LOCoCAT: Low-Overhead Classification of CAN Bus Attack Types

Aneeshvar R Gunari[1] , Dr. Dilip R[2]

Student, Department of Electronics and Communication, Dayananda Sagar Academy of Technology and Management, Bangalore, India[1]

Assistant Professor, Department of Electronics and Communication, Dayananda Sagar Academy of Technology and Management, Bangalore, India [2]

Abstract-Despite research highlighting vulnerabilities and shortcomings of the controller area network bus (CAN bus) protocol and proposed alternatives, it remains the industry standard and is used in most vehicles. Much work has gone into detecting intrusions on the CAN bus because of its susceptibility to potential intruders who could impede execution or even take control of the vehicles. The majority of literature, however, does not offer defenses or responses to attacks, making it impossible for the system to operate securely in the face of an intrusion. This letter suggests a low-overhead approach to automatically identify and categorize intrusions into pre-established groups when they are found. Our framework consists of three steps: 1) classifies messages pertaining to the same attack into blocks; 2) extracts pertinent features from each block; and 3) uses a lightweight classifier model to predict the type of attack. Within the first 50 milliseconds of the attack, the initial models shown in this letter demonstrate an accuracy of up to 99.16%, enabling the system to respond to the intrusion before the malicious actor can complete their attack. This letter, in our opinion, establishes the framework for vehicles to respond differently at runtime depending on the kind of attack.

Index Terms: embedded systems, controller area network, attack type classification.

I. INTRODUCTION

By nature, modern cars are distributed systems, with all of their functions divided among several electrical component units (ECUs) with distinct functions.Some vehicles require more than 70 ECUs to function properly because of this distributed design [11]. Most cars use the controller area network bus (CAN bus) to allow the numerous ECUs to exchange the necessary data with one another,despite being the industry standard since 1996 [3], malicious actors can still target the CAN bus. A malicious agent can read all shared messages and send messages that other ECUs will receive if it manages to get access to the CAN bus because messages are neither signed nor encrypted [2, 7, 9, 11]. Such assaults occur in our daily lives outside of research and commercial settings, as in [5].A great deal of work, including [2], [3], [7], [8], [9], and [12], has been proposed in the literature to detect intrusions in the CAN bus. Unfortunately, the majority of intrusion detection systems (IDSs), such as those found in [2], [7], [9], and [12], end the analysis at the detection stage, meaning that no analysis is conducted once an intruder has been found. As a result, there are no pathways provided to enable safe responses. These suggested frameworks prevent the messages from being processed at ECUs, but they do not examine whether an improved defense against the current attack exists. Safety-critical systems need to classify attack types in order to respond to current attacks and get

ready for new ones [6]. A significant amount of overhead is added to the system runtime by the few works that have addressed the problem of attack classification on the CAN bus [3], [8]. These works fail to consider the potential impact of high overheads on safe responses.

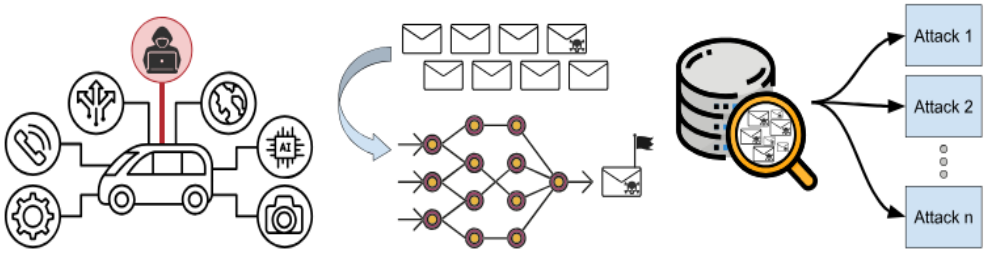
We present LOCoCAT, an attack-type classification system, in this letter that can quickly and efficiently determine the kind of attack the hacker is carrying out. Our solution adds very little power, memory, or processing overhead to the overall system and can be implemented on any modern intrusion detection system. Detected malicious messages are grouped into attack blocks, discrete features are calculated for each attack block, and a lightweight classifier model is used to determine the type of attack for each attack. This is how it operates. Three of the most widely used CAN bus datasets are used by us to assess the applicability of our proposal [2], [7], and [9]. We find that, in an environment with limited resources and energy, our approach produces excellent classification results with low latency.
This letter makes two main contributions.:
1) To minimize the number of inferences needed, we suggest combining several CAN bus messages into blocks that can be classified collectively.

2) Using several datasets from the literature, we compare our method's accuracy and overhead with related work when operating on a low-power platform.

