Generating Adversarial Malware Examples with API Semantics-Awareness for Black-Box Attacks

Xiaowei Peng, Hequn Xian(✉), Qian Lu, and Xiuqing Lu

Qingdao University, Qingdao 266071, China
xianhq@qdu.edu.cn

Abstract. As deep learning plays a key role in malware detection, it is of great practical significance to study adversarial malware examples to evaluate the robustness of malware detection algorithm based on deep learning. A black-box attack is performed while malware authors are allowed only access to the input and output of the malware detection model. Due to the transferability of deep learning model, it is an effective way to train a substitute model to fit the black-box model. Generative adversarial network based models show good performance in generating adversarial examples and training substitute models. However, because of the discrete output, generative adversarial networks are unable to compute gradient for their back-propagation, which makes it difficult to update the weights of the neural network in the training process. In addition, APIs are important features in representing malware, but their potential semantic features are usually ignored. To address the above problems, a generative adversarial network based algorithm with API word embedding method is designed, which adopts CNN structure to train a substitute model. The substitute model is utilized to analyze the semantic association of sequential API calls. Then, we employ a long short-term memory framework to generate antagonistic examples. The experimental results show that the proposed scheme is efficient and effective.

Keywords: Adversarial malware examples · Black-box attack · Generative adversarial network(GAN) · API word embedding · Long short-term memory(LSTM)

1 Introduction

With the booming of Social Networks, tremendous amount of data is being produced world-widely. New methods and new software tools are continuously developed and dispersed to capture, manage, and process Big Data. Various categories of malwares have raised security issues and posed critical challenges for Social Networks and Big Data applications. Malware detection methods and techniques are considered to be of great importance. Deep learning has been widely used in computer vision, natural language processing and many other application realms [1, 2]. In malware detection, deep learning models outruns traditional tools with its outstanding ability to recognize the essential

Y. Xiang et al. (Eds.): SocialSec 2020, CCIS 1298, pp. 52–61, 2020.
https://doi.org/10.1007/978-981-15-9031-3_5

features of examples and classify them effectively. Powerful as they are, deep learning models is vulnerable to disturbance from slightly tempered input [3, 4]. Adversarial examples, generated by adding perturbations to normal examples, can easily compromise the working of a deep learning model. They reveal basic blind spots in deep learning algorithms [5].

Researchers try to generate adversarial examples through countermeasure techniques. In some early research works, gradient based methods were used to generate adversarial examples, which fool the detection model through exemplifying the prediction error of the neural network and optimizing the input [7]. However, all these researches are carried out based on a fundamental assumption that malware authors have been granted full access privileges to the target model and can train detection models of their own. Actually, under most circumstances, malware detection algorithms based on deep learning are integrated into an antivirus software or deployed in the cloud. From the perspective of malware authors, the malware detection model is a black box without any information about the internal structure.

With the transferability of adversarial examples, an adversary in a confrontational attack can compromise the target detection model by simply training and attacking a substitute model. This type of attack is also recognized as the black-box attack [7]. Black-box attacks are usually built with supervised learning mechanisms [8]. The substitute model is trained with examples labeled by the target model. Then, adversarial examples can be generated according to the substitute model which is fully transparent to the attacker. However, once the algorithm of the target model is updated or replaced, malware authors will have to train another substitute model. There is another substitute model training method which is based on the generative adversarial network (GAN) [6]. It trains the substitute model with help of the multi-cognitive network within GAN, which trains a generation model simultaneously. The generation model and the substitute model train each other and account for each, as stated in the game theory, until a Nash equilibrium can be achieved. Eventually adversarial examples generated by the generation model can fool the substitution model. The GAN based model can enter a new round of adversarial training once the algorithm in the target model is changed. However, API is adopted as the character of malware examples, and sparse vectors are used to represent APIs in a software program [9], which is directly fed into the multi-cognitive network. The Euclidean distance between APIs is rendered identical, and the potential semantic features between APIs are overlooked [10].

According to the problems above, we propose a generative adversarial network model and adopt the method of word vector expression in natural language processing [11, 12], so that the neural network can extract characters from the API word list. Convolutional neural network (CNN) is used to train the substitute model, which analyzes the semantic association of sequential API calls, and explains the latent semantic features obtained by APIs. Then, we employ a long short-term memory framework to generate antagonistic examples.

