

## Obfuscated computer virus detection using machine learning algorithm

Tan Hui Xin<sup>1</sup>, Ismahani Ismail<sup>2</sup>, Ban Mohammed Khammas<sup>3</sup>

<sup>1,2</sup>Department of Electrical and Computer Engineering, University Teknologi Malaysia, Malaysia

<sup>3</sup>Department of Network Engineering, Al-Nahrain University, Iraq

---

### Article Info

#### Article history:

Received Mar 29, 2019

Revised Jun 16, 2019

Accepted Jul 5, 2019

---

#### Keywords:

Machine learning

Obfuscated computer virus

Signature based

SMO classifier model

String features

---

### ABSTRACT

Nowadays, computer virus attacks are getting very advanced. New obfuscated computer virus created by computer virus writers will generate a new shape of computer virus automatically for every single iteration and download. This constantly evolving computer virus has caused significant threat to information security of computer users, organizations and even government. However, signature based detection technique which is used by the conventional anti-computer virus software in the market fails to identify it as signatures are unavailable. This research proposed an alternative approach to the traditional signature based detection method and investigated the use of machine learning technique for obfuscated computer virus detection. In this work, text strings are used and have been extracted from virus program codes as the features to generate a suitable classifier model that can correctly classify obfuscated virus files. Text string feature is used as it is informative and potentially only use small amount of memory space. Results show that unknown files can be correctly classified with 99.5% accuracy using SMO classifier model. Thus, it is believed that current computer virus defense can be strengthening through machine learning approach.

Copyright © 2019 Institute of Advanced Engineering and Science.  
All rights reserved.

---

### Corresponding Author:

Tan Hui Xin,  
Department of Electrical and Computer Engineering,  
University Teknologi Malaysia,  
81310 Skudai, Johor Bahru, Johor, Malaysia.  
Email: hxtan95@gmail.com

---

## 1. INTRODUCTION

Malicious software (malware) is specifically designed by computer virus writers to exploit vulnerabilities and infect the PC without the users' consent. It is commonly used by the hackers to steal confidential user information for illegal purposes [1]. Malware including computer viruses, worms and trojans can be easily transported over network by email or through removable media. When users visit websites containing exploit code, they may accidentally download and install malware into their PC [2]. Nowadays, computer virus has become more complex as new computer virus variants are constantly evolved from old computer virus and their internal functions are modified across generations [3]. This has made the computer virus detection process very difficult. Obfuscated computer virus such as metamorphic and polymorphic are created to bypass the most popular computer virus detection approach which is signature based method. This technique identifies unique sequences of bytes or strings as signature, just like a fingerprint of the virus [4]. However, this kind of computer virus does not contain a specific signature in the code. It will just increase the database list with unnecessary signatures when storing new signatures of each computer virus variant into the database.

To overcome the limitation of signature based detection method, machine learning is proposed to detect obfuscated computer virus. This has been proven by C. Richardson [3] which applied machine

learning technique in detecting unknown computer virus. Machine learning (ML) is generally focused on learning from past experience to classify and predict future data based on the similarity with the training data. Basically, this approach will need to do some data mining, statistical analysis and data aggregation (classification). In this paper, text strings are used and have been extracted from virus program codes as the features. It was believed that text strings can retrieve full information of the program while using small amount of memory address, which results in faster runtime. Therefore, this technique can effectively detect unknown computer virus based on their pattern similarities.

## 2. LITERATURE REVIEW

### 2.1. Overview

Computer virus detection is very important as it is generally considered as the first step in computer virus defense. As shown in Figure 1, there are 3 types of computer virus detection analysis, which are static, dynamic and hybrid analysis. For this work, it can be categorized into static analysis and the similarities of patterns are analyzed through downloaded files [5-6]. Static analysis is done by analyzing the source code generated by reverse engineering tools like disassemblers or debuggers to study how the computer virus operates [7]. It can be divided into two: signature based and non-signature based. Signature based method is one of the most popular way to detect computer virus in the world. It relies on the identification of unique sequences of bytes (strings) as signature, just like a fingerprint of the virus [4]. While for non-signature based, one of the approaches exists is ML technique. It detects computer viruses by observing the patterns of the computer virus, extract them as features and train the features to generate classifier models. Various kinds of classification models can be generated using different ML algorithms for example Naïve Bayes, Sequential Minimal Optimization (SMO) and J48. Naïve Bayes algorithm uses probability of every feature independently and makes prediction using Bayes Theorem. The advantage of this method is the irrelevant features which have low probability will not be taken into consideration when making prediction [8]. Sequential Minimal Optimization (SMO) is widely used in Support Vector Machine (SVM) training to solve for quadratic programming optimization problem. It can handle a large size of training sets as memory needed is linear in the training set size [9]. J48 uses training sets for creating decision tree and makes prediction based on the tree. It is quite popular to be used due to its simplicity and the “open” decision making process that clearly shows users which decisions lead to the outcome [8].

