Malware Detection Using Binary Visualization and Neural Networks

Yamini Devi Jonnala[1*], *Vamshi Sai* Mahajan[1], *Dheeraj* Menon[1], *Sampath Reddy* Kothakapu[1], and *Sumanth Reddy* Chandamollu[1]

[1] Department of Information Technology, GRIET, India

Abstract. Any programme or code that is damaging to our systems or networks is known as Malware or malicious software. Malware attempts to infiltrate, damage, or destroy our gadgets such as computers, networks, tablets, and so on. Malware may also grant partial or total control over the affected systems. Malware is often detected using classic approaches such as static programme analysis or dynamic execution analysis. The exponential rise of malware variations requires us to look beyond the obvious in order to identify them before they do harm or take control of our systems. To address these drawbacks, malware detection based on binary visualisation followed by the deployment of powerful machine learning techniques such as Convolutional Neural Networks (CNN) performs better than the ones we now use. We use these discoveries in our efforts to identify malware in different files and websites. We strive to complete the objective by employing representations of malware software binaries. With this concept, we can construct a better bridge for developing a functioning model that can identify malware in real time.

1 Introduction

Malware is often discovered in the form of software that masquerades as a legitimate file but, when opened, exposes its actual form, which may be a hazard to the system or network on which we are working. To keep their computers secure, users must ensure that all of these files are deleted. The first step is to determine which of the countless files currently on our computer, which have collected via various downloads, shared files, metadata, and so on, are malware files. Our project assists in classifying the file provided as input as malware or safe. The project also contains a tool that can categorise a malware file into one of 25 categories of malware and display which close malware traces are located in the provided file. The main goal is to determine whether or not a particular executable file contains malware by employing binary grayscale representations of the files.

* Corresponding author: yaminidevijj@gmail.com

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

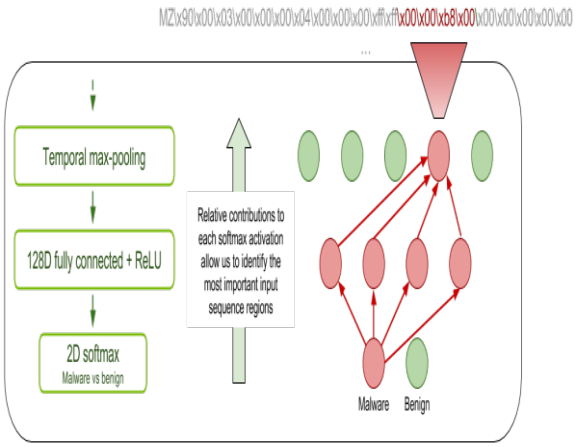

Fig. 1. Example Figure.

Malware is becoming a serious security problem. Malware is a malicious software that is designed to damage other computers or infrastructure. This implies that malware should be caught as soon as possible before it causes harm to the user. Existing detection methods use signature-based detection. Every known malware sample has a signature, and when a new sample is discovered, its signature is computed and compared to the current database. Although this is a quick procedure, the mechanism utilised to construct the signature may be crude. In many circumstances, modifying some source code or even inserting random code that does nothing to confuse antivirus systems may prevent the same dangerous software from being identified. This implies that the same virus may be rebuilt with little effort and will remain undetected. Other ways of identifying new malware samples include binary disassembly and static or dynamic analysis. This may be automated or done manually. When a researcher discovers a new piece of malware, the signature is uploaded to the database and utilised for future comparisons. However, this procedure takes time. Binary analysis is the process of inferring data from binaries or extracting data from binaries using either static or dynamic analysis. In this scenario, we use the binary executable's characteristics to determine its identification. This information is then compared to known samples to assess whether or not the application in question is malicious. In this paper, we apply multiple machine learning models to identify the presence of malware using binary attributes.

