Building a Malware Detection System Based on a Machine Learning Method

Cho Do Xuan, Tisenko Victor Nikolaevich, Do Minh Tuan, Nguyen the Lam, Nguyen Anh Tuan

***Abstract-** Malware attacks are dangerous and difficult to detect and prevent. Therefore, the task of detecting signs of malware and alerting it for users or the system is very necessary today. One of the most effective malware detection approaches is applying machine learning or deep learning to analyze its behavior. There have been many studies and recommendations to analyze malicious behavior then combined with some sorting or clustering methods to find their signs. In this paper, we will propose a method to use machine learning to detect malicious signs based on their unusual behavior. Accordingly, in our research, we will conduct malicious analysis using static and dynamic analysis methods to detect abnormal behaviors and combine them with a supervised classification algorithm to the conclusion on malware behavior.*

Malware detection, feature selection, machine learning

I. INTRODUCTION

Malware is software programs designed to harm or perform unwanted actions on a computer system. Malicious software is essentially a software like other software on the computer that is used every day and has all the characteristics and properties of a normal software, except that it is more malicious. The study listed some common types of malware including Virus, Worm, Trojan Horse, Malicious Mobile Code, Tracking Cookie, Attacker Tool, Phishing, Hoax Virus. According to statistics [15], the situation of malware distribution in 2019 increased by 79% compared to 2018. This is entirely reasonable because hackers used to focus on information systems in the past. This usually chooses to attack the user primarily. Therefore, malware rapidly increases not only in a number of attacks but also its dangerous levels. In the study [10], there are several approaches to detecting malware. The two basic methods used to detect malware are the sign-based detection method and based on behavioral analysis. Methods of detecting malware based on a set of signs have been studied and applied early because of its rapidity and accurate detection capability. Commonly used signs in this method include hash code, IP, Domain or Indicators of compromise. However, the disadvantage of this method isn't able to detect new malware samples that are not in the signature database. In this paper, we propose a method to detect malware based on machine learning techniques. In the paper [1], there are some difficulties in the method of detecting malware based on machine learning. In our study, we propose a malware detection process based on static and dynamic analysis. Finally, to conclude the existence of malware in the system we propose to use machine learning algorithms.

Revised Manuscript Received on March 30, 2020.

* Correspondence Author

Cho Do Xuan, FPT University, Hanoi, Vietnam. E-mail: chodx@fe.edu.vn,

Tisenko Victor Nikolaevich , Peter the Great St. Petersburg Polytechnic University Russia, St.Petersburg, Polytechnicheskaya, 29 E-mail: v_tisenko@mail.ru

Do Minh Tuan, FPT University, Hanoi, Vietnam. E-mail:tuandmse05518@fpt.edu.vn,

Nguyen The Lam, FPT University, Hanoi, Vietnam. E-mail: lamntse63326@fpt.edu.vn,

Nguyen Anh Tuan, FPT University, Hanoi, Vietnam. E-mail: tuannase62864@fpt.edu.vn

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)

II. RELATED WORKS

2.1. Malware detection technique

a) Detection technique based on static analysis

The malware detection technique is based on the static analysis method, which is characterized by the detection of malware without having to run or execute any of its code, including three main methods: scanner technology, diagnostics based on Heuristics and Integrity Checkers.

b) Detection technique based on dynamic analysis

The malware detection technique based on dynamic analysis is the technique of determining whether a file is infected by executing program code and observing its behavior. Two main techniques for malware analysis include:

- Behavior Monitors/Blockers: Behavior blocker is a technique that monitors the execution behavior of a program in real-time. Besides, this technique also allows the monitoring of suspicious actions and blocks of malware.

- Emulation: Malware detection techniques use emulation that allows the code to be run and analyzed in a simulated environment. Two main techniques are Dynamic heuristic and Generic decryption.

