Addressing the C/C++ Vulnerability Datasets Limitation: The Good, the Bad and the Ugly

Claudio Curto[1][a], Daniela Giordano[1] and Daniel Gustav Indelicato[2]
[1]*Department of Electrical Electronic and Computer Engineering (DIEEI), University of Catania, Catania, Italy*
[2]*EtnaHitech S.c.p.A., Darwin Technologies S.r.l., Catania, Italy*

Keywords: Vulnerable Code Datasets, Vulnerability Detection, Deep Learning, Data Analysis.

Abstract: Recent years have witnessed growing interest in applying deep learning techniques to software security assessment, particularly for detecting vulnerability patterns in human-generated source code. Despite advances, the effectiveness of deep learning models is often hindered by limitations in the datasets used for training. This study conducts a comprehensive evaluation of one widely used and two recently released C/C++ real-world vulnerable code datasets to assess their impact on the performance of transformer-based models, focusing on generalization across unseen projects, unseen vulnerability types and diverse data distributions. In addition, we analyze the effects of aggregating datasets and compare the results with previous experiments. Experimental results demonstrate that combining datasets significantly improves model generalization across varied distributions, highlighting the importance of diverse, high-quality data for enhancing vulnerability detection in source code.

1 INTRODUCTION

Software security has become a critical priority in an era where vulnerabilities in source code can lead to severe consequences, including data breaches, service disruptions, and compromised system integrity. The need for effective and scalable methods to detect vulnerabilities has driven extensive research into automated solutions, particularly those leveraging deep learning techniques (Curto et al., 2024a). Transformer-based models, such as RoBERTa and its variants (Feng et al., 2020)(Guo et al., 2020), have shown substantial promise in identifying intricate vulnerability patterns in source code (Mamede et al., 2022), (Fu and Tantithamthavorn, 2022), (Curto et al., 2024b). These models, trained on large-scale datasets, offer the potential to surpass traditional rule-based methods in both scalability and precision. However, their reliability and effectiveness remain heavily dependent on the quality and diversity of the datasets used for training.

Recent advancements in software vulnerability prediction (SVP) have demonstrated that high-quality datasets are essential for training effective machine learning models. Studies have highlighted significant challenges in dataset quality, including inaccurate labels, data duplication, and limited diversity in vulnerability types (Croft et al., 2023). For instance, Croft et al. revealed that up to 71% of labels in real-world datasets are inaccurate, which undermines the performance and generalization of models trained on such data. Furthermore, while synthetic datasets like Juliet[1] provide an additional source of training data, they often lack the complexity and variability of real-world vulnerabilities. This has led to an increased focus on real-world datasets (Fan et al., 2020), (Chen et al., 2023), (Ni et al., 2024), which, despite their inherent imperfections, offer a more realistic foundation for training models capable of detecting vulnerabilities in diverse codebases.

One of the main limitations in applying large language models (LLMs) to vulnerability detection is the significant data requirements for training. Beyond the volume of data, the origin of the data, whether synthetic or real-world, plays a critical role in shaping model performance (Chakraborty et al., 2022b). Real-world datasets, derived from actual software projects, are particularly valuable for evaluating the generalization capabilities of models. However, achieving robust generalization remains a significant challenge.

[a] https://orcid.org/0009-0006-6516-7671

[1] https://samate.nist.gov/SARD/test-suites/112

Curto, C., Giordano, D. and Indelicato, D. G.
Addressing the C/C++ Vulnerability Datasets Limitation: The Good, the Bad and the Ugly.
DOI: 10.5220/0013495200003979
In *Proceedings of the 22nd International Conference on Security and Cryptography (SECRYPT 2025)*, pages 355-362
ISBN: 978-989-758-760-3; ISSN: 2184-7711

Generalization in this context refers to a model's ability to detect vulnerabilities across unseen projects, novel vulnerability types (e.g., new Common Weakness Enumeration, CWEs), and varied data distributions. High performance on specific datasets often masks the inability of models to generalize effectively, an issue that has been underexplored in the literature.

This study bridges existing gaps by conducting an in-depth evaluation of three widely-used C/C++ vulnerable code datasets. Using a RoBERTa-based model, we examine its generalization capabilities across three key scenarios: unseen projects, novel vulnerability types, and varying data distributions. Furthermore, we analyze the effects of aggregating these datasets into a unified collection to assess its impact on model performance.