2 Literature Survey

Protecting prejudice against security risks is becoming more difficult; as malware changes, protection mechanisms struggle to keep up. In this research, Convolutional Neural Networks (CNN) and double visualisation are utilised to provide a concept for supporting security systems. Several research have focused on constructing analytic fabrics, collecting static characteristics, and linking malware families[1],[2]. A method that uses machine learning to detect an unknown harmful cargo while avoiding complications. Analyzing data and identifying malware has become a significant difficulty for those attempting to resolve security concerns and create safer working circumstances. [3] The solutions developed are primarily focused on the creation and construction of frameworks. They are concentrating on static feature selection and categorising malware into family types. [4]

Important characteristics capable of detecting the existence of malicious payload are collected from the picture during the pre-processing step and employed in the classification phase. The size of the photos is an essential aspect in the algorithm's performance. While pre-

processing, our primary attention should be on selecting the key features. These characteristics must be capable of detecting the existence of a malicious payload. Another consideration that must be considered when carrying out the operation is the size of the picture input. Indeed, having smaller pictures offers the essential ability to calculate quickly, but we lose the opportunity of having it big enough to incorporate the significant information required for distinguishing between dangerous and benign files. Methods for malware analysis that encompass both of the above orders have received little attention. On a data set with 37K samples, we used computer vision methods and SVMs to detect malware and reached a 95 percent accuracy. To describe packed malware, a combination of machine literacy, visualisation, and steganalysis was used; both SVMs and a mutant of the k-nearest neighbour (KNN) were used, with the final type perfection of 99.5 percent. [5]

A similarity discovery framework with 1.2 million samples was also used to categorise malware, with a delicacy of 99 percent achieved (for about half of the samples), although similarity considerably lessens the liability of linking the found malware with dissimilar structure. Similarly, malware's ability to adapt to its gestation terrain, the enormous number of features that may be extracted per sample, and the high number of malware instances reported all lessen the probability of correct detection. The maturity of the efforts focuses on malware grouping by families or similarities, but others estimate sample to picture metamorphoses, with maturity employing grayscale film land. The Visualization of Executables for Reversing and Analysis (VERA)[6] frame represents a program's inputs on a 3-dimensional space. It is used to indicate a hazy or confusing legal corridor. These are two examples of the intended outcome.

Contietal[7] developed an interconnected visualisation system that enables the examination of malware samples' byte information using colourful graphical rudiments. Each byte in the double sample is formed by a pixel in a "byte view visualisation" and the multitudinous bytes that have occurred are constituted by a "byte presence visualisation. Nataraj et al. are among those who predicted the usage of malware binaries as pictures. This is because complex visual patterns are simpler to perceive and distinguish. Actually, new malware that has emerged as a result of evolution or changes in behaviour is simply based on modifications made to existing malware. All of these versions have almost identical material.

So far, disadvantages have been experienced. Unfortunately, the options listed above are equally susceptible to conventional bushwhacking techniques. As a result, it seems that binary representation and machine literacy continue to offer issues and are more of a recent addition to existing systems than a relief. Furthermore, the offered methodologies are not resilient enough to strategies utilised by bushwhackers, such as data movement and data redundancy, to avoid the detection medium.

3 System Architecture

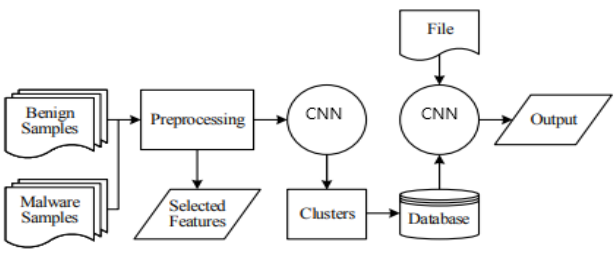

Fig. 2. System Architecture.

4 Methodology

4.1 Flow of the Project

The project's process is roughly separated into two segments. The first component of this is the process of acquiring binary information about the file from the operator. Then, utilising PIL loaded with Python, we carry out this action. The information generators from the provided file are the procedures get size and obtain binary data. The information gathered from this is utilised to create a grayscale picture of the binary data in the file.

4.2 Image in Grayscale

An image is anything that can be seen when a large number of pixels are combined. All of these pixels are just a block of an integer that represents the intensity of the light at that spot. A colour picture contains three layers, each representing red, blue, and green. Similarly, a grayscale picture has just one layer that displays only two hues, black and white.

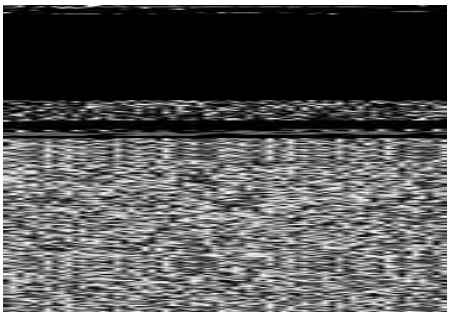

Fig. 3. Grayscale Image.