Dynamic analysis is based on the information retrieved from the operating system during the execution of the program [10]. The program is usually executed and monitored either in a real or virtual machine environment such as in the sandbox and its behavior is being observed [11]. It can be much faster and accurate, while giving control and data flow information [11]. However, it is difficult to have suitable conditions where the malicious code will be triggered or activated other than to identify the time required for observing the appearance of malicious activity [10]. Hybrid analysis is the combination of static and dynamic analysis. Siddiqui [12] had proposed an anomaly based technique where hybrid analysis was used to detect obfuscated code. In this technique, static analysis was used to locate the system calls in the program while the program is dynamically monitored later. Islam and Altas [13] had used three different feature set in their study. API parameters are used as dynamic features whereas function length frequency and printable string information as static features. Using 2398 malware executable files, they had tested the experiments using four different ML algorithms such as Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF), Instance-based with boosting techniques (IB1) and they found out that RF shows the best performance in classifying the data.

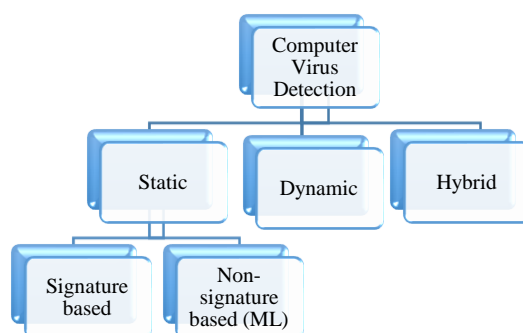


Figure 1. Overview of computer virus detection analysis

## 2.2. Related works

Obfuscated computer virus uses a combination of various code obfuscation techniques to avoid detection, or example Garbage Code Insertion. The obfuscated virus will keep inserting trash instructions like NOP at random places in the code to make the virus body looks different in each generation [14]. In other words, a single computer virus can produce thousands of new variants with different fingerprints [15]. Therefore, storing signatures of each computer virus variant is not practical as it just increase the database list with unnecessary signatures.

Schultz et al. [16] were the pioneers to introduce ML in computer virus field using binary codes to detect new malware. Extracted system resource information, strings and byte sequences from malicious executable are used as features. Byte code or byte sequence in form of n-gram has been used in ML classification [17-18]. However, due to the large data size, more features are created, causing memory overhead and slowing down the runtime of the classification process [8].

In order to solve the limitation of byte code, opcode (short for operational code) is introduced to improve the performance. It is retrieved from the assembly format which consists of opcode and operands [19]. The operation of opcode can be arithmetic, program control, logical operations and data manipulation [10]. There are two main techniques to perform disassembling process from executable into assembly codes such as linear sweep and recursive traversal algorithm. According to Ahmadi et al. [20], recursive traversal algorithm is less susceptible to mistakes as the code is disassembled based on jump and branch instructions. For example, Interactive Disassembler (IDA Pro) tool [21] utilizes cross-reference between code sections, parameters of API calls and other information to perform automatic code analysis on binary files. As the model is trained on disassembled virus executables, the quality of the disassembler may affects the results [22]. Other than that, since it just retrieves part of the program, it may miss some important information of malicious code [23]. But the computation time is faster as the data size is smaller.

In this project, we extract text strings as the features in machine learning classification. It can retrieve full information of the program while using smaller amount of memory address, which results in faster runtime. WEKA [24] is used as the workbench for developing our approach and doing the analysis on computer virus detection. However, as both opcode and string are using assembly format dataset, they may have the risk that some executable files cannot be disassembled properly. Figure 2 shows an overview for static analysis techniques with different type of features.

