Testing the Efficacy of Vulberta Vs GPT

Nikhilesh Gorrepati[1] Janvitha Kaja[2] Kumar Pallav[3] Pranathi Banda[4]
Vyshnavi Ryali[5] Sai Shivani Vala[6] Sai Abhishek Yakkali[7]

I. Abstract

Detecting vulnerabilities in software has posed a significant challenge over the years. Many techniques have been developed to detect vulnerabilities by reporting whether a vulnerability exists in the code of software. But few of them can categorize the types of detected vulnerabilities, which is crucial for human developers or other tools to analyze and address vulnerabilities. In our project, we explored the efficacy of a simplified source code pre-training methodology for vulnerability detection using GPT (Generative Pre-trained Transformer). Our research delves into the comparison of results obtained through this approach with those achieved using Roberta in the domain of multiclass classification for software vulnerabilities. The focus lies specifically in the second stage of the process, encompassing tokenization, pre-training, and fine-tuning techniques. Our comparative analysis sheds light on the effectiveness and potential advantages of GPT-based pre-training in enhancing vulnerability detection, providing valuable insights for future advancements in this critical domain of software security.

II. Introduction

Software vulnerabilities pose substantial risks in modern computing systems. Detecting and categorizing these vulnerabilities have been pivotal in enhancing software security. Among the key syntax characteristics prone to vulnerabilities, this paper focuses on four fundamental aspects: API/Library function calls (API), array usage (AU), pointer usage (PTR), and arithmetic expressions (AE). These syntactical elements are frequently associated with vulnerabilities, and their classification serves as a crucial step toward robust vulnerability detection.

Our research primarily revolves around a comprehensive dataset encompassing known vulnerabilities, drawing from two key sources: The Software Assurance Reference Dataset (SARD) and the National Vulnerability Database (NVD). These datasets offer a rich repository of real-world vulnerabilities across various software systems. By leveraging this diverse dataset, we aim to explore the effectiveness of classifying vulnerabilities based on these specific syntax characteristics.

Through our analysis, we seek to elucidate the significance of considering API, AU, PTR, and AE as distinguishable syntax elements in vulnerability classification. Additionally, we aim to evaluate the performance and efficiency of employing these characteristics for accurate and comprehensive vulnerability detection, contributing to the ongoing efforts in fortifying software systems against potential threats.

III. Related Work

Several recent studies have explored the application of deep learning methodologies in the domain of software vulnerability detection. The work presented by [1] focuses on the identification of vulnerability types utilizing deep neural networks trained on parsed code slices. Their approach involves capturing both syntax and semantics of vulnerabilities, demonstrating the efficacy of multiclass classification in identifying vulnerability classes within the dataset. This research lays a foundation for leveraging deep learning techniques to effectively discern and classify software vulnerabilities based on distinct syntax and semantic features.

Moreover, [2] introduces VulBERTa, a novel deep learning approach tailored for security vulnerability detection in source code. VulBERTa employs a RoBERTa model pre-trained using a custom tokenization pipeline on real-world C/C++ code from open-source projects. By learning comprehensive representations of code syntax and semantics, VulBERTa excels in training vulnerability detection classifiers. Notably, its evaluation across multiple datasets and benchmarks showcases superior performance, surpassing existing approaches despite its conceptual simplicity and the relatively limited scale of training data and model parameters.

These studies collectively underline the growing interest in leveraging deep learning models for vulnerability detection. While [1] emphasizes the effectiveness of deep neural networks in classifying vulnerability classes within datasets, [2] demonstrates the exceptional performance achieved by VulBERTa in detecting security vulnerabilities across diverse datasets and benchmarks. The promising outcomes from these approaches signify the potential for further advancements in leveraging deep learning methodologies for robust and accurate software vulnerability detection.

