SN Computer Science (2021) 2:275
https://doi.org/10.1007/s42979-021-00672-y

ORIGINAL RESEARCH

Study of a Hybrid Approach Towards Malware Detection in Executable Files

Akshara P[1] · Bhawana Rudra[1]

Received: 11 November 2020 / Accepted: 28 April 2021

Abstract

With the ever-increasing number of Internet users in this digital age, exposure to malicious attacks is increasing. Every day, large volumes of malicious content are generated to exploit 0-day vulnerabilities. There is every possibility of downloading malicious files unintentionally, which could corrupt the system and user data. With the advancements in technology and growing dependence on digital data, malicious software detection has become a crucial task. The existing approaches need modifications to support and detect the latest attacks. Recently, artificial intelligence-based malicious file detection methods have been proposed. In the past, most of the works analyzed the executable file features and visual features from their corresponding images independently. Additionally, image-based analysis has been exploited for categorical classification, i.e., finding the family once it is known to be malware. We propose a CNN-based model that extracts visual features from malware images, which outperforms existing approaches on a benchmark dataset like MalImg. We study the effect of using a hybrid feature set containing these visual features integrated with statically obtained opcode frequencies for the detection of malware. Our experiments on standard datasets demonstrate that there is no significant performance improvement using this hybrid approach.

Keywords Cyber security · Malware detection · Hybrid feature extraction

Introduction

Malware are the computer programs with malicious characteristics intending to harm the operating system, steal personal information, and damage the user network. Detection and mitigation of malware are an evolving problem in the field of cyber security. Traditional as well as commercial antivirus softwares (AV) available in the market usually rely on a signature-matching method, where a local signature is required to be stored in the database for matching patterns from well-known malware. A signature is a short sequence of bytes (hash) that can be used to identify malware uniquely with small error rates. However, techniques like code obfuscation, encryption, and morphing can significantly change the malware signature. Due to this, the signature-matching method cannot provide security against 0-day attacks as malware generation tool kits like Zeus[1] can generate many variants of the same malware using obfuscation techniques [1].

Malware detection can mainly be performed in two ways using static analysis or dynamic analysis. Certain features are extracted from the portable executable (PE) files without actually executing the code in static analysis. Static analysis is safe to adopt as it does not harm the user's system. It is less resilient towards the encrypted and compressed malware executables. In dynamic analysis, features are derived while running the executable in a controlled environment called sandbox. Dynamic analysis is more suited for real-time detection and reveals the true nature of the code. However, the execution path taken is the same in every run and the analysis time is increased [2]. Researchers and anti-malware communities have reported that using machine learning, and deep learning-based methods, malware detection systems

This article is part of the topical collection "Cyber Security and Privacy in Communication Networks" guest edited by Rajiv Misra, R K Shyamsunder, Alexiei Dingli, Natalie Denk, Omer Rana, Alexander Pfeiffer, Ashok Patel and Nishtha Kesswani.

✉ Akshara P
akshblr555@gmail.com

[1] National Institute of Technology Karnataka, Surathkal, India

[1] https://www.sophos.com/en-us/medialibrary/PDFs/technical-papers/Sophos-what-is-zeus-tp.pdf.

Published online: 15 May 2021

are more robust to code modifications. An essential component of machine learning-based malware detection is feature extraction from PE headers, and subsequent dimensionality reduction of the obtained feature space, which is a challenging problem in both practical and theoretical machine learning [3].

The proposed method is based on constructing a hybrid feature space using enhanced feature extraction to detect malicious executable files on Windows. The first set of features are based on the opcode frequency, obtained from static analysis of PE files after disassembling the executable files. The second set is obtained from grayscale images by converting the binary files using visual analysis and Convolution Neural Network (CNN) along with a transfer learning-based approach. The CNN model will be first trained for detection of malware on the MalImg dataset [4] to learn the structural features of malware based on its local image patterns. It performs better than most pre-trained deep learning models, which have been experimented in the past on the MalImg dataset. CNN-based feature representation transfer is then utilized to extract 128 features from the primary dataset. This extended feature set yields similar performance using only the opcode frequency features.

