CSE3502- Information Security Management

Project Report

Image Malware Detection using Deep Learning

By

Reg. No. :19BAI1026 Name: Pranaydeep Mayank

B. Tech Computer Science and Engineering

Submitted to

Dr. MANAS RANJAN PRUSTY

School of Computer Science and Engineering

April 2022

CONTENTS

DECLARATION

I hereby declare that the report titled "**Image Malware Detection"** submitted by me to VIT Chennai is a record of bona-fide work undertaken by me under the supervision of Dr. MANAS RANJAN PRUSTY, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai.

Signature of the Candidate

(Pranaydeep Mayank)

19BAI1026

CERTIFICATE

This project report entitled "**Image Malware Detection using Deep Learning"** is a bonafide work of **Pranaydeep Mayank (19BAI1026)** and he carried out the Project work under my supervision and guidance for CSE3502 – Information Security Management.

Dr. MANAS RANJAN PRUSTY

SCOPE, VIT Chennai

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to Dr. MANAS RANJAN PRUSTY Sir for guiding me along the project and helping me out with the issues faced during the completion of my project.

Pranaydeep Mayank

19BAI1026

ABSTRACT

Traditional approaches to analyzing malware involve the extraction of binary signatures. Due to the rapid increase in the number of new malware signatures, it has become difficult to analyze them. Other techniques such as static code analysis or dynamic code analysis can also be utilized to identify and remove malicious software. Static analysis works by disassembling the code and exploring the control flow of the executable to look for malicious patterns. On the other hand, dynamic analysis works by executing the code in a virtual environment and a behavioral report characterizing the executable is generated based on the execution trace. Both these techniques have their pros and cons. Static analysis offers the most complete coverage but it usually suffers from code obfuscation. Dynamic analysis is more efficient and does not need the executable to be unpacked or decrypted. However, it is time intensive and resource consuming, thus raising scalability issues.

Moreover, some malicious behaviors might be unobserved because the environment does not satisfy the triggering conditions.

Hence converting malware byte code to grayscale image and then classifying it is a much more efficient way of detecting malwares.

1. Introduction

This project aims to build a simple yet effective method for visualizing and classifying malware using image processing techniques. Malware binaries are visualized as gray-scale images, with the observation that for many malware families, the images belonging to the same family appear very similar in layout and texture. Motivated by this visual similarity, a classification method using standard image features is proposed. Neither disassembly nor code execution is required for classification.

Preliminary experimental results are quite promising with CNN classification accuracy on a malware database of 9,339 samples with 25 different malware families.

Dataset Used

Malimg Dataset

The Malimg dataset consists of 9339 images of 25 classes, and hence these samples require no preprocessing before applying image-based analysis.

Malware Classes:

Tab.3 Malware Dataset of 25 Families

#	Class	Family	#
1.	Worm	Allaple.L	1591
2.	Worm	Allaple.A	2949
3.	Worm	Yuner.A	800
4.	PWS	Lolyda.AA 1	213
5.	PWS	Lolyda.AA 2	184
6.	PWS	Lolyda.AA 3	123
7.	Trojan	C2Lop.P	146
8.	Trojan	C2Lop.gen!g	200
9.	Dialer	Instantaccess	431
10.	TDownloader	Swizzot.gen!I	132
11.	TDownloader	Swizzor.gen!E	128
12.	Worm	VB.AT	408
13.	Rogue	Fakerean	381
14.	Trojan	Alueron.gen!J	198
15.	Trojan	Malex.gen!J	136
16.	PWS	Lolyda.AT	159
17.	Dialer	Adialer.C	125
18.	TDownloader	Wintrim.BX	97
19.	Dialer	Dialplatform.B	177
20.	TDownloader	Dontovo.A	162
21.	TDownloader	Obfuscator.AD	142
22.	Backdoor	Agent.FYI	116
23.	Worm:AutoIT	Autorun.K	106
24.	Backdoor	Rbot!gen	158
25.	Trojan	Skintrim.N	80

2. Literature Survey

Using convolutional neural networks for classification of malware

Represented as images

Daniel Gibert · Carles Mateu · Jordi Planes· Ramon Vicens

This paper presents a novel file agnostic deep learning system for classification of malware based on its visualization as gray-scale images. As far as we know, it is the first approach.

