**ML Model analysis for CAN Bus Intrusion Detection Systems – Cybersecurity**

Saumya R. Padhi

(GitHub: https://github.com/saumyasam/analytics\_papers/tree/main/Can\_BUS)

DeVos Graduate School, Northwood University

MGT 661: Artificial Intelligence & Business Analytic Graduate

Dr. Itauma Itauma

March 3rd, 2025

**ML Model analysis for CAN Bus Intrusion Detection Systems**

**Abstract**

Every moving vehicle has an equipment called Controller area network (CAN Bus) for in-vehicle communication. This has been very standard practice followed by every automobile or respective auto parts manufacturers. However, the need for compact built mechanisms of these CAN buses, they lacks authentication, injection, encryption and relevant security features. Thus, making it an easy candidate for cyber-attacks. This raised the need to develop better Intrusion Detection Systems (IDSs). This paper compares the effectiveness of six machine learning models, which were filtered out from tens of more studies. As mentioned in (Rajapaksha et al., 2024) The dataset includes 17 hours of benign data, various scenarios, crucial for training IDSs. In addition, it comprises physically verified real injection attacks, including Denial-of-Service (DoS), fuzzing, replay, and spoofing. This paper aims to compare various models and validation in-vehicle network Intrusion Detection Systems.

**Introduction**

As modern vehicles integrate increasingly sophisticated Controller Area Network (CAN) bus systems, their vulnerability to cyber threats grows exponentially. The CAN bus, a most important in-vehicle communication network, was not originally designed with security in place. These devices are prone to highly susceptible to cyberattacks such as spoofing, denial-of-service (DoS), and message injection, due to the lack of built-in security features during manufacturing of CAN bus systems. Provided the critical role of CAN bus for in-vehicle communications in ensuring vehicle safety and efficiency of operations, securing these networks is highly essential.

Traditional intrusion detection systems (IDSs) for automotive cybersecurity greatly depend on methods which are rule-based, hence struggle with detecting real world threats. As cyber threats become more ultramodern, machine learning (ML)-based intrusion detection systems offer a favourable alternative. Based on availability of large datasets of CAN bus traffic, ML models can learn normal communication patterns and detect anomalies of malicious activities.

This paper aims in investigating the application of ML models for CAN bus intrusion detection using the CAN MIRGU dataset (UCI Machine Learning Repository, 2024). The dataset comprises both normal and attack scenarios, making it a suitable benchmark for evaluating model performance. The study explores several ML approaches, from classical algorithms like Logistic Regression and SVM to advanced models like Random Forest, XGBoost, and LSTM. Through comparative analysis, the study assesses their efficacy in detecting CAN bus anomalies and propose enhancements for further real-world deployment.

**Literature Review**

Research and studies on CAN bus security has nurtured significant attention due to the increasing number of cyberattacks in the vehicles. Earlier approaches relied on rule-based anomaly detection systems that used predefined fact findings to identify and flag suspicious messages. these systems lacked adaptability to newer threats even while the attack types are known.

Machine learning-based intrusion detection systems (IDSs) has shown as a better alternative, which offers the ability to predict from training data and detect unseen anomalies. Harsha et al. (2023) explored supervised learning techniques for automotive cybersecurity, demonstrating that ensemble learning models outperform traditional rule-based approaches. Moreover, McAuley and Leskovec (2013) highlighted the advantages of deep learning, particularly LSTMs, in identifying sequential anomalies in CAN bus traffic.

Though there have been positive prospects of Machine learning based Intrusion detection seen, still there are major challenge remains alive due to the availability of datasets, feature selection, and model interpretability. Many publicly available CAN datasets studied by researchers, usually lack diverse attack scenarios, limiting model adaptability. Moreover, black-box ML models lead to more challenges in understanding decision-making processes, making them less suitable for cybersecurity vertical. These issues are still important for the widespread adoption of AI-driven intrusion detection systems.

