# Towards an Efficient Approach Using Graph-Based Evolutionary Algorithm for IoT Botnet Detection

Quoc-Dung Ngo[1], Huy-Trung Nguyen[2*]
[1]Posts and Telecommunications Institute of Technology, Hanoi, 10000, Vietnam
[2]People's Security Academy, Hanoi, 10000, Vietnam
E-mails: dungnq@ptit.edu.vn, huytrung.nguyen.hvan@gmail.com

**Keywords:** IoT botnet, evolutionary algorithm, IoT security, PSI graph

**Received:** September 1, 2021

*In recent years, a large number of Internet of Things devices are used in life, many of which are vulnerable to attacks from a security perspective. Botnet malware is one of the main threats to IoT devices. Hence detection of IoT botnet is one of the most important challenge for IoT devices. This paper proposes an IoT botnet detection approach based on PSI graph data combine with evolutionary algorithm-based technique. In recent years, a large number of Internet of Things devices are used in life, many of which are vulnerable to attacks from a security perspective. Botnet malware is one of the main threats to IoT devices. Hence detection of IoT botnets is one of the most important challenges for IoT devices. In the paper, a IoT botnet detection approach based on PSI graph analysis by using the evolutionary algorithm-based technique. It applies bacterial evolution algorithm (BEA) in the training process on PSI graph multi-architecture IoT Botnet data to detect IoT Botnet. The PSI graphs were extracted from executable files and transform into vectors to feed into the classical machine learning classifiers. The result of the classifiers is then combine using soft voting method with BEA. The proposed method has achieved good experimental results (i.e., Accuracy at 95.30%, F1 at 96.15%). The approach also achieves a relatively low false-positive rate at 4.59%.*

*Povzetek: Predlagan je pristop za odkrivanje botnetov IoT z uporabo PSI grafov in evolucijskega algoritma.*

## 1 Introduction

The fourth industrial revolution explicitly resulted in the boundless growing scale of the Internet of things globally. For instance, the number of connected devices was forecasted by Statista [1] to reach the milestone of 75.4 billion in the next 5 years. This means that IoT application and devices have been increasing their presence in every daily activity. Nevertheless, this popularization has exposed myriads of important information security matters namely violation of data breach, privacy, etc. In these problems, malicious code has emerged in popularity. There are several categories of malwares but ransomware and botnet are the two types having unique behaviors.

A botnet is a group of internet-connected devices infected by malware that allow cyber-criminals to control them. Botnets carry out many malicious behaviors such as data theft, unauthorized access, credentials leaks, unauthorized access, data theft and distributed denial-of-service (DDoS) [2], [3]. Along with the immense growing of Internet of things application, there have been countless number of botnet attacks originated from IoT devices. For instance, the legendary DDoS attack that turn half of the internet down for several hours in 2016 was launched by Mirai botnet from about 1.2 million infected devices [4]. Besides, successors of Mirai known as Reaper and Hajime also infect IoT devices then turn them into bots for DDoS purposes.

To alleviate the destruction of IoT botnet attacks, security researchers have been frequently examining on state-of-art malware detection techniques. There is some noticeable effort on fitting rule-based methods in analyzing abnormal traffic [5], leveraging machine learning based classifier on engineered sets of features such as opcodes [6], processor contexts [7], etc. From the point of view of a security researcher, malware detection technique can be divided into two categories: static and dynamic analysis.

Dynamic analysis [8] requires a separated and supervised environment to executing then monitoring the suspicious executables to record its footprints including system calls, network traffic and register values. The most challenging aspect of dynamic analysis is the process of designing and constructing an appropriate virtual machine that has the capabilities of luring the malware to active all

*Corresponding Author

its characteristic. Furthermore, IoT malware can operate on multiple architecture namely SPARC, ARM, MIPS, x86, PowerPC. Hence, virtualizing an environment that satisfies all the action conditions of the IoT botnet is expensive. In other words, the most critical drawback when applying dynamic analysis for IoT malware is the technical difficulties in building a suitable environment for the fullest activation of each malicious samples.

Contrarily, static analysis [9] leveraged a wide range of techniques to identify the malicious characteristic without execution. Evaluated features in static analysis include printable strings information, grayscale images, control flow graph, opcodes, etc. The plus points of this method are not only limited to the ability of depicting the structure and functionality of multi-arch malware but also included the reduction of computational resource since it does not require any supervised environment. In addition, static analysis ensure the safety of the system as well as enforcing the ethical constraints [10] because of the lack of sample execution. Although static analysis has its own drawbacks in handling obfuscated files, there are many proposals to solve this problem with a satisfactory result. In brief, static analysis is a feasible solution in detection IoT malware [11].

