

Research Paper Review - Vullibgen

Gerald T. Rigdon

March 29, 2025

The paper "*Vullibgen: Generating Names of Vulnerability Affected Packages via a Large Language Model*" (Chen T., Li L., Zhu L., Li, Z., Liu X., Liang G., Wang Q., Xie, T., 2023) addresses the critical task of identifying software packages affected by cybersecurity vulnerabilities with the use of Large Language Models (LLMs). In the present cyber landscape, security practitioners rely on databases such as GitHub Advisory to track vulnerabilities and their associated affected software packages. Given the continuous increase in cyber-attacks and vulnerability disclosures, the ability to quickly identify affected software packages is an important factor in effective vulnerability management and mitigation response. To that end, this paper introduces a novel LLM framework, VulLibGen, which generates the names of affected software packages rather than employing the common sub-optimal rank/retrieval method.

Research Problem

The research focuses on the challenge of accurately identifying software packages that are affected by vulnerabilities and highlights both the accuracy and scalability limitations of conventional approaches involving Named Entity Recognition (NER). The proposed solution uses Vicuna-13B, an LLM that is part of the VulLibGen framework and trained or fine-tuned to generate the names of vulnerability-affected software packages directly from vulnerability descriptions. The paper demonstrates that this method enhances the accuracy of identification across various programming languages. Moreover, the practical use of the framework is further evidenced by the acceptance of its generated names in real-world environments such as GitHub Advisory. Hence, VulLibGen offers a viable solution to the critical task of vulnerability-affected software package identification by providing a tool that facilitates more accurate and scalable security practices.

Research Questions

Some research questions that can be derived from the paper's content include:

- Can LLMs be used to reliably generate the names of affected software packages directly from vulnerability descriptions?

- How does the VulLibGen generative approach compare to traditional rank/retrieval-based methods with respect to key metrics?
- What are the advantages of using a fine-tuned LLM like Vicuna-13B for generating vulnerability-affected software package names?
- How does VulLibGen perform across different programming languages and their related software packaging?
- What are the practical implications and acceptability of the generated software package names by security practitioners and databases such as GitHub Advisory?

Hypothesis and Methodology

While there is no stated hypothesis, the goal was to answer the research questions, primarily focused on the comparison of the VulLibGen framework with the existing approaches. In contrast to the rank/retrieval competition, the methodology revolves around the following three core elements which are listed in order of importance in the framework.

1. Supervised Fine-Tuning: The core of the VulLibGen framework is the Vicuna-13B LLM which was fine-tuned using supervised learning techniques, specifically tailored to the task of generating names of vulnerability-affected packages. The dataset used for fine-tuning consists of vulnerability descriptions and their corresponding affected software package names. The training process involved adjusting the model parameters to minimize the difference between the generated package names and the ground truth package names.

2. Retrieval Augmented Generation: This enhancement was integrated into the VulLibGen framework to leverage external knowledge sources during the generation of vulnerable package names. This was to address the knowledge gaps present in base LLMs, particularly in less common programming languages and newer software package ecosystems.

3. Local Search: This enhancement was employed to address the challenge of generating accurate and existing software package names from the LLM output. It refined the initial LLM predictions by matching them with the closest existing package names.

Most importantly, this methodology revolves around a unique approach that directly generates the affected software package names instead of retrieving them from existing database lists.

Dataset and Sampling

The GitHub Advisory was used to procure a dataset to provide authoritative records of vulnerabilities and the software packages they impact. This dataset was manually verified by security experts and contains 2789 Java, 3193 JavaScript, 2237 Python, and 1351 Go language vulnerabilities. The dataset is essentially comprised of vulnerability descriptions or textual descriptions of various vulnerabilities which detail the nature, cause, and potential impact of the security issues, along with the associated software package names.

This dataset was used to fine-tune the Vicuna-13B LLM through supervised learning to optimize its performance in generating accurate package names from the given vulnerability descriptions. The evaluation metrics used to assess its effectiveness include Accuracy@1, Recall@1, and F1@1. Typically, the metrics are as follows:

TP = True Positive (Predicted positive and is positive)
TN = True Negative (Predicted negative and is negative)
FP = False Positive (Predicted positive but is negative)
FN = False Negative (Predicted negative but is positive)

Accuracy = $\frac{TP + TN}{TP + TN + FP + FN}$
Recall = $\frac{TP}{TP + FN}$
Precision = $\frac{TP}{TP + FP}$
F1 = $(\text{Recall} * \text{Precision} / (\text{Recall} + \text{Precision})) * 2$

However, in this research context, Accuracy@1 is often referred to simply as top-1 accuracy and is a specific form of accuracy used to evaluate the performance of a LLM based on its top-ranked prediction. Recall@1 is a performance metric used to evaluate the effectiveness of LLM predictions and measures the proportion of times the top-ranked prediction is relevant among all relevant instances. The F1@1 score provides a single metric that accounts for both false positives and false negatives and offers a more comprehensive evaluation of LLM performance at the first prediction position.