The contribution of this paper can be summarized as follows:

- We devised a method to map the behavior of API calls into a vector space, so that the connection between APIs can be evaluated through measurement. The behavior pattern of the malwares can be effectively analyzed with machine learning method.

- We design a deep learning model that can extract the semantic association of APIs, which can make the most out of the sematic features to produce dedicated malware API sequences.

2 Adversarial Example Generation

2.1 Black-Box Attack and Substitution Model

From the perspective of an attacker, black-box attack may be the most challenging situation. Black-box attack can be divided into ordinary black-box attack and completely restricted black-box attack according to the degree of the restrictions [7]. Access to the input and output are allowed in an ordinary black-box attack, while in a completely restricted black-box attack, no information can be acquired. Neither kind of black-box attack can use the back propagation of the target model, so the question falls in the attack transferability from the self-trained model to the restricted black-box model [3]. Due to the enormous challenge of the completely restricted black-box attack, it falls out of the scope of this paper. In fact, attackers do have the right to inquire about the target model to obtain useful information for generating adversarial examples in many scenarios. For example, malware classifiers (usually trained by deep neural networks) allow attackers to input any binary file and they output the classification results, such as confidence score or category of classification. Then the attacker can use the classification results to design more effective adversarial examples to fool the target classifier. In this kind of black-box attack, the back propagation for gradient calculation of the target model is still prohibited, because the back propagation needs to understand the internal configuration of the target model, which is not available for a black-box. A better way to realize the attack transferability of the adversarial examples is to use the ability of free query to train the substitute model [13, 14]. Then, any white-box attack technology can be used to attack the substitute model, and the generated adversarial examples can be used to attack the target model.

The main advantage of the training a substitute model is that it is completely transparent to the attacker [15]. Therefore, the basic attack process of the target model, such as the back propagation of gradient calculation, can be achieved with the substitute model which is used to generate adversarial examples.

2.2 API Word Vectors

Malware programs call different APIs to implement corresponding malicious actions. We focus on the API call types distinguished from the naming, regardless of their parameters and return values. It can be seen that each API is represented by a number of English words or character combinations, which is similar to the text analysis problem in natural language processing [11]. Therefore, we try to extract all the API calls, regard the API sequence as a text sequence, and use the technology of word vectors to digitize each API.

First, we regard each API in the API sequence as an independent "word", and generate a word vector for it. A complete API sequence is denoted as W. In order to facilitate the

learning with neural networks, we set the length of each API sequence to the maximum API sequence length T in the sample set, T is the number of APIs contained in the sequence. For a sequence shorter than T, we make tail padding with 0. The sequence W composed of T API "words"$w_t(1 \leq t \leq T)$ is represented as:

$$W = (w_1, w_2, \ldots w_T) \tag{1}$$

Each API "word" w_t is then encoded using one-hot encoding. Suppose that the size of the input API dictionary is V_1. In this way, an API to one hot encoding mapping can be established. Each API uses a V_1 dimensional one-hot vector representation, that is,

$$w_t = \left(o_1, o_2, \ldots o_{V_1}\right)^T \tag{2}$$

There are t ones and the rest of the elements are zeros. If the one-hot encoded vector is directly feed into the deep learning model for training, the number of model parameters will be huge and extremely sparse [10]. Moreover, the one-hot encoding ignores the semantic relationship between words. In natural language processing, the one-hot encoded character text sequence needs to go through the word embedding layer [16] to generate the word vector whose dimension is much lower than V_1.

We feed the one-hot code of the API sequence into the word embedding layer, then obtaining a word embedding matrix $W_e \in R^{V_1 \times K}$ which is a matrix of parameters. According to the classification label of the API sequence, a word vector can be acquired by supervised learn, in which W_e is the table of word vectors. Each line of W_e is the K dimension word vector $x_t(1 \leq t \leq T)$ of every API.

$$x_t = w_e^T w_t \tag{3}$$

Given the one-hot vector of any API word in an API sequence,$o_t = 1, o_i = 0(1 \leq i \leq T, i \neq t)$, the word vector of the API is the t^{th} line of W_e. A word sequence X can be acquired from an API call sequence W.

$$X = (x_1, x_2, \ldots x_T) \in R^{K \times T} \tag{4}$$

2.3 Model Architecture

Our designing goal is to capture the dynamic behavior characteristics of malware calling API, and to reveal the collaborative relationship between API calling sequence and malicious purpose from a large number of training data. Our model consists of three components, the LSTM generator, the CNN substitute detector and the black-box. The architecture is shown in Fig. 1.

