北京航空航天大學
BEIHANG UNIVERSITY

VuLASTE: Long Sequence Model with Abstract Syntax Tree Embedding for Vulnerability Detection

Botong Zhu, Huobin Tan

Beihang University, School of Software, China

INTRODUCTION

- Vulnerability detection is the process of identifying and locating vulnerabilities, which are weaknesses or security flaws in software that can be exploited by attackers to gain unauthorized access or perform malicious actions.
- With natural language models, vulnerability detection in source code can be regarded as a text classification task.
- We proposed VuLASTE (Vulnerability detection Long sequence model with Abstract Syntax Tree Embedding), a deep-learning model for vulnerability detection.

DATA

We built a new cross-language vulnerability dataset from real open-source projects.

- GitHub Advisory Database (GHSA) as data source, generated in 2022 March.
- Contains a variety of popular programming languages (C, C++, Java, Python, Go).
- Negative cases are 8601, positive cases are 2471.

Table 1: Length statistics of dataset

Length	Count
[0.0, 512.0)	6200
[512.0, 1024.0)	1616
[1024.0, 2048.0)	1560
[2048.0, 5096.0)	1180
[5096.0, inf)	516

DESIGN OF MODEL

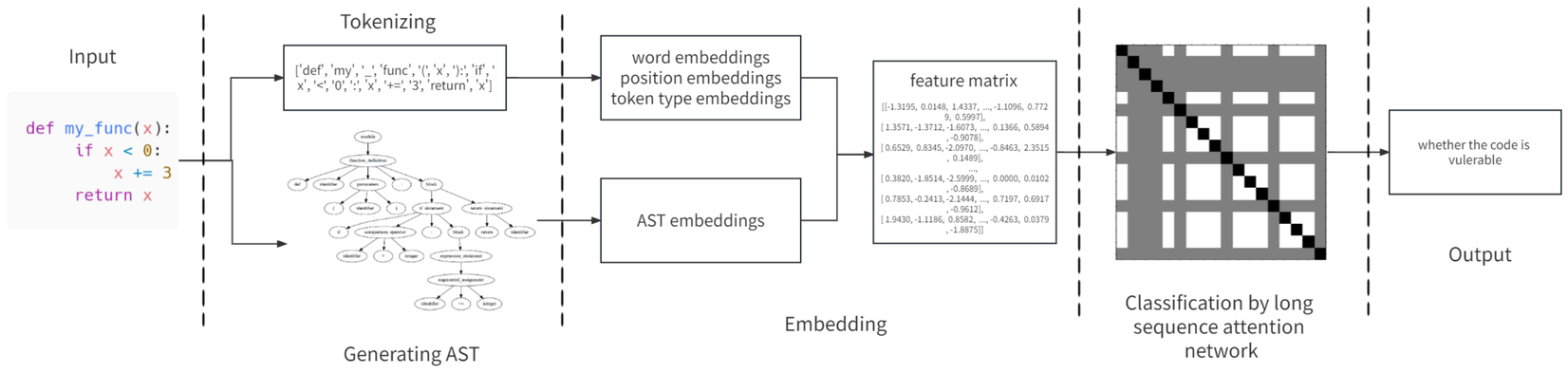

Figure 1: The general framework of VuLASTE

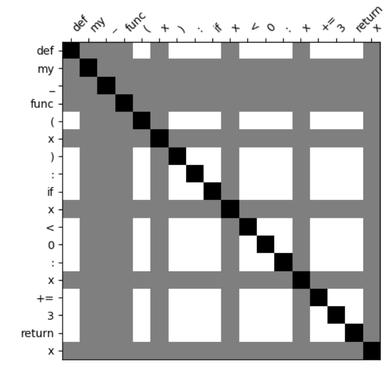

Figure 3: Long sequence attention

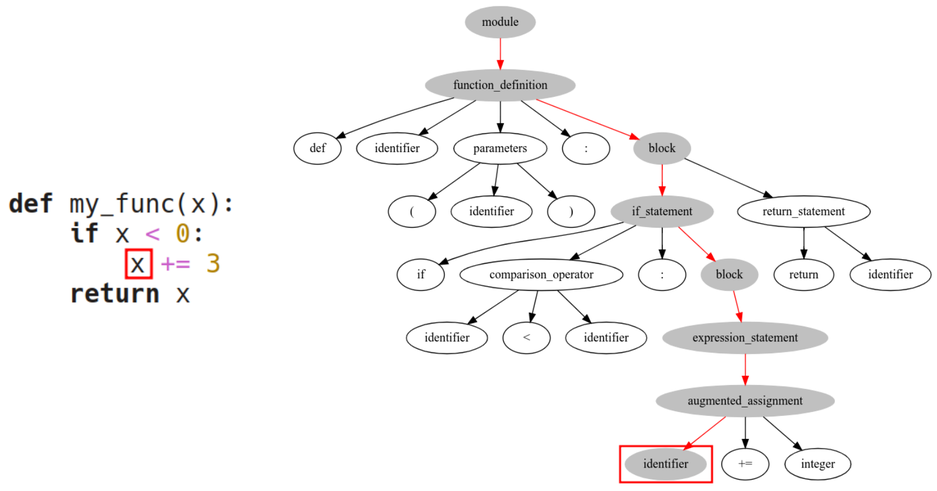

Figure 2: AST path Embedding

AST Path Embedding

Compared with natural language text, programming language text is more structured and the structural information has a deeper influence on the semantics. This paper adds ast path embedding to the model, to provide a lightweight representation of ast structure to encode the nested structural information of token in programming language text.