Our results reveal that dataset aggregation significantly improves model generalization, particularly in challenging cases involving unseen projects and previously unencountered vulnerabilities. This finding emphasizes the value of combining datasets to mitigate biases and enhance the robustness of deep learning models in software vulnerability detection.

By addressing dataset limitations and exploring their role in transformer-based models, this work underscores the importance of diverse, high-quality datasets as a foundation for advancing automated software security assessment. The insights presented aim to guide researchers and practitioners in developing scalable and reliable solutions for software vulnerability detection.

2 BACKGROUND

2.1 Transformer Models for Vulnerability Detection

Ensuring software security heavily relies on detecting vulnerabilities, traditionally approached with rule-based methods. However, these methods often struggle with scalability and precision. Recent advancements in deep learning, particularly transformer-based models, have automated vulnerability detection across large codebases (Jiang et al., 2025). Models like CodeBERT (Feng et al., 2020) and other code-specific transformers (Xu et al., 2022), (Chakraborty et al., 2022a), (Rozière et al., 2023) have proven effective in identifying complex vulnerability patterns by understanding both syntax and semantics of programming languages. Their ability to capture long-range dependencies has positioned them at the forefront of the field.

However, their success depends critically on the quality and diversity of training datasets. Issues like data imbalance, inaccurate labels, and lack of diversity can hinder generalization, limiting performance on new codebases or unseen vulnerabilities. This challenge remains a core issue in software security research (Chen et al., 2023). Our study explores how combining multiple datasets and enhancing vulnerability diversity can improve the generalization of transformer models, with a particular focus on the MITRE Top 25 CWEs and new vulnerability categories.

2.2 Vulnerability Datasets Quality

The quality of vulnerability datasets is a crucial, often overlooked, factor in the effectiveness of machine learning models for software security (Croft et al., 2023). While transformer-based models show high performance on specific datasets, their ability to generalize to diverse, unseen data remains a challenge (Chen et al., 2023). Most studies focus on performance metrics within specific datasets (Zhou et al., 2019), neglecting how models handle real-world data distributions.

This emphasis on dataset-specific performance can give a false sense of model effectiveness. High accuracy on a particular dataset may mask weaknesses when confronted with new vulnerabilities or projects. This issue is exacerbated when datasets have biases in the types of vulnerabilities, code structure, or programming languages represented.

To address these challenges, it is essential to focus on both dataset diversity and generalization, ensuring that vulnerability datasets are not only large and comprehensive but also varied enough to represent real-world applications. This diversity is crucial for training models capable of detecting vulnerabilities across different codebases, vulnerability types, and project configurations.

3 RELATED WORKS

The use of deep learning techniques for software vulnerability detection has been an active area of research, with a particular focus on applying both Graph Neural Nework (GNN) (Cheng et al., 2021), (Nguyen et al., 2022), (Hin et al., 2022) and transformer-based models (Curto et al., 2024b), (Fu and Tantithamthavorn, 2022) to identify vulnerability patterns in source code. Several studies have explored the effectiveness of learning-based approaches for vulnerability prediction, demonstrating improvements over traditional

rule-based methods in scalability and accuracy. For instance, recent advancements in software vulnerability prediction (SVP) models have leveraged large-scale datasets to train deep neural networks, achieving state-of-the-art results in predicting security flaws (Curto et al., 2024a). However, the reliability of these models is often compromised by data quality issues, such as inaccurate labels, data duplication, and inconsistency, which can significantly affect model performance (Croft et al., 2023). The importance of dataset diversity and high-quality data was also emphasized by other works (Zheng et al., 2021), which suggested that aggregating multiple datasets could alleviate data bias and improve model robustness and generalization. Recently, (Chakraborty et al., 2024) addressed critical limitations in existing datasets used for evaluating deep learning-based vulnerability detection models. They introduce the RealVul dataset, designed to represent realistic usage scenarios by including complete codebases rather than focusing solely on isolated snippets from fixing commits, as seen in prior datasets like BigVul and SARD (National Institute of Standards and Technology, 2025). They highlight significant discrepancies between the performance of deep learing models on synthetic or limited datasets and their practical application in real-world scenarios. Furthermore, they identify overfitting as a primary issue and propose an augmentation technique to improve generalization. This study underscores the necessity of realistic datasets for evaluating and improving the robustness of vulnerability detection models. Our study extends this body of work by evaluating the impact of real-world vulnerability datasets on transformer-based models, examining the importance of diverse data distributions for improving vulnerability detection and model's generalization through various experimental setups.

