Malware Detection Using Deep Transferred Generative Adversarial Networks

Jin-Young Kim, Seok-Jun Bu, and Sung-Bae Cho(✉)

Department of Computer Science, Yonsei University, Seoul, Korea
{seago0828,sjbuhan,sbcho}@yonsei.ac.kr

Abstract. Malicious software is generated with more and more modified features of which the methods to detect malicious software use characteristics. Automatic classification of malicious software is efficient because it does not need to store all characteristic. In this paper, we propose a transferred generative adversarial network (tGAN) for automatic classification and detection of the zero-day attack. Since the GAN is unstable in training process, often resulting in generator that produces nonsensical outputs, a method to pre-train GAN with autoencoder structure is proposed. We analyze the detector, and the performance of the detector is visualized by observing the clustering pattern of malicious software using t-SNE algorithm. The proposed model gets the best performance compared with the conventional machine learning algorithms.

Keywords: Malicious software · Zero-day attack · Generative adversarial network · Autoencoder · Transfer learning

1 Introduction

Malicious software called malware is a generic term for all software that adversely affects computers. As the malware is growing endlessly in speed, number and discrepancy [1], malware detection is an important issue. However, malware developers modify the existing malware in order not to be detected by the vaccine. To detect and treat malware, sampling and detecting each feature of them are necessary because even malwares with the same function have different characteristics. As the number of malwares has been increasing, so does that of malwares that need to be sampled, so that automatic detection of malware reduces the cost of it.

Another problem of malware detection is the zero-day attack. It is undisclosed computer-software vulnerability that hackers can exploit to adversely affect computer programs, data, additional computers, or a network. This kind of malware is more threatening because it causes damage before enough data is collected to make the vaccine. To cope with the zero-day attack rapidly, we propose tGAN that detect and treat malware with a small amount of data.

D. Liu et al. (Eds.): ICONIP 2017, Part I, LNCS 10634, pp. 556–564, 2017.
https://doi.org/10.1007/978-3-319-70087-8_58

2 Related Work

There are many kinds of research to analyze and detect malware. The study on malware is to visualize malware or to figure out its function through malware code. The malware code was made up into a tree-like graph, and each tree was compared with others to classify malware. Recently, the trend on the research shows the malware classification using machine learning are increasing, and the zero-day attack. Since malware has common characteristics, the zero-day attack is detected by features such as diversity of destinations, payload repetition, small size and so on. In other approaches, there are two modules that detect malware in a different way. For example, one module detects malware through explicit analysis. Another module detects malware that appears to be normal, but can damage computer programs through relationships with others. Table 1 summarizes the examples of malware analysis and classification studies. They did not use machine learning to find out the complex relationship of data, and even if they use, they do not cope with the zero-day attack rapidly because they use a lot of malware data. For this reason, we propose a method of detecting malware using machine learning with a small amount of data so that the zero-day attack is detected in a short time.

Table 1. The relevant works on malware

Malware analyzing and classification		
Author	Method	Description
M. Christodorescu (2005) [2]	Type comparison with example data	Using the stored information of malware
L. Nataraj (2011) [3]	Visualization through code vectorization	Convert malware codes to malware images
D. Kong (2013) [4]	Clustering	Malware detection with clustering method
R. Pascanu (2015) [5]	Recurrent neural networks	Interpret malware as time series data
Zero-day attack		
Author	Method	Description
P. Akritidis (2005) [6]	Content-based detection	Malware detection based on several observations
M. Grace (2012) [7]	Two modules in in different way	Ensemble of two modules to detect malware

3 tGAN Model

To use deep learning for malware detecting, we convert malware codes to images, called malware images, as shown in Sect. 4.1. Deep learning requires a lot of data,