Moreover, the selection of features and preprocessing of dataset plays a vital role in identifying the effectiveness of Machine learning models used for Intrusion detection solutions. The major concern with this dataset was with CAN bus messages consists of structure but they has limited number of data fields such as payloads and message IDs, thus creating more challenges in extracting meaningful features those distinguish between normal versus attacking activity.

In summary the machine learning based intrusion detection systems has show significant adaptability over traditional rule-based approaches, several more challenges still need to be addressed considering real world application. Future research opportunities should include more improved dataset quality which will influence more effective model analysis and more feature engineering.

**Methodology**

**Problem Formulation**

This paper aims of this study is to evaluate the performance of ML models in detecting intrusions within CAN bus in-vehicle communication networks. By analysing normal versus malicious CAN traffic, the goal is to identify models that can effectively differentiate between legitimate and anomalous messages.

**Data Sources and Preprocessing**

The CAN MIRGU dataset, which consists of normal and attack data, was used for training and testing. CAN Bus message structure from log file.

* A screenshot of a computer

  AI-generated content may be incorrect.

[Image 1]

* Image 1 describes the CAN MIRGU dataset. It contains CAN bus data from an Opel Astra for intrusion detection research.

A graph of a number of columns

AI-generated content may be incorrect.

Image 1.1

Image 1.1 shows two histograms, one for payload\_length and one for time\_diff, likely from a CAN bus dataset.

A screenshot of a computer

AI-generated content may be incorrect.

[Image 1.2]

Image 1.2 illustrates the structures of CAN Bus logs captured in text files.

1. **CAN Bus Log Structure:**
   * The dataset contains **timestamps, CAN IDs, and payloads** from the CAN bus logs.
   * The timestamps are in **epoch format** but have been successfully converted to a readable **datetime format** (e.g., 2018-09-10 10:06:04.242068).
   * **CAN IDs** (e.g., 1C8, 1E9, 232, etc.) represent unique message identifiers sent over the CAN bus.
   * **Payload** contains hexadecimal data representing actual message contents.
2. **Payload Length Analysis:**
   * The payload\_length column indicates the **size of each CAN message** in bytes.
   * The first three messages have **16-byte payloads**, while the next two have **10-byte payloads**.

**Descriptive Statistics:**

A screenshot of a computer

AI-generated content may be incorrect.

[Image 1.3]

Image 1.3 provides a statistical summary of a CAN bus dataset. Following is a break down of the findings:

**BASIC Statistics:**

* **total\_messages: 2690069:** This indicates the total number of CAN frames captured in the dataset. A large number suggests a substantial recording period.
* **unique\_can\_ids: 85:** There are 85 distinct CAN IDs present in the dataset, meaning there are 85 different types of messages being transmitted on the CAN bus.
* **data\_duration\_seconds: 1382.225713968277:** The data was captured over approximately 1382 seconds (about 23 minutes).
* **messages\_per\_second: 1946.1864822909372:** On average, approximately 1946 CAN frames were captured per second. This is a very high message rate, indicating a busy CAN bus.

**CAN\_ID Statistics:**

* **most\_frequent\_can\_id: 1E5:** The CAN ID "1E5" (hexadecimal) is the most frequently occurring message type in the dataset.
* **messages\_most\_frequent: 138277:** The CAN ID "1E5" appears 138,277 times in the dataset.
* **least\_frequent\_can\_id: 120:** The CAN ID "120" (hexadecimal) is the least frequently occurring message type.
* **messages\_least\_frequent: 276:** The CAN ID "120" appears 276 times in the dataset.

**TEMPORAL Statistics:**

* **mean\_interval: 0.0005138255664794634:** The average time interval between consecutive CAN frames is approximately 0.00051 seconds (0.51 milliseconds). This confirms the high message rate calculated earlier.
* **median\_interval: 0.0002410411834716797:** The median time interval is approximately 0.00024 seconds (0.24 milliseconds). The fact that the median is lower than the mean suggests that the data might be skewed towards shorter intervals, with occasional longer intervals.
* **std\_interval: 0.0007212248356755257:** The standard deviation of the time intervals is approximately 0.00072 seconds. This indicates the variability in the time between messages.

