Midterm Project

AI and CyberSecurity
DSCI6015
Simplified Midterm

Renuka gadde
00881378
University of New Haven
Dr.Vahid Behzadan
April 02,2024

Overview

This document details the process of creating a system on the cloud that can detect malware, specifically focusing on Portable Executable (PE) files. We built this system using AWS Sagemaker. It works by using a type of binary classifier called Random Forest, which we trained on labeled datasets of binary feature vectors. To develop, train, and deploy the model, we used Python and machine learning libraries like sklearn, pefile, and nltk. We also created a user-friendly web application where people can upload executable (.exe) files to check if they're malicious or not in real-time. We tested the model's performance using a dataset that included 100 malware samples and 100 benign samples, and it proved to be effective at distinguishing between the two. Overall, this project emphasizes the importance of being proactive about cybersecurity and using advanced machine learning techniques to stay ahead of evolving threats.

Introduction

The project encompasses three primary tasks aimed at developing, deploying, and testing a machine learning-based malware detection system.

1. Model Training and Deployment:

The initial phase involves training a random forest classifier to discern malware based on features extracted from Portable Executable (PE) files. Leveraging AWS SageMaker, the trained model is deployed as an API endpoint, utilizing its comprehensive suite of tools for efficient model development, training, and deployment.

2. Client Development:

Following the deployment of the model, a Python client application is created to facilitate interaction with the deployed system. This client accepts executable files as input, extracts relevant features from the PE files, and forwards these features to the SageMaker endpoint for classification. The client then presents the model's prediction regarding the file's malicious or benign nature to the user.

3. Testing Client and Endpoint:

In this phase, the developed client application and the deployed model endpoint are tested using representative samples from the test dataset. One malware PE file and one benign PE file are selected for testing. The objective is to demonstrate the system's functionality in accurately classifying the provided files. A demonstration video is compiled to showcase the entire process, including file selection, upload, classification, and results display.

Through the execution of these tasks, the project demonstrates the practical application of machine learning in malware detection. By developing a robust detection system, deploying it as an API endpoint, and validating its functionality through testing, the project underscores the effectiveness and reliability of the developed solution in combating malicious software threats.

Methodology

The methodology employed in this project encompasses feature extraction from Portable Executable (PE) files, training of a Random Forest classifier, deployment of the model as a cloud API, creation of a client application for user interaction, and performance validation with PE files.

1. Feature Extraction from PE Files:

Byte Sequence n-grams: Consecutive sequences of bytes are extracted to capture patterns in the raw byte sequences of PE files.

Imported DLLs: Names of dynamically linked libraries (DLLs) imported by PE files are extracted and preprocessed for feature representation.

Section Names: Names of sections within PE files are extracted and processed to reveal potential structural patterns, along with extracting the number of sections as a feature.

2. Model Training with Random Forest Classifier:

A Random Forest classifier is trained using scikit-learn in AWS SageMaker with a labeled dataset of binary feature vectors.

The trained model is saved into a joblib file for future deployment.

3. Deployment of the Model as a Cloud API:

The trained model is deployed on Amazon SageMaker, creating an endpoint for a cloud-based API for real-time predictions.

The model is loaded using the saved joblib file and deployed using sagemaker.SKLearn module.

4. Creation of a Client Application:

A Streamlit web application is developed to provide a user-friendly interface for users to upload executable (.exe) files.

The application extracts features from the uploaded .exe files using pefile library and other pretrained data.

Extracted features are converted into JSON format and sent to the deployed API.

The application displays the classification results (Malware - Danger or Benign - Safe).

5. Performance Validation with PE Files:

The performance of the deployed model is validated using PE files.

Representative samples of PE files, including both malware and benign files, are selected from the test dataset.

These PE files are uploaded to the client application, and their classification results are compared against ground truth labels.

Metrics such as accuracy, precision, recall, and F1 score are computed to evaluate the model's performance in classifying PE files accurately.

Execution Steps:

1. Model Development and Training:

1. Install scikit-learn version 1.2.1 within the AWS SageMaker environment.
2. Prepare the labeled dataset of binary feature vectors for training.
3. Train a Random Forest binary classifier using scikit-learn.
4. Evaluate the model's performance using cross-validation or a separate validation dataset.
5. Serialize the trained model into a joblib file for future use.

2. Deployment of the Model as a Cloud API:

1. Log in to the Amazon SageMaker account.
2. Navigate to the SageMaker console and create a new model.
3. Upload the serialized joblib file containing the trained model.
4. Configure the deployment settings, including the instance type and number of instances.
5. Create an endpoint for the deployed model.
6. Test the endpoint to ensure functionality.

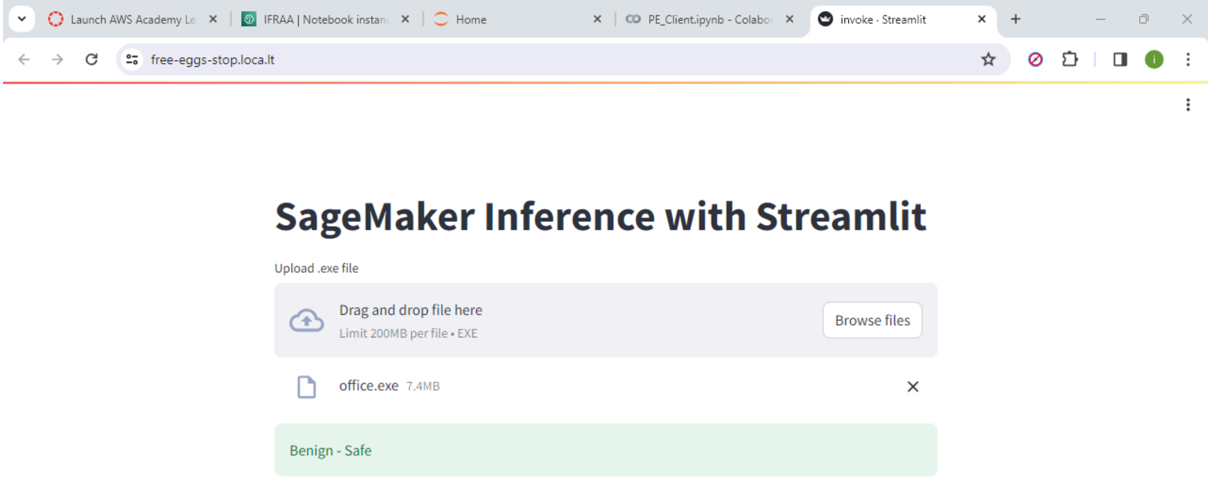
3. Development of Client Application:

1. Install Streamlit and other necessary libraries for the client application.
2. Design the user interface for the Streamlit web application.
3. Implement functionality to allow users to upload executable (.exe) files.
4. Utilize the pefile library to extract features from uploaded .exe files.
5. Convert the extracted features into JSON format for transmission to the deployed API endpoint.
6. Display the classification results (Malware - Danger or Benign - Safe) within the application interface.

4. Validation:

1. Select a representative set of PE files for validation, including known malware and benign samples.
2. Develop a validation script or pipeline to upload PE files to the deployed API endpoint.
3. Capture the classification results returned by the endpoint for each file.
4. Compare the classification results against ground truth labels to evaluate the model's performance.
5. Compute metrics such as accuracy, precision, recall, and F1 score to assess the model's effectiveness.
6. Iterate on model improvements based on validation results if necessary.

Output :



SageMaker Inference with Streamlit

Upload .exe file

Drag and drop file here

Limit 200MB per file • EXE

Browse files

office.exe 7.4MB

Benign - Safe