IEICE TRANS. INF. & SYST., VOL.E103–D, NO.4 APRIL 2020

LETTER

Mal2d: 2d Based Deep Learning Model for Malware Detection Using Black and White Binary Image

Minkyoung CHO[†a)], ***Member***, **Jik-Soo KIM**[†b)], **Jongho SHIN**[†c)], ***and*** **Incheol SHIN**[††d)], ***Nonmembers***

SUMMARY We propose an effective 2d image based end-to-end deep learning model for malware detection by introducing a black & white embedding to reserve bit information and adapting the convolution architecture. Experimental results show that our proposed scheme can achieve superior performance in both of training and testing data sets compared to well-known image recognition deep learning models (VGG and ResNet).

key words: *malware detection, deep learning, binary image*

1. Introduction

Conventional antivirus solutions detect a malware by extracting the signature from a binary file, and comparing it with the malware database [1]. However, over 100 millions of new malwares are published every year [2], [3] so that the overall size of signature database also needs to increase significantly and rapidly, which can be a potential performance bottleneck in malware detection mechanisms. In addition, the built signatures can be easily bypassed by changing only a small part of binary signature [4].

Alternatively, researchers have investigated exploiting deep learning mechanisms in malware detection to overcome these weakness of signature-based approaches. Deep learning does not require any fixed features or signatures *a priori*, and it can automatically learn features from data. Since the extracted features are coming from common parts of *various* malware samples rather than only one (as in the existing signature-based malware detection approaches), it becomes much more difficult to bypass these features. Also, the inference time is constant (actually linear to the input dimension), so that we can effectively check if a given file is malware, independent of the number of malware samples retrieved.

There have been several studies to detect malwares based on 2d image similarities. Some researchers observed that binary images of malwares are closely *correlated*, i.e., similar images are belonging to the same type of malwares, while distinct images are based on different malware types. Nowadays, the deep learning in image recognition shows an excellent performance, and in this work, we effectively extend the previous image based malware detection approach by employing several optimization techniques for binary image processing in deep learning models.

In this paper, we design and implement an end-to-end 2d-based deep learning model for malware detection ("Mal2d"). Our algorithm does not require *any* preprocessing for extracting features or interpreting a portable execution header (PE header) of files. Existing approaches of 2d image similarity based malware detection deep learning models mainly treat binary files as grayscale images (i.e., converting a byte to a gray scale $0 \sim 255$), which can potentially result in information loss of bits in a byte. To address this problem, we introduce a black & white embedding to reserve bit information and adapt the convolution architecture. Given an input binary file, we simply transform it into a black and white image where each bit considers a pixel with 0 or 1, which can be subsequently deployed to our customized deep learning model. To validate the effectiveness of Mal2d, we have performed a comparative analysis of our proposed malware detection deep learning model with two well-known models for image recognition (e.g. VGG [5], ResNet [6]) with various data sets and performance metrics. Evaluation results show that our proposed model achieves better performance in both of training and testing data sets.

2. Related Work

Nataraj et al. [7] proposed a malware detection based on image similarity. They treated a binary file as an 8-bit gray scale image where each byte considered a pixel with value $0 \sim 255$. Using Gabor filters, they extracted 320 features from each binary file and then classified them by k-nearest neighborhood algorithm. They observed that the malwares in the same type had similar features on their grayscaled images. Hence, it is a smooth extension to apply image based deep learning models for malware detection.

Rezende et al. [8] introduced a malware detection method using pre-trained VGG16, one of well-known deep learning models for object detection. They first extracted features from the bottleneck layer of VGG16, trained by ImageNet dataset, and then trained a SVM classifer with the extracted features to detect malwares.

Some approaches directly train a deep learing model with malware data, instead of using pre-trained models.

Manuscript received August 7, 2019.
Manuscript revised October 16, 2019.
Manuscript publicized December 25, 2019.

[†]The authors are with Dept. of Computer Eng., Myongji Univ., Korea.
[††]The author is with Dept. of Computer Eng., Mokpo Nat'l Univ., Korea.
a) E-mail: mkcho@mju.ac.kr
b) E-mail: jiksoo@mju.ac.kr
c) E-mail: jh3314130@mju.ac.kr
d) E-mail: ishin@mokpo.ac.kr (Corresponding author)

DOI: 10.1587/transinf.2019EDL8146

Khan et al. [9] used two famous deep learning models, ResNet [6] and GoogLeNet [10], for malware detection without modification. In their experiments, ResNet has better performance than GoogLeNet.

