VulnBench: A Comprehensive Benchmark for Transformer-Based Vulnerability Detection

Jake Norton
University of Otago
norja159@student.otago.ac.nz

David Eyers
University of Otago
david.eyers@otago.ac.nz

Veronica Liesaputra
University of Otago
veronica.liesaputra@otago.ac.nz

Abstract

Reproducible benchmarking in vulnerability detection remains challenging due to inconsistent implementations, varying data preprocessing, and methodological flaws that compromise fair model comparison. Recent analysis reveals that 9 in 10 vulnerability detection studies use inappropriate evaluation approaches, with models achieving high scores through spurious correlations rather than actual vulnerability detection. We present VulnBench, an extensible open-source framework that standardizes evaluation methodology to enable fair comparison across models and datasets. Our systematic evaluation of CodeBERT, GraphCodeBERT, CodeT5 (encoder-only and full), and NatGen across nine datasets reveals that proper threshold optimization can improve F1-scores by up to 58%, while exposing severe dataset quality variations ranging from 25% to 82% F1-score that question the validity of many existing comparisons. By providing rigorous evaluation protocols, VulnBench enables researchers to distinguish between genuine model improvements and methodological artifacts as well as reducing duplicate time spent on reproducing results.

Introduction

Software vulnerabilities represent critical security risks that can lead to data breaches, system compromises, and substantial economic losses. Traditional vulnerability detection methods rely heavily on static analysis tools and manual code reviews, which are resource-intensive and often miss complex vulnerability patterns. The emergence of machine learning approaches, particularly deep learning models trained on large code corpora, has shown promise in automating vulnerability detection with improved accuracy and coverage.

Recent advances in transformer-based models have revolutionized code understanding tasks. Models like CodeBERT (Feng et al., 2020) and CodeT5 (Wang et al., 2021) leverage pre-training on massive code repositories to capture semantic relationships in source code. Graph-enhanced approaches such as GraphCodeBERT (Guo et al., 2021) incorporate structural code information to improve understanding of data flow and control dependencies.

However, evaluating and comparing these models faces significant methodological challenges. Recent analysis reveals that 9 in 10 vulnerability detection papers use inappropriate problem formulations, with models achieving high scores through spurious correlations rather than actual vulnerability detection (Risse, Liu, and Böhme, 2025). Implementation inconsistencies are common, with studies often relying on incomplete model reproductions or subtly different data preprocessing approaches (Varkey, Jiang, and Huang, 2024).

Furthermore, critical methodological considerations are often overlooked. Default classification thresholds (0.5) are frequently suboptimal for the severely imbalanced datasets common in vulnerability detection, yet systematic threshold optimization remains rare (Esposito et al., 2021; Leevy et al., 2023). Recent work specifically examining vulnerability detection datasets confirms that proper threshold selection and loss function choice significantly impact performance on imbalanced data (Ma et al., 2025), yet many comparative studies fail to address these fundamental evaluation requirements.

Reproducible benchmarking in vulnerability detection remains challenging due to inconsistent implementations, varying data preprocessing, and substantial duplicate effort required for fair model comparison. While prior work has evaluated state-of-the-art models across different datasets, these studies often rely on incomplete implementations or subtly different data handling, hindering reproducibility and collaborative progress.

VulnBench Architecture and Features

VulnBench provides a comprehensive benchmarking platform addressing systematic evaluation challenges in vulnerability detection research. The framework implements four key architectural components:

- **Unified Model Interface:** Supporting diverse transformer architectures through consistent APIs. Researchers can easily add new models by implementing a standard interface, with automatic handling of architectural differences (encoder-only vs. encoder-decoder, different tokenization schemes, loss functions).
- **Automated Data Pipeline:** Handles nine vulnerability datasets with standardized preprocessing. Features in-

clude reproducible seed-based splitting, automatic dataset downloading, consistent function anonymization, and configurable train/validation/test ratios. Custom datasets will require work to convert to the standard JSONL format, however, there are many script recipes to use as templates for this.

- **Rigorous Evaluation Engine:** Implements threshold optimization, multi-seed statistical testing, and comprehensive metrics. Automatically generates performance reports with confidence intervals, statistical significance tests, and comparative analysis.
- **Experiment Tracking:** Integrated Weights & Biases support with standalone logging fallback. Provides real-time training monitoring, hyperparameter comparison, and automated results aggregation across multiple seeds.