2.2. Detecting malware based on machine learning

To solve the disadvantage of the method of detecting malware based on the signal set, the technique of detecting malware based on analyzing the behavior of malware was born. In the research [1], the authors presented the idea of detecting malware based on file abnormal behavior based on machine learning algorithm. The paper [2], presented a number of basic approaches in the problem of malware detection based on machine learning including how to extract malicious features and detection algorithms. For extracting feature data, recent studies often use three main techniques including [2 -7]: static analysis, dynamic analysis, and combined analysis.

Based on these analysis techniques, the malware will be analyzed and synthesized into the corresponding sequence of behaviors. In this paper, we will use static and dynamic analysis methods to look for abnormal behavior of malware based on the sandbox tool [8]. After configuring and analyzing malware with a sandbox tool, the main groups of behaviors that can be selected and used to extract malicious behavior include Byte sequences, Opcodes, network Activity., System files, API and System Calls, Windows Registry, PE file characteristics. This method has two basic algorithms that are machine learning and deep learning algorithm [9, 10]. In this paper, we use a supervised machine learning algorithm. This detection method is relatively effective and has been researched and experimented in many studies.

2.3. Some malware detection tools

To be able to implement the above methods, we need the support tools corresponding to each specific method. Below, I offer five main groups of tools:
- Antivirus software: Kaspersky, Bitdefender, Avast, Norton, Bkav…
- Network monitoring tool group: TCPView, Wireshark.
- A group of tools for monitoring file system resources: AutoRun. ProcessExplorer, ProcessMon, …
- Registry monitoring tool group: ProcessMon, AutoRun, …
Automatic analysis tool: Sandboxie, Cuckoo Sandbox.