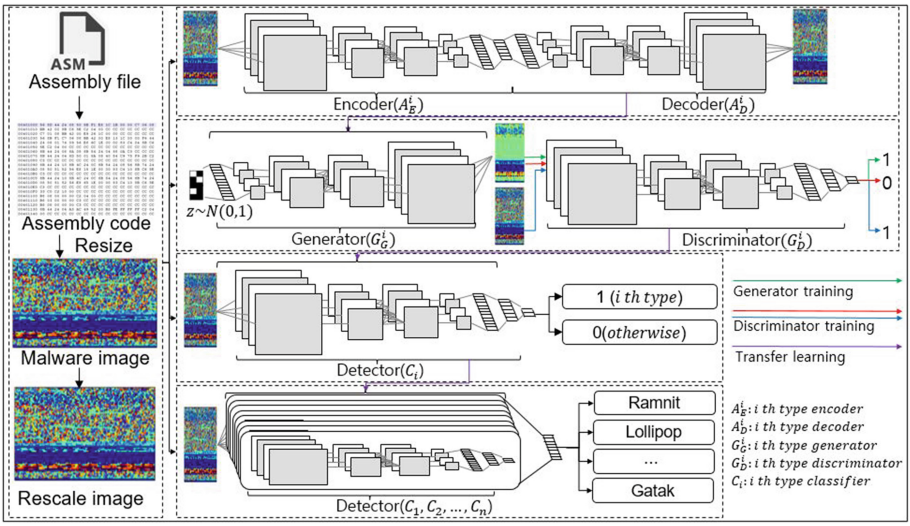

Fig. 1. The architecture of proposed model.

so we have to increase the number of data. We create the data using tGAN model based on GAN. Figure 1 illustrates the architecture of proposed model. It is consists of three parts; pre-training module, generating data module and malware detecting module. First module pre-train second module which has a generator that generates data similarly for real data, and a discriminator that distinguishes real data from generated data [8]. The discriminator is trained to distinguish the actual data from the generated data, and the generator is trained to make the discriminator to classify the generated data into the real data. Equation (1) shows the objective function of GAN model as described by Goodfellow et al. [8].

$$\min_G \max_D V(D,G) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))]. \quad (1)$$

As the learning progresses, the discriminator has robustness against the transformation of the data through the data generated continuously [9]. This is effective for classifying malware by the fact that malware is continuously generated by being deformed differently from the existing features. Generator creates the data which is similar to the actual data. Since the discriminator continuously learn with data generated from the generator, GAN model can extract and learn features of data even with a small amount of malware data. It allows the GAN model to be effective regarding to zero-day attack. However, it has been known to be unstable in training process, often resulting in generator that produces nonsensical outputs [9]. To solve this problem, the generator is trained through the autoencoder. Autoencoder is a unsupervised learning method that finds the characteristics of data given only the input values of the data [10].

There were many applications of autoencoder such as denoising autoencoder [11], sparse autoencoder [12] and language learning [13]. Given an input value x, the encoder transforms x into compressed data $f(x)$ and the decoder reconstructs the compressed data $f(x)$ to z. Equations (2) and (3) show the process of autoencoder.

$$z = g_{\theta'}(y) = s(W'y + b') \tag{2}$$

$$y = f_{\theta}(x) = s(Wx + b) \tag{3}$$

where s is the activation function. Deep autoencoder is used by the model proposed in this paper to train more complex characteristics: The encoder and decoder of the autoencoder have several layers.

To apply the autoencoder to the GAN model, transfer learning method is used. It is a method of learning by transferring the learned model to the other model that performs similar work in different domain [14]. From the viewpoint of generating data after the generator learns the characteristics of the data, if the autoencoder is transferred into the generator of the GAN, it can be intuitively known that data is generated better than when the transfer learning is not used.

We create tGAN models that generate malware image at each type for professional data generation. The encoder of autoencoder consists of a fully-connected layer that grows the data size up to the middle size of the actual data, deconvolutional layers [16] and pooling layers that generate data whose size is same to real data. We stack two layers of 3×3 deconvolutional layers and pooling layers. The decoder of autoencoder consists of convolutional layers [17] and pooling layers, which extract the characteristics of data, and fully-connected layer. To transfer decoder of autoencoder into generator of GAN, the architecture of generator is same for the decoder of autoencoder. Finally, we transfer each discriminator of GAN into detector that detects whether it is the same type of malware image with the type of detectors. The result of each detector for the given data is calculated and the type of data is determined by the detector with the largest value. The training process of the proposed model is as follows.

Input: Preprocessed data $T = \{T^1, \ldots, T^N\}$, where $T^i = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, Encoder of autoencoder A_E, Decoder of autoencoder A_D, generator of GAN G_G, discriminator model of GAN G_D, the number of malware types N
Output: Detector $D = \{D_1, \ldots, D_N\}$
for $i = 1, \ldots, N$ do
  Train A_E^i, A_D^i with $T_A^i = \{x_1^i, \ldots, x_n^i\}$;
  Transfer A_D^i into G_G^i;
  Train G_G^i, G_D^i with $T_{(G_G)}^i = \{(G^i(z)_1, 1), \ldots, (G^i(z)_n, 1)\}$, where $z \in N(0,1)^k$ and $T_{(D_G)}^i = \{(x_1^i, 1), \ldots, (x_n^i, 1), (G^i(z)_1, 0), \ldots, (G^i(z)_n, 0)\}$ using Equation (1);
  Transfer G_D^i into D_i;
end for
return D=$\{D_1, , D_N\}$

4 Experiment

We conducted experiments to test whether the malicious software could be detected sufficiently even with a small amount of data. Two experiments are conducted. One is using entire data for learning, and the other is using only a small amount of data for learning.

