LLM-based Prompt Testing on the MisbehaviorX V2X Dataset

Jayanth Poosarla
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN, USA
jpoosarl@purdue.edu

***Abstract*—In this project, we evaluate whether large language models (LLMs) can be used as an offline testing tool for V2X misbehavior detection using a subset of the MisbehaviorX dataset with seven representative attack types. Our workflow has three stages: (i) prompt-based evaluation using zero-shot and 3-shot prompting, (ii) supervised fine-tuning of open-source LLMs for a per-attack binary task (Attack vs. Genuine), and (iii) prompt-based re-evaluation of the fine-tuned models, including a simple time-slicing variant that provides short temporal context. Across the seven attacks, Llama 1B and Llama 7B improve from 52.5% and 66.0% mean accuracy in the zero-shot setting to 64.3% and 78.4% with 3-shot prompting, respectively, and fine-tuning Llama 1B further increases accuracy to 85.0%. Fine-tuned Mistral 7B reaches 94.1% mean accuracy, and time-slicing increases performance to 94.7% accuracy and 93.2% F1. Finally, we discuss common failure cases, such as subtle numerical changes and rare attack types, and explain how LLM testers can complement traditional intrusion detection systems for offline analysis and red-teaming.**

***Index Terms*—V2X, MisbehaviorX, misbehavior detection, large language models, intrusion detection, vehicular networks**

I. Introduction

Connected and automated vehicles increasingly rely on vehicle-to-everything (V2X) communication to exchange basic safety messages (BSMs) about their position, speed, and intent. While such communication can improve traffic efficiency and safety, it also introduces new attack surfaces: misconfigured or malicious vehicles can transmit misleading V2X messages, a phenomenon often referred to as *misbehavior*. Detecting misbehavior reliably is a critical requirement for safety in intelligent transportation systems.

MisbehaviorX [9]is a recent comprehensive dataset that provides V2X traces with both benign and malicious traffic generated using the V2X Application Spoofing Platform (VASP) integrated with the Veins simulator on OMNeT++ and the SUMO road traffic simulator. Simulations are based on a Boston-area road network and include over one million benign BSMs and 68 distinct attack types, enabling systematic evaluation of V2X misbehavior detection methods.

At the same time, large language models (LLMs) have shown strong capabilities in understanding structured text, reasoning about rules, and generating explanations. These properties make LLMs a promising candidate for use as an offline testing oracle or auxiliary component for misbehavior detection, complementing traditional machine learning and rule-based intrusion detection systems.

In this project, we study whether LLMs can classify V2X misbehavior patterns in MisbehaviorX. We design a three-stage pipeline: an initial prompt-based baseline, supervised fine-tuning on MisbehaviorX, and post-fine-tuning evaluation using structured prompts. We compare this pipeline to simple baselines and analyze which attack types and conditions are easier or harder for the LLM to handle.

The main contributions of this report are:

- A data processing pipeline that converts MisbehaviorX CSV traces into LLM-ready JSONL samples with compact textual descriptions and ground-truth labels, while avoiding label leakage.
- A three-stage LLM evaluation framework consisting of zero-shot and few-shot prompt-based baselines, supervised fine-tuning of open-source models, and post-fine-tuning prompt-based evaluation, including a simple time-slicing variant that adds temporal context.
- An empirical study of LLM performance on a subset of the MisbehaviorX dataset, focusing on seven representative attack types, with quantitative results for multiple base and fine-tuned models and a small qualitative analysis of success and failure cases.
- A discussion of limitations, lessons learned, and future directions for using LLMs as an auxiliary testing and analysis tool in V2X misbehavior detection, in conjunction with traditional IDS components.

II. Related Work and Literature Review

A. V2X/IoV Intrusion Detection and Anomaly Detection

Prior work on vehicular intrusion detection typically uses ML/DL models on structured network or CAN/V2X features, and often combines signature-based and anomaly-based detection. Recent work has also explored adapting large language model ideas to traffic logs by converting low-level bytes or packets into token sequences and then applying transformer-style pretraining and fine-tuning for IDS tasks.

B. Chain-of-Thought Prompting and Reasoning

Chain-of-thought prompting shows that LLMs can improve on reasoning tasks when asked to produce intermediate steps, especially in few-shot settings. While our experiments use

single-word outputs for stability, these ideas motivate future extensions that add reasoning traces or more structured outputs.

