**AI-Driven Vulnerability Analysis in Critical Software Systems**

**Swathi Selvakumaran**

**UID:119090206**

**ENPM 808: INDEPENDENT STUDY**

**Dr. Tony D. Barber**

**DESCRIPTION OF PROJECT**

This project aims to develop an artificial intelligence (AI)-powered software system for detecting cybersecurity vulnerabilities in critical software systems before deployment. By integrating AI with existing static and dynamic code analysis tools, the project seeks to enhance the identification and mitigation of potential security, ultimately strengthening the defenses of critical systems. Through this project an AI vulnerability analysis model was developed and successfully integrated with IDE code analysis.

**ARCHITECTURE APPROACH**

**1. Flask Web API:**

It handles incoming requests from users submitting Java code snippets for vulnerability prediction. It interacts with the vulnerability prediction model to process the code and generate predictions.It returns the model's output as a JSON response to the user.

**2. Pre-trained JavaBERT Model:**

This model is used for feature extraction. It is a transformer-based model pre-trained on a massive dataset of Java code.

JavaBERT extracts essential features from the code, capturing relationships between code elements and potential vulnerabilities.

**3. Vulnerability Classifier Model**:

It takes the feature vector extracted by JavaBERT as input. It utilizes a Multi-Layer Perceptron (MLP) architecture with several hidden layers.

The MLP processes the features to learn complex relationships between code characteristics and vulnerability types.

The final layer of the MLP generates probabilities for each vulnerability class the model is trained to predict.

**4. Multi-label Classification:**

The model's output might be a list of predicted vulnerability classes or probabilities for each class, depending on the configuration.

**5. Code Preprocessing and Tokenization:**

Before feeding code into the model, the API performs preprocessing tasks like cleaning comments and whitespaces. The code is then tokenized using the tokenizer used during model training for consistency.

**DATA FLOW**

1. User submits a Java code snippet through the web API.
2. Flask receives the request and extracts the code snippet.
3. The code is preprocessed and tokenized.
4. The API loads the pre-trained vulnerability classifier model.
5. The preprocessed code and its tokens are fed into the JavaBERT model for feature extraction.
6. The extracted features are passed through the MLP layers of the vulnerability classifier model.
7. The model outputs probabilities for each vulnerability class.
8. Depending on the chosen classification strategy, the API might apply thresholds or select top-k probabilities.
9. The API prepares the final output containing the predicted vulnerability class(es) or probabilities.
10. Flask sends the prediction results back to the user as a response.

**TESTING**

Test Data Loading:

The test dataset is loaded and preprocessed similarly to the training and validation data.

Model Evaluation:

The trained model is used to predict vulnerabilities on the test set.

Evaluation metrics like accuracy, confusion matrix, and classification report are generated to assess model performance comprehensively.

The accuracy, classification report, confusion matrix and training and validation loss are provided in Appendix - Figure 1, 2, 3 and 4 respectively.

**CHALLENGES AND LIMITATIONS**

Model Size: The trained model size was large, exceeding the free tier memory limitations of cloud platforms. This poses a challenge for deploying the model in a cost-effective manner.

Compatibility Issues: The project encountered errors when using the latest PyTorch version. It required installation using conda for compatibility.

Limited Training Resources: Training the model was time-consuming and resource-intensive due to the absence of a GPU.

**WHAT I LEARNED**

Deep Learning for Vulnerability Prediction: This project provided experience in applying deep learning models for code analysis and vulnerability detection tasks.

Multi-label Classification: I gained experience working with multi-label classification problems.

Data Preparation: Data preparation plays a crucial role in the success of machine learning projects. Data cleaning, balancing, and transformation are important for model training.

Flask Web API Development: I learned how to develop a web API using Flask, enabling interaction with the vulnerability prediction model.

**Appendix**

Figure 1. Testing accuracy:



Figure 2. Classification report:

A table of numbers with numbers on it

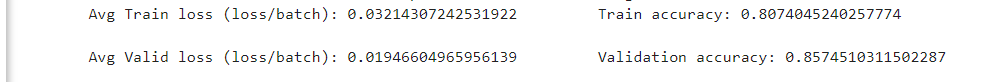
Description automatically generated

Figure 3. Confusion Matrix:

A screenshot of a computer code

Description automatically generated

Figure 4. Training and Validation loss:



**References**

[AI Based Cyber Threats and Vulnerability Detection, Prevention and Prediction System | IEEE Conference Publication | IEEE Xplore](https://ieeexplore.ieee.org/document/9103372)

[Revolutionizing Code Quality with AI-Based Static Code Analysis Tools | by Katie Wanders | Medium](https://medium.com/@katie.wan/revolutionizing-code-quality-with-ai-based-static-code-analysis-tools-c4b4a2992237)

[CodeQL team uses AI to power vulnerability detection in code - The GitHub Blog](https://github.blog/2023-09-12-codeql-team-uses-ai-to-power-vulnerability-detection-in-code/)

[Leveraging machine learning to find security vulnerabilities - The GitHub Blog](https://github.blog/2022-02-17-leveraging-machine-learning-find-security-vulnerabilities/)

[Code scanning finds more vulnerabilities using machine learning - The GitHub Blog](https://github.blog/2022-02-17-code-scanning-finds-vulnerabilities-using-machine-learning/)

[Secure Software Development Framework | CSRC (nist.gov)](https://csrc.nist.gov/projects/ssdf)

[Source Code Security Analyzers | NIST](https://www.nist.gov/itl/ssd/software-quality-group/source-code-security-analyzers)

[Static Analysis: 20th International Symposium, SAS 2013, Seattle, WA, USA, June 20-22, 2012, Proceedings | SpringerLink](https://link.springer.com/book/10.1007/978-3-642-38856-9)

<https://huggingface.co/docs/transformers/en/model_doc/bert>

<https://code.visualstudio.com/api/get-started/your-first-extension>

Analysis tools available: <https://github.com/analysis-tools-dev>