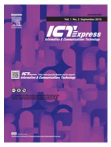
Available online at www.sciencedirect.com
ScienceDirect
ICT Express 6 (2020) 280–286
www.elsevier.com/locate/icte



A survey of IoT malware and detection methods based on static features

Quoc-Dung Ngo[a], Huy-Trung Nguyen[b,c,*], Van-Hoang Le[c], Doan-Hieu Nguyen[c]

[a] *Posts and Telecommunications Institute of Technology, Viet Nam*
[b] *Graduate University of Science and Technology, Vietnam Academy of Science and Technology, Hanoi, Viet Nam*
[c] *People's Security Academy, Hanoi, Viet Nam*

Received 11 February 2020; received in revised form 9 April 2020; accepted 18 April 2020
Available online 15 May 2020

Abstract

Due to a lack of security design as well as the specific characteristics of IoT devices such as the heterogeneity of processor architecture, IoT malware detection has to deal with very unique challenges, especially on detecting cross-architecture IoT malware. Therefore, the IoT malware detection domain is the focus of research by the security community in recent years. There are many studies taking advantage of well-known dynamic or static analysis for detecting IoT malware; however, static-based methods are more effective when addressing the multi-architecture issue. In this paper, we give a thorough survey of static IoT malware detection. We first introduce the definition, evolution and security threats of IoT malware. Then, we summarize, compare and analyze existing IoT malware detection methods proposed in recent years. Finally, we carry out exactly the methods of existing studies based on the same IoT malware dataset and an experimental configuration to evaluate objectively and increasing the reliability of these studies in detecting IoT malware.

Keywords: Internet of Things (IoT); Static-based; IoT botnet malware; Survey detection

Contents

1. Introduction

Internet of Things (IoT) is the largest digital mega-trend that bridges physical and virtual worlds. The increment in the connectivity of people, objects, machines and the Internet is

* Corresponding author at: Graduate University of Science and Technology, Vietnam Academy of Science and Technology, Hanoi, Viet Nam.
E-mail addresses: dungnq@ptit.edu.vn (Q.-D. Ngo), huytrung.nguyen.hvan@gmail.com (H.-T. Nguyen), levanhoang.hvan@gmail.com (V.-H. Le), doanhieunguyen.psa@gmail.com (D.-H. Nguyen).

https://doi.org/10.1016/j.icte.2020.04.005
2405-9595/

leading to the emergence of new business models as well as new interactions between mankind and the rest of the world. Due to the complexity of design and implementation in both hardware and software, as well as the lack of security functions and abilities, IoT devices are becoming an attractive target for cyber criminals who take advantage of weak authentication, outdated firmwares, and malwares to compromise IoT devices. By 2020, it is estimated that 25% of all cyber-attacks target IoT devices (https://www.gartner.com/). With the rapid adoption of IoT technologies in the industry, there will be an endless rise in these attacks. One of the most dangerous threats to IoT devices among them is malware. In Oct. 2016, Dyn (major US DNS service provider) was under one of the largest and most powerful DDoS attacks in recent history by Mirai malware family. The malware infected over 1.2 million IoT devices and targeted many popular online services such as Google, Amazon, etc.

Therefore, improving security aspects of IoT devices is becoming more and more urgent for researchers, especially when dealing with IoT malware. There are many research studies on security issues for IoT devices. Typical of such is Granjal et al. [1] who focused on analyzing extant protocols and mechanisms to secure communications for the IoT. Djamel Eddine Kouicem et al. [2] presented a comprehensive top down survey of the most existing proposed security and privacy solution in IoT. While, Djamel Eddine Kouicem et al. have categorized the various applications of IoT to identify security requirements and challenges for them, thereby analyzing traditional encryption solutions to deal with confidentiality, privacy, and availability. Besides, they also reviewed emerging technologies such as Blockchain and Software Defined Networking. In this perspective, a recent survey by Imran Makhdoom et al. [3] is quite comprehensive when presenting security issues and risks of threats to IoT devices. Besides, they emphasized that inherent safety provided by the communication protocols does not guard against harmful IoT malware and node compromise attacks. Hassan et al. [4] conducted a survey on security issues for IoT devices. However, the authors only focus on introducing solutions including lightweight authentication and encryption, not the IoT malware detection problem. Additionally, Felt et al. [5] reviewed the 46 pieces of mobile malware in the wild and collected dataset to evaluate the effectiveness of mobile malware identification and prevention methods. Costin et al. [6] only presented a comprehensive survey and analysis of all currently known IoT malware classes, without discussing IoT malware detection approaches.