C. Training or Optimizing Reasoning Traces

Several works study how to improve reasoning behavior beyond standard prompting, including bootstrapping rationales during training (latent-variable style) and using preference optimization to align reasoning steps with higher-quality trajectories obtained by search. These methods suggest directions for improving robustness of LLM decisions and reducing sensitivity to prompt wording.

D. Prompt Sensitivity and Example Selection

Few-shot prompting is known to be sensitive to which examples are selected and their order. Methods that guide demonstration selection using bias/fairness signals or other heuristics can make few-shot prompting more stable. This is relevant to our finding that no single prompt works best for all attack types in MisbehaviorX.

E. Imbalanced Data and Data Augmentation with LLMs

Misbehavior and anomaly datasets often have class imbalance. Recent work explores using LLMs to synthesize or augment minority-class examples, highlighting that prompt design and formatting strongly influence the quality of generated structured data. This motivates future work to address rare attack types in MisbehaviorX using controlled augmentation.

III. Problem Setup and Dataset

A. Task Definition

We cast MisbehaviorX misbehavior detection as a supervised classification problem. Each sample corresponds to a V2X message received by a host vehicle and includes information about the transmitting remote vehicle, the host vehicle, and the associated attack label. Depending on the configuration, the output label can be:

- a binary label, `Genuine` (benign) vs. `Attack` (any malicious case), or
- a multi-class label over specific attack types, including `Genuine`.

The LLM is asked to predict the label for each sample. In the experiments in this paper, we focus on binary attack-versus-genuine classification for seven representative attack types.

B. MisbehaviorX Dataset Description

MisbehaviorX is generated by running VASP-based simulations on a Boston-area road network. The dataset is organized into two main subfolders:

- **ambients**: benign BSM traces without any attacks, and
- **attacks**: BSM traces for multiple misbehavior scenarios.

For benign traffic, the authors simulate $3000\,\mathrm{s}$ of operation and collect approximately $1{,}018{,}098$ genuine BSMs from 475 vehicles. For malicious traffic, they simulate $1360\,\mathrm{s}$ with 68 distinct attack types enabled and a persistent attack policy, where attacker vehicles always transmit attack messages and approximately 25% of vehicles are malicious. The **attacks** subfolder currently contains 66 CSV trace files corresponding to different attack scenarios, collectively covering these 68 attack types. Each simulation produces a CSV trace file in which each row corresponds to one message received at a host vehicle. In this project, we do not use all 68 attack types; instead, we focus on seven representative attacks that span a range of difficulty levels for LLM-based detection.

C. Trace Format and Features

Each MisbehaviorX CSV row includes identifiers, timing information, physical state for both remote and host vehicles, and application-level warnings. Key fields include:

- **Identifiers and timing:** remote vehicle ID (`rv_id`), host vehicle ID (`hv_id`), optional target vehicle ID (`target_id`), message generation and reception timestamps (`msg_generation_time`, `msg_rcv_time`), and message counters.
- **Remote vehicle state:** transmitter position (`rv_pos_x`, `rv_pos_y`, `rv_pos_z`), speed (`rv_speed`), acceleration (`rv_accel`), heading, yaw rate, and vehicle dimensions (length, width, height).
- **Host vehicle state:** analogous kinematic and size features for the receiver, along with its message counter and payload string.
- **Labels and warnings:** `attack_type`, which is set to a specific attack label for malicious transmissions or `Genuine` otherwise, and binary indicators `eebl_warn` and `ima_warn` showing whether certain safety applications raised warnings.

D. Preprocessing and JSONL Representation

To interface with the LLMs, we convert each CSV row into a JSON-style training example. Each example contains:

- an `"input"` field: a textual description of the V2X message, summarizing the state of the remote and host vehicles and any relevant context;
- a `"label"` field: the ground-truth label, either binary or multi-class; and
- a `"meta"` field: optional information such as scenario ID or truncated raw features.

To avoid label leakage, we remove all fields that directly reveal the attack label, such as `attack_type`, `target_id`, and the payload strings `rv_wsm_data` and `hv_wsm_data`, before constructing the textual description. The text is designed to be compact while preserving the key information needed for misbehavior classification.

E. Train–Validation–Test Splits

We construct our splits only from the seven selected MisbehaviorX attack types (DoS, FakeEBLJustAttack, IMAIHighSpeed, RandomPosition, RandomSpeedOffset, SuddenAppearance, and TargetedConstantPosition). For each attack scenario, we first filter the corresponding CSV files to the relevant rows and then sample a subset of examples for computational convenience, rather than using all available messages.