4.1 Dataset

To test the performance of the tGAN model, we use the malware data used in the Kaggle Microsoft Malware Classification Challenge [15]. Table 2 summarizes the data utilized for the learning and verification. Simda malware is not used in the second experiment because it has too little data. Since the malware data is formed as binary code, it cannot be used in the tGAN model directly. Therefore, each malware data is converted in the size when malware data is compiled. After that, the data is rescaled by an average of the sizes of all the rows and columns of the data. We use the image of transformed malware data, say malware image.

Table 2. Malicious software data used in tGAN

Type	Samples	Type	Samples	Type	Samples
Ramnit (R)	1539	Lollipop (L)	2459	Kelihos_ver3 (K3)	2942
Vundo (V)	451	Simda (S)	42	Tracur (T)	744
Kelihos_ver1 (K1)	391	Obfuscator.ACY (O)	391	Gatak (G)	1011

If k is the length of the binary code, C is the size of the transformed column, and R is the size of the transformed row, the method of calculating the transformed column and row size is shown in Eqs. (4) and (5). After this transforming, data value becomes between zero and one. The top images of Fig. 2 illustrate the data. The closer to one, the more red; the closer to zero, the more blue.

$$C = 2^{\frac{\log \sqrt{k*16}}{\log 2}+1} \tag{4}$$

$$R = \frac{k*16}{C} \tag{5}$$

4.2 Result

Generated Images. The generated malware images are used to pre-train malware detector. We show the malware image generated by the tGAN in Fig. 2. The generated images are not exactly the same to the actual images, but it has only a few variations. Since malware developers develop malware with some modifications from the existing ones, these modified images might be useful to detect malware.

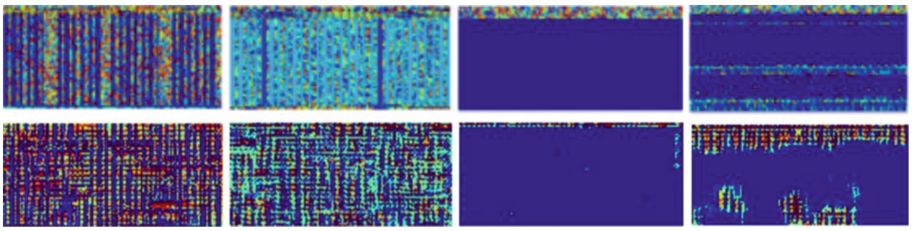

Fig. 2. Actual malware images (Top) and generated malware images (Bottom). (Color figure online)