IoT malware detection approaches could be classified into two main domains based on the type of strategy: dynamic and static analysis. Dynamic approach [7] consists of monitoring executables during run-time period and detecting abnormal behaviors. However, monitoring executing processes is resource-intensive, and in some cases, malware could infect real environments. Besides, during execution time, it is not possible to fully monitor all their behaviors because many types of malware require trigger conditions to perform malicious behaviors. In addition to the common limitations of dynamic analysis, the execution of IoT executable files faces many issues such as diverse architectures (e.g., MISP. ARM, PowerPC, Sparc), and resource constraints of IoT devices. Therefore, it is difficult to configure an environment that meets the requirements for IoT executables to function correctly and fully. By contrast, static approach is performed by analyzing and detecting malicious files without executing them. One of the major advantages of static analysis is the ability to observe the structure of malware. In other words, we can explore all possible execution paths in malware sample without considering the diversity of processor architecture, thus making this approach effective to solve the heterogeneous issues of IoT devices. Thus, although there are many studies on security issues for IoT surveys, especially IoT malware detection, but no research has focused on methods of detecting IoT malware based on static analysis. Different from the existing survey studies when only evaluated based on the published results of the studies, this paper experimented exactly these methods with the same large dataset and the same system configuration.

In summary, the major contributions of this paper can be summarized as follows:

– *First*, we present a comprehensive overview of the interaction between the currently known IoT botnet families.

– *Second*, we provide a detailed taxonomy of static-features based IoT malware detection and discuss their shortcomings.

– *Third*, we accurately re-experimented the existing studies based on static analysis with the same large dataset and system configuration.

The rest of the paper is organized as follows: Section 2 provides an overview of IoT malware and its life-cycle, thereby better understanding the features that can be used by static analysis methods introduced in Section 3. Section 3 discusses the current IoT malware detection methods based on static features with their advantages and disadvantages. Section 4 presents and evaluates the methods by experiments. Furthermore, Section 5 shows the conclusion of this survey.

2. IoT malware

In the past few decades, most malwares were designed to target personal computers running Microsoft Windows, the most popular operating system in the world with 83 percent of market share [8]. However, the diversity of computing devices has rapidly changed in recent years due to the Internet of Things technology. IoT devices are built upon a variety of CPU architectures, even on resource-constrained hardware such as Unix-based operating systems. Along with this change, IoT devices are becoming a favorite target by the attackers due to a lack of security design or implementation. In general, IoT malware has several characteristics such as IoT malware is used to perform DDoS attacks; IoT malware scans the open port of IoT services such as FTP, SSH or Telnet; IoT malware performs a brute-force attack to gain access to IoT devices.

Alex et al. [9] stated that most of the current malware generated by copying the source code according to online instructions or a variant of the same malicious code created by the malware writer. Through analysis, evaluation and synthesis of several studies such as [6,10,11] and manual analysis of some IoT malware samples this paper gives a brief chart of the recent development and evolution of IoT malware, as shown in Fig. 1. Because IoT includes a vast and ever growing array of connected devices (e.g., smart meters, medical devices, public safety sensors, etc.), many IoT malware families such as Aidra, Bashlite and Mirai can utilize scanners that are designed to locate exposed ports and default credentials on these devices. In the last decade, IoT malware keeps developing and targeting new victims with various architecture. Mirai's evolution is gravitated toward changes in enterprise IT operations, extending its attack surface and bringing new zero-day exploits to consumer-level devices. In March 2019, IBM Xforce found a Mirai-like malware aimed at enterprises' IoT devices. These attacks drop crypto currency miners and backdoors onto affected devices.

