TranSec: Multi-Language Vulnerability Detection Using Transfer Learning

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Liam Brennan

Spring, 2022

Technical Project Team Members

Liam Brennan

Yuan Tian

Faysal Hussain Shezan

Tamjid Al Rahat

Wentao Chen

Yuan Tian, Department of Computer Science

TranSec: Multi-Language Vulnerability Detection Using Transfer Learning

CS4991 Capstone Report, 2023

Liam Brennan
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
lfb6ek@virginia.edu

ABSTRACT

Vulnerable code detection is one of the most critical yet challenging tasks in the modern age. Leveraging knowledge of similar vulnerabilities across different programming languages provides an opportunity for richer vulnerability detection. In this work, I built a machine learning framework, TranSec, that extracts high-level information about vulnerabilities to perform vulnerability detection. Building from recent works on Natural Language Processing (NLP), I designed a new model that can detect similar vulnerabilities across many different programming languages. TranSec outperforms various other vulnerability detection approaches while requiring significantly less training data. Additionally, TranSec is exceptional at extracting high-level information, implying that multi-language tasks other than vulnerability detection, such as language translation, can leverage TranSec. In the future, I plan to release a complete vulnerability detection system based on the TranSec model.

1. INTRODUCTION

With the increasing threat of cybercrime and the ever-growing impact of the internet on all sectors of everyday life, code vulnerability detection is one of the most critical problems today. Machine learning-based vulnerability detection methods have been a hot research topic over the past few years, delivering promising results. However, current machine learning models fail to learn vulnerabilities across different programming languages, lack functionality, and fail to leverage crucial vulnerability data.

In order to solve this problem, I designed TranSec, a model that learns high-level informat ion about vulnerabilities to perform detection across many different programming languages. I built TranSec on top of a particular type of Natural Language Processing (NLP) model called Transformers, which are used heavily in NLP systems such as ChatGPT. Leveraging my novel machine learning architecture and training procedure, TranSec is able to outperform previous vulnerability detection methods while requiring significantly less data.

2. BACKGROUND

This section provides a brief overview of two background topics vital to this project: Deep Learning and Vulnerability Detection.

2.1 Deep Learning

Deep Learning is the field that has revolutionized machine learning and AI over the past decade. Based on artificial neural networks and backpropagation, deep learning models are characterized by their immense size, need for enormous datasets, and ability to do challenging tasks, such as image generation (Ramesh et al, 2022), language generation (OpenAI, n.d.), and language translation. Deep learning models utilize representation learning by automatically learning complex features and data representations. In most cases, deep learning models are iteratively trained by being evaluated, punished based on some objective (loss function), then adjusted in tiny steps.

Additionally, Transfer Learning is a common Deep Learning technique that uses an old model as the starting point for a new model. For this project, we use transfer learning by taking a “pre-trained” model that learns the rules of language and uses it as a starting point for our vulnerability detection model. Transfer Learning often improves performance and reduces the need for data. In order to facilitate transfer learning, I used Transformers, a popular subset of Deep Learning models tailored toward learning language. A significant advantage of these models is the ability to “pre-train” by feeding it text without any human intervention.

2.2 Vulnerability Detection

Vulnerability detection is identifying code that could be exploitable by malicious actors. While highly diverse, in this work, we focus on web-based vulnerabilities. This code is used to build websites and is usually open to attackers on the internet.

3. RELATED WORKS

With the explosion of Deep Learning, machine learning based vulnerability detection has quickly taken over the field (Ghaffarian et al, 2017). Commonly, Recurrent Neural Networks, similar but more dated than Transformers, have been used to great success (Fidalgo et al, 2020; Dam, 2017). Convolutional Neural Networks, which revisualize source code as images, have also shown promise (Bilgin, 2021). Most recently, Graph Neural Networks, which work on abstract syntax tree representations of code, have been SOTA (Cheng et al, 2021).

Some works, such as Ahmad, et al. (2021), have shown the ability of transformers to extract high-level information from source code. Representing code in ways other than just text has also shown promise (Li et al, 2021). While many of these deep learning vulnerability detection works have shown promise, there is still large room for improvement. Throughout my extensive research, I have not found any work that has been able to successfully transfer semantics of vulnerabilities across platforms or leverage transfer learning on a large scale.

4. SYSTEM DESIGN

This section provides an Overview of TranSec and details of my implementation.

