Static Malware Analysis Using Machine and Deep Learning

Himanshu Kumar Singh, Jyoti Prakash Singh, and Anand Shanker Tewari

Abstract In the era of digital advancement and innovation, malware (malicious software) still poses major threats to users' privacy and leads to many security breaches. Due to the exponential rise in malware attacks, malware analysis and detection continue to be a hot research topic. Malware analysis plays a vital role in the malware detection process. Currently, the detection process adopts the malware signatures (static analysis) and behavior patterns (dynamic analysis) that have been proven time-consuming and less effective in identifying unknown malware in real time. Recent malware uses abstraction, packing, encryption, polymorphic, and other cryptic methods to hide and change the malware behavior and its signature which makes the detection process complex. Most of the new malware is the variants of existing malware, where machine learning techniques are effective in identifying such malware. However, the traditional machine learning technique is time-consuming because it requires substantial feature engineering and learning. By using the state-of-the-art learning technique such as deep learning, compel the learning process faster. By utilizing the high-level machine learning techniques, the training stage can be completely avoided. In this paper, first, we analyze the old-style MLAs and profound learning models for malware detection using publicly available datasets. Second, we analyze the deep learning models to examine the accuracy over the traditional machine learning technique. Third, our major commitment is in proposing an efficient and accurate model which combines the capabilities of the machine and deep learning technique which detect the zero-day malware efficiently. Our model shows that our proposed method outflanks traditional MLAs and deep learning models.

Keywords Malware · Static analysis · Dynamic analysis · Machine learning · Deep learning

H. K. Singh · J. P. Singh · A. S. Tewari (✉)
National Institute of Technology Patna, Patna, India
e-mail: anand@nitp.ac.in

H. K. Singh
e-mail: himanshus.pg19.cs@nitp.ac.in

J. P. Singh
e-mail: jps@nitp.ac.in

A. K. Bashir et al. (eds.), *Proceedings of International Conference on Computing and Communication Networks*, Lecture Notes in Networks and Systems 394,
https://doi.org/10.1007/978-981-19-0604-6_41

1 Inroduction

In this digital world, where modern technologies like 5G, Internet of Things (IoT), and artificial intelligence (AI) lead to the advancement and innovation of digital society. However, privacy and security breaches pose major challenges as cyber-criminals attack the users computer and networks for stealing sensitive data, spy on the infected system, or take control of the system for self-gain [1]. The attackers use malware (malicious software) to gain access to the target system. Malware is a software, code, or program which performs malicious actions. The term malware is used to generalize any form of malicious software and code. It can get different names based on behavior and purpose like virus, Trojan, adware, worm, and spyware. Malware analysis is used to understand the behavior of malware and also helps in the detection process. Currently, the analysis of the malware process is signature-based or behavior-based, but these are proven to be time-consuming as well less effective in identifying unknown malware in real time. This paper aims to propose a novel architecture that combines the concept of machine learning and deep learning which effectively detects zero-day malware.

1.1 Research Background

When the very first computer virus appeared in 1988–89, antivirus software were designed to detect only the known viruses by searching the virus definition databases which is updated time to time; this method is called signature-based detection. But the challenges with this approach are virus variants use different types of obfuscation which hides the viruses signature. Hence, signature-based method are less efficient in terms of detecting the zero-day attack [2]. Signature-based analysis needs domain-level knowledge to reverse engineer the malware using static and dynamic malware analysis techniques. These techniques are used to identify the important features of the malware which helps in signature-based detection. These methods take larger time to reverse engineer the malware; during that time, hackers might take many valuable information. It is also a resource-extensive method.

