

USENIX

THE ADVANCED COMPUTING
SYSTEMS ASSOCIATION

WIP: On the Real-World Applicability of Automotive CAN Intrusion Detection Systems

Kristijan Lazeski, *Darmstadt University of Applied Sciences and European University of Technology;* Michael Zohner, *University of Applied Sciences Fulda;* Jannis Hamborg, *Darmstadt University of Applied Sciences and European University of Technology;* Christoph Krauß and Alexander Wiesmaier, *Darmstadt University of Applied Sciences, European University of Technology, and National Research Center for Applied Cybersecurity ATHENE*

https://www.usenix.org/conference/vehiclesec26/presentation/lazeski

This paper is included in the Proceedings of the
4th USENIX Symposium on Vehicle Security and Privacy.

August 10–11, 2026 • Baltimore, MD, USA

ISBN 978-1-939133-56-4

Open access to the Proceedings of the
4th USENIX Symposium on Vehicle Security and Privacy
is sponsored by USENIX.

WIP: On the Real-World Applicability of Automotive CAN Intrusion Detection Systems

Kristijan Lazeski[1,2], Michael Zohner[4], Jannis Hamborg[1,2], Christoph Krauß[1,2,3], and Alexander Wiesmaier[1,2,3]

[1]Darmstadt University of Applied Sciences, Germany
[2]European University of Technology, European Union
[3]National Research Center for Applied Cybersecurity ATHENE, Germany
[4]University of Applied Sciences Fulda, Germany

Abstract

Machine Learning (ML)-based Controller Area Network (CAN) Intrusion Detection Systems (IDS) are widely studied and used to detect and mitigate attacks on in-vehicle communication. Despite frequent reports on high detection performance for well-established datasets, the practical applicability and robustness against real-world attackers of the trained solutions is rarely discussed. This work presents a preliminary analysis of the practical robustness of ML-based CAN IDS solutions, with a focus on per-frame, binary intrusion detection approaches. Our approach consists of three steps: (1) We analyze common datasets used for training and evaluating ML-based CAN IDS, (2) train human-interpretable decision trees on the datasets and investigate their learned classification rules[1], and (3) critically evaluate the models against a practical attacker model. We observe that many commonly used CAN IDS datasets only include simple and easily distinguishable attacks. Consequently, ML-based models trained on these datasets tend to learn dataset-specific statistical artifacts or shortcuts rather than robust or semantically meaningful detection logic, a finding we demonstrate specifically within the scope of per-frame, binary detection. While our results are preliminary and limited to this detection paradigm, they show that high detection performance is not inherently indicative of practical robustness. We therefore argue that new detection strategies founded upon vehicle specifications and domain knowledge, in combination with ML tools, may offer more transparent and reliable protection than purely ML-driven approaches, and we call for broader evaluation across more diverse detection settings in future work.

Intrusion Detection System, Automotive, CAN, Machine Learning, Datasets

1 Introduction

The Controller Area Network (CAN) bus is the central technology for in-vehicle communication between Electronic Control Units (ECUs), but was designed without encryption or authentication, making it a well-known attack vector [5, 19, 32]. Numerous machine Learning (ML)-based CAN Intrusion Detection Systems (IDSs) have been proposed to detect attacks such as Denial of Service (DoS), fuzzing, and spoofing [26], with many reporting precision and recall near or at 100% [6, 10, 11, 15–17, 20–22, 24, 36, 37, 41, 46]. However, it remains unclear whether these results reflect models learning robust behavioral invariants or merely exploiting dataset-specific artifacts.

In this paper, we present an initial study aimed at answering this question. We systematically analyze five widely used CAN IDS datasets [10, 16, 21, 22, 41], examining their structural properties and the classification rules learned by interpretable models to assess whether observed separability reflects genuine attack characteristics or dataset-induced biases. Our analysis follows a two-step approach. First, we examine the distinguishing characteristics between benign and malicious traffic, where a *trace* is a recorded CAN message with a timestamp and a benign or malicious label. Second, we train a Decision Tree (DT) model per dataset and analyze its learned classification rules. DTs are chosen for their transparency, enabling direct inspection of which features drive classification [12]. We note that this study is scoped to per-frame, binary intrusion detection, and our findings should be understood within that limitation. Our preliminary findings indicate that many attacks in the datasets analyzed can be distinguished from benign traffic based on features unique to the dataset-specific attack implementation rather than inherent properties of malicious CAN messages. Although our DT models often achieve a precision and recall of over 99%, an analysis of the learned rules shows that they are based on simple features that are easy to circumvent. An attacker who slightly adapts timing or payload values can evade such an IDS without fundamentally changing the attack strategy. These observations raise broader questions about current evaluation methodology that warrant further investigation. In practice, manufacturers often deploy rule-based IDS that explicitly encode domain knowledge such as expected message frequencies, signal ranges, and

[1]https://tudatalib.ulb.tu-darmstadt.de/handle/tudatalib/5167

ECU communication patterns. ML-based IDS, by contrast, are evaluated on publicly available datasets using standard metrics. If these datasets contain artifacts or unrealistic attack characteristics, models may learn shortcuts rather than robust security-relevant properties and reported high performance may not translate into real-world protection. This preliminary study has three main contributions: (1) a systematic analysis of popular CAN IDS datasets to assess the separability of malicious from benign traffic, (2) an investigation into features learned by ML-based CAN IDS using interpretable DT models, and (3) a demonstration that many learned rules are fragile and easily bypassed under realistic attacker models. From these findings, we derive implications for dataset design and outline challenges for more robust evaluation and deployment of ML-based CAN IDS.

The remainder if this paper is structured as follows: Section 2 presents background and related work (including a typical attacker model assumed in our study). Our study is described in Section 3. Finally, we conclude the paper and discuss future work in Section 4.

2 Background and Related Work

2.1 Controller Area Network (CAN)

Introduced in the 1980s [4], the CAN bus remains a core technology for in-vehicle communication, enabling deterministic, fault-tolerant, real-time data exchange between ECUs at low cost. While newer protocols such as Automotive Ethernet [33] are gaining ground for high-bandwidth applications, CAN continues to dominate safety-critical domains such as braking systems [28]. This work focuses exclusively on classical CAN, acknowledging but excluding CAN FD and other protocol variants.

