AVOIDANCE OF MALWARES IN LARGE SCALE NETWORK USING ENSEMBLE DEEP GENERATIVE ADVERSARIAL NETWORKS

K. Janani, R. Gunasundari*

Abstract

Artificial Intelligence (AI) has progressed considerably in recent years and now influences all parts of society and jobs. AI is beneficial to many fields, such as gaming, language processing, healthcare, production, education, and others. This tendency also affects the realm of cyber security, in which AI has been used in cyber space for both attack and defense. False alarms are a problem for end-users which disrupt business by delaying any essential response and generally damage efficiency. The fine-tuning process is a compromise between eliminating false alarms and maintaining the level of safety. In this paper, an Ensemble Deep Generative Adversarial Networks (EDGAN) is developing for the classification of threats in case of large network. The EDGAN is designed in such a way that it undergoes series of process to eliminate the threat in a novel way. The simulation is conducted to test the efficacy of the model and the results of simulation shows higher rate of security in classifying the instances than other methods.

Keywords: Generative Adversarial Networks, Cyber security, Deep Learning, Attacks

I INTRODUCTION

Malware identification is a major concern in the world of computer security nowadays, since many types of software give users an abundance of benefits, but they also carry some risk. Malicious samples are rapidly increasing, according to a recent study. Detecting malware has become increasingly difficult due to the sheer volume of samples [1].

Malware analysis and detection techniques have been examined extensively by a huge number of researchers. In the past, commercial antivirus programmes have relied on signature-based methods that require a database for storing the malware pattern data. Due to the fact that even tiny changes to malware can alter its signature, signature-based detection will continue to have serious limitations as more malware is developed that uses encryption, obfuscation, or packing to avoid detection [2,3,4].

The analysis necessitates simulating the malware operational environment, which can be problematic given the variety of malicious behaviours. Malware behaviour monitoring might take a lengthy period since some malicious actions can be undetected before an attack. Static analysis has the advantage of quickly detecting large malware samples. Static analysis is hampered by numerous encryption and obfuscation methods. Static analysis makes it harder to capture the malware features since attackers can make deliberate changes to malware.

A number of algorithms on machine learning [5,6,7] are now being researched for use in malware detection in order to address the issues listed above. However, for analysis and feature extractionrely on the domain knowledge. With these features, a machine learning model can be trained and a fresh file sample can be classified. However, the fact that malware is continually being generated, updated, and modified is a severe issue [8,9,10,11,12].

In this paper, an Ensemble Deep Generative Adversarial Networks (EDGAN) is developed for the classification of threats in case of large network. The EDGAN is designed in such a way that it undergoes series of process to eliminate the

Department of Computer Science,
Karpagam Academy of Higher Education, Coimbatore, Tamil Nadu, India
*Corresponding Author

threat in large scale Internet of Thing (IoT) network.

II RELATED WORKS

The first-time data mining was used to detect malware was by Schultz et al. [13]. Ripper [14] is a rule-based system for data mining that employs Naïve Bayes learning to find pattern features. It takes harmful code as input and classifies it. For data mining, Kolteret al. [15] found that employing n-gram features rather than non overlapping features produced better results. According to their findings, the decision tree is the most effective method for making a decision.

Instead, Saxe and Berlin [16] advocated using a neural network to separate malicious code from benign code. Their research includes calculating the entropy histogram via input binary data, where counting the total executions of contextual byte data and metadata extraction from the DLL imports and execution files. One by one, each of these four kinds of features is converted to a 256-dimensional vector. In a neural network, unidentified samples are categorised using feature vectors that have been learned. Their TPR is 95.2%, while their FPR is 0.10%.

There are also some fresh approaches to detect malware. There is an image-based malware categorization approach described by Nataraj et al. [17]. Malware data of binary type is converted into an image of gray scale type using a kNN classifier. It leverages global features from the image and hence the attacker makes local changes in order of avoiding them.

Using dynamic analysis and image feature processing, [18] compared two approaches to image feature processing. Using image features as a basis, the researchers were able to demonstrate that the method is scalable, accurate, and efficient. They also discovered that this new method works just as well with malware samples that have been compressed as well as those that have been uncompressed.

Unsupervised clustering learning using structured information was introduced by Kong and Yan [19] for malware classification. In order to cluster the samples of same family, it will use a distance matrix-based discriminating learning algorithm to extract the fine-grained features of the malware function call graph. After that, an ensemble classifier uses these malware distance pairs for categorization.