Many potential researchers have identified that hackers use obfuscation methods to against signature-based detection. To tackle this problem, software are used to manually unpack the file and analyze the APIs calls. But this process is resource-intensive. In [3], author presented a model which automatically extract the APIs call and analyze the binary in four-step. In step 1, unpacking of malware. In step 2, disassembling of binary. In step 3, extraction of APIs call, and in step 4, APIs call mapping and feature analysis. This work was further enhanced in [4] by adding a extra step using machine learning. SVM is used with n-gram feature extraction from both goodware and malware binary with tenfold cross validations. In [5], author proposed a hybrid model which combines support vector machine (SVM) and maximum-relevance minimum redundancy filter (MRMRF) with API calls feature for enhanced

malware detection. With the increase in malware variants due to obfuscation, recently many potential researcher are improving the malware detection methods [6]. This forms the motivation of this research.

2 Related Work

Machine learning algorithms works on feature engineering, selection, and representations. The set of features of different class is used to train the model in order to create a plane of goodware and malwares. This plane helps to classify the malwares and goodwares. Both feature selection and engineering requires domain level knowledge. Various features can be obtained by static and dynamic analysis explained in Sect. 3 of this paper.

The problem with classical machine learning-based malware detection system is that they rely on the feature engineering, learning, and feature representation [7–9] and once an attacker have the knowledge about the features used in model, the malware detector can be easily bypassed [10].

To be accurate, machine learning algorithms requires variety of data. The publicly available data for malware analysis is very less due to privacy and security concerns, and each available data has their own limitations. Many researchers prepare their own datasets and preparing their own dataset by using data science explained in [11] for research is a daunting task. These are the major limitations for developing a machine learning-based malware detection system that can be used in real time.

Nowadays, deep learning models, an improved model of neural networks better performed compared to machine learning models in many of the task in the field of natural language processing, robotics, and others [12]. In training phase, it tries to grab high-level representation of features in hidden layers with the capability to learn from mistakes. These are [7–9, 13–20] are the few research studies which uses the application of deep learning models for malware analysis.

3 Methods for Malware Analysis

3.1 Static Analysis

In static analysis, executables are analyzed without actually executing them. It is the very first and less risky process and does not require any safe environment or sandbox for analyzing them. Static analysis involves the analysis of the internal structure of the program. It involves various steps: (a) Determining the file type of the malware: It helps in identifying the malware's target operating system and architecture. (b) Fingerprinting the malware: By fingerprinting means generating the hash value based

on its file content. It helps in identifying whether this particular malware is identified before by searching in multi-anti-virus databases like VirusTotal. (c) Extracting Strings: Executable strings can be extracted using the string utility tool available in the linux system. Extracted strings can give clues about the program functionality and indicators associated with a suspect binary. (d) Determining file obfuscation: Obfuscation is a method used by the malware authors to hide the inner working of the binary. Packers and cryptors are obfuscation methods used by the malware authors.

3.2 Dynamic Analysis

Dynamic analysis is the way toward extricating data from malware while it is running. Not at all like the restricted view, the static analysis gives of the malware being broke down, powerful examination offers a more top to bottom view into the malware's capacities since it is gathering data while the malware is executing its capacities and orders. To lead dynamic malware analysis, two things are required: malware test environment and dynamic analysis tools.

A malware test environment is a framework where malware is executed with the end goal of examination. It should comprise of a working framework that the malware is composed for and should have most, if not all, of the conditions the malware needs to execute appropriately.

The dynamic analysis tool, otherwise called framework checking apparatuses, is the one observing the malware test environment for any progressions made by the malware to the objective framework. A portion of the progressions that are observed and recorded remember changes for the document framework, adjustments in setup documents, and whatever other important changes that are set off by the malware's execution. The powerful investigation devices likewise screen inbound and outbound organization correspondences and any working framework assets utilized by the malware. With these tools, the investigator can comprehend what the malware is attempting to never really target framework.

A completely executed malware test climate with the fitting powerful investigation instruments is otherwise called a malware sandbox. A malware sandbox is a place where an examiner can run and notice a malware's conduct. A malware sandbox can be a solitary framework or an organization of frameworks planned exclusively to break down malware during runtime.