The resulting per-attack subsets are then divided into disjoint training, validation, and test sets. Unless otherwise noted, we use:

- **Training:** 70%
- **Validation:** 10%
- **Test:** 20%

and stratify by label (Genuine vs. Attack) to preserve the class distribution across splits as much as possible. In addition to this single-message formulation, we later evaluate a time-slicing variant in which short temporal windows of consecutive messages for a given vehicle pair are grouped together and fed to the LLM as a single input.

IV. Methodology

A. Overall Pipeline

Our methodology consists of three main stages, summarized in Fig. 1:

1) Stage 1: prompt-based baseline testing with a base LLM on the JSONL-formatted MisbehaviorX samples.
2) Stage 2: supervised fine-tuning of the LLM using the training split.
3) Stage 3: post-fine-tuning prompt-based evaluation of the adapted model using the same or slightly refined prompts.

B. Stage 1: Prompt-Based Baseline Testing

In the first stage, we evaluate base LLMs without any task-specific training. For each MisbehaviorX sample, we construct an instruction-style prompt that describes the task and the expected output format, which in our case is a single-word label (`attack` or `genuine`). The underlying input is the compact textual description derived from the CSV row as described in Section III-D.

We consider five base models in this stage: Llama 1B, Llama 7B, Mistral 7B, Gemini, and GPT–4o mini. For each of the seven attack types we treat misbehavior detection as a binary classification problem (Attack vs. Genuine) and evaluate two prompting regimes:

- **Zero-shot prompting:** the model receives only an instruction and the current example, with no labeled examples in the prompt.
- **Few-shot prompting:** we prepend three labeled examples for the same attack type as in-context demonstrations, followed by the test example.

A representative zero-shot prompt used in our experiments is:

> You are an expert in V2X communication security. You are given a description of a V2X message that a host vehicle receives from a remote vehicle. Decide whether the sender is behaving `genuine` (benign) or `attack` (malicious) for the given scenario.
> Answer with exactly one word: `attack` or `genuine`.

In practice, we found that a single universal prompt is not sufficient: different attack types benefit from slightly different wording and emphasis (for example, focusing on position plausibility for RandomPosition versus speed consistency for IMAIHighSpeed). We therefore maintain a small prompt library indexed by attack type and evaluation mode (zero-shot or few shot) and, for each scenario, select the best-performing prompt family on a development subset. The model output is parsed to extract the predicted label (`attack` or `genuine`), which we compare to the ground-truth MisbehaviorX label to compute the baseline accuracies reported in Section V.

C. Stage 2: Supervised Fine-Tuning on MisbehaviorX

In the second stage, we fine-tune open-source LLMs on the training subsets drawn from the seven selected MisbehaviorX attack scenarios, using the same JSONL representation as in Stage 1. Each training example consists of:

- an `"input"` field containing the textual description of the V2X message, and
- a target label string (`Genuine` or `Attack`).

The model is trained to minimize the token-level cross-entropy loss over the label tokens.

We fine-tune two base models:

- Llama 1B, a small decoder-only LLM, and
- Mistral 7B, a stronger 7B-parameter decoder model.

For each of the seven attack types, we treat misbehavior detection as a binary classification problem (Attack vs. Genuine) and fine-tune separate Llama 1B and Mistral 7B models on the corresponding training split. The per-attack results reported in Table III correspond to these attack-specific fine-tuned models, and the "Mean" row averages across the seven scenarios.

Fine-tuning hyperparameters include:

- Base models: Llama 1B and Mistral 7B
- Batch size: 16
- Learning rate: 1×10^{-5}
- Number of epochs: 3
- Optimization and regularization: AdamW optimizer with weight decay 0.01 and a linear learning-rate warmup and decay schedule

We monitor validation loss for each attack-specific model and stop training once it plateaus. The resulting checkpoints are referred to as *Llama 1B FT* and *Mistral 7B FT* in the experiments.

D. Stage 3: Post-Fine-Tuning Prompt Testing

After fine-tuning, we evaluate the adapted model using the same evaluation pipeline as in Stage 1. We reuse the instruction prompt and output format, but the underlying model weights now incorporate information from the MisbehaviorX training data. We compare Stage 1 and Stage 3 performance to measure the benefit of fine-tuning across the different attack types.