III. DEVELOP A MALWARE DETECTION SYSTEM

3.1. Proposing a model to detect malware

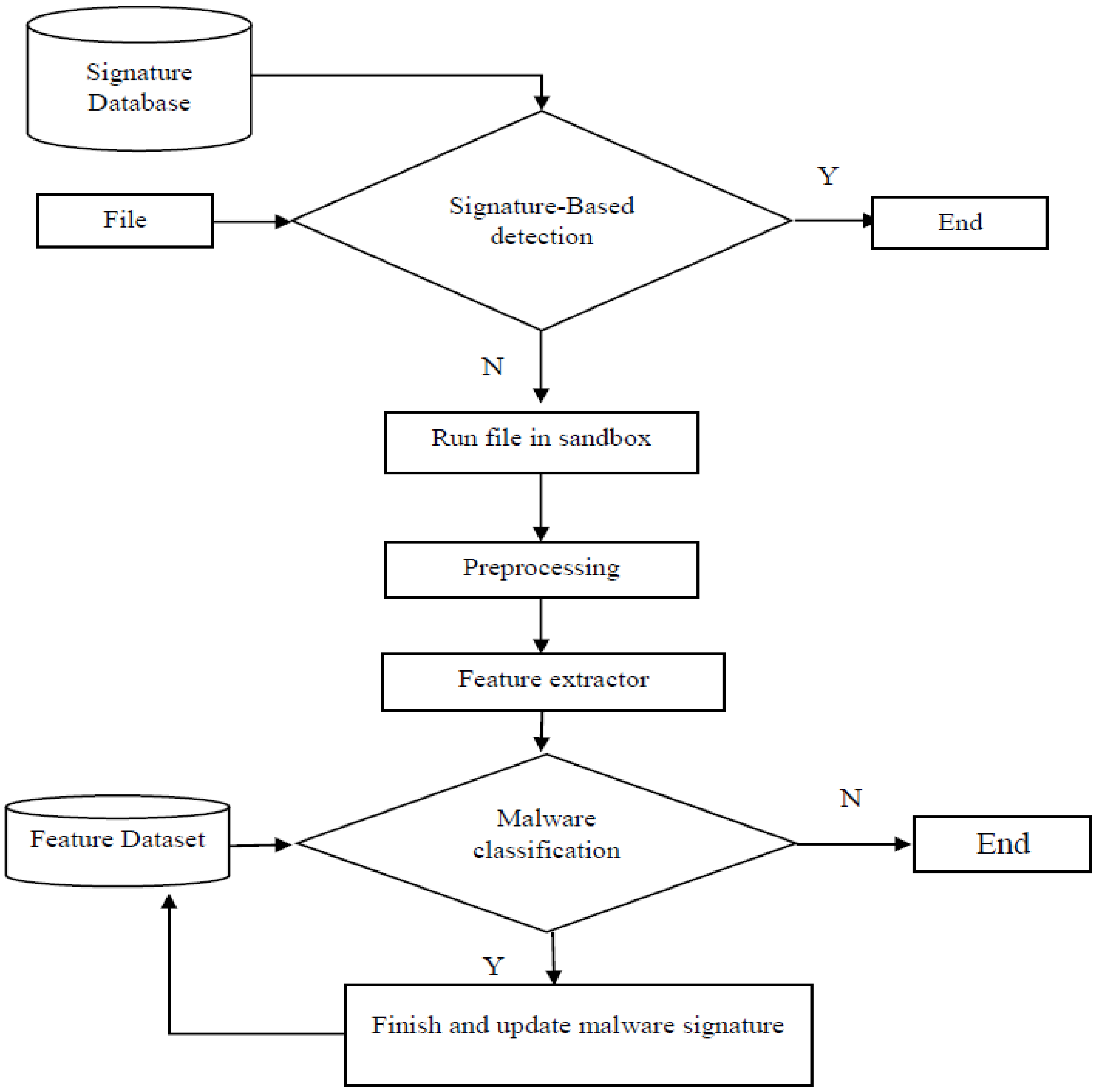

Figure 1. Malicious detection system based on rules and machine learning algorithms

The operating process of a malware detection system is as follows:
- First of all, the data will be checked by the malware signature database. If a match is detected, it will immediately report the occurrence of malware in the system. To accomplish this task, the malware will be sent to the Virus total and checked by the signature database. We use Yara to build a signature database, Virus total as a tool to check data if the database sign not detected. The process of detecting malware with signs does not take much time and has accurate results. However, this method is very difficult to detect new malware. If files don't match with the malware signature database, it will be detected by a machine learning method.
- If the file doesn't found in the signature database, it will be sent to the sandbox. In the sandbox, the file will be loaded into the virtual environment running the test, the sandbox system records the file's behavior logs, the system's machine learning module learns the behavior of the test file in the Feature Dataset of the system by Random Forest algorithm.

3.2. Select and extract features

In the document [11] listed the properties to detect malware. In this paper, we use some of the following features to detect malware. The list of features is shown in the following table.

Table 1. List of malicious features

No	Category	Features	Data Type	Desciption
1	**Static**	Size	Numberic	File size.
2		Timestamp	Numberic	Date file created.
3		Signature	String	File signature.
4		Packer	String	Packer
5		Section Features	String	Section and resource features
6		Static Import	String	Import library.
7		Mutex		Allow only one malware to execute in the system at a time…
8		Processes	String	Process.
9		Dynamic imports	String	Import external library.
10		File Read	String	Get links to malware to read.
11	**Dynamic**	File Written	String	Get links to malware to over write.
12		File Delete	String	Get links to malware to delete.
13		File Copied	String	Get links to malware to copy.
14		File Renamed	String	Get links to malware to rename.
15		File Open	String	Get links to malware to open.
16		File Exists	String	Get links to opened files.
17		File Failed	String	Get links to error files.
18		File Operations	String	List all type of above file.
19		TCP	String	Get ip and using TCP to connect to outside.
20		UDP	String	Get ip and using UDP to connect to outside.
21		HTTP	String	Find all list of connect Http
22		Registry Written	String	Get links to malware to edit registry.
23		Registry Delete	String	Get links to malware to delete registry.
24		API Stats	String	Get API.