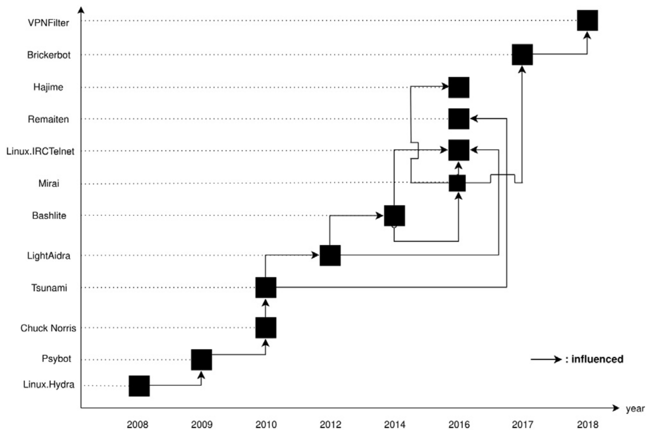


Fig. 1. The evolution of IoT malware.

There is a close correlation between IoT malware families reflected by the similarity of their functions as well as their source codes. Fig. 1 presents Linux.Hydra as the first DdoS capable IoT malware that appeared since 2008. Since the release of Linux.Hydra's code, IoT malware developers have evolved into variants of Psybot, Chuck Norris and the last variation, Tsunami. However, part of the Tsunami's code was developed into the Remaiten and LightAidra, which are among the newest IoT malware. Also, the figure shows that Tsunami is the ancestor of Bashlite and from Bashlite, the Mirai malware inherited and evolved more and more complex in 2016. From there, Mirai has continued to develop through variations that make it as a malware family rather than an individual stream of malware such as BrickerBot, VPNFilter. Accordingly, it cannot be denied that today's popularity of DDoS-capable IoT malware is steadily increasing because malware authors will continue to apply their creativity and programming skills to mutate their malwares for more critical infection on IoT devices. In summary, based on the analysis of the characteristics, evolution of IoT malware, we have found that there are existing static characteristics of IoT malware that could be used as the features to detect malicious code, such as elf structure, strings, function call graph, grayscale image, etc. These features will be discussed in detail in Section 3.

3. Iot malware detection based-on static feature analysis

In this section, we discuss the static IoT malware detection methods that have been proposed since 2013. In static analysis, existing studies [7,12–21] commonly used the following static characteristics: control flow graph (CFG), operation codes (opcode), strings, file header, gray-scale image, etc. The way these features were extracted and processed greatly affects the accuracy and complexity of IoT malware detection methods. In the next subsection, we will demonstrate our method to divide these static-based features, illustrated in Fig. 2.

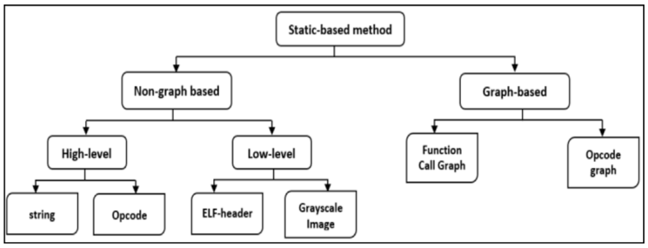


Fig. 2. Taxonomy of static-based features in IoT malware detection.

3.1. Non graph-based IoT malware detection methods

Non graph-based detection methods aim at building a model that contains the properties of binary file structure to classify whether a binary file is malicious or benign. These methods rely on extracting the static features such as Operation Codes, Strings or File Structure to distinguish malicious samples. These characteristics can be divided into two groups: high-level features and low-level features. In particular, the low-level features can be obtained directly from binary file structure itself, while the high-level features must be extracted by disassembler (e.g., IDA Pro, radare2).

