Malware Prediction

on basis of research done by
Angelo Oliveira

By:

KUNAL TANEJA

2K18/IT/067

KARTIK BANSAL

2K18/IT/064

Angelo Schranko de Oliveira

Cyber Security Specialist

Research and Dataset of current work:
Research Paper and Dataset

Malware Prediction

using API calls

MALWARE and Relation with API CALLS

What is Malware?

Malware, or malicious software, is any program or file that is harmful to a computer user.

How APIs are related?

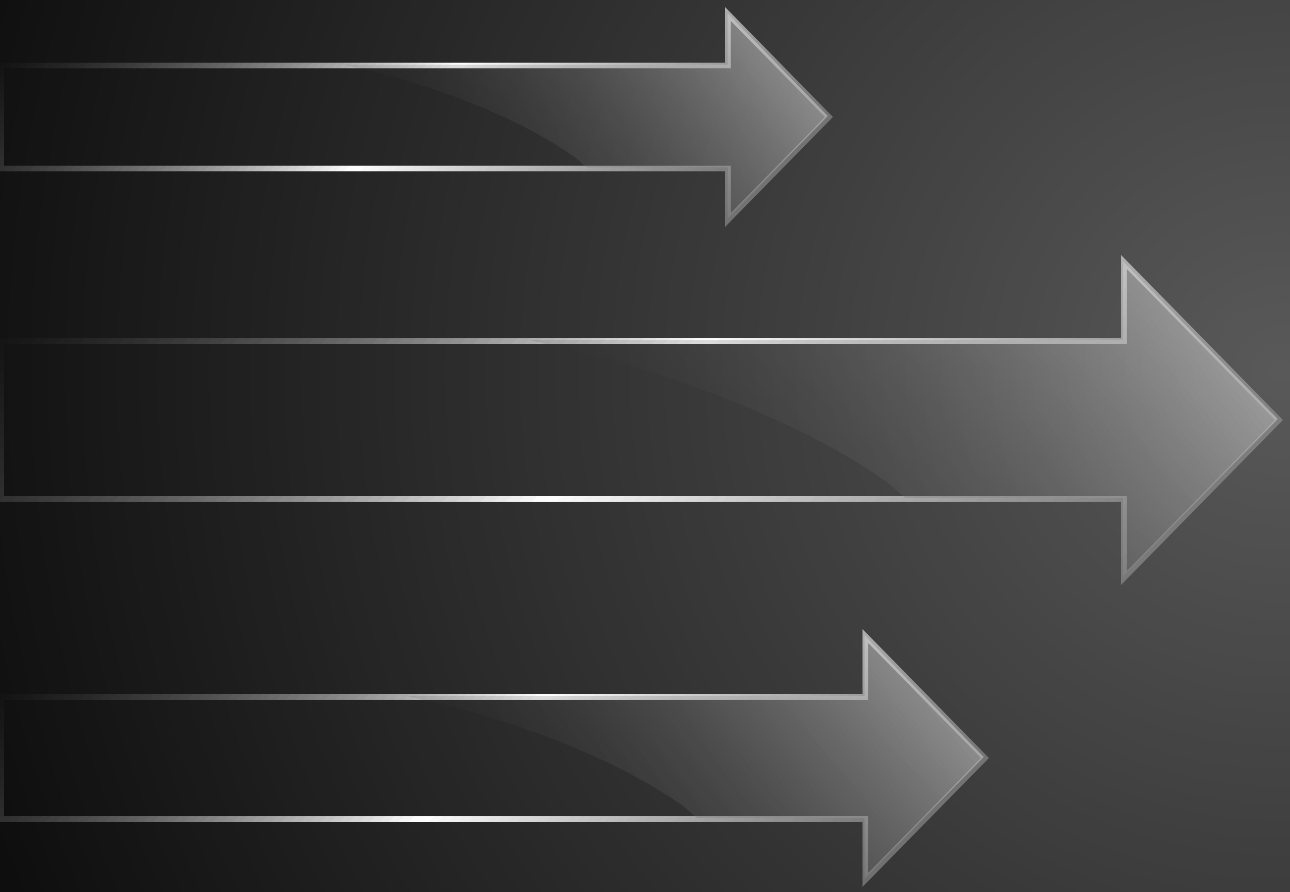
Every program uses api calls to request resources and perform tasks. Even editing files need an api call to write in the file. Malwares too use api calls to modify the system information or for other purposes.