N-gram characteristics were introduced by Santos et al. [20] as a way of differentiating between malicious and benign software. K-nearest samples with the most similar N-gram characteristics are most effective in detecting unknown malware, according to their findings. Another way to discover malware is to look at the frequency with which an API is called. This is known as a -gram.

For this reason, some studies have begun to combine static and dynamic features in order to more correctly and efficiently perform malware detection. [21] A malware detection technique was proposed by Santos et al. [22] based on machine learning that used information from static and dynamic analyses of harmful code. SVM classifier has an accuracy of up to 96.6% in detecting malware. Studies showed that combining static and dynamic analysis yielded better results than doing it independently on their own.

Most of these machine learning-based malware detection technologies rely largely on expert expertise when it comes to the construction of their features. Human-designed features have numerous issues as malware evolves and grows, necessitating large amounts of effort and money to manually update them when new malware emerges. As a result, the goal of this research is to lower the feature engineering cost, while also extracting usable information from large amounts of raw data.

III PROPOSED METHOD

In this section, an EDGANclassifies the cyber security threats or malware in large scale Internet of Thing (IoT) network. The EDGAN detects the malware using binary classification, where it receives the raw data files as its input and the output is represented in the form of a discrimination probability, which indicates the likeliness of being a malware.

Therefore, to concretely attain the task of binary detection, the proposed EDGAN is split into two different segments. The former pre-process the sample is given as input data and extracts binary data form of the executable file. The generation of gray scale image is followed by the extraction of opcode sequence using a decompilation tool, followed by the metadata feature. The latter uses EDGAN to learn from the opcode sequence and grayscale image. The ensemble stacking is applied to integrate the outputs of two learning patterns namely the opcode sequence and grayscale image in order to obtain the predicted results.

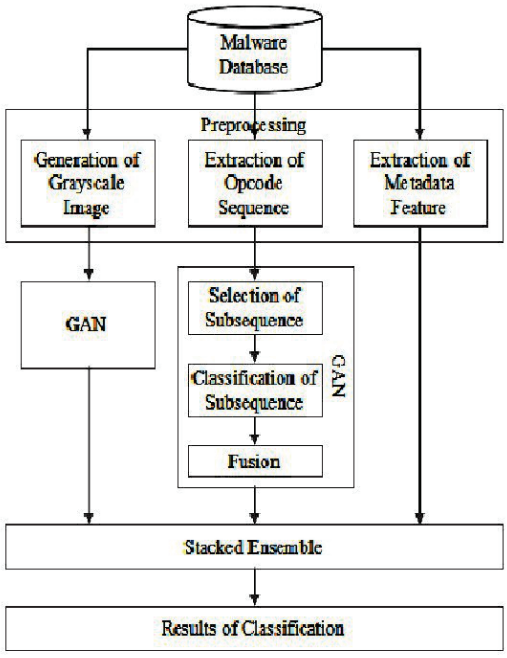

Figure 1: Proposed Malware Detection Model

Using the raw data, EDGAN learns three different feature sets: harmful file structure features from an image (grayscale type), pattern features of malicious code from the opcode sequence, and finally malicious code pattern features. In order to describe the global information, a metadata features on the feature sets reflects local pattern information. The following section goes into detail about the EDGAN design and how it detects threats.

3.1 Learning Malware via GAN

GAN networks are introduced in this section, along with a description of how malware structural features can be built using gray scale images learned by GAN. Using this technique, malware binaries can be seen as grayscale images, which reveal the structural raw fileproperties.

3.1.1 Generation of Grayscale Image

Raw data must be processed and turned into a format (image) before it can be used to create a virus grayscale image. A binary stream file is used as input data instead of an executable file. To use the binary stream file, simply converted every four bits to a hexadecimal value. The hexadecimal number range is 0 to 16, and the grey value of a 256-pixel is formed by adding two hexadecimal integers of the same length. This straightforward mapping operation can turn the original RGB data into a grayscale image. For each pixel grey level, the binary stream gets segmented over each 8-bit data and then ordered consecutively to create the appropriate grey image.

3.1.2 Data Preprocessing

The study preprocesses the image to meet the GAN input data criteria. The input picture data is the same size when GAN performs image classification. The image data should, in general, be the same width and height. It is purely for the sake of saving time during the convolution process. Because executive files come in a variety of file formats, the sizes of grayscale images vary widely.

When it comes to normalisation, we apply an image scaling method called the bilinear interpolation algorithm. In order to determine the pixel value, it uses the 4-neighborhood pixels. This method outperforms nearest neighbour interpolation in terms of quality. The more information the GAN receives from a larger normalised image, where the detection result will be improvedon the network. As a result, the normalised grayscale image size has been determined to be 32x32.