3.1.1. High-level features

Operation Code (Opcode) is one of the most popular features for malware detection. An Opcode is a single instruction that can be executed by the processor (CPU), which describes the behaviors of an executable file. In assembly language, an opcode is a command such as CALL, ADD or MOV. Based on this feature, Hamed HaddadPajouh et al. [12] proposed a method using Recurrent Neural Network (RNN) deep learning to detect IoT malware using the sequences of Opcodes. This method achieved the accuracy of 98.18 percent with the dataset of 281 ARM-based IoT malicious and 270 ARM-based IoT benign samples. Ensieh Modiri Dovom et al. [13] converted the executable files' Opcodes into a vector space and applied the fuzzy and fast fuzzy pattern tree methods to detect IoT malware. To prove this method in malware detection, they conducted an experiment on an ARM-based IoT dataset, comprising 1078 benign and 128 malware samples, and the experimental results achieved an accuracy of 99.83%. In similarity, Hamid Darabian et al. [14] presented a sequential opcode-based technique for IoT malware detection. By counting the number of opcode repetitions in executable files, the authors found that some opcodes in malware samples have higher frequency of repetition than benign files. Their experimental result achieved 99% accuracy and F-measure in the detection of IoT malware from benign samples. Nghi Phu et al. [22] proposed a feature selection method to detect cross-architecture malware, called CFDVex. The experiment achieved good results for cross-architecture malware detection. The experimental results show that the method is positive when it can detect the IoT malware for the MIPS architecture samples with a 95,72% accuracy rate by only train Intel 80386 architecture samples.

Strings — A string in an executable file is a sequence of characters such as "gayfgt" that is usually stored in ASCII (1 byte per character) or Unicode (2 bytes per character) format. Each printable string within an executable file could extract valuable information such as IP address, URL to connect, etc., to determine whether an executable is malicious or not [23]. Mohannad Alhanahnah et al. [15] generated good signatures for multi-architecture IoT malware classification based on printable strings. For evaluations, they used two IoT-POT malware datasets with 5150 malware samples, and with this signature-based detection mechanism, their experimental results achieved 95.5% IoT malware detection rate

3.1.2. *Low-level features*

ELF file header — An ELF (Executable and Linkable format) file format contains lots of interesting information that can be used in the malware detection. Based on this context, Farrukh Shahzad and Muddassar Farooq [16] presented a Linux malware detection tool, called ELF-Miner. To demonstrate their method, the dataset comprising 709 ELF samples was used and achieved more than 99.9% detection accuracy with less than 0.1% false alarm rate. In another work, Jinrong Bai et al. [17] introduced a method that extracts system call information from the symbol table of ELF files, they applied four machine learning algorithms for Linux malware detection. Using a dataset consisting of 756 benign and 763 malware samples, the experimental results achieved more than 98% accuracy on detecting an ELF file is malicious or benign.

Grayscale images — A grayscale image is type of image that each pixel has a value in the range from 0 to 255. In the malware detection problem, the executable files are analyzed and converted into binary strings 0 and 1, then combining those binary values into 8-bit vector segments that represent hex value from 00 to FF. These vectors are eventually converted into image data with a pixel value range between 0 and 255, where 0 as black and 255 as white. In this perspective, Su et al. [18] proposed a lightweight solution to distinguish between IoT malware and benign samples by feeding these gray-scale images to convolutional neural network model for detection. Besides, files of any size will be normalized to fit with a 64×64 grayscale image and its remaining content will be deleted if redundant or covered with zero values if missing. The experiments achieved 93.33% accuracy for detecting IoT malware.