TARGET

How API calls distinguish malwares

We know that malware uses API calls to retrieve/modify information or to access resources. There may exist some specific pattern of API calls which are common among set of malwares. We target in finding those patterns.

Steps towards the Goal



We analysed top 1000 API calls which are common among the malicious executables.

For the given test executable file, we extracted API call imports and match them with set of top-1000 API calls dataset that we have.

Then we try to find the pattern among the different set of API calls. For this we use Neural Network which can be used to generate high dimensional non-linear function.

Some API calls in our List

- NtOpenThread
- WSAAccept
- CopyFileA
- DeviceIoControl
- CryptUnprotectMemory
- SetWindowsHookExW
- DeleteService
- GetSystemInfo

Malware Prediction

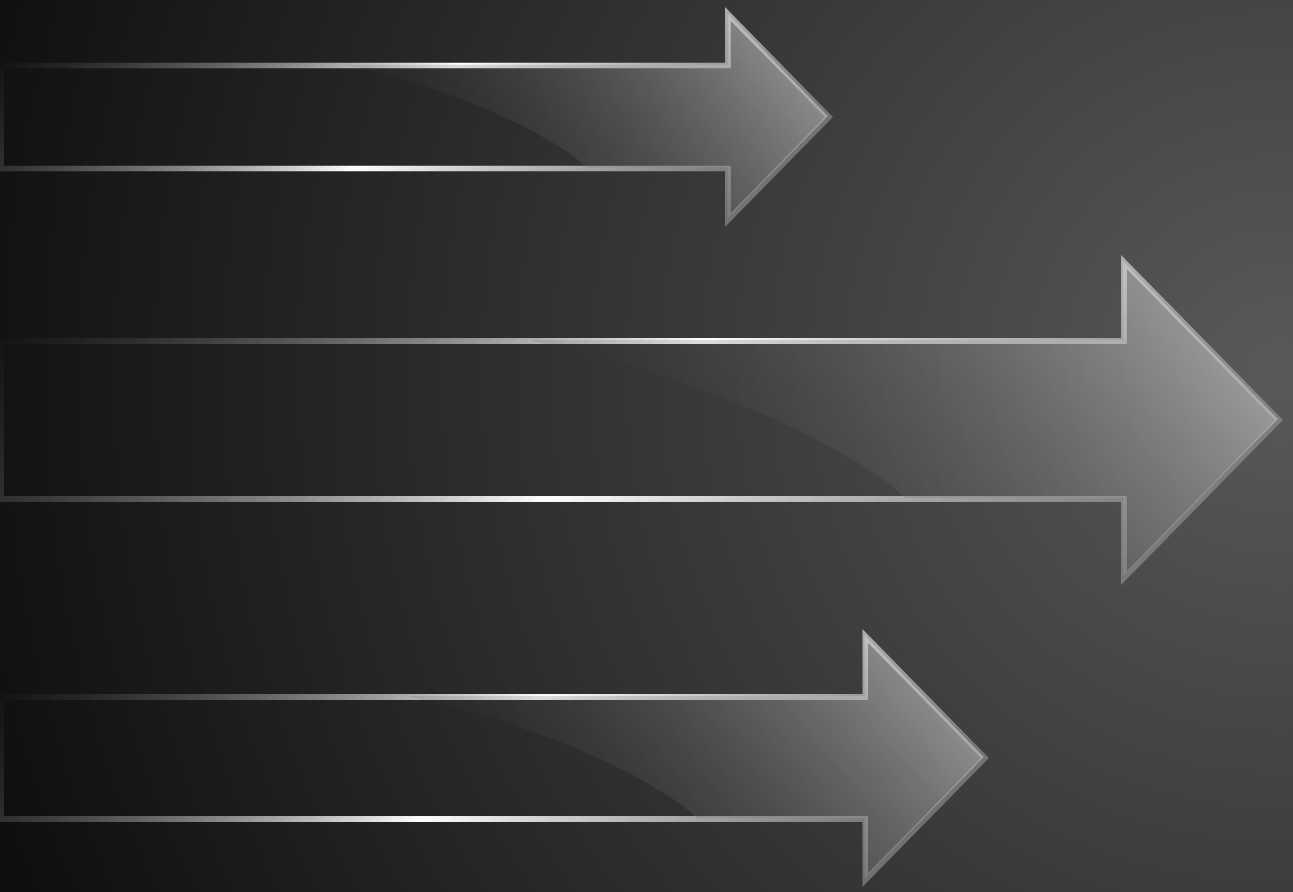
using Raw PE as Image

TARGET

How Image of Raw PE section works

Angelo in his work converted raw byte code of PE file and converted it in a grayscale image of size 32x32 using Nearest Neighbour Interpolation. He assumed that image could give a overview of byte stream of PE file.

Implementation

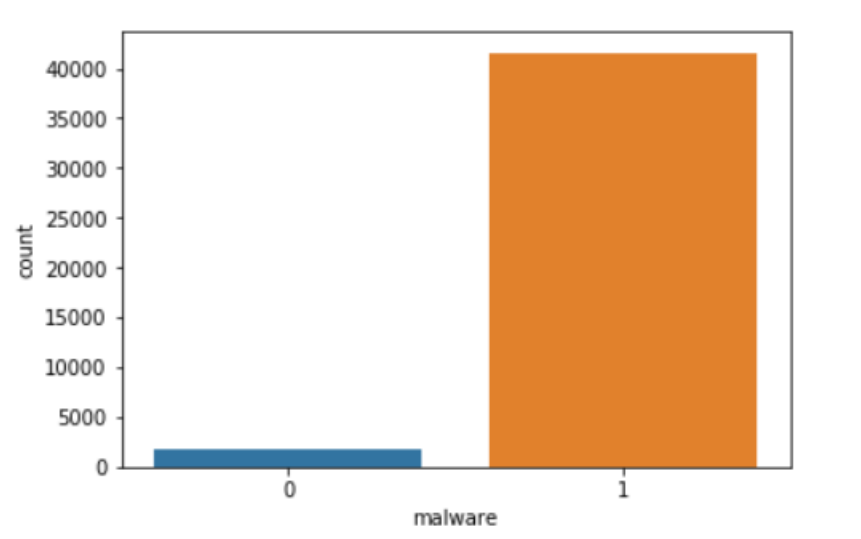


Byte code of PE section is retrieved from the executable file. Length of the byte code is stored in a variable n.

Then this byte code is converted to a matrix of shape (sqrt(n), sqrt(n)) this represent the image of the PE section Header

To reduce the computational cost and to generalise the image, this image is rescaled to smaller image of shape (32,32) using Nearest Neighbour Interpolation Algorithm.

Problems and drawbacks of this approach



- The major problem with the approach was the unevenly distributed dataset. Dataset was comprising only about 2200 non malwares and 40000+ malwares. This suppressed the potential difference between malwares and goodwares.

- Secondly, what we saw while going through the work was that there was actually no potential weightage in this image vector. they were just random for each data. So it was really difficult to differentiate between malware and goodware on basis of byte code image.

Conclusion of this approach

Creating an image out of the bytecode was just an arbitrary method which author thought would work, but it failed in such a manner that the model(which was used to classify) predicted every file as malicious. Fact that dataset was strongly biased towards malwares cannot be ignored and plays important role in bad performance of the model. A more balanced dataset in future might lead to satisfactory results for this approach.

