

**MALWARE ANALYSIS**

**PROJECT REPORT**

**BY:**

**KARTIK BANSAL**

**2K18/IT/064**

**KUNAL TANEJA**

**2K18/IT/067**

**SUBMITTED TO:**

**Dr. Kapil Sharma**

**Table of Contents**

[1. INTRODUCTION 1](#_Toc5847)

[1.1 What is Malware? 1](#_Toc14133)

[1.2 What is Static Malware Analysis? 2](#_Toc29466)

[1.3 Objective 2](#_Toc7968)

[1.4 Problem Statement 2](#_Toc6109)

[1.5 How this problem is related to this Class? 2](#_Toc9069)

[2. DATASETS USED 3](#_Toc6868)

[2.1 Top 1000 PE imports [1] 3](#_Toc13321)

[2.2 Raw PE as Image [2] 3](#_Toc16669)

[2.3 PE Section Header [3] 4](#_Toc19186)

[3. PREVIOUS WORK 5](#_Toc10417)

[4. METHODOLOGY 6](#_Toc12909)

[4.1 Data Analysis and Preprocessing 6](#_Toc11841)

[4.1.1 Top 1000 API call imports 6](#_Toc10714)

[4.1.2 Raw PE as Image 7](#_Toc31550)

[4.1.3 PE Section Header 8](#_Toc17973)

[4.2 IMPLEMENTATION DETAILS 11](#_Toc17626)

[4.2.1 Implementation Environment 11](#_Toc4992)

[4.2.2 Pseudo Code 11](#_Toc19516)

[4.3 Program Flow Chart 12](#_Toc26289)

[5. RESULT AND OBSERVATIONS 13](#_Toc28999)

[6. CONCLUSION AND FUTURE WORK 14](#_Toc3030)

[7. REFERENCES 15](#_Toc29857)

1. **INTRODUCTION**
   1. **What is Malware?**

Malware refers to any software/Executable that is designed to harm any computer or a computer network. There are many types of malware including viruses, worms, Trojan horses, ransomware, spyware, etc.

When we download any executable from the internet then we might not know whether it is safe to execute that software/executable file on our system or not. So, in order to validate the executable, various malware analysis techniques are deployed.

Malware analysis techniques can be classified into two broad categories:- Static and Dynamic analysis.Static analysis involves analysis without actually executing the software.Thus it doesn’t involve the risk of malware affecting our system and can be easily performed.Whereas Dynamic analysis involves actual execution of the program and thus we require a safe environment/ sandbox to perform the analysis, which is a little bit hefty.

So, in our project we only focused on static malware analysis and tried to classify any executable into malware/goodware.

* 1. **What is Static Malware Analysis?**

Static analysis involves analysis of executable or software without actually executing it on the system.In Static Analysis we try to focus on the static parameters of an executable which generally includes API call imports, Hash Digest(md5sum, SHA1, SHA256,etc), PE section header parameters, strings, etc.

Advantages of using Static Malware Analysis are:

1. No requirement of any safe environment/ sandbox.
2. Faster as compared to Dynamic analysis.
3. We study the code of the program rather than the effects of executing it on the system.

However, it might not prove as efficient as dynamic analysis for Packed and compressed files.

* 1. **Objective**

To design a GUI tool for client so that he can easily upload a windows executable and perform Static Analysis and classify the executable into malware/goodware so that client easily gets to know about the nature of the executable and investigate the file for further analysis.

* 1. **Problem Statement**

Given a windows executable we need to extract Static parameters such as API call imports, md5 hash, entropy, virtual size, size of raw data, etc. and pass these inputs to our trained mathematical model which will finally classify the file into goodware/ malware. Along with these give the client a brief idea of static parameters of file in a presentable form.

* 1. **How this problem is related to this Class?**

Since our project mainly focus on static malware analysis technique which is a significant part of Malware Analysis field that is directly related to this class and hence our project is also related to this class and very important to dig deep in.

1. **DATASETS USED**
   1. **Top 1000 PE imports** [1]

It contains static analysis data: Top-1000 API calls imported and are extracted using pefile python module. PE malware examples were downloaded from virusshare.com. PE goodware examples were downloaded from portableapps.com and from Windows 7 x86 directories.