3.2. *Graph-based IoT malware detection methods*

The most popular feature for malware detection is the Control Flow Graph. A control flow graph is a directed graph that represents all the possible execution paths that can be taken during the program, where each vertex (node) is represented by a basic block and each directed edge represents a possible control flow between the basic blocks. Control flow information in two key forms (1) inter-procedural control flow and (2) intra-procedural control flow. The interprocedural control flow graph demonstrates the association between functions and procedures in an executable file as a single CFG. Meanwhile, the intra-procedural control flow graph is demonstrated as a set of control flow graphs with one graph for a procedure or a function. Hisham Alasmary et al. [19] used the control flow graph (CFG) as an abstract structure to highlight the similarities and differences between IoT and Android malware binaries. Experimental results on the dataset consisting of 2.874 IoT malware samples and 201 android malware samples showed that IoT malware is more likely to contain a lesser amount of nodes and edges than Android malware, and graph-theoretic features between the IoT and android malware CFGs have a major variation. Therefore, they demonstrate the usefulness of CFGs in detecting IoT malware and android malware. Continuing to improve and expand this research direction in detecting IoT malware, Hisham Alasmary et al. [20] showed an IoT malware detection method that utilized Control Flow Graphs (CFGs) by using 23 static features to represent the characteristic of the CFG of IoT malware. By using this simple approach, this work achieved the accuracy of 99.66 percent with the dataset of 6000 malware and benign samples.

Follow the same approach, Azmoodeh et al. [21] proposed a deep learning-based method to detect Internet Of Battlefield Things (IoBT) malware that is based on the Opcodesequence graph. They evaluated the proposed method with a dataset including 128 malwares and 1078 benign files then achieved accuracy and precision rate of 98.37 percent and 98.59 percent respectively. HT Nguyen et al. [7] proposed an IoT botnet detection method based-on tracking footprints leaving at the steps of the botnet life cycle. These footprints were displayed as Printable String Information (PSI) which are used in the programming phase of any program such as IP address, username/ password patterns. In this work, they defined a graph-based data structure called PSI-Graph to represent the life cycle behavior of IoT botnet. On the dataset of more than 10000 samples, this study achieved the accuracy of more than 98 percent. In comparison with other methods, this work shows a better result in both detection rate and classifying time. Based on the above literature review, we compare the static IoT malware detection methods in Table 1 regarding their features used for detection, classification algorithms as well as the weaknesses that could affect the performance of these methods.

4. Experimental results and comparison

4.1. *Dataset description*

All methods for IoT malware detection based on static features (in Section) have been experimented on the same dataset of executable files on IoT. That dataset is described

Table 1
Comparison between static-based methods for IoT malware detection.

Method	Features	Mechanism	Passive technique	Weakness
[12]	Opcode	Identify malicious code through the sequences of Opcode	Neural networks	Only for ARM-based samples
[13]	Opcode	Apply fuzzy pattern tree to detect malicious sample	Fuzzing	Only for ARM-based samples. The dataset is not had enough samples.
[14]	Opcode	Detect malware by analyzing Opcode frequency	Machine learning	Only for ARM-based samples.
[22]	Opcode	Detect malware by using Vex intermediate representation	Machine learning	Only experiment with MIPS-based samples
[15]	Strings	Generate signature to classify IoT malware	Clustering	Time consuming Only for 4 malware families
[16]	ELF header	Extract features from sections of a binary file to detect malware	Machine learning	The structure of binary file is easy to modify.
[18]	Grayscale Image	Represent binary sample as grayscale image to detect malicious code	Neural network	Lose the accuracy when obfuscation or encrypt technique was applied
[20]	CFG (Function Call Graph)	Calculate 23 properties of CFG to separate malicious and benign samples	Machine learning, Neural network	Time consuming The defined properties is not correct.
[21]	Opcode graph	Construct Opcode graph as a type of CFG to detect malware	Graph theory	Only for ARM-based samples
[7]	PSI graph	Use the PSI-Graph extracted from function call graph to detect malware	Neural network	Transforming graphs into vector data is time consuming

as follows. Our datasets consist of 7199 malware samples and 4001 benign sample. Herein, the malware dataset is provided by IoTPOT team [24] (within a 1-year collected period between October 2016 to October 2017) and VirusShare. Besides, the benign samples were extracted from the firmware of IoT SOHO (Small Office/Home Office) devices with the binwalk tool and collected from the Internet repository. These extracted files are verified to ensure benignity by VirusTotal. We conducted the experiment with Python on Ubuntu OS 16.04, with an Intel Core i5-8500 processor running at 3.00 GHz, 12 GB NVIDIA GeForce GTX1080Ti graphic card and 32 GB memory.

