Cloud-based PE Malware Detection API

AI and Cybersecurity - Spring 2024

Jafar Vohra

Project Purpose:

The purpose of this project is to deploy a machine learning model for malware classification. This project comprises of three tasks. The initial task is to train a deep neural network to classify PE files as malware or benign using Ember opensource dataset, EMBER-2017 v2. The second task deals with deploying the model to cloud and creating an endpoint, or an API, to the model. The final task is to create a web application using Streamlit where the user of the application can upload a PE file that is then classified as malicious or benign.

Requirements:

To complete this project, one must use Google Colab, Amazon Sagemaker, and Streamlit. Google Colab is utilized for the training of the model. Amazon Sagemaker is leveraged for the model deployment. Streamlit is employed for the Client creation and user interaction portion of project. Each of these tools is essential to the success of the project.

Training Implementation:

In order to complete the model training portion of the project, a notebook is created in Google Collaboratory. The training and testing data as well as the metadata is acquired from an Amazon S3 bucket and copied to Google Drive in order to be accessed. All of the code from the

Ember repository is installed before using one of the methods from there to read the vectorized features and metadata and save them to usable objects in the notebook. Rows in the training datatset with no labels are then discarded. At this point, the preprocessing steps are nearly complete.

In order to effectively work on the model and train more quickly and improve model performance, a subset of the entire dataset is used. A 5% sample of the original 600,000 training data records and 200,000 test data records is taken via the Pandas DataFrame sample() method, ensuring the class weights are maintained. This sampled data is not used in the final model, however it is essential to the process of completing the project. Once this is complete, the datasets are scaled via StandardScaler from sklearn. The data is then reshaped to be easily consumed by the model before being tensorized and split into training and validation sets. A batch size of 64 is set before the model is then set to be trained.

The MalConv Neural Network model is built using PyTorch. It starts with an embedding layer that utlizes 8 channels over the 256 unique bytes. It is followed by a permutation of the shape of the data in order to be consumed by the convolution layers of the model. There are two convolution layers implemented in the model with a dropout rate of 25% used between each after passing through the ReLU activation. After the convolution and dropout layers, a global max pooling is completed. Next, two fully connected layers are implemented before data passes through the final sigmoid activation function to create the classifications.

This model is then trained with the Binary Cross-Entropy Loss Function and the Adam Optimizer. Trained for 15 total Epochs, the model is passed forward with zeroed parameter gradients in order to get the outputs and loss value before being passed backwards and enforcing

a step in the optimizer. Training and validation loss is printed for each Epoch while checkpoints for the model are saved at every 5 epochs.

Finally, the model is evaluated on the separate test dataset after also being tensorized and loaded. After the model is set to evaluation mode, the test data is passed through the model. During this process logit values are converted to probabilities via the sigmoid function before predictions and labels are made and flattened in order to be evaluated by the functions to compute the evaluation metrics.

Performance Analysis:

The performance metrics provided indicate several important aspects of the model's performance, but they also raise some concerns. The test accuracy represents the proportion of correctly classified samples out of the total test set. In this case, the accuracy is approximately 52.31%, which means the model is slightly better than random guessing. However, this accuracy level might not be satisfactory, especially in the case of malware detection. Precision measures the proportion of true positive predictions out of all positive predictions made by the model. A precision of 0.5118 suggests that around 51.18% of the samples predicted as positive are actually positive. This indicates that the model's ability to avoid false positives is relatively low. Recall, also known as sensitivity, measures the proportion of true positive predictions out of all actual positive samples. A recall of 0.9999 indicates that the model is capturing almost all positive samples in the dataset. While high recall is desirable, such a high value, especially when combined with lower precision, could indicate potential issues, such as the model being overly sensitive and possibly prone to false positives.

Overall, the model's performance appears to have a significant trade-off between precision and recall, where it tends to classify many samples as positive (either Malware or Benign) but struggles with precision, leading to a high false positive rate. This imbalance might be due to class imbalance in the dataset or model complexity. Further investigation into the model's behavior on different subsets of the data, analysis of misclassifications, and potential adjustments to the model architecture or training process could help improve its performance. Additionally, considering alternative evaluation metrics like F1-score, which combines precision and recall, might provide a more comprehensive understanding of the model's performance. With more time and resources allocated to this portion of the project, the model's performance can be significantly improved.

Deployment Implementation:

Client Implementation:

Conclusion:

Overall, this project proved to be an extremely difficult one. With many challenges faced in the model training portion of the project, the second and third tasks were unable to be completed by the extended deadline. The plan is to continue to work on the uncompleted tasks as they are invaluable to the current project as well as the skills they are able to reinforce through their implementation.