3.3. Malware detection algorithm

Random forest is a member of the decision tree algorithm chain. The main idea of this algorithm is to create some decision trees. When the data have a height, build a long tree and the quality of identification (classification/regression) is low, instead of using other methods, people use the Random forest. A random forest is an identifier consisting of a set of decision tree parts combined by the voting method. Decision trees are constructed from different sub-datasets, with different characteristic subset taken randomly from the observed data set. The construction of the Random forest consists of three phases:
- Data creation (random vector generator)
- Building decision trees.
- Incorporate decision trees by the voting method.

The Random forest algorithm is generically described through the following steps:
Step 1: For k = 1, L where L is the number of decision trees:
- Get random set R_k including M data in D.
- Dk is the projection of R_k on the characteristics of the data set to be taken.
- Building T_k decision tree from D_k set, obtaining C_k identifier.

Step 2: Combine voting results $\{Ck\}$ $k = 1L$ to give identifier results to new subjects.

The Random forest algorithm will allow decision trees to run and produce independent results. The answer predicted by the most decision trees will be selected by the Random forest. But if the decision tree is accidentally the same, the result of the decision tree will be the result of the entire model. To ensure that decision trees are not the same, the Random forest will randomly pick a subset of the features at each node. The remaining parameters are used in the Random forest just like in the decision tree. This process is called Bootstrapping. Using the Bootstrap Aggregating Tree (Bagging Tree) helps reduce discrepancies, increase stability and accuracy. In addition, the Random forest model calculates the importance of features, unlike the algorithm that the features are equal. The following figure shows an example of a Random forest:

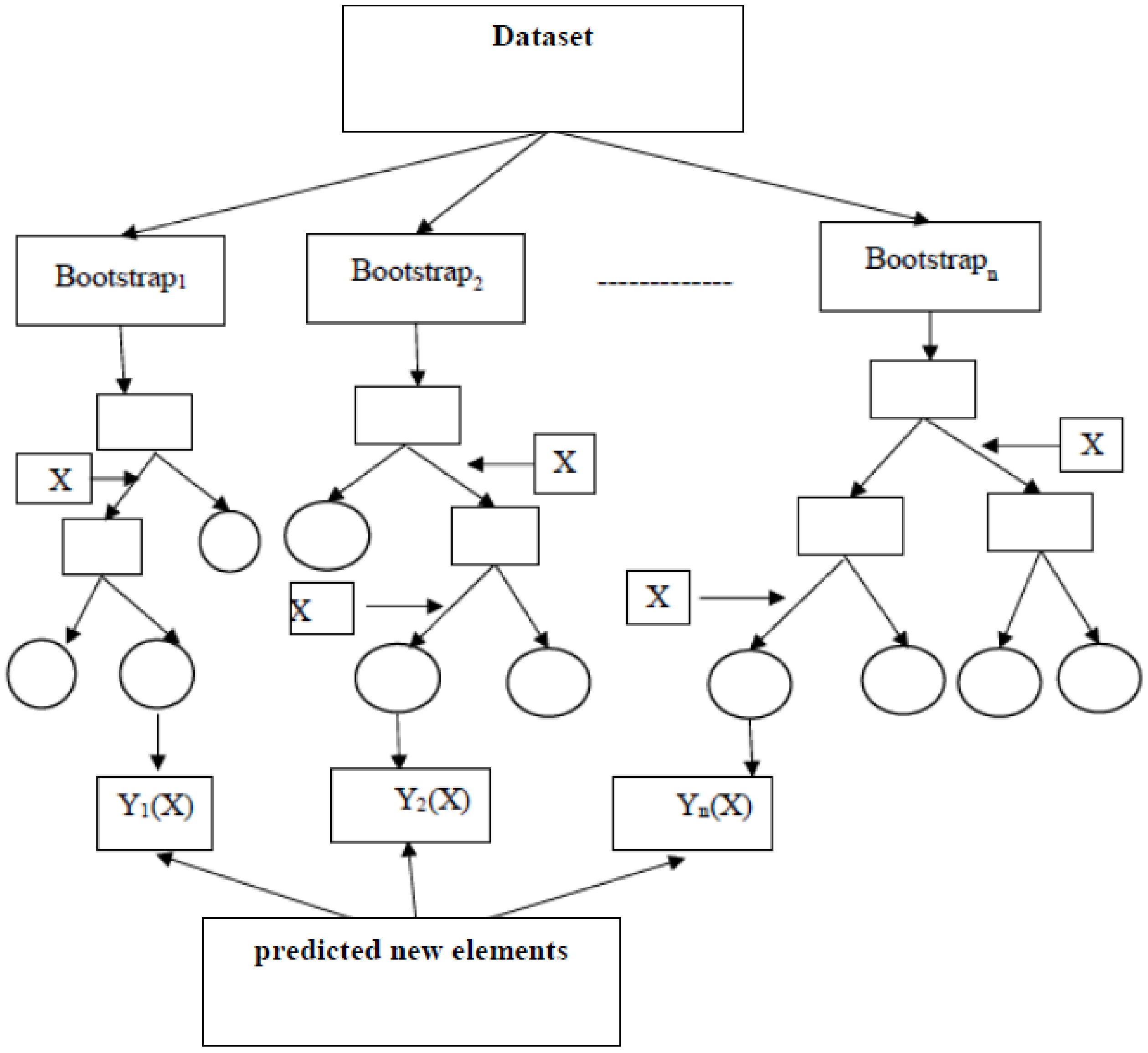

Fig. 1. The operation model of the Random forest algorithm

The random forest depicted in Figure 2 creates a set of unconditioned decision trees, each built on a bootstrap sample set, at which the best partition node is performed from randomly selecting a subset of features.

IV. EXPRIMENTAL AND EVALUATED

4.1. The experimental model of detecting malware based on machine learning

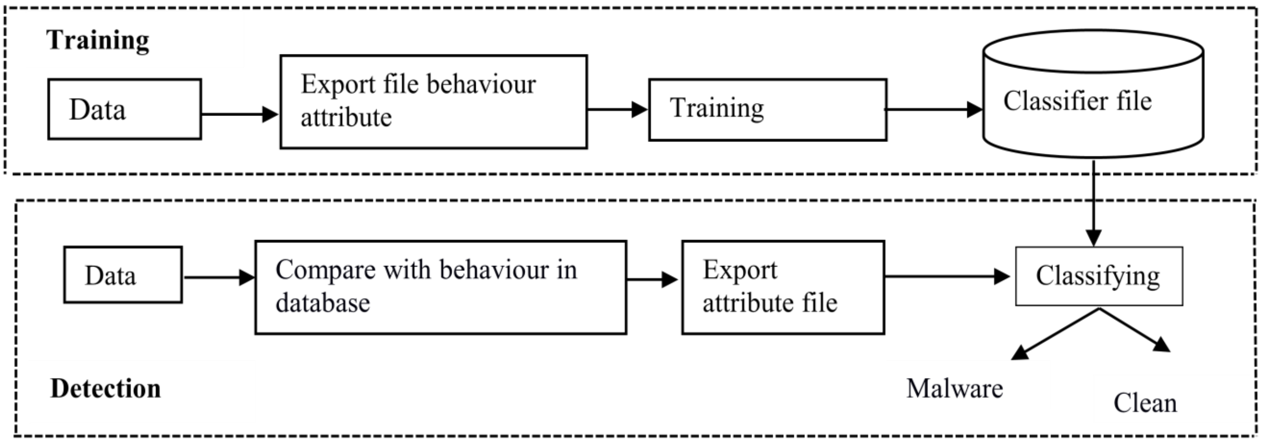

Fig. 3. The experimental model of malware detection using machine learning

Training and Learning phase: Detector tries to learn the behavior andfeature of malware. This the phase where the model was built for malware detection
Detection phase: Based on machine learning at phase 1, the detector will detect malware and send notification base on a machine learning algorithm.
Experiment Process:
- Start Cuckoo Sandbox like a service, wait for command via REST API.
- Use REST API Cuckoo Sandbox to upload non-malware data (document, images, audio, …) and run analysis on Cuckoo Sandbox.
- Use REST API Cuckoo Sandbox to upload entries malware and run analyses on Cuckoo Sandbox.
- Take log (report. jsonvàdump.pcap) of harmless data and malware.
- Export entries feature from file report and file dump (table 1).
- UseRandom Forest Classification algorithm and begin training and build the model.

