A Graph-based Approach for IoT Botnet Detection Using Reinforcement Learning

Quoc-Dung Ngo[1,*], Huy-Trung Nguyen[2,4,*], Hoang-Long Pham[1], Hoang Hanh-Nhan Ngo[3], Doan-Hieu Nguyen[2] , Cong-Minh Dinh[2] and Xuan-Hanh Vu[5]

[1] Posts and Telecommunications Institute of Technology, Vietnam
[2] People's Security Academy, Hanoi, Vietnam.
[3] Academy of Cryptography Techniques, Hanoi, Vietnam
[4] Institute of Information Technology, Vietnam Academy of Science and Technology, Hanoi, Vietnam
[5] Faculty of Information Technology, Hanoi Open University, Vietnam
[*] dungnq@ptit.edu.vn, huytrung.nguyen.hvan@gmail.com

Abstract. In recent years, the strong development of the Internet of Things (IoT) has resulted in an unprecedented rise in the number and variation of IoT malware. One of the most common malware on IoT devices today is the bot malware, popularly referred to as Botnets. This type of malware has been used to carry out many network attacks such as a denial of service, fraud, user data theft, etc. Over time, IoT botnets have been crafted with a higher level of complexity and danger, so existing researches based on non-graph structure features typically incur a high computational overhead and do not completely explore the relationship between behaviors, which can expose further characters of a botnet. Recently, to cope with these limitations, the approach of detecting IoT botnet based on structured-graph features using machine learning has gained its popularity. However, most of the existing graph-based approach has not fully explored the malicious behavior characteristics of IoT botnet yet. In this paper, we propose an effective method to classify executables as either IoT malware or IoT benign based on mining graph feature. Firstly, generating PSI-walks from PSI graphs via Reinforcement learning to form a dataset of PSI-walks before feeding it into a shallow LSTM network for IoT botnet detection. The methodology is verified using a dataset consisting of 6165 IoT botnet samples and 3845 benign samples. Experimental results indicate that the proposed approach can efficiently detect IoT botnets with achieving high accuracy as 97.14% and a low false-positive rate of 2.7%.

Keywords: IoT botnet detection, PSI-graph, Static analysis, Cyber security, Reinforcement-learning.

1 Introduction

The fourth industrial revolution known as Industry 4.0 is backed by the outstanding development in physics, digitalization, biology, and having a great impact on other fields as well as the economic platform of every country. This trend explicitly causes

the immense growing scale of the Internet of Things at a global scope. For instance, a recent report [1] forecasted that the estimated number of connected devices would be roughly 75.4 billion by 2025. This means that the usage of IoT systems is increasing dramatically all over the world. IoT applications have been applied in almost every aspect of daily life. However, this popularization poses a lot of critical information security issues including privacy, authentication, data storing and management, etc. Among these topics, malware attacks have been receiving a great deal of public attention. There have been a whole host of malware attacks which originate from IoT devices. For example, the biggest DDoS attack launched by Mirai botnet from 1.2 million infected devices [2] in the last quarter of 2016 had rocketed an enormous bandwidth capacity of 1.5 Tbps [3]. Furthermore, two successors of Mirai known as Hajime and Reaper also infected vulnerable IoT devices at first then used them as bots to launch several massive DDoS attacks [4]. As a result, to mitigate this kind of attack, researchers have been continuously investigating malware detection techniques. For instance, detecting botnet attacks by applying association rule learning to analyzing darknet traffic [5], and classifying between malware and benign by applying Machine Learning techniques on specific set of features of executables namely opcodes [6], processor information [7] etc.,. From the perspective of malware analysts, there are two approaches in malware detection known as static and dynamic analysis.

Dynamic analysis [8] is equivalent to detect abnormal behaviors while executing and monitoring executables based on a list of features namely system calls, network traffic, and register value in a supervised environment. The most critical part in the process of dynamic analysis is building a suitable environment that allows the malware to activate all of their malicious behaviors. Nevertheless, IoT malware can operate with tricky activating conditions on multiple architectures such as MIPS, ARM, SPARC, PowerPC. Therefore, it is costly to virtualize an entire compatible system that satisfies all the activation conditions of IoT malware. In other words, the critical drawbacks of dynamic analysis are technical difficulties in constructing a suitable environment for the full activation of malicious samples and an expensive monitoring process. On the other hand, static analysis [9] includes a wide range of techniques to detect malicious samples without executing them. Static approach relies on the extraction of known features from executables such as operation code sequences (opcode), printable string information, grayscale image and control flow graph (CFG), etc. The merits of this method are the ability to illustrate the structure and functionality of multi-architecture malware. Moreover, static analysis can analyze malicious files without executing, which reduces the cost of computational resources and ensures the safety of the system as well as . Therefore, the static analysis method is considered as a compatible solution in the problem of detecting IoT malware [9]. Although static analysis has limitations in dealing with encrypted or obfuscated files, there are several approaches to address this issue and these methods achieved satisfactory results. In the previous study, we proposed a graph-based structure feature that was effective in the problem of detecting IoT botnet malware, called PSI-graph [10]. In this study, H.T Nguyen et al. found that the malicious behavior of IoT botnets often took place according to a certain infectious process known as the law of action. On