The main contributions of the paper are:

(i) A generic CNN architecture is proposed for image-based malware detection, which outperforms with the previous studied pre-trained models.

(ii) A hybrid feature space containing opcode frequency from static PE files and visual descriptors from the corresponding grayscale images for the detection of malware is explored and validated.

The rest of the paper is organized as follows. The second section discusses about the "Related work" followed by third section that represents the "Methodology". The forth section gives the "Experimental results" and "Conclusions and future work" are drawn in last section.

Related Work

Several security researchers have applied domain-level knowledge on PE files for static malware detection. Here, the crucial step is the feature selection, which is mostly hand-picked. Schultz et al. [5] introduced static feature analysis using a data mining method to detect malware with three different types of static features, i.e., PE head, string sequence, and byte sequence. Shabtai et al. [6] provided a taxonomy for malware detection using machine learning algorithms by reporting feature types, feature selection techniques, and ensemble algorithms. Firdausi et al. [7] proposed an automatic behavior-based malware detection approach using Anubis, a free online dynamic analysis service to monitor the collected samples. They used Naive Bayes, decision trees, and k-nearest neighbors for classification.

Ahmadi et al. [8] proposed a novel paradigm to group malware variants into different families. The authors gave importance to feature extraction and selection methods. The features were extracted from content and the sample structure, so the proposed method will work on packed and obfuscated samples. The features were grouped based on malware behavior characteristics, and fusion is performed according to a per-class weighting paradigm. Rathore et al. [1] found opcode frequency to be a discriminatory feature of PE files; they used Random Forest as a baseline and other deep learning models for the comparison. Igor Santos et al. [9] proposed a hybrid malware detection system that detects malware by combining features extracted from opcode frequency and executing the dynamic features that resulted in better performance compared with each alone.

Another interesting approach involves viewing this task as an image recognition one by converting the binaries to grayscale images and employing several vision-based techniques. One of the first works that visualized malware classification through image processing was by Nataraj et al. [4], where high-level gist descriptors were used and the classification was done by kNNs. Sravani et al. [10] used a Deep Learning (DL) approach that obtained slightly better results without using gist descriptors. They compared feature-based classical methods like kNN with DL ones to demonstrate the superior performance [11]. Transfer learning technique was introduced using pre-trained ResNet, VGGNet, and Inception models for malware classification. This is because images tend to smooth out minor within-family differences, while significant (inter-family) differences were clearly observed [3, 11]. A generic CNN architecture for malware classification was proposed [12, 13] and experimented on popular benchmark datasets to obtain good accuracy. These works mainly aim at family classification rather than detection.

The opcode sequences from PE files obtained dynamically were widely studied. The major issue with this method was the high execution time. An RNN-based approach using API calls as features during run-time [14]. An LSTM-based approach was presented in [15], where first an opcode vocabulary is built using opcode sequences from the assembly code. This was then used to generate CBOW embeddings [16], which were sent through a two-stage LSTM model. Jipei et al. [17] proposed MalNet, a stacked ensemble of CNN (image-based) and LSTM (opcode sequence based) that demonstrated enhanced performance in both malware detection and classification. They reduced the prediction time by adopting several parallelization techniques. Some concerns with these techniques include accounting for huge opcode sequences of