3.2 Opcode SequenceLearning

The focus is on the opcode sequence, which the study uses to discover harmful sequence characteristics and patterns using EDGAN. Decompiled files yield a list of opcode sequences. These sequences reflect the logic of the code and the execution of the programme in the executive files themselves. As a result, GAN can extract from them malicious code sequence traits that correspond to highly malicious behaviour.

3.3 Extraction of Opcode Sequence

The study extracts the opcode sequence and then learns it from the raw executive files. In order to resolve malware into assembly instructions, a popular decompilation and debugging tool is used. The opcode sequences length is affected by the size of the opcode set during opcode sequence extraction. More types of opcodes will be accepted if the opcode set is expanded. Given the large amount of noise data and the difficulty of learning with GAN caused by a long opcode sequence, we must keep the opcode set size within a suitable range and only include the valid information. Since all decompiled.asm files are treated as text, we use vocabularies to represent the instructions within them. The low-frequency vocabulary is then filtered out using frequency statistics.

The study classifies each vocabulary frequency using a random forests model, which may assign a value to each feature importance based on its frequency. When selecting vocabulary, we look for ones that place emphasis on a certain quality. After the digitisation of the opcode sequences, the GAN is applied over it. The study uses one-hot encoding to attain the digitisation, which is mapping transformation for the generation of a sparse vector.

3.4 Learning Long Sequence by GAN

To understand long-term dependencies and context, GAN uses loss to flow backwards via larger timestamps. GAN makes and enforces rules for the Constant Error Carousel (CEC) state. To detect malware, a GAN network will learn from the opcode sequence.

If the sequence is long, a GAN has a hard time training properly, even if it can collect time series. The size of the executive file determines the size of extracted opcode sequence. Ramnit malware samples, for instance, had a sequence length of 36,000. When the input sequence length exceeds 200, however, GAN performance rapidly degrades. So, it is vital to figure out how to use a GAN network to process extremely long sequences.

When working with GAN networks on very long sequences, Truncating and Padding (TAP) is a simple yet effective technique. When a sequence is too long, TAP trims it down to the shortest possible length and then uses a predetermined identifier to pad the ends. It quick and easy, but it sacrifices a lot of data in the process because of the truncation. A technique known as Truncated GAN through time limits the greatest distance an error can travel by adding a time window constraint. Since only the nodes inside of the window are updated, no nodes outside of the window are affected by error propagation or gradient calculation. Because the BGANTT calculation cost increases if the GAN network is too long, but it improves computing efficiency by sacrificing a small portion of accuracy. Additionally, abbreviated BGANTT is better suited for online learning due

to its ability to swiftly adjust to the newly created portion of a lengthy sequence.

The study first partitions an opcode sequence into several subsequences, with each subsequence length equal to the truncated BGANTT window length. Once this is done, we just perform a full BGANTT on each subsequence, which is the same as performing only a truncated BGANTT on the entire sequence. Most importantly, this makes it possible for GAN to work on multiple sequences at the same time.

3.5 Subsequence Selection and Fusion

As a result of the above subsequence training technique, the GAN network receives subsequence as input and generates an output for each subsequence in turn. GAN concluded that this subsequent event is malevolent based on how likely it is. However, particularly for malware samples, subsequences might have a lot of noise in them. There are many subsequences of this type of sample that are safe because many malware programmers simply insert a malicious code inside benign to begin with. Therefore, malware subsequences must be well-maintained. To clean these subsequences and give a dataset to GAN, we devised a subsequence selection algorithm.

To finish subsequence selection, we use the GAN network by itself, without using any extra models. A GAN is used in binary classification tasks to determine whether a class is negative or positive.

3.6 Stacking Ensemble

EDGAN recovers some malware metadatabecause GANs can get a global featuresof malware data and it captures the local features and metadata features as well. The study easily discovers metadata elements and starting address of byte file, as well as sizes, rows and length ofdecompiled file by looking at decompiled files.This model uses the original training data to create training dataset and then exploit the output. Stacking ensemble method aggregates the outcomes of heterogeneous learners using GAN model. The study combines the three components by stacking ensemble to get a final detection result.

IV RESULTS AND DISCUSSIONS

In this section, the validation of EDGAN is conducted to check the efficacy of the entire model. The model is validated against various models like Ensemble CNN-RNN (ECR), Ensemble CNN-MLP (ECM) and Ensemble CNN-DBN (ECD). The simulations are conducted in terms of validating its accuracy, precision, recall and F-measure.