that basis, the authors came up with a graphical feature called PSI-graph efficiency in the problem of detecting IoT botnet. However, the study only focuses on the overall structure of the PSI graph and does not explore the information on the path and subgraph in the PSI graph. According to the research hypothesis, the PSI graph contains many executable paths of the executable file, including the path of maliciousness and also the normal path. Therefore, based on the characteristic of PSI graphs, if it is possible to trim and eliminate unnecessary paths, then it is possible to keep the routes with behavioral characteristics of malicious code or those with the greatest probability of malicious behavior only. Therefore, this method will contribute to the improvement of efficiency in IoT botnet detection. Thus, the issue that plays an important role here is how to determine the path in the graph that reveals the most highlighted characteristics of IoT botnet. In other words, this is the problem of path-finding in a graph.

Many existing algorithms [11], [12] was designed to handle the path-finding tasks in a graph, but those algorithms often require a defined and fixed environment. With the constantly evolving characteristics of malware, it is costly to fulfill these requirements, so those algorithms are not feasible in this case. In recent years, an effective method of finding and predicting behaviors that satisfy a certain purpose, called reinforcement learning. Reinforcement learning (RL) [13] is a branch of machine learning-based methods that implements dynamic learning by adjusting actions when interacting with the environment. Therefore, it maximizes the performance of systems based on continuous feedback to maximize a reward. Therefore, to reduce the complexity in graph analysis for IoT botnet malware detection problems based on structure-graph features, in this work, we propose an effective method that utilizes RL in finding the suspicious path that has many characteristics of a IoT botnet. These paths will be the most effective representation of the IoT botnet malware detection problem. The fundamental contributions of this work are summarized as follow:

- To the best of our knowledge, our research is one of the first approaches to IoT botnet detection with a combination of RL and graph mining techniques.
- We evaluate our approach on large datasets achieving high accuracy as 97.14% and a low false-positive rate of 2.7% for IoT botnets detection.

To achieve these contributions, the rest of this paper is organized as follows: Section 2 lists several significant researches which also leverages reinforcement learning in malware detection. Section 3 describes our proposed method in detail as well as the evaluations. Finally, we conclude the paper in Section 4.

2 Related Works

According to the characteristics of features which are extracted from malicious executables, malware detection techniques can be classified into dynamic analysis and static analysis.

Dynamic analysis focuses on inspecting the run-time behaviors of malwares by activating malicious samples in an appropriate environment [14]. On the contrary, static analysis includes a whole host of techniques to depict the structure as well as the malicious characteristics of malwares without any execution [10]. Generally, IoT devices

adopt various processor architectures which are costly to virtualize precisely. Therefore, static analysis is often leveraged to deal with multi-architecture issues and tackle down the drawbacks of dynamic analysis in detecting IoT botnet. However, the malware is growing immensely in both number and complexity, so researchers have come up with machine learning based detection methods [15], [16] because of its natural advancement in detecting novel samples, fast processing, real-time predictions, improved detection accuracy and much more. Generally, machine learning has three basic learning paradigms namely supervised learning, unsupervised learning and reinforcement learning [17]. Supervised and unsupervised learning algorithms [18] such as DBN, SAE, Decision Tree and SVM are often utilized as classifiers which directly take part in the process of detecting malicious samples. In comparison, reinforcement learning is leveraged swiftly as an optimization method to maximize the accuracy and minimize the false-positive rate in recent malware detection proposals. There are a number of recent proposals which leverage reinforcement learning in distinct steps of the detection process such as [19], [20], [21].

