LLM-Enhance: Fine-Tuning Large Language Models for Enhanced Detection of Common Weakness Enumerations

Shrey Modi, Rahul Vishwakarma, and Amin Rezaei
California State University, Long Beach

Abstract

Detecting Common Weakness Enumerations (CWEs) across hardware-software systems remains a critical challenge due to data scarcity, complex vulnerability patterns, and limited generalization of traditional analysis tools. This work introduces **LLM-Enhance**, a Machine Learning (ML) framework that leverages fine-tuned Large Language Models (LLMs) to generate high-quality CWE datasets and improve vulnerability detection performance. Using real-world samples combined with synthetic data augmentation, we construct two large-scale CWE datasets of 50,000 entries each. We evaluate six ML models on both pre-trained and custom LLM-generated datasets. Experimental results demonstrate that transformer models consistently outperform other approaches, achieving up to 84% accuracy with superior probabilistic calibration. Our findings highlight the effectiveness of LLM fine-tuning for scalable CWE dataset generation and robust vulnerability detection, underscoring the potential of transformer-based models for cross-domain security analysis in modern hardware-software systems.

Introduction & Background

CWEs are critical vulnerabilities in hardware and software systems [1]. This paper presents an ML and LLM-based framework for CWE detection using both real and synthetic data. We construct two large CWE datasets enhanced via data augmentation and LLM fine-tuning and evaluate different ML models on these datasets. Recent studies emphasize the advantages of ML over traditional static analysis for early vulnerability detection, including applications in Internet of Things (IoT) [2] and log-based anomaly detection [3]. LLMs have shown promise in extracting vulnerability information from unstructured sources [4], zero-shot vulnerability repair [5], code vulnerability detection [6], policy-based protection [7], and hardware security assertion generation [8]. However, challenges remain, such as prompt sensitivity and incomplete responses. Despite these limitations, LLMs demonstrate strong potential in automating security tasks, including penetration testing [9] and solving offensive security challenges [10]. Fig. 1 illustrates the selected CWEs that lie at the intersection of software and hardware vulnerabilities, providing context for the scope of our dataset and analysis.

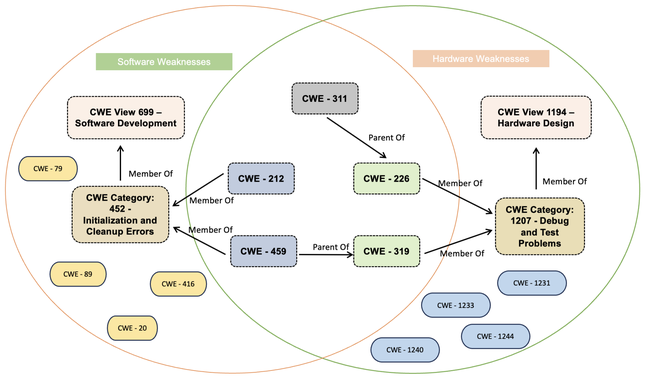

Fig. 1: CWEs at the Intersection of Software and Hardware Weaknesses

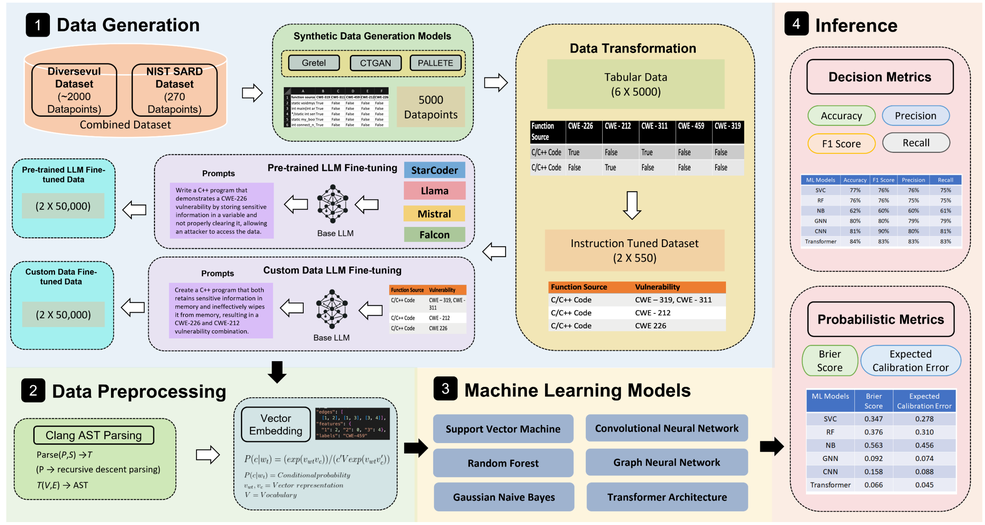

Fig. 2: LLM-Enhance Workflow

LLM-Enhance

We propose **LLM-Enhance**, an end-to-end workflow for hardware-software CWE detection shown in Fig. 2:

A. Data Generation
- Collected ~2,270 real CWE samples from DiverseVul [11] and NIST SARD [12]
- Focused on five CWEs: CWE-212, CWE-226, CWE-311, CWE-319, CWE-459
- Expanded to ~5,000 samples using synthetic data generation (i.e., Gretel [13], CTGAN [14], PALETTE [15])
- Fine-tuned multiple transformer-based LLMs (i.e., LLaMA, Falcon, Mistral, StarCoder)
- Adapted data formats per model (CSV, JSONL, syntax trees, dependency graphs)
- Generated two datasets of ~50,000 samples each: Pre-trained LLM dataset and Custom fine-tuned LLM dataset