2.1.1 CAN Frame Structure

CAN encapsulates data in frames, most commonly the *Base Frame Format* (11-bit identifier) or the *Extended Frame Format* (29-bit identifier), consisting of the following fields (cf. Figure 1): Start of Frame (SOF), CAN ID / arbitration field, Remote Transmission Request (RTR) bit, Control field (Identifier Extension (IDE) bit and Data Length Code (DLC)), data payload (0–8 bytes), CRC, Acknowledgment field, and End of Frame (EOF) [4]. CAN uses a multi-master architecture in which bus conflicts are resolved via *non-destructive bitwise arbitration*: each transmitting node continuously monitors the bus and relinquishes transmission if it detects a dominant bit (logical '0') while asserting a recessive bit (logical '1'), thereby granting bus access to the frame carrying the lowest CAN ID without loss of data or time.

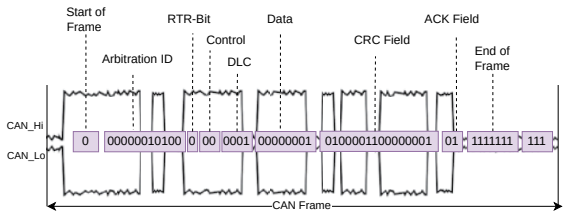


Figure 1: The structure of a CAN frame.

2.1.2 CAN Data Interpretation and Frame Timing

The 8-byte CAN payload is interpreted through signals defined in a Database CAN (DBC) file [7, 8], which is authored by the vehicle manufacturer and distributed to subcontractors for ECU configuration. Each signal definition specifies its bit position and length, endianness, a linear scaling function (scale and offset), a valid value range, and an optional unit. Figure 2 shows an exemplary DBC entry from the OpenDBC project[2]: the steering angle is encoded in bits 0–15 (scaled by 0.0428316886, Most Significant Bit (MSB) as sign, valid range −600 to 600), a counter in bits 20–23, and a complement value in bits 24–34, illustrating that only a subset of all bit-encodable values carries semantic meaning.

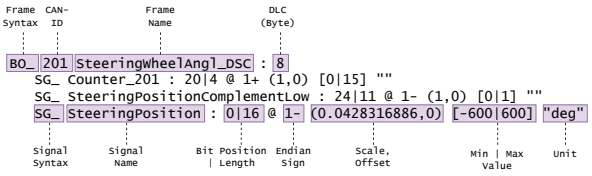


Figure 2: Example DBC entry for a BMW E9x/E8x.

Beyond payload structure, CAN frame timing follows three distinct patterns [31, 47]: safety-critical frames are transmitted periodically with tight, well-defined intervals and minimal jitter [2, 47]; non-critical synchronization frames are periodic but tolerate higher jitter [2]; and event-driven frames (e.g., door events, diagnostics) are transmitted only upon specific triggers and may be absent during normal operation [31, 43]. Both signal definitions and timing specifications are proprietary and not publicly disclosed, making it necessary for academic CAN IDS research to reverse-engineer this information from observed traffic or available datasets [10, 16, 21, 22, 41].

2.2 CAN bus Attacks

Attacker Model. Following Hamborg et al. [9], we distinguish two attacker classes. A *standard-compliant attacker* operates within the CAN protocol specification and, after gaining electrical bus access (e.g., via OBD-II or a compromised ECU), can passively observe traffic, inject crafted

[2] https://github.com/commaai/opendbc

frames, replay recorded messages, and generate DoS conditions, but cannot modify frames already in transmission. A *physical-layer attacker* additionally manipulates the electrical signaling of the bus, enabling suppression or corruption of legitimate frames beyond protocol-compliant injection, for instance by forcing dominant states or pushing ECUs into error-passive or bus-off conditions. In this work, we assume an adversary with physical bus access who can observe, inject, and replay CAN frames and adapt injected messages based on observed CAN IDs, payload ranges, and timing behavior, without requiring cryptographic keys or proprietary specifications.

Attack Types. The assumed attacker model enables several well-established attack classes [39]. **DoS** attacks exploit CAN's priority-based arbitration by flooding the bus with high-priority frames, starving lower-priority communication. **Fuzzing** injects random or malformed frames to trigger unintended ECU behavior or expose implementation weaknesses. **Spoofing** exploits the absence of sender authentication to impersonate legitimate ECUs by injecting fabricated frames with valid CAN IDs, enabling targeted manipulation of vehicle functions. **Replay attacks** retransmit previously recorded valid frames to reproduce specific actions such as door unlocking or component activation. Attackers with additional physical-layer capabilities can further perform selective frame suppression, arbitration manipulation, and fault injection, allowing not only malicious traffic injection but also targeted shaping or suppression of legitimate communication.

2.3 Machine Learning

ML has become the dominant approach for intrusion detection due to its ability to learn behavioral characteristics directly from data, where traditional rule-based systems fail against unseen attack patterns [25,45]. Detection performance is strongly dependent on dataset quality and representativeness [40], and high numerical results alone do not guarantee robustness. Explainable AI (XAI) and interpretable models are therefore critical for bridging automated detection and human understanding [30].

Decision Trees (DTs). We employ DTs [44] as interpretable proxies to examine which features drive classification decisions. Following Quinlan [34], DTs recursively partition data by maximizing information gain via Shannon entropy until a stopping criterion is met. Crucially, dataset artifacts or trivially separable features that dominate DT decisions are likely to be exploited equally, or more aggressively, by opaque models such as random forests or neural networks, which may achieve even higher numerical performance without learning more semantically meaningful or robust rules.

Evaluation Metrics. We address binary classification between benign (negative) and malicious (positive) traffic, with True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) defined accordingly. Since the analyzed datasets are highly class-imbalanced, with malicious traces often constituting only a small fraction of all data, accuracy alone is insufficient, as a trivial majority-class predictor would appear performant. We therefore evaluate using False Positive Rate (FPR), False Negative Rate (FNR), precision, recall, and F1-score [3, 27, 29].