Using convolutional neural networks for classification of malware represented as images to apply deep learning to find patterns from malware's binary content represented as images. The proposed solution has a number of advantages that allow malicious programs to be detected in a real-time environment. Firstly, it is file agnostic and is based solely on the binary code of an executable.

Secondly, the transformation of an executable into a gray-scale image is inexpensive. Thirdly, the prediction time is lower than the rest of approaches. Fourthly, it obtained greater classification accuracy than all previous methods in the literature that were based on the representation of malware as gray-scale images.

Malware Images: Visualization and Automatic Classification

L. Nataraj, S. Karthikeyan, G. Jacob, B. S. Manjunath

This paper presented a novel approach to malware analysis based on visualizing and processing malware as images. A commonly used image feature descriptor is used to characterize the malware globally. The preliminary results are very encouraging, with high accuracy classification that is competitive with the state of the art results in the literature at a significantly less computational cost. We believe that using computer vision techniques for malware analysis opens the path to a broader spectrum of novel ways to analyze malware.

3. Requirements Specification

– CPU: Intel i7 or any equivalent

– Memory: 8 GB RAM

– GPU: Nvidia GTX 1050 Ti

-Tensorflow, Python

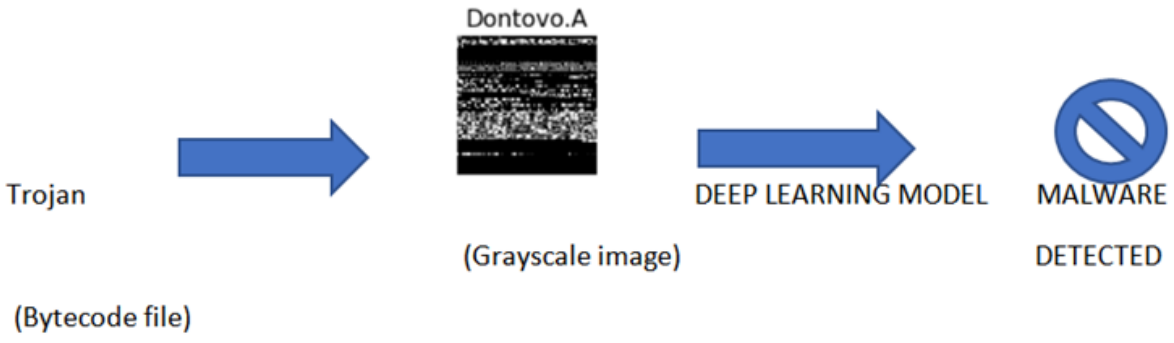
4. Implementation

1.) Convert Malware byte code to gray-scale png image.

2.) Create deep learning models for classification of these images:

1. CNN
2. CNN with Fourier transform
3. Transfer Learning(Inception model,VGG19,AlexNet)

3.) Use this built model to predict any intruder malware in the system.

