CROSS DOMAIN VULNERABILITY DETECTION USING GRAPH CONTRASTIVE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

To overcome the difficulty of finding good-quality labeled data in domains such as vulnerability detection, Self–Supervised Learning (SSL) methods such as Contrastive Learning (CL) algorithms were developed. In this work we evaluate the performance of one such state-of-the-art CL method, GraphCL, that trains on graphs of code repositories. For this evaluation task, we generate a custom dataset with graphs having a new structure that combines both code-level and binary-level CPG graphs. We developed this new graph type, Cross Domain Control Property Graph (CDCPG), with the intuition that more feature data will help to improve the detection performance of the models and also there are software vulnerabilities which are not detectable with only source code. We evaluate different augmentation techniques and loss functions to see which technique mitigates the class imbalance issue in graphs and in the process find the optimal performance of GraphCL for vulnerability detection task.

1 INTRODUCTION

Detecting software vulnerabilities automatically have been a major topic of research in the cybersecurity domain for years because of the tendency of these vulnerabilities to cause significant damage to the system's confidentiality, integrity, and availability. A vulnerable program contains flaws that allows malicious actors to exploit its contents and harm any hardware or software associated with it.

Previously, Fabian Yamaguchi had introduced the Control Property Graph (CPG) in 2014 (Yamaguchi et al., 2014). To detect software vulnerabilities in function-level granularity, many researchers relaid on three different analyzing methods including the software metrics, source code analysis using Natural Language Processing (NLP) and graph learning methods. We have done extensive experiments with multiple graphs for different purposes such as syntax analysis using Abstract Syntax Tree(AST), semantic analysis using Control Flow Graph (CFG), Data Dependency Graph (DDG), Control Property Graph (CPG) which is a graph with combination of AST, CFG and DDG. There are projects which their source codes are not available and the owners did not make the projects open-source and CDCPG can play an import role.

Initially, researchers employed traditional detection methods (Li et al., 2019; Xu et al., 2017) that yielded poor true positive and true negative rates. This led them to use machine learning techniques (Nguyen & Tran, 2010; Neuhaus et al., 2007) that used features extracted by humans. But, since the feature extraction process was expensive and a labor intensive task, researchers soon started using Deep Learning (DL) techniques. DL techniques Li et al. (2018; 2021) showed promising results but missed one crucial aspect of the data in this domain. These methods disregarded the real world features of a program such as the control flow, data flow and syntactic structure of the program and considered them as natural language problems instead. Also, it is very expensive to collect large volume of labelled data that is required for training these DL models.

2 PROPOSED METHOD

One key design consideration that was taken into account when coming up with the architecture was the different contrasting methods. Both the node and graph representations learned for each graph

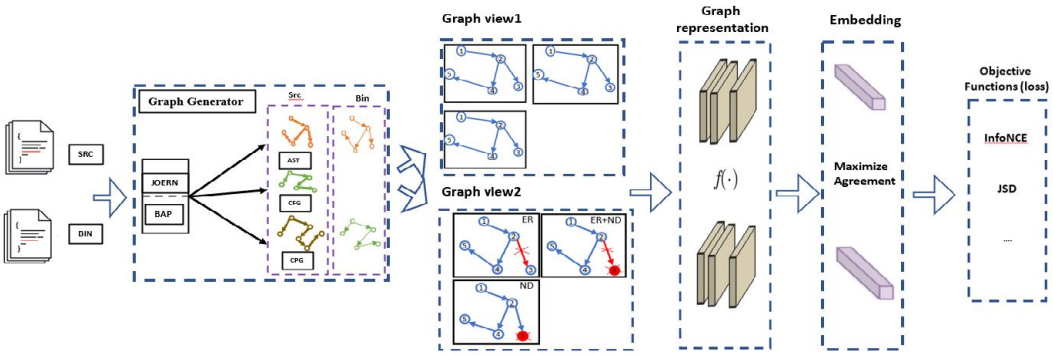

Figure 1: Data collection along with graph contrastive learning technique

by the graph neural network that were contrasted in multiple views depending on the objective function used. These contrasting methods could be broadly grouped into three modes which are: 1) Local-local CL, where the node representations are contrasted only, 2) Global-global CL, where the representations considered are only in graph level and 3) Global-local CL, where the comparisons were across scales, between node and graph representations. In this research, the Global-global mode has been used with the CDCPGs extracted from LibPNG software package to detect the vulnerabilities and the same process can perform on other applications.