4 PE File Format

The Windows PE document is the record sort of Windows working frameworks beginning in Windows NT and Windows 95. It is called Portable Executable because

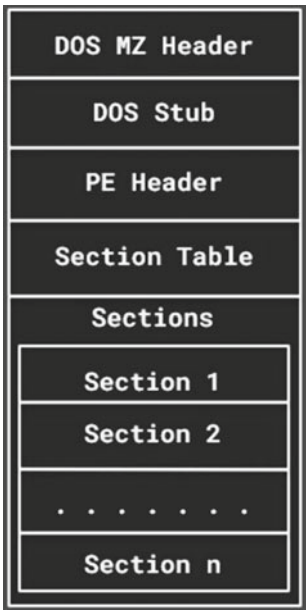

Fig. 1 PE file format

Microsoft's vision was to utilize a similar document design in future kinds of Windows, making the PE document basic to all Windows stages paying little mind to what central processing unit (CPU) they support.

The Windows PE document design is gotten from the Common Object File Format (COFF) that was utilized in Virtual Address extension (VAX) frameworks running the Virtual Memory System (VMS) working framework created by Digital Equipment Company (DEC), which was procured by Compaq in 1998 and converged with HP in 2002. The majority of the first Windows NT improvement group came from DEC (Fig. 1). The PE File design comprise of the accompanying:

- DOS MZ Header
- DOS Stub
- PE Header
- Section Table
- Sections

5 Dataset Description

The dataset is obtained from the publicly available dataset from IEEE Dataport. It contains information from around 48K malware and goodware. The dataset is gotten by exploiting the openly accessible reports from malware administration. It is a free

online assistance that does a static and dynamic examination on submitted records utilizing the Cuckoo sandbox, which are then available in an HTML report.

To ensure the credibility of the dataset, we turned to two other online archives: National Software Reference Library (NSRL) and VirusShare.com, these give metadata (for example MD5 hash) in regards to known goodware and malware samples, separately. As NSRL contains an assortment of advanced marks of known, traceable software applications, if an example is available in this assortment, we are more sure it is without a doubt goodware. Then again, VirusShare.com is a vault of malware tests, henceforth an example present in this storehouse gives us higher certainty it is malware.

When the information validness is affirmed, we began the extraction process, where online information is saved locally or in a central database for additional examination. This method is called scraping, and it is done by using Python scraping library. Concerning the NSRL repository, data was given in textual format, which drove us to utilize Pandas, a Python data analysis library, to extract and dissect the information. The data extracted from the PE samples are visualized in three different datasets:

1. PE_Import dataset contains the top 1000 imported function information extracted from the import section of the PE sample.
   It has 1002 columns in which 1000 columns are the features, one column is for the hash, and one column for the label.
2. PE_Section_Header contains the information of the section header of .text, .data, .code, and code section of the PE sample.
   It has 6 columns in which 4 columns are the features, one column is for the hash, and one column for the label.
3. PE_Raw_Image contains the raw PE byte stream re-scaled to 32×32 grayscale images of PE sample.
   It has 1026 columns in which 1024 columns are the pixel value, one column is for the hash, and one column for the label.

6 Model Implementation

Proposed model uses the combination of both machine and deep learning as shown in the Fig. 2 based on static analysis. It uses deep learning for the feature extraction process and classical machine learning model for the classification process. For the PE_Section_Header, we used fully connected artificial neural network, for the PE_Import, we also used fully connected neural network, and for Raw_PE_Image, we have used convolution neural network.

For the fully connected neural network, we have used adam optimizer and binary cross entropy as loss function and ReLU as the activation function.

