BÁO CÁO TỔNG KẾT ĐỒ ÁN MÔN HỌC

**Môn học: Hệ thống tìm kiếm, phát hiện và ngăn chặn xâm nhập**

**Tên chủ đề: G-IDCS: Graph-Based Intrusion Detection and**

**Classification System for CAN Protocol**

*Mã nhóm: G8………. Mã đề tài: B18…….*

**Lớp**: **NT204.P22.ANTT**

1. **THÔNG TIN THÀNH VIÊN NHÓM:**

*(Sinh viên liệt kê tất cả các thành viên trong nhóm)*

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** | **Email** |
| 1 | Phạm Trường Thiên Ân | 22520028 | 22520028@gm.uit.edu.vn |
| 2 | Nguyễn Hữu Bình | 22520132 | 22520132@gm.uit.edu.vn |
| 3 | Bùi Phương Đại | 22520180 | 22520180@gm.uit.edu.vn |
| 4 | Lê Công Danh | 22520199 | 22520199@gm.uit.edu.vn |

**MỤC LỤC**

[**1.** **THÔNG TIN THÀNH VIÊN NHÓM:** 1](#_Toc198621394)

[**2. TÓM TẮT NỘI DUNG THỰC HIỆN:** 2](#_Toc198621395)

[A.Chủ đề nghiên cứu trong Hệ thống tìm kiếm, phát hiện và ngăn ngừa xâm nhập: 2](#_Toc198621396)

[B.Tên bài báo tham khảo chính: 3](#_Toc198621397)

[C.Dịch tên Tiếng Việt cho bài báo: 3](#_Toc198621398)

[D.Tóm tắt nội dung chính: 3](#_Toc198621399)

[E.Tóm tắt các kỹ thuật chính được mô tả sử dụng trong bài báo: 4](#_Toc198621400)

[F.Môi trường thực nghiệm của bài báo: 6](#_Toc198621401)

[G.Kết quả thực nghiệm của bài báo: 7](#_Toc198621402)

[H.Công việc/tính năng/kỹ thuật mà nhóm thực hiện lập trình và triển khai cho demo: 8](#_Toc198621403)

[I.Các khó khăn, thách thức hiện tại khi thực hiện: 10](#_Toc198621404)

[**3. TỰ ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH SO VỚI KẾ HOẠCH THỰC HIỆN:** 10](#_Toc198621405)

[**4. NHẬT KÝ PHÂN CÔNG NHIỆM VỤ:** 10](#_Toc198621406)

[A.Phương pháp thực hiện 12](#_Toc198621407)

[a) Kiến trúc, thành phần của hệ thống trong bài báo: 12](#_Toc198621408)

[b) Kiến trúc nhóm đã thực hiện: 14](#_Toc198621409)

[B.Chi tiết cài đặt, hiện thực: 22](#_Toc198621410)

[C.Kết quả thực nghiệm 30](#_Toc198621411)

[D.Hướng phát triển 32](#_Toc198621412)

[**YÊU CẦU CHUNG** 34](#_Toc198621413)

1. **TÓM TẮT NỘI DUNG THỰC HIỆN:[[1]](#footnote-1)**

## Chủ đề nghiên cứu trong Hệ thống tìm kiếm, phát hiện và ngăn ngừa xâm nhập:

🗹 Định hướng nghiên cứu

🞎 Định hướng ứng dụng

## Tên bài báo tham khảo chính:

|  |
| --- |
| Sung Bum Park, Hyo Jin Jo and Dong Hong Lee "G-IDCS: Graph-Based Intrusion Detection and Classification System for CAN Protocol” |

## Dịch tên Tiếng Việt cho bài báo:

|  |
| --- |
| "Hệ thống phát hiện và phân loại xâm nhập dựa trên đồ thị cho giao thức CAN." |

## Tóm tắt nội dung chính:

|  |
| --- |
| <mô tả nội dung tóm tắt của bài báo/chủ đề trong vòng 350 từ>  - Sự gia tăng của các phương thức kết nối (Wi-Fi, V2X, v.v.) làm tăng bề mặt tấn công (attack surfaces) của CAN. Các nghiên cứu trước đây đã chỉ ra rằng kẻ tấn công có thể thực hiện các cuộc tấn công như chiếm quyền điều khiển xe hoặc làm xe hoạt động bất thường thông qua việc tiêm tin nhắn độc hại và những hạn chế của các IDS hiện có (Packet-based IDS, Window-based IDS):   * Packet-based IDS: Nhanh nhưng không phân tích tương quan gói tin. Không phân loại được các cuộc tấn công. * Window-based IDS: Dùng thống kê/ML, cần nhiều thông điệp. Không giải thích được, nhạy cảm với thay đổi.   - Bài báo đề xuất G-IDCS, một hệ thống kết hợp hai bộ phân loại:  - **TH\_classifier:** Bộ phân loại dựa trên ngưỡng (threshold-based) để phát hiện xâm nhập nhanh chóng, bền vững với sự thay đổi loại tấn công, và cung cấp khả năng giải thích.  - **ML\_classifier:** Bộ phân loại dựa trên học máy để phân loại loại tấn công, hỗ trợ phân tích sau sự cố (digital forensics).  -Kết quả cho thấy:  - **TH\_classifier:** Phát hiện **chính xác trên 99%** với phương pháp ngưỡng Max-Min, so sánh với IDS dựa trên đồ thị cũ (cần 6,000 gói tin), G-IDCS cần chỉ **200 gói,** tốc độ phát hiện: ~142 µs/graph → **nhanh hơn IDS** dựa trên đồ thị cũ.  - **ML\_classifier:** Chính xác hơn phương pháp H-IDS, có thể phân loại chính xác cả replay attacks – điều H-IDS không làm tốt. |

## Tóm tắt các kỹ thuật chính được mô tả sử dụng trong bài báo:

|  |
| --- |
| <mô tả các kỹ thuật chính được dùng trong nghiên cứu của bài báo>  (liệt kê tóm tắt vai trò của các kỹ thuật đó – kèm hình ảnh minh họa về hệ thống/kỹ thuật/phương pháp)  Bài báo sử dụng 2 thuật toán: 1 thuật toán huấn luyện và 1 thuật toán thực nghiệm.      **Algorithm 1 (Classifier Configuration):**   * Nhận đầu vào là các đặc trưng đồ thị. * Nếu dữ liệu chỉ chứa tin nhắn bình thường: Tính ngưỡng cho TH\_classifier. * Nếu dữ liệu có nhãn loại tấn công: Huấn luyện ML\_classifier. * Trả về TH\_classifier và ML\_classifier.     **Algorithm 2 (Attack Detection and Classification):**   * Nhận đầu vào là đặc trưng đồ thị và loại bộ phân loại (TH hoặc ML). * TH\_classifier: Phân loại nhị phân (bình thường/tấn công). * ML\_classifier: Phân loại đa lớp (loại tấn công cụ thể). |