FEATURES:

Column 1: hash

Description: MD5 hash of the example

Type: 32 bytes string

Column 2: GetProcAddress

Description: Most imported function (1st)

Type: 0 (Not imported) or 1 (Imported)

…

…

…

Column 1001: LookupAccountSidW

Description: Least imported function (1000th)

Type: 0 (Not imported) or 1 (Imported)

Column 1002: malware

Description: Class

Type: 0 (Goodware) or 1 (Malware)

* 1. **Raw PE as Image** [2]

It contains static analysis data: Raw PE byte stream rescaled to a 32 x 32 greyscale image using the Nearest Neighbor Interpolation algorithm and then flattened to a 1024 bytes vector.

FEATURES:

Column name: hash

Description: MD5 hash of the example

Type: 32 bytes string

Column name: pix\_0

Description: The first greyscale pixel value

Type: Integer (0-255)

…

…

Column name: pix\_1023

Description: The last greyscale pixel value

Type: Integer (0-255)

Column name: malware

Description: Class

Type: 0 (Goodware) or 1 (Malware)

* 1. **PE Section Header** [3]

It contains static analysis data PE Section Headers of the .text, extracted by using python pefile module.

FEATURES:

Column name: hash

Description: MD5 hash of the example

Type: 32 bytes string

Column name: size\_of\_data

Description: The size of the section on disk

Type: Integer

Column name: virtual\_address

Description: Memory address of the first byte of the section relative to the image base

Type: Integer

Column name: entropy

Description: Calculated entropy of the section

Type: Float

Column name: virtual\_size

Description: The size of the section when loaded into memory

Type: Integer

Column name: malware

Description: Class

Type: 0 (Goodware) or 1 (Malware)

1. **PREVIOUS WORK**

We mainly focused on the work done on Static analysis as shown in by Angelo Oliveira [4].

First technique which was proposed by the author was based on the top thousand API call imports made by the executable as the parameter. We tried to replicate this technique using a Neural Network Binary Classifier which gave a quite good performance(as shown further).

Second technique which was proposed by the author was based on Converting the Raw PE byte stream into a GrayScale Image and then rescaling the image into 32x32 dimension using Nearest Neighbour Interpolation Algorithm. Then try to generalize the pattern using different algorithms, but no significant results were shown.

The third technique that was proposed by the author involved classification of executable into malware/ goodware based on the PE File .text section Header Parameters. We tried to analyse the malware dependency on these parameters using different plots, and further extracted most relevant features from the dataset.

1. **METHODOLOGY**
   1. **Data Analysis and Preprocessing**

In this section we analyzed all the datasets by visualizing data through various plots and normalizing the data wherever required, also we discussed about the input extraction for each technique from executable using pefile python module.

* + 1. **Top 1000 API call imports**

In this technique we implemented a neural network model for the task of classification, which takes an input of array of shape (1000, 1) where the ith element of the input array is set to 1 if uploaded executable imports that particular API corresponding to the ith column of the dataset, otherwise set to 0.

So firstly we prepared the list of API call imports read from PE file imports table using pefile module and in those API calls which belong to the dataset we set there corresponding indexes to 1 in the input array.

The image below shows the distribution of the dataset which was used for training the model. Y-axis(vertical) shows the count of data and bars corresponding to 0 and 1 value represents the goodware and malware respectively.

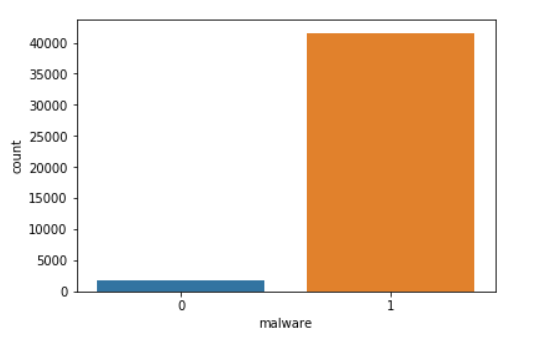


Image 1: Distribution of Dataset

Although the dataset used by the author was unevenly distributed but we trained the model using weighted loss function to penalize the model evenly for wrong prediction.

