The Investigation of Malware Detection Model Construction Based on Deep Learning Algorithms

Jiansen Lu
Department of Cyberspace Security, Huazhong University of Science and Technology, Wuhan, 430074, China

Keywords: Component; Malware Detection, Machine Learning, Deep Learning.

Abstract: Over the past decade, the evolution of computer malware has been swift. Current malware detection technologies primarily rely on static and dynamic detection methods, yet both struggle to achieve high levels of detection efficiency and accuracy. Furthermore, the majority of existing malware detection methods build classification models using machine learning algorithms, often resulting in less-than-ideal detection outcomes. In light of these limitations within current malware detection techniques, this paper proposes a deep learning-based approach to malware detection. The article begins by categorizing malware types and then presents an overview of traditional malware detection methods. It subsequently delves into the fundamental concepts of deep learning algorithms. Following that, a novel malware detection model is constructed leveraging deep learning techniques. To conclude, a dynamic malware detection model is established using deep learning, and a virtual platform for malware detection is implemented. Ultimately, this platform automatically extracts worm feature codes, enabling real-time dynamic software detection.

1 INTRODUCTION

Malware, also known as rogue software, is a virus (Shara 2021) that is generated in the process of Internet development and forcibly runs without user permission and affects user use. The progress of computer technology has also led to the increasing number of rogue software and its composition becoming more and more complex (Suciu et al 2018), which has seriously affected network security. In addition, affected by information technology, the attack methods of malware are constantly updated, so it is urgent to detect malware. The traditional detection methods of malware are limited by the number and attack methods of malware. Unable to meet the existing detection requirements, it still relies on manual methods to extract the relevant information in the detection, so it cannot meet the existing detection requirements, and it is urgent to design a new malware dynamic detection method to solve the existing malware dynamic detection problem.

The popularity of smart devices has broadened the transmission channels of malware, and more and more malware has taken advantage of the speed of information transmission to illegally steal user information, posing a great threat to users' network information security. Therefore, malware has become an important issue affecting users' network security. Relevant researchers investigated the changes of malware in 2018 (Li et al 2020). The results show that the number of malicious software is increasing rapidly, and the total number of samples in one quarter even exceeds 63 million (Dong et al 2023), and the trend continues to increase. In this case, various network security companies have studied the relevant dynamic detection methods of malicious software, but due to the limitation of feature codes, these detection methods have obvious shortcomings. Deep learning can carry out intelligent analysis and identification according to the expression rules of samples (Javaid et al 2015), and has a prominent effect in the feature recognition of malware.

In recent years, there has been a lot of research on how to detect malware effectively. Wang Yinglong et al. built a malware classification model by converting binary malware into grayscale maps and extracting texture features in the images (Wang et al 2018). Luo Wenyan et al. extracted non-user operations and sensitive Application Programming Interface (API) sequences, and finally determined the similarity of sequences by measuring distance (Luo et al 2018). Xiang Zihao et al. used sandbox to extract executable files to operate the file system and registry to invoke

Lu, J.
The Investigation of Malware Detection Model Construction Based on Deep Learning Algorithms.
DOI: 10.5220/0012808200003885

In *Proceedings of the 1st International Conference on Data Analysis and Machine Learning (DAML 2023)*, pages 332-337
ISBN: 978-989-758-705-4

API and other features (Xiang et al 2018). Mohaisen et al. extracted the dynamic detection features of malware during its execution, and used K-Nearest Neighbor (KNN), Support Vector Machine (SVM), decision tree and other classification algorithms to build the final detection model (Mohaisen et al 2015). Xin Wang et al. proposed a detection model that combines Recurrent Neural Network (RNN) networks and autoencoders (Wang et al 2016). Mehadi Hassen et al. used the supervised learning algorithm to learn low-dimensional features, and finally used the anomaly detection method to detect malware (Hassen et al 2018). S. Pai et al. used the expectation maximization algorithm in the cluster detection analysis of malware (Pai et al 2017).

Current malicious software detection primarily relies on two main methods: static analysis and dynamic analysis. Most of these methods use either a single machine learning algorithm or a combination of multiple learning algorithms to construct classification detection models. Static analysis is efficient but may have lower accuracy, while dynamic analysis provides high accuracy but may be less efficient. Therefore, relying solely on static or dynamic analysis alone may not simultaneously meet the dual requirements of high efficiency and high accuracy. Hence, this paper introduces a malicious software detection technique based on deep learning algorithms. The fusion of dynamic and static detection techniques ensures that the final detection process is both efficient and accurate.