In the related study which nominated PSI graph [9] as a novel feature in detecting IoT botnet, Nguyen et al. only focused on the overall structure of the PSI graph. According to the proposed hypothesis, PSI graph contains a huge number of executables paths of an executable file, including both normal and abnormal paths. However, graph exploration is an expensive operation according to the number of vertices as well as the interconnection between them. Therefore, if it is possible to efficiently extract the necessary route which depict the characteristic of the original PSI graph, the computational complexity of the entire botnet detection process would be greatly reduced.

The paper expands the research results of [9] combined with an evolutionary algorithm into the ensemble process aimed towards an effective method in detecting IoT botnets. In summary, the key contributions of this work are:

(1) Proposing an approach in IoT botnet detection model that bases on graph data combine with evolutionary algorithm.

(2) Experimenting the proposed method with large IoT Botnet datasets result in higher accuracy than normal voting method for ensembling weak learner.

In addition to the presented content, the rest of the paper is structured as follows: Section 2 presents related works in the research field; then Section 3 describes in detail the proposed method; then describes the empirical data set and evaluation criteria; Finally, the analysis and evaluation of the experimental results and conclusions.

## 2 Related works

The process of analyzing malware samples can be categorized into static and dynamic analysis. In general, static analysis can depict the structure and maliciousness without the need of executing the malware sample [9]. On the other hand, dynamic analysis aims to investigate the behavior of a malware by activating its sample in a supervised environment [8]. Furthermore, there is a combination inherited the advantages of both dynamic and static analysis techniques which was known as hybrid analysis [12].

There is a featured characteristic of IoT botnets which known as the diversity of operating architectures such as x86, MIPS, ARM, PowerPC [13]. In addition, according to the requirements of dynamic analysis method, it would be costly to simulate an entire environment of a single architecture to perform dynamic analysis techniques. Therefore, when it comes to investigate IoT botnets, static analysis methods allow researchers to solve multi-architecture issues and mitigate the limitations of dynamic analysis.

In recent years, the number and complexity as well as the notorious level of malwares have been sky-rocketed. While signature-based classifier [14] were almost useless in detecting novel types of malwares, security researchers often leverage Machine Learning algorithms as an alternate yet effective solution to deal with unseen malwares [15]. Besides, evolutionary algorithms and their variants are another considerable technique to deal with the rapid mutation of unseen malwares [16], [17], [18], [19].

An overview of general application of evolutionary algorithms on rule-based system was described by Shafiq et al. in [17]. This comparative study leveraged static features from executables then picked five well-known evolutionary algorithms including XSC, GAssist-ADI, UCS, SLAVE, GAssist-Intervalar and benchmarking these against another five non-evolutionary algorithms in classifying malicious executables. The experiment dataset consisted of 11,786 Window PE in which 1,447 PE were benign and 10,339 malicious PE from VH Heavens Virus Collection which was later divided into eight major classes. The accuracy of these evolutionary-based models is promising with the lowest value equaled to 0.95, mostly the accuracy of them ranged from 0.98 to 0.99. However, by considering all suggested four performance metrics: (1) classification accuracy, (2) number of rules, (3) comprehensibility of the rules, (4) processing overheads, this paper stated that non-evolutionary rule learning algorithms clearly outperform evolutionary rule learning ones for every performance metrics. Besides, the processing costs and comprehension of evolutionary rule learning algorithms can be improved by combining some concepts from non-evolutionary rule learning algorithms.

Another combination from Rafique et al. leveraged dynamic analysis technique and evolutionary algorithms to automatically classify malware families and their polymorphic variants [18]. By using protocol-aware modeling to handle formal protocol traffic and state-space modeling to handle unknown protocol traffic, this solution was able to extract features from network behaviors which

were collected from PCAP file after executing and monitoring malware samples in a supervised environment. Next, in the evaluation phase, four evolutionary algorithms (GAssist-ADI, SLAVE, UCS, XCS) were selected to compare against four old-school non-evolutionary classifiers (C4.5, C-SVM, kNN, Naïve Bayes). The experimental dataset contained 6000 binaries of 20 recent malware families, most of them were obtained from MALICIA dataset. Obtained results demonstrated the poor performance of evolutionary classifiers, except UCS, which dominated all the rests with roughly 99.7% of accuracy on the entire dataset and 85.28% per malware family. Another notable downside of examined evolutionary classifiers was the testing time which mostly slower than the non-evolutionary candidates. This paper presented state-space modeling which was a promising technique in extracting unknown protocol network behaviors. However, this approach still needs to be examined further and compared to others network feature extractors. In addition, the applied evolutionary algorithms in this research were used without either any modifications or improvements from their original proposal.