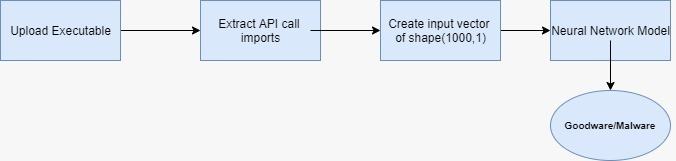


Image 2: flow of classification using API calls

* + 1. **Raw PE as Image**

This technique focuses on extracting the raw PE byte stream and then converting it to grayscale image of 32x32 using Nearest Neighbour Interpolation Algorithm.

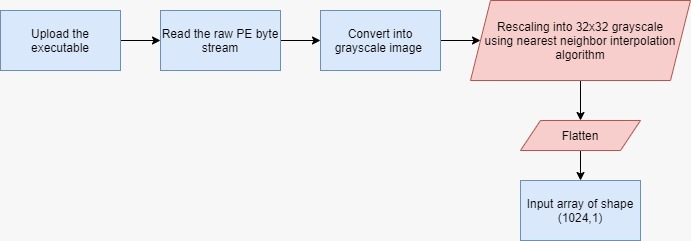


Image 3: flow of the image generation from raw PE byte Stream

Then we applied a Neural Network Model on this 32x32 image. The result were very horrific and model clearly showed over-fitting towards the malware class.This problem can be accounted by either uneven distribution of the data or inadequate feature extraction technique.

The results can be seen in the confusion matrix below.

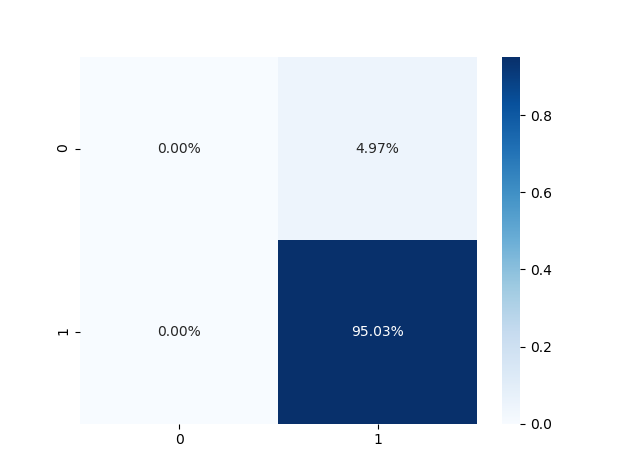


Image 4: confusion matrix for the prediction of the RawPeImage model

* + 1. **PE Section Header**

This Technique involves reading the contents of .text section of PE file and dataset comprised of 4 parameters namely virtual size, size of raw data, virtual address and entropy.

Analysis of Virtual address through plots showed that Virtual address remained constant for almost all the samples and therefore it doesn’t prove to be a good feature for the classification and hence was removed.

The following bar graph shows variation of virtual address for different sample.