4.2. Data and experiment process

a) Preparing data

- Benign data: [12]
Quantity: 7620 (files).
Document File: (Docx, Excel , PDF, Text, Ebook…).
Progamming File: (Java, Python, C, C++…).
ImageFile: (JPG, PNG…).
AudioFile: (MP3, WAV).
Exercutable File: (DLL, EXE).
- Malware: [13, 14]
Quantity: 13966 (files).
Malware File: (Virus, Adware, Malware…).

b) Experiment script

Export data (including harmless data and malware at the 80% – 20% ratio for testing).

c) Calculation parameter

Table 2. Measured values

Parameter	Notes	Calculationprocess
TP	True Positive – result of finding malware correctly	Count number of malware and it was corected (files with malware equal files with files labeled 'malware')
TN	True Negative – result of finding malware incorrectly	Count number of malware and it was incorrect (files with malware not equal files with files labeled 'malware')
FP	False Positive - result of finding harmless file correctly	Count number of harmless files and it was corected (files with harmless files equal files with files labeled 'malware')
FN	False Negative – result of finding malware incorrectly	Count number of malware and it was corected (files with malware equal files with files labeled 'malware')

Where:

$$acc = \frac{TP+TN}{TP+TN+FP+FN} \times 100\%$$

$$precision = \frac{TP}{TP+FP} \times 100\%$$

$$\mathrm{Re}call = \frac{TP}{TP+FN} \times 100\%$$

$$F1 = \frac{2 \times precision \times \mathrm{Re}call}{precision + \mathrm{Re}call}$$

d) Experimental script

Test case:
Test case A: 3000 (clean) + 5000(malware)
Test case B: 5000 (clean) + 8000(malware)
Test case C: 6500 (clean) + 11000(malware)
Test case D: 7620 (clean) + 13966(malware)

e) Experimental results

Table 3. Experimental results of malware detection using machine learning

Case	Test Error(%)	Accuracy (%)	Training time	Testing time (s)
A	25	75,0	9.2	0.17234
B	12,15	87,85	4.2	0.37738
C	4,41	95,59	16.9	0.69376
D	1,06	98,94	25.8	1.05794

From the experimental results in Table 3, the system shows the best detection result of 98.94% with scenario D when the data is complete and large.

However, with scenario D, the training and testing time will be much larger than other scenarios. Besides, the results give the lowest when with data set A 75%. With the above experimental results, it can be said that the malware detection system based on the properties we propose has brought high efficiency.

V. CONCLUSIONS

In this paper, we have proposed the model of the malware detection system based on the machine learning method. The experimental results in this paper have shown that we had a right and reasonable approach to detect and prevent malware. The innovation of our paper expresses not only in the use of machine learning algorithms to detect malware but only in the proposal to use features that aren't too complicated to calculate and extract but still being highly effective in detecting abnormal behavior of the process. However, it is easy to see that, to detect malware, a cumbersome and complex collection and extraction system are required. Besides, many currently advanced malwares are difficult for their behavior to be collected if it is based solely on the Sandbox tool. Therefore, in the future, it is necessary to have research to detect malware based on the processes they generate on the operating system