A noticeable research of Manavi et al. [16] took advantages of static analysis technique to extract OpCodes from executables then utilized an evolutionary-based classifier to detect malicious samples according to a predefined list of 9 malware families. In this work, after the disassembling phase, a graph of OpCode was constructed for the executable file. Then the proposed evolutionary classifier would create the most similar graph to the target. Finally, by applying the Euclidean distance fitness function, the most similar graph of the results would determine the maliciousness of the sample. The experimental dataset of this research was quite diversity since it included 3 sub datasets: 1600 malwares and 1600 benigns from VX Heaven's dataset, 4000 apks with the ratio of 50-50 between benign-malware from Drebin dataset, 2042 samples including 9 different malware families from Microsoft Kaggle malware classification challenge.

In the first two dataset, the experimented results of the proposed method were as good as the related study of Hashemi et al. [20] and Santos et al. [21] which considered OpCode as a feature. Besides, in the third dataset of Microsoft, the evolutionary classifier outperformed the other but the accuracy was limited to 87.67%. Nevertheless, this research took advantages of static feature but did not suggest any in-depth solutions to deal with obfuscated malwares. In addition, the runtime analysis of the proposed evolutionary classifier was omitted. Last but not least, although the dataset was quite varied, it was still lack of botnet, especially IoT botnet.

An efficient complement between genetic algorithms and neural nets called Genetic Neural Network - GNN in botnet detection was proposed in [22], this paper combined the genetic algorithm's significant global search capabilities with the precise local search factor of the backpropagation to provide forward neural nets to improve the initial weight of the neural nets. The performance of the proposed GNN with 7 extracted features from network flow data proved that GNN was a promising model with better accuracy (95.7%) than either back propagation neural nets or genetic algorithm. However, this work did not specify either any deterministic method for feature selection or any description of the experimental dataset.

Nevertheless, to the best of our knowledge, there have not been any proposed researchs that aim to detect IoT botnet leverage the evolutionary algorithm and the novel PSI graph [9] as a feature.

# 3 Methodology

We enhanced the performance of weak classifiers in dectecting IoT Botnet based on PSI-graphs generated from ELF files by apply the bacterial evolutionary algortihm in the ensemble process of these classifers. This section will explain our approach in detail including psi graphs extraction process and the performance of evolutionary voting process in detecting IoT botnet on these graphs.

## 3.1 Overview

The main components of our method are presented in figure 1. There are 3 main processes in our method: extracting PSI graphs from ELF files, training weak classifiers and applying bacterial evolutionary algorithm in the ensemble process of weak classifiers.