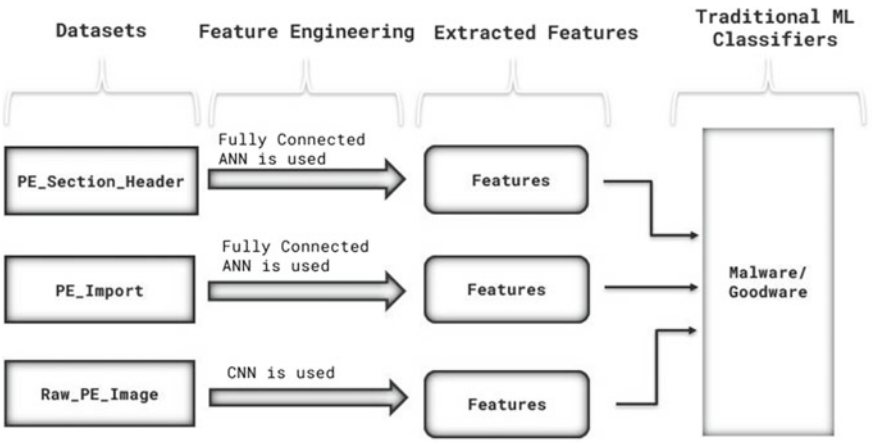

Fig. 2 Proposed model

6.1 Performance Evaluation

We have performed various experiments based on the number of features of the datasets. We evaluate the optimal number of features for our model for that we initially used 50 features out of 1000 from the PE Import, 50 features out of 1024 from the Raw PE Image, and 4 features from the PE Section Header. For the selection process, we tuned the second-last layer of the neural network as per our requirements and later stored these intermediate values in a new .csv file using which we create a more informative and efficient datasets. Later, these new datasets are given to machine learning models for the classification process. The experiment results are in Table 1.

The best result we got when we selected 100 features from the Import dataset and 100 from Image dataset and 4 from the Section dataset. We have also performed our experiment using various machine learning algorithm and also deep learning model and compared our model which can be seen in Table 2.

Table 1 Features analysis result

S. No.	Number of features from import	Number of features from image	Number of features from section	Accuracy
1	50	50	4	97.22
2	50	100	4	97.86
3	100	50	4	97.86
4	100	100	4	98.91

Table 2 Experiment result

Model	Classifier	Accuracy
Voting method based on Machine Learning	Ensemble Voting	97.66
Voting method based on Deep Learning	–	95.99
[21]	CNN 2 layer+ LSTM	98.8
[22]	RNN	96.01
[23]	–	98.4
[24]	–	97.4
Proposed model	KNN	96.95
Proposed model	Random Forest	98.91
Proposed model	SVM	98.64

7 Conclusion

In this work, we analyzed how ML and DL techniques fit into the scope of malware detection and how could the chosen dataset influence the results of the classifier. We analyzed, trained, and validated multiple models to better understand how laboratory conditions vary from real-world conditions. We compared our model with others models which is based on machine learning, deep learning, and combinations on both, but doing so our model provided us with results as high as 98.91%.

We have also concluded that the model combined with ML and DL both gives better and promising results. Having a solid knowledge of the effects of temporal consistency in the task of malware detection, we improved our base model for better results. This was done by using the DL feature extraction approach to provide the ability to extract information regarding malware classes and by adding more features to the model.

The task we set ourselves to achieve was not without its difficulties, but all in all, we believe our work shows that the path to malware detection via machine learning and deep learning is feasible, not only theoretically, as related work as shown, but also with practical implications.

8 Future Work

In the above work, we used static analysis for the feature analysis and selection; in future, we want to incorporate the dynamic analysis for the feature engineering doing

so give us a new prospect toward the datasets and also obtain new feature which can increase the accuracy of the model.

