ANALYSIS OF RESNET MODEL FOR MALICIOUS CODE DETECTION

RIAZ ULLAH KHAN, XIAOSONG ZHANG, RAJESH KUMAR, HUSSAIN AHMAD TARIQ

School of Computer Science & Engineering, University of Electronic Science and Technology of China.
E-Mail: rerukhan@outlook.com, rajakumarlohano@gmail.com

Abstract:

In this study, we have used the ResNet model to detect the new type of malware. ResNet model is investigated and tested which belongs to Microsoft. We have used two types of datasets to train and test the model. We collected dataset from Microsoft Malware Classification Challenge which contains 10868 Binary files, these binary files are further divided in nine different malware families and second dataset is benign dataset which contained 3000 different kinds of benign software. Benign dataset and dataset from Microsoft Malware Classification Challenge were initially .exe files which were converted in to opcode and then converted in to image files. We obtained a testing accuracy of 87.98% on ResNet model.

Keywords:

Malware Detection; Malware Classification; Opcode; ResNet

1. Introduction

One of the major challenges in the realm of security threats is malicious software which is also referred as malwares. The main focus of malware is, to gather the personal in- formation without the attention of users and to disturb the computer operations which makes problems for users. There are many kinds of malwares i.e. Virus, Worm, Trojan-horse, Rootkit, Backdoor, Spyware, Adware etc. Annual reports from antivirus companies show that thousands of new malwares are created every single day. These new malwares become more sophisticated that they could no longer be detected by the traditional detection techniques such as signature-based detection, heuristic detection or behavior-based detection.

Signature-based detection searches for specified bytes sequences into an object so that it can identify exceptionally a malware. Its drawback is that it cannot detect zero-day or new malware since these malware signatures are not supposed to be listed into the signature database. Heuristic-based detection was developed to basically overcome the limitation of the signature detection technique, in the way that it scans the system's behavior in order to identify the activities which seems to be not normal, instead of searching for the malware signature. Heuristic-based detection method can be applied to newly created malware whose signature has not yet been known. The limitation of this technique is that it affects the system's performance and requires more space. Behavior-based detection technique is more about the behavior of the program when it is executing. If a program executes normally, then it is marked as benign, otherwise it is marked as a malware. By analyzing this definition of the behavior-based detection, we can directly conclude that the drawback of this technique is the production of many false positives and false negatives, because a benign program can have crashed and be marked as a virus or virus can execute as if it was a normal program and simply be marked as benign.

1.1. Motivations

Malware is growing in the huge volume every day, we used image processing technique to improve accuracy and performance. Image processing technique analyzes malware binaries as gray-scale images and opcode images. The previous research [10] proposed a gray scale image technique to classify malware using image processing technique. Some of mature image processing techniques are widely used for object recognition e.g. Taobao is popular shopping website in china which find's the product using image recognition technique. This method performs high accuracy in practice. In this study, we converted binary code to opcode images for recognizing malwares which preserve the similarities variant images. We compared ResNet model to achieve better performance in terms of speed and accuracy.

1.2. Contributions

The main contributions of the paper are summarized as follows;

- We proposed a model for data preparation in Section 3.
- We used all the sub-models of ResNet model for

detection of malwares, based on opcode technique which is further described in Section 4

- We analyzed and detected unknown or new type of malwares successfully.
- We compared the performance of ResNet model in brief.

1.3. Structure of paper

The Section 1 discusses the introduction, motivation and main contributions of this work. Section 2 Discusses malware detection techniques and the related work. Section 3 give a brief overview on the methodology that how to convert executable files in to images and setup the python libraries. Section 4 proposes a malware detection technique, also describes the implementation and experimental results in terms of speed and accuracy. Finally, Section 5 concludes the paper.

2. Background

2.1. Malware Detection Techniques

Deep Learning: Recent success in deep learning research and development attracts people's attention [7]. In 2015, Google released TensorFlow [1], a framework of realizing deep learning algorithm. Deep learning is a specific type of machine learning. More specifically, deep learning is an artificial neural network, in which multiple layers of neurons are interconnected with different weights and activation functions to learn the hidden relationship between input and output. Intuitively, input data is fed to the first layer that generates different combinations of the input [12]. The weights on links between layers are adjusted according to backward propagation, depending on the distance or less function between true output label and the label calculated by neural network. Note that deep learning can be seen as a neural network with a large number of layers. After the above learning process via multiple layers, we can derive a better understanding and representation of distinguishable features, enhancing the detection accuracy [9]. Also notice that the effectiveness of deep learning increases by the network size. In addition to deep neural networks, the most well-known deep networks are convolutional neural networks (CNN). The representation of CNN includes AlexNet, VGG, ImageNet, and ResNet [6-8][12]. More specifically, CNN is composed of hidden layers, fully connected layers, convolution layers, and pooling layers. The hidden layers are used to increase the complexity of the model. If the same number of neural is associated with the input image, the number of parameters can be significantly reduced, adapting to the function structure much properly.