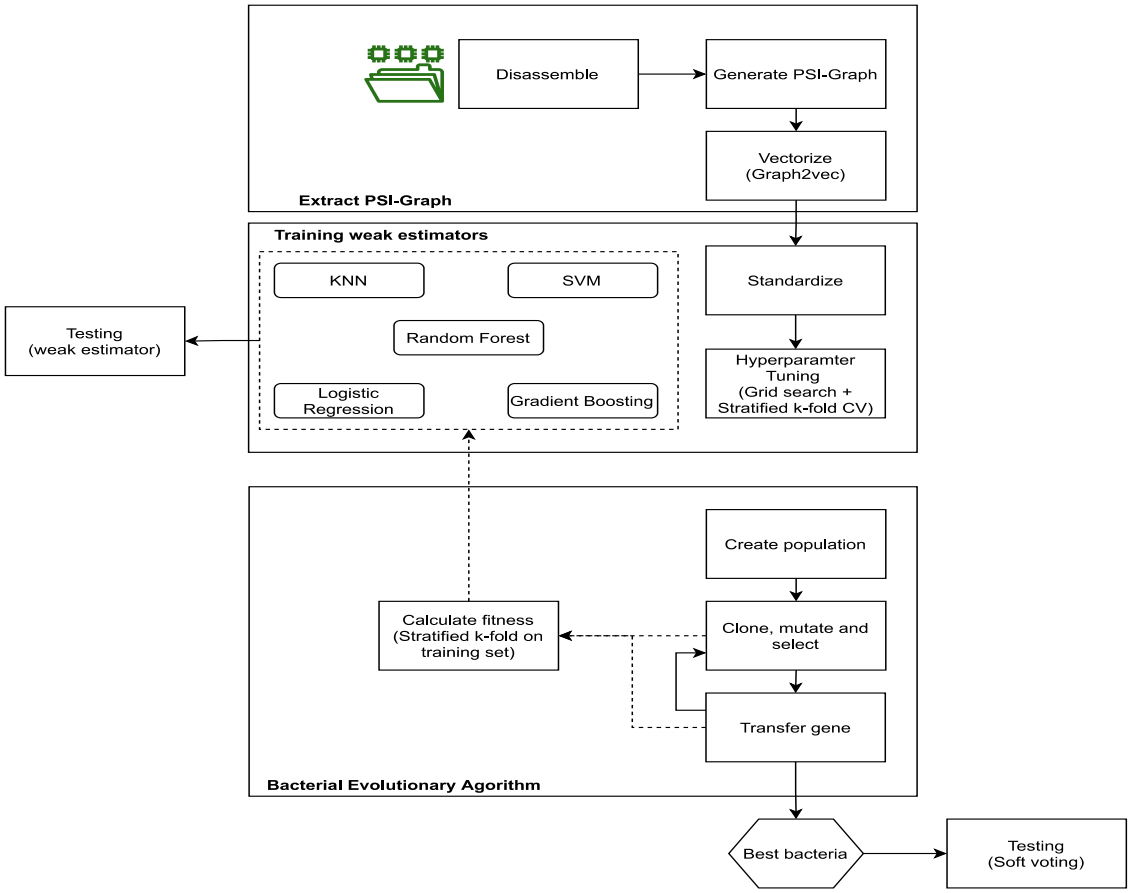


Figure 1: The overview of proposed method.

Firstly, we execute the ELF files of malware and benign samples to generate PSI graph from these files. After that, we preprocess the graph using graph2vec [23] algorithm embedding vector of similar structured graphs in near feature space. After that, we use classical machine learning classifiers to classify the graph vectors generated from graph2vec. We then perform different voting strategies for the ensemble process of weak classifiers including hard voting and soft voting. The bacterial evolutionary algorithm is applied in the soft voting phase to improve voting process accuracy. Finally, we compare the classification result of each classifier and ensemble method to estimate decide whether the method is effective or not.

## 3.2 PSI graph extraction

Printable String Information (PSI) is a set of string usually contain important sematic information that can reflect the attacker's intent. PSI was used in static analysis method to identify ELF malware files. In this research, the author doesn't give enough attention to the linkages of the PSI element which give more information about the context and could greatly improve the result. In our work, we collected our PSI graph dataset generated by Nguyen et al. [9] from our previous research and inherited the way to represent IoT executable file with PSI graph.

Definition 1: PSI graph is a directed graph defined as $G(E,V)$, where $V$ is a set of vertices called PSI elements and $E$ is a set of edges which represents for function calls.

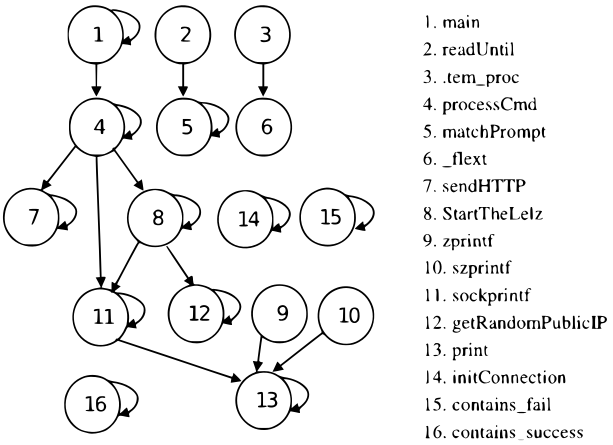


Figure 2: An example of PSI graph.

## 3.3 Traing weak classifiers

After obtaining PSI-Graph, we have to convert the graph data into input for machine learning classfiers. Using graph2vec algorithm, we turn our PSI-graph data into vectors where graph with similar structure are embedded in near feature space. Then, we standardize the feature vector for better converging process by scale down feature

that have large value to make all feature stay in the same range of value. The standardize process is applied using the formular:

$$x_{standardized} = \frac{x - \mu}{\sigma}$$

where $\mu$, $\sigma$ is the mean and the standard deviation of original data, respectively.