Another design consideration that we looked into was the different loss functions and augmentation techniques that could be used with GraphCL algorithm to improve its performance. The loss functions(LF) we considered were InforNCE, Jensen-Shannon Divergence(JSD) with significant performance compare with other LFs.

3 EVALUATION

Table 1: F1-scores for different augmentation techniques and loss functions for GraphCL algorithm with CDCPG graphs. Results are shown for different contrastive modes of the algorithm.

Augmentation Types	**F1Mi**	**F1Ma**
Edge Adding (EA)	0.94	0.48
Edge Removing (ER)	0.91	0.47
Node Dropping (ND)	0.76	0.64
Feature Masking (FM)	0.96	0.82
Feature Dropping (FD)	0.91	0.48

Table 2: Evaluation of different augmentations based on CDCPG for LibPNG application

Modes	**InfoNCE-F1Mi**	**InfoNCE-F1Ma**	**JSD-F1Mi**	**JSD-F1Ma**
EA	**0.94**	**0.82**	0.79	0.55
ER	0.91	0.47	**0.97**	**0.50**
ND	0.88	0.46	**0.97**	**0.50**
FM	0.79	0.55	0.91	0.48
FD	0.91	0.47	**0.91**	**0.67**

4 CONCLUSION

We proposed a new graph dataset called CDCPG (CPG and Binary Graph) to detect vulnerabilities in source and binary level. To train the model, we used contrastive learning technique with multiple graph augmentation methods and evaluated the result based on different loss functions.

URM STATEMENT

The authors acknowledge that at least one key author of this work meets the URM criteria of ICLR 2023 Tiny Papers Track.

REFERENCES

Yuekang Li, Yinxing Xue, Hongxu Chen, Xiuheng Wu, Cen Zhang, Xiaofei Xie, Haijun Wang, and Yang Liu. Cerebro: context-aware adaptive fuzzing for effective vulnerability detection. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 533–544, 2019.

Zhen Li, Deqing Zou, Shouhuai Xu, Xinyu Ou, Hai Jin, Sujuan Wang, Zhijun Deng, and Yuyi Zhong. Vuldeepecker: A deep learning-based system for vulnerability detection. *arXiv preprint arXiv:1801.01681*, 2018.

Zhen Li, Deqing Zou, Shouhuai Xu, Hai Jin, Yawei Zhu, and Zhaoxuan Chen. Sysevr: A framework for using deep learning to detect software vulnerabilities. *IEEE Transactions on Dependable and Secure Computing*, 19(4):2244–2258, 2021.

Stephan Neuhaus, Thomas Zimmermann, Christian Holler, and Andreas Zeller. Predicting vulnerable software components. In *Proceedings of the 14th ACM conference on Computer and communications security*, pp. 529–540, 2007.

Viet Hung Nguyen and Le Minh Sang Tran. Predicting vulnerable software components with dependency graphs. In *Proceedings of the 6th International Workshop on Security Measurements and Metrics*, pp. 1–8, 2010.

Zhengzi Xu, Bihuan Chen, Mahinthan Chandramohan, Yang Liu, and Fu Song. Spain: security patch analysis for binaries towards understanding the pain and pills. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, pp. 462–472. IEEE, 2017.

Fabian Yamaguchi, Nico Golde, Daniel Arp, and Konrad Rieck. Modeling and discovering vulnerabilities with code property graphs. In *2014 IEEE Symposium on Security and Privacy*, pp. 590–604. IEEE, 2014.

A APPENDIX

A.1 DATA COLLECTION

Our dataset consists of graphs from six different applications as Abstract Syntax Tree (AST), Control Flow Graph (CFG), Data Dependency Graph (DDG), Control Property Graph (CPG). In this research we have reported only CDCPG 3 results with a few number of augmentation methods.