2.4 Related Work

Several surveys structure the CAN IDS landscape: Karopoulos et al. [18] and Rajapaksha et al. [35] both highlight inconsistencies in evaluation practices and dataset realism, while Sharmin et al. [38] emphasize the lack of standardized benchmarking protocols. Stabili et al. [42] and Abgaje et al. [1] provide structured evaluations of detection methods, with the latter reporting strong ML performance on the Car Hacking dataset [41], yet neither investigates whether this performance stems from robust behavioral modeling or dataset-specific artifacts. Jacobs et al. [12] raise analogous concerns for network-based IDS, arguing that many reported results rely on unrealistic assumptions; we transfer this critical perspective directly to the CAN domain. On dataset quality, Lee et al. [23] conclude that no existing dataset fully satisfies representativeness and diversity requirements, and Lampe et al. [21] propose improved construction methodologies, yet neither examines how dataset properties shape learned classification rules. The Car Hacking Competition [16] further corroborates our findings: the best-performing final-round solution was rule-based, suggesting that semantic signal understanding may outweigh statistical learning.

In contrast to prior work, we neither propose a new detection method nor a new dataset. Instead, we systematically analyze five widely used CAN IDS datasets through the lens of interpretable ML, examining what decision tree models actually learn and evaluating learned rules under a realistic attacker model, a gap that performance-focused surveys and benchmarks have not addressed.

3 Analysis of Datasets for ML-Based CAN IDS

We assess the practical robustness of ML-based CAN IDS trained on commonly used datasets, analyzing dataset properties (Section 3.2), empirically verifying findings via DTs (Section 3.3), evaluating robustness (Section 3.4), and discussing implications (Section 3.5). Our goal is not to propose a novel IDS, but to understand what drives near-perfect reported performance and whether learned rules remain meaningful under the attacker model described in Section 2.2.

3.1 Methodology and Limitations

Our methodology proceeds in two steps. First, we analyze widely used CAN IDS datasets (cf. Section 3.2) with respect to the assumed attacker model (cf. Section 2.2), examining trace organization, recorded fields, attack strategies, class distribution, and CAN-ID–payload overlap between benign and malicious traffic. Second, DTs are trained on these datasets (cf. Section 3.3) and their learned classification logic is inspected to determine whether decision rules reflect robust behavioral constraints or dataset-specific regularities.

Limitations.

1. **Decision Trees Only.** We restrict our empirical evaluation to DTs, which provide native interpretability and directly expose structural shortcuts in the data. While tools such as Trustee [12] can convert complex models into interpretable surrogates, tree-based models achieve performance comparable to deep learning in CAN IDS settings [16]. Since all supervised models trained on the same data share the same feature distributions and separability properties, DT analysis generalizes qualitatively, though subtle differences in learned boundaries across model types cannot be excluded.

2. **Binary Classification.** Models are trained separately per attack type to isolate decision criteria and prevent dominant attack patterns from masking subtler ones. Mixed-attack training can obscure whether performance is driven by robust behavioral properties or a single easily separable class; single-attack models serve as a controlled baseline.

3. **Scope of Detection Paradigm.** This study is scoped exclusively to per-frame, binary intrusion detection, in which individual CAN frames are classified as either benign or malicious.

3.2 Dataset-Level Structural Analysis

We analyze five publicly available CAN IDS datasets, Car Hacking [41], OTIDS [22], Survival [10], CT&T [21], and Car Hacking Attack & Defense [16], covering DoS, fuzzing, spoofing, and replay attacks. We focus on whether malicious messages share CAN IDs and payload patterns with benign traffic, since attack messages whose CAN ID and payload combinations never occur in benign traffic are trivially easy to classify. Benign traffic is concentrated on a fixed set of CAN IDs with characteristic payload distributions, reflecting regular ECU communication (cf. Figure 3). Dataset properties are summarized in Appendix Table 3, and empirical results are reported in Table 1.

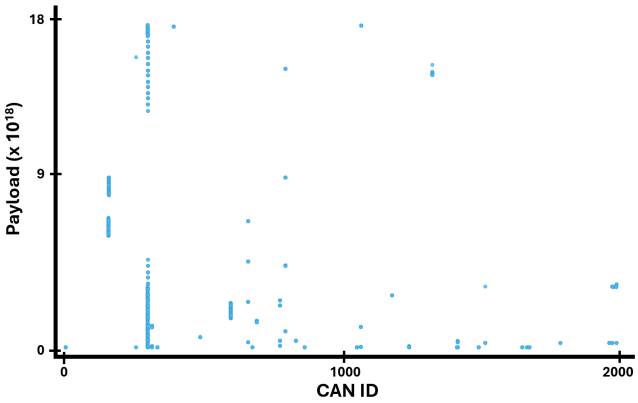
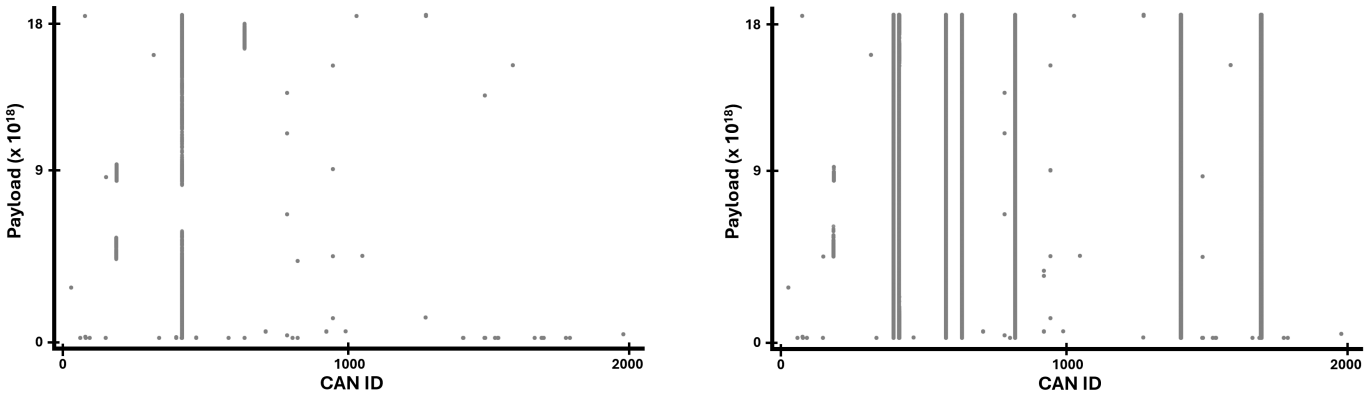


Figure 3: Distribution of benign traces for Car Hacking dataset on different CAN IDs (x-axis) and payload (y-axis).