Machine Learning (ML)			
Features	Byte Code	Opcode	String
Data Size	Large	Small	Medium
Memory Overhead	Large	Small	Small
Runtime	Slow	Fast	Fast
Information retrieved	Full	Part	Full
Risk issue	Computation time is longer when dataset is big.	Some executable files cannot be disassembled properly.	Some executable files cannot be disassembled properly.

**Signature based**

- relies on the identification of unique hex code strings
- need download latest signature updates
- very precise, but useless against unknown malicious code

Figure 2. Overview of signature based and non-signature based detection method (ML)

## 3. RESEARCH METHOD

### 3.1. Pre-processing data

Figure 3 shows the data processing flow of datasets before analysing using machine technique. First, computer virus collections are collected from online computer virus repository such as VXHeaven [5] and Das Malwerk [6]. While assembly codes are collected from assembly tutorials websites and Windows Executable files as normal collections. The distribution of raw datasets collected is shown in Table 1. Computer virus collections will be used as training set and some of them will be obfuscated and be used as testing set. Since string feature is used, these executable files were disassembled into assembly codes using IDA Pro Disassembler Software [21] for the ease of feature extraction as shown in Figure 4.

Perl script is used to filter unwanted information in the assembly codes. This script will read all assembly files in a directory and save as text format with the same name in another directory. When processing a file, it will remove all comments, empty spaces or lines and replace multiple spaces with a

single space. Every processed file will contain sequence of strings and each string is gapped with a single space as shown in Figure 5.

Table 1. Distribution of raw datasets

Raw Datasets	Total
Virus sets	150
Normal sets	150

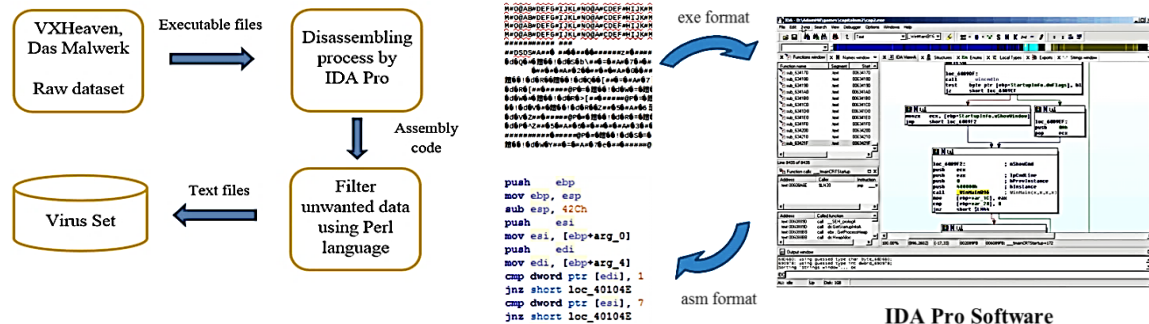


Figure 3. Data processing before entering machine learning tool

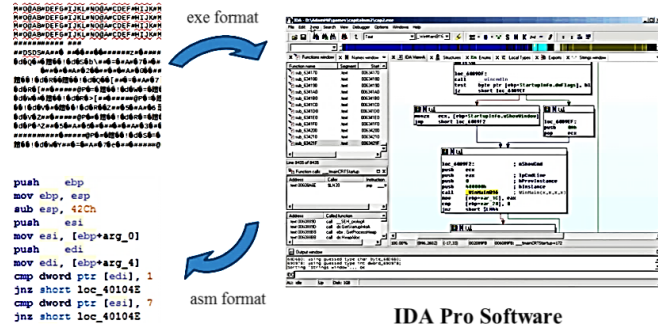


Figure 4. Disassembling process

### 3.2. Learning phase

According to Figure 5, it shows the process taken to train the classifier models with training sets. A total of 300 datasets equally divided by both computer virus and normal files are used as the training sets. Using some functions in ML tool [24] i.e TextDirectoryLoader and StringToWordVector, Attribute-Relation File Format (ARFF) file is created both for training and testing set. By using the generated ARFF file, ML algorithms such as Naïve Bayes, SMO and J48 can train and generate different types of classifier models.