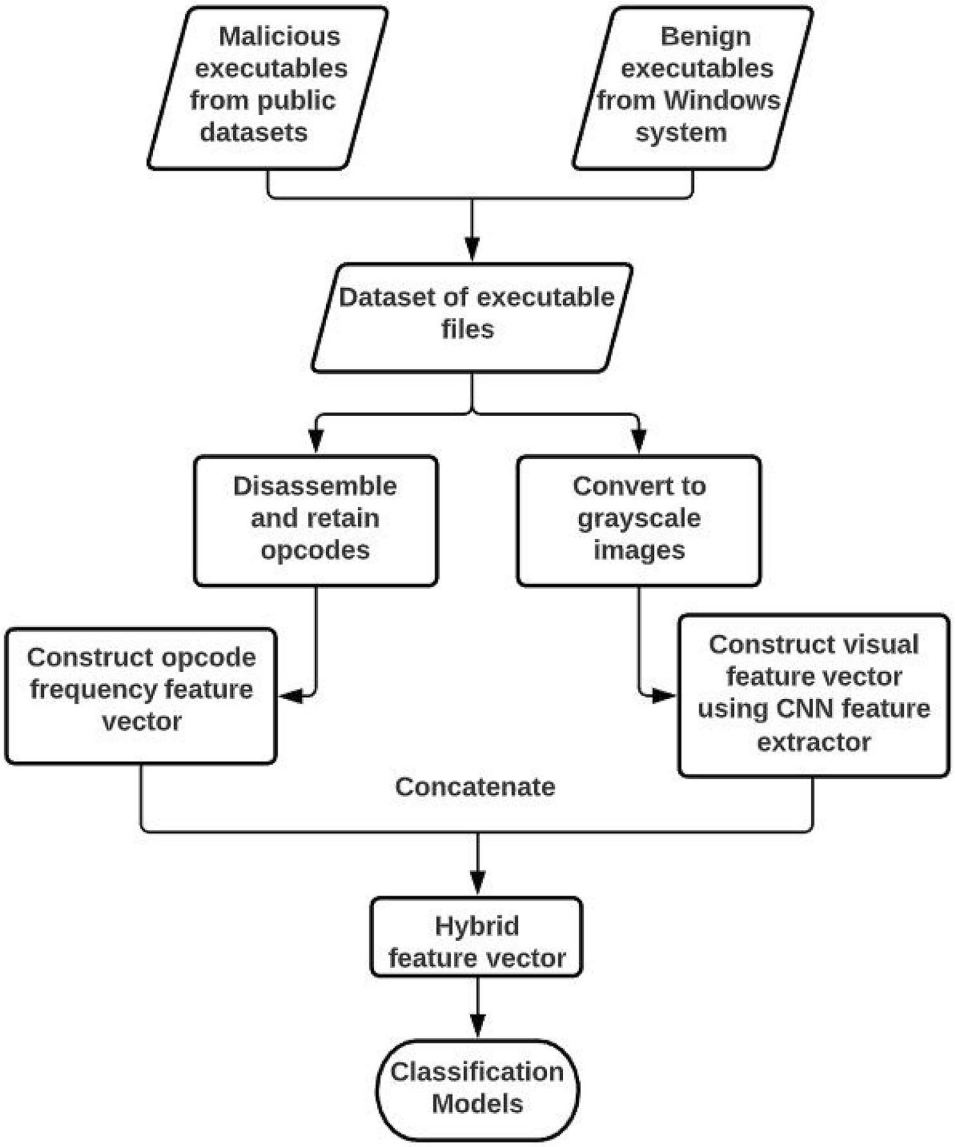

Fig. 1 Classification pipeline

variable length generated, resulting in loss of information and the subsequent delay in procuring them and training.

Extending these works, we have made a first of its kind attempt to detect malware by integrating features from PE files as well as the binary images.

Methodology

The malware analysis and detection is formulated as a binary classification task where malware and benign are two different classes. In this work, Malware Detection PE-Based dataset [18] is used as the primary dataset for the study. It contains 8970 malware and 1000 benign binary files. Malware files are divided into five types: Locker (300), Mediyes (1450), Winwebsec (4400), Zbot (2100), and Zeroaccess (690). All the malware files have been collected from VirusShare[2] and Malicia Project [19]. Benign executable files are taken from installed folders of applications of legitimate software from different categories. All files have been verified using VirusTotal 5,[3] an anti-virus aggregator. Additionally, the MalImg dataset has been used for image-based visual analysis and to train CNN model, and then used as a feature extractor on the primary dataset. We considered total of 9339 grayscale images from 25 malware families.

