AI & Cybersecurity
MID-TERM PROJECT

Cloud-based PE Malware Detection API

- Prof. Vahid Behzadan

REPORT

By

Vatsalya Nayudu

MS in Data Science

University New Haven

1.Project Objective

The project's main goal is to deploy a malware detection model on AWS SageMaker, turning it into an API endpoint for easy access. This includes developing a Python client that communicates with this endpoint, enabling users to remotely classify files. Testing with various malware and benign files ensures the system can effectively differentiate threats.

Through rigorous validation, the project ensures the reliability and accuracy of the system in identifying malicious executables. Overall, it provides a simple and efficient solution for remote malware detection, utilizing the power of machine learning and cloud computing.

2.Requirements:

The project utilizes multiple platforms: Amazon SageMaker for deploying the model, Jupyter Notebook for development, and Google Colab for executing the client file. This setup enables seamless model deployment, interactive development, and remote file classification, streamlining the process of malware detection.

3.Implementation:

This project is implemented task wise. So, the details in the report are also given based on the tasks performed.

3.1 Data Preparation and Preprocessing:

In this stage, the project sets up directories containing labeled samples, ensuring a structured dataset. The **train_test_split** function from **sklearn.model_selection** is employed to partition the dataset into distinct training and testing subsets, facilitating subsequent model evaluation.

3.2 Feature Extraction:

This phase involves the extraction of features from the raw data. Byte sequences of files are transformed into n-grams using the **byteSequenceToNgrams** and **extractNgramCounts** functions. N-gram features are then derived from each sample utilizing the **getNGramFeaturesFromSample** method.

3.3 Metadata Extraction:

The extraction of metadata, including imports and section names from Portable Executable (PE) files, is crucial for enriching the feature set. This is achieved through the **getImports** and **getSectionNames** functions, providing additional contextual information for the classifier.

3.4 Feature Engineering:

Feature engineering plays a pivotal role in enhancing the model's predictive capabilities. Here, a pipeline comprising sequential transforms (**HashingVectorizer** and

TfidfTransformer) is constructed to extract features from DLL imports and section names. These features are then integrated with binary N-gram features to construct comprehensive vectorized training samples.

3.5 Model Training:

A Random Forest classifier, implemented through the **RandomForestClassifier** from **sklearn.ensemble**, is trained on the enriched dataset. This stage establishes the core predictive framework for malware/benign classification.

3.6 Model Evaluation::

Training accuracy is computed using **clf.score** on the training data, providing insights into the model's performance on the training set. Subsequently, feature vectors are generated for the test samples, and the test accuracy is computed, offering a comprehensive assessment of the model's predictive capabilities.

4.Model Deployment Process

4.1 Directory Setup and Artifact Copying:

This section sets up the directory structure and copies the necessary model artifacts to the appropriate location.

4.2 Inference Script Creation:

The inference script is written to handle input data on the SageMaker endpoint and return predictions. It includes functions for deserializing the fitted model, handling input data, making predictions, and formatting output.

4.3 Dependency Specification:

A requirements.txt file is generated to specify dependencies required to run the model on the SageMaker container.

4.4 Artifact Packaging:

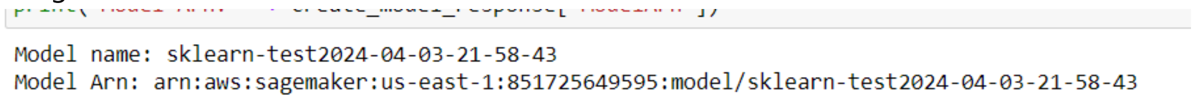
Model artifacts and the inference script are compressed into a single `model.tar.gz` file for easy deployment.

4.5 SageMaker Environment Setup:

Libraries are imported, and Boto3 is configured to interact with AWS services. The SKLearn image URI is retrieved for model deployment. Libraries are imported, and Boto3 is configured to interact with AWS services. The SKLearn image URI is retrieved for model deployment.