2 TRADITIONAL MALWARE DETECTION METHODS

2.1 Static Analysis

Static analysis is the initial phase of the malicious software analysis process, primarily focused on examining executable files without delving into specific instructions. Basic static analysis can determine the presence of malicious characteristics in a file, offer insights into its expected functionality, or generate fundamental network feature identifiers. However, static analysis has its limitations when dealing with complex malicious software and may occasionally overlook critical malicious behaviours.

2.1.1 Message Digest Algorithm 5

Message Digest Algorithm 5 (MD5) is a commonly used technique for identifying malicious software. The MD5 method involves subjecting malicious software to a hash function, resulting in a unique hash value generated for each malicious software instance. In the field of deep learning, feature extraction hashing is a commonly employed algorithm that can map data of varying sizes into standardized fixed-size representations.

2.1.2 PEiD Detection

PEiD is a common way to detect wrapped files, and is often used to detect files generated by a packer or compiler. Because malware is often packaged or obfuscated, the malicious files it generates are more difficult to detect, which can seriously hinder the analysis of malware. PEiD also has a security risk in its work, because its plug-ins tend to automatically run malicious executables, so it needs to create a safe environment for malicious operation and analysis.

2.1.3 Executable File Format Analysis

PE file format is a type of data structure, and almost all executable code files loaded in Windows systems are PE file formats. The PE file starts with the header and includes information such as code, application type, library functions, etc. The information in the header of the file is valuable to malware analysts.

2.1.4 Interactive Disassembly Expert (IDA Pro)

As an advanced static analysis method, IDA Pro is also the preferred disassembly tool for most malware analysts and vulnerability analysts (Raff et al 2017). Strings are the starting point for malware static analysis, using their cross-reference feature to see exactly where and how strings are used in code, and disassembler provides a snapshot of the program before the first instruction is executed.

2.2 Dynamic Analysis

Dynamic analysis is the second phase in the process of analysing malicious software. It is typically employed when basic static analysis fails to yield definitive results. Dynamic analysis involves monitoring the behaviour of malicious software while it actively runs or examining system changes after the execution of malicious software. Unlike static analysis, dynamic analysis provides a deeper understanding of the actual functionality and internal workings of malicious software. It has been proven to be an effective method for identifying malicious software.

2.2.1 Sandbox

Sandbox technology serves as a fundamental dynamic analysis tool while also functioning as a security mechanism. Its primary role is to establish a secure environment that allows untrusted programs to execute within it without posing a real threat to the underlying system, effectively addressing the limitations of PEiD detection. Sandboxes rely on predefined conditions in a virtual environment to simulate network services, ensuring that the software or malicious software being examined behaves normally. However, it is worth noting that sandboxes also have their limitations; they lack command-line options for executing specific commands, and due to inadequate waiting times, they may fail to capture all events.

2.2.2 Procmon

Process Monitor (Procmon for short) is an advanced tool in Windows that monitors the activities of certain registries, network processes, and threads to determine whether there is malicious behaviour. With Procmon, you can monitor all system calls at run time, but it is not possible to check them all. In addition, its long work takes up available memory so that it runs out of memory, eventually rendering the virtual machine inoperable.

2.2.3 Process Explorer

Process Explorer, as the task manager, is often used when performing dynamic analysis. It can provide a favourable analysis of the processes running within the system (including processes, DLLS, system information). It can also be used to start, validate, and terminate processes. The Process Explorer visualizes the process being monitored in a tree structure.

2.2.4 OllyDbg

OllyDbg is an advanced dynamic debugger for malware analysis testing. A debugger is a piece of software or hardware that tests the performance of another program and sees its dynamic view while the program is running, information that is difficult to obtain from a disassembler. As an upgraded version of OllyDbg, WinDbg can assist it in kernel debugging and Rootkit analysis.

Dynamic techniques have their limitations as they may not cover the complete code path during the execution of malicious software. Currently, a more effective approach is to leverage advanced dynamic or static techniques to address the challenges of compelling malicious software to execute.

2.2.5 Common Detection Tools

VirusTotal is an online service that uses many different antivirus programs to scan for malware, as well as a comprehensive malware dataset. Wireshark is a tool that dynamically analyses network packets and network protocols. It can capture malicious network traffic and analyse many different network protocols. PE Explorer is an analysis tool for viewing executable file formats and is used as a static decompressor to automatically unpack files. Capture BAT is a dynamic analysis tool for monitoring running malware that monitors file system, registry, and process activity.