DoS Attacks. Across all datasets, DoS attacks use a small number of static, uniquely identifiable combinations of CAN ID and payload that do not occur in normal traffic, making them trivially easy to detect. Car Hacking [41] (3 078 250/587 521), Survival [10] (148 760/33 141), OTIDS [22] (320 957/335 623), and Car Hacking Challenge [16] (3 207 280/171 824) all use CAN ID `0x000` with an all-zero payload, which is absent from benign traffic. CT&T [21] (2 246 099/51 798) deviates slightly, using CAN IDs `0x000` and `0x002` with three distinct constant payload variants, likewise absent from benign traffic. In all cases, detecting DoS attacks reduces to identifying a fixed injection pattern. Under the attacker model, however, an adversary can preserve the DoS effect while selecting a different high-priority CAN ID or varying the payload, thereby bypassing any detector that relies on static signatures.

Spoofing Attacks. Spoofing attacks are also implemented as fixed patterns consisting of CAN IDs and payload data that are not present in normal traffic, meaning that classification relies more on memorization than on the detection of semantic inconsistencies. In Car Hacking [41], gear spoofing (3 845 890/597 252) uses CAN ID `1087` with a static payload, and RPM spoofing (3 966 805/654 897) injects a single fixed payload for the targeted CAN ID. In Survival [10] (166 035/7 401), the malfunction attack targets CAN ID `0x153` with payloads outside the benign range. OTIDS [22] modifies RTR frames for CAN ID `0164`: benign requests carry no payload (DLC=0), injected frames carry 8 bytes (DLC=8). In CT&T [21], speed spoofing targets CAN ID `212` with three fixed payload values encoding 0,mph, 10,mph, and 60,mph, none of which appear in the benign traces for that CAN ID; script-based attacks (interval: `0x0D4`/`3501340134013501`; standstill: `0x161`/`2823FFFF002000F0`) and triple-spoofing follow the same static pattern structure. The Car Hacking Challenge [16] uses 2 patterns in training (1 648 114/3 877), 3



(a) Distribution of traces in the benign file.

(b) Distribution of traces in the fuzzing file.

Figure 4: Trace distributions in the benign and fuzzing files of OTIDS [22] (CAN ID on x-axis, payload on y-axis). Dots are grey as frames are unlabeled and thus indistinguishable.

in submission (1 559 166/20 093), and 4 in the final round (1 090 312/149 999), with one final-round pattern overlapping benign traffic. Under the attacker model, an adversary can observe valid payload ranges and inject within-range values, bypassing detectors that rely on fixed static signatures.

Fuzzing Attacks. Fuzzing attacks inject random CAN IDs and payloads, deliberately producing CAN ID and payload combinations that also occur in normal traffic, which forces models to rely on secondary artifacts such as recording structure or temporal boundaries rather than the content of individual frames. In Car Hacking [41] (3 347 013/491 847), eleven benign CAN IDs appear exclusively in the fuzzing file, likely triggered as ECU responses to injected frames. The benign-only file therefore does not cover all observable CAN IDs, risking misclassification of legitimate condition-triggered traffic at deployment. Survival [10] (210 178/39 812) follows the same principle without this discrepancy. OTIDS [22] targets nine CAN IDs (`153`, `164`, `1F1`, `220`, `2C0`, `4B0`, `4B1`, `5A0`, `5A2`) also present in benign traffic (cf. Figure 4). For IDs with near-constant benign payloads (e.g., `5A0`) fuzzed frames are visually distinguishable, while for variable IDs (e.g., `164`) they are not. Without frame-level labels, OTIDS does not support principled per-frame fuzzing detection. CT&T [21] provides standard fuzzing (2 413 372/37 391) with random CAN IDs and payloads, and systematic fuzzing (3 374 832/23 814) whose temporally structured progression introduces strong time-locality that enables models to exploit absolute timestamp boundaries. The Car Hacking Challenge [16] injects all possible payloads only for CAN IDs present in benign traffic (training: 1 648 112/44 405, submission: 1 559 163/51 923, final: 1 090 311/29 999).

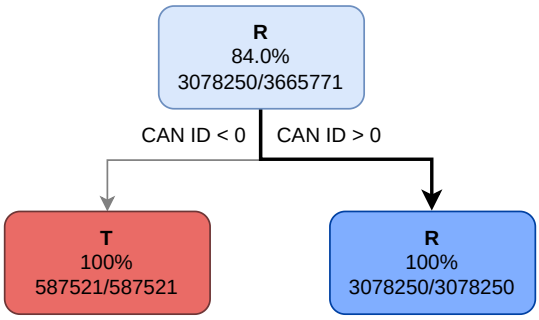
Replay. Replay attacks appear only in the Car Hacking Challenge [16]. Malicious traces consist of replayed benign messages. They fall within normal CAN-ID–payload value ranges and are timestamped consistently with the overall communication flow, making them difficult to distinguish. The training file contains 1 468 112/23 818 and the submission file (the only test split) 1 559 163/25 012 benign/malicious traces.

Dataset-Level Structural Anomalies and Artifacts. Several irregularities warrant attention. In Car Hacking [41], eleven benign CAN IDs appear only in the fuzzing file, meaning the benign recording does not fully represent observable CAN traffic. OTIDS [22] lacks frame-level labels in attack files, preventing reliable per-frame supervised learning. Several datasets exhibit strong class imbalance — DoS in CT&T constitutes 2% of traces, in Survival 4% — which can inflate accuracy metrics by rewarding majority-class predictions. Finally, attack scenarios in several datasets form contiguous time blocks with absolute timestamps, making temporal boundaries directly visible and constituting recording artifacts unrelated to vehicle behavior.