## Môi trường thực nghiệm của bài báo:

|  |
| --- |
| <mô tả môi trường thực nghiệm của nhóm tác giả, bao gồm cấu hình máy tính (phần cứng, phần mềm), các chương trình hỗ trợ để hiện thực phương pháp, ngôn ngữ lập trình được sử dụng, các chương trình phần mềm dùng để kiểm tra khả năng/tính năng của phương pháp>   * Cấu hình máy tính: CPU Intel Core i7-9700K 3.60GHz, RAM 32GB, chạy hệ điều hành Windows 10. * Các công cụ hỗ trợ sẵn có: NetworkX (xây dựng đồ thị), Scikit-learn (học máy), và Pandas (xử lý dữ liệu) đã được sử dụng trong các giai đoạn xây dựng đồ thị, trích xuất đặc trưng, và huấn luyện mô hình. * Ngôn ngữ lập trình để hiện thực phương pháp: Python. * Đối tượng nghiên cứu (chương trình phần mềm dùng để kiểm tra tính khả thi của phương pháp/tập dữ liệu – nếu có):   + Car Hacking-Attack & Defense Challenge dataset [41]: Tập dữ liệu này chứa các thông điệp CAN từ các cuộc tấn công thực tế (flooding, fuzzy, replay, spoofing), được sử dụng để kiểm tra khả năng phát hiện và phân loại của G-IDCS.  + Dataset từ ba loại xe: Bao gồm dữ liệu từ HYUNDAI YF Sonata, KIA Soul, và CHEVROLET Spark, với các cuộc tấn công DoS, fuzzy, và malfunction, nhằm đánh giá khả năng tổng quát hóa trên các loại xe khác nhau.   * Tiêu chí đánh giá tính hiệu quả của phương pháp:   + Accuracy (): Tỷ lệ trung bình các cửa sổ được phân loại đúng trên tổng số cửa sổ.  + Precision (): Tỷ lệ trung bình các cửa sổ được phân loại đúng trong số các cửa sổ được phân loại vào từng lớp.  + Recall (): Tỷ lệ trung bình các cửa sổ thuộc từng lớp được phân loại đúng.  + F1-score: Trung bình có trọng số của Precision và Recall.    **Hệ thống:**  C:\Users\HP\Pictures\Screenshots\Ảnh chụp màn hình 2025-05-19 222004.png   * Cấu trúc mạng nội bộ của xe:   + Xe hiện đại được trang bị nhiều ECU (Electronic Control Units) như: ECU 1 (Engine ECU) dùng để điều khiển động cơ, ECU 2 (Powertrain ECU) dùng để kiểm soát hệ thống truyền động,...  + Các ECU này kết nối thành các sub-networks thông qua gateway trung tâm.   * Xe có cổng kết nối chuẩn OBD-II, cho phép: Kỹ thuật viên chẩn đoán lỗi, cũng là nơi mà thiết bị tấn công hoặc giám sát có thể kết nối vào hệ thống. * Gateway là điểm kết nối chính giữa các subnet của các ECU và chịu trách nhiệm điều phối luồng dữ liệu. * G-IDCS giả định rằng một thiết bị giám sát – Monitoring Unit sẽ được cài đặt trực tiếp vào gateway, hoặc kết nối vào CAN bus thông qua cổng OBD-II. * Chức năng của Monitoring Unit:   + Trích xuất đặc trưng đồ thị từ gói tin CAN.  + Huấn luyện mô hình phân loại với các đặc trưng đồ thị.  + Phân loại gói tin CAN chưa biết (tấn công hay không / loại tấn công). |

## Kết quả thực nghiệm của bài báo:

|  |
| --- |
| <mô tả ngắn gọn kết quả thực nghiệm của bài báo, tự nhận xét về khả năng, ưu và nhược điểm của phương pháp được đề cập trong bài báo>  Kết quả chính:   * TH\_classifier: Đạt độ chính xác trung bình trên 99% trong phát hiện tấn công, giảm số gói tin cần thiết xuống hơn 1/30 (200 gói tin so với 6,000 gói tin của IDS dựa trên đồ thị cũ [21]), và cải thiện độ chính xác phát hiện tấn công kết hợp hơn 9%. Thời gian phát hiện trung bình là 142.6 µs/graph. * ML\_classifier: Đạt độ chính xác trung bình 98.17%, vượt trội hơn H-IDS (97.25%) trong phân loại loại tấn công, với thời gian phân loại trung bình 139.1 µs/graph.   Nhận xét về khả năng, ưu và nhược điểm:   * Ưu điểm:   + TH\_classifier phát hiện nhanh, chính xác với ít gói tin hơn, cung cấp khả năng giải thích đặc trưng.  + ML\_classifier phân loại chính xác các loại tấn công, hỗ trợ phân tích sau sự cố.   * Nhược điểm:   + Xây dựng đồ thị và trích xuất đặc trưng có thể tốn tài nguyên tính toán, đặc biệt trong hệ thống thời gian thực.  + Hệ thống có thể gặp khó khăn với các tấn công mới chưa được huấn luyện. |

## 

## Công việc/tính năng/kỹ thuật mà nhóm thực hiện lập trình và triển khai cho demo:

|  |
| --- |
| <liệt kê các công việc mà nhóm thực hiện cho đề tài dựa trên phân tích phương pháp/hệ thống được sử dụng trong bài báo đã tham khảo>  <liệt kê các công việc đã thực hiện+ tóm tắt kết quả của công việc này>  Công việc đã thực hiện:   1. **Tìm và thu thập dataset**  * Tải về bộ dữ liệu **“Car Hacking Attack Defense Challenge 2020**” từ [IEEE DataPort](https://ieee-dataport.org/open-access/car-hacking-attack-defense-challenge-2020-dataset). * Liên hệ Professor Huy Kang Kim qua gmail để truy cập và tải về bộ dữ liệu **“Car-Hacking Dataset for Intrusion Detection”** từ [Hacking and Countermeasure Research Lab](https://ocslab.hksecurity.net/Datasets/car-hacking-dataset).  1. **Tiền xử lý dữ liệu CAN**  * Đồng bộ định dạng file cho bộ dữ liệu Car-Hacking Dataset for Intrusion Detection đều là CSV; Thêm cột SubClass để thống nhất định dạng giữa dữ liệu Normal và Attack. * Viết hàm **parse\_csv\_row()** để đọc từng dòng CSV (Timestamp, Arbitration\_ID, DLC, Data, Class, SubClass). * Lọc giữ các bản tin có Class = Normal cho module TH\_classifier để tính các ngưỡng đặc trưng đồ thị.  1. **Xây dựng đồ thị tuần tự**  * Chia dữ liệu thành các windows có kích thước cố định 200 messages. * Triển khai hàm **build\_window\_graph()** tạo đồ thị có hướng từ mỗi windows, mỗi node là một tin nhắn/Arbitration\_ID, mỗi cạnh nối giữa tin nhắn i → i+1.  1. **Trích xuất đặc trưng đồ thị**   - Triển khai hàm **extract\_graph\_features()** tính 3 đặc trưng cho mỗi windows:   * **Graph elapsed time** = timestamp cuối – timestamp đầu * **Max degree** = bậc lớn nhất (in + out) trên các node * **Number of edges** = tổng số cạnh  1. **Triển khai TH\_classifier (Threshold based)**  * Tính ngưỡng cho từng đặc trưng từ các cửa sổ chỉ chứa Normal (ε = 0.03), trong đó ngưỡng trên upper\_threshold sẽ lấy max, ngưỡng dưới lower\_threshold sẽ lấy min trong tập các ngưỡng thu được. * Dự đoán “Attack” nếu bất kỳ đặc trưng nào vượt ngoài khoảng [lower\_threshold, upper\_threshold], ngược lại thì là “Normal”.  1. **Triển khai ML\_classifier (Random Forest)**  * Từ bộ dữ liệu **Car Hacking Attack Defense Challenge 2020,** sử dụng thư mục \0\_Preliminary\0\_Training để huấn luyện mô hình, thư mục \0\_Preliminary\1\_Submission để thử nghiệm mô hình. * Huấn luyện **RandomForestClassifier** để phân loại **5 lớp: Normal, Flooding, Fuzzing, Replay, Spoofing**. * Lưu mô hình (joblib) sau khi train.  1. **Thử nghiệm và đánh giá**  * Từ bộ dữ liệu Car Hacking Attack Defense Challenge 2020, sử dụng thư mục \0\_Preliminary\1\_Submission để thử nghiệm mô hình và đánh giá bằng classification\_report. * Chạy thử nghiệm tương tự bài báo, tinh chỉnh các tham số và train lại mô hình để có kết quả cao như bài báo. Tuy nhiên, thu được kết quả accuracy/precision/recall/f1 không cao như công trình gốc. |

