Malware Detection Tool using Machine Learning

By Kunwarvir Singh & Niraj Malpani

Problem Overview –

Malware is any software intentionally designed to cause damage to a computer, server, client, or computer network. It is collective term that includes Malicious Files, Malicious URL's etc. Various kinds of Malware exist like Worms, Trojan horses, Ransomware, Adware, Spyware. Malware is typically delivered in the form of a link or file over email and requires the user to click on the link or open the file to execute the malware. Malware is used to extract secured information from Users and is mostly used by Hackers for illegal activities.

Detection and Prevention to the entry of Malicious files into the computer system is necessary. Thus, using the techniques of Machine Learning our main idea is to design a model that could detect any Malicious Software that enters into the system prevent it from its operation.

Solution Overview –

Before proceeding further into the implementation of Machine Learning Model, it is important to know what kind of files are considered as malicious. As stated, these malicious files that can threaten the security and privacy of any Information System. Each kind of file has a certain file format along with many other factors that allow machine learning model to classify it as either Malicious or Legitimate. Observing the trends of virus attack into the systems by Malicious Files. It is found that such files are mostly in the form of PE (Portable Executable) Format Files.

PE Files - The Portable Executable (PE) format is a file format for executables, object code, DLLs and others used in 32-bit and 64-bit versions of Windows operating systems. The PE format is a data structure that encapsulates the information necessary for the Windows OS loader to manage the wrapped executable code.

```
.acm, .ax, .cpl, .dll, .drv, .efi, .exe, .mui, .ocx, .scr, .sys, .tsp
```

File Extensions of PE Format

As stated in the research paper on "*Statistical Detection of Malicious PE-Executables for Fast Offline Analysis"* by *Ronny Merkel, Tobias Hoppe, Christian Kraetzer, Jana Dittmann*, "PE Format files represent the biggest fraction of today's Malicious Code". Thus, the main medium for transferring Malware is in the form of PE Format Files.

So, our Machine Learning model is only accustomed to classify and detect PE Format files as Malicious or Legitimate.

Recent research also indicates that nowadays malware is being transferred through various other forms of files like PDF's. The variables are different for each kind of file and are still in the process of research. For now, on the basis of availability of dataset we are limiting our ML model to PE Format files only.

The details of the project are as follows: -

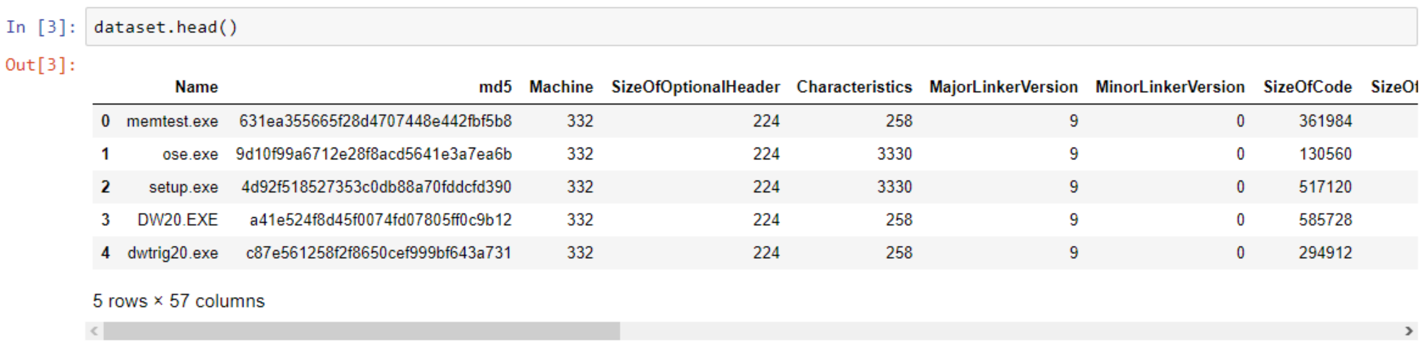
Platform – Jupyter Notebook

Language – Python (3.7)

Important Libraries – Scikit Learn, Numpy, Pandas, Pickle, Pefile, OS, ArgParse, win10toast

Project Walkthrough -

Dataset – The dataset is available in various public domain and is provided by "*Virusshare.com*" – A repository of malware samples to provide security researchers, incident responders etc. access to sample of live malicious code. The dataset consists of 138048 data entries listing PE Information from Various sources. Out of these 96724 are malicious while 41323 are legitimate. The dataset consists of total 57 features for every File.



```
In [3]: dataset.head()
Out[3]:
```

	Name	md5	Machine	SizeOfOptionalHeader	Characteristics	MajorLinkerVersion	MinorLinkerVersion	SizeOfCode	SizeOf
0	memtest.exe	631ea355665f28d4707448e442fbf5b8	332	224	258	9	0	361984	
1	ose.exe	9d10f99a6712e28f8acd5641e3a7ea6b	332	224	3330	9	0	130560	
2	setup.exe	4d92f518527353c0db88a70fddcfd390	332	224	3330	9	0	517120	
3	DW20.EXE	a41e524f8d45f0074fd07805ff0c9b12	332	224	258	9	0	585728	
4	dwtrig20.exe	c87e561258f2f8650cef999bf643a731	332	224	258	9	0	294912	

5 rows × 57 columns

Taking all features into consideration may lead to overfitting and may produce poor results. In order to find which feature, contribute significantly towards building the model, we employed feature selection algorithm using *Extra Tree Classifier*. On executing the algorithm, we found that out of 57 features only 13 features were important. Hence only 13 these features were taken into consideration while building the model.