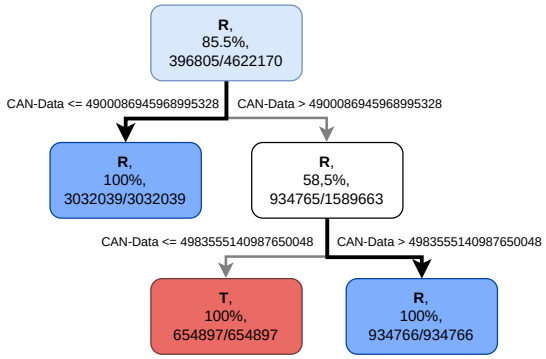
3.3 Decision Tree Training and Evaluation

We train DTs on all datasets except OTIDS, whose attack files lack frame-level labels. Prior work [13] incorrectly labeled all frames in these files as malicious, yielding no meaningful insight. We extend all datasets with interval features (time since the CAN ID last appeared) and represent payloads both as a single integer and as individual bytes. DTs are trained in Orange [44] with at least two instances per leaf, no splits on fewer than three traces, and a 99% stopping criterion (raised to 100% for strongly imbalanced subsets or fuzzing attacks). Datasets with dedicated test sets (Car Hacking Challenge) are evaluated on provided test files; others use 10-fold random sampling (66%/34%, averaged). Results are in Table 1.

Overall, DT models achieve F1-scores close to or equal to 100% across nearly all datasets and attack types. Lower performance arises where benign and malicious traces over-



(a) DT for the DoS attack, classifying on a static CAN ID unique to malicious traces.



(b) DT for the RPM spoofing attack, classifying on a static payload value unique to malicious traces.

Figure 5: DTs trained on DoS and RPM spoofing in the Car Hacking dataset (blue: benign, red: malicious). Arrow labels indicate decision criteria; box percentages indicate the class ratio.

lap in CAN-ID–payload space or where test files contain spoofing patterns unseen during training. High performance consistently coincides with static, uniquely identifiable attack patterns that would not survive adversarial adaptation.

DoS and Spoofing. Models for DoS and single spoofing attacks achieve 100% across all metrics for nearly every dataset, directly exploiting the unique static patterns identified in the data analysis (cf. Figure 5a). DoS classification relies solely on CAN ID `0x000`, and RPM spoofing (cf. Figure 5b) on a static payload not present in benign traffic. The exception is the Car Hacking Challenge: as the submission and final-round files contain different spoofing patterns than the training file, the model fails to detect any attack, highlighting the dependence on observing the specific pattern during training.

Fuzzing. Fuzzing models achieve F1-scores between 93.7% (CT&T) and 99.8% (Car Hacking and Survival), with tree sizes from 25 nodes (99%-split, Survival) to 389 nodes (100%-split, Car Hacking). Across all DTs, the interval feature is used first, followed by byte-level payload patterns and, less frequently, the CAN ID. Some nodes classify on absolute UNIX timestamps, which we attribute to overfitting on temporal recording boundaries rather than learned behavioral invariants.

Replay. Replay detection yields the poorest performance, ranging from 54.7% (100%-split, submission file) to 76.4% F1-score (100%-split, training file), with the largest models overall (3 809 nodes for 99%-split, 10 707 for 100%-split). This reflects overfitting: no general distinguishing rule can be learned, and unlike fuzzing where the timestamp appears only near the leaves, here it is already used at the second level of the tree.

3.4 Assessment Under the Attacker Model

We reassess the identified dataset characteristics against the attacker model in Section 2.2, examining whether learned decision rules remain robust when an adversary adapts CAN IDs, payload values, or timing behavior.

DoS Attacks. In Car Hacking [41], Survival [10], OTIDS [22], and the Car Hacking Challenge [16], DoS detection reduces to a binary check on the CAN ID `0x000`–all-zero-payload pair. Under the attacker model, an adversary can inject high-frequency frames using a different high-priority CAN ID or a modified payload while preserving the DoS effect, bypassing any detector tied to this static pattern. CT&T's slightly more complex instantiation (CAN IDs `000`/`002` with constant payload variants) likewise falls to an attacker using alternative CAN IDs, since the model only learns the specific observed combinations.

Static Spoofing Attacks. In all evaluated spoofing scenarios, learned rules memorize fixed CAN-ID–payload combinations or value ranges: RPM and gear spoofing in Car Hacking [41] use fixed payloads absent from benign traffic; the malfunction attack in Survival [10] targets payloads outside the benign range; speed spoofing in CT&T [21] uses three static payload values (0 mph, 10 mph, 60 mph); and the Car Hacking Challenge [16] uses 9 distinct static traces with varying patterns across files. Under the attacker model, an adversary can observe valid payload ranges and inject within-range values while still influencing vehicle behavior, rendering all static learned rules ineffective.

Fuzzing and Timestamp Dependence. For fuzzing in Car Hacking [41], Survival [10], CT&T [21], and the Car Hack-

ing Challenge [16], benign and malicious traces overlap in CAN-ID–payload space, so DTs frequently resort to absolute timestamp splits. These effectively encode dataset-specific temporal boundaries (e.g., a specific date in the Car Hacking dataset) rather than behavioral properties. Under the attacker model, an adversary can adapt injection timing to invalidate such splits, causing the IDS to misclassify adapted attacks or mislabel benign traffic recorded outside the original temporal window.

Replay Attacks. Replay attacks re-inject previously observed legitimate frames, meaning replay traffic can be indistinguishable in CAN-ID–payload space from benign operation. ML-based IDS models consequently overfit on timestamps, yielding very poor generalization (cf. Table 1). Under the attacker model, an adversary can replay a carefully selected frame subset and align timing to the current bus schedule, bypassing any detector that relies on recording-specific artifacts rather than semantic consistency.

OTIDS Label Ambiguity. OTIDS [22] attack files mix benign and malicious frames without frame-level labels. Prior work [13] labels entire attack files as malicious, causing models to learn file-level differences rather than frame-level attack characteristics, with timestamps serving as the primary discriminating feature. Under the attacker model, an adversary injecting frames during normal operation would not reproduce file-level structural differences, rendering such detection meaningless.

