá khớp (Overfitting): Việc Decision Tree đạt độ chính xác gần như tuyệt đối (99.98%) trên tập kiểm thử gợi mở lo ngại về hiện tượng quá khớp. Do các mẫu tấn công trong tập dữ liệu thực nghiệm (PortScan, DDoS) có đặc trưng quá rõ ràng ("chữ ký" mạnh), mô hình có thể đang học thuộc lòng các quy tắc này. Điều này có thể dẫn đến hiệu suất giảm sút khi đối mặt với các biến thể tấn công mới lạ (Zero-day attacks) hoặc các tấn công "nằm vùng" chậm hơn (Low-and-slow) chưa có trong dữ liệu huấn luyện.
2. Giới hạn về phạm vi triển khai: Hệ thống hiện tại được xây dựng và đánh giá trong môi trường phân tích ngoại tuyến (Offline Analysis) trên Google Colab. Đồ án chưa thực hiện tích hợp mô hình vào luồng mạng thực tế (Real-time traffic stream) để đánh giá độ trễ (Latency) và khả năng chịu tải khi lưu lượng mạng tăng đột biến.
3. Vấn đề mất cân bằng dữ liệu: Dù đã tích hợp 3 file, tỷ lệ giữa các loại tấn công vẫn chưa đồng đều (ví dụ: DDoS thường chiếm số lượng lớn hơn Brute Force). Đồ án chưa áp dụng các kỹ thuật cân bằng dữ liệu nâng cao (như SMOTE) để cải thiện khả năng học các lớp thiểu số.

Để nâng cao tính ứng dụng và độ tin cậy của hệ thống, các hướng nghiên cứu tiếp theo được đề xuất bao gồm:

1. Tối ưu hóa khả năng tổng quát hóa: Nghiên cứu áp dụng kỹ thuật Cắt tỉa cây (Pruning) cho Decision Tree hoặc giới hạn độ sâu (max_depth) để giảm thiểu Overfitting. Đồng thời, mở rộng huấn luyện trên toàn bộ 8 file của bộ CIC-IDS2017 để mô hình tiếp xúc với nhiều dạng tấn công phức tạp hơn (như Infiltration, Botnet, Web Attacks).
2. Triển khai thời gian thực (Real-time Deployment): Đóng gói mô hình đã huấn luyện (sử dụng thư viện Joblib) và tích hợp vào một hệ thống giám sát thực tế. Xây dựng một Dashboard (sử dụng Streamlit hoặc Flask) kết nối với công cụ bắt gói tin (như Snort/Wireshark) để hiển thị cảnh báo tấn công ngay lập tức cho quản trị viên.

3. Nghiên cứu so sánh với Deep Learning: Triển khai các mô hình Học sâu như CNN (Convolutional Neural Network) hoặc LSTM (Long Short-Term Memory). Các mô hình này có khả năng học các đặc trưng chuỗi thời gian (Time-series) tốt hơn, giúp phát hiện các cuộc tấn công phức tạp kéo dài mà Decision Tree có thể bỏ sót.

Link Github của dự án: ducngyen0201/Project-3

TÀI LIỆU THAM KHẢO

[1] "Logistic Regression in Machine Learning," GeeksForGeeks, 23 12 2025. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/understanding-logistic-regression/.

[2] "Random Forest Algorithm in Machine Learning," GeeksForGeeks, 23 12 2025. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/random-forest-algorithm-in-machine-learning/.

[3] "Decision Tree in Machine Learning," GeeksForGeeks, 23 12 2025. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/decision-tree-introduction-example/.

[4] "Difference Between Random Forest and Decision Tree," GeeksForGeeks, 23 7 2025. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/difference-between-random-forest-and-decision-tree/.

[5] "Intrusion Detection System (IDS)," GeeksForGeeks, 11 7 2025. [Online]. Available: https://www.geeksforgeeks.org/ethical-hacking/intrusion-detection-system-ids/.