## Các khó khăn, thách thức hiện tại khi thực hiện:

|  |
| --- |
| <mô tả khó khăn của nhóm (nếu có), ….>   * Hiệu suất thử nghiệm của mô hình không được cao như bài báo do không rõ cách huấn luyện mô hình ML. |

1. **TỰ ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH SO VỚI KẾ HOẠCH THỰC HIỆN:**

|  |
| --- |
| <VD: 60%>  95% |

1. **NHẬT KÝ PHÂN CÔNG NHIỆM VỤ:**

|  |  |  |
| --- | --- | --- |
| **STT** | **Công việc** | **Phân công nhiệm vụ** |
| 1 | **Tìm và thu thập dataset**  – Tải về dataset “Car Hacking Attack Defense Challenge 2020” từ IEEE DataPort.  – Liên hệ Prof. H.K. Kim để lấy dataset “Car‑Hacking Dataset for Intrusion Detection”. | - Thiên Ân |
| 2 | **Tiền xử lý dữ liệu CAN**  – Chuẩn hoá định dạng CSV, thêm cột SubClass.  – Viết parse\_csv\_row().  – Lọc giữ các bản tin Class = Normal. | - Thiên Ân  - Phương Đại |
| 3 | **Xây dựng đồ thị tuần tự**  – Chia thành windows 200 messages.  – Triển khai build\_window\_graph(). | - Phương Đại  - Công Danh |
| 4 | **Trích xuất đặc trưng đồ thị**  – Tính 3 đặc trưng (graph\_elapsed\_time, max\_degree, num\_edges). | - Thiên Ân |
| 5 | **Triển khai TH\_classifier (Threshold‑based)**  **– Tính ngưỡng với ε=0.03 trên tập Normal.**  **– Thử nghiệm dự đoán “Attack” nếu giá trị vượt ngưỡng.** | **- Thiên Ân**  - Công Danh |
| 6 | **Triển khai ML\_classifier (Random Forest)**  **– Chuẩn bị data train/test từ thư mục 0\_Training/1\_Submission.**  **– Huấn luyện và lưu mô hình RandomForest.** | - Công Danh  - Hữu Bình |
| 7 | **Thử nghiệm và đánh giá**  **– Chạy thử nghiệm, xuất kết quả đánh giá từ classification\_report.**  **– Tinh chỉnh tham số, so sánh kết quả thực nghiệm vs. bài báo.** | - Thiên Ân  - Hữu Bình  - Phương Đại  - Công Danh |

BÁO CÁO TỔNG KẾT CHI TIẾT

**Phần bên dưới của báo cáo này là tài liệu báo cáo tổng kết - chi tiết của nhóm thực hiện cho đề tài này.**

*Qui định: Mô tả các bước thực hiện/ Phương pháp thực hiện/Nội dung tìm hiểu (Ảnh chụp màn hình, số liệu thống kê trong bảng biểu, có giải thích)*

## Phương pháp thực hiện

*<Trình bày kiến trúc, thành phần của hệ thống trong bài báo>*

*<Trình bày kiến trúc, thành phần đã thực hiện (nội dung mà nhóm đã thực hiện)>*

### Kiến trúc, thành phần của hệ thống trong bài báo:

A diagram of a system

AI-generated content may be incorrect.

**Kiến trúc:**

1. **Classifier Configuration Module**

* Đây là module để tính các giá trị ngưỡng và huấn luyện mô hình ML sử dụng RandomForest. Bộ dữ liệu sử dụng cho module này đã được gắn nhãn (Normal/Attack), nhưng chỉ những messages CAN có nhãn Normal được dùng để tính giá trị ngưỡng cho TH\_classifier, trong khi những messages CAN có nhãn Normal hoặc Attack đều được dùng cho ML\_classifier.
* Khi dữ liệu đầu vào chỉ có nhãn Normal, module sẽ dùng để cấu hình cho TH\_classifier với quy trình: phân tách 200 messages liên tiếp thành 1 windows, xây dựng đồ thị tuần tự từ windows này và trích xuất 3 đặc trưng đồ thị, với mỗi đặc trưng tính được, module sẽ tính min và max các giá trị này và lưu lại để làm đầu ra là ngưỡng cho đặc trưng. Khi dữ liệu có cả nhãn Normal và Attack, module sẽ dùng huấn luyện model với quy trình gần như vậy: phân tách messages thành windows, xây dựng đồ thị và trích xuất đặc trưng tương tự, nhưng các giá trị đặc trưng này được lưu lại chung với windows và nhãn, sau đó dùng tất cả windows thu được để huấn luyện mô hình ML.

1. **Classification Module**

* Đây là module được áp dụng và thực thi để phát hiện và phân loại tấn công, đầu vào là các giá trị ngưỡng của 3 đặc trưng, mô hình ML đã được huấn luyện cùng với bộ dữ liệu chưa được gắn nhãn. Quy trình xử lý dữ liệu tương tự như trước cho đến bước trích xuất 3 đặc trưng đồ thị. Nếu chọn mô hình TH\_classifier dựa trên các ngưỡng đặc trưng để đánh giá liệu một cửa sổ là bình thường hay tấn công. Nếu chọn mô hình ML\_classifier sử dụng mô hình học máy đã huấn luyện để phân loại chi tiết loại tấn công.

**Thành phần chi tiết:**

1. Windowing Module

* Gom 200 message liên tiếp thành 1 windows để đảm bảo đủ ngữ cảnh cho phân tích đồ thị.

1. Graph Builder

* Xây dựng đồ thị tuần tự từ windows.

1. Feature Extractor

* Trích xuất 3 đặc trưng **Graph elapsed time -** ∆ timestamp giữa node cuối và đầu, **Max degree** – bậc lớn nhất (tổng bậc vào và ra) trong đồ thị, **Num edges -** tổng số cạnh trong đồ thị.

1. TH\_classifier (Threshold based binary classifier)

* Dùng các giá trị threshold đã được tính trên tập window chỉ chứa Normal phân loại các giá trị đặc trưng đồ thị đầu vào là Attack hoặc Normal.

1. ML\_classifier (Training ML based multiclass classifier)

* Nếu các giá trị đặc trưng đồ thị đầu vào là Attack, phân loại tiếp là kiểu tấn công nào.