3. Data Preparation and Environment Setup

This section is divided into two parts. The first part is, to collect malware and benign datasets from different sources and second part describes the techniques of preparation of the dataset. In second part we used a technique to prepare dataset which is described in Section 3.2.

3.1. Collection of Dataset

We have collected two datasets from different sources. One of them is malicious dataset from Microsoft Malware Classification Challenge. We also have collected 3000 benign files from different sources. Both datasets are discussed briefly in the following discussions.

1) **Microsoft Malware Classification Challenge Dataset:** The Microsoft dataset contains 9 classes for training and testing purposes. Microsoft provided 500GB of data which includes 21741 malware samples. 10868 of samples are used for training, and the remaining samples are used for testing.

2) **Benign files:** We collected 3000 benign files from different sources.

Table 1 Microsoft malware classification challenge dataset

No	Family Name	No of Samples
1	Ramnit	1541
2	Lollipop	2478
3	Kelihos_ver3	2942
4	Vundo	475
5	Simda	42
6	Tracur	751
7	Kelihos_ver1	398
8	Obfuscator.ACY	1128
9	Gatak	1013

3.2. Data Preparation Technique

This paper proposes the following technique to process the data.

Opcode to Image:

3) **Unpacking:** In this method we use two tools for extracting hidden code from binary files.

a) PEID: It is used for static code
b) Poly Unpack: it is used for dynamic code

4) **Decompiling:** In this phase we decompile opcode sequence from assembly code and then convert 2-tuple opcode sequence rather than larger length of opcode sequence.

5) **2-Tuple code to Opcode Sequence:** In this step, we

converted 2-tuple code to opcode sequence.

6) **Binary Image Reconstruction and Enhancement:** Images are constructed by binary opcode frequency. Opcode sequences will be enhanced by using histogram normalization, dilation, and erosion techniques.

In the above method of data preparation, we can easily identify malware and benign files by visual analysis as shown in Fig.1.

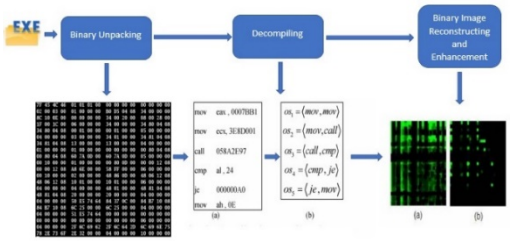

Fig.1 Overview architecture of Preparation Dataset using opcode

3.3. Environment Setup

Ubuntu system with 64bit with 8 GB RAM environment is used to perform tests. We used Python programming language to perform the experiments. Python packages and libraries such as Tensor Flow, Docker Server, Anaconda are used which helped to detect the malware. The Tensor Flow Library is used for training the model which uses the convolutional natural network (CNN).

4. Implementation and performance evaluation of the proposed model

4.1. Proposed Model

This section divided into two parts, first part is data preparation and the other part is train and test the model. Fig.2 illustrates all the phases of our proposed model.

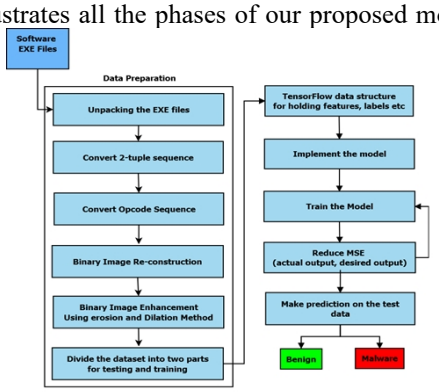

Fig.2 Architecture of Proposed Model i.e. from Data preparation to prediction

We divided model in two phases i.e. i) Training phase and ii) Detection phase. For the training and the detection of malware we used CNN model, as shown in Fig.3. We prepare the dataset using different techniques shown in data preparation section. The output of the data preparation section is "image files". Images have binary labels i.e. either benign or malware. we used supervised learning model in which the features are extracted automatically. The detection phase is shown in Fig.3. The same exe file convert in image and trained classifier detect the malicious code.