3 DEEP LEARNING ALGORITHMS

Deep learning is gradually replacing some traditional machine learning algorithms, becoming a prominent focus of research in the field of artificial intelligence. Unlike traditional machine learning algorithms, deep learning has the capability to stack multiple layers of different neural networks, forming deep neural networks (Tao et al 2016). The deep learning algorithm employed in this article is a type of artificial neural network that structurally resembles the connectivity patterns in biological neural networks, with multiple neurons within each layer. Artificial neural networks process input starting from the input layer, then perform successive mappings through hidden layers, ultimately generating outputs at the output layer. Figure 1 illustrates the architecture of an artificial neural network (Tao et al 2016).

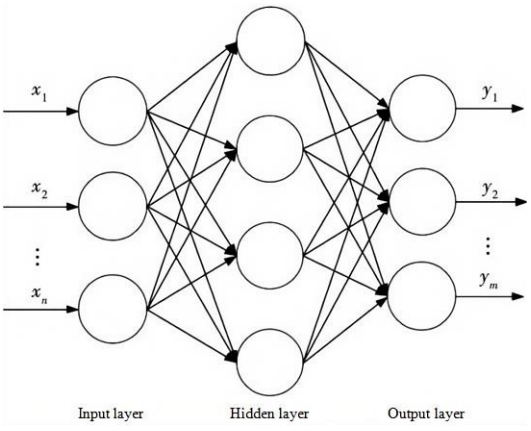

Figure 1: Artificial neural network structure diagram.

When a neuron receives input from the neuron above it, it computes an output result using a mathematical transfer function and transmits this result as the input to the neuron below it. In practical research, commonly employed transfer functions include the Rectified Linear Unit (ReLU) function, the hyperbolic tangent (tanh) function, the sigmoid function, and various other mathematical formulas:

$$\mathrm{Re}\,LU(x)=\begin{cases}x, & if \quad x>0\\ 0, & if \quad x\le 0\end{cases} \tag{1}$$

$$\tanh(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2}$$

$$sigmoid(x)=\frac{1}{1+e^{-x}} \tag{3}$$

The loss function is also used in the optimization of the trained model, and the loss function can also be called the cost function, which can make the predicted output value better fit the actual target value. In the construction of classification models, the loss function often uses the cross-entropy function, the formula is:

$$D_{CE}=-\sum_{i=1}^{N} p\left(x^{(i)}\right)\log q\left(x^{(i)}\right) \tag{4}$$

4 MALWARE DETECTION MODEL

Through the analysis of the malware detection methods based on static features and dynamic features, it can be found that the static malware detection method has higher detection efficiency, but it is easy to miss the alarm, while the dynamic malware detection method has higher detection accuracy, but the detection time is relatively large. Considering the advantages and disadvantages of the two methods, this paper presents a new malware detection method which is static first and then dynamic. When training the malware classification model, the static feature and dynamic feature are extracted respectively from the malware training sample set, and then the malware classification model based on static feature and the malware classification model based on dynamic feature are trained respectively by artificial neural network algorithm.

The specific workflow of the new malware detection method proposed in this paper is shown in Figure 2. After static and dynamic malware classification models are obtained, the unknown software is detected. First, the static features of the unknown software are extracted. If it is judged to be normal software, it is necessary to carry out further dynamic detection analysis of the unknown software. The dynamic features of unknown software are extracted and analyzed by the malware dynamic detection model. If the unknown software is classified as normal software by the dynamic detection model, the software can be classified as normal software; otherwise, it can be classified as malware. The new malware detection technology combines the advantages of static detection and dynamic detection methods, which can not only reduce the time cost of detection process, but also ensure the accuracy of final detection.

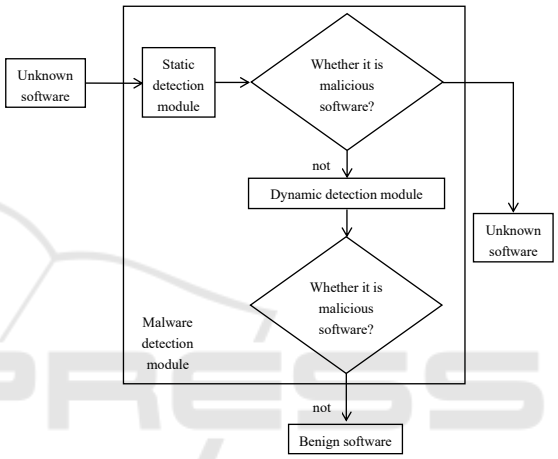

Figure 2: Architecture diagram of new malware detection methods (Photo/Picture credit : Original).