We then feed the standardize graph vectors data into classical machine learning classifiers for classification process to detect IoT Botnet samples. K-nearest neighbor (KNN), Support Vector Machine (SVM), Logistic Regression, Gradient Boosting and Random Forest are the chosen machine learning algorithm for individual classifiers. The classifiers are well known for their effectiveness in classificatin problem and have been used by many researchers for the intrusion detection problem. In the training phase, we use use k-fold cross-validation combining with hyperparameter tuning with a grid of parameter values. The model with best hyperparameters is trained and tested on testing set to evaluate the performance. To combine the prediction from different classifiers, an ensemble method is required. Here we used 2 different voting method: hard voting and soft voting. Hard voting is combining all the prediction of the classifiers and made the final prediction base on the majority of the vote while soft voting calculate the prediction probability of each classifer's prediction and the final prediction is the largest summed probability from classifiers. Hard voting method is pretty straight forward. Ensemble method usually result in higher accuracy for classification task because it concludes the prediction from each involved classifier, that's why individual classifiers are referred to as "weak classifiers".

### 3.4 Bacterial evolutionary algorithm

Bacterial evolutionary algorithm (BEA) is a kind of evolutionary algorithm base on bacteroa, and its properties are similar to those of the GA's (Genetic Algorithm): it is also a global optimization technique, and provides a near-optimal, approximate solution for the problem. It is useful even if the objective function is non-linear, non-continuous, multimodal or high-dimensional. BEA does not use the derivatives of the objective function, thus it does not cause a problem, if they are not known or do not exist [24].

In our approach, the BEA algorithm is used in the ensemble phase to improve the soft voting strategy and was depicted in figure 1. The BEA algorithm has 3 main steps: generate population; clone, mutate and select; gene transfer. The details about BEA algorith in our approach is descibe as follow:

Generate population: we create the initial population of the algorithm with number of population N_POP = 100. Each chomosome in the population is one bacteria which contains N_GENES = 5, these 5 genes represent the weights of 5 weak classifiers in the soft voting proces. Genes have the value in the range of Gauss destribution with mean value = 1 and standard deviation value = 0.2. The use of Gauss destribution will make the weight contain value with the range close to 1. This avoid the situation when maybe one classifier has very large weights and the others one is too small for comparison.

Clone, mutate and select: In the beginning of 1 generation, each bacteria create 20 clones of itself N_CLONE = 20. At a given time, one random gene is selected from all the clones and these clones will mutate by changing the chosen gene into a random value that belong the distribution mentioned above. After that, we calculate the fitness score of each clone from the average accuracy in 10-fold on training set with the weight of the clone. If the clone has higher fitness score than the original then it will be selected to replace all the other clonel. The mutating process repeat 10 times N_MUTATE = 10 which guarantee that all the gene will be mutated for N_GENE = 5.

Gene transfer: After the mutating process, all the bacteria are sorted by fitness score. The population is seperated in 2 halves. We then select 2 random bacteria, one for the upper half and one for the lower half. One or several random gene from upper half bacteria will be copied to the lower half bacteria. The population is then reorganized for all the lower half bacteria will have the chance to join the upper half and the upper half will always contain quality bacteria. This process repeats 50 times N_TRANS = 50. This is the end of a generation.

Cloning and gene transfer process is repeated in 10 generation N_GENERATIONS = 10. When we perform the experiment, we realize the algorithm has fast convergence rate so we don't need any further local optimization algorithm (memetic algorithm).

## 4 Experimental and evaluation

This section gives the information about our experimental enviroment and results, the evaluation metrics, dataset used and discussion.

### 4.1 Dataset description

We inherit the PSI graph dataset from previous researches on PSI graph. This dataset consists of 10010 PSI graph samples with fairly balance botnet and benign samples including 3845 IoT botnet samples and 6165 benign samples. IoT botnet samples belong to two typical botnet families which are Gagyft and Mirai and other less popular malware such as Tsunami, Aida, as shown in figure 3.

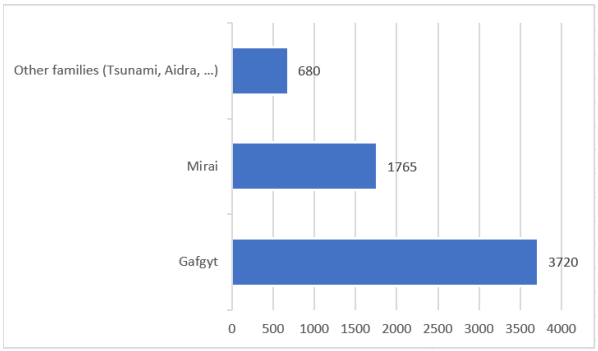


Figure 3: Distribution of the botnet sample in the dataset.