B. Data Preprocessing
- Converted C++ source code into Abstract Syntax Trees (ASTs) using Clang
- ASTs capture structured code semantics and programming constructs
- Transformed ASTs into JSON format for flexibility and efficient analysis
- Enabled compatibility with advanced ML and deep learning models

C. Machine Learning Algorithms

Evaluated six ML models for CWE detection: SVC, Random Forest, Gaussian Naive Bayes, CNN, GNN, and Transformer
- CNNs capture structural code patterns
- GNNs model dependencies via graph representations
- Transformers leverage self-attention to capture complex contextual relationships
- Enables direct comparison between traditional and state-of-the-art models

Experimental Results

Each dataset is split into 70% training, 15% validation, and 15% testing to ensure balanced evaluation and robust training. Our training minimizes cross-entropy loss with the Adam optimizer, using learning rate (η) of 0.001 over 30 epochs. The experiments are run on a 12-core Intel Xeon Silver 4310 processor with 16GB of DDR4 RAM and an NVIDIA 64GB A16 GPU.

Table 1 show the decision metrics for pre-trained and custom fine-tuned LLM datasets. Traditional models (SVC, Random Forest, GNB) show moderate to low accuracy, while deep learning models (CNN, GNN) achieve improved detection performance. All models perform significantly better on the custom fine-tuned LLM dataset compared to the pre-trained dataset.

Table 2 show the probabilistic metrics for pre-trained and custom fine-tuned LLM datasets. Brier Score (BS) and Expected Calibration Error (ECE) evaluate the quality and calibration of predicted probabilities, not just classification accuracy. Deep learning models (CNN, GNN, Transformer) achieve substantially lower BS and ECE than traditional ML models, indicating more reliable confidence estimates. Again, all models show improved probabilistic performance on the custom fine-tuned LLM dataset compared to the pre-trained dataset.

Table 1: Decision Metrics

ML Model	Pre-Trained LLM Dataset				Custom LLM Dataset			
	Acc.	F1 Sc.	Prec.	Rec.	Acc.	F1 Sc.	Prec.	Rec.
SVC	46%	43%	43%	44%	77%	76%	76%	75%
RF	44%	43%	43%	42%	76%	76%	75%	75%
GNB	37%	36%	36%	35%	62%	60%	60%	61%
GNN	48%	45%	46%	47%	80%	80%	79%	79%
CNN	47%	45%	46%	46%	81%	80%	80%	81%
Trans.	53%	53%	53%	52%	84%	83%	83%	83%

Table 2: Probabilistic Metrics

ML Model	Pre-Trained LLM Dataset		Custom LLM Dataset	
	Brier Score	Exp. Cal. Error	Brier Score	Exp. Cal. Error
SVC	0.456	0.347	0.347	0.278
RF	0.487	0.391	0.376	0.310
GNB	0.620	0.510	0.563	0.456
GNN	0.342	0.287	0.092	0.074
CNN	0.320	0.236	0.158	0.088
Trans.	0.290	0.187	0.066	0.045

References

1. Common weakness enumeration: Root cause mapping of vulnerabilities. In https://cwe.mitre.org/.
2. Rakibul Hassan et al., "Automated supervised topic modeling framework for hardware weaknesses," In *24th International Symposium on Quality Electronic Design (ISQED)*, pp. 1-8, 2023.
3. Van-Hoang Le et al., "Log-based anomaly detection with deep learning: How far are we? In *44th International Conference on Software Engineering (ICSE)*, pp. 1356–1367, 2022.
4. Virendra Ashiwal, et al., "LLM-based vulnerability sourcing from unstructured data," In *IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pp. 634-641, 2024.
5. Hammond Pearce et al., "Examining zero-shot vulnerability repair with large language models," In *IEEE Symposium on Security and Privacy (S&P)*, pp. 2339-2356, 2023.
6. Jin Wang et al., "DefectHunter: A novel LLM-driven boosted-conformer-based code vulnerability detection mechanism," In *arXiv*, 2309.15324, 2023.
7. Sudipta Paria et al., "DIVAS: An LLM-based End-to-End Framework for SoC Security Analysis and Policy-based Protection," In *arXiv*, 2308.06932, 2023.
8. Rahul Kande et al., "(Security) assertions by large language models," In *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 4374-4389, 2024.
9. Jiaxin Yu et al., "An insight into security code review with LLMs: Capabilities, obstacles, and influential factors," In *arXiv*, 2401.16310, 2025.
10. Minghao Shao et al., "An empirical evaluation of LLMs for solving offensive security challenges, In *arXiv*, 2402.11814, 2024.
11. Yizheng Chen et al., "DiverseVul: A new vulnerable source code dataset for deep learning based vulnerability detection," In Int. Sym. on Research in Attacks, Intrusions and Defenses, pp 654-668, 2023.
12. Paul E Black, "A software assurance reference dataset: Thousands of programs with known bugs," In Journal of Research of the National Institute of Standards and Technology, vol. 123, pp. 1-3, 2018.
13. Ainul Haezah Noruzman et al., "Gretel.ai: Open-source artificial intelligence tool to generate new synthetic data. In Journal of Innovation in Engineering and Applied Social Sciences, vol. 1, pp 15-22, 2021.
14. Virginia Cortes et al., "Adaptive pruning of neural language models," In Advances in Neural Information Processing Systems (NeurIPS), pp. 6598-6607, 2019.
15. Rahul Vishwakarma et al. "Risk-aware and explainable framework for ensuring guaranteed coverage in evolving hardware trojan detection," In Int. Conf. on Computer Aided Design (ICCAD), pp 1-9, 2023.