AST edge $\phi\left(e_{n_i,n_j}\right) =< n_i, n_j >$

AST path $W_{a,z} =< e_{a,b}, e_{b,c}, \ldots, e_{y,z} >$

AST path Embedding $AE(t) = \sum_{\substack{e \in W_{root,l} \\ n_i \in \phi(e)}} vec(n_i)$

Final Embedding $Embedding(t) = WE(t) + PE(t) + TT(t) + AE(t)$

RESULTS

Table 2: The metrics of different models

Model	hits @50	@100	@200	@500	recall	f1
VuLASTE	29	51	86	228	0.482	0.4801
VulDeePecker	6	12	33	77	0.5404	0.1941
SySeVR-BGRU	5	15	37	85	0.3826	0.241
CodeBERT	3	6	24	63	0.4463	0.4203
VUDDY	0	0	0	0	-	0

Table 3: The results of ablation study

Model	hits @50	@100	@200	@500	recall	f1
VuLASTE	29	51	86	228	0.482	0.4801
no AST	3	15	27	66	0.4983	0.3582
self attention	4	15	28	81	0.5369	0.3794
cross entropy	0	0	0	0	-	0
VuLASTE	29	51	86	228	0.482	0.4801

Metrics

We use top-k hits as the main metric of evaluating the performance of models, and recall as the second evaluation method. One reason to use top-k hits as a metric is that it can be more forgiving of false negatives. In a vulnerability detection task, it is often more important to identify as many vulnerabilities as possible, even if this means that the model may also produce some false positives.

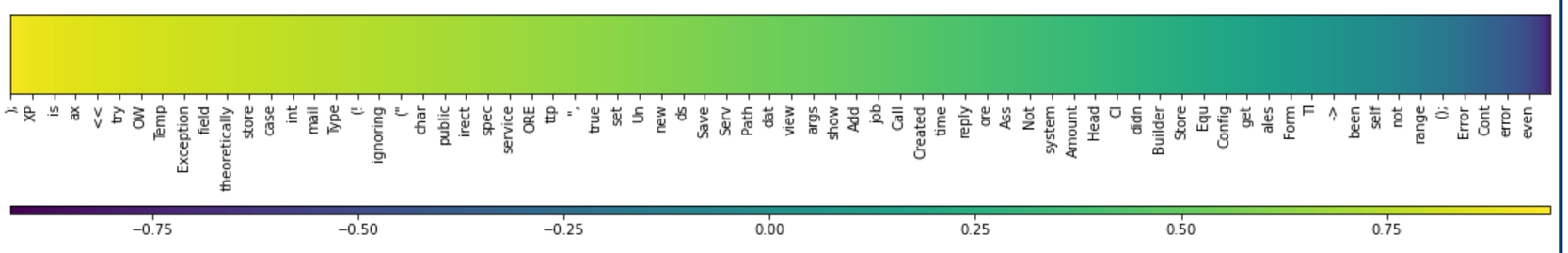

Figure 4: Heat visualization of attention weights

Attention weights visualization

To understand how the model is weighting different parts of the input code when making a prediction, attention weights are visualized. As the attention weights picture shows, the model mainly focuses on high-risk operations that may cause vulnerabilities, such as left shift and Exception handling. This result may suggest that the VuLASTE is able to identify these patterns effectively and correctly. This is a great indication that the model is working effectively and has learned to recognize the characteristics of vulnerable code.

CONCLUSION

In this paper, we proposed VuLASTE, a deep learning model to detect vulnerable codes. To deal with vocabulary explosion problem, our model use BPE algorithm in tokenizing. This model adds AST path embedding to provide a lightweight representation for programming language nesting structure. To replace the program slicing method, we use a long sequence attention mechanism from Longformer, combining global attention and windowed attention, to capture long-term semantic in source code. We also extracted a dataset from real-world open source repositories from Github Security Advisory Database. Experiment results show that comparing with existing researches, VuLASTE can better select source code pieces that may be vulnerable when candidate number is limited.

REFERENCES

1. M. Allamanis, E. T. Barr, P. Devanbu, and C. Sutton, “A survey of machine learning for big code and naturalness,” ACM Computing Surveys (CSUR), vol. 51, no. 4, pp. 1–37, 2018.
2. I. Beltagy, M. E. Peters, and A. Cohan, “Longformer: The long-document transformer,” arXiv preprint arXiv:2004.05150, 2020.
3. Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang et al., “Codebert: A pre-trained model for programming and natural languages,” in Findings of the Association for Computational Linguistics: EMNLP 2020, 2020, pp. 1536–1547.