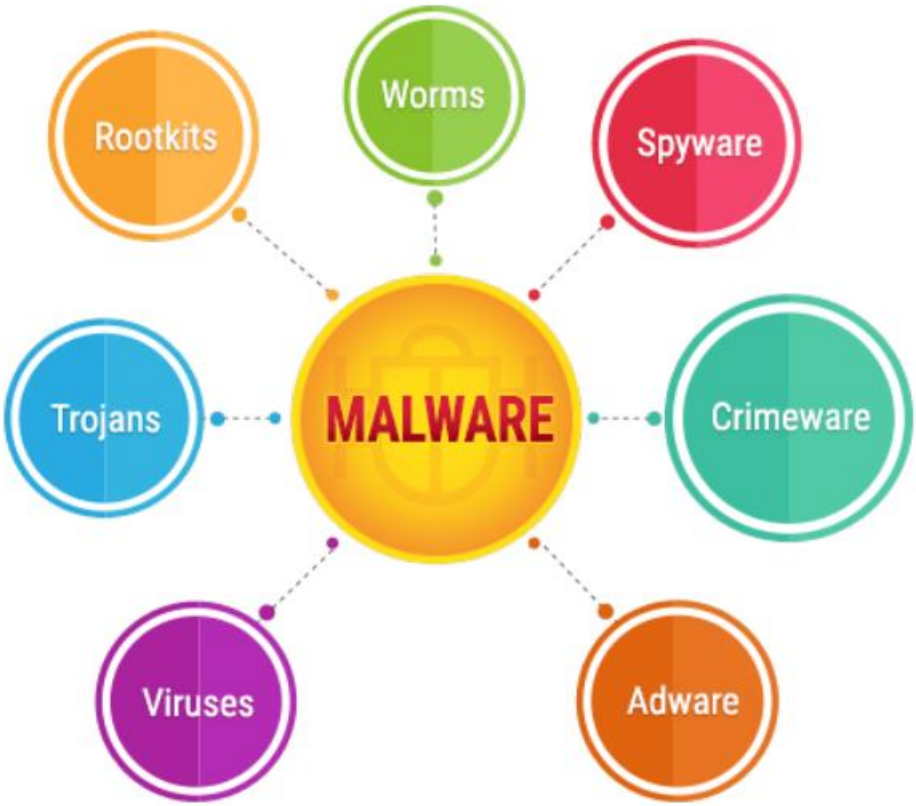
Introduction

This project is going to propose machine learning models to detect malwares using PE headers data set. Some of the most annoying threats in information security are malicious programs, every day, we hear news about data breaches and cyber-attacks with malware. Attackers are enhancing their development skills and building new malware that are able to bypass company safety measures.

Malware are malicious pieces of software that are designed to infiltrate and damage information systems without the users' consent. The term malware covers a lot of categories. There are many different types of malware:



Adware

- Adware (short for advertising-supported software) is a type of malware that automatically delivers advertisements.

Crimeware

- Crimeware is any computer program or set of programs designed expressly to facilitate illegal activity online.

Spyware

- Spyware is a type of malware that functions by spying on user activity without their knowledge. These spying capabilities can include activity monitoring, collecting keystrokes, data harvesting, and more.

Worms

- Computer worms are among the most common types of malware. They spread over computer networks by exploiting operating system vulnerabilities.

Rootkits

- A rootkit is a type of malicious software designed to remotely access or control a computer without being detected by users or security programs.

Trojans

- A Trojan horse, commonly known as a "Trojan," is a type of malware that disguises itself as a normal file or program to trick users into downloading and installing malware.

Viruses

- A virus is a form of malware that is capable of copying itself and spreading to other computers. Viruses often spread to other computers by attaching themselves to various programs and executing code when a user launches one of those infected programs.