Samples from the dataset come from multiple CPU architectures including ARM, MIPS, Intel 80386, x86-64, PowerPC, Motorola, Spark, and SuperH. The number of IoT botnet belong to each CPU architecture is describe in figure 4.

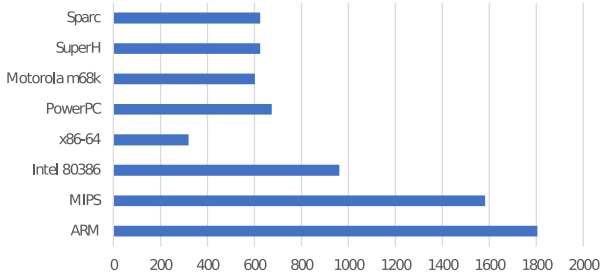


Figure 4: Number of botnet in each CPU architecture in the dataset.

The following configuration was used when we conduct the experiment: Ubuntu 16.04LTS 64-bit, Intel Xeon, 8Gb RAM. The experiment is built in Python language.

## 4.2 Evaluation metric and results

The following terms are used to evaluate the effectiveness of the proposed method.

- True positive (TP): the number of malicious samples that are properly recognised

- True negative (TN) is the number of benign programs that are correctly recognised

- False positive (FP) is the number of benign programs that are incorrectly identified

- False negative (FN) is the number of malicious programs that are incorrectly

The following metrics are used to evaluate the precision-efficiency of the proposed method:

- True positive rate (TPR) or Sensitivity, Recall is the number of predicted malware samples correctly classified as malicious divided by total malware. This metric shows the probability of detecting malware samples.

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN}$$

- False positive rate (FPR) or Fall-out: the number of predicted benign samples falsely marked as malicious divided by total benign samples. This metric shows the probability of false alarm.

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN}$$

- Accuracy (ACC): the ratio of the number of corrected samples to the number of both malware and benign samples. However, accuracy is not trustful in imbalanced dataset.

$$ACC = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + TN + FP + FN}$$

- F1-score is the harmonic mean of Precision and Recall (TPR). F1-score is a combining metric to estimate the entire model performance and is defined as follow:

$$F_1 = 2\frac{Precision.Recall}{Precision + Recall}$$

We ran the experiment training weak classifer, perform ensemble process and improve the ensemble process using bacterial evolutionary algorithm, as shown in table 1.

| Classifier | Accuracy (%) | Precision (%) | Recall (%) | F1 (%) | FPR (%) | Average 10-fold CV accuracy (on training set) |
|---|---|---|---|---|---|---|
| Best weak estimator (KNN) | 94.54 | 95.17 | 96.00 | 95.58 | 7.80 | 94.25 |
| Hard voting | 95.07 | 96.96 | 94.97 | 95.96 | 4.77 | - |
| Soft voting (Equal weights) | 95.14 | 96.76 | 95.29 | 96.02 | 5.11 | 94.82 |
| Soft voting (BAE) | 95.30 | 97.08 | 95.24 | 96.15 | 4.59 | 95.08 |

Table 1. Experimental results of the proposed method with different classifiers.

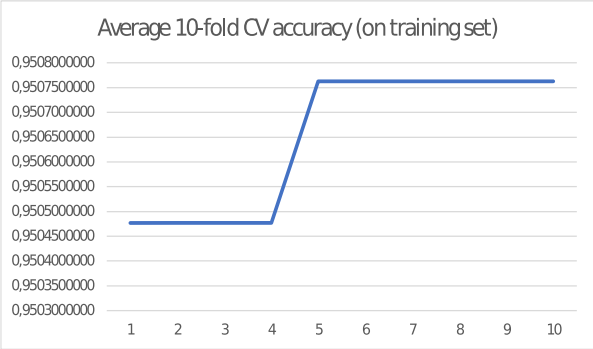


Figure 5: Bacterial evolution algorithm training accuracy for soft voting.