Training Coverage and Behavioral Completeness. In Car Hacking [41], additional benign CAN IDs appear only in attack files due to ECU responses triggered by injected frames, a behavior typical of event-based signals. If training does not capture the full range of legitimate signals, an IDS may misclassify valid but previously unseen communication. Under the attacker model, an adversary may deliberately trigger such rare legitimate signals to increase false positives or mask malicious activity. Overall, when evaluated against the assumed attacker model, the majority of learned decision rules reduce to recognizing static, dataset-specific artifacts that no adaptive adversary would reproduce.

3.5 Insights and Limitations

The combined analysis reveals three central limitations. First, many datasets implement attacks using static, uniquely identifiable CAN-ID–payload combinations, reducing detection to recognizing dataset-specific constants rather than genuinely malicious behavior. Second, incomplete benign coverage leaves models vulnerable to misclassifying rare but valid signals. Third, file-level separation, timestamp discontinuities, and labeling ambiguities introduce structural artifacts that models exploit as shortcuts, properties no adaptive attacker would reproduce in live vehicle communication.

Although DT-based models achieve between 90% and 100% F1-score across nearly all datasets, inspection of their learned decision logic reveals that this performance largely stems from structural regularities that do not persist under realistic adversarial adaptation. Against an attacker capable of adjusting CAN ID selection, payload values, and timing, many learned rules lose their security relevance, rendering the IDS ineffective against real-world attacks.

4 Conclusion & Future Work

In this paper, we show our analysis of ML-based CAN IDS through dataset structure and model interpretability. Using DTs, we show that widely used benchmark datasets contain systematic artifacts in CAN ID selection, payload values, and timing, allowing models to achieve near-perfect detection by learning dataset-specific regularities rather than robust attack signatures. More fundamentally, training ML models directly on raw CAN traffic is inherently limited: models tend to reverse-engineer static ID and payload ranges already specified in the manufacturer's DBC file, a task more efficiently handled by rule-based pre-filtering. A more effective architecture would apply such pre-filtering first, then train ML models on the residual traffic to target genuinely anomalous behavioral patterns, an approach that also scales better to real-world multi-bus, multi-gateway in-vehicle networks. High benchmark performance therefore does not imply practical security; the bottleneck is dataset realism, not model choice.

Future Work. Future work should establish clear criteria for security-relevant automotive IDS datasets: realistic attacker capabilities and injection strategies, proper frame-level labeling, representative benign coverage across operating conditions, avoidance of structural artifacts such as file-level or timestamp-based separability, and multi-bus topologies reflecting real in-vehicle networks. Only once such criteria are in place can datasets meaningfully support robustness evaluation. Extending this analysis to additional datasets, such as X-CANIDS [14] which may better reflect realistic attack characteristics, represents a natural next step and window based IDS approaches. More broadly, ML-based IDS becomes most valuable in architecturally realistic settings, e.g., multi-bus environments, gateway-level monitoring, cross-domain anomaly correlation, where complex interdependencies are difficult to express manually, requiring a shift from isolated single-bus trace classification toward system-level security modeling.

Acknowledgments

This work was supported by the German Federal Ministry of Research, Technology and Space (BMFTR), under the project

PARFAIT (Grant:16KIS2143).

References

[1] Paul Agbaje, Afia Anjum, Arkajyoti Mitra, Gedare Bloom, and Habeeb Olufowobi. A Framework for Consistent and Repeatable Controller Area Network IDS Evaluation. In *Proceedings Fourth International Workshop on Automotive and Autonomous Vehicle Security*, San Diego, CA, USA, 2022. Internet Society.

[2] Zixiang Bi, Guoai Xu, Guosheng Xu, Miaoqing Tian, Ruobing Jiang, and Sutao Zhang. Intrusion detection method for in-vehicle can bus based on message and time transfer matrix. *Security and Communication Networks*, 2022(1):2554280, 2022.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, USA, 2006.

[4] R. Bosch. Controller area network (can): An in-vehicle serial communication protocol. In *Proceedings of the SAE International Congress and Exposition*, number SAE Technical Paper 860391, Detroit, Michigan, USA, 1986. SAE International.

[5] Mehmet Bozdal, Mohammad Samie, and Ian Jennions. A survey on can bus protocol: Attacks, challenges, and potential solutions. In *2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE)*, pages 201–205, 2018.

[6] G. Dupont et al. Automotive controller area network (can) bus intrusion dataset v2, 2019.

[7] Daniel Frassinelli, Sohyeon Park, and Stefan Nürnberger. I know where you parked last summer: Automated reverse engineering and privacy analysis of modern cars. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1401–1415. IEEE, 2020.

[8] Robert Bosch GmbH. Can specification. 1991.

[9] Jannis Hamborg, Timm Lauser, and Christoph Krauß. RollCAN – CAN-bus based RollJam-Attack. CSCS '25, New York, NY, USA, 2025. Association for Computing Machinery.

[10] Mee Lan Han, Byung Il Kwak, and Huy Kang Kim. Anomaly intrusion detection method for vehicular networks based on survival analysis. *Vehicular communications*, 14:52–63, 2018.

[11] Markus Hanselmann, Thilo Strauss, Katharina Dormann, and Holger Ulmer. Canet: An unsupervised intrusion detection system for high dimensional can bus data. *Ieee Access*, 8:58194–58205, 2020.

[12] Arthur S. Jacobs, Roman Beltiukov, Walter Willinger, Ronaldo A. Ferreira, Arpit Gupta, and Lisandro Z. Granville. AI/ML for Network Security: The Emperor has no Clothes. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, pages 1537–1551, New York, NY, USA, 2022. Association for Computing Machinery.

[13] Abdul Rehman Javed, Saif Ur Rehman, Mohib Ullah Khan, Mamoun Alazab, and Thippa Reddy G. CANintelliIDS: Detecting In-Vehicle Intrusion Attacks on a Controller Area Network Using CNN and Attention-Based GRU. *IEEE Transactions on Network Science and Engineering*, 8(2):1456–1466, April 2021.

