OPEN ACCESS

Volume: 11

Special Issue: 1

Month: July

Year: 2023

E-ISSN: 2582-0397

P-ISSN: 2321-788X

Impact Factor: 3.025

Received: 02.05.2023

Accepted:14.06.2023

Published: 01.07.2023

Citation:
Deepa, KR, and V. Bhavyashree. "Deep Learning Malware Detection Using Auto Encoder." *Shanlax International Journal of Arts, Science and Humanities*, vol. 11, no. S1, 2023, pp. 42–48.

DOI:
https://doi.org/10.34293/sijash.v11iS1-July.6314

Deep Learning Malware Detection Using Auto Encoder

Deepa K.R
Department of Master of Computer Applications
Raja Rajeswari College of Engineering, Bengaluru

Bhavyashree V
Department of Master of Computer Applications
Raja Rajeswari College of Engineering, Bengaluru

Abstract

Nowadays, in the arena of malware detection, the growing constraints of traditional detection methods, laterally with the improving precision of detection methods based on artificial intelligence algorithms, are moving research findings in this zone in favour of the latter. As a result, we offer a novel. This work includes a malware detection model. In a deep learning model, this model combines a grey-scale picture representation of malware with an autoencoder network, examines the viability of the grey-scale image approach malicious software based on the autoencoder reconstruction error, and employs the dimensionality.The auto encoder's reduction characteristics are used to distinguish malware from benign software. Using the suggested detection model, the proposed detection model attained an accuracy of 96% and a steady F-score of about 96%. We collected an Android-side dataset that outperformed certain classic machine learning detection techniques.Malware detection, autoencoders, malware images, mobile application security.

Keywords: Autoencoder, Grey-scale image, F-score

Introduction

The fast development of mobile internet technology in the outcomes of the last couple of centuries in the rise the software industry. Every day, malware is spreading more and more. Considering the most recent China Internet Annual Network Security Report [1], there were 13,510,900 cases of mobile Internet malware programmes as of 2019, with over 2,791,300 new cases introduced this year alone. Since of the open the nature of Android application market, the Android system has been the secret to numerous mobile-based malware assaults.With the rise in Android malware security concerns, it is critical to devise an effective and creative mobile malware detection approach to address the issue.

Current malware identification approaches are constrained by the amount of detection criteria that must be manually configured. In today's world of rising malware, it is hard to identify many new malware versions [2]. With the rise of artificial intelligence, malware detection approaches integrated with AI algorithms have performed better in recent years. These detection approaches are more accurate, resilient, and generalizable than standard malware detection

techniques, and they can reduce the danger of false detection in the case of many freshly developed malware[3]. As a conclusion, investigating malware detection systems based on these algorithms is more intriguing scientifically.

In the information preliminary processing timing, popular ways to extract for data with features include static extraction and dynamic extraction. Static the outcomes of the last couple of centuries without executing the software programme, such as obtaining byte code file header information. The fundamental premise of static analysis features is to retrieve the program's source code or byte code via software de-compilation and analyse the semantic features and semantic information stored within it. Such detection methods include MaMadroid proposed by Mariconti et al. and a malware detection technique proposed by Wenjin Li et al[4], that used Android-side application permission information, API call information, and other static data for malware detection.

Literature Survey

Malware Identification in Android

Static evaluation screens application components without actually executing them. W. Li and colleagues created a malware finding method based on a deep belief network. They suggested API functions and risky privileges as two categories of Android app features for malware classification. the programme. analysis, which looks at the Android API calls made by a software programme, was the focus of R. Nix et al[5]. How an application relates to the system that runs on Android is determined by network API calls. A software programme needs this kind of interactions for proper operation, and hence provides critical data about an application's behaviour.