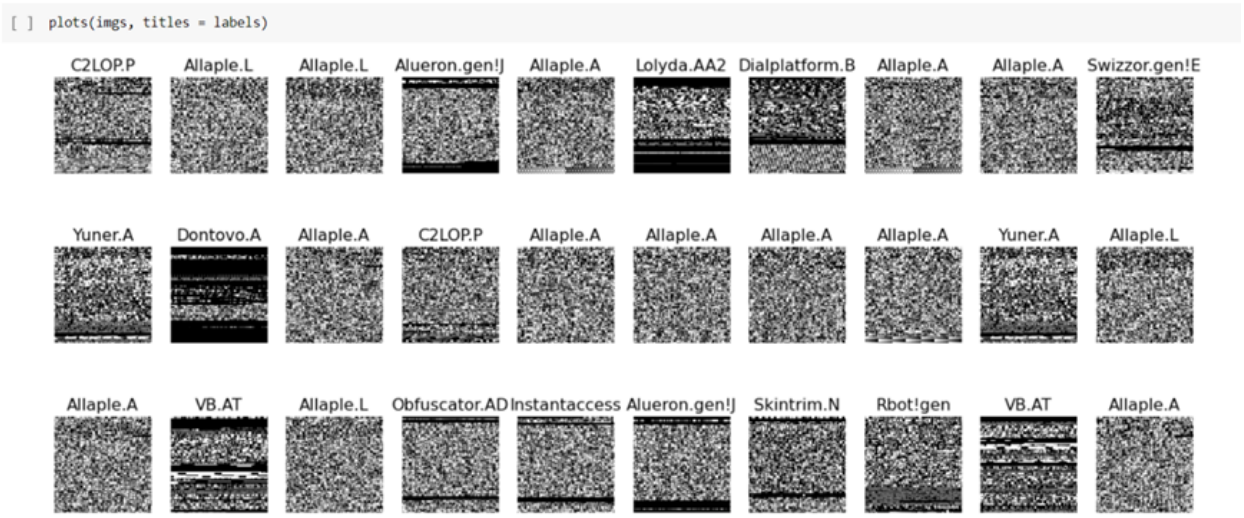


CNN Architecture Used:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 30)        840

 max_pooling2d (MaxPooling2D  (None, 31, 31, 30)       0
 )

 conv2d_1 (Conv2D)           (None, 29, 29, 15)        4065

 max_pooling2d_1 (MaxPooling  (None, 14, 14, 15)       0
 2D)

 dropout (Dropout)           (None, 14, 14, 15)        0

 flatten (Flatten)           (None, 2940)              0

 dense (Dense)               (None, 128)               376448

 dropout_1 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 50)                6450