The transformed image will be parallel to the ones above. The CNN model is the second component. To make this work, the model must be constructed using several approaches such as sequential, dense, and flattening, as well as various activation functions such as Sigmoid, ReLu, and so on.

4.3 Neural Networks

A neural network is a collection of 'neurons'. In neural networks, a neuron is often represented as a node. Every data picture we submit to the model generates additional intermediate nodes/neurons. The network is divided into three tiers. One is an entry node that will accept input. The second kind of network is the hidden network, which contains the whole network logic. The output nodes come in third. According to the results, the third layer for binary data classifiers includes just two nodes. All nodes will be assigned weights based on the inputs we provide, and the pathways we follow will be recorded.

4.4 Convolutional Neural Networks

A CNN is one of several deep learning approaches or processes that primarily concentrate on the modification of pictures and objects or elements within one. All of the classifiers that we create using the CNN model's input will receive photos as inputs, analyse the images, and then categorise the images into certain categories using various models. For example, if we

feed the model photos of various cars and have created a vehicle classification model, the output categories will be Car, bike, truck, and so on.

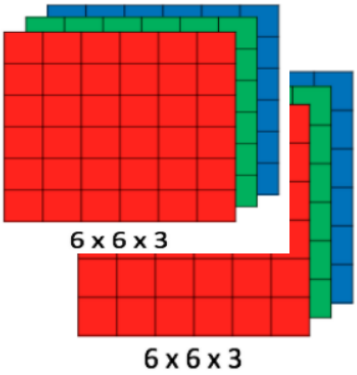

Fig. 4. CNN Diagram.

The image depicts how an RGB image will be internally formatted.

Conv2d is a class that functions as a layer, and we are developing it for our project in order to create the CNN model. We provide it with the following:
1. The total number of filters that the layer we are training should learn.
2. Convolution window width and height
3. Yet another activation function whose outputs are determined by the input.
4. Image size of our input

The evolved NN is produced by subjecting it to various techniques and modifications such as: activation functions, max pooling, dropout, flatten, and density.

4.4.1 Max Pooling

The pooling layer is used to collect the characteristics or qualities from the maps obtained by the convoluted filters' filters. By doing so, we may reduce the amount of space used up by the representation. On the other hand, it has another use in that it reduces overall computation during picture processing.

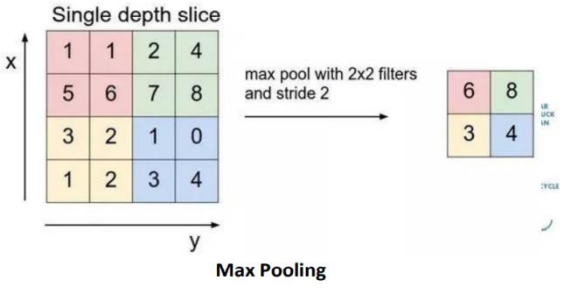

Fig. 5. Max Pooling Diagram.

4.4.2 Activation Function

As we all know, when an image is delivered to a model while it is being trained, the input adds bias and weights to the nodes that it will follow to reach the final layer. The activation function is employed at this moment. When the weights are applied to a node, that node is

activated using the appropriate activation function. ReLu is a non-linear activation function in multi-layer neural networks or deep neural networks.

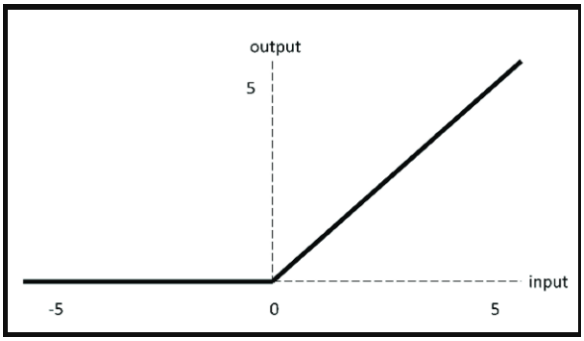

Fig. 6. ReLU Activation Graph.

The graph generated for the ReLu activation is shown in the image. This is the definition of the ReLu activation function that we are employing in the present project.