5 CONSTRUCTION OF MALWARE DETECTION MODEL BASED ON DEEP LEARNING

Deep learning is an important branch of machine learning (Qiu et al 2020 & Shen et al 2017). In the face of massive network application traffic data, it is difficult for traditional detection methods to effectively judge the characteristics of new malicious programs, even if the malicious characteristics database is designed, it needs a lot of manpower cost. The intelligent malicious detection method designed based on deep learning technology can effectively detect newly emerged samples, greatly reduce the workload of manual participation, and has strong generalization ability. Here are some typical deep learning detection models.

5.1 MalConv Model

The MalConv model is constructed based on the Gated Convolutional Neural Network (Gated-CNN) architecture, designed for end-to-end malicious code detection. The input layer receives binary files as the dataset (Wu 2020), with a focus on analyzing the PE header information. During the embedding layer, data preprocessing occurs, generating a fixed-size feature map matrix that maps each input byte to a D-dimensional vector. Convolutional layers perform the dot product operation between feature maps and convolutional kernels. Gate layers are employed to address the gradient vanishing problem, and max-pooling is utilized for down sampling. Finally, the probability of being a malicious program is obtained through fully connected layers and a SoftMax layer.

5.2 ScaleMalNet Model

ScaleMalNet is an adaptable deep learning architecture designed for malicious software detection. This framework efficiently collects malicious software samples from various sources in a distributed manner and preprocesses them at scale. It possesses real-time capabilities to handle a large volume of malicious software samples, meeting on-demand processing requirements. ScaleMalNet adopts a comprehensive learning approach, analysing malicious software collected from end-user hosts using a two-stage malware analysis method. In the initial stage, a combination of static and dynamic analysis techniques is applied to classify malicious software. In the subsequent stage, image processing methods are utilized to categorize the malicious software into their respective malware types. Figure 3 illustrates the real-time analysis framework of ScaleMalNet for malicious programs (Vinayakumar et al 2019).

5.3 DroidDetector Model

DroidDetector is an Android malicious software detection engine developed using deep learning technology (Yuan et al 2016). It primarily utilizes association rule mining techniques to characterize malicious software attributes. This engine provides online malicious software detection, automatically inspecting applications for malicious features. Upon receiving an .APK file of an application, DroidDetector verifies its integrity to determine whether it is a complete, accurate, and legitimate Android application.

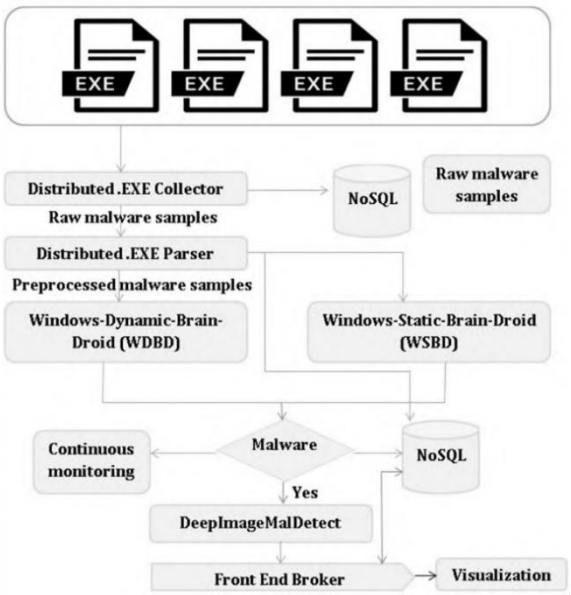

Figure 3: ScaleMalNet model real-time analysis frame diagram.

Subsequently, DroidDetector conducts static and dynamic analysis. The static analysis phase involves extracting information related to the permissions and sensitive APIs used by the application. In the dynamic analysis phase, the application is installed and run within DroidBox for a period to identify its dynamic behavior. Fig. 4 illustrates the framework model of DroidDetector (Yuan et al 2016).