In this paper we focus on 2d image based deep learning models for malware detection, to decide if given a file is a malware. You may confuse with malware classification problems, to classify what type of malware is for a given malware. Some deep learning approaches for malware classification have been researched seperately [11], [12].

3. Architecture and Implementation

Our new 2d-based deep learning model for malware detection accepts raw binary file itself and decides if it is malicious or not. We introduce a new embedding method where a binary input file is considered as a black and white binary image, not a grayscale image (as in the most previous 2d-based studies). In addition, we optimize model parameters (filter size, kernel size, pool size) to be suitable for the binary image data. In the experiments, we use two different datasets for evaluating models. For training purpose, we use KISA Malware Challenge 2018 [13] dataset, and for testing, we utilize a dataset for Microsoft Malware Classification Challenge (BIG 2015) at Kaggle [14].

3.1 Black and White (0/1 Bit) Embedding

Figure 1 shows an illustrative example of transforming the input file. In the previous approaches, each byte in a binary file considers a pixel value from 0 to 255, which may not be an appropriate way for binary file conversion. For example, some fields of PE Header, or some bit positions represent a flag that a certain characteristic is on or off. 1(0x0001), 2(0x0010), 16(0x1000) are different values, and it does not mean that 0x0010 is nearer than 0x1000 at 0x0001. One hot encoding may use this case, however, each byte transforms to 255 dimensional vectors so that it makes the model parameter extremely huge. Therefore, we treat the binary file as an *one/zero (black and white)* bit image, rather than as a grayscale byte image.

3.2 Architecture

Our model architecture depicted in Fig. 2 is similar to other convolution neural network (CNN) models, consisting of two blocks by two convolutions and a max pooling to extract features from input, and three dense layer to classify with the extracted features. However, our model is *specialized* for handling binary image data. For example, in our convolution layers, the kernel size and stride size are the multiplies of power of two (i.e., (16, 64), (8, 8), (8, 32), (16, 16)) not conventional odd numbers (e.g., (3, 3) or (5, 5)). Since an executable file format strictly follows byte alignment, the choice of the multiplies of power of two at kernel size and stride size will prevent from disrupting byte alignment during training. The global max pooling layer basically

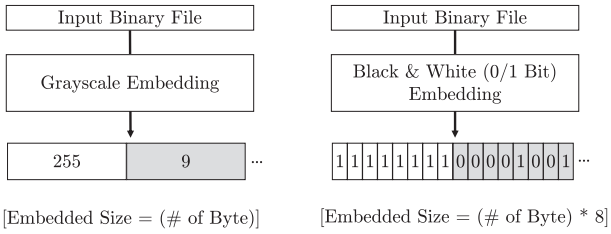

Fig. 1 Embedding: grayscale (left) v.s. black and white (right)

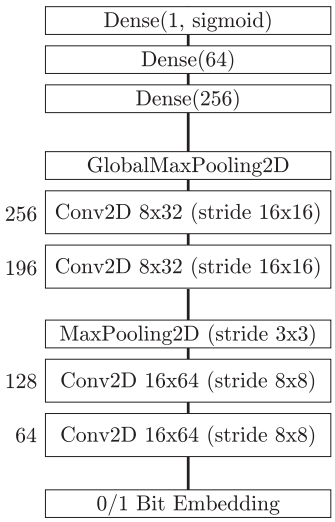

Fig. 2 Diagram of our model

captures the most significant features from each convolution layer's activation map. In addition, it can help to deal with various file sizes without affecting any layers in our model (especially dense layers). In our model, most activation functions use rectified linear unit (ReLU) except the last dense layer which uses sigmoid for activation function. Note that the stride size at the third and fourth convolution layers is relatively bigger than the kernel size so that we skip some output features from the previous layers. Nevertheless, with various combination of model parameters, this type of a little odd architecture shows the best and stable performance. Further research may be required to investigate the effects of this approach.

3.3 Data Sets

One of the most difficult problems to malware research is collecting malware samples. Most existing research works have collected some malware samples from antivirus companies or well-known malware websites. However, they cannot freely open and share these datasets because they do not have any copyrights, and real malware samples could be reused for malicious purposes. Recently, EMBER [15], a malware dataset with 1.1 million samples has been published. Unfortunately, they only provide the analysis results of malware's portable executable (PE) header along with SHA256 hash codes, which identify original binary files.

