University *of* New Haven

Midterm Project

AI and CyberSecurity

DSCI6015

Simplified Midterm

Yuva Krishna Kishore Inapala

00866841

University of New Haven

Dr. Vahid Behzadan

April 01, 2024

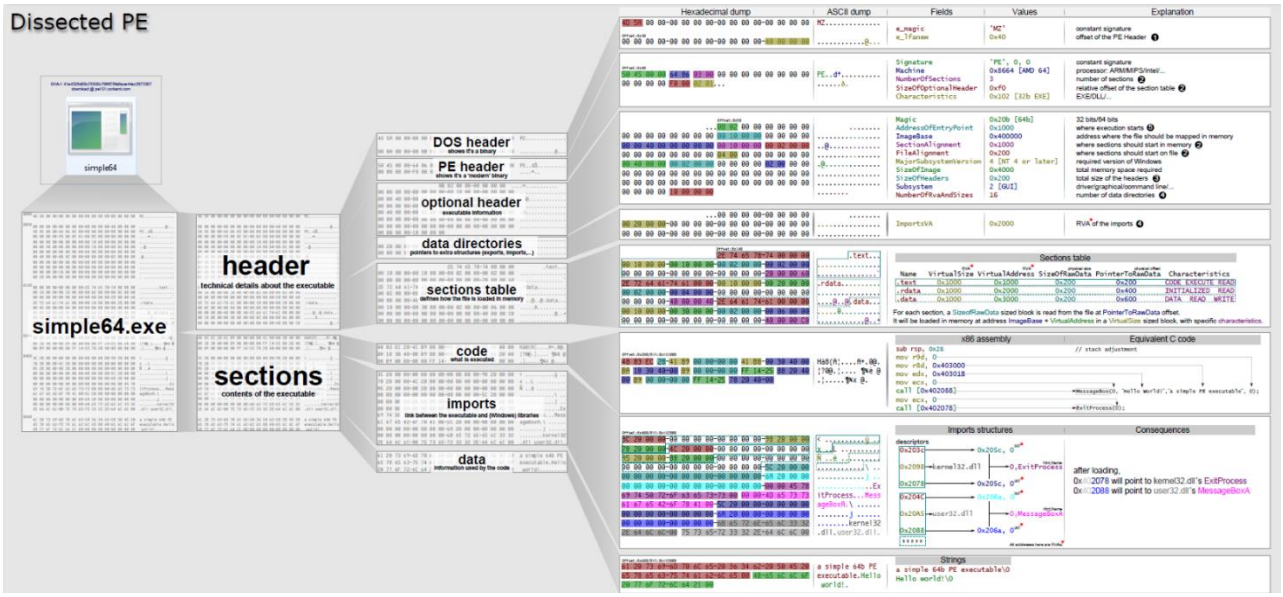
Summary

This report documents the successful development of a cloud-based PE (Portable Executable) static malware detection API using AWS Sagemaker. The API utilizes a Random Forest binary classifier, trained with labeled dataset of binary feature vectors dataset, to classify Portable Executable (PE) files as malicious or benign.
The project leveraged AWS Sagemaker from building and training the model, and using same Amazon Sagemaker instance for model deployment. A web app is created for user-friendly client application, where a remote user can upload their executable(.exe) file and find out the threat. Python language was used for the project and the ML libraries like sklearn, pefile, nltk etc was used for the model creation and implementation.

Introduction

PE Files

PE files, utilized by Windows operating systems, serve as a file format for storing executable code and associated data. These files encompass vital details necessary for program execution, encompassing machine instructions, resources, imported libraries, and metadata. Predominantly employed for applications, drivers, and dynamic link libraries (DLLs), PE files adhere to a structured layout featuring headers that furnish insights into the file's attributes, including its architecture, entry point, and section arrangement. Proficiency in comprehending the PE file format proves invaluable for activities such as software analysis, reverse engineering, and malware detection, enabling the examination and modification of executable content.