Deep Refiner malware identification system created by K. Xu et al. use complex networks with a variety of layers that are invisible. XML values are extracted from XML files by DeepRefiner during the preparation phase. Retrieving byte code semantics from deconstructing modules at the initial detection layer. The dex file is located in the second detection layer [6]. Applications are then referred to by DeepRefiner as matrices that sophisticated neural networks may use as inputs. Through non-linear transformation hidden layers in brain networks produce recognition traits from incoming matrices.

Detecting Malware Via Dynamic Analysis

The programme is executed on either a virtual computer or a real device as segment of the dynamic analysis approach. H. Liang et al. created based on the assumption the subject painting and ransomware classification are comparable, methods of natural language processing for Android malware examination have been developed. They developed a technique that considers pathogen of malware as thematic extraction and treats system calling sequences as texts[7]. The system calls were initially translated into vector space using an embedding layer. The excessive presentation matrices of the extended sequence were then processed by the vertical multiple convolutions modules for obtaining probable the highest level knowledge. The regression function was completed using a perception that had multiple layers with a SoftMax layer[8].

Existing Approach

Modern spyware identification approaches are limited by the amount of detection criteria that must be manually configured. In today's world of rising malware, it is hard to identify many new malware versions [2]. With the rise of artificial intelligence, malware detection approaches integrated with AI algorithms have performed better in recent years. These detection approaches are more accurate, resilient, and generalizable than standard malware detection techniques, and they can reduce the danger of false of false detection when it comes to many freshly developed

malware[8]. As a conclusion, it is greater scientifically interesting to investigate malware detection systems based on these algorithms.

The major approaches in the classical development and classification phase include malware revealing systems based on deep learning models and machine learning techniques. In techniques like those utilised by Wang et al. [18], who used five machine learning models for software classification, including the Support Vector Machine, machine learning algorithms are predominantly used as classification models. (SVM), K-Nearest Neighbour (KNN), Naive Bayes (NB), Classification Regression Tree (CART), and Random Forest (RF). Kumar et al. suggested a feature learning model that employs a collection several approaches for machine learning identify malware with little overhead and good accuracy.

In this paper, we extract static characteristics from the bytecode of several command methods of Android applications [9]. The grey-scale picture corresponding to each malware is then reconstructed using an auto-encoder convolution neuronal network-based architecture. Finally, the auto-encoder is tested in use of recreating the high-dimensional aspects of malware performance. To accomplish malware categorization and detection, we created a neural network with an auto-encoder-based design. Additionally, experiments were conducted using data sets provided from VisureShare. The studies' findings show that our strategy excels over existing machine learning techniques and certain deep learning malware detection models based on malware photos.

Proposed System

We suggest a method for creating feature pictures for each type of malware and gentle software. The basic strategy is to transform the software's byte code's for the various procedures into grey-scale pictures for further model training and classification[10]. We employed an auto-encoder based on a convolution neuronal network to detect the high-dimensional characteristics in such grey-scale pictures, and we experimentally verified the scheme's practicality. We present a model of a neural network based on auto encoder networks for the malware detection classification problem and empirically demonstrate its excellent accuracy.

This part highlights the work on malware picture production and the static malware detection consider deep learning-based models[18]. As a result, this segment is separated into two parts: the malware picture production method and the static malware revealing performance deep learning-based models. Artificial neural networks can be applied to feature extraction phase for the recognition of malware to remove the corresponding features of the software additional to extracting the corresponding static feature information, such as API calls, permission information, and so on, dynamic component information, such as network activity, log files, and so on[11]. This methodology for feature extraction is more automated and simpler than previous manual feature extraction methods.