Blount et al. proposed a method along with its proof of concepts for adaptive rule-based malware detection employing learning classifier systems, which combines a rule-based expert system with evolutionary algorithm based reinforcement learning [19]. This proposal suggested and implemented a self-training adaptive malware detection system which dynamically evolves detection rules. As described, the main responsibility of a Q-Learning Reinforcement Technique in this paper is to evolve rules of a LCS system to identify malwares based on function name and library name which were extracted in the Import Address Table of PE files. The best detection rate of this method is 96.1% with False Alarm Rate of 0.02. The drawbacks of this modified LCS system are the inability to process packed PE samples and PE files which have corrupt or missing IATs.

While most malware detectors simply leverage the power of Machine Learning into their classifier [22], [23] without any significant optimization, Cangshuaiwu et al. [21] proposed gym-plus, which based on an unique solution that includes utilizing Reinforcement Learning to create new malwares then labeled them as training data for their ML classifier. This creative architecture takes advantage of a malware generating system which is built on top of Reinforcement Learning algorithms and accumulates new malware manipulating functions into the proposed feature sets of gym-malware to increase the evasion rate of incoming generated malwares. The improvements made by Cangshuai Wu et al. were implemented on 4 different Reinforcement Learning agents namely Acer, DQN, Double DQN, Sarsa and have the best evasion rate of 47.5% . Therefore, after being trained with these samples, the detection rate of the LightGBM model achieves its best value at 93.5% compared to 15.75% before training with newly generated malwares. However, this solution has not proved its efficiency against novel IoT botnet.

Reinforcement Learning can also be leveraged to indirectly speeding up the process of malware detection. Liang Xiao et al. formulates a static malware detection game and follows up with a malware detection scheme with Reinforcement Learning [24]. There are a number of Reinforcement Learning algorithms modifications that have been applied in this research. Firstly, Q-learning algorithm is applied to achieve

the optimal offloading rate without knowing the trace generation and radio bandwidth model of the neighboring IoT devices [24]. This scheme improves the detection accuracy by 40% and reduces the detection latency by 15% and increases the utility of the mobile devices by 47% compared with the benchmark offloading strategy in [15]. Secondly, by utilizing both the real defense and virtual experiences generated by the Dyna architecture, Dyna-Q-based malware detection scheme as proposed in [24] not only improves learning performance but also reduces the detection latency by 30% and increases the accuracy up to 18% compared to the detection with Q-learning. In addition, PDS-based malware detection scheme as developed in [24] leverages the known radio channel model to accelerate the learning speed as well as utilizing Q-learning to study the remaining unknown state space. Therefore, the detection accuracy was increased by 25% compared with the Dyna-Q based scheme in a network consisting of 200 mobile devices. Conversely, this scheme and its modifications only leverage reinforcement learning techniques during the data transferring process, not directly into the classification task.

Detecting IoT malwares has gained its popularity. Several feasible solutions have taken advantage of diverse Machine Learning techniques in combination with distinct feature sets to accomplish the malware classification tasks. Beside, several studies of malware detection have shown that methods based on structured features (i.e graph) are more effective than using unstructured features in malware detection [25]. A typical example is the study of H.T Nguyen et al. [10] presented a lightweight method for detecting IoT botnet based on a graphical feature called PSI-graph. This feature shows the effectiveness of solving the multi-architecture problem of IoT botnet, and the experimental results on the data set of more than 11,000 samples achieved high accuracy at 98.7%. However, graphs are often complex and diverse. It involves restoring a significant number of subgraphs or paths as features, and the processing of those features can be very time-consuming. As the above studies, it seems that none of the above studies solved the problem of graph mining to extract the optimal and effective features in IoT botnet detection. Therefore, in this work, we propose an approach using reinforcement learning in graphical structured data mining, bringing high efficiency and low time-consuming when compared with other methods based on PSI-graph for detecting IoT botnet [10], [26]. Method details are presented in section 3.

3 Proposed Method

We propose an effective method to classify execution files into IoT malware or IoT benign based on graph feature mining. The flow of this method is generating PSI-walks from PSI graphs via Reinforcement learning to form a dataset of PSI-walks before feeding it into a shallow Long Short Term Memory (LSTM) network for IoT botnet detection. In this section, we present the structure of our proposed method including the main components as well as the preprocessing steps. Our proposed method consists of three main steps as described in Figure 1. First of all, we collect PSI graph dataset generated by H.Trung et al. [10] and feed it into a reinforcement learning block. Reinforcement learning considers each PSI graph as an environment

and uses agents to interact with it and find an optimal policy with maximum expected return, then the algorithm generates a PSI-walk which best represents the features of an executable. Next, PSI-walk data is augmented so as to increase the total number of samples. This stage is done by processing self-loops in each node of the PSI-walk. Finally, we apply a shallow LSTM network which is described later to the augmented dataset for classification tasks.