4 METHODOLOGY

4.1 Datasets

This section offers a comprehensive overview of the analyzed datasets, detailing their key characteristics.

4.1.1 BigVul

BigVul (Fan et al., 2020) is a large C/C++ vulnerability dataset, collected from open-source GitHub projects. Big-Vul dataset contains the details of CVE entries from 2002 to 2019 and covers 358 different projects that are linked to 4,432 unique code commits. The 4,432 code commits contain the code fixes for 3,754 vulnerabilities in 91 Common Weakness Enumeration (CWE) types. We obtained the current version of BigVul from the official GitHub repository [2].

4.1.2 DiverseVul

DiverseVul (Chen et al., 2023) is another large C/C++ vulnerability dataset obtained by the following procedures: crawling of security issue websites, collection of vulnerability reports, extraction of vulnerability fixing commits, and, for each vulnerability, cloning of the project and extraction of vulnerable and non-vulnerable source code. The result is a total of 16,109 vulnerable functions and 311,560 non-vulnerable functions, extracted from 7,514 commits and covering 150 CWEs. DiverseVul is one of the largest vulnerability codebases, built to provide more quality data for the training of large deep learning models. We obtained the dataset from the author's GitHub repository[3]. As for BigVul, we performed a cleaning procedure on the data. As a matter of fact, all the dataset entries came with the CWE feature as a list of zero, one, or more CWE IDs. For simplicity, we removed all the entries with more than one CWE ID, converted them as a string and all the entries without a CWE ID, represented as '[]' in the table. As a result, we removed 45,922 entries. The resulting dataset is composed of 16,109 vulnerable functions and 265,638 not vulnerable functions.

4.1.3 MegaVul

MegaVul (Ni et al., 2024) is a comprehensive and continually updated dataset developed to support vulnerability detection in C/C++ software. MegaVul aggregates data from the Common Vulnerabilities and Exposures (CVE) database and associated Git-based repositories to address the limitations of previous datasets, including incomplete data, outdated entries, and limited diversity. The dataset encompasses 17,380 vulnerabilities extracted from 992 repositories and spans 169 vulnerability types, covering the period from January 2006 to October 2023. Unlike prior datasets, MegaVul is designed for continuous updates, ensuring its relevance in vulnerability research. We obtained the dataset from the author's GitHub repository, [4].

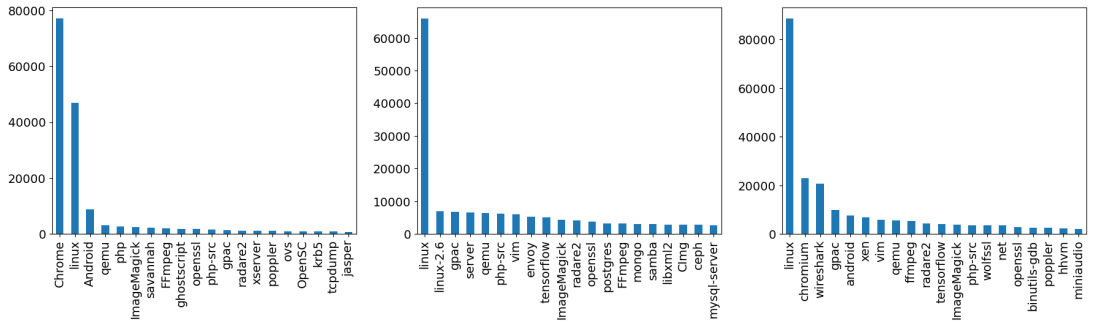
[2]https://github.com/ZeoVan/MSR_20_Code_vulnerability_CSV_Dataset

[3]https://github.com/wagner-group/diversevul

[4]https://github.com/Icyrockton/MegaVul

Table 1: Characteristics of the most used vulnerability C/C++ source code datasets compared with our merged dataset.