They do not reveal binary format data which we need in order to train and test our models.

For our experiments, we tried to find reliable and trustable data sets with binary format. At KISA Malware Challenge 2018, KISA provides total 8,904 binary files for the purpose of contests which are composed of 2,721 benign files and 6,183 malware files. KISA dataset consists of 2,478 malware families classified by avclass program [16]. Among these malware families, 2,368 ones are singleton (only one file) so that this dateset includes a quite wide range of malware samples. The top 5 malware families are Darkcomet(a.k.a Backdoor), Upatre(a.k.a Trojan), Virut(a.k.a Botnet), Gandcrab(a.k.a Randsomeware), and Virlock(a.k.a Randsomeware).

In our experiments, KISA dataset is used for training phase. The dataset is split into two sub-datasets for training (8,000 samples) and validation (904 samples) purposes. The validation dataset is used to fairly evaluate a training model and help to prevent a model from over-fitting against the training dataset too much. Also, for the purpose of testing models, we use a well-known data set from Microsoft Malware Classification Challenge (BIG 2015) at Kaggle which provides 10,868 malware files with 9 different malware types: Ramdit(a.k.a Worm), Lollipop(a.k.a Adware), Kelihos.ver3(a.k.a Backdoor), Kelihos.ver1(a.ka.a Backdoor), Vundo(a.k.a Trojan), Simda(a.k.a Backdoor), Tracur(alias Genome, a.k.a Trojan), Obfuscator.ACY(a.k.a Obfuscated malware), and Gatak(a.k.a Backdoor) [14]. Note that KISA dataset also includes all Microsoft's 9 malware families (among 2,478 families). The contest gives us text type disassembly files, rather than binary format. Fortunately, we can translate disassembly files into binary format files. Note that disassembly files do not have PE header of the file so that we cannot fully restore the original malwares. However, this data set is useful enough to compare the performance.

3.4 Implementation

We implemented our model with Keras in Tensorflow 1.13 and performed experiments on Ubuntu 18.04 desktop 64bit OS with AMD 2700 (16 Core CPU), 32GB Memory and NVIDIA Titan RTX (vram 24GB) GPU. The source code and data will be published at github [17].

For the experiments, the hyper-parameters for optimizations have been selected as follows. For the optimization function, the stochastic gradient descent (SGD) is chosen, the learning rate is set to 0.01, the nesterov option to be true, the decay factor to 1e-6, and the momentum is set to 0.9. The loss function is binary crossentropy since we only need to decide whether the input file is malware or not. The maximum epoch is set to 50. During the training process, we use early stopping with validation accuracy to obtain a reasonable model which does not substantially overfit to the training data. All comparison models use these hyper-parameters. Note that any changing of hyper-parameters should not affect overall performance significantly.

4. Evaluation

In this section, we present comparative experimental results of our proposed 2d-based malware detection deep learning model with VGG and ResNet which are widely used in existing image recognition areas. We use VGG16, ResNet50 models implemented by official Keras team [18]. As we described in Sect. 3.3, we leverage two different data sets for training and testing purposes respectively. During the training phase, KISA dataset is used for training and validation, and Microsoft dataset [14] is employed for testing purpose. This is because we want to estimate how the models are generalized enough to effectively detect malwares.

For fair comparison, we use the first 65,536 (=256x256) bytes from each binary file because the architecture of VGG16 and ResNet50 was originally designed to be suitable with 256x256 images. In our model, we apply our black and white embedding respect to 256x256 input bytes, then finally obtain 256x2048 black and white image. To verify the effectiveness of our embedding algorithm, we introduce a modified version of our model, called Mal2d/8, which divides the second dimension of the kernels and strides by 8 in order to handle 256x256 grayscale images. This will demonstrate how much our black and white embedding can affect the overall performance. Moreover, we make another test cases for 4,096 (64x64) bytes in order to investigate the performance impacts according to the size of binary files. For each model, we have performed experiments 10 times and choose the best one on the training set. For the performance metrics, we use *Accuracy*, *Precision*, and *Recall* as followings:

$$\begin{aligned} Accuracy &= (TP + TN) \,/\, (TP + TN + FP + FN) \\ Precision &= TP \,/\, (TP + FP) \\ Recall &= TP \,/\, (TP + FN) \end{aligned}$$

where TP, FP, TN, FN denote the number of malware detections proved to be true positive, false positive, true negative, and false negative respectively.