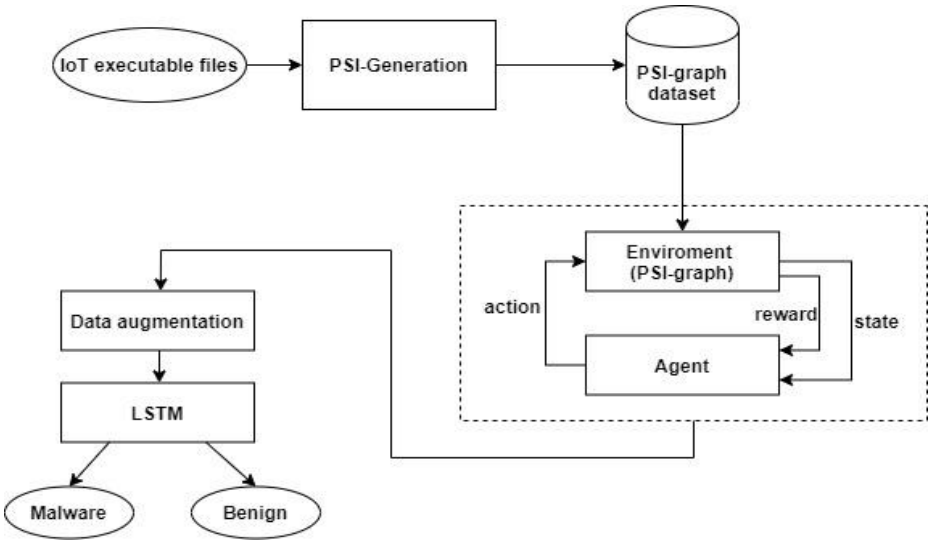

Fig. 1. Overview of proposed method.

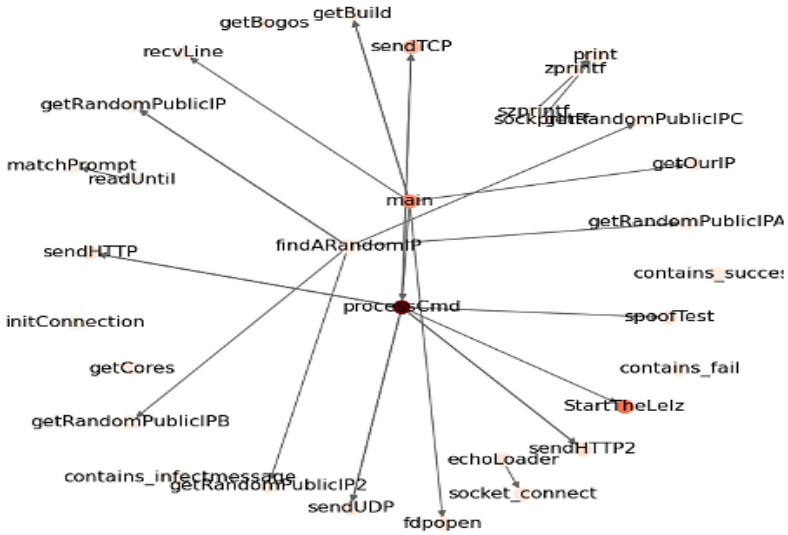

Fig. 2. An example of a simple PSI graph in our dataset.

3.1 PSI-walk

We have inherited the approach to represent IoT executable files with PSI-graphs [10]. However, the authors in [10], [26] focus on the structure of the whole PSI graph or a subset of it called the PSI-rooted subgraph. These approaches can achieve a good classification benchmark, but the used features do not figure out the most significant characteristics of the IoT botnet. In this paper, we focus on a different way to process PSI graphs by extracting the most informative walk in this type of graph. That means some walks in the PSI graph of IoT botnet are more valuable than the others. Therefore, these walks can represent the executable with its highlighted characteristics at its best .

Definition 1: PSI-walk is a directed walk defined as a finite sequence of edges $(e_1, e_2, \dots, e_{m-1})$ which joins a sequence of vertices $(v_1, v_2, \dots, v_m)$ such that e_i connects v_i and v_{i+1}. Whereas $(v_1, v_2, \dots, v_m)$ is the vertex sequence of the directed walk and is the way we store a PSI-walk in our dataset.

Obviously, in an executable, there are many good walks which can show the feature of that ELF (Executable and Linkable Format). We do not explicitly point out which walk is the best or the most significant. An optimal or near-optimal walk is enough for us to achieve a good result in the classification task.

