ISSN 2278-3091
Volume 9, No.4, July – August 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse214942020.pdf
https://doi.org/10.30534/ijatcse/2020/214942020

Detecting malware using the MLP algorithm

Do Hoang Long[1], Tisenko Victor Nikolaevich[2],Do Minh Tuan[3], Nguyen Anh Tuan[4], Nguyen The Lam[5]
[1,3,4,5]Information Assurance dept. FPT University, Hanoi, Vietnam, longdhse05220@fpt.edu.vn, tuandmse05518@fpt.edu.vn, tuannase62864@fpt.edu.vn, lamntse63326@fpt.edu.vn,
[2]Department Quality Systems, Peter the Great St. Petersburg Polytechnic University, Russia, St.Petersburg, Polytechnicheskaya, 29, v_tisenko@mail.ru

ABSTRACT

Malware attacks are dangerous and difficult to detect and prevent. Therefore, the task of detecting signs of malware and alerting it for users or the system is very necessary today. One of the most effective malware detection approaches is applying machine learning or deep learning to analyze its behavior. There have been many studies and recommendations to analyze malicious behavior then combined with some sorting or clustering methods to find their signs. In this paper, we will propose a method to use machine learning to detect malicious signs based on their unusual behavior. Accordingly, in our research, we will conduct malicious analysis using static and dynamic analysis methods to detect abnormal behaviors and combine them with a Multi-layer perceptron (MLP) to the conclusion on malware behavior.

Key words: Malware detection, feature selection, MLP.

1. INTRODUCTION

Malware is software programs designed to harm or perform unwanted actions on a computer system. Malicious software is essentially a software like other software on the computer that is used every day and has all the characteristics and properties of a normal software, except that it is more malicious. The study listed some common types of malware including Virus, Worm, Trojan Horse, Malicious Mobile Code, Tracking Cookie, Attacker Tool, Phishing, Hoax Virus. According to statistics [1], the situation of malware distribution in 2019 increased by 79% compared to 2018. This is entirely reasonable because hackers used to focus on information systems in the past. This usually chooses to attack the user primarily. Therefore, malware rapidly increases not only in a number of attacks but also its dangerous levels. In the study [10], there are several approaches to detecting malware. The two basic methods used to detect malware are the sign-based detection method and based on behavioral analysis. Methods of detecting malware based on a set of signs have been studied and applied early because of its rapidity and accurate detection capability. Commonly used signs in this method include hash code, IP, Domain or Indicators of compromise. However, the disadvantage of this method isn't able to detect new malware samples that are not in the signature database. In this paper, we propose a method to detect malware based on machine learning techniques. In the paper [1], there are some difficulties in the method of detecting malware based on machine learning. In our study, we propose a malware detection process based on static and dynamic analysis. Finally, to conclude the existence of malware in the system we propose to use machine learning algorithms.

2. RELATED WORKS

2.1. Malware detection technique

a) Detection technique based on static analysis
The malware detection technique is based on the static analysis method, which is characterized by the detection of malware without having to run or execute any of its code, including three main methods: scanner technology, diagnostics based on Heuristics and Integrity Checkers.

b)Detection technique based on dynamic analysis
The malware detection technique based on dynamic analysis is the technique of determining whether a file is infected by executing program code and observing its behavior. Two main techniques for malware analysis include:
- Behavior Monitors/Blockers: Behavior blocker is a technique that monitors the execution behavior of a program in real-time. Besides, this technique also allows the monitoring of suspicious actions and blocks of malware.
- Emulation: Malware detection techniques use emulation that allows the code to be run and analyzed in a simulated environment. Two main techniques are Dynamic heuristic and Generic decryption.

2.2. Detecting malware based on machine learning

To solve the disadvantage of the method of detecting malware based on the signal set, the technique of detecting malware based on analyzing the behavior of malware was born. In the research [1], the authors presented the idea of

detecting malware based on file abnormal behavior based on the machine learning algorithm. The paper [2] presented a number of basic approaches in the problem of malware detection based on machine learning including how to extract malicious features and detection algorithms. For extracting feature data, recent studies often use three main techniques including [2 -7]: static analysis, dynamic analysis, and combined analysis. Based on these analysis techniques, the malware will be analyzed and synthesized into the corresponding sequence of behaviors. In this paper, we will use static and dynamic analysis methods to look for abnormal behavior of malware based on the sandbox tool [8]. After configuring and analyzing malware with a sandbox tool, the main groups of behaviors that can be selected and used to extract malicious behavior include Byte sequences, Opcodes, network Activity, System files, API and System Calls, Windows Registry, PE file characteristics. This method has two basic algorithms that are machine learning and deep learning algorithm [9, 10]. In this paper, we use theMLP algorithm. This detection method is relatively effective and has been researched and experimented in many studies.