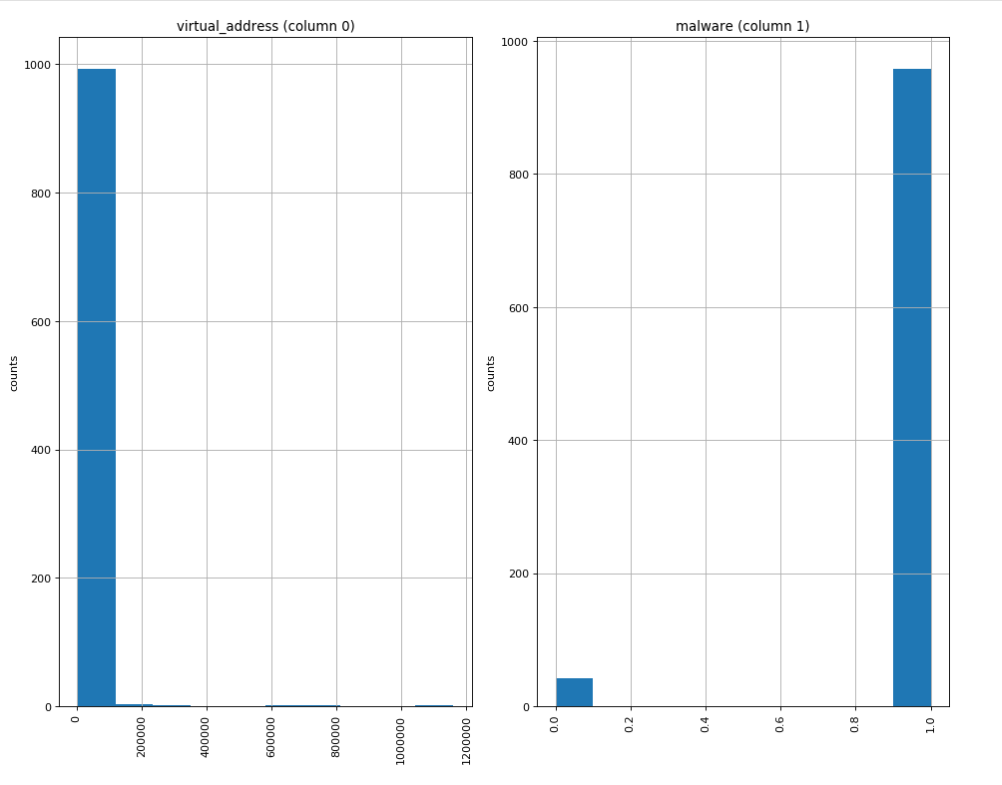


Image 5 : variation of Virtual Address for different sample.

From the analysis of Virtual size and size of raw data we found out that for most of the malware samples virtual size was found to be greater than the size of raw data, which might represent the packed/compressed malware.

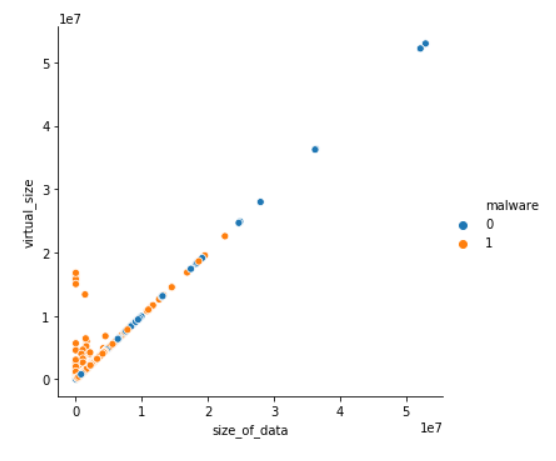


Image 6 : relation of virtual\_size and size\_of\_data for dataset

Thus we derived a new feature ‘size\_diff’.

***Size\_diff = virtual\_size - size\_of\_raw\_data***

Analysis of entropy, gave us a hint about the variation of the entropy for malware and goodware.

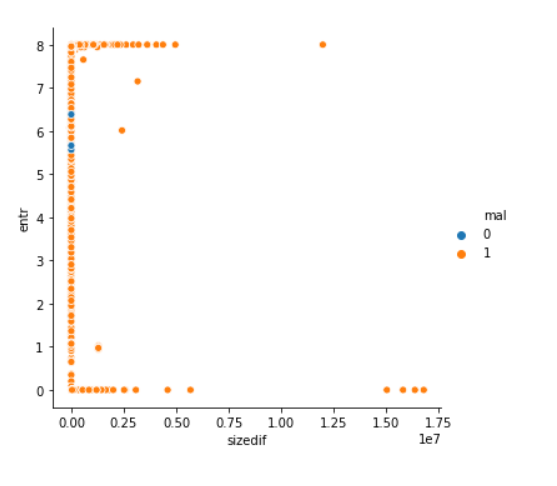


Image 7: variation of entropy with size for dataset.

Along with analysis, we referred to the research done by **ROBERT LYDA, JAMES HAMROCK** [5].

Which clearly showed the entropy range for various executables.