3.2 Reinforcement learning in PSI-Walks generation

In order to find the expected PSI-walk of a PSI graph, we apply Q-learning [13] – a simple reinforcement learning algorithm. Q-learning is an off-policy reinforcement learning algorithm that seeks to find the best action to take given the current state. More specifically, Q-learning seeks to learn a policy that maximizes the total reward. In order to solve the problem of PSI-walk, we consider our model as a robot finding the most optimal walk to achieve as high reward as possible. Therefore, main components of the inforcement learning model are defined as below.

- *Agent is* the robot standing in one vertex of the PSI-graph. Agent takes actions by travelling to a neighbor vertex. In the end, by following the optimal found policy, agent generates the PSI-walk.
- *Action is* a set of all possible moves the agent can make. Therefore, the action space is all vertices in PSI-graph. However, we control that agent can only take valid actions by moving to a neighbor vertex of the current vertex. Note that, to avoid stucking in one vertex's self-loop, the algorithm does not allow the agent to move into its self-loop
- *Environment* is the world through which the agent moves, and which responds to the agent. This includes a PSI graph and a mechanism to take the agent's current vertex and the vertex it wants to move to and return the agent's reward and its next state.
- *State* is the current situation which includes the current vertex and a list of visited edges. When the environment returns the agent's next state, the state is updat-ed by changing the current node and appending the edge to the list of visited edges.

- *Reward* is how we measure the success or failure of an agent's action in a given state. We define some rules below to give each action a weight of I portance as well as the reward.

To some extent, an important function in a program often calls many other functions. For example, the function "processCmd" of BASHLITE source code is in charge of receiving command input from the main function and depends on that command input, the function triggers attack by calling "sendTCP", "sendUDP", v.v. Moreover, the bigger a function is, the more external library functions it usually calls. Therefore, the weight of each vertex is depending on the out-degree of that vertex.

Definition 2: Outdegree of a vertex v in graph $G = (V, E),\ (v \in V)$ is defined as the number of tail ends adjacent to a vertex and is denoted $deg^+(v)$

- a vertex with $deg^+(v) = 0$ is called a sink, since it is the end of each of its incoming arrows. This is the situation of such functions as "printf".
- The degree sum formula states that:

$$\sum_{v \in V} deg^+(v) = |E|$$

However, the PSI-graph is a directed multigraph, so a vertex may have all of its out-degree coming from its self-loops. Self-loops are not as much informative as connections to other vertices. Therefore, a vertex with a lot of self-loops will receive less reward than one with few of self-loops, and a vertex whose edges are all loops will receive nothing. Moreover, a vertex's reward must be as big as its out-degree and its real out-degree (eliminating self-loops).

In summary, given the number of self-loops of a vertex v $loops(v)$, moving from vertex u to vertex v in PSI-graph will receive a reward of R, where:

$$R = \begin{cases} 2 * deg^+(v) - loops(v) & if\ deg^+(v) \neq loops(v) \\ 1 & if\ deg^+(v) = loops(v) \end{cases}$$

Additionally, due to name convention, a vertex whose name starts with "_" or "__" may not be one of the main function of a program, so these vertices only receive a reward of 1 like the case $deg^+(v) = loops(v)$ above. Besides, moving to a vertex having self-loops is more promising than moving to one having no self-loops. Therefore, a vertex with self-loops will have a reward of $R' = 2 * R$.

Finally, when travelling in PSI-graph, agent may go into some cycles. In order to break cycle at some points and try some other walks, we reduce the reward of a visited edge. It means that moving to a vertex that agent has already visited that edge will receive only a reward of $R' = \left\lfloor \frac{R}{2^{m_e}} \right\rfloor$. Where m_e is the number of times agent repeat visiting that egde e. The formula means that the longer staying in a cycle, the smaller reward the agent receives. We apply Q-learning algorithm with epsilon-greedy algorithm to each PSI-graph with a number of episodes of 1000. When the algorithm ends iterating, using the learned Q table, we get the optimal PSI-walk. As we state earlier, we do not travel into self-loops. As a result, to show these self-loops, we duplicate vertices having self-loops in the final PSI-walk.

3.3 Data augmentation

One problem of the generated PSI-walks is that many PSI-graphs can generate the same PSI-walk, therefore, the dataset shrinks. A small number of samples of PSI-walk in dataset may influence the performance of machine learning classifiers. A small dataset can be prone to overfitting or data mismatch problem. In our case, a lot of vertices is just found in only one PSI-walk. That can cause data distribution of training set is too much different from development/testing set.