Table 1 shows the experimental results in terms of accuracy on training and validation based on the KISA dataset (as described in Sect. 3.3). Most models can achieve over 97% of training accuracy, and also in validations they exceed over 85% of accuracy, which are reasonably high enough. This can show that 2d image based approaches for malware detection are actually effective.

According to Saito et al. [19], checking *precision* and *recall* of models is more accurate to measure the model per-

Table 1 The accuracy result at the training set

models	64x64		256x256	
	train acc.	valid acc.	train acc.	valid acc.
VGG16	0.9819	0.9215	0.9888	0.8905
ResNet50	0.9748	0.8927	0.8751	0.8451
Mal2d/8	0.9765	0.8938	0.9831	0.8518
Mal2d	**0.9925**	**0.9281**	**0.9986**	**0.8971**

Table 2 Model comparison with precision and recall at the training set

models	type	64x64		256x256	
		precision	recall	precision	recall
VGG16	malware	0.9850	0.9854	0.9833	0.9838
	benign	0.9669	0.9658	0.9632	0.9621
ResNet50	malware	0.9534	**0.9926**	0.9168	0.9481
	benign	**0.9814**	0.8897	0.8721	0.8045
Mal2d/8	malware	0.9820	0.9694	0.9775	0.9765
	benign	0.9325	0.9596	0.9468	0.9489
Mal2d	malware	**0.9941**	0.9871	**0.9921**	**0.9905**
	benign	0.9711	**0.9868**	**0.9784**	**0.9820**

Table 3 Model comparison with precision and recall at the test set

models	type	64x64		256x256	
		precision	recall	precision	recall
VGG16	malware	0.9860	0.4659	**0.9799**	0.7437
	benign	0.1167	0.9142	0.1946	0.8021
ResNet50	malware	0.9746	**0.9637**	0.9388	0.4845
	benign	0.5890	0.6746	0.0813	0.5912
Mal2d/8	malware	0.9853	0.7239	0.9631	0.8214
	benign	0.1939	0.8605	0.2039	0.5924
Mal2d	malware	**0.9893**	0.8561	0.9735	**0.9220**
	benign	0.3206	0.8796	0.4003	0.6746

formance in the case of binary classification and unbalanced data set. Our malware detection mechanism uses binary classification and the data set is unbalanced since the number of malware files are two times more than that of benign files. Therefore, we have employed recall and precision of models as additional performance metrics and the results are presented in Table 2.

In the training data set, the overall performances of all comparison models are reasonably good enough as expected (as seen from Table 2). Our proposed model ("Mal2d") consistently outperforms other models in all factors. In addition, based on the comparison between "Mal2d" and "Mal2d/8", our black and white embedding has $2 \sim 3\%$ performance gain for all cases. This empirically shows that our embedding has meaningful effects. Since it only has the value of 0 or 1 like one-hot encoding, it may make a problem simple to learn along with preserving the information of some bit flags.

In Table 3, we present results against the test data set from Microsoft Malware Contests, which does not disclose themselves during the training stage. This result may show the generality of a deep learning model. The original data set consists of only malware samples so that for comparative analysis with the previous results in Table 2, we added 893 benign files obtained from 32bit application executable files in MS Windows 7.

Based on the experimental results in Table 3, we observed that precisions for malware detection are relatively preserved well, however the recall rates of VGG and ResNet are significantly dropped over $20\% \sim 50\%$. On the other hand, our model's recall rate still preserves over 85% overall, potentially due to our relatively shallow layers than other comparison models. Typically, deeper models can be expressed with much more complex functions, so that they have higher chances to fall in over-fitting. Also with our black and white embedding and the enforcement of byte alignment (as discussed in Sects. 3.1 and 3.2), our model can reduce the overall search space and achieve better generalization.

For the case of benign files in Table 3, we should consider the results for reference only. The number of benign files are relatively small and it has not been engineered for a useful benchmark. However, these results of benign files in a trained model gives us a hint about whether the model shows only a biased answer without considering inputs. For example, in the case of 256x256, the precision gap between malware and benign files at ResNet is over 85%. Although our proposed model's gap is also high, but it can achieve the *smallest* gap among the all comparison models. Interestingly, ResNet in the case of 64x64 demonstrates the best recall rate among all models. In this case, most experimental results were poor, however, only one out of ten was extremely superior. Similarly, we have found that some of our models were superior to any other ResNet models in test cases. However, the model cannot be chosen since we typically select a model having the best performance in the training dataset, *not* on the test dataset.