The 14 features are listed as follows: -

Features	**Importance**
DllCharacteristics	0.247721
Characteristics	0.121755
VersionInformationSize	0.104208
SizeOfOptionalHeader	0.063926
Subsystem	0.056308
ImageBase	0.056308
MajorSubsystemVersion	0.034867
ResourcesMaxEntropy	0.032489
MinorLinkerVersion	0.032356
SectionsMaxEntropy	0.032356
Machine	0.027969
ResourcesMinEntropy	0.022495
SectionsMinEntropy	0.019386

Thus, Feature Selection Analysis shows that 13 Features out 57 Features contribute significantly in deciding whether the file is legitimate or malicious.

Implementation –

The important steps involved in implementing the machine learning model are described as follows: -

Step 1: Collecting and Importing Dataset: Dataset described above is imported using *Pandas* library.

Step 2: Data Pre-processing: The CSV files contain unimportant parameters like Name of the file, md5 of file that needs to be removed before training the model from its features. So, the *Independent variable* consist of 13 features listed above while *Dependent variable* consist of binary digit 1/0 (Legitimate or Malicious).

Step 3: Splitting the dataset: We have used the *train_test_spli*t function from *Scikit Learn* Library to split our dataset in the ratio 1:4 as Test set and Train Set respectively.

Step 4: Building the Training Model: Dataset is fit into5 different machine learning models using *Scikit Learn* library function. The models under which testing has been done are as follows: -

- Random Forest Classifier: n_estimators = 50
- Decision Tree Classifier: Max_depth = 10
- Gradient Boosting Classifier: n_estimators = 50
- GaussianNB
- Linear Regression

Results Obtained are as follows: -

Model	Accuracy	Time Complexity
Random Forest	99.47%	O(v*log(n)) n-Records, v-Attributes
Decision Tree	98.99%	O(n*v*d) n-Records, d-Depth, v-Attributes
Gradient Boosting	98.88%	-
GaussianNB	69.85%	O(v*k) v- Attributes, K- No. of Labelled Classes
Linear Regression	53.67%	-

From above analysis, '*Random Forest Classifier*' gives best accuracy and acceptable Time Complexity. Thus, we will employ *Random Forest Classifier* as mode of Classification of files as Malicious or Legitimate.

Step 5: Calculation of False Positive and False Negative Rate: Taking Random Forest Model as our classifier we calculate the *False Positives* and *False Negatives* of the trained model by making the *Confusion Matrix*. Apart from Accuracy it is also necessary that for any classifier to be acceptable, false positives and false negatives must be as low as possible. The results obtained are as follows: -

```
False positive rate: 0.082710 %
False negative rate: 0.196017 %
```

Step 6: Final Step – Saving the model: Finally, after training the model, the trained model is saved using *JOBLIB* and *PICKLE*. Classifier Model is saved using *JOBLIB* while Features that are used are saved using *PICKLE*. Both files are saved in the file extension of "*.pkl*". This saved model is further used for predicting new results as well as for deploying it in real time platform.

Practical Implementation –

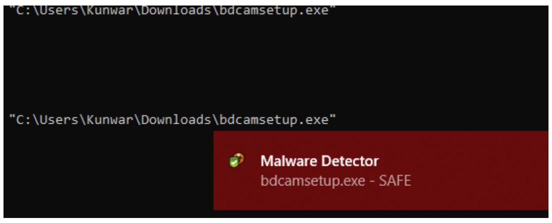
To implement it practically with any random PE format file, it is necessary to extract the features from a file before sending it for predictions. After the required features are extracted, predictions are made using the saved model.

Feature Extraction - For feature extraction we have used a *Pefile* library function. The PE file format is a data structure that contains the information necessary for the Windows OS loader to manage the wrapped executable code. With the help of this function we have extracted the details of 13 features as listed above.

Finally, after Feature extraction we load our model and pass the extracted features to the loaded model for prediction.

The entire code used for Feature extraction is available in the *File_Tester.py* file.

User Interface and Notification System: -



```
"C:\Users\Kunwar\Downloads\bdcamsetup.exe"

"C:\Users\Kunwar\Downloads\bdcamsetup.exe"
```

Malware Detector
bdcamsetup.exe - SAFE

Windows Notification

The UI for the above Malware Testing model has been deployed on local host using FLASK as well as on GUI using PyQt5 platform. We have built a WINDOWS Notifier system with *win10toast* library. On executing the final model on Testing File, Windows Notification pops up notifying whether the file is Malicious or Legitimate.

****