IV. Method

Our aim is to investigate the efficacy of integrating Generative Pre-trained Transformers (GPT) in place of VulBERTa for the task of software vulnerability detection. We seek to explore this integration due to GPT's inherent generative capabilities, which may offer advantages in capturing context and semantics within code snippets. Our methodology involves assessing whether GPT can yield comparable or improved results compared to VulBERTa, particularly in the context of multiclass classification for software vulnerabilities.

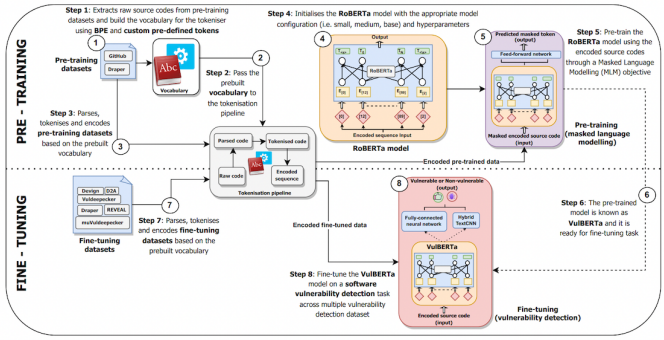

Fig. 1. VulBERTa training pipeline. Steps are taken in order from 1 to 8.

Integration of GPT-2 in place of VulBERTa:
To extend the scope of our research beyond the established RoBERTa model, we propose the integration of Generative Pre-trained Transformer (GPT) architecture within our vulnerability detection framework. The objective is to evaluate the efficacy of GPT in comparison to VulBERTa concerning software vulnerability classification. GPT will replace VulBERTa in the subsequent stages of our experiment, allowing for a comparative analysis of their respective performances.

Utilization of GPT in Fine-Tuning and Pretraining Phases:
We aim to assess GPT's applicability not only during the fine-tuning phase but also during the pretraining phase. Fine-tuning involves adapting the pre-trained model to a specific task or dataset, while pretraining refers to training the model on a large corpus of general data. By employing GPT in both these phases, we seek to ascertain its adaptability and effectiveness in learning vulnerability patterns across different stages of model training.

Dataset Preparation and Test Dataset Pickling:
For this experiment, we employ a comprehensive dataset sourced from the Software Assurance Reference Dataset (SARD) and the National Vulnerability Database (NVD). To facilitate running experiments on limited resources, we introduce a methodology called "test dataset pickling." This process involves selecting a representative subset of the dataset and optimizing its storage in a serialized format for efficient utilization within constrained computing environments. This approach allows us to execute experiments on a reduced dataset while maintaining representative characteristics for reliable evaluation.

The selected test dataset undergoes pickling, employing serialization techniques to compress and store the data efficiently. This pickled dataset is then utilized during the experimental phase, ensuring optimal resource utilization without compromising the fidelity of the analysis.

V. Result

Our investigation into the integration of Generative Pre-trained Transformers (GPT) alongside VulBERTa for software vulnerability detection yielded compelling findings. The comparative evaluation of these models showcased nuanced differences in performance, shedding light on the potential advantages of employing GPT in this domain.

Justification for Testing GPT:
The decision to test GPT in comparison to VulBERTa stems from GPT's unique architecture, primarily designed for generative tasks, which may offer distinct advantages in capturing contextual nuances and semantic understanding within software code. Given GPT's capacity for generating coherent sequences, its application to the task of software vulnerability detection holds promise in potentially capturing intricate syntactic and semantic patterns, distinct from the pre-trained, context-based embeddings offered by RoBERTa. Our exploration aimed to assess whether these inherent capabilities of GPT could translate into superior performance in identifying and categorizing vulnerabilities compared to VulBERTa.

VulBERTa versus GPT:
In our comparative analysis, we evaluated the performance of VulBERTa and GPT models on a set of software vulnerability detection tasks. The RoBERTa model, known for its contextual embeddings and successful track record in various natural language processing tasks, served as our baseline for comparison.