Image 1.2

**1. payload\_length Histogram:**

X-axis: Represents the length of the payload in bytes. The values range from 2 to 16 bytes.

Y-axis: Represents the frequency or count of CAN frames with each payload length.

**2. time\_diff Histogram:**

X-axis: Represents the time difference (in seconds) between consecutive CAN frames. The values range from 0.0 to 0.012 seconds (12 milliseconds).

Y-axis: Represents the frequency or count of time differences falling within each bin.

**Busy CAN Bus**: Both histograms point to a busy CAN bus with a high message rate and a predominance of messages with 16-byte payloads.

**Possible Real-Time System**: The skewed distribution of time\_diff suggests that the system might have real-time constraints, where messages need to be transmitted with minimal delay.

**Payload Length Consistency**: The consistency in payload length (mostly 16 bytes) could indicate a specific protocol or system design.

**Further Analysis:** These histograms provide a general overview, and further analysis is needed to understand the specific context and meaning of the data.

The preprocessing steps included:

* **Timestamp conversion:** Epoch timestamps were converted to human-readable formats for temporal analysis.
* **Feature extraction:** CAN IDs, payload lengths, and message intervals were derived.
* **Normalization:** Data was scaled to ensure consistent input distribution across models.

**Models Used**

The following ML models were evaluated against the dataset:

* **Logistic Regression**
* **Support Vector Machine (SVM)**
* **K-Nearest Neighbors (KNN)**
* **Random Forest**
* **XGBoost**
* **Long Short-Term Memory (LSTM)**

Each model was trained using a 70-30 train-test split, with hyperparameter tuning performed to optimize performance.

**Results**

**Model Comparison**

A graph of different colored rectangular shapes

AI-generated content may be incorrect.

Image 2

Image 2 represents the performance of different machine learning models.

**Key Observations:**

**X-axis (Models):** The x-axis lists the different machine learning models being compared: Logistic Regression, SVM (Support Vector Machine), KNN (K-Nearest Neighbors), Random Forest, XGBoost, and LSTM (Long Short-Term Memory).

**Y-axis (Accuracy):** The y-axis represents the accuracy score, ranging from 0.0 to 1.0. Higher values indicate better performance.

**Bar Heights:** The height of each bar corresponds to the accuracy score of the respective model. Taller bars represent higher accuracy.

**Color Coding:** Each model is represented by a distinct color, making it easy to visually compare their performances.

**Specific Model Performance:**

The following is a breakdown of each model:

**Logistic Regression:** Shows the lowest accuracy, with a bar height around 0.49. This indicates poor performance, suggesting the model is not well suited for the task.

**SVM**: Achieves a slightly higher accuracy, with a bar height around 0.56. This is still relatively low, indicating limited effectiveness.

**KNN**: Performs similarly to SVM, with a bar height around 0.55. This suggests marginal improvement over Logistic Regression.

**Random Forest:** Shows significantly higher accuracy, with a bar reaching 1.0. This indicates perfect accuracy on the evaluation set.

**XGBoost:** Also achieves perfect accuracy, with a bar reaching 1.0.

**LSTM**: Similarly, achieves perfect accuracy, with a bar reaching 1.0.

**Overall Analysis:**

**Performance Discrepancy**: The chart clearly demonstrates a substantial performance difference between the first three models (Logistic Regression, SVM, KNN) versus the last three (Random Forest, XGBoost, LSTM).

**Perfect Accuracy:** Random Forest, XGBoost, and LSTM all achieve perfect accuracy, which is highly unusual and warrants further investigation.

**Model Selection:** Based on this chart, Random Forest, XGBoost, or LSTM would be the preferred models due to their significantly higher accuracy.

**Overfitting**: The perfect accuracy of Random Forest, XGBoost, and LSTM raises concerns about overfitting. These models might have memorized the training data instead of learning generalizable patterns.

**Data Leakage/Easy Dataset**: The perfect scores also suggest the possibility of data leakage (since the dataset was generated in custodial circumstances) or an may be an exceptionally easy dataset.

**Model evaluation:**

A screenshot of a document

AI-generated content may be incorrect.