Besides, self-loops are not processed completedly in previous reinforcement learning model. A program can be inside a loop for multiple times before breaking it and do something else. For example, IoT botnet will try to connect to C&C server and only continue after a connection is accepted.

To solve those problems above, we apply data augmentation to the PSI-walk dataset. Data augmentation helps to improve the performance of classification problems. Algorithm 1 takes each PSI-walk and generates an augmented PSI-walk by randomly repeating a vertex having self-loops. In our case, we limit the maximum number of times repeating is r.

Algorithm 1: $\boldsymbol{Random\_augment(W, r)}$

Input $W = [v_i]$: list of vertices in PSI-walk
r: maximum number of loops
Output $W_a = [v_i']$: PSI-walk after data augmentation

```
Initialization: W_a = ∅, prev = None
for each v ∈ W do
    W_a := W_a ∪ {v}
    if v = prev then
        for i = 0 to random(0, r) do
            W_a := W_a ∪ {v}
    prev := v
return W_a
```

By applying the data augmentation algorithm described above, we managed to make our dataset fifth times bigger. That can prevent overfitting as well as data mismatch in the classification task.

3.4 Classification

As noted previously, PSI-walk is a sequence of vertices. Therefore, PSI-walk can be regarded as a sequence of words and the problem of malware classification can be considered as sentiment analysis. Since vertex data is text data, words in sequence, we can use a LSTM - advanced version of Recurrent neural network (RNN) to build a powerful model that does not only consider the individual words, but the order they appear in. It is reasonable because PSI-walk is a directed walk.

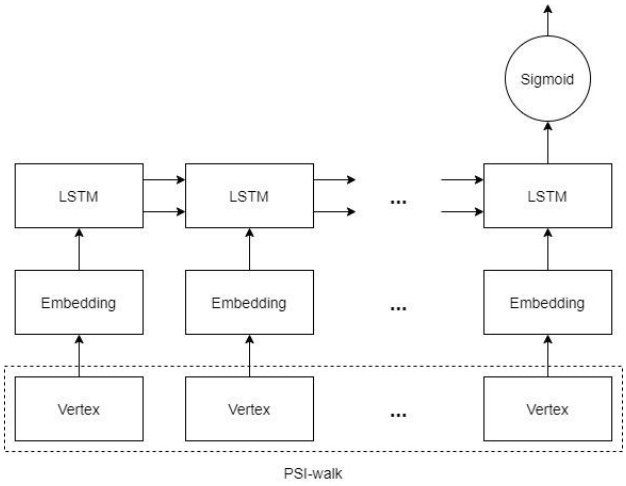

Fig. 3. LSTM model's architecture

In this work, we utilize the shallow many-to-one LSTM model, as shown in Figure 3. First, vertices have to be preprocessed so that we can get them into the network. We use embedding layers to encode each word with a vector. Then, these word embedding vectors are fed into LSTM and trained as a binary classification task. The output layer contains only one neuron whose output is the probability of being malware or benign. We also apply dropout technique to avoid overfitting.

3.5 Experimental result

In this section, we evaluate the proposed method by experiment. We inherit the multi-architecture PSI-graph dataset from previous research of PSI-graph. This dataset contains 10010 PSI-graph samples including 6165 IoT botnet samples and 3845 samples. Experiment is conducted with a desktop with the following configuration: Windows 10 Pro 64-bit, Intel Core i5, 8Gb RAM. Source codes are written in Python language. We divide the dataset into three sections including training set, test set, and validation set at the rate of 70%, 15%, and 15%, respectively. Validation set are used to evaluate the performance of model during hyperparameter tuning. After that, we use the final model to predict on the testing set.

a. Augmented corpus

The Reinforcement learning model extracts PSI-walk from PSI-graph dataset. It is obvious that multiple IoT botnet samples which belong to the same family or are cross-compiled version of the same botnet have the same sequence of PSI-walk. The reason is that all these botnets have a similar source code. Tables 2 show the number of PSI-walks generated by Reinforcement learning model before and after data augmentation. In classification task, we only consider binary classfication problem, so we group three malware classes into one malware class.

Table 1. Comparation of number of samples before and after data augmention

	Malware	**Benign**	**Total**
Before	1703	2857	4560
After	8350	10734	19084

From Table 1, we can see that data augmentation helps to generate new dataset approximately 5 times bigger than the original one. The augmented PSI-walk is just a modification of loops of the original PSI-walk so that it can represent well PSI-graph's features. Table 2 below shows some information about length of PSI-walk after augmented. Note that PSI-graphs having cycle are limited to travel in 50 steps for each episode of Reinforcement learning.