### 3.3. Classification phase

The flow for classification phase in machine learning is shown in Figure 6. The performance of these classifier models can be tested with cross-validation method. In this work, 10 fold cross-validation is chosen. Basically, this technique will split the training sets into several partitions according to the number of folds. For this case, the classifier models will be tested by setting the first fold as test set and the remaining nine folds as training set. Then, the same process is done by using the next fold as test set until tenth fold. Based on the similarity in its features with the trained model, each classifier model will predict which class the unknown dataset belongs to. The best classifier model will then be chosen after analysing the performance for different classifier models.

To test the functionality of the system on obfuscated computer virus, OWASP-ZSC Obfuscated Code Generator tool [25] is used to obfuscate 25 viruses from the training sets. These obfuscated virus files are processed and filtered into text files using the same approach as the pre-process of virus set previously. When the virus set is obfuscated, the file size will become bigger. This means that some exploits or garbage codes have been thrown into the code during the obfuscation process. By using WinMerge [26], the difference of virus file before and after obfuscation can be seen in assembly format as shown in Figure 7. A big paragraph of garbage code or exploits has been added into the obfuscated virus file at the left side compared to the original virus file at the right side. Then, a collection of unknown datasets consist of virus sets, obfuscated virus sets and normal sets are keyed into machine learning as test sets as shown in Figure 6. The performance of the proposed system is analysed to see whether it has improved the computer virus detection compared to the traditional signature based method from VirusTotal [27].

### 3.3. Evaluation metrics

Using ML tool [24], the performance of each generated classifier model is evaluated through four metrics. True Positives (TP) represents the correctly identified virus programs while True Negatives (TN) represents the correctly identified normal programs. False Positives (FP) represents the normal programs that were incorrectly identified as virus programs while False Negatives (FN) represents the virus programs that were incorrectly identified as normal programs. Accuracy (ACC) is the percentage of correctly identified

programs. The concern for TP here is that for computer security, it should not be compromised with virus files that can spread to the systems. FP also plays an important role in our verification because we need to reduce the number of normal files that were wrongly classified as viruses. As conclusion, high TP and less FP (high TN) indicate good performance for the algorithms

