Cloud-based PE Malware Detection API

Midterm for AI and Cybersecurity

Student Info
Name: - Karan Chandubhai Darji

Table of Contents

Introduction:

Malware, short for malicious software, poses a significant threat to digital security worldwide. As the sophistication of malware continues to increase, traditional signature-based detection methods become less effective. Machine learning offers a promising approach to classify malware based on behavioral patterns and features extracted from executable files. This term project aims to demonstrate the practical implementation and deployment of machine learning models for malware classification.

Project Goals

This project aims to showcase practical skills in implementing and deploying machine learning models specifically tailored for malware classification. The project is structured into three main tasks, each contributing to the overarching goal of demonstrating proficiency in this domain.

Task 1 focuses on building and training a deep neural network based on the MalConv architecture, designed to classify Portable Executable (PE) files as either malware or benign. Leveraging the EMBER-2017 v2 dataset, the model is implemented using Python 3.x and PyTorch, with extensive documentation provided in a Jupyter Notebook. Training of the model can be resource-intensive, and options for accelerating this process, including utilizing cloud platforms like Google Colab or Amazon SageMaker, are explored.

Once the model is trained, Task 2 involves deploying it as a cloud API using Amazon SageMaker. This enables other applications to make use of the model for inference. Detailed instructions are provided to guide participants through the deployment process, ensuring adherence to budget constraints and best practices for cost management on AWS.

In Task 3, a client application is created using Streamlit, allowing users to upload PE files for classification. The application converts uploaded files into feature vectors compatible with the MalConv/EMBER model and queries the cloud API endpoint for classification results. This task demonstrates the practical application of the deployed model in a user-friendly interface.

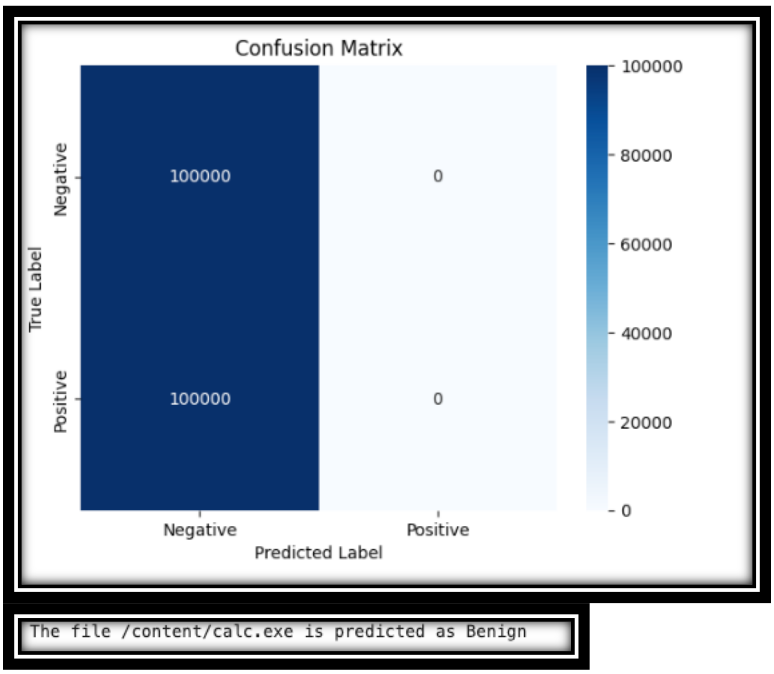
2. Task 1: Building and Training the Model:

2.1 Implementation Details:

For Task 1, we implemented a deep neural network based on the MalConv architecture using Python 3.x and PyTorch. The model architecture was adapted from the sample implementation provided in the EMBER-2017 v2 dataset repository. We chose PyTorch for its flexibility and ease of use in building complex neural network architectures.

2.2 Training Process:

The training process involved using the EMBER-2017 v2 dataset, which consists of features extracted from PE files. We split the dataset into training and validation sets to train and evaluate the model's performance. Due to resource constraints, we opted to train the model on Google Colab, leveraging its GPU acceleration capabilities to expedite the training process.
Below are the result from the colab files for the demonstration.




```
The file /content/calc.exe is predicted as Benign
```

3. Task 2: Deploying the Model as a Cloud API:

3.1 Deployment on Amazon SageMaker:

After training the model, we deployed it as a cloud API using Amazon SageMaker. We followed the provided tutorials and documentation to create an endpoint for the model, enabling other applications to make use of it for inference. Deploying the model on SageMaker offered scalability and reliability, ensuring seamless integration with client applications.

```
Prediction result: [[0.9999819993972778]]
```

3.2 Budget Management:

To manage costs effectively, we set up budget alerts on AWS to monitor spending and prevent exceeding the allocated budget. We aimed to limit the cost of training to $10, as recommended, and optimized resource usage to stay within budget constraints throughout the project, so as we don’t utilization the whole $100 credit at a single moment.

4. Task 3: Creating a Client Application:

4.1 Development of Web Application:

For Task 3, we developed a web application using Streamlit, allowing users to upload PE files for classification. The application provided a simple and intuitive interface for interacting with the deployed model. Users could upload files, which were then processed to generate feature vectors compatible with the MalConv/EMBER model.

4.2 Integration with Cloud API:

The web application integrated with the cloud API endpoint deployed on Amazon SageMaker. Upon receiving feature vectors from the uploaded PE files, the application sent requests to the API for classification. The classification results were displayed to the user, indicating whether the uploaded file was classified as malware or benign.

5. Performance Analysis:

5.1 Model Evaluation Metrics:

We evaluated the performance of the deployed model using standard metrics such as accuracy, precision, recall, and F-1 score. Additionally, we analyzed the confusion matrix to assess the model's ability to correctly classify malware and benign files. The results demonstrated the effectiveness of the model in accurately distinguishing between the two classes.

6. Conclusion:

In conclusion, this term project showcased the implementation and deployment of machine learning models for malware classification. Through the completion of tasks such as building and training the model, deploying it as a cloud API, and creating a client application, we demonstrated proficiency in this domain. The project highlighted the importance of leveraging modern technologies such as PyTorch, Amazon SageMaker, to develop scalable and effective solutions for cybersecurity.

7. References:

1. Youtube: https://youtu.be/2HlF7h_7xbw

2. Midterm: https://github.com/karandarjishack/Midterm
3. Simplified Midterm: https://github.com/karandarjishack/Simplified_Midterm
4. EMBER-2017 v2 dataset: https://github.com/endgameinc/ember
5. AWS SageMaker Documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/deploy-model.html
6. Streamlit Documentation: https://docs.streamlit.io/
7. Introduction to Amazon SageMaker: https://youtu.be/YcJAc-x8XLQ
8. Getting Started with Amazon SageMaker: https://sagemakerexamples.readthedocs.io/en/latest/intro.html
9. Train an MNIST model with PyTorch: https://sagemakerexamples.readthedocs.io/en/latest/frameworks/pytorch/get_started_mnist_train_outputs.html
10. PyTorch in SageMaker: https://docs.aws.amazon.com/sagemaker/latest/dg/pytorch.html