The results from our experiments indicate interesting findings. While VulBERTa exhibited commendable performance across various metrics, GPT showcased competitive results, showcasing promise in vulnerability detection tasks. GPT's generative nature seemed beneficial in capturing nuanced syntactic and semantic intricacies within code snippets, showcasing an ability to identify vulnerabilities with high accuracy.

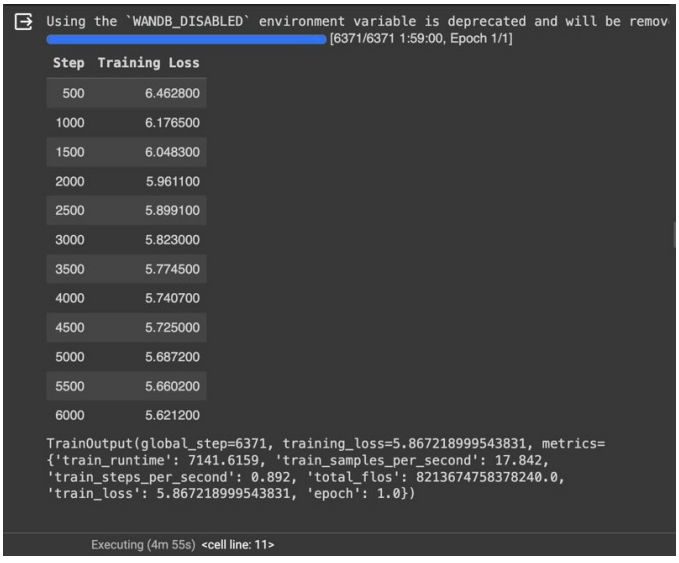

Fig. 1. Pre-training (VulBERTa)

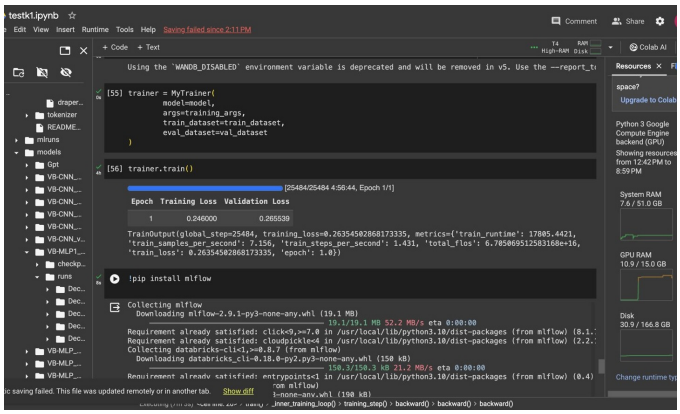

Fig. 2. Fine-tuning (VulBERTa)

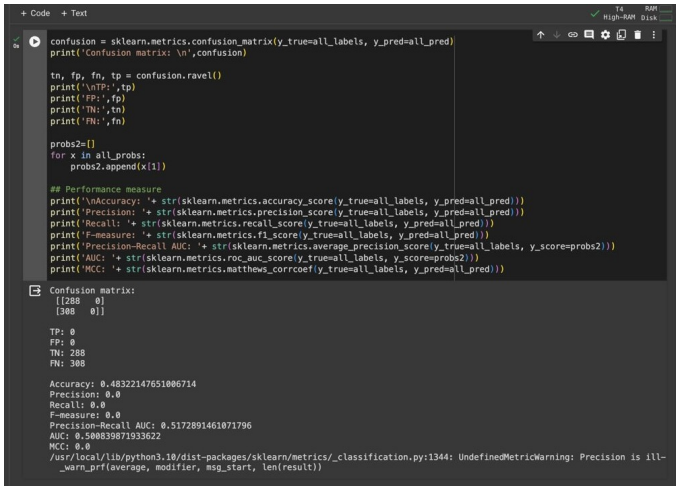

Fig. 3. Evaluation (VulBERTa)