### Kiến trúc nhóm đã thực hiện:

**Kiến trúc đã thực hiện:**

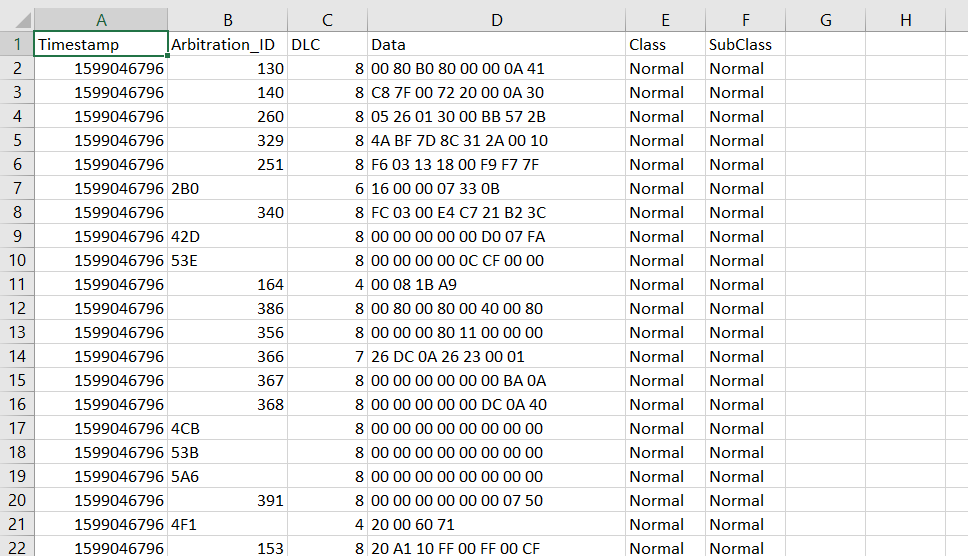
- Kiến trúc kết hợp 2 phương pháp chính: TH\_classifier (phân loại dựa trên ngưỡng – Thresold-based Classifier) và ML\_classifier (phân loại dựa trên học máy – Machine Learning classifier).

- Hệ thống được chia làm 2 giai đoạn chính: Huấn luyện (train.py) và kiểm thử (test.py). Kiến trúc bao gồm các bước sau:

* Bước 1: Tiền xử lý dữ liệu và chia cửa sổ:
  + Dữ liệu CAN được chia thành các cửa sổ có kích thước cố định (window\_size=200) để phân tích.
  + Mỗi cửa sổ được biểu diễn dưới dạng đồ thị có hướng (Directed Graph) và trích xuất đặc trưng đồ thị.
* Bước 2: Trích xuất đặc trưng:
  + Từ mỗi cửa sổ, trích xuất 3 đặc trưng đồ thị: graph\_elapsed\_time, max\_degree và num\_edges.
* Bước 3: Phân loại hai giai đoạn:
  + TH\_classifier: Phân loại dựa trên ngưỡng, sử dụng các ngưỡng được tính từ dữ liệu “Normal” trong giai đoạn huấn luyện.
  + ML\_classifier: Phân loại dựa trên học máy, sử dụng Random Forest để phân loại chi tiết các loại tấn công (Flooding, Fuzzing, Spoofing, Replay).
* Bước 4: Đánh giá hiệu suất:
  + Sau khi huấn luyện xong sẽ tiến hành kiểm thử cả hai bộ phân loại bằng các chỉ số như Accuracy, Precision, Recall và F1-score.

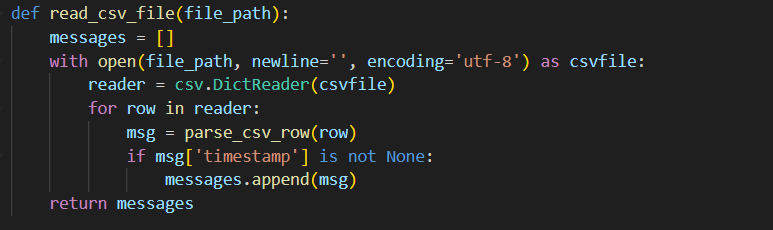
**Chi tiết các thành phần trong kiến trúc:**

- Tiền xử lý dữ liệu và chia cửa sổ: Chuyển đổi các file dữ liệu dùng để tính giá trị thành các đơn vị phân tích cửa sổ, từ đó xây dựng đồ thị và trích xuất đặc trưng.



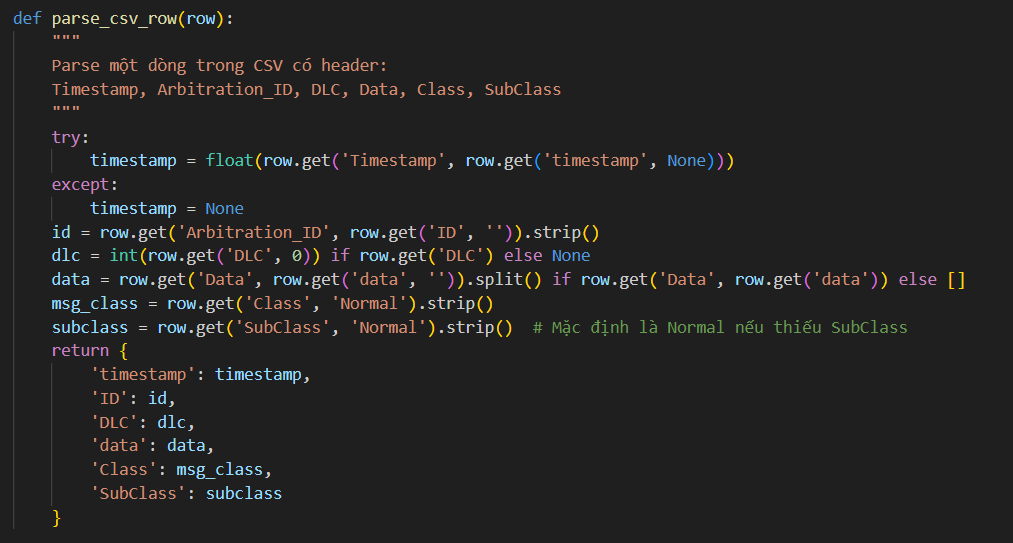
1. Dữ liệu ban đầu

* Hàm read\_csv\_file trong cả train.py và test.py đọc dữ liệu từ các file CSV trong bộ dữ liệu Car Hacking Challenge. Mỗi dòng CSV chứa các trường: Timestamp, Arbitration\_ID, DLC, Data, Class, Subclass.



1. Hàm read\_csv\_file()

* Hàm parse\_csv\_row chuyển đổi mỗi dòng CSV thành một dictionary với các trường: timestamp, arbitration\_ID, DLC, data, class và subclass. Các giá trị mặc định được gán (như Class=’Normal’, Subclass=’Normal’) để xử trường hợp dữ liệu thiếu.