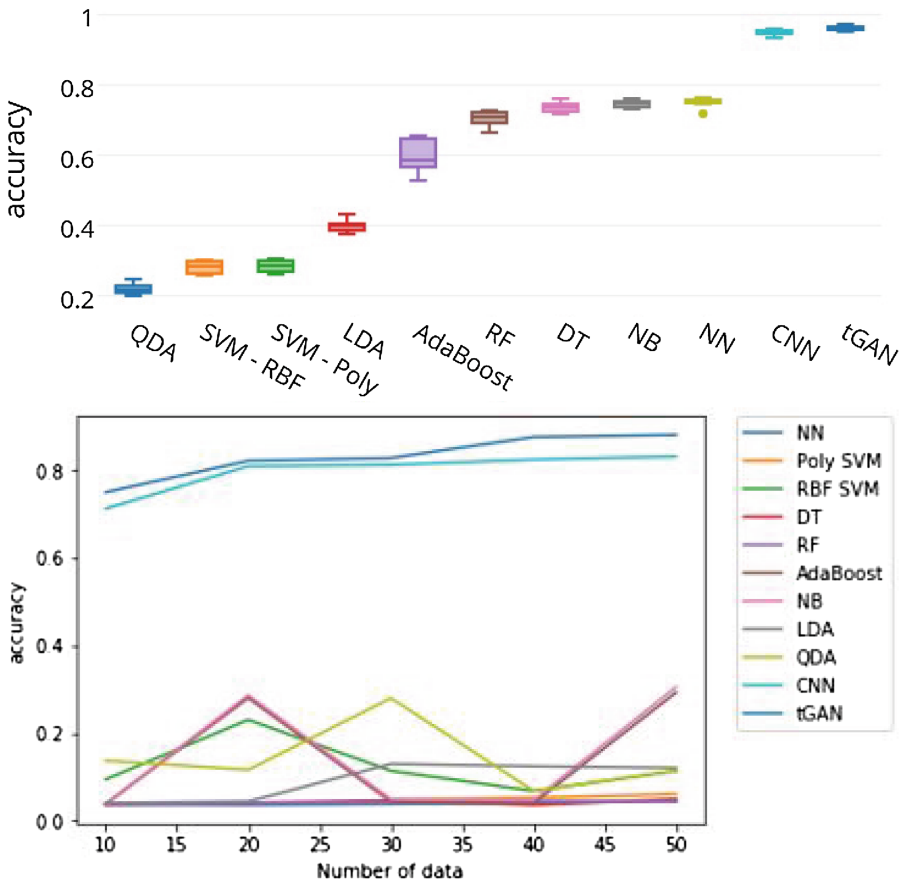

Fig. 3. Comparison with other models in experiment 1 (Top). Comparison with accuracy by number of data variation. (Bottom). (RF: Random Forest, DT: Decision Tree, NB: Nave Bayes, and NN: Nearest Neighbor). The parameters of the other models are set as the default value in sklearn except $k = 3$ in k-NN, penalty parameter $C = 0.025$ in SVM-Poly, kernel coefficient $\gamma = 2$ in SVM-RBF, max depth $= 5$ in decision tree and random forest and max features $= 1$ in random forest.

Accuracy. In the first experiment, we use all the data to train tGAN models and malware detector. The accuracy of malware type detection is 96.39%. The entire data is divided into training and test data at a ratio of 90:10. We compare accuracy with other models through 10-fold cross validation. Figure 3 illustrates the box plot of the result. Compared with other papers [18,19], our results are similar in performance despite rescaled data used in proposed model.

In the second experiment, we use only 10, 20, 30, 40 and 50 data for each malware type in training process. Therefore, overall accuracy is lower, but the proposed model still performs better than the others. The proposed model can detect malware with a small amount of data; it can quickly respond to the malware of which we do not have enough information. A comparison of the accuracy according to the number of data is shown in Fig. 3.

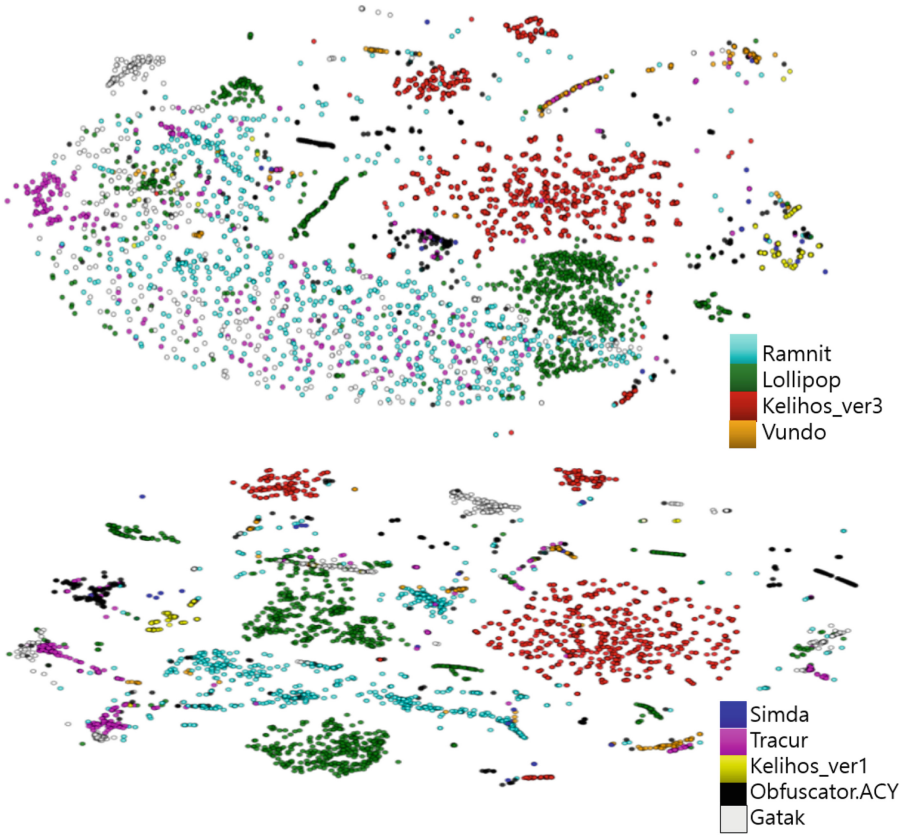

Fig. 4. Distributions of the raw data of malware (Top) and the malware detector output values (Bottom). Malware detector clusters data well by type of malware.