Related Work

General Code Understanding Benchmarks. CodeXGLUE (Lu et al., 2021) represents the most comprehensive benchmarking effort for code intelligence, providing 14 datasets across 10 tasks including defect detection, with standardized evaluation protocols and baseline models. However, CodeXGLUE focuses broadly on code understanding rather than specifically addressing the methodological challenges in vulnerability detection evaluation.

Vulnerability-Specific Benchmarks. Several frameworks target vulnerability detection specifically. CASTLE (Dubniczky et al., 2025) introduces a curated collection of 250 compilable C programs for evaluating static analyzers, formal verification tools, and LLMs, with balanced vulnerable/non-vulnerable distributions. VulnLLMEval (Zibaeirad and Vieira, 2024) specifically evaluates large language models on vulnerability detection and patching tasks using real-world CVE data.

Critical Evaluation Issues. Recent work has identified fundamental problems with current evaluation approaches. Risse et al. (Risse, Liu, and Böhme, 2025) demonstrate that 9 in 10 ML-based vulnerability detection papers use inappropriate problem formulations, with models achieving high scores through spurious correlations rather than actual vulnerability detection. Studies of popular datasets (BigVul, CVEFixes, DiverseVul) reveal significant data leakage issues and labeling problems that compromise evaluation validity (Ullah et al., 2024).

Static Analysis Tool Benchmarking. Traditional vulnerability detection tool evaluation has focused on static analysis tools. Web service vulnerability detection studies (Antunes and Vieira, 2010) propose benchmarking approaches for comparing penetration testing tools, static analyzers, and anomaly detectors using metrics like precision, recall, and F-measure. Comprehensive evaluations of memory error detectors (Nong et al., 2021) against benchmark datasets reveal varied accuracy across vulnerability categories and highlight the challenges of fair comparison.

Dataset Quality and Standardization. Systematic surveys (Shiri Harzevili et al., 2024) reveal that 39.1% of vulnerability detection studies use hybrid data sources combining benchmarks, repositories, and projects, while 37.6% rely on standardized benchmark datasets. However, many proposed benchmarking frameworks (Lin et al., 2019) focus on specific neural network architectures rather than providing systematic evaluation methodology.

Distinguishing Characteristics of VulnBench. Unlike existing work, VulnBench addresses the methodological evaluation crisis by providing: (1) standardized threshold optimization addressing severe class imbalances, (2) architecture-aware evaluation protocols handling transformer model differences, and (3) systematic dataset quality assessment revealing artifacts that compromise research validity. While frameworks like CodeXGLUE provide broad code understanding benchmarks and OWASP/CASTLE focus on specific tool evaluation, VulnBench uniquely combines rigorous evaluation methodology with comprehensive transformer model comparison across diverse vulnerability datasets.

Demonstration Results

Table 1: Model Performance (F1-Score %) with Optimized Thresholds

Dataset	CodeBERT	GraphCodeBERT	CodeT5-Enc	CodeT5-Full	NatGen	Best
DIVERSEVUL	25.1±1.2	26.3±1.5	23.8±1.1	27.2±1.4	**28.5±1.6**	28.5
ICVUL	57.8±2.1	59.2±1.8	55.3±2.3	61.4±2.0	**63.7±1.9**	63.7
MVDSC	80.6±0.8	81.2±0.9	79.8±1.1	**82.1±0.7**	81.5±1.0	82.1
Devign	62.3±1.7	**64.8±1.5**	59.1±2.0	63.2±1.8	62.9±1.6	64.8
CVEFixes	41.2±1.9	42.6±2.1	38.7±2.2	43.1±1.8	**44.3±2.0**	44.3
Draper	71.5±1.3	72.8±1.1	69.9±1.6	**74.2±1.2**	73.6±1.4	74.2
VulDeepecker	68.4±1.5	69.7±1.3	66.8±1.7	**71.2±1.4**	70.5±1.6	71.2
Juliet	76.2±1.0	77.1±0.9	74.6±1.2	**78.5±1.1**	77.8±1.0	78.5
Reveal	54.7±2.3	56.2±2.0	52.1±2.5	57.3±2.1	**58.9±1.9**	58.9
Average	59.8	61.1	57.8	**64.2**	62.4	-

We demonstrate VulnBench's capabilities through systematic evaluation of five transformer variants across nine datasets. The tool automatically:

- Identifies datasets with potential quality issues (e.g., MVDSC's unusually high 82% F1 across all models)
- Reveals significant threshold optimization opportunities (15-58% F1 improvements)
- Enables architectural comparisons showing T5-full models consistently outperform encoder-only variants
- Provides statistical confidence through multi-seed evaluation with error bars

Practical Impact: VulnBench enables researchers to distinguish between genuine model improvements and methodological artifacts. The framework has successfully exposed evaluation inconsistencies that could mislead research directions and provides standardized protocols for fair model comparison.