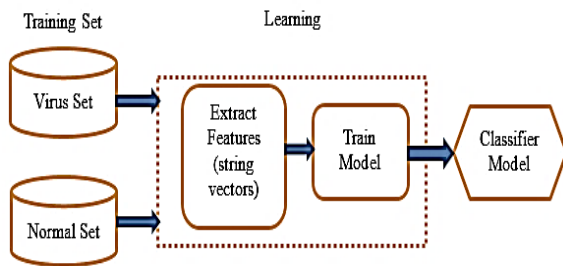


Figure 5. Learning phase in ML

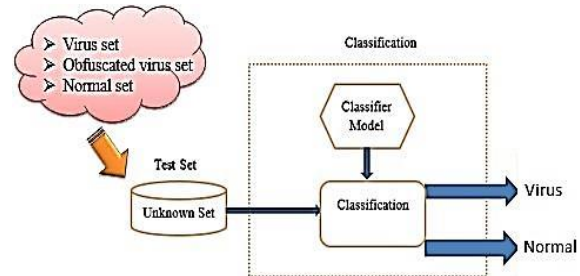


Figure 6. Classification phase in ML

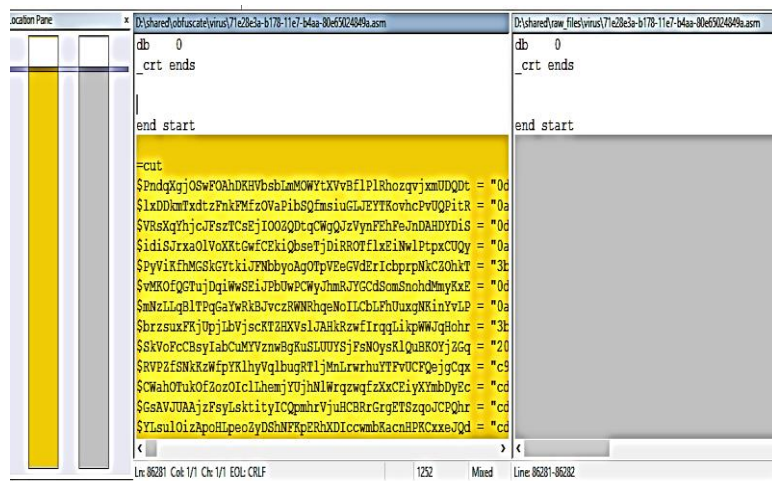


Figure 7. Comparison of virus files after and before obfuscation

#### 4. RESULTS AND ANALYSIS

Figure 8 shows the virus file set obtained after doing the data filtering using Perl script. Compared to the normal set, the content of virus set is much larger which causes more memory size is needed. Besides, virus set consists a lot of special character like “?” when allocate memory space for initialized data in the program. Each processed text files contain strings that gapped with a single space. This will help to differentiate the string features and extract them out when the files are fetched into the ML tool.

```

dd ? wShowWindow dd ? cbReserved2 dd ? lpReserved2 dd ? hStdInput dd ? hStdOutput dd :
cbClsExtra dd ? cbWndExtra dd ? hInstance dd ? hIcon dd ? hCursor dd ? hbrBackground ?
message dd ? wParam dd ? lParam dd ? time dd ? pt POINT ? MSG ends POINT @hex x dd ?
? tagMSG ends PAINTSTRUCT @hex hdd dd ? fErase dd ? rcPaint RECT ? fRestore dd ? fIn
bottom dd ? RECT ends tagPAINTSTRUCT @hex hdd dd ? fErase dd ? rcPaint RECT ? fResto
page, string, zero ixxc c, <string> db 'c', page endm ifnb <zero> dd zero endif endm .6f
aa:nothing, da:_data, fa:nothing, gs:nothing align 10h sub 401010 pxxc near var_42C=
pff 8 arg_4= dhexd pff 0Ch push ebp mov ebp, esp sub esp, 42Ch push esi mov esi, [ebp
[esi], 7 jnz short loc_40104E mov eax, hWnd push 8 push 0 push 4DBh push eax mov dhexd
loc_40104E: cwp dword_455EF4, 0 jz short loc_4010BB mov ecx, dword_455F0C push ecx lea
dword_455EEC mov ecx, [esi] inc ecx imul ecx, ecx add esp, 0Ch mov dword_455EEC, eax
[ebp+var_42C] push eax mov [ebp+var_20], ecx mov edx, dword_455EF4 push 1061h push edi
edi pop esi mov esp, ebp pop ebp ret sub 401010 endp align 10h dd 48C48B48h, 5510508!
6883F98Bh, 0D23300C0h, 0D04D8D48h dd 315215FFh, 48900006h, 400A158Dh, 8D480009h push
inc ecx jmp short loc_401172 dd 0D04Dh dd 328015FFh, 0B8410006h, 80070057h, 48D68B48h
dd 4D8D4800h, 0CB15FFD0h, 84000630h, 481274C0h dd 48D0558Dh, 15FFCB8Bh, 6322Ah db 0E9h
inc ecx cwp eax, ecx jnz short loc_401188 dec eax jz short loc_40118A inc eax jmp sho
cl dec ebp sar bh, 1 add eax, 6307Dh dec eax mov edx, eax dec eax mov ecx, ebx call d
dec eax mov eax, ebx dec eax mov ebx, [esp+80h] dec eax add esp, 60h pop edi pop esi
[edx+48h] dec eax jmp dhexd pff da:62FF8h inc eax push ebp dec eax sub esp, 20h dec ei