4.1 Overview

TranSec is built in three distinct steps: 1) Transformer Pre-training, 2) Vulnerability Classification, and 3) Transfer Learning. In Step 1, we pre-trained the model on a large corpus of C++, Python, and Javascript code. I based my model on ALBERT, a popular transformer architecture. I trained using the Masked-Language Model (MLM) objective, in which the model learns by guessing hidden words; and the Semantic Sovreighty Objective (SSO). By the end of this step, the model has been trained to output Semantic Embeddings, which are language independent representations of the text.

In Step 2, I used the Semantic Embeddings from Step 1 to train a model for Vulnerability Detection in either C++, Python, or Javascript. We use standard classification techniques.

In Step 3, I used the model from step 2 as a starting point and train for vulnerability detection in a new language.

4.2 Data Preprocessing

In order to feed the data into my model in each step, I must preprocess the data. To do this, I build tokenizers for each language, which take the given source code and turn it into a list of tokens, which include operators, keywords, variable names, and identifiers. I anonymize all variable names by giving them generic names. I create a vocabulary of tokens and give each unique token a corresponding vocabulary number (0-vocab_size), which I feed into the model. I wrote the preprocesssing code in a variety of languages as needed, including Javascript, C++, and Python.

4.3 Semantic Sovereignty Objective

As mentioned previously, during the pre-training step (Step 1), I trained the model on the Semantic Sovreighty Objective (SSO) in addition to the standard MLM objective. During training, SSO will take an average of the Semantic Embeddings across each language and punish the model based on the standard deviation between them. This optimizes the model to create language-independent embeddings, where the Semantic Embeddings, on average, are the same with each language.

4.4 Training

I built my system in Python, using PyTorch as my main machine learning framework. I used HuggingFace Transformers as the basis for my model. Additionally, I trained my model using GPUs on the UVA Computer Science Servers. Each model took roughly two days to train. I trained one pre-trained model, which I used as the basis for vulnerability detection in C++, Python, and Javascript.

5. RESULTS

I compared TranSec with other SOTA vulnerability detection methods, using F1 Score, MCC, and PR AUC as my comparison metrics. As shown in Figures 1, 2 and 3 below, TranSec outperformed other SOTA methods for C++, Javascript, and Python, reaching F1 scores of 87.7%, 87.1%, and 82.6%, respectively.

	F1	MCC	PR AUC
OSF'19 *(Russell et al, 2018)*	56.6%	0.54	51.8%
Code2Image *(Bilgin, 2021)*	50.5%	0.52	46.1%
Ast2Vec *(Paaßen et al, 2021)*	44.0%	0.38	37.2%
Cisco *(Tanwar et al, 2020)*	75.0%	-	-
AI4VA *(Suneja et al, 2020)*	56.0%	-	52.0%
ALBERT	82.1%	0.81	72.2%
TranSec	**89.7%**	**0.89**	**78.1%**

Table 1: Results of TranSec on the Draper VDISC C/C++ dataset compared to other methods

	F1	MCC	PR AUC
RAISE'19 *(Ferenc et al, 2019)*	76%	-	-
Szeged *(Viszkok et al, 2021)*	85%	-	-
GRU	66.5%	0.65	60.7%
Bi-LSTM	63.4%	0.61	59.0%
CNN-1D	79.1%	0.77	69.6%
CNN-2D	83.1%	0.81	72.2%
ALBERT	79.3%	0.76	68.2%
TranSec	**87.1%**	**0.85**	**75.0%**

Table 2: Results of TranSec on the RAISE Javascript dataset compared to other methods

	F1	MCC	PR AUC
GRU	75.7%	0.75	71.2%
Bi-LSTM	64.2%	0.62	59.8%
CNN-1D	82.1%	0.81	**77.7%**
CNN-2D	74.2%	0.73	70.1%
ALBERT	68.3%	0.67	63.8%
TranSec	**82.6%**	**0.82**	**77.7%**

Table 3: Results of TranSec on our Python dataset in different transfer settings compared to other methods

These results show the potential of TranSec and multi-language vulnerability detection. As a supplement to software designing, TranSec can be a powerful tool by catching many common vulnerabilities.

6. CONCLUSION

I built a novel machine-learning system called TranSec that leverages recent advances in Natural Language Processing and transfer learning. Using my Semantic Sovreignty Objective, I can directly embed language-independent semantic information between different programming languages, a task infeasible with previous methods. Consequently, TranSec outperforms previous SOTA methods in vulnerability detection for various programming languages.