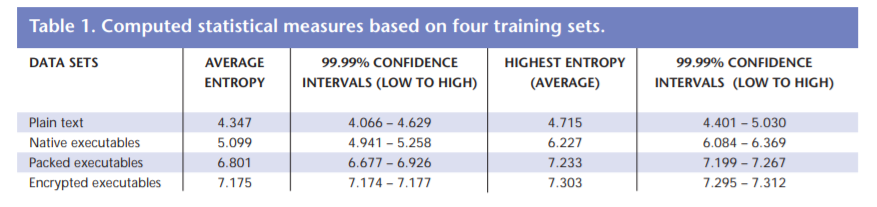


Image 8 : Entropy for various executables as shown in [5].

Thus from both the analysis we concluded that if entropy of the file lie in the range of 4.5 - 6.5, then it most probably a good ware.

Thus we normalized this dataset into 2 parameters ‘size\_diff’, ‘entropy’.

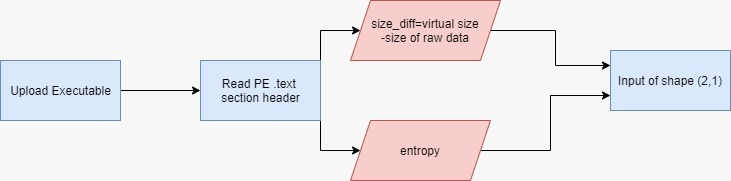


Image 9 : flow of input extraction from PE section Parameters dataset.

* 1. **IMPLEMENTATION DETAILS**
     1. **Implementation Environment**

|  |  |
| --- | --- |
| Programming Language(s) | Python 3 |
| Operating System | Windows |
| Library Packages | pefile, tkinter, numpy, pandas, pillow, tensorfow, matplotlib, sklearn, seaborn. |
| Interface Design(GUI/ web/others) | GUI using tkinter. |

* + 1. **Pseudo Code**

1. Upload the executable and extract all the inputs which includes API call imports, PE section header info, md5 sum hash digest.

2. Compare current md5 hash with hashes of the malware present in the dataset. If found return malware. Else go to step 3.

3. Run Neural network model for the API call input and store result in x.

4. If size\_diff > 0 and 4.5 > entropy or entropy > 6.5 then increase the probability for malware by 25%. else if 4.5 <= entropy <=6.5 then decrease the probability of malware by 25%.

5. if probability of malware is greater than equal to 50% then return malware else return goodware.