4.2. Evaluation metrics

In order to evaluate the effectiveness of the proposed method, we used TP, TN, FP, and FN standard values, these metrics are described as follows:

– True positive (TP): indicates that a malware sample was detected correctly and labeled as malware

– True negative (TN): indicates that a benign sample was correctly detected and labeled as benign

– False positive (FP): indicates that a benign sample was wrong and labeled as malicious

– False negative (FN): indicates a malware sample was wrong and labeled as benign

Based on the above criteria, we measure the following performance metrics:

– Accuracy = (TP + TN)/(TP + FP + TN + FN)

– TPR = TP/(TP + FN)

– FPR = FP/(FP + TN)

4.3. Results

As shown in Table 2, the methods of using non-graph based features have achieved better results than using a graph-based features method in detecting IoT botnet, both in terms of accuracy and time cost complexity. ELF-header method shows a promising result with different algorithms. By using RIPPER, PART and Decision Tree C4.5 algorithm, this method achieved the accuracy of more than 99 percent for each algorithm. Although there are about 2000 malware samples among the dataset of 7000 samples that cannot read and calculate the file header, this method proves its efficiency when tackling the IoT malware detection problem. Header features playing an important role in this method are STB_WEAK, SectionHeaderOffset, STT_NOTYPE, .bss_alignment, STT_OBJECT_STB_GLOBAL. The main advantage of this method is the simplicity of the feature extraction phase as well as the simplicity of these features itself. The proper usage of this data type allows to obtain an initial state of an application being stored by an executable binary that in some sense defines its further functionality. Unfortunately, the availability of advanced protection techniques to hide the real initial state of the process does not allow applying methods adopting this feature group comprehensively.

From the results, it can be seen that the image-based method, proposed by Su et al. [18] achieved a slightly lower detection rate, compared with other non graph-based methods. By using the concept of image processing as well as utilizing a lightweight convolutional neural network, this work is capable of implementing on IoT devices with fewer less computation resources. This method achieves the detection rate, the false positive rate and the false negative rate of 89 percent, 12.7 percent and 1.4 percent respectively. It is understandable since there are several obfuscation techniques that could change the structure of binary files as the authors claimed. Despite this inherent weakness of static analysis generally, this work showed promising scalability due to its simplicity in design as well as its capacity of protection against node failures.

In comparison, by using Strings features, the method proposed by Mohannad Alhanahnah et al. [15] also achieved a good result with an accuracy of around 99 percent for each classifier. Based on observing the cross-architectural similarity among malware samples from the same family, this method showed the advantage in grouping IoT malware samples into multiple families using the code statistics feature. Thus, they can design a signature generation scheme to create signatures using printable strings and statistical features. Although the detection rate on the dataset is good, this method could lose accuracy when detecting new malware families as other signature-based methods did.

From the results, it can be seen that the opcode-based method, proposed by Nghi Phu et al. [22] achieved a good result but there are still some potential weaknesses of these works. Firstly, due to the differences between the instruction sets on different architectures, this study needs extensions as well as improvements to cover multiple architectures and time consuming processing. Secondly, the most inherent weakness of these studies is that Opcode could be easily obfuscated by attacker's simple techniques which could lead to false detection when it faced heavily obfuscated malwares.

From the CFG-based method, by using the properties of the function call graph, the method proposed by Hisham Alasmary et al. [19] achieved a quite low which is 89, 85 and 95 percent for SVM, Logistic Regression (LR) and Random Forest classifier respectively with a high FPR and FNR. Besides, calculating CFG properties to input the detection model was a time-consuming process. The main limitation of this approach is that the evaluation of graph metrics was not exact. The authors claimed that the CFGs of IoT malware have one component due to the simplicity in the operation of IoT malware. However, by applying this work on our dataset, we notice that the control flow of some IoT malware families such as Mirai or Bashlite is complicated, so the CFGs of them have multiple components, which could be named as the subgraph of CFG.