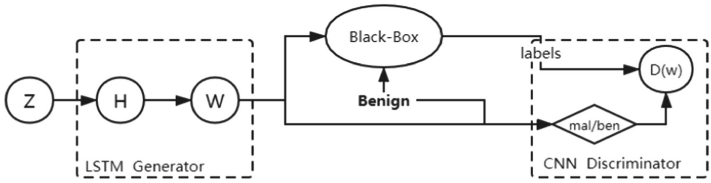

Fig. 1. The architecture of adversarial malware examples generation model.

LSTM Generator

The generator is based on long short-term memory network, known as LSTM. An LSTM unit is made of an input gate i_t, an output gate o_t, and a forget gate f_t. The input gate controls the extent of the new input value, the forget gate is used to manage how much a value should be left in the unit, the output gate decides which values are used to activate the LSTM unit. Repeated LSTM units can record some status of each moment, which in our design is the position of each API in the generated API sequence. An API sequence is synthesized by the LSTM generator with random noise vectors. Every LSTM unit is determined by its previous hidden status h_{t-1}, previous output y_{t-1} and the noise vector z. i_t and f_t are computed and then the current unit c_t and current hidden status h_t are computed. Functions within a LSTM unit is defined as follows.

$$i_t = \sigma(W_i y_{t-1} + U_i h_{t-1} + C_i z) \tag{5}$$

$$f_t = \sigma\left(W_f y_{t-1} + U_f h_{t-1} + C_f z\right) \tag{6}$$

$$o_t = \sigma(W_o y_{t-1} + U_o h_{t-1} + C_o z) \tag{7}$$

$$\tilde{c}_t = \tanh(W_c y_{t-1} + U_c h_{t-1} + C_c z) \tag{8}$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes \tilde{c}_t \tag{9}$$

$$h_t = o_t \otimes \tanh(c_t) \tag{10}$$

$W_{\{i,f,o,c\}}$, $U_{\{i,f,o,c\}}$ and $C_{\{i,f,o,c\}}$ are weight matrices, $\sigma(\cdot)$ is sigmoid function, $\tanh(\cdot)$ is hyperbolic tangent function, $\otimes$ stands for multiplication of elements. To simplify the computation, we omit the deviation term. Similar to the seq2seq (sequence to sequence) model [17], we use LSTM as the decoding network. Noise z is used as the input vector to control the generation of each API, which has the same dimension as the API word vector.

Substitute Detector

The substitute detector is constructed with convolutional neural network (CNN), which is supposed to simulate the target black box. An API sequence of length T contains T consecutive word vector, it goes through the CNN and produces the prediction result. In

the convolution layer, the hidden patterns in different API sequences can be found automatically by the convolution window sliding on the word vector sequence and detecting the features in different positions. A window is set up at the position of each word in the API sequence. The window size is the same as that of the convolution kernel. That is to say, convolution operation is performed on consecutive k APIs to generate a feature map. In the convolution neural network, multiple convolution kernels are often used at the same time. Here we record the number of convolution kernels as n_c. In this way, each convolution kernel generates a dimension's characteristic graph, and n_c convolution kernels will generate n_c dimension's characteristic graph, and connect them by columns, and finally get the characteristic graph set of API sequence.

Training
In the training processes, the generated API sequence and the benign API sequence are used to train the substitute detector together with the labels given by the black box. By optimizing the loss function, the substitute detector simulates the black box, and the same classification is given to the input API sequence as the black box.

The ultimate goal of the generator is to generate malware API sequences, send them to the substitute detector for detection, and let them be recognized as benign API sequences.

By optimizing the loss function, the probability of generated malware API sequences being recognized as malicious is reduced. When the adversarial training between the generator and the substitute detector reaches a balance point [7], the malware API sequence generated by the generator can fool both the substitute detector and the target black box simulated.

3 Experiments

Our experiments are implemented with Keras and Tensorflow. Keras is a high-level deep learning programming framework, integrating the implementation of a variety of neural networks. Based on python programming language and tensorflow backend, Keras supports accelerated training model on GPU.