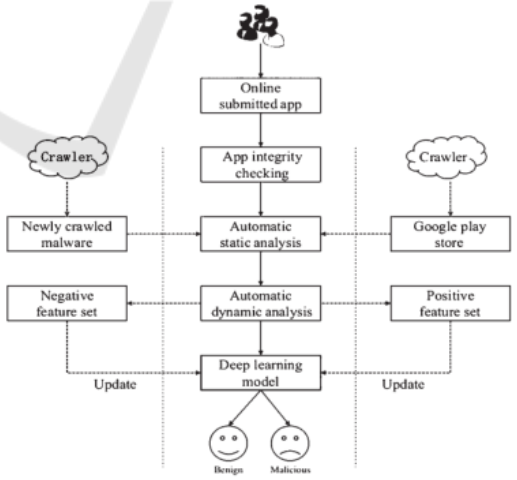

Figure 4: DroidDetector frame structure diagram.

6 CONCLUSION

As information technology continues to advance, the malicious software created by attackers is becoming increasingly sophisticated, making it challenging for computer users to defend against these threats. Malicious software can compromise user privacy, disrupt computer systems, and often result in severe consequences and significant economic losses for individuals and organizations. Therefore, the development of effective malicious software detection techniques is of paramount importance. Currently, existing malicious software detection techniques primarily rely on single static or dynamic detection methods, which may not effectively overcome the inherent limitations of each approach. Therefore, this paper introduces a two-stage detection method: firstly, static analysis is used to identify easily detectable malicious software, while the remaining samples considered as normal software undergo dynamic detection and analysis to mitigate potential oversights in static analysis. The dynamic detection method is employed to determine the characteristics of unknown software. Compared to other single detection methods, the approach proposed in this paper not only ensures the efficiency of malicious software detection but also guarantees a high level of accuracy in the final verdict.

Current malicious software detection techniques typically leverage machine learning algorithms to construct classification models. Traditional machine learning algorithms heavily rely on the quantity of samples and feature selection, which can impact detection performance. Therefore, this paper introduces the application of deep learning algorithms to build malicious software classification models, effectively enhancing the effectiveness of malicious software detection.

REFERENCES

J. Shara. Deep Learning Methods for Cybersecurity. ICSNS XV-2021 (2021).

O. Suciu et al. Exploring Adversarial Examples in Malware Detection (2018).

H. Li et al. Adversarial-Example Attacks Toward Android Malware Detection System. IEEE systems journal, (1):14(1) :653-656 (2020).

S. Dong et al. Action recognition based on dynamic mode decomposition. Journal of ambient intelligence and humanized computing (2023).

A. Y. Javaid et al. A Deep Learning Approach for Network Intrusion Detection System. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering) (2015).

Y. L. Wang et al. Test of malicious code detection method based on texture features. Yunnan Electric Power Technology, 46(1):131 - 133 ,138 (2018).

W. Luo et al. Malware detection method based on non-user operation sequence. Computer Application, 38(1):56 - 60,66 (2018).

Z. Xiang et al. Ransomware detection method based on machine learning. Information Technology, 42(5):79 - 82,89 (2018).

A. Mohaisen et al. Amal: high-fidelity. behaviour-based automated malware analysis and classifi-cation. Computers & Security,52 ;251 – 266 (2015).

X. Wang et al. A multi-task learning model for mal-ware classification with useful file access pattern from API call sequence. CoRR, (2016).

M. Hassen et al. Learning a neural-network-based representation for open set recognition. arXiv preprintarXiv:1802.04365, (2018).

S. Pai et al. Clustering for mal-ware classification. Journal of Computer Virology and Hacking Techniques,13(2):95 -107 (2017.)

E. Raff et al. Malware Detection by Eating a Whole EXE. DOI:10.48550/arXiv.1710.09435 (2017).

M. Tao et al. A Hybrid Spectral Clustering and Deep Neural Network Ensemble Algorithm for Intrusion Detection in Sensor Networks.Sensors, 16(10):1701 (2016).

Y. Qiu et al. Improved denoising autoencoder for maritime image denoising and semantic segmentation of USV. China Communications, 17(3): 46-57 (2020).

D. Shen et al. Deep learning in medical image analysis. Annual review of biomedical engineering, 19: 221-248 (2017).

R. Wu. Research on attack and defense methods of malicious code classifier based on adversarial principle. Harbin: Harbin Institute of Technology, (2020).

R. Vinayakumar et al. Robust intelligent malware detection using deep learning. IEEE Access, 7;46717-46738 (2019).

Z. Yuan et al. Droid detector: Android malware characterization and detection using deep learning. Tsinghua Science and Technology, 21(1):114-123 (2016).