2.3. Some malware detection tools

To be able to implement the above methods, we need the support tools corresponding to each specific method. Below, I offer five main groups of tools:

- Antivirus software: Kaspersky, Bitdefender, Avast, Norton, Bkav, etc.
- Network monitoring tool group: TCPView, Wireshark.
- A group of tools for monitoring file system resources: AutoRun. ProcessExplorer, ProcessMon, etc.
- Registry monitoring tool group: ProcessMon, AutoRun, etc.

Automatic analysis tool: Sandboxie, Cuckoo Sandbox, etc.

3. DETECTING MALWARE USING THE MLP ALGORITHM

3.1 Introducing the MLP algorithm

The mathematical basis and operating principle of MLP have been presented in [9, 10]. Generally, MLP consists of 3 main components: input layer, output layer, both of which usually consist of 1 layer on neutrons, and a hidden layer which may have one or more layers of neurons depending on specific problems. In deep neural networks, the hidden layer of MLP may consist of tens to hundreds of neuron layers. MLP was basically designed to describe how the nervous system works. In MLP, except for the input layer, all neurons of other layers are fully linked to the neurons of the previous layer. Each neuron receives the input vector, combines with its weights, then input to get the transfer function to output a result. The output is calculated based on the following formula:

$$a_i^{(l)} = f(w_i^{(l)T} a^{(l-1)} + b_i^{(l)}) \quad (1)$$

Where $a_i^{(l)}$ is the output of neuron *i* in layer *l*; $w_i^{(l)}$ is the weight vector of $a_i^{(l)}$; $b_i^{(l)}$ is the bias; $f(\bullet)$ is the activation function. In this research, the activation function ReLU [10] is used (Figure 1).

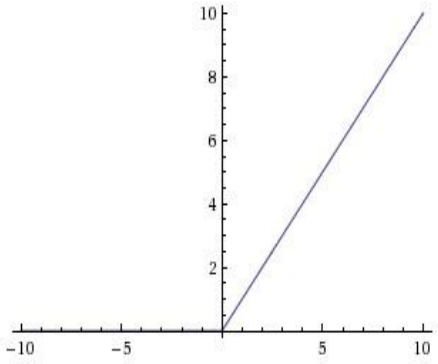

Figure 1 : Graph of ReLU function

The ReLU function is presented by following formula:

$$f(s) = \max(0, s)$$

The advantages of using ReLU are not only to help deep networks converge faster but also to facilitate the computation process [11].

The number of output neurons is equal to the number of output classes. The output of each neuron in the output layer is formulated as [12]:

$$a_i = \frac{\exp(z_i)}{\sum_{j=1}^{C} \exp(z_j)}, \forall i = 1,2,3....C \quad (2)$$

Where, z_i is the input to the Softmax function, which is the production of the inputs and the neuron's weights adding the bias; *C* is the number of output classes.

3.2 Selecting and extracting features of malware

3.2.1. The features extraction process

In the study [13], the author proposed the process of extracting the features of the malware based on the malware analysis method. Figure 2 below describes the process of selecting and extracting features.

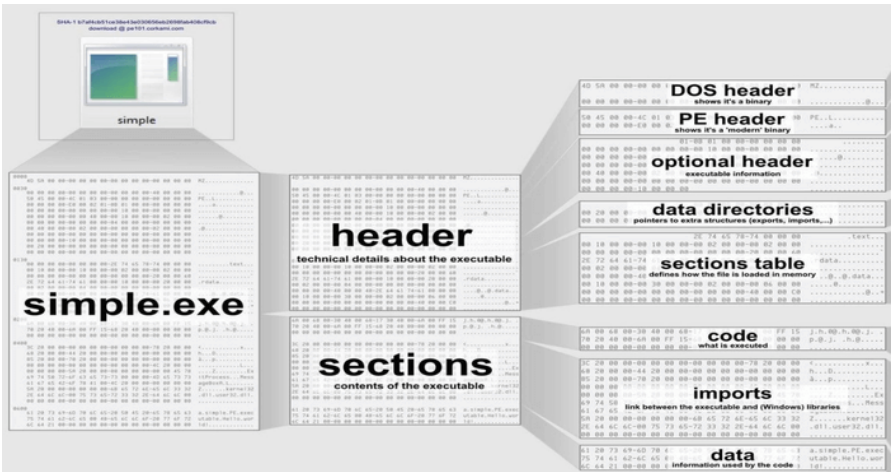

Figure 2: The process of selecting and extracting the features of the malware

3.2.2. The list of features

The features in Table 1 below are used in our malware detection technique using the MLP algorithm.