Random Forest Classifier

A Random Forest classifier is a versatile and powerful machine learning algorithm used for both classification and regression tasks. It belongs to the ensemble learning family, which combines multiple individual models to make predictions. In the case of Random Forests, the individual models are decision trees. The "forest" in Random Forest refers to a collection of decision trees, where each tree is built independently and operates by making decisions based on a set of input features.

The key idea behind Random Forests is to create a diverse set of decision trees by introducing randomness during both the training process and the prediction phase. During training, each tree is built using a random subset of the training data and a random subset of features for each split. This randomness helps to decorrelate the individual trees, reducing the risk of overfitting and improving the overall performance of the model. During prediction, the output of each tree is combined through a process called averaging or voting, where the most frequent class (for classification tasks) or the average (for regression tasks) is taken as the final prediction.

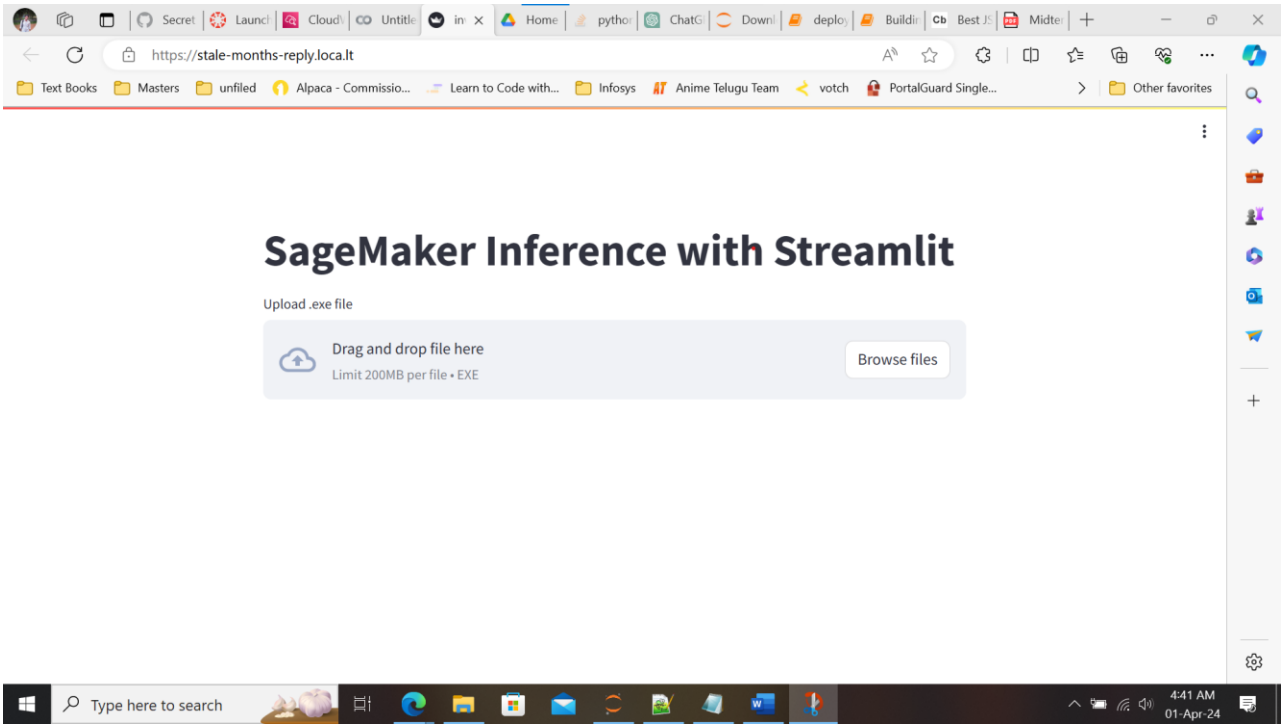
One of the main advantages of Random Forests is their ability to handle high-dimensional data with many features. They are also robust to noise and outliers in the data and require minimal hyperparameter tuning compared to other complex models. Additionally, Random Forests provides insights into feature importance, allowing users to understand which features contribute the most to the model's predictions. Overall, Random Forests are widely used in various fields such as finance, healthcare, and bioinformatics due to their excellent performance, scalability, and interpretability.