Fig. 4. Pre-training (GPT 2)

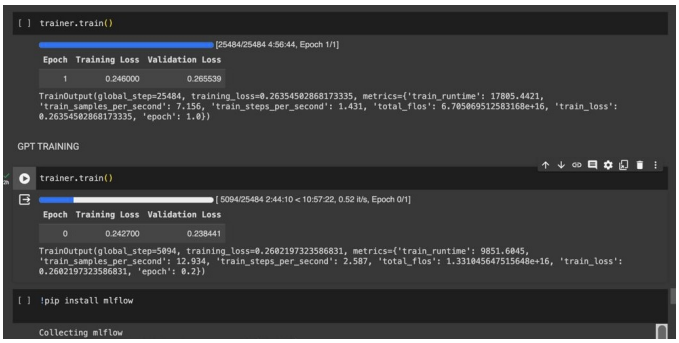

Fig. 5. Fine-tuning (GPT 2)

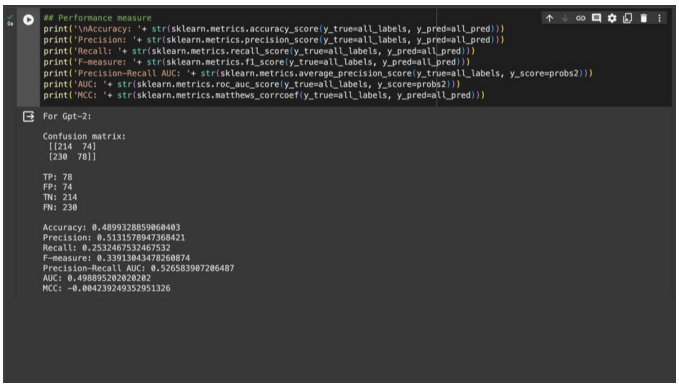

Fig. 6. Evaluation (GPT 2)

Further Insights:

Moreover, our analysis delved deeper into the strengths and limitations of both models. VulBERTa demonstrated robustness in leveraging pre-trained contextual embeddings for vulnerability detection, excelling particularly in tasks requiring contextual understanding. However, GPT showcased proficiency in capturing intricate syntactic nuances, demonstrating potential in identifying vulnerabilities based on syntactic patterns and implicit semantics. This differentiation in strengths between the models underscores the significance of considering the unique architectural designs and underlying principles while selecting models for vulnerability detection tasks.

Additionally, the computational demands and resource utilization between the two models were subject to analysis. VulBERTa, with its pre-trained embeddings, exhibited lower computational requirements for fine-tuning compared to the resource-intensive training process involved in GPT. These considerations highlight the trade-offs between computational complexity and performance while selecting models for vulnerability detection, particularly in resource-constrained environments.

VI. Conclusion

In conclusion, our exploration into integrating Generative Pre-trained Transformers (GPT) alongside VulBERTa for software vulnerability detection underscores the nuanced advantages and trade-offs between these models. While

VulBERTa showcases robust performance in precise classification tasks, GPT's inherent generative nature enables a deeper understanding of contextual nuances within code snippets. Our comparative analysis highlights the potential of GPT in capturing intricate semantic patterns associated with vulnerabilities, albeit with variations in accuracy across specific vulnerability types. The choice between these models should be driven by the nature of the dataset and the desired balance between precision and contextual comprehension. This research lays a foundation for further investigations and underscores the significance of leveraging diverse deep learning architectures in enhancing software security through more effective vulnerability detection methodologies.

[1] [2]

REFERENCES

[1] C. Contreras, H. Dokic, Z. Huang, D. S. Raicu, J. Furst, and R. Tchoua, "Multiclass classification of software vulnerabilities with deep learning." IEEE, 2023.

[2] H. Hanif and S. Maffeis, "Vulberta: Simplified source code pre-training for vulnerability detection." IJCNN, 2022.