3.1 Dataset and Configuration

We collected 6946 malware from the malware sample website ZOO and some open source projects of malware. The benign software comes from different types of common applications, such as complete free software, Softonic, Microsoft Windows system files, etc., totaling 2749. We use APIs as the dynamic characteristics of the samples. Cuckoo is chosen as the sandbox tools, which can extract malware behavior data such as Windows API call sequence.

We set the virtual machine VMware 15.5 in Ubuntu 16.04. The malicious samples are executed within Windows 7 through the virtual box. The running time threshold is 2 min. According to the output JSON file, we extract the API call sequence of the program. In order to offset the unnecessary behaviors on the call data and increase the generalization ability of the training model on other data sets, it is necessary to de

duplicate the continuous repeated API calls. After deduplication, we truncate the length of the sequence used for the experimental data to 400, and shorter API call sequences are padded with zeros. The API call sequence is represented as 400 * 300 dimensional API word vector matrix by word embedding.

For all of the sequence of the API calls dataset, we split 80% of the dataset as the training set and the remaining 20% as the test set. Then we randomly select 25% of the training set as the validation set.

Our experiments are carried out on a GPU workstation, which is equipped with NVIDIA Tesla V100 GPU, Intel Xeon 10 core CPU, 64 GB memory, CUDA 9.0 and cuDNN7.1 library, Cuckoo Sandbox with Windows7 X64.

Black box classification algorithm is usually not open-source to the public, and there is no tracking version available for experiments. The deep learning black-box malware classifier based on API call is difficult to obtain in the actual application scenarios. Therefore we train the malware classification algorithm separately as the black-box classifier needed in the experiment.

3.2 Evaluation

We create our own black-box malware classifier for the adversarial training process, which also allows us to evaluate the attack performance against many classifier types. The input of all the classifiers is a vector of 400 API calls in word embedding, each with dimension of 300. The output is binary: malicious or benign.

Black-Box Classifier performance
For all neural networks, we use the Adam optimizer. The output layer is fully-connected with sigmoid activation function, and a rectified linear unit RELU is chosen as the activation function of input layer and hidden layer due to its fast convergence compared with sigmoid () or tanh (). Dropout is used to improve the generalization potential of the network. We conduct training for a maximum of 100 epochs, but convergence is usually reached after 15-20 epochs, which depends on the type of classifier. Batch size of 256 samples is used.

We measured the performance of the classifiers using the accuracy ratio on the test set. The performance of all the black-box classifiers is shown in Table 1.

Table 1. Classifier performance

Classifier Type	Accuracy (%)	Classifier Type	Accuracy (%)
LR	90.59	MLP	95.08
DT	90.55	CNN	95.04
RF	92.49	LSTM	94.96
SVM	90.51	BiLSTM	95.39

As can be seen in Table 1, the classifier based on deep learning have a good performance in detecting malwares. BiLSTM is one of the classifier most resistant to our proposed attack based on the semantic features of API sequence.

Attack Performance of Generated API Sequence We evaluate the efficiency and effectiveness of our scheme by comparing it with Hu's work on true positive rate (TPR), which is the percentage of the number of malicious examples detected and the number of all examples samples. After adversarial attack, the reduction of TPR can effectively reflect the ability of adversarial examples to successfully bypass the black-box detector. The result is shown in Table 2.

Table 2. True positive rate on original malware samples and adversarial examples for both the train set and test set.

Classifier Type	Training Set		Test Set	
	Hu et al. (%)	Generated (%)	Hu et al. (%)	Generated (%)
LR	0.00	0.00	0.00	0.00
DT	0.16	0.10	0.16	0.12
RF	0.20	0.18	0.19	0.18
SVM	0.00	0.02	0.00	0.02
MLP	0.00	0.00	0.00	0.00
CNN		0.00		0.00
LSTM		0.01		0.01
BiLSTM		0.06		0.06

As can be seen in Table 2, we achieve comparable attack results to five black-box classification algorithms used in Hu's proposed MALGAN. Although the TPR is not decreased completely to zero for SVM, 0.02% is enough to be ignored. For random forest and decision trees, those are quite different with the structure of neural networks, our proposed attack is able to decrease the TPR on generated adversarial examples to the range of 0.10% to 0.18% for both the training set and the test set. In addition, for the deep learning based three classifiers, the TPRs is also reduced to nearly zero, while the malware detection accuracy ratio on the original samples range from 94.96% to 95.39%.

We also plot the convergence curve of the TPR on the training set and the validation set during the training process of our work, with using random forest as the black-box classifier. The result is shown in Fig. 2.