5 Results Analysis

We feed our model the transformed picture of a file for prediction. If a file includes malware, our result array will contain the number 1; otherwise, all of the array's values will be 0.

```
In [77]: test_image = image.load_img("C:\\Users\\user\\Desktop\\Mini Project\\Implementation\\mcpatcher_.png", target_size = (64, 64))
         test_image = image.img_to_array(test_image)
         test_image = np.expand_dims(test_image, axis = 0)
         result = Malware_model.predict(test_image)
         type(result)
         result.view()

Out[77]: array([[0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0.,
                 0., 0., 0., 0., 0., 0., 0., 0., 0.]], dtype=float32)
```

Fig. 7. Result array.

We can determine whether or not malware is present by taking the total of the whole array that is sent as output. If the total is more than one, the location of the one inside the array will indicate the kind of malware traces found in the provided PE file.

```
    else:
        width  = int(math.sqrt(data_length)) + 1
        height = width

    return (width, height)
createGreyScaleImage("C:\\Users\\mahaj\\Desktop\\Malware Detection\\Programs\\mcpatcher.exe")
```

```
The file C:\Users\mahaj\Desktop\Malware Detection\Programs\mcpatcher_.png saved.
```

Fig. 8. Grayscale image is generated.

The generated picture is now transmitted to the trained CNN model, which predicts the sort of malware traces found in the provided portable executable file.

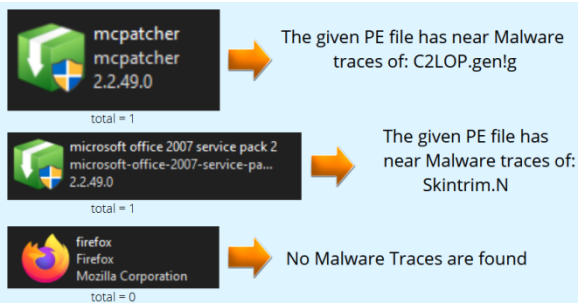

Fig. 9. Various possible results from the model.

6 Conclusion

To summarise our study, we developed a tool that can identify whether form of malware is contained in a given portable executable file, i.e., application files. This would make it simpler to identify malware whenever a new file is installed or transmitted from and to our personal computers, which must not be assaulted by viral assaults. So, using our technology, we can determine if a file is dangerous and take the necessary steps, such as pausing the installation and looking for alternatives. Because we know that data is what drives our contemporary lives, we can only choose pure data that is not harmful to our systems or networks. In addition, through working on this project, it was observed that many malware detection systems use signature-based classifications to discover the pure ones among damaged files.

7 Future Enhancements

According to all prior studies and research, we observed that all malware detection approaches, such as static and dynamic methods, decision tree classifiers, and so on, are based on the basic notion of detecting malware by comparing it to known signatures of previously discovered malwares. There is future possibility for really detecting the signatures of all the irregularities in the file's contents and then demonstrating the existence of malware or malicious software that might be potentially damaging to our networks or systems. As a result of this concept, we can identify threats even without knowledge of previously discovered malware signatures since we just look for anomalous signatures. This manner, we can even locate malware that has yet to be found or deemed significant to name.

References

1. D. Gavrilut, M. Cimpoesu, D. Anton, and L. Ciortuz, “Malware Detection Using Perceptrons and Support Vector Machines,” in proc. 2009 Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, pp. 283–288, (2009)
2. J. Z. Kolter and M. A. Maloof, “Learning to detect malicious executables in the wild,” in proc. 10th ACM SIGKDD Int’l Conf. Knowledge Discovery and Data Mining (KDD), pp. 470–478 (2004)

3.]D. Uppal, R. Sinha, V. Mehra, and V. Jain, "Malware Detection and Classification Based on Extraction of API Sequences," in proc. 2014 Int'l Conf. Advances in Comput, (2014)
4. R. S. Pirscoveanu, et al., "Analysis of Malware behavior: Type classification using machine learning," in proc. 2015 Int'l Conf. Cyber Situational Awareness, Data Analytics and Assessment (CyberSA), pp (2015)
5. D. Kirat, L. Nataraj, G. Vigna, and B. S. Manjunath, "SigMal: a static signal processing based malware triage," in proc. 29th Annual Computer Security Applications Conf. (ACSAC), pp. 89–98 (2013).
6. R. Gove, et al., "SEEM: a scalable visualization for comparing multiple large sets of attributes for malware analysis," in proc. 11th Workshop on Visualization for Cyber Security (VizSec), pp. 72–79 (2014).
7. D. A. Quist and L. M. Liebrock, "Visualizing compiled executables for malware analysis," in proc. 6th Int'l Workshop on Visualization for Cyber Security, pp. 27–32 (2009)