1. Hàm parse\_csv\_row()

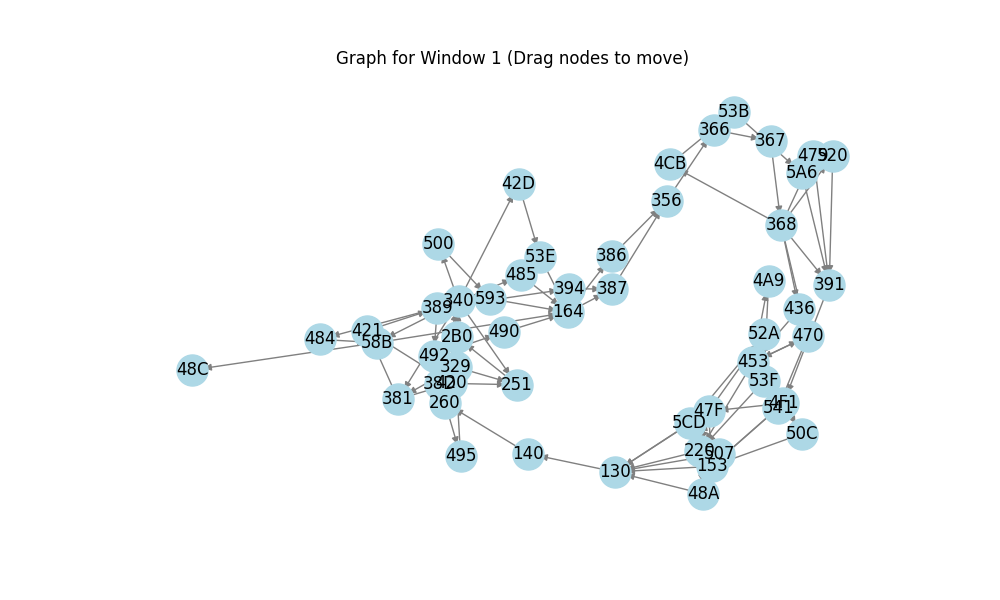
* Hàm process\_csv\_to\_windows chia dữ liệu thành các cửa sổ có kích thước window\_size=200 tin nhắn. Các cửa sổ có ít hơn 10 tin nhắn bị bỏ qua để đảm bảo đủ dữ liệu phân tích. Ở đây chương trình sẽ chia theo cách xử lý non-overlapping (jumping) window, tức là cửa sổ thứ 2 bắt đầu ngay sau khi cửa sổ thứ nhất kết thúc, vì trong bài báo có viết “*constructs a graph whenever the number of CAN messages reaches the pre-defined value, which is also known as the window size*”.

A screen shot of a computer program

AI-generated content may be incorrect.

1. Hàm process\_csv\_to\_windows()

- Xây dựng đồ thị và trích xuất đặc trưng: Biểu diễn mỗi cửa sổ dưới dạng đồ thị có hướng và trích xuất các đặc trưng để phân loại:



1. Đồ thị của cửa sổ đầu tiên

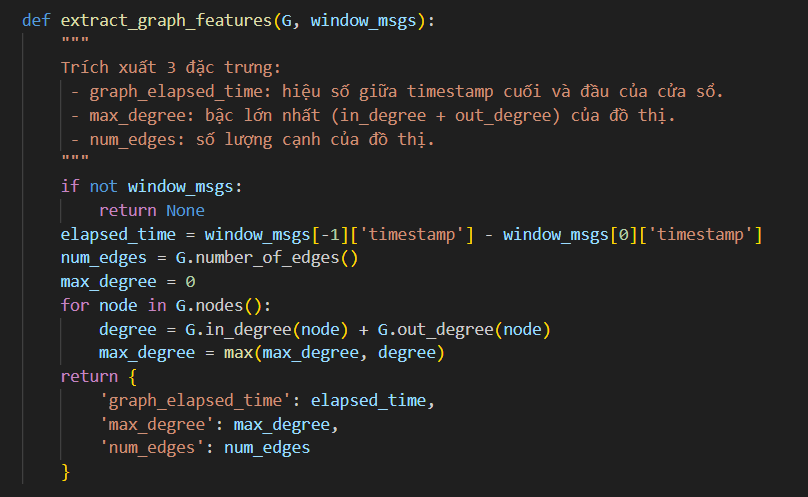
* Hàm build\_window\_graph xây dựng đồ thị có hướng từ mỗi cửa sổ. Nút: Đại diện cho Arbitration\_ID của các tin nhắn. Cạnh: Đại diện cho sự chuyển tiếp giữa hai tin nhắn liên tiếp (từ ID của tin nhắn hiện tại đến ID của tin nhắn kế tiếp.

A computer screen shot of a program code

AI-generated content may be incorrect.

1. Hàm build\_window\_graph()

* Hàm extract-graph\_features trích xuất 3 đặc trưng từ đồ thị:
  + graph\_elapsed\_time: Hiệu số giữa timestamp của tin nhắn đầu và cuối trong cửa sổ, phản ánh thời gian tổng cộng của của cửa sổ.
  + max\_degree: Bậc lớn nhất (tổng in\_degree và out\_degree) của một nút trong đồ thị, phản ánh mức độ kết nối cao nhất giữa các Arbitration\_ID.
  + num\_edges: Số lượng cạnh trong đồ thị, tương ứng với số lần chuyển tiếp giữa các tin nhắn liên tiếp.

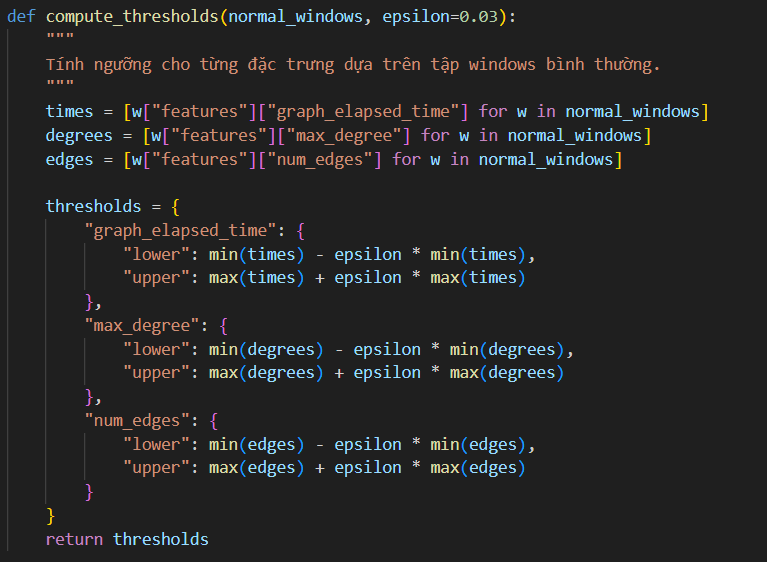


1. Hàm extract\_graph\_features()

* Đồ thị có hướng giúp mô hình hóa mối quan hệ thời gian giữa các tin nhắn CAN, từ đó phát hiện các hành vi bất thường (như lặp lại ID trong Replay hoặc kết nối bất thường trong Flooding).
* graph\_elapsed\_time nhạy với tần suất tin nhắn (cao trong Flooding, thấp trong Replay), max\_degree nhạy với mức độ kết nối bất thường (cao trong Flooding và Fuzzing), num\_edges nhạy với số lượng chuyển tiếp (cao trong tất cả các cuộc tấn công).

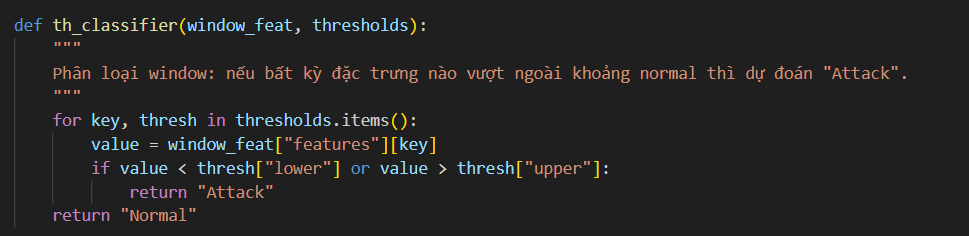
- TH\_classifier (Phân loại dựa trên ngưỡng): Phân loại sơ bộ các cửa sổ thành ‘Normal’ hoặc ‘Attack’ dựa trên ngưỡng của các đặc trưng.