The classification pipeline is a multi-step process, as shown in Fig. 1. The overview of the algorithm for the proposed pipeline is explained in Algorithm 1 and in detail in the following sections.

[2] https://virusshare.com/.

[3] https://www.virustotal.com/.

Algorithm 1: Malware detection algorithm

Precondition : Trained CNN model. Opcode master list.
Result: class label (malicious/benign)
dataset ← *all .exe files*;
threshold ← 0.01;
for *every file in dataset* **do**
  disassemble *file* to get assembly code;
  scan assembly code to generate opcode frequency vector *op_vec* ;
  ▷ `using master list`
  if *var*(*opcode*) *in op_vec* < *threshold* **then**
    drop opcode from *op_vec*;
  end
  convert *file* to gray scale image;
  use trained CNN to extract visual feature vector *vis_vec*;
  concatenate *op_vec* and *vis_vec* to get hybrid vector *hyb_vec*;
  send *hyb_vec* to classification models;
end

Data Preprocessing

The first step involves disassembling the executable files (.exe) to assembly code (.asm). This is performed using the object dump utility, which is a part of the GNU Binutils package.[4] For the conversion of binary file to an image, the bytes representing the files are scanned sequentially, and their corresponding unsigned integer is treated as bytes of a grayscale image. Images are created having a predefined width of 256 pixels and variable height, depending on the size of the file—similar to the approach adopted in [11]. After carrying out the above steps and removing corrupt and encrypted files, the final dataset under study contains 8938 malicious and 967 benign files. Since the MalImg dataset directly provides images, no preprocessing steps are required. All images are finally re-scaled to 256 × 256 size before feeding to the CNN model.

Feature Extraction

Opcode Frequency Feature Space

The opcode feature vector space is generated from the disassembled files using static analysis of executable files. Similar to the approach adopted in [1], an exhaustive opcode list containing 1808 opcodes is prepared. Then, the assembly files are scanned to obtain the frequencies of opcodes occurring in all the files. Some opcodes did not appear in any file or are very similar and hence grouped or removed accordingly. Finally, a 1431-dimensional feature space is constructed for the analysis. To overcome class imbalance issue, the variance threshold method is considered and fixed the threshold at 0.01. If the variance of any opcode occurrence is lower than the threshold, then it is considered to be approximately constant and would not improve the performance of the model significantly and hence dropped.

CNN Extracted Features

Literature shows that the deep pre-trained networks such as ResNet and VGGNet yielded poorer accuracy compared to classical ML models like kNNs and generic CNN architectures, so a custom CNN architecture as shown in Fig. 2 is considered for our approach.

It has 4 convolution layers, followed by a pooling layer, respectively, and 2 fully connected dense layers. Using pooling layers, the model is robust to the small changes in the images too. This model is trained on the MalImg dataset for 10 epochs using a 70-10-20 train-validation-test split, using dropout and early stopping to prevent over-fitting. Due to

[4] https://www.gnu.org/software/binutils/.

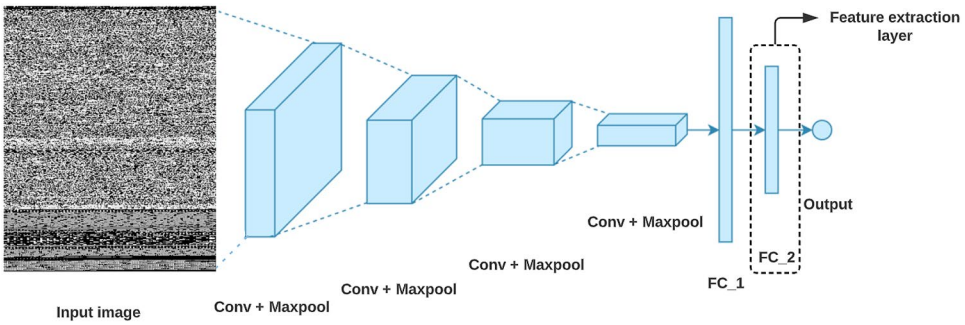

Fig. 2 Proposed CNN architecture: input dimension is 256 × 256 × 1. 4 Conv+Maxpool operations using 8, 16, 32, 64 conv filters, respectively. Finally flattened to get 512 and 128 dimensions in FC_1 and FC_2

class imbalance, data augmentations are tried, but led to a decrease in accuracy. This is because most of the programs are executed in top-to-bottom order without many jumps, and horizontal and vertical flipping would result in flipping the order—thus changing the program itself. This trained model is then used on the actual dataset to extract the second last layer's features, i.e., *FC_2*.