 dense_2 (Dense)             (None, 25)                1275

=================================================================
Total params: 389,078
Trainable params: 389,078
Non-trainable params: 0
```

4. Results and Discussion

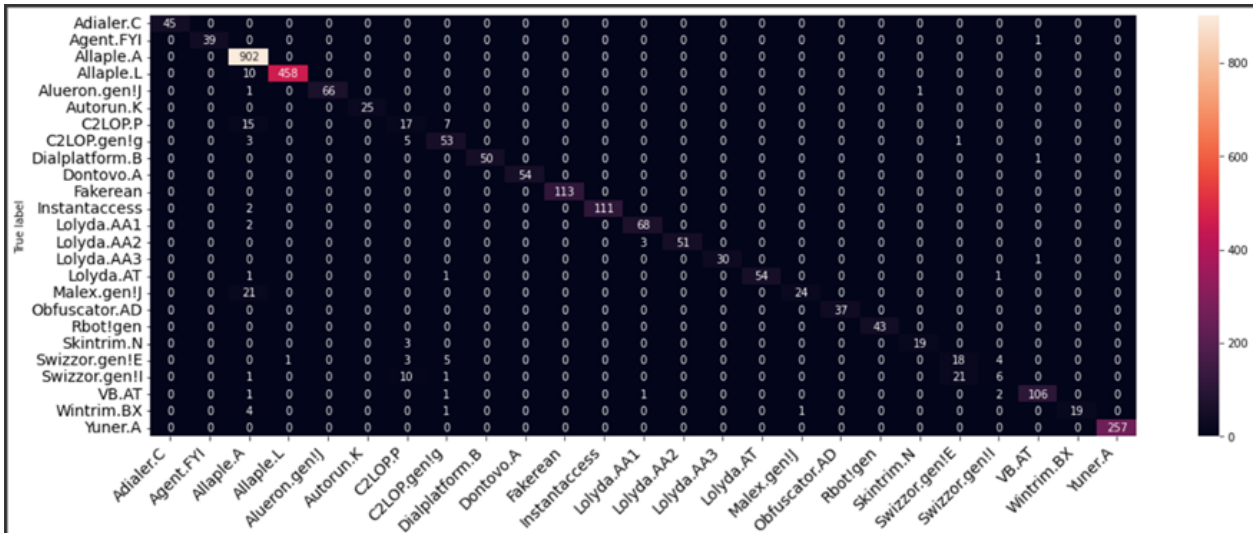
Malware byte code after converting to grayscale images:



5.1. CNN Model

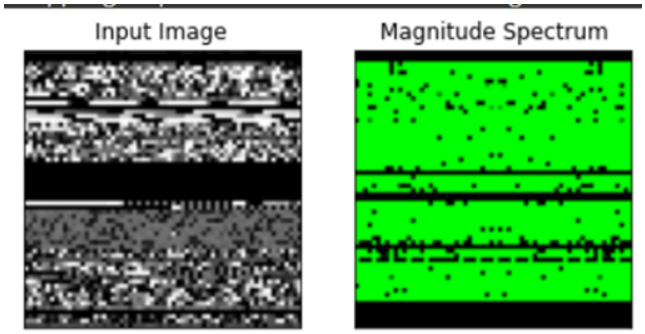
Final CNN accuracy: 0.9511063694953918

Confusion Matrix:



5.2 CNN + Fourier

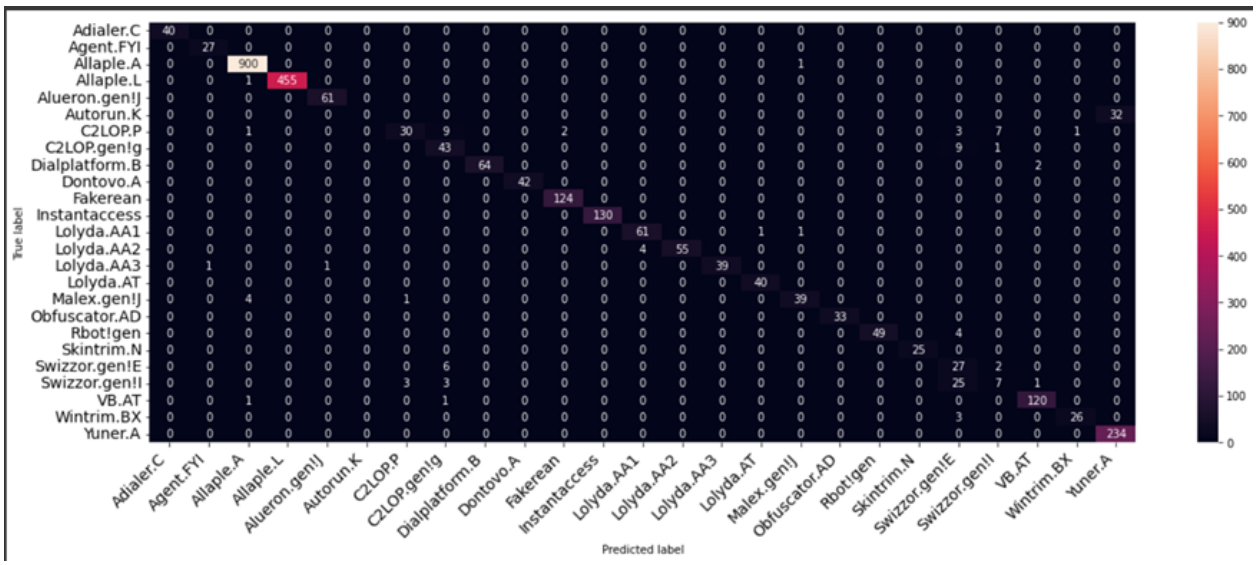
- Converting malware byte code to grayscale image.
- Applying Fourier transform to images.



- Build a CNN model and feed the obtained images to this model.

Final Fourier + CNN accuracy: 0.9532476663589478

Confusion Matrix:



5.3 Transfer Learning Model

1. VGG19:

Final accuracy: 0.9110

2. ResNet101:

Final accuracy: 0.7407

3. InceptionV3:

Final accuracy: 0.7603

6. Conclusion and Future Work

The proposed methodology of converting a malware byte code to an image and then deploying a Deep Learning Model to detect it was successfully implemented .

Fourier + CNN gave the best results with 95.32476663589478 % and outperformed all other models.

Future Works includes using GANs for generating new types of malwares from the existing ones which could be used to detect more types of malwares.

8. REFERENCES

https://paperswithcode.com/paper/using-convolutional-neural-networks-for-1

https://paperswithcode.com/paper/an-empirical-analysis-of-image-based-learning

https://www.superdatascience.com/blogs/the-ultimate-guide-to-convolutional-neural-networks-cnn