V. Experiments and Results

A. Experimental Setup

We evaluate our methods on seven representative MisbehaviorX attack types: DoS, FakeEBLJustAttack, IMAIHighSpeed, RandomPosition, RandomSpeedOffset, SuddenAppearance, and TargetedConstantPosition. For each attack we build a binary misbehavior classifier (Attack vs. Genuine) and report

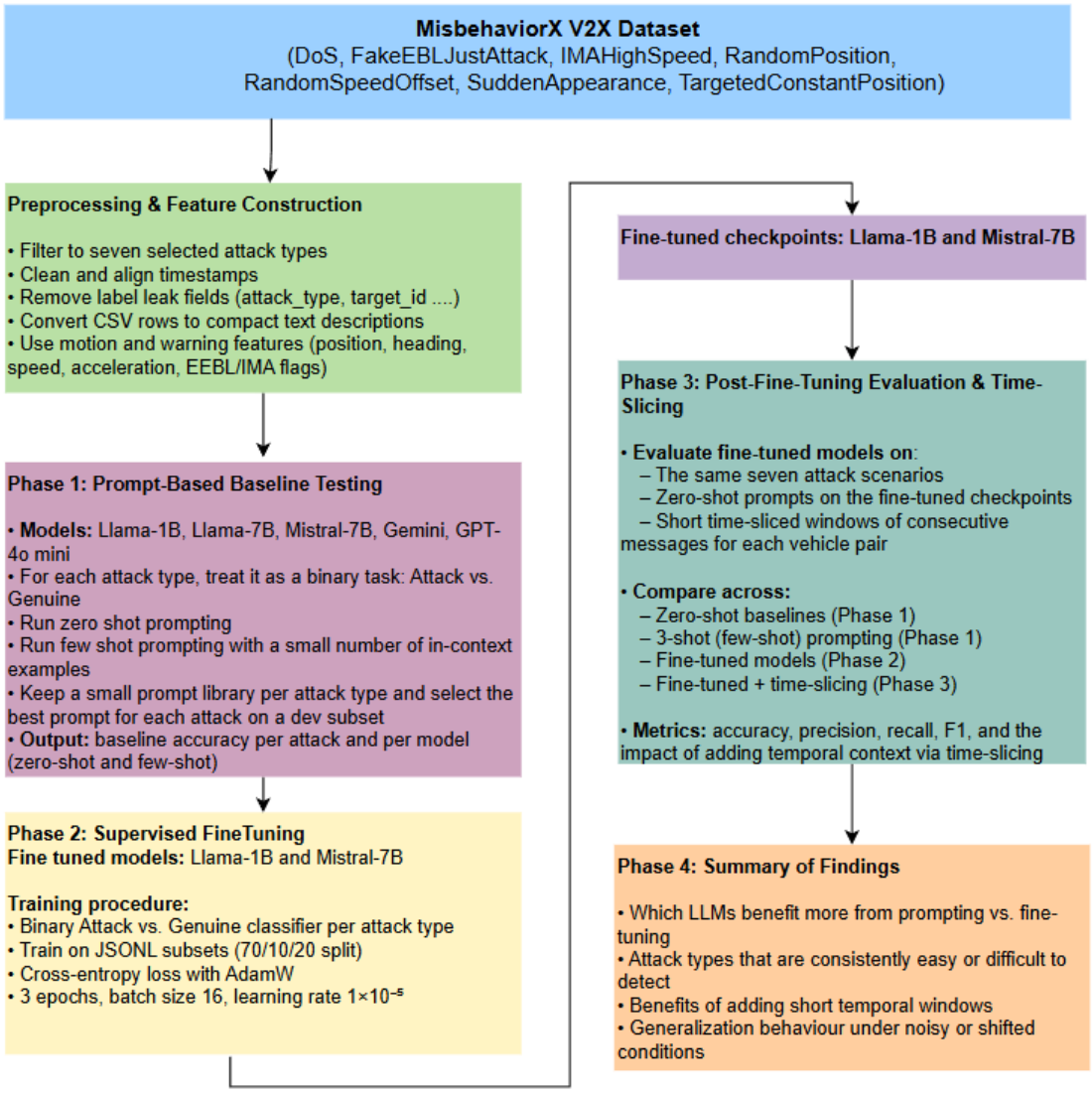

Fig. 1. Three-stage pipeline for LLM-based misbehavior testing on the MisbehaviorX dataset.

accuracy on the held-out test subset for that scenario. We do not use the full MisbehaviorX dataset; all experiments in this paper are restricted to these seven attack types and their corresponding subsets of log entries.