Ground Truth, Analysis and Results

The ground truth used for training and evaluation consisted of the actual names of software packages that are known to be affected by specific vulnerabilities. The ground truth data provided a benchmark against which the LLM performance was measured by comparing the names generated by VulLibGen to the ground truth package names. Furthermore the analysis involved comparing the performance of VulLibGen with and without various enhancements including retrieval augmented generation and local search techniques.

The paper documents several metrics outcomes in multiple tables where the general summary is:

- VulLibGen has an average accuracy of 0.806 for identifying vulnerable packages in the four most popular ecosystems in GitHub Advisory (Java, JavaScript, Python, Go) while the best average accuracy in previous work is 0.721.
- The supervised fine-tuning on the Vicuna-13B LLM has the best Recall@1 and F1@1 results, even outperforming the larger ChatGPT and GPT4 models on all datasets besides Java.
- The enhanced retrieval augmented generation technique demonstrated improvement in Accuracy@1 across different programming languages, specifically: Java 9.3%, JavaScript 1.8%, Python 8.9%, and Go 15.7%.
- The local search enhancement demonstrated improvement in Accuracy@1 across different programming languages, specifically: Java 3.4%, JavaScript 1.0%, Python 1.4%, and Go 6.2%.

The success of VulLibGen can be attributed to several factors. The novel approach of generating software package names leverages the full contextual understanding provided by LLMs. Additionally, fine-tuning the Vicuna-13B LLM for the specific task of vulnerability-affected package identification enhances its accuracy and relevance. Further this generative approach to software name production scales better with larger datasets and more complex vulnerability descriptions while specifically addressing the performance limitations of rank/retrieval-based methods.

Limitation and Gaps

The effectiveness of the VulLibGen framework is highly dependent on the quality and comprehensiveness of the training data. Any gaps or inaccuracies in the vulnerability descriptions or affected package names within the training dataset could potentially affect performance. If the dataset does not adequately cover the diversity of real-world scenarios, the LLM may struggle with less common or more complex vulnerabilities. Moreover, any static LLM solution is likely to suffer long-term reliability given the dynamic nature of software vulnerability disclosures. The paper does not address how VulLibGen would adapt to such newly discovered vulnerabilities that are not present in the existing training data. This would likely require continuous updates and retraining of the LLM to maintain its effectiveness over time. Finally, the research only validates the generated package names through acceptance in GitHub Advisory. It would be a much stronger proposal if it presented a broader validation across other security databases and real-world environments. This would provide a more comprehensive understanding of the framework's practical utility and highlight any potential limitations for different contexts in advance.

Research Extension Possibilities

The paper suggests several avenues for future research such as further fine-tuning and training on larger and more diverse datasets, integration of VulLibGen with other security tools and databases, and exploring other LLMs to compare their efficacy within this generative framework approach. There are also some adjacent opportunities not discussed in the research. For example, Agentic AI applications could be used to continuously monitor software repositories and systems for new vulnerabilities. This could be used as an input source for VulLibGen or expand into its own unique solution space. One could easily imagine Agentic AI systems that autonomously download and deploy software patches or analyze historical data to predict more likely future threats. Finally, given the generative focus of VulLibGen, another possible research endeavor would be to revisit the opposing retrieval methods using more recent advances in this area of research in tandem with the Agentic AI methodology.

Conclusion

Overall, the paper presents an advancement in the field of vulnerability-affected software package identification. By leveraging the capabilities of LLMs and adopting a unique generative approach, VulLibGen achieves higher accuracy and practical applicability compared to existing methods. The framework's success in real-world validations, as evidenced by the acceptance of generated names in GitHub Advisory, demonstrates its potential for broader industry adoption as a tool to enhance vulnerability management and mitigation processes.

Marymount Honor Pledge

I agree to uphold the principles of honor set forth by this community in the Marymount University mission statement and the Academic Integrity Code and Community Conduct Code, to defend these principles against abuse or misuse, and to abide by the regulations of Marymount University.

References

- Chen T., Li L., Zhu L., Li, Z., Liu X., Liang G., Wang Q., Xie, T. (2023). *Vullibgen: Generating Names of Vulnerability Affected Packages via a Large Language Model*. arxiv.org. <https://arxiv.org/abs/2308.04662>
- Ahmad B., Tan B., Karri R., Pearce H. (2023) *Flag: finding line anomalies (in code) with generative AI*.org. <https://arxiv.org/abs/2306.12643>[Links to an external site.](#)

Alam MT., Bhusal D., Nguyen L., Rastogi N. (2024) *CTIBench: a benchmark for evaluating LLMs in cyber threat intelligence*.org. In: The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track. <https://arxiv.org/abs/2406.07599>[Links to an external site.](#)

Bhusal D., Alam MT., Nguyen L., et al (2024) *Secure: benchmarking generative large language models for cybersecurity advisory*.
harvard.edu. <https://ui.adsabs.harvard.edu/abs/2024arXiv240520441B/abstract>[Links to an external site.](#)

Fayyazi R., Yang SJ. (2023). *On the uses of large language models to interpret ambiguous cyberattack descriptions*. org. <https://arxiv.org/abs/2306.14062>[Links to an external site.](#)