* Trong huấn luyện (train.py):
  + Hàm compute\_thresholds tính ngưỡng cho từng đặc trưng (graph\_elapsed\_time, max\_degree, num\_edges) dựa trên các cửa sổ “Normal”:
    - Ngưỡng dưới (lower): Giá trị nhỏ nhất trừ đi một khoảng (epsilon=0.03).
    - Ngưỡng trên (upper): Giá trị lớn nhất cộng thêm một khoảng (epsilon=0.03).



1. Hàm compute\_threasolds()

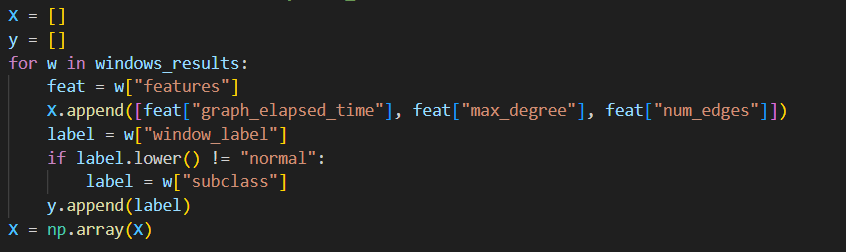
* Kiểm tra (test.py):
  + Hàm th\_classifier kiểm tra từng đặc trưng của cửa sổ:
    - Nếu bất kỳ đặc trưng nào nằm ngoài ngưỡng (lower, upper), của sổ được phân loại là “Attack”.
    - Ngược lại, cửa số đó sẽ được phân loại là “Normal”.



1. Hàm th\_classifier()

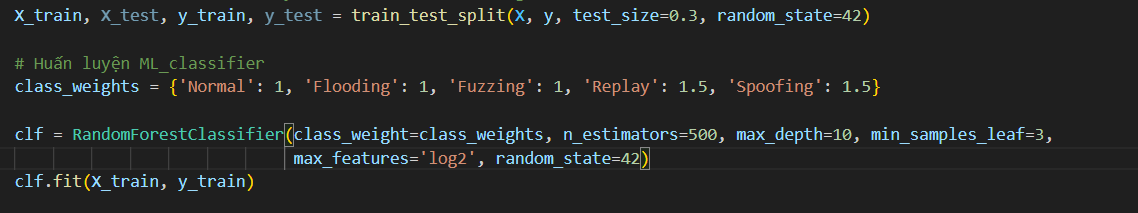
- ML\_classifier (Phân loại dựa trên học máy): Phân loại chi tiết các cửa sổ “Attack” thành các loại tấn công cụ thể (Flooding, Fuzzing, Spoofing, Replay) bằng thuật toán Random Forest. Random Forest là một thuật toán học máy thuộc nhóm ensemble learning, cụ thể là Bagging (Bootstrap Aggregating). Nó được xây dựng dựa trên nhiều cây quyết định (decision trees) và tổng hợp kết quả đầu ra từ tất cả các cây đó để đưa ra quyết định cuối cùng. Mỗi cây trong rừng sẽ đưa ra một dự đoán riêng, và Random Forest sẽ sử dụng đa số phiếu bầu (majority voting) để quyết định nhãn cuối cùng trong bài toán phân loại.

* Trong huấn luyện (train.py):
* Tạo x (ma trận đặc trưng) với 3 cột: graph\_elapsed\_time, max\_degree, num\_edges.
* Tạo y (vector nhãn) với các lớp: Normal, Flooding, Fuzzing, Spoofing, Replay.



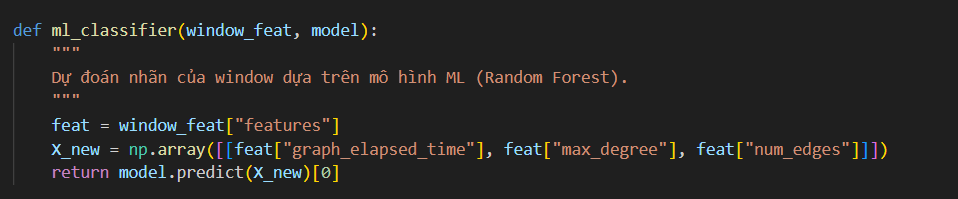
1. Xử lý dữ liệu để huấn luyện ML

* Chia dữ liệu thành tập huấn luyện và kiểm tra nội bộ (70% huấn luyện, 30% kiểm tra) bằng train\_test\_split.
* Khởi tạo và huấn luyện mô hình Random Forest. Sử dụng RandomForest
  + n\_estimators=500: Số lượng cây quyết định trong rừng.
  + Max\_depth=10: Độ sâu tối đa của mỗi cây.
  + Min\_sample\_leaf=3: Số mẫu tối thiểu tại mỗi lá.
  + Max-features=’log2’: Số đặc trưng tối đa để chia mỗi nút).
  + class\_weight={'Normal': 1, 'Flooding': 1, 'Fuzzing': 1, 'Replay': 1.5, 'Spoofing': 1.5}: Trọng số để xử lý mất cân bằng dữ liệu.
  + random\_state=42: Đảm bảo tính tái hiện.



1. Train mô hình Random Forest

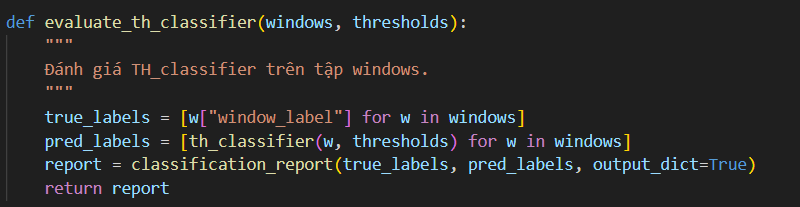
* Trong kiểm tra (test.py):
  + Hàm ml\_classifer dự đoán nhãn của mỗi cửa sổ bằng cách sử dụng mô hình Random Forest đã huấn luyện.



1. Hàm ml\_classifier()

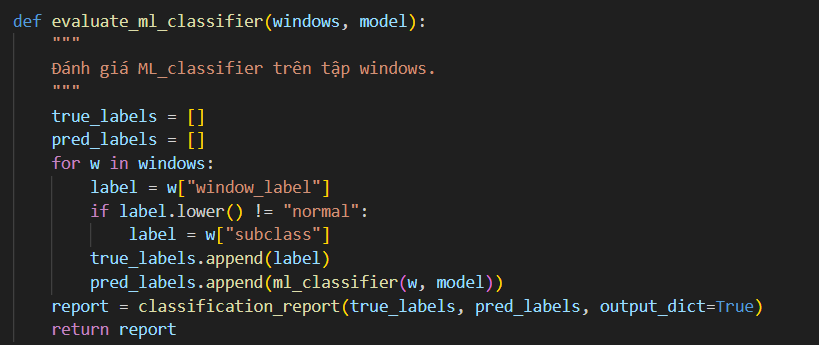
- Đánh giá hiệu suất: Đánh giá hiệu suất cho cả TH\_classifier và ML\_classifier trên tệp kiểm tra.

* Hàm evaluate\_th\_classifier tính các chỉ số Precision, Recall, F1-score cho hai lớp “Normal” và “Attack” cho TH\_classifier.



1. Hàm evaluate\_th\_classifier()

* Hàm evaluate\_ml\_classifier tính các chỉ số Precision, Recall, F1-score cho tất cả các lớp (Normal, Flooding, Fuzzing, Spoofing, Replay) cho ML\_classifier.