7. FUTURE WORK

While this work focuses primarily on vulnerability detection, the novel pre-training method using the Semantic Sovreignty objective has other use cases. Due to TranSec's ability to extract semantic knowledge between languages, the same architecture can be used directly for tasks such as code translation. Additionally, TranSec can be extended to other domains, such as natural language processing.

More immediately, this work focuses on validating the TranSec framework experimentally. The next step is to use TranSec to build vulnerability detection applications, such as a web API or standalone program.

8. ACKNOWLEDGMENTS

I would like to thank Professor Yuan Tian (University of California, Los Angeles), Tamjid Al Rahat (University of California, Los Angeles), and Faysal Hussain Shezan (University of Virginia) for direction and guidance for this project. Professor Tian originally proposed building a transfer learning based vulnerability detection model named TranSec. Additionally, I would like to thank the UVA Computer Science department for funding and computing resources.

REFERENCES

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., & Chen, M. (2022). Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*.

OpenAI. (n.d.). *Introducing chatgpt*. Introducing ChatGPT. Retrieved April 8, 2023, from https://openai.com/blog/chatgpt

Ghaffarian, Seyed & Shahriari, Hamid Reza. (2017). Software Vulnerability Analysis and Discovery Using Machine-Learning and Data-Mining Techniques: A Survey. ACM Computing Surveys. 50. 1-36. 10.1145/3092566.

A. Fidalgo, I. Medeiros, P. Antunes and N. Neves, "Towards a Deep Learning Model for Vulnerability Detection on Web Application Variants," 2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), Porto, Portugal, 2020, pp. 465-476, doi: 10.1109/ICSTW50294.2020.00083.

Dam, H. K., Tran, T., Pham, T., Ng, S. W., Grundy, J. & Ghose, A. (2017). Automatic feature learning for vulnerability prediction. *arXiv preprint arXiv:1708.02368*.

Bilgin, Z. (2021). Code2image: Intelligent code analysis by computer vision techniques and application to vulnerability prediction. *arXiv preprint arXiv:2105.03131*.

Shezan, F. H., Cheng, K., Zhang, Z., Cao, Y. & Tian, Y. (2020). TKPERM: Cross-platform Permission Knowledge Transfer to Detect Overprivileged Third-party Applications. Network and Distributed System Security Symposium.

Ahmad, W. U., Chakraborty, S., Ray, B. & Chang, K. W. (2021). Unified pre-training for program understanding and generation. *arXiv preprint arXiv:2103.06333*.

Li, Z., Zou, D., Xu, S., Jin, H., Zhu, Y. & Chen, Z. (2021). Sysevr: A framework for using deep learning to detect software vulnerabilities. *IEEE Transactions on Dependable and Secure Computing*, *19*(4), 2244-2258.

Cheng, X & Wang, H & Hua, J & Xu, G & Sui, Y. (2021). DeepWukong: Statically Detecting Software Vulnerabilities Using Deep Graph Neural Network. *ACM Transactions on Software Engineering and Methodology. 30. 1-33. 10.1145/3436877.*

Russell, R., Kim, L., Hamilton, L., Lazovich, T., Harer, J., Ozdemir, O., ... & McConley, M. (2018, December). Automated vulnerability detection in source code using deep representation learning. In *2018 17th IEEE international conference on machine learning and applications (ICMLA)* (pp. 757-762). IEEE

Paaßen, B., McBroom, J., Jeffries, B., Koprinska, I., & Yacef, K. (2021). ast2vec: Utilizing recursive neural encodings of python programs. *arXiv preprint arXiv:2103.11614*.

Tanwar, A., Sundaresan, K., Ashwath, P., Ganesan, P., Chandrasekaran, S. K., & Ravi, S. (2020). Predicting vulnerability in large codebases with deep code representation. *arXiv preprint arXiv:2004.12783*.

Suneja, S., Zheng, Y., Zhuang, Y., Laredo, J., & Morari, A. (2020). Learning to map source code to software vulnerability using code-as-a-graph. *arXiv preprint arXiv:2006.08614*.

Ferenc, R., Hegedűs, P., Gyimesi, P., Antal, G., Bán, D. & Gyimóthy, T. "Challenging Machine Learning Algorithms in Predicting Vulnerable JavaScript Functions," *2019 IEEE/ACM 7th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE)*, Montreal, QC, Canada, 2019, pp. 8-14, doi: 10.1109/RAISE.2019.00010.

Viszkok, T., Hegedűs, P., & Ferenc, R. (2021). Improving Vulnerability Prediction of JavaScript Functions Using Process Metrics. *arXiv preprint arXiv:2105.07527*.