```

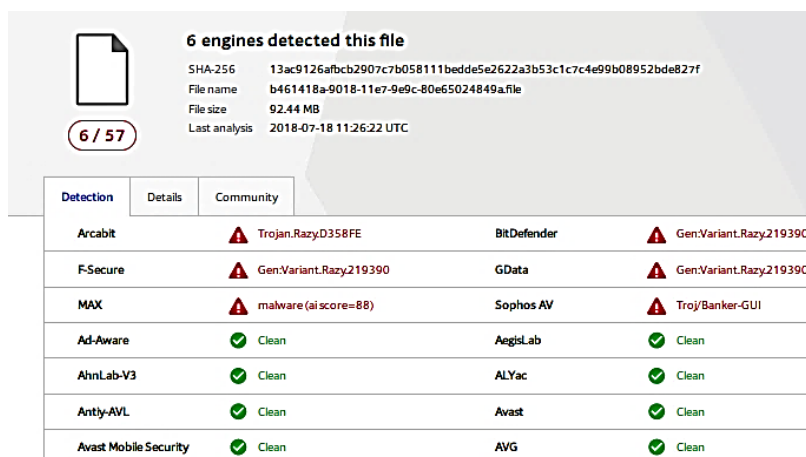
Figure 8. Processed virus set

During the learning phase, the performance factors for respective classifier models are extracted and listed out as shown in Table 2. Based on the experiment, SMO classifier model shows the best performance, which achieves 98% accuracy in classification. Besides, the overall time taken to build model and model testing on training data is ranked at first among all classifier models. Therefore, SMO classifier model is chosen as the most suitable model to undergo obfuscated computer virus detection

Table 2. Performance factors of classifier models

Classifier model	Time to generate model (s)	Testing time (s)	Acc (%)
Naïve Bayes	0.05	0.19	94.33
SMO	0.08	0.08	98
J48	0.16	0.01	96.67

In this work, 25 virus executable files are extracted from the training set to undergo obfuscation process using Obfuscated Code Generator Tool [25]. These files are uploaded to VirusTotal website [27] to check whether they can avoid the detection from anti-virus softwares which use signature based detection method. After obfuscation, these files are considered as unknown files by VirusTotal and it will generate a new SHA-256 for the files as shown in Figure 9.



6 engines detected this file			
SHA-256	13ac9126afbcb2907c7b058111bedde5e2622a3b53c1c7c4e99b08952bde827f		
File name	b461418a-9018-11e7-9e9c-80e65024849a.file		
File size	92.44 MB		
Last analysis	2018-07-18 11:26:22 UTC		
Detection	Details	Community	
Arcabit	Trojan.Razy.D358FE	BitDefender	Gen:Variant.Razy.219390
F-Secure	Gen:Variant.Razy.219390	GData	Gen:Variant.Razy.219390
MAX	malware (ai score=88)	Sophos AV	Troj/Banker-GUI
Ad-Aware	Clean	AegisLab	Clean
AhnLab-V3	Clean	ALYac	Clean
Antiy-AVL	Clean	Avast	Clean
Avast Mobile Security	Clean	AVG	Clean

Figure 9. Result from VirusTotal for a virus file (after obfuscate)

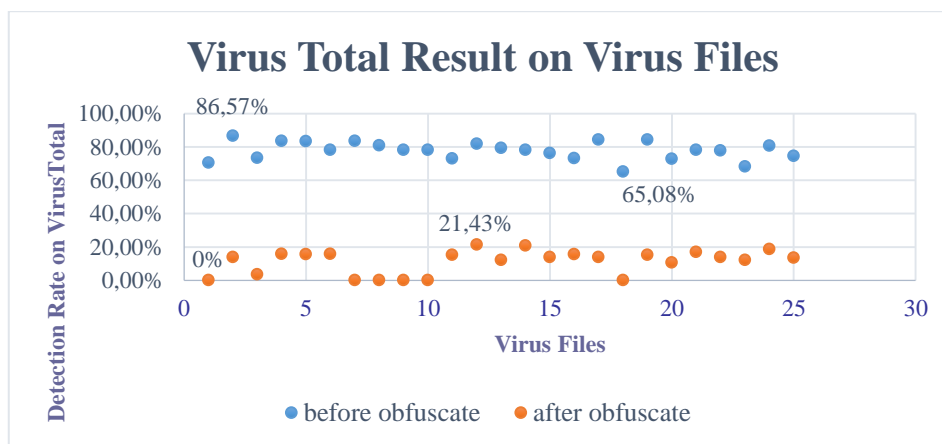


Figure 10. Virus total result on virus files



Next, the results from VirusTotal [27] for 25 virus files before and after obfuscation are analyzed and compared as shown in Figure 10. These results can be obtained using (1). Before the obfuscation, the highest percentage for anti-virus engines that can detect it as virus file is 86.57% and the lowest is 65.08%. It can be seen clearly that the average percentage for virus sets before obfuscate to be detected is quite high since VirusTotal which use signature based detection method can detect the signature in the virus files. However, the signature is missing once the virus files are obfuscated, which will cause a significant decrease in computer virus detection. Among 25 obfuscated virus sets, the highest anti-virus engines that consider the file harmful is only 12 out of 56 (21.43%) while some of the virus files are even successfully avoid the detection and being detected as normal or safe files. This means that the conventional anti-virus engines in the market may have the risk to pass through obfuscated virus and infect the PC without users' consent.