1. Hàm evaluate\_ml\_classifier()

## Chi tiết cài đặt, hiện thực:

*<cách cài đặt, lập trình trên máy tính, cấu hình máy tính sử dụng, chuẩn bị dữ liệu, v.v>*

**Cấu hình máy tính:**

* CPU i5-12450HX 2.4 Ghz.
* RAM 20GB,
* SSD 1TB,
* Hệ điều hành Windows 11.

**Ngôn ngữ lập trình:** Python 3.12.7.

**Thư viện**: networkx, scikit‑learn, joblib, numpy.

**Chuẩn bị dữ liệu:**

- Car Hacking Attack Defense Challenge 2020:

* Bước 1: Tải bộ dữ liệu về từ [IEEE DataPort](https://ieee-dataport.org/open-access/car-hacking-attack-defense-challenge-2020-dataset).

A screenshot of a computer

AI-generated content may be incorrect.

1. Tải bộ dữ liệu từ IEEE

* Bước 2: Giải nén và truy cập vào thư mục vừa giải nén.

A screenshot of a computer

AI-generated content may be incorrect.

1. Giải nén và truy cập vào thư mục vừa giải nén

* Bước 3: Truy cập vào thư mục 0\_Preliminary, trong đây có hai thư mục con là 0\_Training và 1\_Submission.

A screenshot of a computer

AI-generated content may be incorrect.

1. Vào thư mục 0\_Preliminary

* Bước 4: Truy cập vào thư mục 0\_Training, copy các file CSV vào thư mục Train\_ML để train model. Tiếp đó, truy cập vào thư mục 1\_Submission và copy các file CSV vào thư mục Test\_ML để thử nghiệm model.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. Các file .csv chứa dữ liệu CAN

- Car-Hacking Dataset for Intrusion Detection:

* Bước 1: Tải bộ dữ liệu về từ [Hacking and Countermeasure Research Lab](https://ocslab.hksecurity.net/Datasets/car-hacking-dataset).

A screenshot of a computer

AI-generated content may be incorrect.

1. Tiến hành tải bộ dữ liệu

* Bước 2: Giải nén và truy cập vào thư mục vừa giải nén.

A screenshot of a computer

AI-generated content may be incorrect.

1. Truy cập thư mục vừa giải nén

* Bước 3: Truy cập vào thư mục normal\_run\_data và copy file TXT vào thư mục Train\_TH để tính các giá trị cho phân loại dựa trên ngưỡng.

A screenshot of a computer

AI-generated content may be incorrect.

1. File chỉ chứa dữ liệu normal

* Bước 4: Copy 4 file CSV ở thư mục này vào thư mục Test\_TH để thử nghiệm phân loại dựa trên ngưỡng.

A screenshot of a computer

AI-generated content may be incorrect.

1. File chứa các dữ liệu attack và normal

* Lúc này các file dữ liệu dùng để tính giá trị và thử nghiệm phân loại dựa trên ngưỡng chưa đồng bộ về định dạng (CSV, TXT) và các file chưa có header nên chưa thể sử dụng.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

1. Dữ liệu trong file .csv

* Dựa trên mô tả của tác giả về bộ dữ liệu, tiến hành viết chương trình Python để convert các file này về chung định dạng.

A black text on a white background

AI-generated content may be incorrect.

1. Mô tả về dataset

* Để thuận tiện việc xử lý sau này, có thể sử dụng lại các header của các file dùng huấn luyện và thử nghiệm mô hình ML.

A screen shot of a computer

AI-generated content may be incorrect.

1. Dữ liệu trong file .csv

* File processed.py có nội dung như sau:

A screen shot of a computer code

AI-generated content may be incorrect.

1. Hàm process\_normal\_txt()
   * Hàm process\_normal\_txt() dùng để chuyển file gốc normal\_run\_data.txt về file CSV chuẩn với các cột cố định. Sử dụng Regex để lấy ra các giá trị:

* **Timestamp: (\d+\.\d+)** – bắt phần số thực (ví dụ 123.456) sau “Timestamp: ”.
* **.\*?ID: (\w+)** – sau đó tìm “ID: ” rồi nhóm ký tự chữ/ số (Arbitration\_ID).
* **.\*?DLC: (\d+)** – rồi tới “DLC: ” và số nguyên.
* **\s+([\w\s]+)$** – ở cuối dòng, lấy chuỗi các ký tự và dấu cách làm Data. Giá trị này được tách bằng .split() rồi ghép lại để loại bỏ khoảng trắng dư thừa.

Nội dung ghi ra CSV thêm vào hai cột Class và SubClass với giá trị là 'Normal', vì đây là dữ liệu thuần Normal.

A screen shot of a computer code

AI-generated content may be incorrect.

1. Hàm process\_hacking\_csv()

* Hàm process\_hacking\_csv() dùng để chuyển 4 file CSV có chứa dữ liệu 4 loại tấn công về chung định dạng. Dùng csv.reader để đọc các dòng thành danh sách các row:
* row (row[0] → Timestamp, row[1] → ID, row[2] → DLC)
* data\_bytes = row[3:-1] – lấy tất cả cột dữ liệu giữa cột thứ 4 và cột cuối (nhãn T/F).
* Nếu cột cuối row[-1] == 'T', gán Class='Attack' và SubClass=attack\_type; ngược lại gán Normal/Normal.

A screen shot of a computer program

AI-generated content may be incorrect.

1. Đặt giá trị đường dẫn input, output
   * Phần thân chương trình, tiến hành đặt giá trị đường dẫn file đầu vào, đầu ra và gọi hàm tương ứng để xử lý.

* Kết quả sau khi chạy file processed.py sẽ có các file CSV chung định dạng như sau:

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

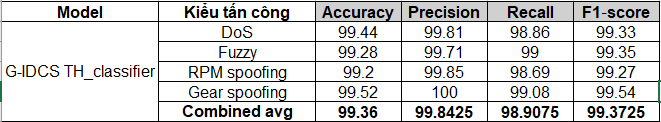
1. Kết quả đồng bộ dữ liệu

## Kết quả thực nghiệm

*<mô tả hình ảnh về thực nghiệm, bảng biểu số liệu thống kê từ thực nghiệm, nhận xét về kết quả thu được.>*

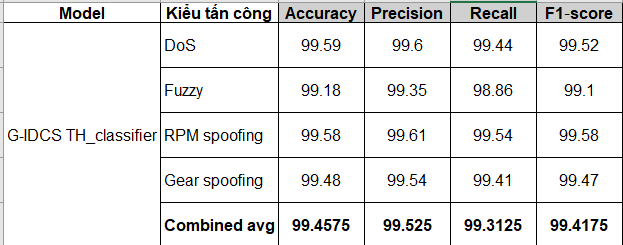
Đối với model TH\_classifier nhóm sẽ tiến hành kiểm thử trên dataset CAR-HACKING DATASET FOR INTRUSION DETECTION:

* Kết quả từ bài báo khi chạy model TH\_classifier trên dataset CAR-HACKING DATASET FOR INTRUSION DETECTION.