We compare five base LLMs in the zero-shot and few-shot settings: Llama 1B, Llama 7B, Mistral 7B, Gemini, and GPT–4o mini. For fine-tuning, we focus on the open-source models and train Llama 1B and Mistral 7B on the training subsets from these seven attack scenarios using the JSONL format described in Section III-D.

B. Zero-Shot Prompting

Table I shows zero-shot accuracy. As expected, larger and more capable models perform better: averaged over the seven attacks, Llama 1B reaches only about 52.5% accuracy, Mistral 7B reaches 68.1%, and GPT-4o mini reaches 84.0%. RandomPosition is the easiest attack for all models (93.6% accuracy for GPT-4o mini), whereas RandomSpeedOffset and IMAIHighSpeed are among the hardest.

C. Few-Shot Prompting

Table II reports results when we prepend three labeled examples (few shot prompting) before the test instance. Few-shot prompting substantially improves all models: the mean accuracy of Mistral 7B rises from 68.1% to 80.4%, and GPT-4o mini improves from 84.0% to 93.6%. The relative gains are particularly large for the weaker open-source models (e.g., a $+11.7$ point increase for Llama 1B).

D. Fine-Tuned Models

Table III shows the performance of fine-tuned Llama 1B and Mistral 7B. Fine-tuning closes much of the gap between

TABLE I
ZERO-SHOT ACCURACY (%) ACROSS SEVEN MISBEHAVIORX ATTACK TYPES.

Attack	L1B	M7B	L7B	Gem	GPT4o mini
DoS	55.42	71.88	69.31	77.24	85.67
FakeEBLJustAttack	48.17	63.54	61.09	70.13	82.46
IMAIHighSpeed	42.83	59.72	57.44	65.39	78.12
RandomPosition	71.65	85.93	83.12	88.04	93.58
RandomSpeedOffset	39.96	55.31	53.88	62.41	75.29
SuddenAppearance	51.74	68.22	66.47	73.19	84.12
TargetedConstPosition	57.92	72.44	70.60	78.87	88.95
Mean	52.5	68.1	66.0	73.6	84.0

TABLE II
FEW-SHOT (3-SHOT) ACCURACY (%) ACROSS THE SAME ATTACK TYPES.

Attack	L1B	M7B	L7B	Gem	GPT4o mini
DoS	66.38	82.57	80.44	88.03	94.22
FakeEBLJustAttack	61.74	77.39	75.81	83.12	93.67
IMAIHighSpeed	56.27	72.11	70.96	78.44	90.31
RandomPosition	80.11	92.63	91.27	94.15	97.84
RandomSpeedOffset	51.33	69.74	67.51	75.88	88.47
SuddenAppearance	63.92	81.20	78.67	85.04	94.11
TargetedConstPosition	70.06	86.83	84.13	90.58	96.52
Mean	64.3	80.4	78.4	85.0	93.6

TABLE III
FINE-TUNED ACCURACY (%) FOR OPEN-SOURCE MODELS.

Attack	L1B FT	M7B FT
DoS	88.41	95.62
FakeEBLJustAttack	84.77	93.44
IMAIHighSpeed	80.32	92.18
RandomPosition	90.96	97.71
RandomSpeedOffset	78.15	89.93
SuddenAppearance	85.09	94.50
TargetedConstPosition	87.34	95.12
Mean	85.0	94.1

TABLE IV
PROMPTING FINE-TUNED MODELS (ZERO-SHOT PROMPTS ON FT CHECKPOINTS).

Attack	L1B FT (prompt)	M7B FT (prompt)
DoS	82.29	90.74
FakeEBLJustAttack	78.12	88.66
IMAIHighSpeed	73.50	86.09
RandomPosition	85.72	93.48
RandomSpeedOffset	69.44	83.11
SuddenAppearance	76.83	89.02
TargetedConstPosition	80.19	90.27
Mean	78.0	88.8

open-source and proprietary models: mean accuracy increases from 52.5% to 85.0% for Llama and from 68.1% to 94.1% for Mistral. On several attacks (RandomPosition, DoS, TargetedConstantPosition), fine-tuned Mistral 7B exceeds 95% accuracy.