The datasets consist of 40000 samples out of which 21000 samples are of malware one and remaining ones are benign type. The simulation is conducted on i5 11th generation processor running on a primary memory of 16GB and graphical processing unit of 16GB.

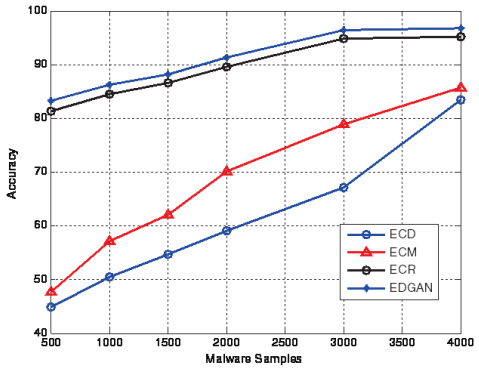

Figure 2: Accuracy

Figure 2 shows the results of accuracy between the proposed EDGAN and existing ECR, ECM and ECD. The result of simulation shows that the proposed EDGAN model is effective in classifying the features of malwares over a large IoT network.

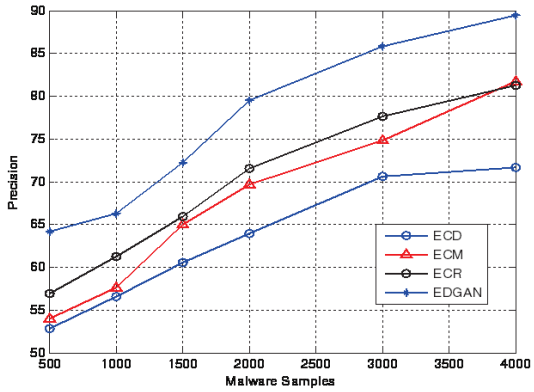

Figure 3: Precision

Figure 3 shows the results of precision between the proposed EDGAN and existing ECR, ECM and ECD. The result of simulation shows that the proposed EDGAN model has higher level of precision in classification process. The result thus shows that the correctly predicted positive observations are high using EDGANs than other classifiers.

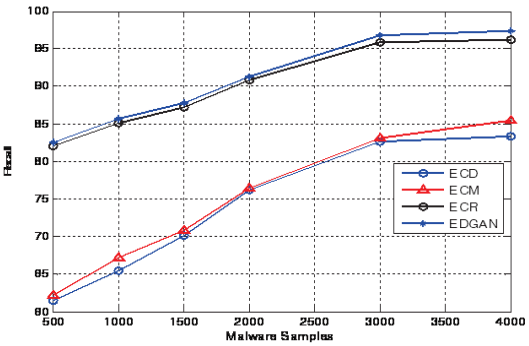

Figure 4: Recall

Figure 4 shows the results of recall between the proposed EDGAN and existing ECR, ECM and ECD. The result of simulation shows that the proposed EDGAN model has higher level of recall rate than other methods, where the total number of retrieved instances is accurate to attain the improved rate. It is seen that the EDGAN model has higher rate of true positives than other parameters.

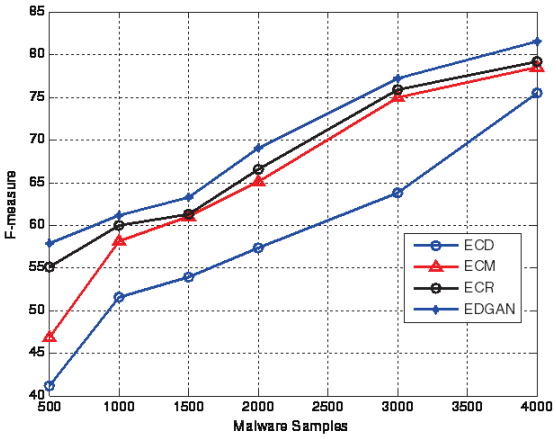

Figure 5: F-measure

Figure 5 shows the results of F-measure between the proposed EDGAN and existing ECR, ECM and ECD. The result of simulation shows that the proposed EDGAN model has higher level of F-measure than other methods in case of classifying truly the instances. Even in the presence of uneven distribution of classes, the proposed method obtains higher rate of weighted average on precision and recall values. Since the rate of true positive instances is high, the occurrence of errors tends to get reduced.

V CONCLUSIONS

In this paper, the malwares in a large scale IoT network are detected using EDGAN model that uses stacked ensemble of GAN to learn from varied types of feature extracted from input data. These features include opcode sequence and grayscale image, thereby fusing them to find the predicted output. The experiment is conducted on 40000 input samples out of which 1-fold is utilised for validation. The results of validation show an improved accuracy of 99% with reduced FPR of lesser than 0.1%. The comparisons with conventional method show that the proposed method is accurate in predicting the malwares than other methods.