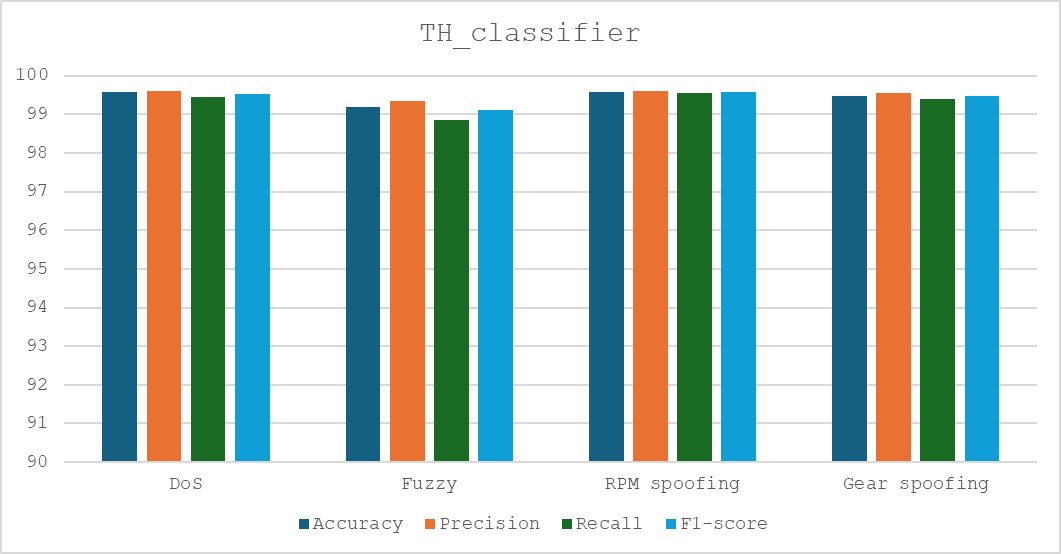


1. Kết quả từ bài báo

* Kết quả của nhóm khi chạy cùng với dataset trên:



1. Kết quả của nhóm



1. Biểu đồ khi chạy TH\_classifier

* Nhận xét:
  + DoS: F1-score đạt 99.52%, với Recall (99.44%) cao hơn so với bài báo (98.86%), cho thấy hệ thống phát hiện tốt hơn các mẫu tấn công DoS.
  + Fuzzy: F1-score đạt 99.1%, với Recall (98.86%) tương đương bài báo (99%), nhưng Precision (99.35%) thấp hơn một chút (99.71%), có thể do một số mẫu "Normal" bị phân loại nhầm.
  + RPM spoofing: F1-score đạt 99.58%, với Recall (99.54%) cao hơn đáng kể so với bài báo (98.69%), cho thấy cải thiện trong việc phát hiện mẫu tấn công này.
  + Gear spoofing: F1-score đạt 99.47%, với Recall (99.41%) cao hơn bài báo (99.08%), nhưng Precision (99.54%) thấp hơn so với 100% của bài báo.

Đối với model ML\_classifier nhóm sẽ tiến hành kiểm thử trên dataset CAR HACKING-ATTACK & DEFENSE CHALLENGE:

* Kết quả từ bài báo khi chạy model ML\_classifier trên dataset CAR HACKING-ATTACK & DEFENSE CHALLENGE:



1. Kết quả từ bài báo

* Kết quả của nhóm khi chạy cùng với dataset ở trên:



1. Kết quả của nhóm

* Nhận xét:
  + Accuracy: Hệ thống của nhóm thấp hơn bài báo 2.02%, cho thấy mô hình của nhóm chưa đạt được hiệu suất tối ưu như bài báo.
  + Precision: Precision của nhóm thấp hơn bài báo 3.86%, mô hình của nhóm có tỉ lệ false positives cao hơn, tức là phân loại nhầm nhiều mẫu “Normal” thành tấn công.
  + Recall: Recall của nhóm thấp hơn bài báo 1.28%, nhưng sự khác biệt này nhỏ, cho thấy khả năng phát hiện tấn công từ mô hình của bài báo tương đối tốt.
  + F1-score: Kết quả của bài báo cao hơn hệ thống của nhóm 3.61%.

## Hướng phát triển

*<Nêu hướng phát triển tiềm năng của đề tài này trong tương lai. Nhận xét về tính ứng dụng của đề tài>.*

Hướng phát triển tiềm năng:

1. Áp dụng các thuật toán học máy tiên tiến hơn:

- XGBoost hoặc LightGBM: Các thuật toán dựa trên gradient boosting, có khả năng xử lý dữ liệu mất cân bằng tốt hơn và thường đạt hiệu suất cao hơn Random Forest.

- Mạng nơ-ron sâu (Deep Neural Networks): Sử dụng các mô hình như LSTM hoặc GNN (Graph Neural Networks) để trực tiếp học từ đồ thị, tận dụng mối quan hệ thời gian và cấu trúc của dữ liệu CAN.

2. Mở rộng cho các giao thức khác:

- Ngoài mạng CAN, có thể mở rộng hệ thống để áp dụng cho các giao thức khác trong ô tô như CAN-FD (CAN with Flexible Data Rate), Ethernet, hoặc LIN (Local Interconnect Network), vốn đang trở nên phổ biến trong các xe hiện đại.

Nhận xét về tính ứng dụng:

* Hệ thống phát hiện xâm nhập trên mạng CAN có tính ứng dụng cao trong ngành công nghiệp ô tô, đặc biệt trong bối cảnh các phương tiện ngày càng kết nối (connected vehicles) và phát triển xe tự hành.
* Bảo mật xe ô tô: Hệ thống có thể được tích hợp trực tiếp vào các phương tiện để phát hiện và cảnh báo các cuộc tấn công mạng như Flooding, Fuzzing, Spoofing, và Replay, từ đó bảo vệ các chức năng quan trọng (phanh, động cơ, lái tự động).
* An toàn giao thông: Việc phát hiện kịp thời các cuộc tấn công giúp ngăn chặn các hành vi nguy hiểm (như giả mạo tín hiệu phanh), góp phần đảm bảo an toàn cho người lái và hành khách.
* Tuân thủ tiêu chuẩn: Các tiêu chuẩn an ninh mạng trong ô tô (như ISO/SAE 21434) yêu cầu các nhà sản xuất xe triển khai hệ thống IDS. Hệ thống này có thể đáp ứng yêu cầu đó, giúp các công ty tuân thủ quy định.

*Sinh viên báo cáo các nội dung mà nhóm đã thực hiện, có thể là 1 phần hoặc toàn bộ nội dung của bài báo. Nếu nội dung thực hiện có khác biệt với bài báo (như cấu hình, tập dữ liệu, kết quả,...), sinh viên cần chỉ rõ thêm khác biệt đó và nguyên nhân.*

---

***Sinh viên đọc kỹ yêu cầu trình bày bên dưới trang này***

# **YÊU CẦU CHUNG**

* Sinh viên tìm hiểu và thực hiện bài tập theo yêu cầu, hướng dẫn.
* Nộp báo cáo kết quả chi tiết những việc (**Report**) bạn đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả (nếu có); giải thích cho quan sát (nếu có).
* Sinh viên báo cáo kết quả thực hiện và nộp bài.

**Báo cáo:**

* File .PDF. Tập trung vào nội dung, không mô tả lý thuyết.
* Đặt tên theo định dạng: [Mã lớp]-Project\_Final\_NhomX\_Madetai. (trong đó X và Madetai là mã số thứ tự nhóm và Mã đề tài trong danh sách đăng ký nhóm đồ án).

*Ví dụ: [NT521.N11.ANTT]-Project\_Final\_Nhom03\_CK01.*

* Nếu báo cáo có nhiều file, nén tất cả file vào file .ZIP với cùng tên file báo cáo.
* Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

**Đánh giá**:

* Hoàn thành tốt yêu cầu được giao.
* Có nội dung mở rộng, ứng dụng.

*Bài sao chép, trễ, … sẽ được xử lý tùy mức độ vi phạm.*

**HẾT**

1. Ghi nội dung tương ứng theo mô tả [↑](#footnote-ref-1)