$$\text{Detection rate on VirusTotal} = \frac{\text{No.of engines that detect it as virus}}{\text{Total engines that review this file}} \quad (1)$$

Then, 5 test sets are created with a mixture of normal sets, virus sets and obfuscated virus sets. In each test set, it consists of 20 normal files, 15 virus files and 5 obfuscated virus files. By using ML tool [24], the classification result of test set 1 on SMO classifier model is shown in Figure 11. Same approach is done for the remaining 4 test sets and the classification results are analyzed.

```

Summary
Correctly Classified Instances      40      100 %
Incorrectly Classified Instances    0        0 %
Kappa statistic                    1
Mean absolute error                 0
Root mean squared error            0
Total Number of Instances          40

Detailed Accuracy By Class
      TP Rate  FP Rate  Precision  Recall  F-Measure  MCC  ROC Area  PRC Area  Class
      1.000    0.000    1.000    1.000    1.000    1.000  1.000    1.000    normal
      1.000    0.000    1.000    1.000    1.000    1.000  1.000    1.000    virus
Weighted Avg.    1.000    0.000    1.000    1.000    1.000    1.000  1.000    1.000

Confusion Matrix
a b <-- classified as
20 0 | a = normal
0 20 | b = virus

```

Figure 11. Summary test set 1 classification result on SMO classifier model

According to the confusion matrix in Figure 11, it shows that there are 20 correctly identified instances for normal files (TN) and 20 correctly identified instances for computer virus (TP). This means that SMO classifier model can correctly classify all the instances with 100% accuracy even though the test set consists of obfuscated virus files. Same approach is done for the remaining test sets and the classification results are listed in Table 3. From the table, it can be seen that most of the test sets with obfuscated computer virus can be classified well. From this result, there is only 1 out of 25 obfuscated virus files wrongly identified as normal based on the 5 test sets, which comes out with 99.5% accuracy in average.

Table 3. Classification result on SMO classifier model

Test set	TP	TN	ACC (%)
1	20	20	100
2	20	20	100
3	19	20	97.5
4	20	20	100
5	20	20	100
Average			99.5

## 5. CONCLUSION

As a conclusion, an obfuscated computer virus detection system has been developed using machine learning technique using strings as features. In this work, various classifier models have been generated and their performances are analysed and compared. The results show that SMO classifier model can classify the

obfuscated computer virus well with the promising performance. This method can compensate and overcome the limitation of signature based computer virus detection and help to fight the rising of obfuscated computer virus which has brought up many serious cases nowadays like Ransomware Attack. To further improve the system, heap memory size can be adjusted so that more training sets can be used to train the classifier model. Choosing a better obfuscated code generator tool may also help to generate smaller obfuscated virus files and reduce memory overhead. In future, it can be used as the groundwork for the real-time implementation of computer virus detection at hardware level i.e Field Programmable Gate Array (FPGA).