II. BACKGROUND AND RELATED WORK

A. CAN Bus Attacks

The industry standard for communication between the various ECUs in cars is the CAN bus [11]. However, the ECUs connected to the CAN bus are susceptible to malicious agents gaining access to the bus because of its design limitations.As a result, a lot of work has gone into 1) developing CAN bus attack datasets and 2) developing IDSs for the CAN bus. SynCAN is a synthetic CAN bus dataset that was introduced by Hanselmann et al. [7] to act as a benchmark for CAN bus IDSs. SynCAN supports up to four signal values per message and has five distinct attack types, with a four-to eight-second time span for each attack. The survival-IDS dataset, which includes actual CAN bus data from three distinct cars—the Hyundai Sonata, Kia Soul, and Chevrolet Spark—was first presented by Han et al. [9]. With three distinct attack types and up to eight signal values per message, each Survival-IDS attack lasts five seconds. Car hacking was first presented by Seo et al. [2] using a dataset that included actual CAN bus data from a Hyundai YF Sonata. Car hacking has up to eight signal values per message and contains four different attack types, where each individual attack ranges from 3 to 5 s.



(a) (b) (c)

Fig. 1. Step-by-step process for classifying CAN bus attacks. (a) Attacker gains access to the CAN bus and sends malicious messages. (b) State-of-the-art IDS flags the malicious messages on the bus. (c) LOCoCAT analyzes the flagged messages and identifies the attack type.

In order to detect CAN bus attacks, Kukkala et al. [12] proposed LATTE, an intrusion detection system (IDS) based on a neural network and support vector machine combination that improves accuracy on average by 18.94% when compared to other prior IDSs for CAN bus data. Furthermore, Kukkala et al. present a thorough overhead analysis of LATTE on an NVIDIA Jetson TX2 platform, achieving a 193 µs latency. The Jetson TX2 is a potent embedded device with a GPU that is frequently utilized as a second ECU in the literature, demonstrating the usefulness of LATTE in an actual system.

B. CAN Bus Attack Types

An attacker has the ability to create malicious messages with various goals and consequences in addition to getting access to the bus. The attacker can spoof the drive gear and RPM gauge [2], cause the vehicle to malfunction [9], or even take it by opening the doors and starting it up so they can drive away [5] if they have domain knowledge about the vehicle. Moreover, the attacker can still have an impact on the car even if they lack domain expertise by, for example, flooding the CAN bus to launch a denial-of-service attack [2], [7], [9], or sending messages with random values to cause actuation to become hazy [2], [9]. Therefore, in order for the system to respond appropriately, it is imperative to detect CAN bus attacks and determine the attacker's intent [6]. The types of CAN bus attacks that have been identified in the literature thus far are few. As far as we are aware, the only publications that examine CAN bus attacks as a multiclass classification issue are [3] and [8].

In order to classify car-hacking attacks in multiclass labels, Amato et al. [3] proposed an intrusion detection system (IDS) that uses a neural network model, achieving a weighted F1-score of 96.8%. The platform on which the trials were conducted is not mentioned by the authors, nor is the overhead of their methodology examined.

In order to classify survival-IDS attacks in multiclass labels, Hossain et al. [8] proposed an IDS for survival-IDS that achieves a weighted F1-score of 98.84% using a neural network model. Their model makes use of a robust desktop PC with a GPU, but it doesn't offer any overhead analysis. Since [8] employs a strong platform and neither work offers a comprehensive overhead analysis, it is unclear if their methods would work in the setting of a real-time safety-critical system. We present a comprehensive overhead analysis and compare the accuracy results of our approach with those of [3] and [8] in Section IV.

III. METHODOLOGY

The entire procedure for classifying the types of attacks on the CAN bus is shown in Fig. 1. An attacker with access to the bus and the ability to send malicious messages is depicted in Fig. 1(a). The state-of-the-art IDS from the literature that can identify malicious messages among legitimate data in the CAN bus is then shown in Fig. 1(b). Because of its high accuracy and low latency, we use LATTE [12] as the framework's IDS as a proof of concept. However, as new models emerge in the literature, it would be simple to replace LATTE in the overall framework. Lastly, our contribution is displayed in Fig. 1(c), which is a framework that can categorize attack types of malicious message groups that the IDS has detected into previously recognized categories. Since the literature has extensively discussed Figs. 1(a) and (b) (refer to

Sect. II), the remainder of this section will concentrate on Fig. 1(c), which represents our primary contribution.