REFERENCES

[1] Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., &Venkatraman, S. (2019). Robust

intelligent malware detection using deep learning. IEEE Access, 7, 46717-46738.

[2] Karbab, E. B., Debbabi, M., Derhab, A., &Mouheb, D. (2018). MalDozer: Automatic framework for android malware detection using deep learning. Digital Investigation, 24, S48-S59.

[3] Alzaylaee, M. K., Yerima, S. Y., &Sezer, S. (2020). DL-Droid: Deep learning based android malware detection using real devices. Computers & Security, 89, 101663.

[4] Kim, T., Kang, B., Rho, M., Sezer, S., &Im, E. G. (2018). A multimodal deep learning method for android malware detection using various features. IEEE Transactions on Information Forensics and Security, 14(3), 773-788.

[5] Ren, Z., Wu, H., Ning, Q., Hussain, I., & Chen, B. (2020). End-to-end malware detection for android IoT devices using deep learning. Ad Hoc Networks, 101, 102098.

[6] Wang, Z., Liu, Q., & Chi, Y. (2020). Review of android malware detection based on deep learning. IEEE Access, 8, 181102-181126.

[7] Feng, R., Chen, S., Xie, X., Meng, G., Lin, S. W., & Liu, Y. (2020). A performance-sensitive malware detection system using deep learning on mobile devices. IEEE Transactions on Information Forensics and Security, 16, 1563-1578.

[8] Pektaş, A., &Acarman, T. (2020). Deep learning for effective Android malware detection using API call graph embeddings. Soft Computing, 24(2), 1027-1043.

[9] Cho, Y. B. (2018). The Malware Detection Using Deep Learning based R-CNN. Journal of Digital Contents Society, 19(6), 1177-1183.

[10] Azmoodeh, A., Dehghantanha, A., &Choo, K. K. R. (2018). Robust malware detection for internet of (battlefield) things devices using deep eigenspace learning. IEEE transactions on sustainable computing, 4(1), 88-95.

[11] Feng, J., Shen, L., Chen, Z., Wang, Y., & Li, H. (2020). A two-layer deep learning method for android malware detection using network traffic. IEEE Access, 8, 125786-125796.

[12] Venkatraman, S., Alazab, M., &Vinayakumar, R. (2019). A hybrid deep learning image-based analysis for effective malware detection. Journal of Information Security and Applications, 47, 377-389.

[13] Schultz, M. G., Eskin, E., Zadok, F., &Stolfo, S. J. (2000, May). Data mining methods for detection of new malicious executables. In Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001 (pp. 38-49). IEEE.

[14] Cohen, W. W. (1995). Fast effective rule induction. In Machine learning proceedings 1995 (pp. 115-123). Morgan Kaufmann.

[15] Kolter, J. Z., &Maloof, M. A. (2006). Learning to detect and classify malicious executables in the wild. Journal of Machine Learning Research, 7(12).

[16] Saxe, J., & Berlin, K. (2015, October). Deep neural network based malware detection using two dimensional binary program features. In 2015 10th International Conference on Malicious and Unwanted Software (MALWARE) (pp. 11-20). IEEE.

[17] Nataraj, L., Karthikeyan, S., Jacob, G., &Manjunath, B. S. (2011, July). Malware images: visualization and

automatic classification. In Proceedings of the 8th international symposium on visualization for cyber security (pp. 1-7).

[18] Nataraj, L., Yegneswaran, V., Porras, P., & Zhang, J. (2011, October). A comparative assessment of malware classification using binary texture analysis and dynamic analysis. In Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence (pp. 21-30).

[19] Kong, D., & Yan, G. (2013, August). Discriminant malware distance learning on structural information for automated malware classification. In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 1357-1365).

[20] Santos, I., Penya, Y. K., Devesa, J., &Bringas, P. G. (2009). N-grams-based File Signatures for Malware Detection. ICEIS (2), 9, 317-320.

[21] Shabtai, A., Moskovitch, R., Feher, C., Dolev, S., & Elovici, Y. (2012). Detecting unknown malicious code by applying classification techniques on opcode patterns. Security Informatics, 1(1), 1-22.

[22] Santos, I., Devesa, J., Brezo, F., Nieves, J., &Bringas, P. G. (2013). Opem: A static-dynamic approach for machine-learning-based malware detection. In International joint conference CISIS'12-ICEUTE´ 12-SOCO´ 12 special sessions (pp. 271-280). Springer, Berlin, Heidelberg.