	Language	Origin	Functions	Vulnerable Functions	Vul/notVul ratio	Last update
BigVul	C/C++	CVE database (MITRE, 2025)	188636	10900	0.0578	2019
DiverseVul	C/C++	Security issue websites (Snyk, 2025)(RedHat, 2025)	281747	16109	0.0572	2023
Megavul	C/C++	NVD database (NIST, 2025)	353873	17975	0.0508	2023
Merged dataset	C/C++	BigVul+DiverseVul+Megavul	599991	33585	0.0559	2023



(a) BigVul (b) DiverseVul (c) MegaVul

Figure 1: Different projects distributions among all the benchmark datasets.

4.1.4 Merged Dataset

To perform comprehensive experiments, we constructed a final dataset by merging BigVul, DiverseVul, and MegaVul, followed by an additional deduplication process. This deduplication was carried out using the SHA256 hashing algorithm, which computed unique hashes for each code snippet. Duplicate entries were identified and removed by eliminating repeated hash values. The resulting dataset comprises approximately 600,000 functions, including a total of 33,585 labeled as vulnerable.

In table 1 the characteristics of the previously mentioned datasets are reported.

4.2 Model

We used a RoBERTa-like model designed for working with source code, namely CodeBERTa (HuggingFace, 2024), trained on the CodeSearchNet dataset (Husain et al., 2019) from GitHub. Since the tokenizer is optimized for code rather than natural language, it represents the data more efficiently, reducing the length of the sequences by 33% to 50% compared to standard tokenizers such as those for GPT-2 or RoBERTa. The model itself is relatively compact, with six layers and 84 million parameters, similar in size to DistilBERT. It was trained from scratch on the entire dataset for five epochs.

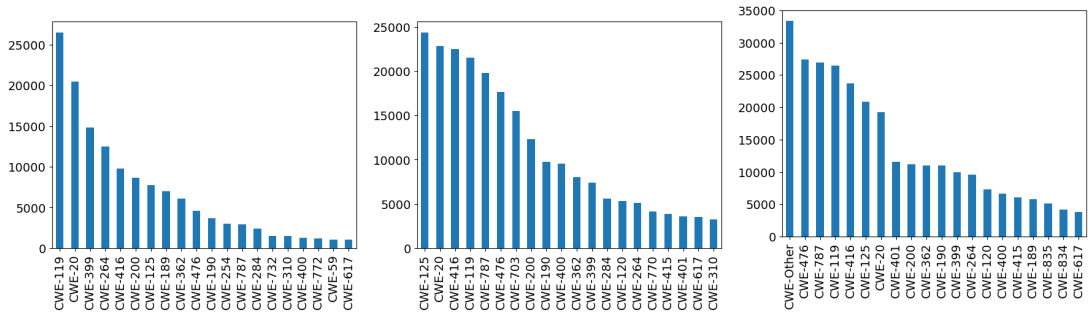
4.3 Evaluation Metrics

Considering the high imbalance in all three datasets,the Area Under the Receiver Operating Characteristic Curve (AUC) is preferred over accuracy, providing a more comprehensive and nuanced evaluation of a model's performance. Alongside AUC, precision, recall, and F1-score are used to assess specific aspects of performance.

5 EXPERIMENTS SETTING

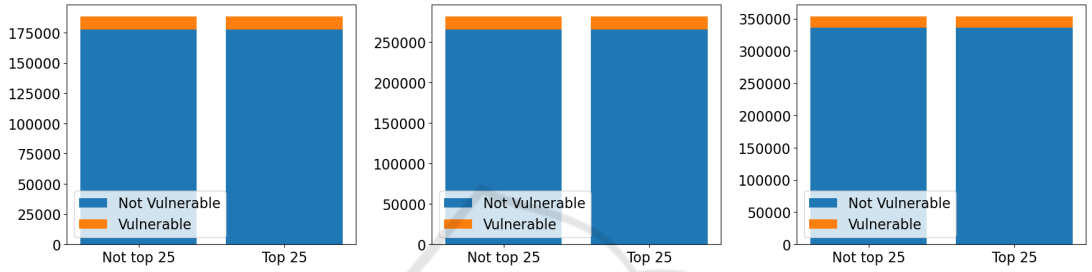
The experiments were conducted to evaluate vulnerability detection in source code as a binary classification task across various generalization scenarios:

- Random Split Benchmarking: Training, validation, and testing were performed on randomly split subsets of the dataset to establish baseline performance.
- Project Generalization: Models were trained and validated on 90% of the source code from specific projects and tested on entirely different projects not included in the training set. This setup examines the model's ability to generalize across unseen projects, as prior studies indicate performance drops in such scenarios.
- Vulnerability Type Generalization: Training and validation were conducted on functions associated with vulnerabilities outside the MITRE Top 25 most dangerous CWEs. Testing was performed on data within the Top 25, focusing on how dataset vulnerability distributions impact model performance when detecting high-priority unseen vulnerabilities.
- Cross-Dataset Evaluation: Training and validation were performed on one dataset, followed



(a) BigVul (b) DiverseVul (c) MegaVul

Figure 2: Different CWE IDs distributions among all the benchmark datasets.



(a) BigVul (b) DiverseVul (c) MegaVul

Figure 3: Representation of the number of functions belonging to the MITRE top 25 most dangerous CWEs.

by testing on a combined dataset from the other datasets included in the study.

These settings assess model performance under diverse and realistic conditions, emphasizing generalization across projects, vulnerability types, and datasets. For the hyperparameters, we choose $2 * 10^{-5}$ for the learning rate, with AdamW as the optimizer and 512 sequence length for every model. The training is performed on an NVIDIA RTX A6000 GPU.

6 RESULTS AND DISCUSSION

The experimental results obtained from the three state-of-the-art datasets are presented in Table 2. Table 3 summarizes the performance achieved on the merged dataset across the three analyzed scenarios. Finally, Table 4 illustrates the model's generalization capability when trained on the merged dataset and evaluated on previously unseen data from the three original datasets.

6.1 The Good

CodeBERTa has shown excellent performance under favorable conditions, particularly when trained and tested on random splits of the same dataset. This scenario, often used as a baseline, allows the model to benefit from shared patterns and characteristics within the data. For example, on the BigVul dataset, the model achieved an impressive AUC of 92.24, coupled with a Precision of 81.22 and an F1-score of 83.39. These results demonstrate the model's capacity to effectively detect vulnerabilities when the distribution of the training and test data is consistent. Even on a more challenging dataset like Megavul, where diversity and noise are higher, the model maintained solid performance, with an AUC of 71.38 and an F1-score of 49.81, showing its robustness under less ideal conditions.

Beyond single-dataset experiments, combining the three datasets (BigVul, DiverseVul, and Megavul) into a unified training set further enhanced the model's performance, particularly on datasets that initially posed challenges. The DiverseVul dataset is a notable example: by merging data and removing duplicates, the model's F1-score increased by 6.18 points, rising from 28.5 to 34.68. This improvement highlights the potential benefits of addressing data sparsity and increasing the diversity of training samples. Combining datasets, therefore, emerges as a promising strategy for handling underrepresented vulnerabilities and improving overall model robustness.

Table 2: CodeBERTa performance on the different tested use cases. The values between parenthesis indicate the metrics' differences with respect to the "Random Splits" case, that is the case when the model is tested on a random split of the base dataset.