Malware Prediction

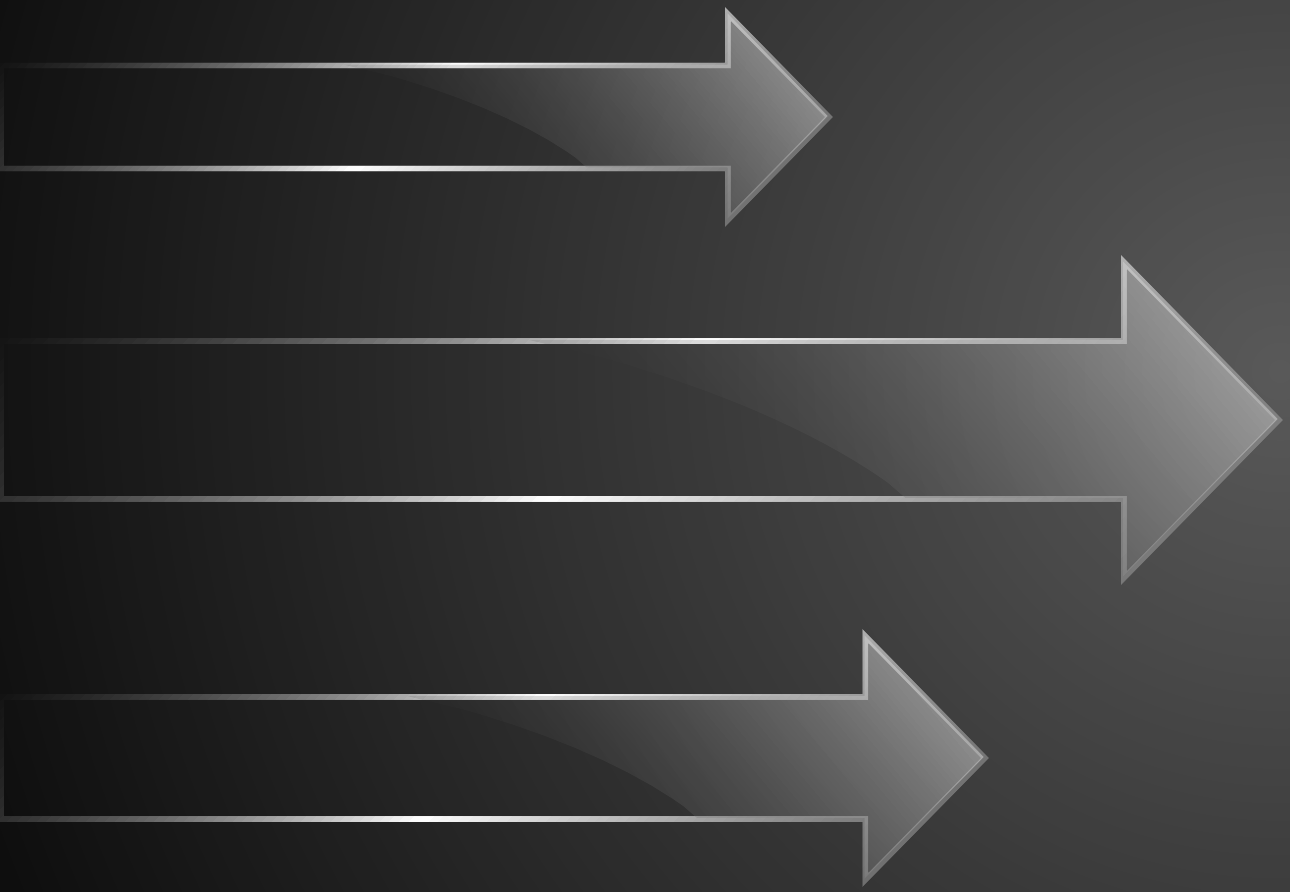
using PE section Header

TARGET

PE section header

PE section header includes virtual size ,size of raw data and entropy that mostly differs for the packed and unpacked executables. This observation gave author a spark to analyse the executable for goodware/malware.