Table 1: List of features

Features	Description	Type
has_configuration	True if the PE has a Load Configuration	Boolean
has_debug	True if the PE has a Debug section.	Boolean
has_exceptions	True if the PE is using exceptions.	Boolean
has_export	True if the PE has any exported symbol.	Boolean
has_import	True if the PE is importing any symbol.	Boolean
has_nx	True if the PE has the NX bit set.	Boolean
has_relocations	True if the PE has relocation entries.	Boolean
has_resources	True if the PE has any resource.	Boolean
has_rich_header	True if a rich header is present.	Boolean
has_signature	True if the PE is digitally signed.	Boolean
has_tls	True if the PE is using TLS	Boolean
entrypoint0 - entrypoint64	The first 64 bytes of the PE entry point function	Float
byte(00) - byte(ff)	Repetition of each byte in the ASCII table	Float
import(...)	Frequencies of API used in the PE	Float
vsize_ratio	Ratio of the PE size of disk vs virtual size	Float
code_sections_ratio	Ratio of the code-containing section VS the data-containing section	Float
pec_sections_ratio	Ratio of portable executable sections	Float
sections_avg_entropy	The average Shannon entropy	Float
sections_vsize_avg_ratio	The average Ratio of sections physical and virtual size	Float

4. EXPERIMENTS AND EVALUATIONS

4.1. Installation requirements

The experimental result of the paper is performed on the server with the following configuration:

- Software Installations: Python version 3.6.7; Tensorflow-gpu 0.13; networkx 2.4; Ubuntu 16.04.6.
- Hardware requirements: RAM 32GB; CPU Intel Core i5-7500 CPU @3; 4GHz; GPU GeForce GTX 1080 Ti; 1TB SSD Harddisk.

4.2. Data and experiment process

a) Preparing data

- Benign data [13]: 24603
- Malware [13]: 24526

b) Experiment script

Export data (including harmless data and malware at the 80% – 20% ratio for testing).

c*) Calculation parameter*

Table 2 below lists the parameters used to evaluate the effectiveness of the model and the accuracy of the algorithm.

Table 2: Measured values

Parameter	Definition	Calculationprocess
TP	True Positive – result of finding malware correctly	Count number of malware and it was corected (files with malware equal files with files labeled 'malware')
TN	True Negative – result of finding malware incorrectly	Count number of malware and it was incorrect (files with malware not equal files with files labeled 'malware')
FP	False Positive - result of finding harmless file correctly	Count number of harmless files and it was corected (files with harmless files equal files with files labeled 'malware')
FN	False Negative – result of finding malware incorrectly	Count number of malware and it was corected (files with malware equal files with files labeled 'malware')

Where:

$$acc = \frac{TP + TN}{TP + TN + FP + FN} \times 100\ \%$$

$$precision = \frac{TP}{TP + FP} \times 100\ \%$$

$$\text{Re } call = \frac{TP}{TP + FN} \times 100\ \%$$

$$F1 = \frac{2 \times precision \times \text{Re } call}{precision + \text{Re } call}$$

4.3. Experimental results

Table 3 below shows the results of classifying malware using the MLP algorithm. In which, to evaluate the effectiveness of the MLP algorithm, we change the hidden layers and architecture of MLP.

Table 3: Experimental results of detecting malware using the MLP algorithm

HiddenLayers	1	2	3	4
Accuracy	0.8776	0.8662	**0.8889**	0.8880
F1 Score	0.8778	0.8735	**0.8907**	0.8893
Recall	0.878	0.866	**0.8889**	0.888
Precision	0.878	0.871	**0.8889**	0.888
Time taken to build model (s)	196.21	461.55	564.12	697.4
Time taken to test model (s)	49.63	24.86	22.57	22.01

The experimental results in Table 3 show that the use of MLP with 2 hidden layers model architecture gives the lowest result. Besides, the accuracy of the model increases when the scale of the model increases. With model 3, the malware detection process gives the best results for all metrics. However, in the 3 hidden layersMLP model, the time for training and testing is relatively high. Thus, we can conclude that the deep learning algorithm is perfectly suitable to solve the problem of detecting suspicious files quickly and accurately. With 4 MLP models with different architectures in the experimental section, can flexibly choose a suspect file detection model that matches either the purpose of prioritizing the detection accuracy or the purpose of thorough detection and not omission.