References

1. G. McGraw, G. Morrisett, Attacking malicious code: a report to the Infosec Research Council. IEEE Softw. **17**(5), 33–41 (2000)
2. B. Li, K. Roundy, C. Gates, Y. Vorobeychik, Large-scale identification of malicious singleton files, in *Proceedings of 7th ACM conference on data and application security and privacy, New York, NY, USA: ACM*, Mar 2017, pp. 227–238
3. M. Alazab, S. Venkataraman, P. Watters, Towards understanding malware behaviour by the extraction of API calls, in *Proceedings of 2nd Cybercrime Trustworthy Computing Workshop*, July 2010, pp. 52-59
4. M. Tang, M. Alazab, Y. Luo, Big data for cybersecurity: vulnerability disclosure trends and dependencies. IEEE Trans. Big Data (to be published)
5. S. Huda, J. Abawajy, M. Alazab, M. Abdollalihian, R. Islam, J. Yearwood, Hybrids of support vector machine wrapper and filter based framework for malware detection. Fut. Gener. Comput. Syst. **55**, 376–390 (2016)
6. E. Raff, J. Sylvester, C. Nicholas, Learning the PE header, malware detection with minimal domain knowledge, in *Proceedings of 10th ACM Workshop Artificial intelligence and security* (ACM, New York, Nov 2017), pp. 121–132
7. E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, C. Nicholas, Malware detection by eating a whole exe (2017). [Online]. Available:https://arxiv.org/abs/1710.09435
8. M. Krcál, O. Švec, M.Bálek, O. Jašek, Deep convolutional malware classifiers can Learn from raw executables and labels only (2018).[Online]. Available: https://openreview.net/forum?id=HkHrmM1PM
9. M. Rhode, P. Burnap, K. Jones, Early-stage malware prediction using recurrent neural networks. Comput. Secur. **77**, 578–594 (2018). (Aug.)
10. H.S. Anderson, A. Kharkar, B. Filar, P. Roth, *Evading Machine Learning malware Detection* (Black Hat, New York, 2017)
11. R. Verma, Security analytics: adapting data science for security challenges, in *Proceedings of 4th ACM International Workshop on security and privacy analytics* (ACM, New York, Mar 2018), pp. 40–41
12. Y. LeCun, Y. Bengio, G. Hinton, Deep learning. Nature **521**(7553), 436–444 (2015)
13. A.F. Agarap, F.J.H. Pepito, Towards building an intelligent anti-malware system: a deep learning approach using support vector machine (SVM) for malware classification.v [Online] (2017). Available:https://arxiv.org/abs/1801.00318
14. E. Rezende, G. Ruppert, T. Carvalho, A. Theophilo, F. Ramos, P. de Geus, Malicious software classification using VGG16 deep neural network's bottleneck features, in *Information Technology-New Generations* (Springer, Cham, 2018), pp. 51–59
15. J. Saxe, K. Berlin, Deep neural network based malware detection using two dimensional binary program features, in *Proceedings of International Conference on Malicious and Unwanted Software (Malware)*, Oct 2015, pp. 11–20
16. S. Tobiyama, Y. Yamaguchi, H. Shimada, T. Ikuse, T. Yagi, Malware detection with deep neural network using process behavior, in *Proceedings of IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2, June 2016, pp. 577–582
17. W. Huang, J. W. Stokes, Mtnet: a multi-task neural network for dynamic malware classification, in *Proceedings of International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (Springer, Cham, July 2016), pp. 399–418
18. R. Pascanu, J. W. Stokes, H. Sanossian, M. Marinescu, A. Thomas, Malware classification with recurrent networks, in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Apr 2015, pp. 1916–1920

19. T. Shibahara, T. Yagi, M. Akiyama, D. Chiba, T. Yada, Efficient dynamic malware analysis based on network behavior using deep learning,' in *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, Dec 2016, pp. 1–7
20. B. Kolosnjaji, A. Zarras, G. Webster, C. Eckert, Deep learning for classification of malware system call sequences, in *Proceedings of Australasian Joint Conference on Artificial Intelligence* (Springer, Cham, Dec 2016), pp. 137–149
21. R. Vinayakumar, M. Alazab, K.P. Soman, P. Poornachandran, S. Venkatraman, Robust intelligent malware detection using deep learning. IEEE Access **7**, 46717–46738 (2019). https://doi.org/10.1109/ACCESS.2019.2906934
22. M. Rhode, P. Burnap, K. Jones, Early stage malware prediction using recurrent neural networks. Available: arXiv:1708.03513
23. S.Z.M. Shaid, M.A. Maarof, Malware behaviour visualization. J. Teknol. **70**(5), 25–33 (2014). (Sep.)
24. H. Zhou, Malware detection with neural network using combined features, *Presented at the Australasian Joint Conference on Artificial Intelligence, Beijing, China, Aug. 2018)*