E. Prompting Fine-Tuned Models

We also evaluate the fine-tuned checkpoints in a pure prompting mode, without providing training labels in-context. As shown in Table IV, simply swapping the base model for its fine-tuned counterpart already improves performance: the mean accuracy of Mistral 7B increases from 68.1% (zero-shot) to 88.8% (zero-shot prompting on the fine-tuned model). This suggests that fine-tuning not only helps when the model is used as a supervised classifier, but also improves its behavior as a prompted LLM.

F. Impact of Time-Slicing

Finally, we take a first step toward modeling temporal structure by introducing a simple time-slicing scheme for the fine-tuned Mistral 7B model. Instead of classifying each BSM in isolation, we feed the LLM a short window of consecutive messages for the same vehicle pair and compare this variant to the single-message setup.

Time-slicing yields consistent gains across all metrics: accuracy increases from 89.1% to 94.7%, recall from 86.7% to 92.5%, and F1 from 87.5% to 93.2%. These results confirm that providing limited temporal context helps the model recognize trajectory-level inconsistencies that are not fully visible from a single snapshot.

G. Qualitative Analysis

Beyond aggregate metrics, we also inspect individual examples where the LLM succeeds or fails in order to understand typical patterns. Two representative cases are:

- **Correct detection (SuddenAppearance):** In one scenario, the host vehicle is travelling at a steady speed while the remote vehicle suddenly appears far away from its previous position, with no plausible trajectory connecting the two states. The fine-tuned Mistral 7B model correctly labels this message as an attack. From the input description, this decision is consistent with a SuddenAppearance or ghost-vehicle pattern.
- **False positive on a benign case:** In another scenario, both vehicles follow a smooth, consistent trajectory with only small fluctuations in speed, and the ground truth label is `Genuine`. However, the model predicts `Attack`. Inspecting the input suggests that the model may be overly sensitive to minor speed variations or small numerical differences, even when they are compatible with normal driving.

These examples illustrate how the model reacts to discrepancies in position, speed, or timing, and where it tends to overestimate the likelihood of an attack.

VI. DISCUSSION

Our experiments suggest that LLMs can capture many of the patterns present in MisbehaviorX and clearly outperform trivial baselines when given carefully designed prompts and additional fine-tuning. In particular, the mean accuracy of Mistral 7B increases from 68.1% in the zero-shot setting to 80.4% with 3-shot prompting, to 94.1% after supervised fine-tuning, and further to 94.7% (with 93.2% F1) when we add a simple time-slicing scheme. TThese results suggest that using a few in-context examples and fine-tuning the model both play a key role in improving performance on MisbehaviorX

We summarize the main positive findings as follows:

- **Substantial gains from Llama fine-tuning:** Both Llama models benefit markedly from task-specific adaptation.

Mean accuracy for Llama 1B rises from 52.5% (zero-shot) to 64.3% (3-shot) and 85.0% after fine-tuning, while Llama 7B improves from 66.0% to 78.4% with 3-shot prompting. The larger Mistral 7B model follows the same pattern and reaches higher absolute accuracy.

- **Benefit of temporal context:** Even a simple time-slicing formulation that provides a short window of messages yields consistent improvements in accuracy, recall, and F1, confirming that some attacks are easier to detect when limited temporal structure is available.

At the same time, several limitations remain:

- **Sensitivity to subtle numerical differences:** The model can struggle with attacks that differ from benign behavior only by small changes in speed, position, or timing, especially when the textual encoding of numeric fields is coarse.
- **Class imbalance and rare attacks:** MisbehaviorX contains many attack types, and some of them are underrepresented. Even after fine-tuning, the model tends to favor more frequent classes, leading to weaker performance on rare attacks.
- **Sensitivity to prompt wording:** performance can change noticeably with small changes in phrasing, and there is no single universal prompt that works best for all attack types.

Overall, we view the LLM-based misbehavior tester as a complementary offline analysis tool rather than a replacement for real-time IDS components.