[Image 3]

Image 3 represents the **Mean Model Evaluation Results** for about six machine learning models. Following is a break down the interpretation of each model's performance:

**Metrics:**

* **Accuracy:** The proportion of correctly classified instances.
* **Precision:** The proportion of correctly predicted positive instances out of all instances predicted as positive.
* **Recall:** The proportion of correctly predicted positive instances out of all actual positive instances.
* **F1-score:** The harmonic means of precision and 1 recall, providing a balance between the two.
* **AUC (Area Under the ROC Curve):** A measure of the model's ability to distinguish between positive and negative classes.

**Conclusion**

* **Linear models (Logistic Regression, SVM, KNN)** performed poorly, highlighting their limitations in capturing complex attack patterns.
* **Ensemble and deep learning models (Random Forest, XGBoost, LSTM)** achieved perfect accuracy, suggesting their robustness in intrusion detection. However, overfitting concerns necessitate further validation.
* **Feature importance analysis** revealed that CAN ID frequency and payload length were key indicators for detecting anomalous activity.

This study demonstrates the potential of ML models in enhancing CAN bus cybersecurity. The superior performance of Random Forest, XGBoost, and LSTM models highlights their suitability for intrusion detection. However, achieving high accuracy in controlled environments does not guarantee real-world efficacy. Future research should focus on:

* **Addressing overfitting risks** by incorporating adversarial training and cross-validation techniques.
* **Exploring real-time deployment** of ML-based intrusion detection systems in automotive networks.
* **Enhancing interpretability** of black-box ML models to facilitate adoption in cybersecurity applications.

By bridging the gap between theoretical research and practical implementation, AI-driven intrusion detection can help practically bolster cybersecurity for in-vehicle communication.

**References**

* Harsha, K., Nitya, S. Y., Kota, S., Satyanarayana, K., & Lakshmi, J. (2023). Empirical evaluation of Amazon fine food reviews using Text mining.
* McAuley, J., & Leskovec, J. (2013). From amateurs to connoisseurs: Modeling the evolution of user expertise through online reviews. [WWW](http://www/).
* Rajapaksha, S., Madzudzo, G., Kalutarage, H., Petrovski, A., & Al-Kadri, M. O. (2024). CAN-MIRGU: A comprehensive CAN bus attack dataset from moving vehicles for intrusion detection system evaluation. In \*Symposium on Vehicles Security and Privacy Internet Society\*.
* C. Miller, “Lessons learned from hacking a car,” IEEE Design & Test, vol. 36, no. 6, pp. 7–9, 2019.
* Z. Cai, A. Wang, W. Zhang, M. Gruffke, and H. Schweppe, “0-days & mitigations: roadways to exploit and secure connected bmw cars,” Black Hat USA, vol. 2019, no. 39, p. 6, 2019.
* S. Nie, L. Liu, and Y. Du, “Free-fall: Hacking tesla from wireless can bus,” Briefing, Black Hat USA, vol. 25, no. 1, p. 16, 2017.
* S. N. Narayanan, S. Mittal, and A. Joshi, “Obd securealert: An anomaly detection system for vehicles,” in 2016 IEEE International Conference on Smart Computing (SMARTCOMP), 2016, pp. 1–6.
* E. Aliwa, O. Rana, C. Perera, and P. Burnap, “Cyberattacks and countermeasures for in-vehicle networks,” ACM Computing Surveys (CSUR), vol. 54, no. 1, pp. 1–37, 2021.
* F. Luo, J. Wang, X. Zhang, Y. Jiang, Z. Li, and C. Luo, “In-vehicle network intrusion detection systems: a systematic survey of deep learning-based approaches,” PeerJ Computer Science, vol. 9, p. e1648, 2023.
* M. E. Verma, M. D. Iannacone, R. A. Bridges, S. C. Hollifield, B. Kay, and F. L. Combs, “Road: The real ornl automotive dynamometer controller area network intrusion detection dataset (with a comprehensive can ids dataset survey & guide),” arXiv preprint arXiv:2012.14600, 2020.