5. CONCLUSION

Detecting and warning malware is one of the most urgent issues right now for the task of preventing cyber-attacks. In this paper, with the support of the MLP algorithm and the proposed features about abnormal behavior of file, we succeeded in processing, analyzing, and detecting malicious files. The innovation of our approach is seeking and extracting characteristic abnormal behavior of files based on both static and dynamic analysis. Experimental results that are presented in Table 3 show that the MLP algorithm was successful in classifying clean and malicious files. In the future, we will improve the accuracy of the detection process by using deep

learning algorithms with more complex architecture. However, in fact, it has been shown that applying deep learning algorithms to classify malware can bring good results, but to prevent attacks through malware, human factors related to ethics, the sense of information security is still the most important.

REFERENCES

[1] A. Shabtai, R. Moskovitch, Y. Elovici, C. Glezer, **Detection of malware by applying machine learning classifiers on static features**: *A stateof-the-art survey*, Inf. Secur. Tech. Rep. 14 (1) (2009) 16–29.

[2] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, J. Nazario, **Automated classification and analysis of internet malware**. *Recent advances in intrusion detection, Springer*, 2007, pp. 178–197.

[3] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, E. Kirda, **Scalable, behavior-based malware clustering**, i*n: NDSS, Vol. 9, 2009*, pp. 8–11.

[4] K. Rieck, P. Trinius, C. Willems, T. Holz, **Automatic analysis of malware behavior using machine learning**,*Journal of Computer Security* 19 (4) (2011) 639–668. https://doi.org/10.3233/JCS-2010-0410

[5] S. Palahan, D. Babi´c, S. Chaudhuri, D. Kifer, **Extraction of statistically significant malware behaviors,** in*: Computer Security Applications Conference*, ACM, 2013, pp. 69–78.

[6] M. Egele, M. Woo, P. Chapman, D. Brumley, **Blanket execution: Dynamic similarity testing for program binaries and components**, in: *USENIX Security '14*, USENIX Association, San Diego, CA, 2014, pp. 303–317.

[7] M. Lindorfer, C. Kolbitsch, P. M. Comparetti, **Detecting environmentsensitive malware**, in: *Recent Advances in Intrusion Detection, Springe*r, 2011, pp. 338–357.

[8] IMPORTANT INFORMATION REGARDING SANDBOXIE VERSIONS. https://www. sandboxie.com/ . [Accessed February 15, 2020].

[9] Hassan Ramchoun; Mohammed Amine JanatiIdrissi; Mohammed Amine; Youssef Ghanou. **Multilayer Perceptron: Architecture Optimization and Training**. *International Journal of Interactive Multimedia and Artificial Intelligence.* 2016, 4, 26-30. https://doi.org/10.9781/ijimai.2016.415

[10] Do Xuan, Cho, Nguyen, HoaDinh, and Dao, Mai Hoang. **APT Attack Detection Based on Flow Network Analysis Techniques Using Deep Learning.***Journal of Intelligent & Fuzzy Systems.* pp. 1 – 17,. 2020. DOI: 10.3233/JIFS-200694

[11] Abien Fred Agarap. **Deep Learning using Rectified Linear Units** (ReLU). *arXiv 2018, arXiv:1803.08375*.

[12] KaiboDuan; SathiyaKeerthi, S.; Wei Chu; ShirishKrishnajShevade; AunNeow Poo. **Multi-category Classification by Soft-Max Combination of Binary Classifiers.***In proceedings of the 4th International Workshop*, MCS 2003 Guildford, UK, 11–13 June 2003; pp 125–134. https://doi.org/10.1007/3-540-44938-8_13

[13] HOW TO CREATE A MALWARE DETECTION SYSTEM WITH MACHINE LEARNING. https://www.evilsocket.net/2019/05/22/How-to-create-a-Malware-detection-system-with-Machine-Learning/?fbclid=IwAR1vuaOJA3UryaQATPsqKErktLft2RtzzAB5kDvgOTo4U3dF4J-Op9teokQ