The primary goal of LOCoCAT (Low-Overhead Classification of CAN bus Attack Types) is to rapidly, precisely, and nonintrusively identify the type of attack that is currently underway so that the system can select the most appropriate countermeasure. Furthermore, LOCoCAT has no discernible impact on the total overhead added to the system because our method depends on an IDS to identify malicious messages, and the most accurate IDSs in the literature (e.g., [7] and [12]) employ sophisticated models that need a GPU-equipped board (e.g., NVIDIA Jetson TX2).
In order to accomplish both of these objectives, we investigate models that: 1) produce good accuracy results; 2) operate well on low-end hardware; 3) are lightweight; and 4) have a latency that is reasonable given the typical duration of CAN bus attacks.
Our suggested classification methodology examines messages that an existing IDS has identified as malicious. Initially, we create attack blocks out of malicious messages whose timestamps are close to one another. In order to obtain a classification result prior to the attack's termination, we secondly select a window at the beginning of the attack blocks for analysis. Third, regardless of the number of messages included in the attack, we compute discrete features for the attack block window, producing a finite set of features for each attack block. In order to determine the type of attack, we feed this discrete feature set into a classifier model at the end. This section goes into great detail about our methodology.

A. Attack Blocks and Features

For every CAN bus attack, the attacker sends a series of messages among the normal messages over a brief period of time. Since LOCoCAT already knows which messages are malicious because of the IDS, we group the malicious messages into blocks while the attack is happening. An attack block has a limited number of features. We compute three metrics (median, average, and standard deviation) for each signal in the messages, as well as the number of messages that comprise this attack block. We also tally the number of messages that comprise this attack block. With up to four signals per message in the SynCAN dataset [7], we have a total of 13 features for each attack block. Similarly, with up to eight signals per message in the Car-Hacking [2] and Survival-IDS [9] datasets, we have 25 features for each attack block.

B. Window Selection

Giving the system more information on how to respond optimally is one of the objectives of determining the type of attack. Therefore, obtaining a classification result prior to the attack's conclusion is essential.We chose to use no more than one second of the attack duration for classification because the duration of each CAN bus attack on the literature datasets [2], [7], and [9] ranges from three to eight seconds. This allows the system enough time to plan and carry out a response to the ongoing attack. As a result, we investigate models that take into account messages sent up to 50, 100, 200, 500, and 1000 milliseconds after the attack began.

C. Model Selection

We utilize Python 3 and the scikit-learn [4] library to create our framework, training all models and making all inferences. A simple method for training and exporting models suitable for CPU-only platforms is offered by Scikit-learn. We take into account the following multiclass classification models that are available in the library: Random Forest, k-Nearest Neighbors, Decision Tree, Gaussian Naive Bayes, Adaptive Boosting, and Quadratic Discriminant Analysis. For every model, we run a parameter grid search

to determine which outcomes, taking into account all possible window lengths, yield the best results. The best parameter combinations for each model are used to produce the results displayed in Section IV.

IV. EXPERIMENTAL RESULTS

Every experiment was run on a Raspberry Pi Zero W, which has 512 MB of RAM and a 1-GHz single-core ARMv6 CPU. The scripts and findings are all available online at https://github.com/cbdm/LOCoCAT. For all experiments, we use three datasets with malicious CAN bus data from the literature: 1) survival-IDS [9], 2) car hacking [2], and 3) SynCAN [7]. You can access all three of the datasets online.Every dataset is prepped for our experiments in the following ways: 1) all normal messages are filtered out; 2) malicious messages that are part of the same attack are created as attack blocks; 3) the attack blocks are shuffled; and 4) these blocks are divided into 80% for training and 20% for testing. Two subheadings that follow examine two distinct facets of our methodology. First, we examine how well our proposal performs in correctly identifying the active attack in Section IV-A. Next, we examine the models' additional overhead in Section IV-B.

A. Correct Classification

The weighted F1-scores for the various models and windows investigated for the Car-Hacking and SynCAN datasets are shown in Tables I and II, respectively. We decided not to include the full survival-IDS results because there weren't enough attack blocks for a comprehensive study. Car hacking (1200) and SynCAN (545) had significantly more distinct attacks than the Survival-IDS dataset, which had only 29 when the malicious messages from each dataset were grouped into attack blocks. The fact that a decision tree model with a 100 ms window achieves a weighted F1-score of 100% is still noteworthy, though. As Table I demonstrates, our approach achieves an F1-score of more than 99% using a Gaussian Naive Bayes model that only takes into account the first 50 ms of each attack block on the Car-Hacking dataset. These results hold great promise for the practicality of our approach in a real-world system, as we are able to achieve excellent classification results for a 4-class problem on a platform with limited resources and low power requirements. Despite the fact that the SynCAN dataset's scores in Table II are not as high as those for vehicle hacking, the findings are nonetheless encouraging because the SynCAN dataset includes one additional attack.