As can be seen, the method using PSI-Graph has a good result in comparison with other methods, especially the CFG based method. This method successfully addressed the IoT malware detection problem based on analyzing the botnet life cycle by investigating the relationship between PSIs in the function call graph of a sample. This study achieved the accuracy of 98.7 percent as well as the false positive and false negative rates of 0.78 and 1.83 percent respectively. Even though this method showed a good performance in both detection rate and time, this work still suffers from the complexity of graph analyzing phase. To improve efficiency, authors should reduce the time of graph pre-processing by using other algorithms as well as trying to travel graph effectively to avoid less-value nodes or edges. In summary, the studies of using non-graph based features are often quite weak with malware using obfuscation techniques. Meanwhile, the graph-based approach can generalize and represent structured information, complex information about the behavior of malware.

5. Conclusion

Detecting IoT malware is becoming an increasingly urgent issue in ensuring the safety of the Internet system and private data. This paper gave a comprehensive review of emerged IoT malware and static-based detection methods. We discussed the main techniques as well as strengths and weaknesses in existing static IoT malware detection. In summary, IoT malware detection methods can be divided into two groups: non graph-based and graph-based methods. The non graph based methods can achieve a good result when detecting "simple" and "forthright" malware without customization or obfuscation, but potentially loses accuracy when detecting unseen malware. On the contrary, the graph-based methods show advantages when analyzing the control flow of IoT malware, thus have the potential to accurately detect unseen or complicated malicious code despite the complexity of these methods. To compare the performance of these studies, we also implemented and evaluated them on the same IoT dataset consisting of 11200 samples. Based on the mechanism, detection analysis and processing time, we summarized the advantages and limitations of these studies that can be used to improve the efficiency in future researches. As further extension of this work, we plan to design and develop a graph-based lightweight detection method that will help to deal with distinguish malicious executable file in IoT devices.

CRediT authorship contribution statement

Quoc-Dung Ngo: Conceptualization, Methodology, Investigation, Writing - original draft, Writing - review & editing, Supervision, Project administration. **Huy-Trung Nguyen:** Investigation, Methodology, Software, Data curation, Writing - original draft, Writing - review & editing, Visualization, Supervision. **Van-Hoang Le:** Software, Visualization. **Doan-Hieu Nguyen:** Data curation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Table 2
Comparison results.

Method	Classifier	Acc (%)	FPR (%)	FNR (%)	Feature extracting and pre-processing time	Classifying time
ELF-header [16]	RIPPER	99.8	0.2	0.2	1 h 50 m	0.75 s
	PART	99.8	0.2	0.2		1.27 s
	DT (J48)	99.6	0.5	0.3		1 s
String-based [23]	SVM	98	0.9	2.2	4 m 47 s	12.4 s
	kNN	99.8	0.4	0.2		1 s
	DT (J48)	99.4	0.4	0.6		8.75 s
	RF	99.7	0.3	0.4		9.71 s
Opcode-based [22]	SVM	97.02	1.05	2.51	4.5 days	2 m 10 s
Image-based [18]	Neural network	89.1	12.7	1.4	14 m 19 s	2 m 19 s
CFG-based	SVM	89	33.8	4.4	5 days	1.45 s
	LR	85	15.1	19.0		0.5 s
	RF	95	7.5	5.9		1.75 s
PSI-graph	Neural network	98.7	0.78	1.83	1 h 28 m	1 m 47 s

References

[1] J. Granjal, E. Monteiro, J.S. Silva, Security for the internet of things: a survey of existing protocols and open research issues, IEEE Commun. Surv. Tutor. 17 (3) (2015) 1294–1312.

[2] Djamel Eddine Kouicem, Abdelmadjid Bouabdallah, Hicham Lakhlef, Internet of things security: a top-down survey, Comput. Netw. 141 (2018) 199–221.

[3] Imran Makhdoom, et al., Anatomy of threats to the internet of things, IEEE Commun. Surv. Tutor. 21 (2) (2018) 1636–1675.