	AUC	Precision	Recall	F1-score	FPR	FNR
Random Splits						
BigVul	92,24	81,22	85,69	83,39	1,22	14,31
DiverseVul	66,28	21,65	41,71	28,5	9,16	58,29
Megavul	71,38	56,4	44,61	49,81	1,85	55,39
Unseen Projects						
BigVul	86(-6,24)	88,7(+7,48)	72,65(-13,04)	79,88(-3,51)	0,64(-0,58)	27,35(+13,04)
DiverseVul	53,39(-12,67)	15,71(-9,02)	10,79(-27,32)	12,79(-17,21)	4,01(-1,99)	8,35(-53,54)
Megavul	58,03(-13,35)	39,46(-16,94)	18,54(-26,07)	25,22(-24,59)	2,47(+0,62)	81,46(+26,07)
Unseen CWEs						
BigVul	88,45(-3,79)	84,4(+3,18)	77,86(-7,83)	81(-2,39)	0,97(-0,25)	22,14(+7,83)
DiverseVul	56.59(-9,69)	17,44(-4,21)	18,67(-24,04)	18,03(-10,47)	5,5(-3,66)	81,33(+23,04)
Megavul	59,4(-11,98)	29,68(-26,72)	21,77(-22,84)	25,12(-24,69)	2,96(+1,11)	78,23(+22,84)
Unseen Datasets						
BigVul	53,94(-38,3)	39,47(-41,75)	8,63(-77,06)	14,16(-69,23)	0,75(-0,47)	91,37(+77,06)
DiverseVul	69,23(+2,95)	53,85(+32,2)	40,42(-1,29)	46,18(+17,68)	1,95(-7,21)	59,58(-7,21)
Megavul	72,58(+1,2)	30,75(-25,65)	52,35(+7,74)	38,75(-11,06)	7,18(+5,33)	47,65(-7,74)

Table 3: CodeBERTa performance when trained with all the data on three test cases.

	AUC	Precision	Recall	F1-score	FPR	FNR
Standard splits	80,16	46,12	64,8	53,89	4,49	35,2
Unseen Projects	58,9(-21,26)	35,1(-11,02)	20,84(-43,96)	26,16(-27,73)	3,03(-1,46)	79,16(+43,96)
Unseen CWEs	66,18(-13,98)	32,18(-13,94)	37,15(-27,65)	34,48(-19,41)	4,8(+0,31)	62,85(+27,65)

6.2 The Bad

Despite these strengths, the model exhibited significant weaknesses when tested on unseen projects. These tests aimed to evaluate CodeBERTa's ability to generalize beyond the specific patterns and styles of the training data. The results, however, revealed limitations. On the BigVul dataset, the F1-score decreased by 3.51 points, dropping from 83.39 to 79.88. While the decline may seem moderate, it reflects a diminished ability to handle new coding styles or project-specific conventions. For DiverseVul, the drop was much more severe: the F1-score fell by 17.21 points, plunging from 28.5 to 12.79. This suggests that the model struggles to adapt to the unique characteristics of different projects, which can vary widely in structure and context.

Testing on vulnerabilities with unseen CWEs revealed a similar trend. Vulnerabilities in this setting often involve patterns or characteristics that the model has not encountered before. Consequently, the performance declined sharply across metrics. In the Megavul dataset, for instance, the AUC dropped to 59.4, while the F1-score decreased by 24.69 points. These findings underline the model's limited capacity to extrapolate from known vulnerabilities to new or rare ones, which poses a significant challenge for real-world applications where unknown vulnerability types frequently arise.

6.3 The Ugly

The most critical shortcomings of CodeBERTa's performance were exposed in cross-dataset testing, where the model was trained on one dataset and tested on entirely unseen datasets. This scenario simulates the real-world challenge of deploying a vulnerability detection system trained on one context to operate effectively in a completely different one. The results were stark. When trained on the combined dataset and tested on BigVul, the F1-score plummeted by 69.23 points, dropping from 83.39 to a dismal 14.16. Similarly, for DiverseVul, although some metrics showed minor improvements, key indicators like Recall remained critically low, indicating that the model could not adequately identify vulnerabilities in this setting.

These results point to a deeper issue: the strong influence of dataset-specific biases. Each dataset carries unique characteristics — ranging from labeling strategies to the types of vulnerabilities represented — that

the model internalizes during training. While this specificity can boost performance within a dataset, it severely limits generalization across datasets. For instance, while the combined dataset approach improved the model's performance on DiverseVul, it had only marginal benefits—or even detrimental effects—for other datasets like Megavul, where the F1-score saw a slight decline.

This variability underscores the challenges of harmonizing diverse datasets, which often suffer from inconsistencies, noise, and differing definitions of what constitutes a vulnerability. The poor cross-dataset performance also raises concerns about real-world applicability, where systems must handle diverse and unseen data without the luxury of retraining or fine-tuning on every new context.

Table 4: CodeBERTa results when tested on unseen data from the three datasets.