The data representation must be considered while automatically extracting software features using neural nets. So that it can extract the main features more effectively and assure the correctness of the test findings. We present a malware detection strategy that is founded on automated encoder network[12]. Figure 2 depicts the general structure and primary responsibilities of our technique for detecting malware. First, innocuous files and malware are converted into appropriate greyscale graphics by decompiling the APK files. The retrieved coded data from software methods and bytes are transformed into decimal data and coated with pixel values. The greyscale photos are then processed by two deep learning networks to perform two tasks. The first deep learning network is called automated encoder network - 1 (AE-1) and is operated to investigate the possibility of employing grey-scale photographs to characterise the relevant properties of software's[13]. The second deep learning network is called automatic encoder network - 2 (AE-2), which we utilise

to distinguish between harmful and benign software. The next parts will go through the thorough design process of AE-1 and AE-2.

The primary goal of the feature data pre-processing step is to supply input data for the neural network model. To symbolize the characteristics of the software, we use a grey-scale image of the software byte code[17]. The so-called grey-scale image of the software bytecode is to decompile the software to obtain its binary byte code, then convert it into a decimal type by byte and fill it into a fixed size two-dimensional matrix, because a byte is 8 bits, that corresponds exactly to the range of data from 0 to 255 and can be composed as a grey-scale image[14].The benefits of utilising this strategy are twofold. For starters, this way of extracting software functionality is less time consuming and more straightforward. Second, because our subsequent network model is composed of a the use of a convolution neural network, which a fixed size multi-dimensional matrix type of data, the grey-scale image converted from the software file byte code is a suitable input to the convolution network for training and classification.

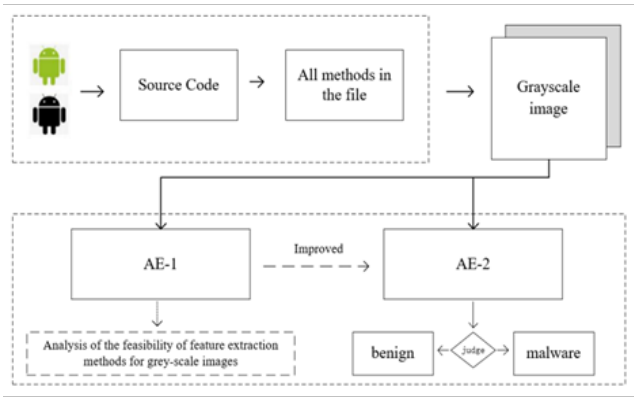


Figure 1 Proposed Architecture

However, there are a number of drawbacks to converting software binary information into greyscale graphics. Although the software binary code contains a variety of features, it also contains a large amount of works that focus on how to convert software of different sizes into images of the same size, and must consider the sticking points of how to do the best possible job of reducing redundancy in the image generation process [15]. Our work diverges from prior work in that we attempt to extract the binary code of the method field in the programme and convert a portion of the data into byte code to complete the development of the grey-scale image [16]. Analysing the viability of such a programme is an important aspect of our work. The duplicated information increases the pre-processing cost and lowers the accuracy and resilience of the model classification later on.

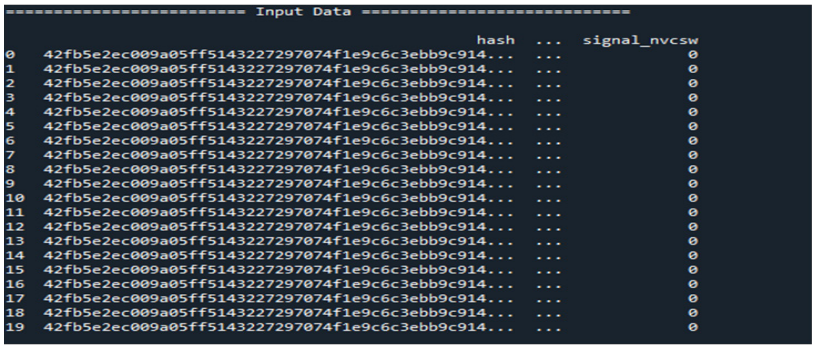
Implementation

Input Data: The input data was collected from the dataset repository. The data selection is the process of choosing the data to be used in malware detection. We must use the malware detection dataset for this project. The dataset that includes data about host, categorization (malware and benign), and other factors. We must use the pandas packages in Python to read the dataset. Our dataset is presented asa '.csv' file extension.