The result show that the best individual classifer achieve 94.54 % accuracy in detecting IoT botnet is KNN. The reason KNN can achieve the highest classification rate among others classifiers is when converting graph data into vector we used graph2vec. In graph2vec, graphs with similar structure usually have vectors embedded near each other's therefore the KNN algorithm can group these graphs more easily which result in higher classification rate. The result is also showing the soft voting process after the BEA algorithm perform better than normal hard voting and soft voting method with high accuracy of 95.30% accuracy and 4.59% FPR. Figure 5 also show that applying evolutionary-based BEA in soft voting process does increase the overall performance of the model.

The author in [16] also represent malware as graph using opCode graph and evolutionary algorithm for classification process. The result from our study produce significantly higher detection rate than the work introduce in Manavi et al. [16] (95.30% compare to 85.8% ~ 87.67%). Haddadpajouh et al. [25] used deep recurrent neural network to classify ARM-based IoT Botnet.Our results reach equivalent accuracy with the research in [25] (94% accuracy), but in their research they used smaller dataset and only focus on ARM-based IoT Botnet. The same thing can be said when compare with study by Su et al. [26] using malware image and CNN (94% accuracy). The result has shown that applying evolutionary algorithm in the process of training on PSI graph data make could improve the process of detecting IoT Botnet.

# 5 Conclusion and future works

In this research, we apply bacterial evolution algorithm (BEA) in the training process on PSI graph multi-architecture IoT Botnet data to detect IoT Botnet. The PSI graphs were extracted from executable files and transform into vectors to feed into the classical machine learning classifiers. The result of the classifiers is then combine using soft voting method with BEA. The result show that our method has achieved higher accuracy to the other research using the graph as input while perfoming on much larger dataset. In the future, we hope to improve our graph method and some modification to the algorithm to achieve higher accuracy for the model.

# References

[1] Statista Research Department., "Internet of Things-Number of connected devices worldwide 2015-2025," 2019. https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

[2] "Al-Hadhrami, Y. and Hussain, F.K., 2021. DDoS attacks in IoT networks: a comprehensive systematic literature review. World Wide Web, 24(3), pp.971-1001." [Online]. Available: https://doi.org/10.1007/s11280-020-00855-2

[3] Sérgio S.C. Silva , Rodrigo M.P. Silva , Raquel C.G. Pinto , Ronaldo M. Salles, "Botnets: A survey," *J. Comput. Netw. Elsevier*, vol. 57, no. 2, pp. 378–403, 2013. [Online]. Available: https://doi.org/10.1016/j.comnet.2012.07.021

[4] Bertino, E. and Islam, N., "Botnets and internet of things security," *Computer*, vol. 50, no. 2, pp. 76–79, 2017. [Online]. Available: https://doi.org/10.1109/mc.2017.62

[5] "Ozawa, S., Ban, T., Hashimoto, N., Nakazato, J. and Shimamura, J., 2020. A study of IoT malware activities using association rule learning for darknet sensor data. International Journal of Information Security, 19(1), pp.83-92.". [Online]. Available: https://doi.org/10.1007/s10207-019-00439-w

[6] "Peters, W., Dehghantanha, A., Parizi, R.M. and Srivastava, G., 2020. A comparison of state-of-the-art machine learning models for OpCode-based IoT malware detection. In Handbook of Big Data Privacy (pp. 109-120). Springer, Cham.". [Online]. Available: https://doi.org/10.1007/978-3-030-38557-6_6

[7] "Takase, H., Kobayashi, R., Kato, M. and Ohmura, R., 2020. A prototype implementation and evaluation of the malware detection mechanism for IoT devices using the processor information. International Journal of Information Security, 19(1), pp.71-81.". [Online]. Available: https://doi.org/10.1007/s10207-019-00437-y

[8] Le, H.V. and Ngo, Q.D., "V-Sandbox for Dynamic Analysis IoT Botnet," *IEEE Access*, vol. 8, pp. 145768–145786, 2020. [Online]. Available: https://doi.org/10.1109/access.2020.3014891

[9] Nguyen, H.T., Ngo, Q.D. and Le, V.H., ., "A novel graph-based approach for IoT botnet detection," *Int. J. Inf. Secur.*, vol. 19, no. 5, pp. 567–577, 2020. [Online]. Available: https://doi.org/10.1007/s10207-019-00475-6

[10] Ma, W., Duan, P., Liu, S., Gu, G. and Liu, J.C., "Shadow attacks: automatically evading system-call-behavior based malware detection," *J. Comput. Virol.*, vol. 8, no. 1, pp. 1–13, 2012. [Online].