Train/Val Dataset	Test Dataset	Previous F1-score	New F1-score	Diff.
Big+Diverse+Mega	BigVul	83.39	83.11	-0.28
Big+Diverse+Mega	DiverseVul	28.5	34.68	+6.18
Big+Diverse+Mega	MegaVul	49.81	49.43	-0.38

6.4 Discussion

Overall, the evaluation of CodeBERTa reveals that the model excels in scenarios where training and testing datasets share similar characteristics, particularly in terms of the same project or similar vulnerability distributions. However, significant challenges arise when faced with unseen projects, vulnerability types, or datasets with varied distributions. The results emphasize the importance of high-quality, diverse datasets to improve the generalizability of deep learning models in vulnerability detection tasks.

While combining datasets leads to improved performance, the limitations observed in cross-dataset testing highlight the need for more sophisticated data aggregation techniques. Current methods, which primarily focus on concatenation and deduplication, fail to fully address inconsistencies in labeling, representation, and the types of vulnerabilities covered. Techniques such as data augmentation, domain adaptation, and representation learning could help harmonize diverse datasets and reduce dataset-specific biases. Such techniques are critical for developing models capable of performing robustly in real-world applications, where the data are heterogeneous, noisy, and constantly evolving. Future research should focus on integrating these approaches into the data preparation pipeline to enhance the generalization capabilities of transformer-based models in vulnerability detection.

7 CONCLUSIONS

This study assessed the impact of dataset quality and diversity on machine learning models for vulnerability detection in source code. By evaluating model performance across unseen projects, new vulnerability types (CWEs), and diverse dataset distributions, we demonstrated that data quality is crucial for the reliability and generalization of these models. The results show that while high-quality datasets enable strong performance in controlled environments, significant declines occur in generalization scenarios, especially when models encounter new projects or vulnerabilities. This highlights the limitations of current datasets, which lack the diversity needed to represent real-world complexity. A key finding is that aggregating datasets enhances model performance. Combining multiple real-world datasets and removing duplicates improved generalization, showing that diverse data distributions mitigate biases. However, simply increasing dataset size is not enough—varied and representative examples are crucial to adapting to different coding styles and vulnerability patterns. Challenges such as imbalanced vulnerability types, inaccurate labels, and limited real-world variability persist. Overcoming these issues requires better dataset curation, improved labeling, and the inclusion of a broader range of vulnerabilities. While synthetic datasets can be useful, they must be designed to reflect real-world complexities. Ultimately, the future of automated vulnerability detection depends more on improving data quality than refining model architectures. Focusing on diverse, accurate, and representative datasets will lead to more reliable and generalizable solutions, advancing software security.

REFERENCES

Chakraborty, P., Arumugam, K. K., Alfadel, M., Nagappan, M., and McIntosh, S. (2024). Revisiting the Performance of Deep Learning-Based Vulnerability Detection on Realistic Datasets. 50(8):2163–2177.

Chakraborty, S., Ahmed, T., Ding, Y., Devanbu, P. T., and Ray, B. (2022a). NatGen: Generative pre-training by "naturalizing" source code. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 18–30. ACM.

Chakraborty, S., Krishna, R., Ding, Y., and Ray, B. (2022b). Deep learning based vulnerability detection : Are we there yet ? *IEEE Transactions on Software Engineering*, 48(9):3280–3296.

Chen, Y., Ding, Z., Alowain, L., Chen, X., and Wagner, D. (2023). DiverseVul: A New Vulnerable Source Code Dataset for Deep Learning Based Vulnerability Detec-

tion. In *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*, pages 654–668. ACM.

Cheng, X., Wang, H., Hua, J., Xu, G., and Sui, Y. (2021). DeepWukong: Statically Detecting Software Vulnerabilities Using Deep Graph Neural Network. 30(3):38:1–38:33.

Croft, R., Babar, M. A., and Kholoosi, M. M. (2023). Data Quality for Software Vulnerability Datasets. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 121–133.

Curto, C., Giordano, D., Indelicato, D. G., and Patatu, V. (2024a). Can a Llama Be a Watchdog? Exploring Llama 3 and Code Llama for Static Application Security Testing. In *2024 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 395–400.