Pre-processing

Data pre-processing is the method of deleting undesirable data from the dataset. Operations for transforming pre-processed data are used to transform the dataset into a structure suitable for machine learning. Missing data removal: In this process, the null values such as missing values and Nan values are replaced by 0.Encoding Categorical data: That categorical data is defined as variables with a finite set of label values. Feature selection: In our process, we have to implement the feature selection such as principle component analysis (PCA).The Principle Component Analysis is an unsupervised learning the algorithm that is used for the dimensionality reduction in machine learning[16]. It is statistical process that converts the observations of correlated features into group of linearly uncorrelated features with the aid of orthogonal transformation.

Results

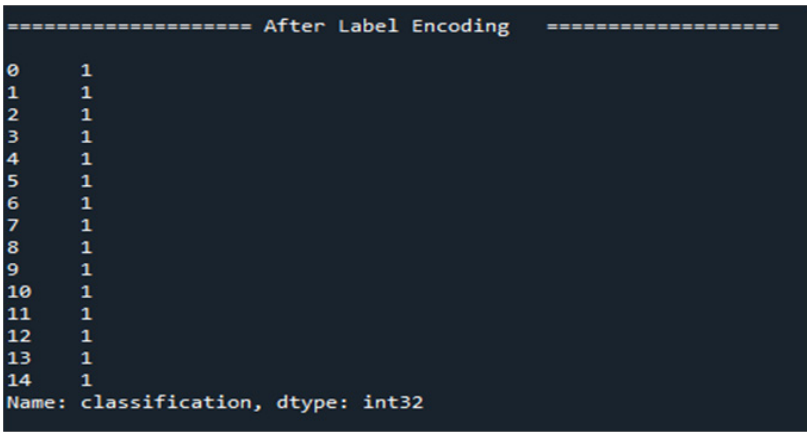


```
------------------------- Input Data ---------------------------

                                                hash  ...  signal_nvcsw
0   42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
1   42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
2   42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
3   42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
4   42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
5   42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
6   42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
7   42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
8   42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
9   42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
10  42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
11  42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
12  42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
13  42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
14  42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
15  42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
16  42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
17  42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
18  42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
19  42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
```

Figure 2 Data Selection

Figure 2. shows the data selection is the process of selecting the data for detecting the malware.



```
------------------- After Label Encoding  -------------------

0     1
1     1
2     1
3     1
4     1
5     1
6     1
7     1
8     1
9     1
10    1
11    1
12    1
13    1
14    1
Name: classification, dtype: int32
```

Figure 3 Preprocessing

Figure 3 shows the data pre-processing is the method of deleting undesirable data from the dataset. Operations for transforming pre-processed data are used to transform the dataset into a structure suitable for machine learning.

Conclusions

In this research, we offer a unique technique to malware detection according to the foundation of combining grey-scale photos to represent malware characteristics and an auto-encoder network

to create a classification model to recognise malware. The investigational findings demonstrate the practicality of our suggested method of transforming the bytecode of all software methods into a greyscale picture to denote the characteristics in a software sample. Our technology detects infection accurately than solutions that use conventional neural network algorithms. In relationship to previous malware detection systems based on deep learning models, our solution requires shorter training and detection time. In the future, we will inspect more efficient procedures for encoding malware feature photos, as well as focus our study on the data pre-processing step to investigate newer malware detection approaches.