As can be seen from the result, the TPR in our scheme decreases dramatically with the increasing of the training epoch. The overall performance of our scheme is as good as that in Hu's work, while the decreasing rate is greater than the opponent due to our design that reveal more of the semantic association of APIs.

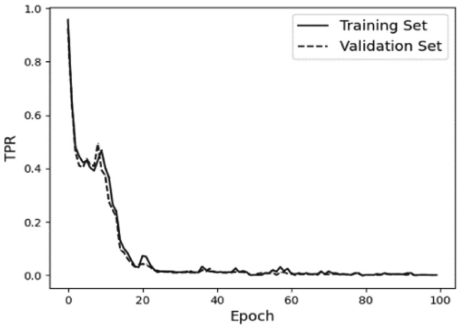

Fig. 2. The change of the true positive rate on the training set and the validation set over time.

4 Conclusions

In this paper, we study the problem of generating adversarial malware examples for black-box attacks. Traditional generative adversarial network based models are unable to compute gradient for their back-propagation due to the discrete output, which makes it difficult to update the weights of the neural network in the training process. Potential semantic features of APIs are usually ignored in existing schemes. A novel generative adversarial network based algorithm with API level word embedding method is proposed, which adopts CNN structure to train a substitute model, which is utilized to analyze the semantic association of composite API calls and sequential API calls. It can reveal the latent semantic features obtained by APIs more extensively. We employ a long short-term memory framework to generate antagonistic examples to enhance the utilization of semantic feature information between APIs. Experimental results show that the proposed scheme is efficient and effective.

References

1. Kai, Z., Shifei, D.: Advances in image super-resolution reconstruction. Comput. Eng. Appl. **53**, 29–35 (2017)
2. Lin, Y., Han, X., Xie, R., Liu, Z., Sun, M.: Knowledge Representation Learning: A Quantitative Review (2018). arXiv e-prints arXiv:1812.10901
3. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks (2013). arXiv preprint arXiv:1312.6199
4. Demetrio, L., Biggio, B., Lagorio, G., Roli, F., Armando, A.: Explaining vulnerabilities of deep learning to adversarial malware binaries (2019)
5. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples (2014). arXiv preprint arXiv:1412.6572
6. Hu, W., Tan, Y.: Generating adversarial malware examples for black-box attacks based on gan (2017). arXiv preprint arXiv:1702.05983
7. Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., Hsieh, C.-J.: Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, pp. 15–26 (2017)

8. Duarte-Garcia, H.L., Morales-Medina, C.D., Hernandez-Suarez, A., Sanchez-Perez, G., Toscano-Medina, K., Perez-Meana, H., Sanchez, V.: A Semi-supervised learning methodology for malware categorization using weighted word embeddings. In: 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), pp. 238–246. IEEE (2019)
9. Johnson, R., Zhang, T.: Supervised and semi-supervised text categorization using one-hot LSTM for region embeddings. Stat. 1050 (2016)
10. Du Peng, D.S.: A DGA domain name detection method based on deep learning models with mixed word embedding. J. Comput. Res. Dev. **57**, 433 (2020)
11. Zhang, Y., Gan, Z., Carin, L.: Generating text via adversarial training. In: NIPS workshop on Adversarial Training (2016)
12. Weigelt, S., Landhäußer, M., Blersch, M.: How to Prepare an API for Programming in Natural Language (2019)
13. Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A.: Practical black-box attacks against machine learning. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, pp. 506–519 (2017)
14. Papernot, N., McDaniel, P., Goodfellow, I.: Transferability in machine learning: from phenomena to black-box attacks using adversarial samples (2016). arXiv preprint arXiv:1605.07277
15. Martins, N., Cruz, J.M., Cruz, T., Abreu, P.H.J.I.A.: Adversarial machine learning applied to intrusion and malware scenarios: a systematic review. IEEE Access **8**, 35403–35419 (2020)
16. Wang, S., Zhou, W., Jiang, C.: A survey of word embeddings based on deep learning. Computing **102**(3), 717–740 (2019). https://doi.org/10.1007/s00607-019-00768-7
17. Wang, Z., Liu, Z., Chen, Z., Hu, H., Lian, S.: A neural virtual anchor synthesizer based on seq2seq and gan models. In: 2019 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct), pp. 233–236. IEEE (2019)