Studies show that generic descriptors extracted from the layers of convolutional neural networks are very powerful [20–23]. Also, the high-level layer features are more specific and discriminant, particularly for target tasks close to the source task.

Compared with lower level layers, high-level layers provide better results when used out-of-the-box to feed classifiers for similar tasks. These findings are statistically quantified using the Kolmogorov–Smirnov statistic (DKS), which measures the distance between two empirical distribution functions (EDF) P and Q [22]. In our study, the source task on MalImg and destination task on the primary dataset is the same, and hence, the features of the second last layer of the CNN model, which is 128-dimensional, are extracted from the primary dataset. This would encode a large amount of visual knowledge from the images, specific to malware detection. Finally, the 1559 (1431 + 128)-dimensional combined feature set is obtained by concatenating the above 2 prepared sets. The 9905 × 1559 dataset is then sent to the classification models.

Classification Models

We used Random Forest (RF) and XGBoost (XGB) for classification. Random Forest is a collection of tree predictors, such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. It consists of a large number of individual decision trees that operate as an ensemble. Many relatively uncorrelated trees individually vote, and the final prediction is based on the majority. This ensures that prediction errors are minimized.

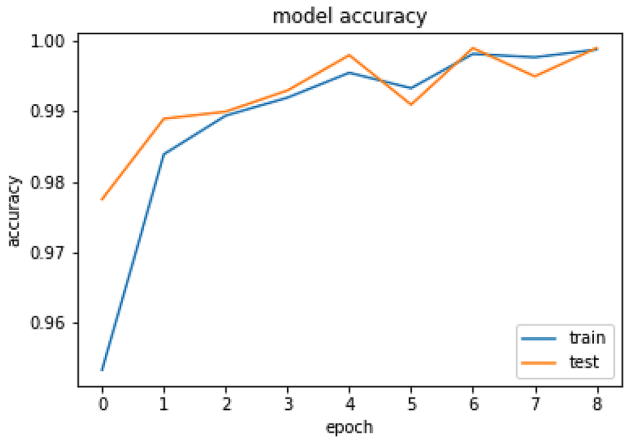

Fig. 3 Model training accuracy

Table 1 Performance comparison on MalImg

Model	Work	Accuracy (%)
ResNet34	[11]	98.39
Generic CNN	**Ours**	**99.6**

XGBoost has been widely used in many fields to achieve state-of-the-art results. It is an efficient implementation of the Gradient Boosting Machine. The main idea of boosting is to combine a series of weak classifiers with low accuracy, sequentially, to build a robust classifier with better classification performance. In gradient boosting, errors are minimized by a gradient descent algorithm. By blending software and hardware capabilities, it provides enhanced accuracy in a short amount of time.

Table 2 Observed malware detection performance

Feature set	Model	Accuracy (%)	AUC score
Opcode frequency	RF	99.4	0.987
Opcode frequency	**XGB**	**99.7**	**0.99**
CNN extracted	RF	98.9	0.969
CNN extracted	XGB	98.89	0.962
Opcode frequency + CNN	RF	99.4	0.981
Opcode frequency + CNN	**XGB**	**99.7**	**0.99**

Experiments and Results

First, we evaluate the performance of our generic CNN model with some of the previous works for malware detection on MalImg dataset in Table 1. Figure 3 shows the model's training phase accuracy. As early stopping mechanism is used, the training converges after 8 epochs.