Availability and Future Extensions

VulnBench is available as an open-source framework with comprehensive documentation and example configurations. The modular design enables easy extension with new models, datasets, and evaluation metrics. Future development includes support for additional transformer architectures, or other state-of-the-art models like LineVul or Vul_Detection_Gnn.

References

Antunes, N., and Vieira, M. 2010. Benchmarking vulnerability detection tools for web services.

Dubniczky, R. A.; Horvát, K. Z.; Bisztray, T.; Ferrag, M. A.; Cordeiro, L. C.; and Tihanyi, N. 2025. Castle: Benchmarking dataset for static code analyzers and llms towards cwe detection.

Esposito, C.; Landrum, G. A.; Schneider, N.; Stiefl, N.; and Riniker, S. 2021. Ghost: Adjusting the decision threshold to handle imbalanced data in machine learning. *Journal of Chemical Information and Modeling* 61(6):2623–2640. PMID: 34100609.

Feng, Z.; Guo, D.; Tang, D.; Duan, N.; Feng, X.; Gong, M.; Shou, L.; Qin, B.; Liu, T.; Jiang, D.; and Zhou, M. 2020. Codebert: A pre-trained model for programming and natural languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 1536–1547. Association for Computational Linguistics.

Guo, D.; Ren, S.; Lu, S.; Feng, Z.; Tang, D.; Liu, S.; Zhou, L.; Duan, N.; Svyatkovskiy, A.; Fu, S.; Tufano, M.; Deng, S. K.; Clement, C.; Drain, D.; Sundaresan, N.; Yin, J.; Jiang, D.; and Zhou, M. 2021. Graphcodebert: Pre-training code representations with data flow. In *International Conference on Learning Representations*.

Leevy, J.; Johnson, J.; Hancock, J.; and Khoshgoftaar, T. 2023. Threshold optimization and random undersampling for imbalanced credit card data. *Journal of Big Data* 10.

Lin, G.; Xiao, W.; Zhang, J.; and Xiang, Y. 2019. Deep learning-based vulnerable function detection: A benchmark. In *Information and Communications Security: 21st International Conference, ICICS 2019, Beijing, China, December 15–17, 2019, Revised Selected Papers*, 219–232. Berlin, Heidelberg: Springer-Verlag.

Lu, S.; Guo, D.; Ren, S.; Huang, J.; Svyatkovskiy, A.; Blanco, A.; Clement, C. B.; Drain, D.; Jiang, D.; Tang, D.; Li, G.; Zhou, L.; Shou, L.; Zhou, L.; Tufano, M.; Gong, M.; Zhou, M.; Duan, N.; Sundaresan, N.; Deng, S. K.; Fu, S.; and Liu, S. 2021. Codexglue: A machine learning benchmark dataset for code understanding and generation. *CoRR* abs/2102.04664.

Ma, X.; He, Y.; Keung, J.; Tan, C.; Ma, C.; Hu, W.; and Li, F. 2025. On the value of imbalance loss functions in enhancing deep learning-based vulnerability detection. *Expert Systems with Applications* 291:128504.

Nong, Y.; Cai, H.; Ye, P.; Li, L.; and Chen, F. 2021. Evaluating and comparing memory error vulnerability detectors. *Information and Software Technology* 137:106614.

Risse, N.; Liu, J.; and Böhme, M. 2025. Top score on the wrong exam: On benchmarking in machine learning for vulnerability detection.

Shiri Harzevili, N.; Boaye Belle, A.; Wang, J.; Wang, S.; Jiang, Z. M. J.; and Nagappan, N. 2024. A systematic literature review on automated software vulnerability detection using machine learning. *ACM Comput. Surv.* 57(3).

Ullah, S.; Han, M.; Pujar, S.; Pearce, H.; Coskun, A.; and Stringhini, G. 2024. Llms cannot reliably identify and reason about security vulnerabilities (yet?): A comprehensive evaluation, framework, and benchmarks.

Varkey, A.; Jiang, S.; and Huang, W. 2024. Codecse: A simple multilingual model for code and comment sentence embeddings.

Wang, Y.; Wang, W.; Joty, S.; and Hoi, S. C. 2021. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 8696–8708. Association for Computational Linguistics.

Zibaeirad, A., and Vieira, M. 2024. Vulnllmeval: A framework for evaluating large language models in software vulnerability detection and patching.