To summarize, our proposed model has achieved superior performance to the other models in both training and test data sets. Through the experiments, we verified that our architecture can prevent from disrupting the byte-alignment, and help to learn the proper malware features. Also, we observed that conventional deep learning approaches with image recognition models can show reasonable performance for malware detections, however, the models may have a certain limit to generalize unseen data.

5. Conclusion

In this paper, we proposed an effective 2d based deep learning model for malware detection. We treat a binary file as one/zero bit image and build our learning model with the kernel and stride sizes to multiplies of power of two, in order to optimize it for processing binary image files. Evaluation results show that our proposed model can achieve higher accuracy than other 2d based object recognition deep learning models, VGG and ResNet.

Although, our solution has showed better performance than other 2d based comparison models, it may not reach the level of commercial malware detection solutions. The first ranked team at MS Malware Classification Contest at Kaggle has deployed several features such as opcode counting with 2,3,4-gram, assembly image feature, function names, and consequently showed extremely high accuracy. Currently, merging multiple features might be a promising way for malware detection. However, our ultimate goal is to devise a simple yet effective mechanism to train a single deep learning model with only binary raw input files.

Acknowledgments

This research was supported by 2018 Research Fund of Myongji University.

References

[1] J. Lee, M.J. Jo, and J.S. Shin, "Ligeroav: A light-weight, signature-based antivirus for mobile environment," IEICE Trans. Inf. & Syst., vol.E99-D, no.12, pp.3185–3187, Dec. 2016.
[2] V. Harrison and J. Pagliery, "Nearly 1 million new malware threats released every day," http://money.cnn.com/2015/04/14/technology/security/cyber-attack-hacks-security/.
[3] McAfee, McAfee Labs Threats Report, Dec. 2018.
[4] J. Scott, Signature Based Malware Detection is Dead, Institute for Critical Infrastructure Technology, 2017.
[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," ICLR 2015, San Diego, CA, May 2015.
[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," CVPR 2016, Las Vegas, NV, pp.770–778, June 2016.
[7] L. Nataraj, S. Karthikeyan, G. Jacob, and B.S. Manjunath, "Malware images: visualization and automatic classification," VizSec 2011, Pittsburgh, PA, pp.4:1–4:7, July 2011.
[8] E. Rezende, G. Ruppert, T. Carvalho, A. Theophilo, F. Ramos, and P.D. Geus, "Malicious software classification using vgg16 deep neural network's bottleneck features," Information Technology - New Generations, vol.738, pp.51–59, 2018.
[9] R.U. Khanz, X. Zhang, and R. Kumar, "Analysis of resnet and googlenet models for malware detection," Journal of Computer Virology and Hacking Techniques, vol.15, no.1, pp.29–37, 2019.
[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," CVPR 2015, Boston, MA, pp.1–9, June 2015.
[11] R. Kumar, Z. Xiaosong, R.U. Khan, I. Ahad, and J. Kumar, "Malicious code detection based on image processing using deep learning," 2018 International Conference on Computing and Artificial Intelligence, pp.81–85, 2018.
[12] D. Gibert, C. Mateu, J. Planes, and R. Vicens, "Using convolutional neural networks for classification of malware represented as images," Journal of Computer Virology and Hacking Techniques, vol.15, no.1, pp.15–28, 2019.
[13] Korea Internet and Security Agency (KISA), "Data Challenge 2018," http://datachallenge.kr/challenge18/malware/.
[14] R. Ronen, M. Radu, C. Feuerstein, E. Yom-Tov, and M. Ahmadi, "Microsoft malware classification challenge," arXiv:1804.10135, Feb. 2018.
[15] H.S. Anderson and P. Roth, "Ember: An open dataset for training static pe malware machine learning models," arXiv:1804.04637, April 2018.
[16] M. Sebastián, R. Rivera, P. Kotzias, and J. Caballero, "Avclass: A tool for massive malware labeling," International Symposium on Research in Attacks, Intrusions, and Defenses, vol.9854, pp.230–253, Springer, 2016.
[17] Mal2D. https://github.com/minkcho/mal2d.
[18] Keras team, https://github.com/keras-team/keras.
[19] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets," PLOS ONE, vol.10, no.3, pp.1–21, 03 2015.