[14] Seonghoon Jeong, Sangho Lee, Hwejae Lee, and Huy Kang Kim. X-canids: Signal-aware explainable intrusion detection system for controller area network-based in-vehicle network. *IEEE Transactions on Vehicular Technology*, 73(3):3230–3246, 2024.

[15] Christian Kaiser, Alexander Stocker, and Andreas Festl. Automotive can bus data: An example dataset from the aegis big data project. *OpenAIRE*, 2019.

[16] Hyunjae Kang, Byung Il Kwak, Young Hun Lee, Haneol Lee, Hwejae Lee, and Huy Kang Kim. Car hacking: Attack & defense challenge 2020 dataset. *Third International Workshop on Automotive and Autonomous Vehicle Security*, 2021.

[17] Min-Joo Kang and Je-Won Kang. Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security. *PLOS ONE*, 11(6):e0155781, June 2016.

[18] Georgios Karopoulos, Georgios Kambourakis, Efstratios Chatzoglou, José L Hernández-Ramos, and Vasileios Kouliaridis. Demystifying in-vehicle intrusion detection systems: A survey of surveys and a meta-taxonomy. *Electronics*, 11(7):1072, 2022.

[19] Mohamed Karrouchi et al. Analysis of current attacks on the can bus and countermeasures. In *2023 ICEGC Conference Proceedings*, 2023. Accessed: 2025-11-04.

[20] C. Lab. Datasets. https://www.crysys.hu/research/vehicle-security/, 2017. Accessed: 2024-10-10.

[21] Brooke Lampe and Weizhi Meng. can-train-and-test: A Curated CAN Dataset for Automotive Intrusion Detection, August 2023. arXiv:2308.04972 [cs].

[22] Hyunsung Lee, Seong Hoon Jeong, and Huy Kang Kim. Otids: A novel intrusion detection system for in-vehicle network by using remote frame. In *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, pages 57–5709. IEEE, 2017.

[23] Seyoung Lee, Wonsuk Choi, Insup Kim, Ganggyu Lee, and Dong Hoon Lee. A Comprehensive Analysis of Datasets for Automotive Intrusion Detection Systems. *Computers, Materials & Continua*, 76(3):3413–3442, 2023.

[24] Seyoung Lee, Hyo Jin Jo, Aram Cho, Dong Hoon Lee, and Wonsuk Choi. Ttids: Transmission-resuming time-based intrusion detection system for controller area network (can). *IEEE Access*, 10:52139–52153, 2022.

[25] Hongyu Liu and Bo Lang. Machine learning and deep learning methods for intrusion detection systems: A survey. *Applied Sciences*, 9(20):4396, 2019.

[26] Yangyang Liu, Lei Xue, Sishan Wang, Xiapu Luo, Kaifa Zhao, Pengfei Jing, Xiaobo Ma, Yajuan Tang, and Haiying Zhou. Vehicular intrusion detection system for controller area network: A comprehensive survey and evaluation. *IEEE Transactions on Intelligent Transportation Systems*, 26(7):10979–11009, 2025.

[27] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, USA, 1997.

[28] Alberto Morato and Federico Tramarin. Bridging real-time can networks with in-vehicle single pair ethernet: A time-sensitive networking approach. *Acta IMEKO*, 14(3):1–10, 2025.

[29] Andreas C. Müller and Sarah Guido. *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media, Sebastopol, CA, USA, 2016.

[30] Alfredo Nascita, Giuseppe Aceto, Domenico Ciuonzo, Antonio Montieri, Valerio Persico, and Antonio Pescapé. A survey on explainable artificial intelligence for internet traffic classification and prediction, and intrusion detection. *IEEE Communications Surveys & Tutorials*, 2024. early access.

[31] N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert. Trends in automotive communication systems. *Proceedings of the IEEE*, 93(6):1204–1223, 2005.

[32] Damilola Oladimeji et al. Canattack: Assessing vulnerabilities within controller area network. *PMC / Scientific Reports(?)*, 2023. Accessed: 2025-11-04.

[33] Anonymous or CSE570-19 Course Material. A survey of automotive ethernet technologies and protocols. 2019. Available as open PDF overview: https://www.cse.wustl.edu/~jain/cse570-19/ftp/auto_eth.pdf.

[34] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1:81–106, 1986.

[35] Sampath Rajapaksha, Harsha Kalutarage, M. Omar Al-Kadri, Andrei Petrovski, Garikayi Madzudzo, and Madeline Cheah. Ai-based intrusion detection systems for in-vehicle networks: A survey. *ACM Comput. Surv.*, 55(11), February 2023.

[36] Muhammad Sami, Matthew Ibarra, Anamaria C Esparza, Saleh Al-Jufout, Mehrdad Aliasgari, and Mohammad Mozumdar. Rapid, multi-vehicle and feed-forward neural network based intrusion detection system for controller area network bus. In *2020 IEEE Green Energy and Smart Systems Conference (IGESSC)*, pages 1–6. IEEE, 2020.

[37] Eunbi Seo, Hyun Min Song, and Huy Kang Kim. GIDS: GAN based Intrusion Detection System for In-Vehicle Network. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pages 1–6, August 2018. arXiv:1907.07377 [cs].

[38] Shaila Sharmin, Hafizah Mansor, Andi Fitriah Abdul Kadir, and Normaziah A. Aziz. Benchmarking frameworks and comparative studies of controller area network (can) intrusion detection systems: A review, 2024.

[39] Craig Smith. *The Car Hacker's Handbook: A Guide for the Penetration Tester*. William Pollock, 2016. Last accessed on 2022.05.01.

[40] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, pages 305–316. IEEE, 2010.

[41] Hyun Min Song, Jiyoung Woo, and Huy Kang Kim. In-vehicle network intrusion detection using deep convolutional neural network. *Vehicular Communications*, 21:100198, 2020.

[42] Dario Stabili, Francesco Pollicino, and Alessio Rota. A benchmark framework for CAN IDS. *ITASEC21: Italian Conference on Cyber Security*.

[43] Adrian Taylor, Nathalie Japkowicz, and Sylvain Leblanc. Frequency-based anomaly detection for the automotive can bus. In *2015 World Congress on Industrial Control Systems Security (WCICSS)*, pages 45–49. IEEE, 2015.