Table 2. Length of PSI-walks

	Malware	**Benign**
Minimum	2	2
Maximum	157	166
Average	14	19

Table 3. Result of LSTM model

Metrics	**LSTM**
Accuracy (%)	97.14
FPR (%)	2.70
ROC AUC (%)	97.11

b. Evaluation metrics

The following terms are used to evaluate the precision-efficiency of the proposed method: True positive (TP) is the number of predicted malware samples correctly classified as malware; True negative (TN) is the number of predicted benign samples correctly classified as benign; False positive (FP) is the number of predicted malware samples incorrectly tagged as benign; False negative (FN) is the number of predicted benign samples incorrectly tagged as malware. The following metrics are used to evaluate the precision-efficiency of the proposed method: FPR = FP / (FP + TN); ACC = (TP + TN) / (TP + TN + FP + FN)

c. Result and evaluation

Table 3 shows detailed result of LSTM model in IoT botnet detection. The experimental results show that the proposed method has a satisfactory benchmark. The model has both high detecion rate and low false-alarm rate of 2.70 %. Therefore, the model is robust to have a high ROC AUC score.

Table 4. A comparative summary of some malware detecion approach on IoT devices.

Methods	**Accuracy (%)**	**FPR (%)**
Ours	97.1	2.7
PSI-rooted subgraph based [26]	97.3	4.0
PSI-graph [10]	98.7	0.78
Hamed et al. [27] (ARM only)	96.35	-

Comparing with some recent approaches on PSI-graph for IoT botnet detection, we can see that our proposed method outperforms graph-based methods with Deep Graph Convolutional Neural network and achieve a high detection rate as PSI-rooted subgraph approach. Besides, the model shows a lower false-alarm rate than the subgraph-

based model. It is obvious that PSI-walk can cluster IoT botnets into families, while PSI-rooted subgraph can not represent. Compared to the study [26], [10], the PSI-walk is less accurate due to the paths found by RL are near-optimal. As, there are a few cases where the psi-walk is not the best abstract, and the algorithm used in this research is that Q-learning is quite simple. The existing PSI graph-based methods [10], [26] focus on the brute-force PSI-graphs approach to identify the malicious behavior of IoT botnet to detect IoT botnet malware. Meanwhile, the malicious behavior of IoT botnet only appears in some paths or subgraphs of the PSI-graph. Therefore, brute-force PSI-graphs traversal makes it highly complex to visit redundant paths or subgraphs. In addition, our model is compared with a promising method proposed by Hamed et al., [27]. This method also uses static features as well as machine learning classifiers to detect IoT botnet. Nevertheless, this method only applies to ARM samples. Table 4 shows that our multi-architecture model can generalize even better than single-architecture method. All of these papers are using the same exact training and testing datasets.

4 Conclusions

With the gradual development pace and the ubiquitous nature of IoT, cybercriminals are always aiming to exploit, attack, and abuse IoT applications. Moreover, the threats that violate privacy when transmitted/received, and processed via the IoT network is also an endless race between security researchers and malicious code authors. In this paper, we have presented a novel PSI-walk based approach for IoT botnet detection. We propose a method to use a reinforcement learning model to extract the PSI-walk features and effective classify IoT botnet malware. In summary, our main contributions are the following: (1) we present a PSI-walk based effective feature for IoT botnet detection as a proper representation ELF files; (2) we proposed a reinforcement learning model and a LSTM model to achieve high accuracy and precision in classifying malicious and benign samples which achieved high accuracy as 97.14% and low false positive rate of 2.7%. However, the proposed method depends on the name of functions or vertices of PSI-graph, so attackers have their chance to bypass this method by changing the function name. Therefore, in the future work, we will try to embed the structure of PSI-graph to create more robust novel features.