Results indicate that our generic CNN outperforms the very deep pre-trained networks like ResNet. Furthermore, when this model was used on the primary dataset for direct prediction using a pure transfer learning approach, it yielded an accuracy of 99.3%, thus representing its generalizability.

Next, we carried out 3 different experiments to understand the relative performance and improvements by adopting combinations of the prepared feature sets.

1. Using the opcode frequency feature set: this considers only the statically obtained variance threshold opcode features, which is in line with the approach described in [1].
2. Using the CNN extracted features: a total of 128 features were extracted from the primary dataset's images using our trained CNN model and are passed to the classification models.
3. Combination of both feature sets: in this, both feature sets are concatenated and sent to the classification models.

Table 2 summarizes the results of these experiments. As observed from the results, the hybrid feature set (which encodes a large amount of static and visual knowledge from executables and their images) exhibits similar performance as of the opcode frequency feature.

We further evaluated our trained model on a test set by considering the sample of another benchmark dataset Big 2015 [24], which has more recent malware families which were not included in our primary dataset. We obtained an accuracy of **96**%, again verifying with the obtained observation. From these results, we can say that visual features do not seem to add significant influence towards the detection of malware samples. When both the features are used together, the result was not improved, but when considered the images, then there is a large enhance in accuracy.

Conclusion and Future Work

The alarming growth of malware attacks in recent years is a major concern as it poses a serious security threat to computer systems and data. Devising robust malware detection techniques is the need of our current society to overcome the threats. Classical signature-matching mechanisms used by most AVs tend to become unreliable when they find new malware. Towards this endeavor, we utilized a hybrid feature set obtained from the discriminatory feature of PE files, opcode frequency, and high-level image descriptors from a trained CNN model for the detection of malicious code in the Windows executable files. However, we did not observe any significant improvement over the opcode frequency feature set, probably due to an increase in dimensionality, but we observed a drastic change in accuracy when the images were used for the detection of malware. In future, we will study the effect of integrating other features such as opcode sequence calls information in more effective manner. We also extend this approach to classify malware into their corresponding families once detected, and further processes for better accuracy. Our study can guide future researchers who plan to use different data sources for malware analysis [25].

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Rathore H, Agarwal S, Sahay S, Sewak M. Malware detection using machine learning and deep learning. In: 6th international conference, BDA 2018, Warangal, India, December 18–21; 2018. https://doi.org/10.1007/978-3-030-04780-1_28.
2. Cavallaro L, Saxena P, Sekar R. On the limits of information flow techniques for malware analysis and containment. In: Zamboni D (ed) Detection of intrusions and malware, and vulnerability assessment. DIMVA 2008. Lecture Notes in Computer Science, vol 5137. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-70542-0_8.
3. Chen L. Deep transfer learning for static malware classification. 2018. arxiv: 1812.07606.pdf. Retrieved 10 Dec 2020.
4. Nataraj L, Karthikeyan S, Jacob G, Manjunath B. Malware images: visualization and automatic classification. 2011. https://doi.org/10.1145/2016904.2016908.