Curto, C., Giordano, D., Palazzo, S., and Indelicato, D. (2024b). MultiVD: A Transformer-based Multitask Approach for Software Vulnerability Detection. In *Proceedings of the 21st International Conference on Security and Cryptography*, pages 416–423. SCITEPRESS - Science and Technology Publications.

Fan, J., Li, Y., Wang, S., and Nguyen, T. N. (2020). A c / c ++ code vulnerability dataset with code changes and cve summaries. In *IEEE/ACM 17th International Conference on Mining Software Repositories (MSR)*, pages 508–512. ACM.

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. *Findings of EMNLP*.

Fu, M. and Tantithamthavorn, C. (2022). Linevul: A transformer-based line-level vulnerability prediction. In *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*. IEEE.

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. (2020). GraphCodeBERT: Pre-training Code Representations with Data Flow.

Hin, D., Kan, A., Chen, H., and Babar, M. A. (2022). LineVD: Statement-level vulnerability detection using graph neural networks. In *Proceedings of the 19th International Conference on Mining Software Repositories*, pages 596–607. ACM.

HuggingFace (2024). CodeBERTa. https://huggingface.co/huggingface/CodeBERTa-small-v1. accessed: 2024-09.

Husain, H., Wu, H.-H., Gazit, T., Allamanis, M., and Brockschmidt, M. (2019). CodeSearchNet Challenge: Evaluating the State of Semantic Code Search. *arXiv:1909.09436 [cs, stat]*. arXiv: 1909.09436.

Jiang, X., Wu, L., Sun, S., Li, J., Xue, J., Wang, Y., Wu, T., and Liu, M. (2025). Investigating Large Language Models for Code Vulnerability Detection: An Experimental Study.

Mamede, C., Pinconschi, E., Abreu, R., and Campos, J. (2022). Exploring transformers for multi-label classification of java vulnerabilities. In IEEE, editor, *2022 IEEE 22nd International Conference on Software Quality , Reliability and Security (QRS)*, pages 43–52.

MITRE (2025). CVE published by year. https://www.cvedetails.com/browse-by-date.php. Accessed: 2025-01.

National Institute of Standards and Technology (2025). Nist software assurance reference dataset. https://samate.nist.gov/SARD. Accessed: 2025-01.

Nguyen, V.-A., Nguyen, D. Q., Nguyen, V., Le, T., Tran, Q. H., and Phung, D. (2022). ReGVD: Revisiting graph neural networks for vulnerability detection. In *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings*, ICSE '22, pages 178–182. Association for Computing Machinery.

Ni, C., Shen, L., Yang, X., Zhu, Y., and Wang, S. (2024). MegaVul: A C/C++ Vulnerability Dataset with Comprehensive Code Representations. In *2024 IEEE/ACM 21st International Conference on Mining Software Repositories (MSR)*, pages 738–742.

NIST (2025). NVD database. https://nvd.nist.gov/. Accessed: 2025-01.

RedHat (2025). Red Hat Bugzilla website. https://bugzilla.redhat.com/. Accessed: 2025-01.

Rozière, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X. E., Adi, Y., Liu, J., Remez, T., Rapin, J., Kozhevnikov, A., Evtimov, I., Bitton, J., Bhatt, M., Ferrer, C. C., Grattafiori, A., Xiong, W., Défossez, A., Copet, J., Azhar, F., Touvron, H., Martin, L., Usunier, N., Scialom, T., and Synnaeve, G. (2023). Code Llama: Open Foundation Models for Code.

Snyk (2025). Snyk website. https://snyk.io/. Accessed: 2025-01.

Xu, F. F., Alon, U., Neubig, G., and Hellendoorn, V. J. (2022). A systematic evaluation of large language models of code. In *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming*, MAPS 2022, pages 1–10. Association for Computing Machinery.

Zheng, Y., Pujar, S., Lewis, B., Buratti, L., Epstein, E., Yang, B., Laredo, J., Morari, A., and Su, Z. (2021). D2A: A Dataset Built for AI-Based Vulnerability Detection Methods Using Differential Analysis. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 111–120.

Zhou, Y., Liu, S., Siow, J., Du, X., and Liu, Y. (2019). Devign: Effective Vulnerability Identification by Learning Comprehensive Program Semantics via Graph Neural Networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.