Available: https://doi.org/10.1007/s11416-011-0157-5

[11] "Quoc-Dung Ngo, Huy-Trung Nguyen, et al., A survey of IoT malware and detection methods based on static features, ICT Express, Volume 6, Issue 4, pp. 280-286, 2020." . [Online]. Available: https://doi.org/10.1016/j.icte.2020.04.005

[12] "Ngo, Q.D., Nguyen, H.T., Tran, H.A. and Nguyen, D.H., 2021, January. IoT Botnet detection based on the integration of static and dynamic vector features. In 2020 IEEE Eighth International Conference on Communications and Electronics (ICCE) (pp. 540-545). IEEE." . [Online]. Available: https://doi.org/10.1109/icce48956.2021.9352145

[13] "Xiao, L., Wan, X., Lu, X., Zhang, Y. and Wu, D., 2018. IoT security techniques based on machine learning: How do IoT devices use AI to enhance security?. IEEE Signal Processing Magazine, 35(5), pp.41-49." . [Online]. Available: https://doi.org/10.1109/msp.2018.2825478

[14] "Borello, J.M. and Mé, L., 2008. Code obfuscation techniques for metamorphic viruses. Journal in Computer Virology, 4(3), pp.211-220." . [Online]. Available: https://doi.org/10.1007/s11416-008-0084-2

[15] "Souri, A. and Hosseini, R., 2018. A state-of-the-art survey of malware detection approaches using data mining techniques. Human-centric Computing and Information Sciences, 8(1), pp.1-22." . [Online]. Available: https://doi.org/10.1186/s13673-018-0125-x

[16] Manavi, F. and Hamzeh, A., "A new approach for malware detection based on evolutionary algorithm," 2019, pp. 1619–1624. [Online]. Available: https://doi.org/10.1145/3319619.3326811

[17] Shafiq, M.Z., Tabish, S.M. and Farooq, M., "On the appropriateness of evolutionary rule learning algorithms for malware detection," 2009, pp. 2609–2616. [Online]. Available: https://doi.org/10.1145/1570256.1570370

[18] Rafique, M.Z., Chen, P., Huygens, C. and Joosen, W., "Evolutionary algorithms for classification of malware families through different network behaviors," 2014, pp. 1167–1174. [Online]. Available: https://doi.org/10.1145/2576768.2598238

[19] "Lysenko, S., Bobrovnikova, K., Shchuka, R. and Savenko, O., 2020, May. A cyberattacks detection technique based on evolutionary algorithms. In 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT) (pp. 127-132). IEEE.". [Online]. Available: https://doi.org/10.1109/dessert50317.2020.9125016

[20] "Hashemi, H., Azmoodeh, A., Hamzeh, A. and Hashemi, S., 2017. Graph embedding as a new approach for unknown malware detection. Journal of Computer Virology and Hacking Techniques, 13(3), pp.153-166.". [Online]. Available: https://doi.org/10.1007/s11416-016-0278-y

[21] Santos, I., Brezo, F., Nieves, J., Penya, Y.K., Sanz, B., Laorden, C. and Bringas, P.G., "Idea: Opcode-sequence-based malware detection," 2010, pp. 35–43. [Online]. Available: https://doi.org/10.1007/978-3-642-11747-3_3

[22] Yin, C., Awlla, A.H., Yin, Z. and Wang, J., "Botnet detection based on genetic neural network," *Int. J. Secur. Its Appl.*, vol. 9, no. 11, pp. 97–104, 2015. [Online]. Available: https://doi.org/10.14257/ijsia.2015.9.11.10

[23] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal, "graph2vec: Learning distributed representations of graphs," *ArXiv Prepr. ArXiv170705005*, 2017.

[24] F. Hatwágner and A. Horváth, "Maintaining genetic diversity in bacterial evolutionary algorithm," *Ann. Univ Sci Bp. Sec Comp*, vol. 37, pp. 175–194, 2012.

[25] H. HaddadPajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, "A deep recurrent neural network based approach for internet of things malware threat hunting," *Future Gener. Comput. Syst.*, vol. 85, pp. 88–96, 2018. [Online]. Available: https://doi.org/10.1016/j.future.2018.03.007

[26] J. Su, D. V. Vasconcellos, S. Prasad, D. Sgandurra, Y. Feng, and K. Sakurai, "Lightweight classification of IoT malware based on image recognition," 2018, vol. 2, pp. 664–669. [Online]. Available: https://doi.org/10.1109/COMPSAC.2018.10315