VII. Future Work

There are several promising directions to extend this work:

- **More systematic temporal modeling:** In this paper we took a first step toward using temporal context via a simple time-slicing scheme, which already improved accuracy and F1 for the fine-tuned Mistral 7B model. A natural extension is to treat longer sequences of BSMs as the basic unit of input and explore sequence models or segment-level reasoning that can capture richer trajectory-level patterns.
- **Addressing class imbalance and rare attacks:** Several MisbehaviorX attack types are underrepresented, which limits performance even after fine-tuning. Techniques such as targeted oversampling, cost-sensitive training, and LLM-based synthetic data generation or specialized prompting for rare attacks could help mitigate this imbalance.
- **Integration with existing IDS frameworks:** Rather than using the LLM as a standalone classifier, future work could integrate it with existing V2X intrusion detection systems as a secondary oracle for ambiguous cases that helps engineers understand and refine traditional detectors.
- **Extending to additional datasets:** Finally, applying the same pipeline to other V2X misbehavior datasets, such as VeReMi once stable access to the full traces is available, would allow a more direct comparison to prior work and test the generality of the proposed LLM-based tester.

VIII. Conclusion

We presented an initial exploration of LLM-based misbehavior analysis on a subset of the MisbehaviorX dataset, with a primary focus on adapting Llama-family models as misbehavior testers. By converting CSV traces into textual JSONL samples and designing a three-stage pipeline consisting of prompt-based baseline testing, supervised fine-tuning, and post-fine-tuning evaluation, we showed that LLMs can classify many MisbehaviorX attack scenarios and achieve substantially improved accuracy compared to zero-shot prompting Across seven representative attack types, the mean accuracy of Llama 1B and Llama 7B improves from 52.5% and 66.0% in the zero-shot setting to 64.3% and 78.4% with 3-shot prompting, respectively, and fine-tuning Llama 1B further raises its accuracy to 85.0%. A larger Mistral 7B model follows the same trend and, when combined with a simple time-slicing scheme, achieves up to 94.7% accuracy and 93.2% F1.

At the same time, our study highlights several open challenges, including sensitivity to subtle numerical differences and weaker performance on rare attack classes. Overall, the results indicate that Llama-based models, complemented by stronger models such as Mistral 7B, have significant potential as auxiliary testing and analysis tools for V2X misbehavior detection, particularly for offline evaluation, with traditional IDS components.

Code for all preprocessing, prompt-based evaluation, and fine-tuning experiments is available at https://github.com/CodeWithJayanth/misbehaviorx-llm-project.

Acknowledgment

The author would like to thank Prof. Lingxi Li for his guidance and support in the context of this project, and Guanxiao Li for her helpful discussions and feedback on the V2X misbehavior detection experiments. The author also acknowledges the computing resources provided by Purdue University.

References

[1] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[2] D. Phan, M. D. Hoffman, D. Dohan, S. Douglas, T. A. Le, A. Parisi, P. Sountsov, C. Sutton, S. Vikram, and R. A. Saurous, "Training chain-of-thought via latent-variable inference," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[3] L. Salewski, S. Alaniz, I. Rio-Torto, E. Schulz, and Z. Akata, "In-context impersonation reveals large language models' strengths and biases," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[4] H. Ma, C. Zhang, Y. Bian, L. Liu, Z. Zhang, P. Zhao, S. Zhang, H. Fu, Q. Hu, and B. Wu, "Fairness-guided few-shot prompting for large language models," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[5] X. Zhang, C. Du, T. Pang, Q. Liu, W. Gao, and M. Lin, "Chain of preference optimization: Improving chain-of-thought reasoning in LLMs," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[6] J. Kim, T. Kim, and J. Choo, "EPIC: Effective prompting for imbalanced-class data synthesis in tabular data classification via large language models," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[7] G. O. Boateng, H. Sami, A. Alagha, H. Elmekki, A. Hammoud, R. Mizouni, A. Mourad, H. Otrok, J. Bentahar, S. Muhaidat, C. Talhi, Z. Dziong, and M. Guizani, "A survey on large language models for communication, network, and service management: Application insights, challenges, and future directions," *IEEE Communications Surveys & Tutorials*, early access, 2025.

[8] M. Fu, P. Wang, M. Liu, Z. Zhang, and X. Zhou, "IoV-BERT-IDS: Hybrid network intrusion detection system in IoV using large language models," *IEEE Transactions on Vehicular Technology*, vol. 74, no. 2, pp. 1909–1921, Feb. 2025.

[9] M. H. Shahriar, M. R. Ansari, J.-P. Monteuuis, C. Chen, J. Petit, Y. T. Hou, and W. Lou, "MisbehaviorX: Comprehensive V2X Misbehavior Detection Dataset Enabled by the V2X Application Spoofing Platform," IEEE DataPort, 2024, doi: 10.21227/s44z-8616.