5 Acknownlegement

This research was partially funded by Hanoi Open University in Vietnam with project MHN 2020-02.09

References

1. "Internet of Things - number of connected devices worldwide 2015-2025, Available at: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/" (visited on May 3rd 2020)
2. Manos Antonakakis et al.: Understanding the Mirai Botnet. In: Proceedings of the 26th {USENIX} security symposium ({USENIX} Security 17, pp. 1093-1110, 2017
3. Elisa Bertino and Nayeem Islam.: Botnets and Internet of Things Security. Computer, 50(2), pp. 76–79, 2017. https://doi.org/10.1109/MC.2017.62.
4. Flashpoint.: Mirai Botnet Linked to Dyn DNS DDoS Attacks. Available at: https://www.flashpoint-intel.com/blog/cybercrime/mirai-botnet-linked-dyn-dns-ddos-attacks. (visited on May 2nd 2020)
5. Ozawa, Seiichi, et al.: A study of IoT malware activities using association rule learning for darknet sensor data. International Journal of Information Security, 19(1), pp. 83–92, 2020.
6. William Peters et al.: 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, 2020
7. Hayate Takase et al.: 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, 2020
8. Kai-Chi Chang, Raylin Tso, and Min-Chun Tsai.: IoT sandbox: to analysis IoT malware Zollard. In: Proceedings of the Second International Conference on Internet of things, Data and Cloud Computing, pp. 1–8, 2017. https://doi.org/10.1145/3018896.3018898
9. Quoc-Dung Ngo, et al.: A survey of IoT malware and detection methods based on static features. ICT Express, 2020, https://doi.org/10.1016/j.icte.2020.04.005
10. Huy-Trung Nguyen et al., "A novel graph-based approach for IoT botnet detection," International Journal of Information Security, pp. 1–11, 2019
11. Pohl, I.: Heuristic search viewed as path finding in a graph. Artificial intelligence, pp. 193–204, 1970.
12. Anbuselvi, R., and M. Phil.: Path Finding Solutions for Grid Based Graph. Advanced Computing: An International Journal, vol. 4, no. 3, pp. 51–60, 2013
13. Yousefi, Shamim et al.: Applications of big data analytics and machine learning in the internet of things. In: Handbook of Big Data Privacy, pp. 77–108, 2020
14. Rafiqul Islam, Ronghua Tian, Lynn M. Batten, and Steve Versteeg.: Classification of malware based on integrated static and dynamic features. Journal of Network and Computer Applications, vol. 36, pp. 646–656, 2013
15. Liang Xiao, Xiaoyue Wan, Xiaozhen Lu, Yanyong Zhang, and Di Wu.: IoT Security Techniques Based on Machine Learning. Vol. 35, pp. 41–49, 2018
16. Alireza Souri et al.: A state-of-the-art survey of malware detection approaches using data mining techniques. Human-centric Computing and Information Sciences, 8(1), 2018. https://doi.org/10.1186/s13673-018-0125-x
17. Truong, T. C, et al.: Intelligence in the Cyber Domain: Offense and Defense. Symmetry, vol. 12, no. 3, p. 410, 2020.
18. Yang Xin et al.: Machine Learning and Deep Learning Methods for Cybersecurity. IEEE Access, vol. 6, pp. 35365–35381, 2018

19. Blount et al.: Adaptive rule-based malware detection employing learning classifier systems: a proof of concept. In: 35th Annual Computer Software and Applications Conference Workshops, pp. 110–115, 2011
20. Urbanowicz et al.: Learning classifier systems: a complete introduction, review, and roadmap. Journal of Artificial Evolution and Applications, 2009
21. Wu, Cangshuai, et al.: Enhancing machine learning based malware detection model by reinforcement learning. In: Proceedings of the 8th International Conference on Communication and Network Security, pp. 74–78, 2018
22. Jiawei Su, Danilo Vasconcellos Vargas, Sanjiva Prasad, Daniele Sgandurra, Yaokai Feng, Kouichi Sakurai.: Lightweight Classification of IoT Malware based on Image Recognition. In: IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), vol. 2, pp. 664–669, 2018
23. Pektaş, Abdurrahman et al.: Classification of malware families based on runtime behaviors. Journal of information security and applications, vol. 37, pp. 91–100, 2017
24. Xiao, Liang, et al.: Cloud-based malware detection game for mobile devices with offloading. IEEE Transactions on Mobile Computing, 16(10), pp. 2742–2750, 2017
25. Du, Y., et al.: An android malware detection approach using community structures of weighted function call graphs. IEEE Access, vol. 5, pp. 17478–17486, 2017
26. Huy-Trung Nguyen, et al.: PSI-rooted subgraph: A novel feature for IoT botnet detection using classifier algorithms. ICT Express, vol. 6, no. 2, pp. 128–138, 2020
27. Hamed HaddadPajouh, Ali Dehghantanha, Raouf Khayami, Kim-Kwang Raymond Choo.: A Deep Recurrent Neural Network Based Approach for Internet of Things Malware Threat Hunting. Future Generation Computer Systems, vol. 85, pp. 88–96, 2018. https://doi.org/10.1016/j.future.2018.03.007