

Midterm Project

**AI and CyberSecurity**

**DSCI6015**

**Midterm**

**By  
 Sayesh Kumar Chittoor**

**Under the guidance of**

**Dr.Vahid Behzadan**

# Summary

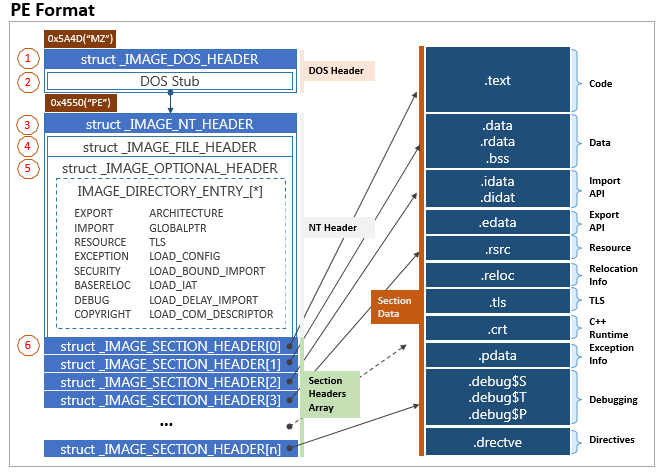
In this we study how AWS Sagemaker is used to develop a cloud-based PE (Portable Executable) static malware detection API Successfully. This API classifies files as malicious or benign using Random Forest binary classifier.

An intuitive web application is developed so that a distant user can submit their executable (.exe) file and discover the hazard. The project was written in Python, and the model generation and implementation were done using ML libraries such as sklearn, pefile, nltk, etc.

# Introduction

## PE Files Format

Operating systems for Windows employ a file format called Portable Executable (PE) to hold executable code and related data. Machine instructions, resources, imported libraries, and metadata are among the crucial details found in these files that are needed for the program to function. For drivers, programs, and dynamic link libraries (DLLs), PE files are frequently utilized. Their arrangement is methodical, featuring headers that offer details on the attributes of the file, like its structure, entry point, and group arrangement. Due to its ability to allow for the inspection and manipulation of executable material, the PE file format is essential for tasks like software analysis, reverse engineering, and virus detection.



## Random Forest Classifier

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. Trees in the forest use the best split strategy, i.e. equivalent to passing splitter="best" to the underlying **[DecisionTreeRegressor](https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html" \l "sklearn.tree.DecisionTreeRegressor" \o "sklearn.tree.DecisionTreeRegressor)**. The sub-sample size is controlled with the max\_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree.

The fundamental idea behind Random Forests is to create a diverse ensemble of decision trees by introducing randomness in both the training and prediction phases. During training, each tree is built using a random subset of the training data and a random subset of features at each split. This randomization helps to reduce the correlation between individual trees, thus lowering the risk of overfitting and enhancing the model's overall performance. During prediction, the outputs of all trees are combined using techniques like averaging or voting, where the most common class (for classification) or the average value (for regression) is chosen as the final prediction. This ensemble strategy helps to mitigate errors and produce more reliable predictions. Random Forests are highly regarded for their versatility and effectiveness across a range of machine learning tasks.

Random Forests offer numerous advantages, making them a favored choice across diverse fields. Firstly, they excel in managing high-dimensional datasets with a multitude of features. Secondly, they exhibit resilience to noise and outliers present in the data. Another noteworthy advantage is their minimal requirement for hyperparameter tuning compared to more intricate models. Additionally, Random Forests yield valuable insights into feature importance, enabling users to identify the most influential features driving the model's predictions. Consequently, Random Forests are extensively applied in domains like finance, healthcare, and bioinformatics, owing to their outstanding performance, scalability, and interpretability.

**Task Completion:**

1. **Build and Train a Model:** Within AWS SageMaker, the training and development phases rely on scikit-learn version 1.2.1. A Random Forest binary classifier is trained using a meticulously labeled dataset comprising binary feature vectors. Following successful training, the resulting model is stored in a joblib file, ready for subsequent utilization.
2. **Deploy the Model as a Cloud API:** The trained model was deployed, creating an endpoint customized for real-time predictions within a cloud-based API. Leveraging the sagemaker. SKLearn module, the model was smoothly loaded from the saved joblib file and made ready for deployment. Following this, an endpoint was configured and activated, simplifying the process of deploying the model onto it, all set to generate real-time predictions.
3. **Create a Client Application:** An interface was created using Streamlit for a web application. Within this interface, users can easily upload executable (.exe) files directly to the web client. After uploading, the application employs the pefile library and other pretrained data to extract the required features from the .exe file. These extracted features are then transformed into JSON format and sent to the deployed API. Subsequently, the application displays the classification results, indicating whether the file is categorized as malware (dangerous) or benign (safe). To deploy the client, Google Colab was utilized to initiate the Streamlit application within it.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. **Performance check with EMBER 2018 dataset:** Over 200 malware and benign samples were obtained from the Ember 2018 dataset. Extracting the essential features from these samples, they were subsequently forwarded to the API. The responses received from the API were documented to assess the model's performance in comparison to the Malconv detection.

# Project Outcomes:

The project achieved successfully with the expected outcomes:

* + **Trained the Malware detection model:** A skillfully trained malware detection model was developed, possessing the capability to differentiate PE files as either malicious or benign.
  + **Deployed Cloud API:** The deployed model is hosted on Amazon SageMaker, functioning as an internet-accessible real-time prediction API.
  + **Web Client:** A user-friendly web interface was developed to facilitate the process for end users to upload their files and determine if they are malicious.

**Results:**

Accuracy: 0.9936842105263158

# Result and Conclusion

The results displayed represent the accuracy of my model. With an accuracy score of 0.9936842105263158, it indicates that the model accurately predicts the type of exe file with an accuracy of approximately 99.36%.

# Conclusion

The project has successfully fulfilled its goal of establishing a cloud-based API for static PE malware detection. It underscores the efficiency of machine learning in classifying malware and highlights the potential of cloud platforms such as Amazon SageMaker and Google Colab in developing scalable and user-friendly applications.