5. Schultz M, Eskin E, Zadok F, Stolfo S. Data mining methods for detection of new malicious executables. In: Proceedings of the IEEE computer society symposium on research in security and privacy; 2001, pp. 38–49. https://doi.org/10.1109/SECPRI.2001.924286.
6. Shabtai A, Moskovitch R, Elovici Y, Glezer C. Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey. Inf Secur Tech Rep. 2009;14(1):16–29. https://doi.org/10.1016/j.istr.2009.03.003.
7. Firdausi I, lim C, Erwin A, Nugroho AS. Analysis of machine learning techniques used in behavior-based malware detection. In: Second international conference on advances in computing, control, and telecommunication technologies, Jakarta; 2010, pp. 201–203. https://doi.org/10.1109/ACT.2010.33.
8. Ahmadi M, Ulyanov D, Semenov S, Trofimov M, Giacinto G. Novel feature extraction, selection and fusion for effective malware family classification. 2016. https://doi.org/10.1145/2857705.2857713. Retrieved 10 Dec 2020.
9. Santos I, Devesa J, Brezo F, Nieves J, Bringas POPEM. A static-dynamic approach for machine-learning-based malware detection. 2013. https://doi.org/10.1007/978-3-642-33018-6_28.
10. Sravani Y, Vikash S, Troia DF, Stamp M. Deep learning versus gist descriptors for image-based malware classification. 2018. https://doi.org/10.5220/0006685805530561. Retrieved 10 Dec 2020.
11. Bhodia N, Prajapati P, Troia DF, Stamp M. Transfer learning for image-based malware classification. 2019;2020. https://doi.org/10.5220/0007701407190726. Retrieved 10 Dec.
12. Kalash M, Rochan M, Mohammed N, Bruce NDB, Wang Y, Iqbal F. Malware classification with deep convolutional neural networks. In: 9th IFIP international conference on new technologies, mobility and security (NTMS), Paris; 2018, pp. 1–5. https://doi.org/10.1109/NTMS.2018.8328749.
13. Choi S, Jang S, Kim Y, Kim J. Malware detection using malware image and deep learning. In: International conference on information and communication technology convergence (ICTC), Jeju; 2017, pp. 1193–1195. https://doi.org/10.1109/ICTC.2017.8190895.
14. Pascanu R, Stokes JW, Sanossian H, Marinescu M, Thomas A. Malware classification with recurrent networks. In: IEEE international conference on acoustics, speech and signal processing (ICASSP), South Brisbane, QLD, Australia; 2015, pp. 1916–1920. https://doi.org/10.1109/ICASSP.2015.7178304.
15. Renjie L. Malware detection with LSTM using opcode language. 2019. arxiv: 1906.04593.pdf. Retrieved 10 Dec 2020.
16. Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. 2013. arxiv: 1301.3781.pdf. Retrieved 10 Dec 2020.
17. Yan J, Qi Y, Rao Q. Detecting malware with an ensemble method based on deep neural network. Secur Commun Netw. 2018. https://doi.org/10.1155/2018/7247095.
18. Tuan AP, Phuong ATH, Thanh NV, Van TN. Malware detection PE-based analysis using deep learning algorithm dataset. 2018. https://doi.org/10.6084/m9.figshare.6635642.v1. Retrieved 10 Dec 2020.
19. Nappa A, Rafique MZ, Caballero J. The MALICIA dataset: identification and analysis of drive-by download operations. Int J Inf Secur. 2015;14:15–33. https://doi.org/10.1007/s10207-014-0248-7.
20. Razavian AS, Azizpour H, Sullivan J, Carlsson S. CNN features off-the-shelf: an astounding baseline for recognition. In: IEEE conference on computer vision and pattern recognition workshops, Columbus, OH; 2014, pp. 512–519. https://doi.org/10.1109/CVPRW.2014.131
21. Azizpour H, Razavian AS, Sullivan J, Maki A, Carlsson S. Factors of transferability for a generic ConvNet representation. IEEE Trans Pattern Anal Mach Intell. 2015. https://doi.org/10.1109/TPAMI.2015.2500224.
22. Garcia-Gasulla D, Parés F, Vilalta A, Moreno J, Ayguadé E, Labarta J, Cortés U, Suzumura T. On the behavior of convolutional nets for feature extraction. J Artif Intell Res. 2017. https://doi.org/10.1613/jair.5756.
23. Yosinski J, Clune J, Bengio Y, Lipson H. How transferable are features in deep neural networks? In: Proceedings of the 27th international conference on neural information processing systems. 2014. arxiv: 1411.1792.pdf.
24. Ronen R, Radu M, Feuerstein C, Yom-Tov E, Ahmadi M. Microsoft malware classification challenge. 2018. arxiv: 1802.10135.pdf. Retrieved 10 Dec 2020.
25. Pandey AK, Tripathi A, Kapil G, Singh V, Khan W, Agrawal A, Kumar R, Khan R. Trends in malware attacks: identification and mitigation strategies. 2020. https://doi.org/10.4018/978-1-7998-1558-7.ch004.