TABLE I
WEIGHTED F1-SCORE FOR CAR HACKING

Model	1000ms	500ms	200ms	100ms	50ms
AdaBoost	73.75%	73.75%	73.75%	73.75%	73.75%
Decision Tree	79.58%	83.33%	97.50%	98.75%	96.66%
Gaussian NB	84.58%	84.58%	97.08%	98.75%	99.16%
kNN	70.83%	76.25%	94.58%	97.08%	95.41%
QDA	20.41%	20.41%	20.41%	20.41%	20.41%
Random Forest	84.58%	84.16%	97.50%	97.91%	97.91%

TABLE II
WEIGHTED F1-SCORE FOR SYNCAN

Model	1000ms	500ms	200ms	100ms	50ms
AdaBoost	77.06%	75.22%	58.71%	60.55%	47.70%
Decision Tree	76.14%	77.98%	71.55%	58.71%	44.03%
Gaussian NB	63.30%	65.13%	59.63%	53.21%	45.87%
kNN	73.39%	68.80%	61.46%	55.04%	43.11%
QDA	41.28%	14.67%	27.52%	23.85%	19.26%
Random Forest	76.14%	77.06%	74.31%	57.79%	53.21%

TABLE III
AVERAGE INFERENCE LATENCY (MS)

Model	1000ms	500ms	200ms	100ms	50ms
AdaBoost	93.6	25.9	68.7	69.2	68.7
Decision Tree	3.3	3.3	3.3	3.3	3.3
Gaussian NB	9.2	9.2	9.2	9.3	9.3
kNN	15.6	15.5	15.4	15.5	15.6
QDA	15.5	15.5	15.5	15.5	15.5
Random Forest	75.7	52.1	40.4	33.4	59.5

B. Framework Overhead

We start by examining the inference latency because we want the system to have time to respond if the classification process ends before the attack does. For a total of 174 500 inferences, Table III displays the average latency in milliseconds for every model-window combination. The two most accurate models, Decision Tree and Gaussian Naive Bayes (see Section IV-A), had very short inference times, corresponding to 9.3 and 3.3 ms. This would account for a total classification time of 59.3–50 ms for message reception and 9.3 ms for inference for the Car-Hacking dataset, leaving at least 2.94 s for system reaction.

TABLE IV
AVERAGE TRAINED MODEL SIZE (KB)

Model	1000ms	500ms	200ms	100ms	50ms
AdaBoost	174.2	56.0	174.2	174.2	174.2
Decision Tree	42.4	5.8	3.4	4.3	13.2
Gaussian NB	1.9	1.9	1.9	1.9	1.9
kNN	1496.5	1496.5	1496.5	1496.5	1496.5
QDA	156.6	156.6	156.6	156.6	156.6
Random Forest	40.2	19.1	228.4	20.3	26.9

This would allow for a total classification time of 503.3–500 ms for message reception and 3.3 ms for inference for the SynCAN dataset, leaving at least 3.49 s for system response.This would allow for 103.3–100 ms of message reception and 3.3 s of inference for the Survival-IDS dataset, leaving at least 4.896 s for the system to respond. Under all circumstances, the system ought to have sufficient time to select and initiate the proper defense against the current kind of attack.

We then examine the necessary space. Table IV displays the average sizes in KB for every model under consideration. The most accurate models would only need 1.9, 4.3, and 5.8 kB, respectively: Gaussian Naive Bayes (50 ms) and Decision Tree (100 and 500 ms).
Compared to more intricate neural network models in the literature, such as LATTE, which requires 1439 kB, these sizes are significantly smaller [12]. Additionally, we can drastically cut down on the additional

power we add to the system by utilizing a low-power platform like the Raspberry Pi Zero W. When running at maximum CPU usage, a Raspberry Pi Zero W consumes 1.512 W [1], while an NVIDIA Jetson TX2 can use up to 15 W, almost ten times as much [10].