References

1. (2019). China Internet Security Research Report. (Nov. 15, 2020). [Online]. Available: https://www.cert.org.cn/publish/main/upload/ File/2019Annual%20report.pdf
2. Ye, Y., Li, T., Adjeroh, D., &Iyengar, S. S. (2017). A survey on malware detection using data mining techniques. ACM Computing Surveys (CSUR), 50(3), 1-40.
3. S. Rastogi, K. Bhushan, and B. B. Gupta, ''Android applications repackaging detection techniques for smartphone devices,'' Proc. Comput. Sci., vol. 78, pp. 26–32, Jan. 2016.
4. Pandita, R., Xiao, X., Yang, W., Enck, W., &Xie, T. (2013). {WHYPER}: Towards Automating Risk Assessment of Mobile Applications. In 22nd USENIX Security Symposium (USENIX Security 13) (pp. 527-542).
5. Klieber, W., Flynn, L., Bhosale, A., Jia, L., & Bauer, L. (2014, June). Android taint flow analysis for app sets. In Proceedings of the 3rd ACM SIGPLAN International Workshop on the State of the Art in Java Program Analysis (pp. 1-6).
6. Wang, Z., Cai, J., Cheng, S., & Li, W. (2016, September). DroidDeepLearner: Identifying Android malware using deep learning. In 2016 IEEE 37th Sarnoff symposium (pp. 160-165). IEEE.
7. Schultz, M. G., Eskin, E., Zadok, F., &Stolfo, S. J. (2000, May). Data mining methods for detection of new malicious executables. In Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001 (pp. 38-49). IEEE in Proc. IEEE Symp. Secur. Privacy. (S&P), May 2001, p. 2001, doi: 10.1109/SECPRI.2001.924286.
8. B. P. Sarma, N. Li, C. Gates, R. Potharaju, C. Nita-Rotaru, and I. Molloy, ''Android permissions: A perspective combining risks and benefits,'' in Proc. 17th ACM Symp. Access Control Models Technol. (SACMAT), 2012, pp. 13–22.
9. C. Zhao, W. Zheng, L. Gong, M. Zhang, and C. Wang, ''Quick and accurate Android malware detection based on sensitive Apis,'' in Proc. IEEE Int. Conf. Smart Internet Things (SmartIoT), Aug. 2018, pp. 143–148.
10. H. Fereidooni, M. Conti, D. Yao, and A. Sperduti, ''ANASTASIA: ANdroidmAlware detection using STaticanalySIs of applications,'' in Proc. 8th IFIP Int. Conf. New Technol., Mobility Secur. (NTMS), Nov. 2016, pp. 1–5.
11. E. Mariconti, L. Onwuzurike, P. Andriotis, E. De Cristofaro, G. Ross, and G. Stringhini, ''MaMaDroid: Detecting Android malware by building Markov chains of behavioral models,'' 2016, arXiv:1612.04433.
12. W. Li, Z. Wang, J. Cai, and S. Cheng, ''An Android malware detection approach using weight-adjusted deep learning,'' in Proc. Int. Conf. Comput., Netw. Commun. (ICNC), Mar. 2018, pp. 437–441, doi: 10.1109/ICCNC.2018.8390391.
13. B. Amos, H. Turner, and J. White, ''Applying machine learning classifiers to dynamic Android malware detection at scale,'' in Proc. 9th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC), Jul. 2013, pp. 1666–1671.

14. S. Nari and A. A. Ghorbani, ‘‘Automated malware classification based on network behavior,’’ in Proc. Int. Conf. Comput., Netw. Commun. (ICNC), Jan. 2013, pp. 642–647.
15. G. Cabau, M. Buhu, and C. P. Oprisa, ‘‘Malware classification based on dynamic behavior,’’ in Proc. 18th Int. Symp. Symbolic Numeric Algorithms Sci. Comput. (SYNASC), Sep. 2016, pp. 315–318.
16. W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, ‘‘TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones,’’ ACM Trans. Comput. Syst., vol. 32, no. 2, pp. 1–29, Jun. 2014.
17. Wang, W., Li, Y., Wang, X., Liu, J., & Zhang, X. (2018). Detecting Android malicious apps and categorizing benign apps with ensemble of classifiers. Future generation computer systems, 78, 987-994.