Implementation

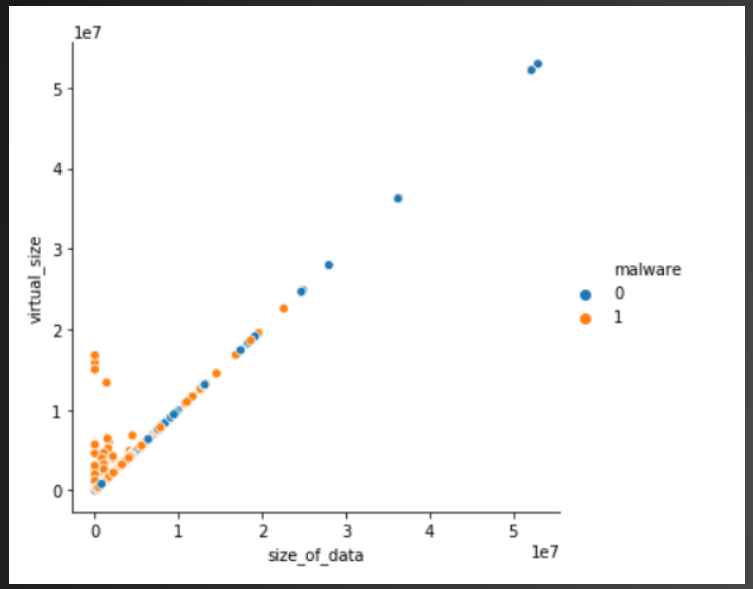


Virtual Size , size of raw data and entropy of the PE section Header is retrieved from the executable.

Then the difference of virtual size and raw data size is considered and store as a new parameter named size_diff.

Entropy of the PE section Header also gave some intuition about the maliciousness of the file since it represents randomness in byte code.

Observation

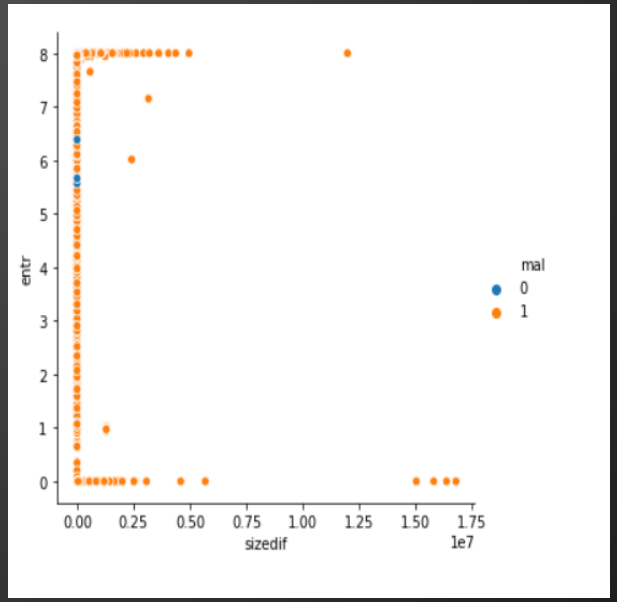


SIZE DIFFERENCE

we saw that difference between virtual size and actual size was generally greater than 0 in case of malicious file and it is close to 0 in case of normal file.