[44] University of Ljubljana – Bioinformatics Lab. Orange data mining (version 3.x) — a visual programming environment for data analysis, 2024.

[45] Abhishek Verma and Virender Ranga. Machine learning based intrusion detection systems for iot applications. *Wireless Personal Communications*, 111(4):2287–2310, 2020.

[46] Miki E Verma, Michael D Iannacone, Robert A Bridges, Samuel C Hollifield, Pablo Moriano, Bill Kay, and Frank L Combs. Addressing the lack of comparability & testing in can intrusion detection research: A comprehensive guide to can ids data & introduction of the road dataset. *arXiv preprint arXiv:2012.14600*, 2020.

[47] Peter Waszecki, Philipp Mundhenk, Sebastian Steinhorst, Martin Lukasiewycz, Ramesh Karri, and Samarjit Chakraborty. Automotive electrical and electronic architecture security via distributed in-vehicle traffic monitoring. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 36(11):1790–1803, 2017.

Table 1: Model Performance Metrics

Dataset	Attack	FPR	FNR	Precision	Recall	F1-score	Comments
HCRL Car Hacking	DoS	0.000	0.000	1.000	1.000	1.000	1 Unique Attack
	Fuzzing (99%-Tree)	0.059	0.001	0.994	0.941	0.967	51 nodes
	Fuzzing (100%-Tree)	>0.001	>0.001	0.996	1.000	0.998	389 nodes
	Spoofing (Gear & RPM)	0.000	0.000	1.000	1.000	1.000	1 Unique Attack
Survival	DoS	0.000	0.000	1.000	1.000	1.000	1 Unique Attack
	Fuzzing	0.024	>0.001	0.997	0.976	0.987	25 nodes
	Fuzzing (100%)	0.002	0.001	0.997	0.998	0.998	165 nodes
	Malfunction	0.003	0.000	0.998	0.995	0.997	5 nodes
Can-Train-and-Test	DoS / Standstill (100%)	0.000	0.000	1.000	1.000	1.000	1 Unique Attack
	Fuzzing (100%)	0.014	0.002	0.893	0.986	0.937	235 nodes
	Systematic (100%)	0.010	>0.001	0.999	0.990	0.995	127 nodes
	Speed-Spoofing (100%)	0.000	0.000	1.000	1.000	1.000	3 Unique Attacks
	Triple (100%)	0.002	0.000	1.000	0.998	0.999	115 nodes
Car-Hacking Attack & Defense	DoS Train / Prelim	0.000	0.000	1.000	1.000	1.000	1 Unique Attack Trace
	Fuzzing Train	0.101	>0.001	0.988	0.899	0.942	101 nodes
	Fuzzing Train (100%)	0.003	>0.001	0.994	0.996	0.995	387 nodes
	Fuzzing Prelim	0.098	>0.001	0.999	0.902	0.948	101 nodes
	Fuzzing Prelim (100%)	0.004	0.001	0.980	0.996	0.988	387 nodes
	Fuzzing Final	0.068	0.001	0.967	0.932	0.949	101 nodes
	Fuzzing Final (100%)	0.006	0.004	0.871	0.994	0.928	387 nodes
	Spoofing Train (100%)	0.000	>0.001	0.999	1.000	1.000	25 nodes
	Spoofing Prelim (100%)	1.000	>0.001	0.000	0.000	0.000	25 nodes
	Spoofing Final (100%)	1.000	>0.001	0.000	0.000	0.000	25 nodes
	Replay Train	0.454	0.002	0.837	0.546	0.661	3 809 nodes
	Replay Train (100%)	0.255	0.003	0.784	0.745	0.764	10 707 nodes
	Replay Prelim	0.555	0.003	0.715	0.445	0.548	3 809 nodes
	Replay Prelim (100%)	0.555	0.003	0.709	0.445	0.547	10 707 nodes

Table 2: Abbreviations used in dataset overview tables.

Category	Abbreviations and Meanings
Attack Types	D – DoS, F – Fuzzing, S – Spoofing, M – Masquerade, R – Replay, Su – Suppression, Di – Diagnostic, RPMt – RPM Tampering.
Data Source	R – Real, S – Synthetic, M – Mixed (Real/Synthetic).
Message Structure	T – Timestamp, ID – CAN ID, DLC – Data Length Code, Data – Payload Data, Sig – Signals.

Table 3: Overview of available CAN intrusion datasets with criteria for selection (real source, labels exist, access is provided)

Dataset	Environment	Attack Types	Source	Attributes	Labels	Access	Analyzed
OTIDS [22]	1 vehicle	D, F	R	T, ID, DLC, Data	✗	✓	✓
Car-Hack. [37]	1 vehicle	D, F, S	R	T, ID, DLC, Data	✓	✗	✗
Survival [10]	3 vehicles	D, F	R	T, ID, DLC, Data	✓	✓	✓
Aut. CAN [6]	2 vehicles; 1 testbed	D, F, R, Di, Su	S	T, ID, Data	-	✓	✗
SynCAN [11]	1 simulation	S, Su, M	S	T, ID, Sig	✓	✓	✗
CarChall. [16]	1 vehicle	D, F, R, S	R	T, ID, DLC, Data	✓	✓	✓
ROAD [46]	1 vehicle	S, F, M	M	T, ID, Data	✗	✓	✗
TTIDS [24]	1 vehicle	M	R	T, ID, DLC, Data	✗	✓	✗
ct&t [21]	4 vehicles; 1 testbed	D, S, F, M, R, Su	R	T, ID, DLC, Data	✓	✓	✓
Sim CAN [17]	1 testbed	D, F	S	T, ID, DLC, Data	✗	✗	✗
CrySyS [20]	1 vehicle; 26 recordings	S, M	R	T, ID, DLC, Data	-	✓	✗
HCRL [41]	2 vehicles	D, S, F, RPMt	R	T, ID, Data	✓	✓	✓
AEGIS [15]	1 vehicle; 3 drivers	None	R	T, ID, DLC, Data	✗	✓	✗
ML350 [36]	1 vehicle	Unknown	R	T, ID, DLC, Data	✓	✗	✗