REFERENCES

1. A. Shabtai, R. Moskovitch, Y. Elovici, C. Glezer, Detection of malware by applying machine learning classifiers on static features: A stateof- the-art survey, Inf. Secur. Tech. Rep. 14 (1) (2009) 16–29.
2. M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, J. Nazario, Automated classification and analysis of internet malware, in: Recent advances in intrusion detection, Springer, 2007, pp. 178–197.
3. U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, E. Kirda, Scalable, behavior-based malware clustering, in: NDSS, Vol. 9, 2009, pp. 8–11.
4. K. Rieck, P. Trinius, C. Willems, T. Holz, Automatic analysis of malware behavior using machine learning, Journal of Computer Security 19 (4) (2011) 639–668.
5. S. Palahan, D. Babi´c, S. Chaudhuri, D. Kifer, Extraction of statistically significant malware behaviors, in: Computer Security Applications Conference, ACM, 2013, pp. 69–78.
6. M. Egele, M. Woo, P. Chapman, D. Brumley, Blanket execution: Dynamic similarity testing for program binaries and components, in: USENIX Security ’14, USENIX Association, San Diego, CA, 2014, pp. 303–317.
7. M. Lindorfer, C. Kolbitsch, P. M. Comparetti, Detecting environmentsensitive malware, in: Recent Advances in Intrusion Detection, Springer, 2011, pp. 338–357.
8. IMPORTANT INFORMATION REGARDING SANDBOXIE VERSIONS. https://www.sandboxie.com/. [Accessed February 15, 2020].
9. Smola, A.; Vishwanathan, S.V.N. Introduction to Machine Learning; Cambridge University Press: Cambridge, UK, 2008.
10. Daniele Ucci, Leonardo Aniello, Roberto Baldoni. Survey of Machine Learning Techniques for Malware Analysis. Computers & Security (2018), doi: https://doi.org/10.1016/j.cose.2018.11.001.
11. MehedyMasud, Latifur Khan, and BhavaniThuraisingham. Data Mining Tools for Malware Detection. CRC Press, 2011.
12. https://mp3.zing.vn. http://chinhphu.vn/portal/page/portal/chinhphu/trangchu
13. VirusShare. https://virusshare.com/. [Accessed February 15, 2020].
14. DAS MALWERK // malware samples. https://dasmalwerk.eu/?fbclid=IwAR1lI91cVexbTj09Qd449PO5y2zoSdq3SxJfxR3-8mxdn1MECA-W3rwtCsw. [Accessed February 15, 2020].
15. 2019 Internet Security Threat Report. https://resources.malwarebytes.com/files/2019/01/Malwarebytes-Labs-2019-State-of-Malware-Report-2.pdf[Accessed February 15, 2020].

AUTHORS PROFILE

- Profile Dr. **Do Xuan Cho** is currently a lecturer at the Faculty of Information Technology at Posts and Telecommunications Institute of Technology in Vietnam. In 2008, received a bachelor's degree in the Saint Petersburg Electrotechnical University "LETI" on a specialty "Computer science and computer facilities", Russia. In 2010, graduated a masters from the Saint Petersburg Electrotechnical University "LETI" on a specialty "Computer science and computer facilities", Russia. In 2013, received a PhD in the Saint Petersburg Electrotechnical University "LETI", on a specialty CAD. Russia. Area of scientific interests - modeling, control systems, algorithmization.

Email: chodx@fe.edu.vn

- Authors Do Minh Tuan, Nguyen The Lam, Nguyen Anh Tuan are fourth-year students majoring in information security at FPT University. These students have over 2 years of experience working with APT attack detection issues.

Email: ***tuandmse05518@fpt.edu.vn***, ***lamntse63326@fpt.edu.vn***, ***tuannase62864@fpt.edu.vn***

Second Author: My position is the professor of Institute of computer sciences and technologies in Peter the Great Saint-Petersburg Polytechnic Ubiversity. I have received the degree Doctor of Technical Sciences in 1998 in accordance of scientific speciality "Systems of automatic Desing" in SPbPY. The area of scintific interest is use of new type of fuzzy logics in different applications. I think that we could cooperate intensively in future.

Email: ***v_tisenko@mail.ru***