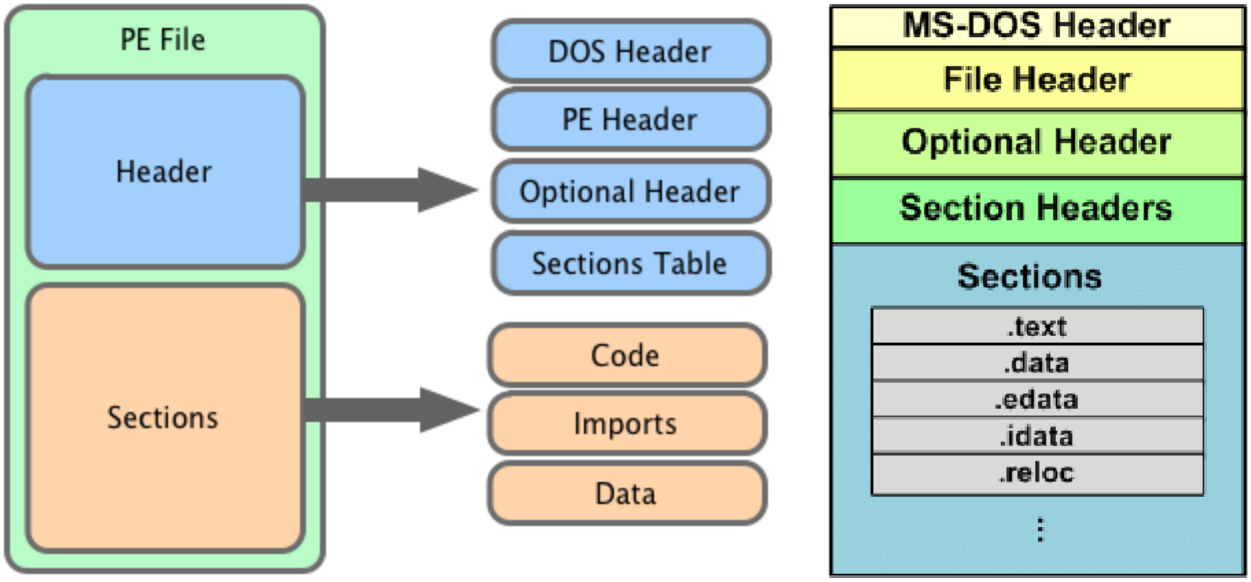
Malware Analysis Technique

Static Malware Analysis is the art of collecting all of the available information about the malicious binary, using different techniques and utilities. In this phase, the I examine the malware without really executing it. I use the static method analysis to examine the file for analysis.

Project Overview

In this Project, I am going to develop machine learning models, train and test them using sample data so that they can be used to detect malwares in future.

First model will be trained using PE header data set. Portable Executable (PE) files are file formats for executables, DDLs, and object codes used in 32-bit and 64-bit versions of Windows. They contain many useful pieces of information for malware analysts, including imports, exports, time-date stamps, subsystems, sections, and resources. The following is the basic structure of a PE file:



Gathering Data & Importing Dataset

Obtaining data is the first step, the training set consisted of 1469 benign and 1361 malicious PE files, fest set contained 367 benign and 12251 malicious executables.

I decided to choose Portable Executable format for simple reasons: Windows is the most common operating system; therefore, the vast majority of malicious files worldwide are targeted towards Windows users. At the same time collecting benign executables for Windows should be fairly easy, therefore the availability of data (files in PE format) is the next reason. Another reason being the usability of methods for detection based on PE header attributes.

As mentioned before, Windows is the prevalent operating system, developing working solution for protection against malicious PE files will affect most users. Third reason is the detailed documentation of this file format.

Below images displays that actual data set had 138k records with 54 columns. After feature selection it has been reduced to 13 attributes,

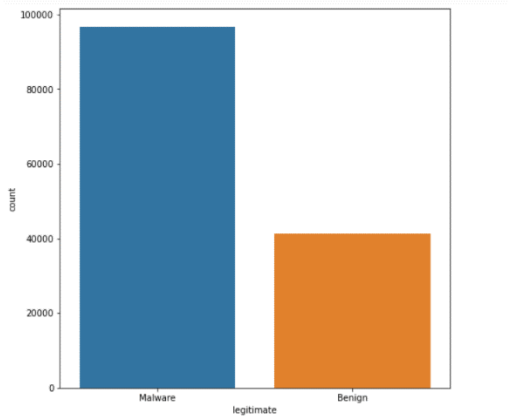
```
print("No of feature",Data.shape[1])
print("No of feature after selection",Data_new.shape[1])
```

```
No of feature 54
No of feature after selection 12
```