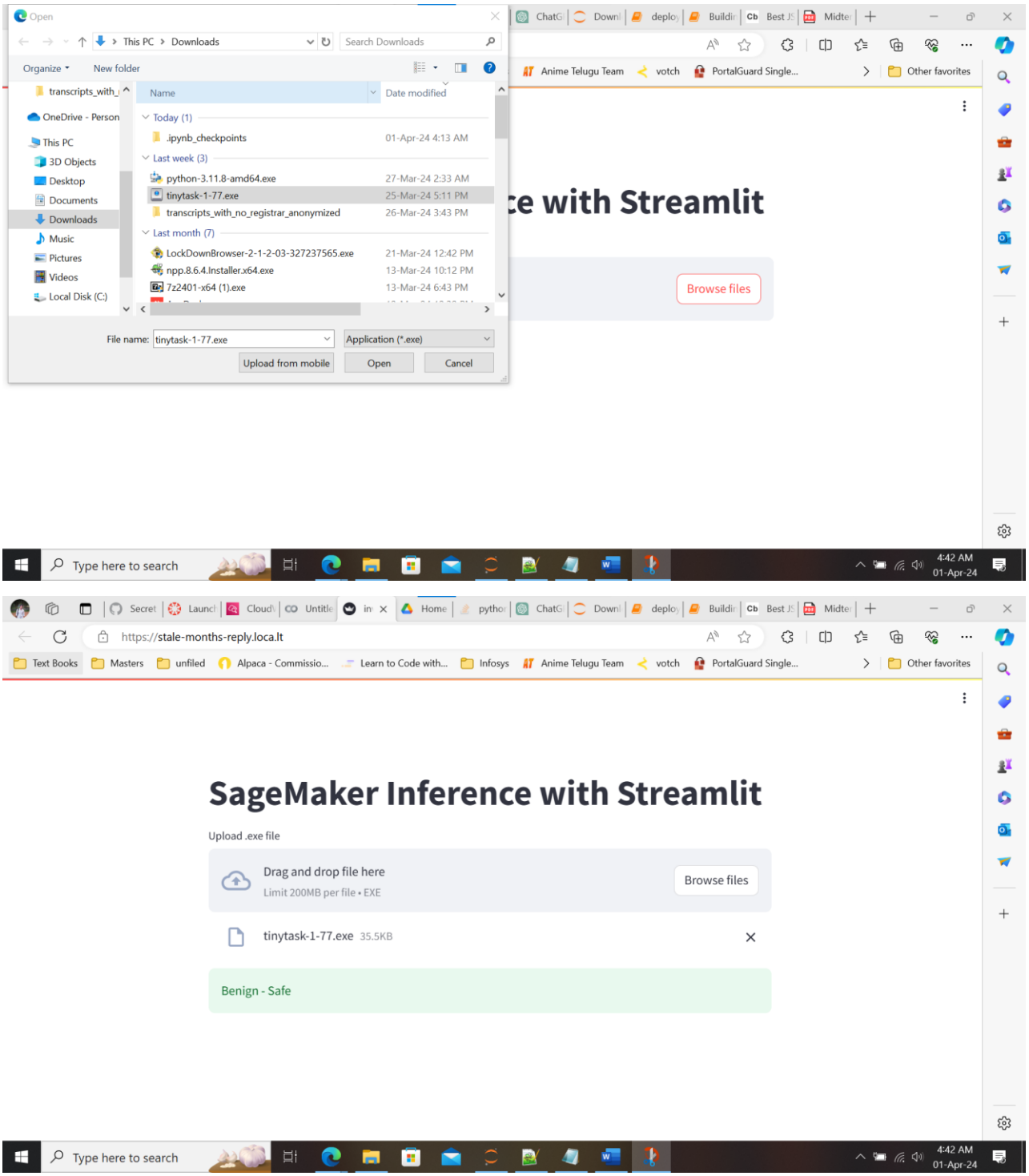
Task Approach:

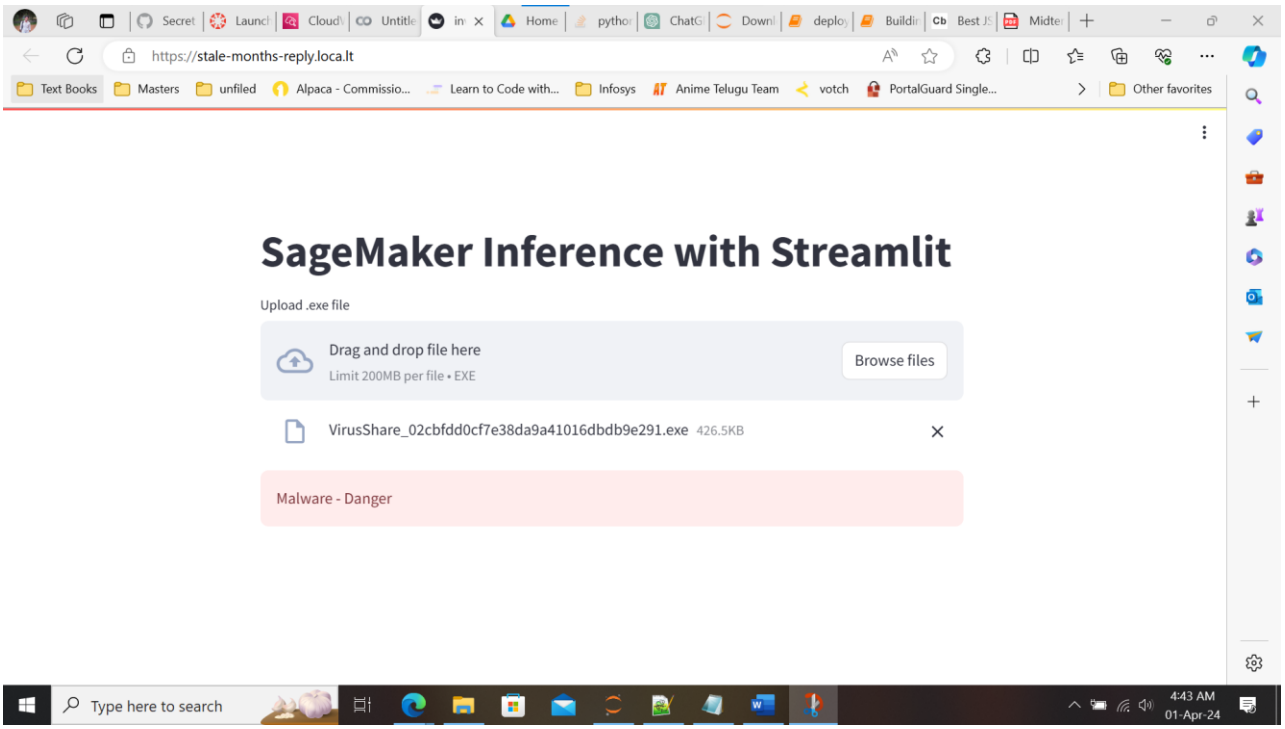
1. **Building and Training the Model:** A scikit-learn version of 1.2.1 is used for training and development in AWS Sagemaker. The Random Forest binary Classifier trained using a proper labeled dataset of binary feature vectors. Later the trained model is dumped into a joblib file.