ENTROPY

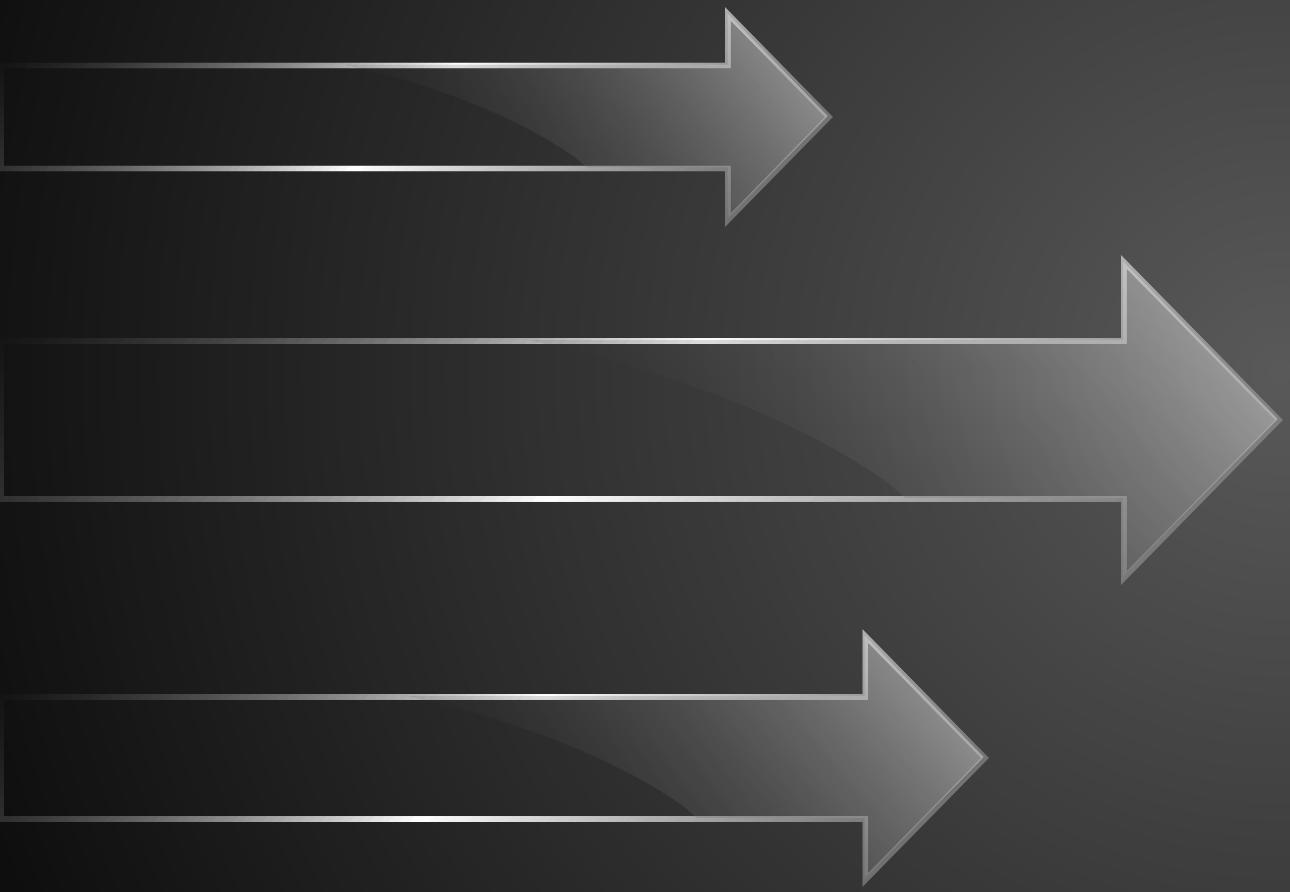
From the plot we saw that entropy of normal file ranges between 4.5 and 6.5 which can be a view point to identify malware.



Malware Prediction

Our contribution

Thats how we roll.



We implemented the API call model which was stated by the author.

We tried to implement the rawByteToImage model. But it failed to classify, due to problems stated earlier.