Bellow image displays important features after feature selection

```
Features = Data_new.shape[1]                                    # geting the number of important features after PCA
Index = np.argsort(FeatSelect.feature_importances_)[::-1][:Features]   # Indexing the important features
i = 1                                                           # loop counter
for feat  in range(Features):                                   # loop in number of important features
    print(i,MalwareDataset.columns[2+Index[feat]])              # adding two becuase we deleted the first two features
    i = i+1                                                     # loop increment
```

```
1 DllCharacteristics
2 Machine
3 Characteristics
4 SectionsMaxEntropy
5 MajorSubsystemVersion
6 VersionInformationSize
7 Subsystem
8 ImageBase
9 SizeOfOptionalHeader
10 ResourcesMaxEntropy
11 ResourcesMinEntropy
12 SizeOfStackReserve
13 SectionsMinEntropy
14 MajorOperatingSystemVersion
```



Data Preprocessing

Multi-variate feature selection

In multi-variate feature selection, multiple features are evaluated at the same time, These methods measure the added value of particular feature subset for classification. The most straightforward multi-variate selection would be training the classifier with each feature subset. In several methods are introduced. For instance, I used the widely used method called Principal Component Analysis (PCA). In PCA, new attributes are computed as the linear combinations of original attributes to provide more information in less attributes.

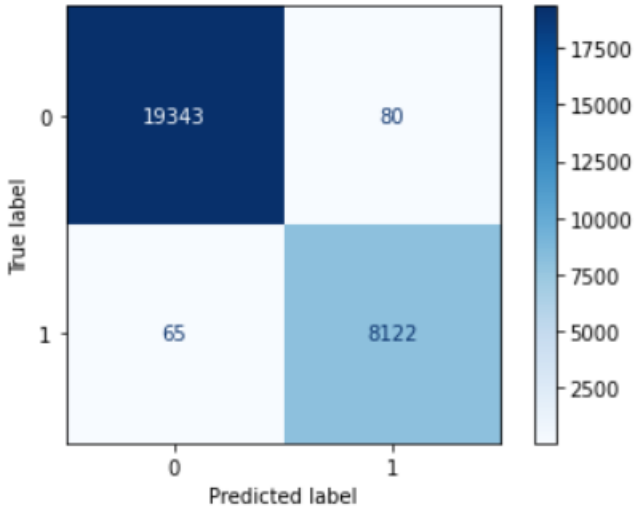
Model Training

I tried the multiple machine learning algorithm for the problem, but accuracy of Random-Forest algorithm was best of all. So, I will go with Random-Forest to gain the best accuracy for the model. I tarin the model of 80% of the dataset and 20% was reserve for testing.

Testing

Once the model is trained, it is tested on the testing data set and the results are recorded. Several metrics are used to evaluate how well the classifier performed on given testing data set. In this section I will describe those that I used in the performance evaluation.

FP, FN, TP, TN
There are four possible results of detection process:
True positive (TP) - Malicious PE file correctly classified as malware
True negative (TN) - Benign PE file correctly classified as benign
False positive (FP) - Benign PE file incorrectly classified as malware
False negative (FN) - Malicious PE file incorrectly classified as benign



Feature Extraction of PeFile

After I successfully trained the model, I needed to extract features that are significant for malware detection. To extract these attributes from a PE file, I chose to build my own python modules built on pefile library. Extraction of PE header attributes was done in two steps. The first step comprised of extracting all the PE header attributes from the file by the extractor tool. The only relevant PE header was then selected, as described in dataset. In the second step, only relevant headers were extracted and new data sets for classifier training were created and saved in the list object.

Malware Prediction

In this part we give a file name as parameter to PE extractor function to get the list relevant header information. Then this list object is converted to NumPy array to make prediction.