2. **Deploying the Model as a Cloud API:** Amazon Sagemaker was used to deploy the trained model, creating a endpoint for cloud-based API for real-time predictions. The model is loaded using the saved joblib file and set for deployment using sagemaker.SKLearn module. A Endpoint is configured and created and the model is deployed into it.

3. **Creating a Client Application:** A Streamlit web application was built to provide a user-friendly interface. Users can upload executable(.exe) files into the webclient, where it is going extract the needed features from the .exe file using pefile lib and other pretrained data and the features are converted into a Json format and sent to the deployed API. The application then displays the classification results (Malware - Danger or Benign - Safe). For the client deployment I have used Google colab and started the streamlit application in it.




Open
This PC › Downloads
Search Downloads
Organize
New folder
Name
Date modified
Today (1)
.ipynb_checkpoints
01-Apr-24 4:13 AM
Last week (3)
python-3.11.8-amd64.exe
27-Mar-24 2:33 AM
tinytask-1-77.exe
25-Mar-24 5:11 PM
transcripts_with_no_registrar_anonymized
26-Mar-24 3:43 PM
Last month (7)
LockDownBrowser-2-1-2-03-327237565.exe
21-Mar-24 12:42 PM
npp.8.6.4.Installer.x64.exe
13-Mar-24 10:12 PM
7z2401-x64 (1).exe
13-Mar-24 6:43 PM
File name: tinytask-1-77.exe
Application (*.exe)
Upload from mobile
Open
Cancel
Anime Telugu Team
votch
PortalGuard Single...
Other favorites
ce with Streamlit
Browse files
Type here to search
4:42 AM
01-Apr-24
https://stale-months-reply.loca.lt
Text Books
Masters
unfiled
Alpaca - Commissio...
Learn to Code with...
Infosys
SageMaker Inference with Streamlit
Upload .exe file
Drag and drop file here
Limit 200MB per file • EXE
Browse files
tinytask-1-77.exe 35.5KB
Benign - Safe



4. Performance check with EMBER 2018 dataset: A 200 or more malware and benign sample fetched from Ember 2018 dataset and the needed features are extracted from them and sent to the API and recorded the responses to validate the model performance with the Malconv model detection.

Project Results

The project successfully achieved its intended outcomes:

- **Trained Malware detection model:** A well-trained Malware detection model capable of classifying PE files as malicious or benign was developed.
- **Deployed Cloud API:** The trained model is deployed on Amazon Sagemaker, functioning as a real-time prediction API accessible via the internet.
- **Web Client:** A web-based interface created for enduser to upload and check their files malicious or not.

Results:

Accuracy: 0.9746835443037974

Result and Conclusion

The above results shows my models accuracy. A accuracy of 0.9746835443037974 indicates that when the model predicts the type of exe file most of the time correctly, it is correct approximately 97.47% of the time.

Conclusion

This objective of the project to successfully develop and deploy a cloud-based PE static malware detection API was achieved. The project demonstrates the effectiveness of machine learning for malware classification and the power of cloud platforms like Amazon Sagemaker and Google Colab for building scalable and user-friendly applications.

Resources:

- https://github.com/endgameinc/ember
- https://github.com/endgameinc/ember/tree/master/malconv
- https://github.com/UNHSAILLab/S24-AISec/tree/main/Midterm%20Tutorial
- https://sagemaker-examples.readthedocs.io/en/latest/intro.html
- https://sagemaker-examples.readthedocs.io/en/latest/frameworks/pytorch/get_started_mnist_train_outputs.html
- https://docs.aws.amazon.com/sagemaker/latest/dg/deploy-model.html
- https://github.com/RamVegiraju/Pre-Trained-Sklearn-SageMaker
- https://youtu.be/ueI9Vn747x4?si=KSFTvR9hBnU0u0DO
- https://youtu.be/oOqqwYI60FI?si=3WKd-iDz93mm1Vbe
- https://youtu.be/g6kQl_EFn84?si=9MHbO9I52AS2pPjx