* 1. **Program Flow Chart**

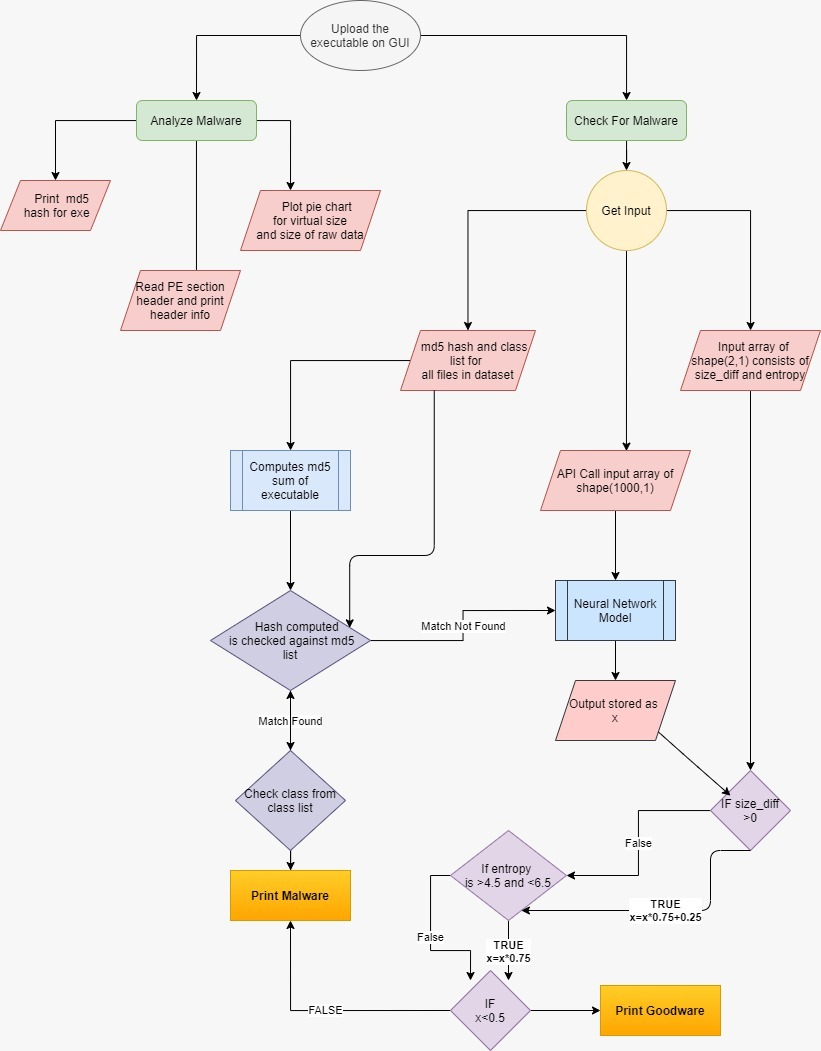


Image 10: Program Flow chart

1. **RESULT AND OBSERVATIONS**

The author of the paper by Angelo Oliveira [4]proposed these techniques as individual classification methods.

We incorporated relevant methods to form a better and more flexible model that have benefits of different aspects stated by the author.

This new model performed good enough with the accuracy of 98.8%.

This new model has increased flexibility which is shown by the precision values of 98.9% and 95% for malware and goodware respectively.

The below confusion matrix represent the prediction of the model on the dataset provided by the author. Where ‘0’ represents the goodware and ‘1’ represents the malware.

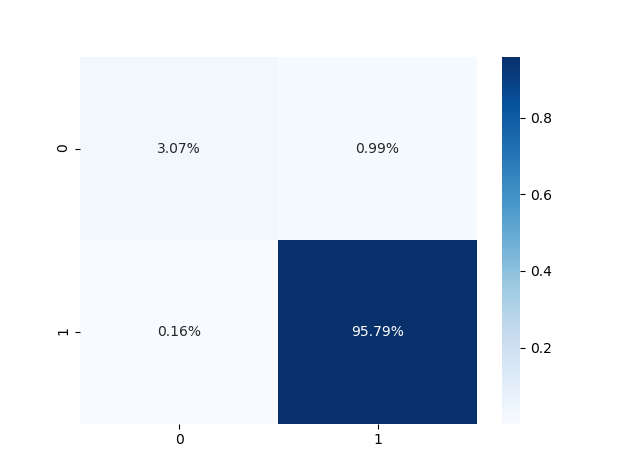


Image 11 : Confusion matrix for the prediction on the dataset

1. **CONCLUSION AND FUTURE WORK**

Our model gave a significantly good result, but we still believe that is there is a lot of scope for improvement since data was highly imbalanced. Techniques and features generated by the author were relevant upto some extent but are not much significant in classification of the malware.

Over all we believe that our software acts as good tool to perform basic static analysis and provide intuition of the malware.

However there are some areas which can be improved and optimized for outstanding results such as

* Dynamic Analysis technique such as classification on the basis of API call sequence and behavioral graphs can be generated during execution of file,which provides deep insight into the working of the executable
* For packed/ confiscated malware we could include a packer detection code and analyze the file after unpacking with suitable tool.
* For malwares which import API during runtime, presence of specific API calls such as GetProcAddress() can be detected using static analysis and therefore dynamic analysis for such executable should be preferred over static analysis.
* By using strings module we could generate files comprising of raw strings included in executable that might contain some malicious IP showing connection to a remote host thereby indicating network malware.

# REFERENCES

|  |  |
| --- | --- |
| [1] | A. Oliveira, "Malware analysis dataset: top-1000 PE imports," 2019. |
| [2] | A. Oliveira, "Malware Analysis Datasets: Raw PE as Image," 2019. |
| [3] | A. Oliveira, "Malware Analysis Datasets: PE section headers," 2019. |
| [4] | A. Oliveira, 2019. [Online]. Available: https://ieee-dataport.org/authors/angelo-oliveira. |
| [5] | L. ROBERT and J. HAMROCK , "Using Entropy analysis to find encrypted and packed malware," 2019. [Online]. Available: http://index-of.es/Viruses/U/Using%20Entropy%20Analysis%20to%20Find%20Encrypted%20and%20Packed%20Malware.pdf. |