Table 2 Comparison results of different models of ResNet

Model	Train Acc.	Validation Accuracy	Loss	Validation Loss	Time
ResNet18	0.83	0.87	1.0436	1.006	2701s
ResNet34	0.8651	0.8519	1.5983	1.7510	4800s
ResNet50	0.8662	0.8095	4.3967	4.5914	5580s
ResNet101	0.8594	0.7884	8.4274	8.7475	6112s
ResNet152	0.8798	0.8836	11.943	12.05	9248s

4.2. ResNet Model for Malware Detection

The ResNet CNN model builds by Microsoft organization in 2015, that contain 152 hidden layers. We have used ResNet convolutional neural networks because it is more accurate and it can be applied to the entire image at a time and then we can assume they are best to use for feature extraction. However, the model increases the dubbing and innovation cost but gives us high accuracy. In this experiment we apply TensorFlow ResNet Library which is easy to deploy and achieve accuracy. We observed that the model has final training accuracy and validation accuracy of 0.83 and 0.87 respectively. We noted that the model has comparable performance on both train and validation datasets (labeled test). In this experiment final loss accuracy and validation loss accuracy are 0.83 and 1.006. Table 2 gives a brief overview of the results about ResNet18, ResNet34, ResNet50, ResNet101 and ResNet152 models. Further, Fig.4 and Fig.5 give a brief overview of Time accuracy of different ResNet Models and Comparison of Training and Testing accuracy among different ResNet Models.

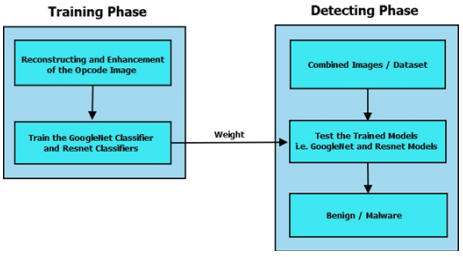

Fig.3 Implementation Phases of the Model in Deep

Learning

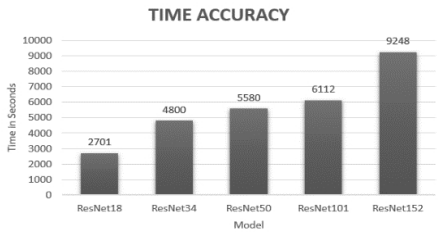

Fig.4 Time Accuracy of Different ResNet Models

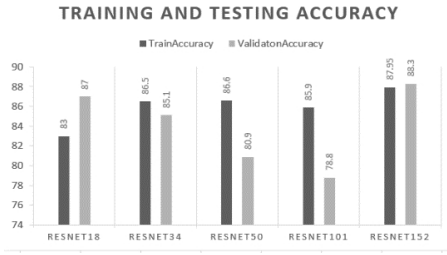

Fig.5 Comparison of the Training and Testing accuracy

5. Conclusion

Being able to visualize the malicious code as a gray-scale image has been a great achievement. Many researchers have been using this technique for the task of malware classification and detection. However, other works have shown that this technique can be easily vulnerable to adversarial attacks and produce erroneous results. We have shown in the work that how a small change in the image could lead to misclassification of images. The biggest challenge is to find an efficient way to overcome the vulnerability of Neural Networks. This could be achieved by carefully analyzing malware binaries.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, Xiaoqiang Zheng, and Google Brain. "TensorFlow: A System for Large-Scale Machine Learning." 12th USENIX Symposium on Operating Systems Design and Implementation, 2016.

[2] F Afifi, N B Anuar, S Shamshirband, and K K R Choo. "DyHAP: Dynamic Hybrid ANFIS-PSO Approach for Predicting Mobile Malware." PloS one Journal, September 2016.

[3] D Barrera, H G Kayacik, and P C van Oorschot. "A methodology for empirical analysis of permission-based security models and its application to android." 17th conference on computer and communications security, pp 73-84, 2010.

[4] W Enck, M Ongtang, and P McDaniel. "On lightweight mobile phone application certification." 16th ACM conference on Computer and communications security, pp 235-245, 2009.

[5] A P Felt, K Greenwood, and D Wagner. "The effectiveness of application permissions." 2nd USENIX conference on the Web Development, pp 75-86, 2011.

[6] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size. Cornell University Library, November 2016.

[7] Allan Jabri, Armand Joulin, and Laurens van der Maaten. "Revisiting Visual Question Answering Baselines." European Conference on Computer Vision, pages 727–739. 2016.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks", Neural Information Processing Systems Conference, pp 1097-1105, 2012.

[9] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." Nature, 521(7553), pp 436–444, May 2015.

[10] Lakshmanan Nataraj, Vinod Yegneswaran, Phillip Porras, and Jian Zhang. "A Comparative Assessment of Malware Classification Using Binary Texture Analysis and Dynamic Analysis." Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, pp 21–30, 2011.

[11] Jürgen Schmidhuber. "Deep learning in neural networks: An overview." Neural Networks, Vol. 61, pp 85–117, Jan 2015.

[12] F Tong and Z Yan. "A hybrid approach of mobile malware detection in Android." Journal of Parallel and Distributed Computing, 2017.