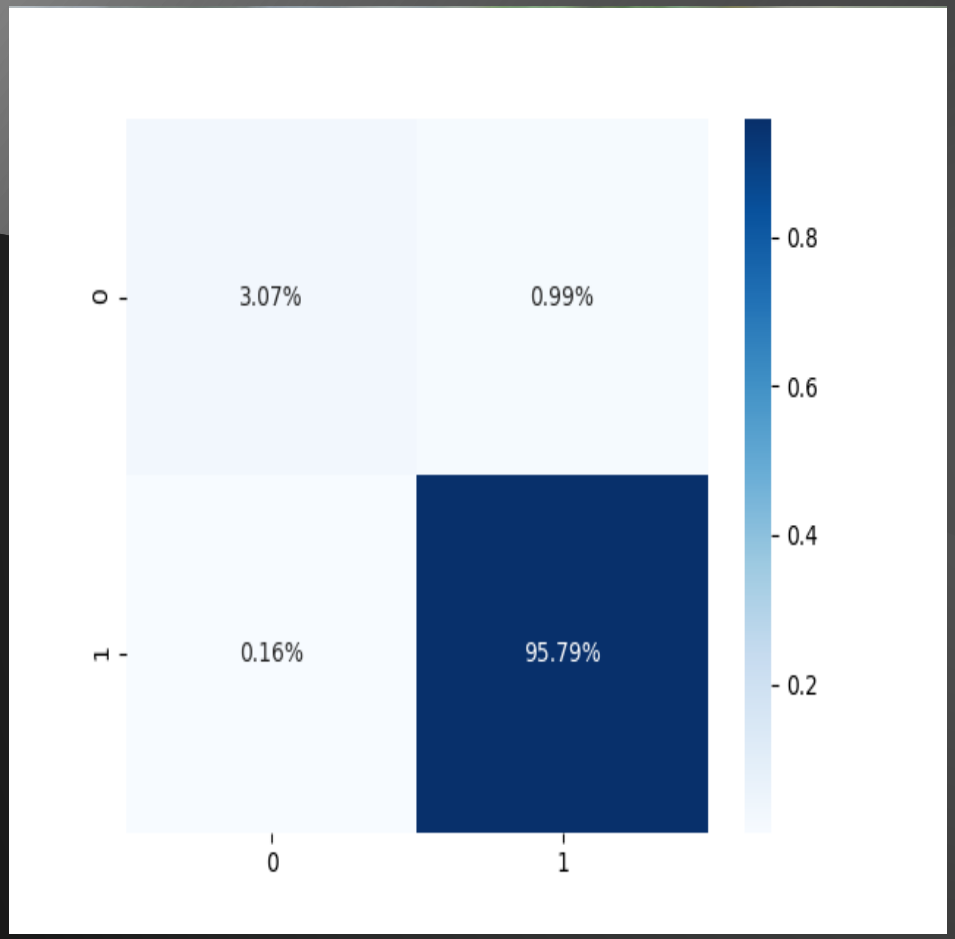
We observed potential of PE section size and Entropy and encorporated it with Api call model to enhance its flexibility and to prevent it from overfitting.

How we did it

We knew the potential of Both API calls and PE section Header in classification of malware. So we tried to merge them.

- First APIcalls Model run on the Executable file and we get the probability of the malware.
- Then we extract the PE Section header info from the file.
- If Entropy of the file is not within the range of 4.5 and 6.5 along with that the virtual size is greater than actual size. We increase the probablity by taking the weighted average.
- else if the entropy of the file is between the range of 4.5 and 6.5 we move the prediction towards the non-malware. This is how merging of both mode results in more flexible model.
- Then if the probability is greater than threshold probability the file is considered to be malicius

Result



confusion matrix show the the odds predicted by model
'0' represents goodware, '1' represents malware

Accuracy

Our model outperformed the previous model with 98.8% accuracy

Precision

Our model was very precise in finding malicious content from the dataset with 98.9% precision and goodware with 95% precision.

Area of Improvement

- A more balanced dataset for malware and goodware might significantly improve the results.
- Addition of some other static features might provide a more clear picture of the executable and improve overall performance.

THANK YOU

❖ Stay protected from malware.