Model Analysis. t-SNE algorithm is used to visually confirm whether the proposed model actually detects malware well. The t-SNE algorithm visualizes the clustering pattern [20]. Figure 4 illustrates the distribution of the raw malware data and the distribution of the output values of malware detector. We can see that the distribution of the output values of malware detector is better clustered than the raw malware data.

5 Conclusion

In this paper, we have proposed a method to pre-train the generator of GAN using the transfer learning and to pre-train malware detector through the discriminator of GAN. It shows the best performance compared with other conventional models, and it enables to detect malware even with a small amount of data. It means that it can quickly respond the malware of which we do not have many information. Even using all of the data, proposed model has the best performance for detecting malware. Therefore, the model is useful for detecting existed malware as well as preventing the zero-day attack. We will experiment to detect the zero-day attack by modeling that in future work. We will test that tGAN has generated the malware well.

Acknowledgements. This work was supported by Defense Acquisition Program Administration and Agency for Defense Development under the contract (UD160066BD).

References

1. Dhammi, A., Singh, M.: Behavior analysis of malware using machine learning. In: IEEE International Conference on Contemporery Computing, pp. 481–486 (2015)
2. Christodorescum, M., Jha, S., Seshia, S.A., Song, D., Bryant, R.E.: Semantics-aware malware detection. In: Security and Privacy, pp. 32–46 (2005)
3. Nataraj, L., Karthikeyanm, S., Jacob, G., Manjunath, B.S.: Malware images: visualization and automatic classification. In: Proceedings of the Conference on Visualizing for Cyber Security, p. 4 (2011)
4. Kong, D., Guanhua, Y.: Discriminant malware distance learning on structural information for automated malware classification. In: Proceedings of the Conference on Knowledge Discovery and Datamining, pp. 1357–1365 (2013)
5. Pascanu, R., Stokes, J.W., Sanossian, H., Marinescu, M., Thomas, A.: Malware classification with recurrent network. In: Acoustics, Speech and Signal Processing, pp. 1916–1920 (2015)
6. Akritidis, P., Kostas, A., Evangelos, M.P.: Efficient content-based detection of zero-day worms. In: Communications, vol. 2, pp. 837–843 (2005)
7. Grace, M., Zhou, Y., Zhang, Q., Zou, S., Jiang, X.: RiskRanker: scalable and accurate zero-day android malware detection. In: Proceedings of the Conference on Mobile Systems, Applications, and Services, pp. 281–294 (2012)
8. Goodfellow, I., Pouget-Abadie, J., Mirze, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 2672–2680 (2014)

9. Radford, A., Luke, M., Soumith, C.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)
10. Bourlard, H., Yves, K.: Auto-association by multilayer perceptrons and singular value decomposition. Biol. Cybern. **59**, 291–294 (1988)
11. Lu, X., Matsuda, Y., Hori, C.: Speech enhancement based on deep denoising autoencoder. In: Interspeech, pp. 436–440 (2013)
12. Ng, A.: Sparse autoencoder. CS294A Lecture notes, vol. 72, pp. 1–19 (2011)
13. Chandar AP, S., Lauly, S., Larochelle, H., Khapra, M., Ravindran, B., Raykar, C.V., Saha, A.: An autoencoder approach to learning bilingual word representations. In: Advances in Neural Information Processing Systems, pp. 1853–1861 (2014)
14. Arnold, A., Nallapati, R., Cohen, W.: A comparative study of methods for transductive transfer learning. In: Proceedings of the IEEE International Conference on Data Mining, pp. 77–82 (2007)
15. Kaggle: Microsoft Malware Classification Challenge (BIG 2015). https://www.kaggle.com/c/malware-classification. Accessed 4 Nov 2015
16. Zeiler, M., Taylor, G., Fergus, R.: Adaptive deconvolutional networks for mid and high level feature learning. In: IEEE International Conference on Computer Vision, pp. 2018–2025 (2011)
17. Lecun, Y., Bengio, Y.: Convolutional networks for images, speech, and time series. In: The Handbook of Brain Theory and Neural Networks, vol. 3361, p. 1995 (1995)
18. Jake, D., Tyler, M., Michael, H.: Polymorphic malware detection using sequence calssification methods. In: Security and Privacy Workshops, pp. 81–87 (2016)
19. Narayanan, B.N., Djaneye-Boundjou, O., Kebede, T.M.: Performance analysis of machine learning and pattern recognition algorithms for malware classification. In: Aerospace and Electronics Conference and Ohio Innovation Summit, pp. 338–342 (2016)
20. Maaten, L., Hinton, G.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9**, 2579–2605 (2008)