## REFERENCES

- [1] M. F. Zolkipli and A. Jantan “Malware Behavior Analysis: Learning and Understanding Current Malware Threats”, *Second International Conference on Network Applications Protocols and Services*, 2010, pp. 218-221.
- [2] Q. Zhang, “Polymorphic and Metamorphic Malware Detection”, Technical Report, ProQuest Dissertations Publishing, 2008
- [3] C. Richardson, “Virus Detection with Machine Learning”, Master’s Thesis, Department of Computer Science, University of BRISTOL, 2009
- [4] S. M. Sridhara, *Metamorphic Worm That Carries Its Own Morphing Engine*, Master’s Projects, paper 240, 2012.
- [5] VXHeaven, <http://vxer.org/vl.php>. [Retrieved on 12 Oct 2017].
- [6] Das Malwerk, <http://dasmalwerk.eu/>. [Retrieved on 12 Oct 2017].
- [7] A. Pratheema and P. Kavitha, “Malware Classification through HEX Conversion and Mining,” *International Conference on EGovernance & Cloud Computing Sevices*, 2012
- [8] K. Chumachenko, “Machine Learning Methods for Malware Detection and Classification”, Bachelor’s Thesis, University of Applied Sciences, 2017.
- [9] S. Singaravelan, D. Murugan and R. Mayakrishnan, “Analysis of Classification Algorithms J48 and SMO on Different Datasets,” *World Engineering & Applied Sciences Journal*, 6 (2): pp. 119-123, 2015.
- [10] A. Shabtai, R. Moskovitch, C. Feher, S. Dolev and Y. Elovici, “Detecting Unknown Malicious Code by Applying Classification Techniques on OpCode Patterns”, *Security Informatics*, 2012.
- [11] B. D. Dati, “A Detector of Metamorphic Viruses based on the Analysis of Instructions’ Distribution”, Technical Report, Università degli Studi del Sannio, Facoltà di Ingegneria, 2011/2012.
- [12] M. A. Siddiqui, *Data Mining Methods for Malware Detection*, PhD’s Dissertation, University of Central Florida, 2008.
- [13] R. Islam and I. Altas, “A Comparative Study of Malware Family Classification”, *14th International Conference on Information and Communications Security*, 2012.
- [14] M. E. Saleh, A. B. Mohamed and A. A. Nabi, “Eigenviruses for Metamorphic Virus Recognition”, *IET Information Security*, 2010.
- [15] A. Govindaraju, “Exhaustive Statistical Analysis for Detection of Metamorphic Malware”, Master’s Projects, paper 66, 2010.
- [16] M. G. Schultz, E. Eskin, F. Zadok and S. J. Stolfo, “Data mining methods for detection of new malicious executables,” *Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001*, Oakland, CA, USA, 2001, pp. 38-49.
- [17] I. Ismail, M. N. Marsono and S. M. Nor. “Detecting Worms Using Data Mining Techniques: Learning in the Presence of Class Noise”, *Sixth International Conference on Signal-Image Technology and Internet Based Systems*, 2010, pp. 187-194
- [18] Ban Mohammed Khammas et al., “Feature Selection and Machine Learning Classification for Malware Detection”, *Jurnal Teknologi*, vol. 77, no. 1, 2015.
- [19] R. K. Shahzad and N. Lavesson, “Detecting Scareware by Mining Variable Length Instruction Sequences”, *Information Security for South Africa*, 2011
- [20] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov and G. Giacinto, “Novel Feature Extraction, Selection and Fusion for Effective Malware Family Classification”, *CODASPY*, 2016.
- [21] IDA Pro Diassamber, [http://download.cnet.com/IDA-Pro/3000-2218\\_4-10800676.html](http://download.cnet.com/IDA-Pro/3000-2218_4-10800676.html). [Retrieved on 2 Nov 2017].
- [22] W. Wong, “Analysis and Detection of Metamorphic Computer Viruses”, Master’s Thesis, Department of Computer Science, San Jose State University, 2006.
- [23] R. K. Shahzad, N. Lavesson and H. Johnson, Accurate Adware “Detection using Opcode Sequence Extraction”, *Sixth International Conference on Availability, Reliability and Security*, 2011.
- [24] WEKA, <https://www.cs.waikato.ac.nz/ml/weka/index.html>. [Retrieved on 10 Oct 2017].
- [25] OWASP ZSC–Obfuscated Code Generator Tool, <https://www.darknet.org.uk/2018/01/owasp-zsc-obfuscated-code-generator-tool/>. [Retrieved on 11 Apr 2018].
- [26] WinMerge, <http://winmerge.org/>. [Retrieved on 15 May 2018].
- [27] Virus Total, <https://www.virustotal.com>. [Retrieved on 12 Oct 2017].



**BIOGRAPHIES OF AUTHORS**

Tan Hui Xin is a student from University Teknologi Malaysia, Malaysia under Electrical and Electronic programme.



Ismahani Binti Ismail received the Ph.D degree in Electrical and Electronic Engineering from University Teknologi Malaysia, Malaysia in 2013. She is currently a senior lecturer in University Teknologi Malaysia, Malaysia under Department of Electrical Electronic and Computer engineering. She works in network algorithmics, digital system and FPGA implementation.



Ban Mohammed Khammas received the Ph.D degree in Electrical and Electronic Engineering from University Teknologi Malaysia, Malaysia in 2017. She is currently a lecturer in Collage of Information Engineering, Al-Nahrain University, Iraq under Department of Network Engineering. She works in neural network and FPGA implementation