Table 3: Dataset Description. Code snippets were LibPNG application and then converted into graphs using our proposed data collection framework. The number of edges and nodes in each application are mentioned in this table along with the number of vulnerable and non-vulnerable instances in each application.

App Name	#Edges	#Nodes	#Graphs	#Vulnerabilities	#Non-Vulnerabilities
LibPNG	8415974	116839	324	25	299

A.2 APPLICATIONS AND ALL GRAPHS' DATASETS

Sample CDCPG can find in Figure (2)

A.3 SAMPLE OF CDCPG GRAPH

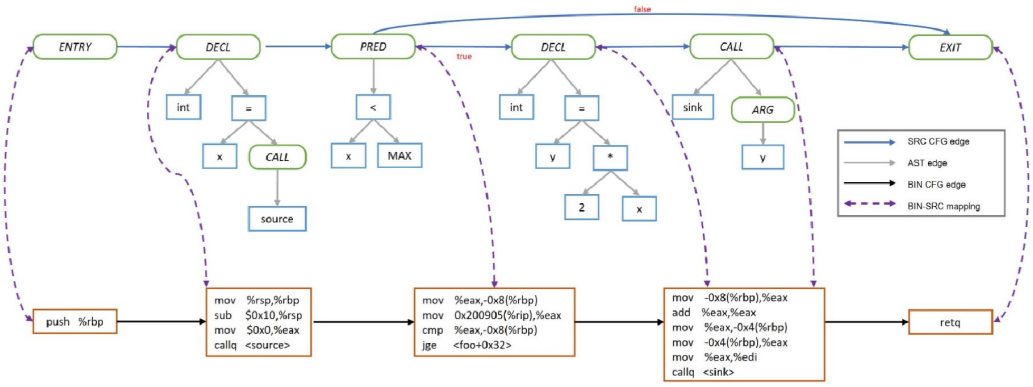

Figure 2: Cross-Domain Code Property Graph (CDCPG)

LIST OF TARGET APPLICATIONS AND THEIR DESCRIPTIONS

Application	Description
Sudo	Delegates security privileges to other users or tasks
Poftpd	Ftp server with configurable features
Libtiff	Tagged Image File Format (TIFF) library
Libpng	Portable Network Graphics (PNG) library
Freetype	Renders text into bitmaps
TinTin	Console telnet client for online gaming
Tcpdump	Data network packet analyzer
OpenSSH	Secure networking utility for Secure Shell protocol

DATASET DESCRIPTION. THE LAST TWO COLUMNS REPORT THE NUMBER OF VULNERABLE AND NON-VULNERABLE FUNCTIONS, RESPECTIVELY.

App Name	Graph Types	Edges	Nodes	Graphs	Vul.	Non-Vul.
LibPNG	CPG	1845382	82413	324	25	299
	CFG	3104607	12487			
	BIN	7861098	28248			
	AST	2628710	81777			
	CDCPG	8415974	116839			
LibTIFF	CPG	5125862	79009	438	33	405
	CFG	4498837	13142			
	BIN	9156127	47104			
	AST	3679494	78132			
	CDCPG	5746318	116839			
TCPDump	CPG	2074116	33461	100	15	85
	CFG	328240	6018	100	15	85
	BIN	607635	13679	100	15	85
	AST	6092203	57465	200	26	174
	CDCPG	4017677	48944	100	15	85
Sudo	CPG	5636108	41858	187	9	178
	CFG	740870	6213			
	BIN	2664002	25731			
	AST	4384746	41506			
	CDCPG	1617438	71463			
TinTin	CPG	8764300	44903	270	4	266
	CFG	1006857	6636			
	BIN	3747693	24376			
	AST	586926	44363			
	CDCPG	2361790	78414			
OpenSSH	CPG	5750707	70560	100	21	79
	CFG	732305	1003			
	BIN	1965966	36199			
	AST	4186296	70352			
	CDCPG	10937966	112250			

Figure 3: Applications and All Graphs Extracted from Source and Binary Codes