C. Comparison With Existing Literature

We trained models based on the descriptions of the models in each corresponding paper in order to compare our findings to earlier suggestions.
We use a 3-hidden-layer multilayer perceptron (MLP-3) with scikit-learn to classify attack blocks in order to replicate [3]. In order to replicate [8], we train a 512-unit single-layer long short-term memory (LSTM) model on the Pi Zero using TensorFlow Lite2 to classify malicious messages. Furthermore, as a baseline for comparison, we included a DummyClassifier that selects the attack type at random. The results we achieved using these three distinct methods are displayed in Table V alongside the highest-performing LOCoCAT models. The F1-score for each model for the corresponding dataset is displayed in the Car Hacking, Survival-IDS, and SynCAN columns. Latency shows the average model latency over all datasets, and Size shows the average model size over all datasets.
These findings demonstrate the superior performance of the LSTM, MLP-3, and LOCoCAT approaches in the Car-Hacking and Survival-IDS datasets. Nevertheless, LOCoCAT outperforms the other methods by more than 1.75× when it comes to classifying attacks in the SynCAN dataset. Furthermore, LOCoCAT and MLP-3 exhibit a significantly reduced latency in comparison to the LSTM model. Last but not least, LOCoCAT exhibits significant memory size improvements, 10× to MLP-3 and 1000× to LSTM. These findings demonstrate that it is possible to accurately perform CAN bus attack classification using LOCoCAT even when resource constraints are strictly enforced.

TABLE V
COMPARISON WITH EXISTING LITERATURE

Model	Car-Hacking	Survival-IDS	SynCAN	Latency	Size
Dummy	26.25%	33.33%	20.18%	8.75ms	0.5KB
LSTM	100%	99.26%	41.04%	35.21ms	4.284.4KB
MLP-3	98.75%	100%	44.03%	5.07ms	45.3KB
LOCoCAT	99.16%	100%	77.98%	5.30ms	4.0KB

V. CONCLUSION AND FUTURE WORK

We introduce LOCoCAT, a framework that can detect the kind of continuous attacks on the CAN bus with minimal overhead, in this letter. We compare the performance of LOCoCAT with related works by accurately classifying attacks using three existing datasets that have been used in the literature to develop IDSs for the CAN bus. Furthermore, the classification latency is reasonable given the attack lengths because it gives the system enough time to respond, and it uses a lot less memory and power than similar works.
In the future, we want to approach two paths. 1) Expand LOCoCAT to work with attack types that are either unknown to the system or are a mix of existing attacks. 2) Explore the feasibility of running our approach on a platform with even lower resource requirements, such as the Raspberry Pi Pico.

REFERENCES

[1]A. G. Millard et al., "The pi-puck extension board: A raspberry pi interface for the e-puck robot platform," in Proc. 2017 IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), 2017, pp. 741–748.

[2] E. Seo, H. M. Song, and H. K. Kim, "Gids: Gan based intrusion detection system for in-vehicle network," in Proc. 2018 16th Annu. Conf. Privacy Secur. Trust (PST), Aug. 2018, pp. 1–6.

[3] F. Amato, L. Coppolino, F. Mercaldo, F. Moscato, R. Nardone, and A. Santone, "Can-bus attack detection with deep learning," IEEE Trans. Intell. Transp. Syst., vol. 22, no. 8, pp. 5081–5090, Aug. 2021.

[4] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, Nov. 2011.

[5] D. Goodin. "There's a New Form of Keyless Car Theft That Works in Under 2 Minutes" Ars Technica. Apr. 2023. Accessed: Apr. 30, 2023. [Online]. Available: https://arstechnica.com/information-technology/2023/04/crooks-are-stealing-cars-using-previously-unknown-keylesscan-injection-attacks/

[6] H. A. Kholidy, "Autonomous mitigation of cyber risks in the cyber– physical systems," Future Gener. Comput. Syst., vol. 115, pp. 171–187, Feb. 2021.

[7] M. Hanselmann, T. Strauss, K. Dormann, and H. Ulmer, "Canet: An unsupervised intrusion detection system for high dimensional can bus data," IEEE Access, vol. 8, pp. 58194–58205, 2020.

[8] M. D. Hossain, H. Inoue, H. Ochiai, D. Fall, and Y. Kadobayashi, "LSTM-based intrusion detection system for in-vehicle can bus communications," IEEE Access, vol. 8, pp. 185489–185502, 2020.

[9] M. L. Han, B.-I. Kwak, and H. K. Kim, "Anomaly intrusion detection method for vehicular networks based on survival analysis," Veh. Commun., vol. 14, pp. 52–63, Oct. 2018.

[10] NVIDIA. Jetson TX2 Module. (2021). NVIDIA Developer. Accessed: Apr. 30, 2023. [Online]. Available: https://developer.nvidia.com/ embedded/jetson-tx2

[11] R. Hegde, G. Mishra, and K. S. Gurumurthy, " An insight into the hardware and software complexity of ECUs in vehicles," in Proc. Commun. Comput. Inf. Sci., 2011, pp. 99–106.

[12] V. K. Kukkala, S. V. Thiruloga, and S. Pasricha, "LATTE: LSTM selfattention based anomaly detection in embedded automotive platforms," ACM Trans. Embed. Comput. Syst., vol. 20, no. 5s, pp. 1–23, 2021.