[4] W.H. Hassan, et al., Current research on internet of things (IoT) security: A survey, Comput. Netw. 148 (2019) 283–294.

[5] A.P. Felt, M. Finifter, E. Chin, S. Hanna, D. Wagner, A survey of mobile malware in the wild, in: Presented at the Security and Privacy in Smartphones and Mobile Devices (SPSM), 2011, pp. 3–14.

[6] Andrei Costin, Jonas Zaddach, Iot malware: Comprehensive survey, analysis framework and case studies, in: BlackHat USA, 2018.

[7] H.T. Nguyen, Q.D. Ngo, V.H. Le, A novel graph-based approach for IoT botnet detection, Int. J. Inf. Secur. (2019) 1–11.

[8] Emanuele Cozzi, et al., Understanding linux malware, in: IEEE Symposium on Security and Privacy, 2018, pp. 161–175.

[9] K. Allix, Q. Jerome, T.F. Bissyande, J. Klein, R. State, Y.L. Traon, A forensic analysis of android malware – How is malware written and how it could be detected? in: Presented at the IEEE 38th Annual Computer Software and Applications Conference, 2014, pp. 384–393.

[10] Kishore Angrishi, Turning internet of things (IoT) into internet of vulnerabilities (IoV): IoT botnets, 2017, arXiv preprint arXiv:1702.03681.

[11] Michele De Donno, Nicola Dragon, Alberto Giaretta, DDoS-Capable IoT malwares: Comparative analysis and mirai investigation, in: Security and Communication Networks, Wiley, 2018.

[12] 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 Gener. Comput. Syst. 85 (2018) 88–96.

[13] Ensieh Modiri Dovom, et al., Fuzzy pattern tree for edge malware detection and categorization in IoT, J. Syst. Archit. (2019).

[14] Hamid Darabian, et al., An opcode-based technique for polymorphic Internet of Things malware detection, Concurr. Comput.: Pract. Exper. (2019).

[15] Mohannad Alhanahnah, Qicheng Lin, Qiben Yan, Efficient signature generation for classifying crossarchitecture IoT malware, Commun. Netw. Secur. (2018) 1–9.

[16] F. Shahzad, M. Farooq, Elf-miner: Using structural knowledge and data mining methods to detect new (linux malicious executables, Knowl. Inf. Syst. 30 (3) (2012) 589–612.

[17] Jinrong Bai, Y. Yang, S. Mu, Y. Ma, Malware detection through mining symbol table of Linux executables, Inf. Technol. J. 12 (2) (2013) 380–383.

[18] Jiawei Su, Danilo Vasconcellos Vargas, Sanjiva Prasad, Daniele Sgandurra, Yaokai Feng, Kouichi Sakurai, Lightweight classification of IoT malware based on image recognition, in: 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), Vol. 2, 2018, pp. 664–669.

[19] Hisham Alashmary, et al., Graph-based comparison of IoT and android malware, in: International Conference on Computational Social Networks, 2018, pp. 259–272.

[20] Hisham Alasmary, et al., Analyzing and detecting emerging internet of things malware: A graph-based approach, IEEE Internet Things J. 6 (5) (2019) 8977–8988.

[21] Amin Azmoodeh, et al., Robust malware detection for internet of (battlefield) things devices using deep eigenspace learning, IEEE Trans. Sustain. Comput. (2018) 88–95.

[22] T.N. Phu, L.H. Hoang, N.N. Toan, N.D. Tho, N.N. Binh, CFDVex: A novel feature extraction method for detecting cross-architecture IoT malware, in: Proceedings of the Tenth International Symposium on Information and Communication Technology, 2019, pp. 248–254.

[23] Rafiqul Islam, et al., Classification of malware based on integrated static and dynamic features, J. Netw. Comput. Appl. (2013) 646–656.

[24] Y.M.P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, C. Rossow, CIoTPOT: A novel honenypot for revealing current IoT threats, J. Inf. Process. 24 (2016) 522–533.