4.6 Model Creation:

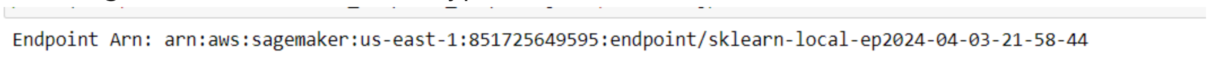
A SageMaker model is created using the SKLearn image URI. Model containers are configured, and model artifacts are associated with the model.



```
Model name: sklearn-test2024-04-03-21-58-43
Model Arn: arn:aws:sagemaker:us-east-1:851725649595:model/sklearn-test2024-04-03-21-58-43
```

4.7 Endpoint Configuration:

An endpoint configuration is created, specifying production variants for the model, including details such as instance type and count.



```
Endpoint Arn: arn:aws:sagemaker:us-east-1:851725649595:endpoint/sklearn-local-ep2024-04-03-21-58-44
```

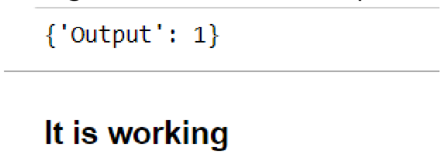
Fig: Endpoint created on sagemaker

4.8 Endpoint Configuration:

The endpoint is created on SageMaker using the configured endpoint configuration. The deployment process is monitored until the endpoint is successfully created.

4.8 Endpoint Testing:

The endpoint's functionality is tested by initializing the SageMaker runtime client, defining input data, making a test call to the endpoint, and verifying the response.



```
{'Output': 1}
```

It is working

Fig: Output of Testing Endpoint

5. Creating Client & Testing

5.1 Importing Libraries:

The necessary libraries, including boto3, pefile, and streamlit, were installed using the pip package manager. Following the installation process, the versions of the installed libraries were verified to be boto3 version 1.34.77, pefile version 2023.2.7, and streamlit version 1.32.2.

5.2 IP Address Retrieval:

The IP address was obtained using the **wget** command, fetching the IP address from **ipv4.icanhazip.com**. The resulting IP address returned was 34.85.163.96.

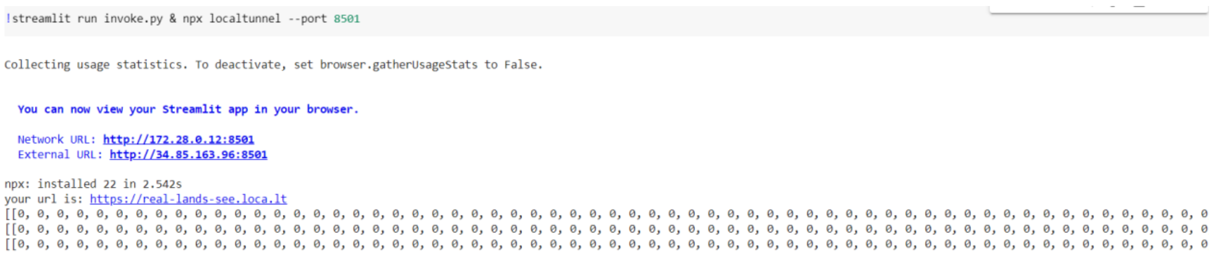
5.3 Code Development:

A Python script named **invoke.py** was created to develop a Streamlit application for SageMaker inference. This script enabled users to upload a **.exe** file and extract features using various functions from the **pefile** library, such as **getImports** and

getSectionNames. These features were then sent to the SageMaker endpoint for inference.

5.4 Streamlit App Execution:

The Streamlit application was executed using the **streamlit run invoke.py** command. Additionally, simultaneous deployment was achieved using **npx localtunnel --port 8501**. Upon execution, the Streamlit app displayed predictions, indicating "Benign - Safe" if the output is 0 and "Malware - Danger" if the output is 1.



```
!streamlit run invoke.py & npx localtunnel --port 8501

Collecting usage statistics. To deactivate, set browser.gatherUsageStats to False.


  You can now view your Streamlit app in your browser.

  Network URL: http://172.28.0.12:8501
  External URL: http://34.85.163.96:8501

npx: installed 22 in 2.542s
your url is: https://real-lands-see.loca.lt
[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
```

Fig: Output after testing benign file

6.Conclusion

This project successfully develops a Streamlit application for SageMaker inference, focusing on malware detection. Leveraging technologies such as boto3, pefile, and streamlit, the application enables users to upload .exe files, extract features, perform inference via a SageMaker endpoint, and visualize predictions. By integrating machine learning models and feature extraction techniques, the application offers a user-friendly interface for efficient malware detection, contributing to enhanced cybersecurity measures. This project showcases the effective utilization of diverse tools and technologies to address a crucial aspect of cybersecurity, highlighting the importance of seamless integration and user-centric design in tackling complex problems.