Indian Journal of Science and Technology, *Vol 12(25), DOI: 10.17485/ijst/2019/v12i25/146005, July 2019*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

N/A and Signature Analysis for Malwares Detection and Removal

Ahmad Ridha Jawad[1*], Khaironi Yatim Sharif[1] and Ammar Khalel Abdulsada[2]

[1]Universiti Putra Malaysia, Faculty of Computer Science and Information Technology. Jalan UPM, 43400 Serdang, Selangor, Malaysia; ahmad.rida10000@gmail.com, khaironi@upm.edu.my
[2]Department of Computer Science, College of Education, University of Kufa, Kufa City, Najaf, Iraq; ammar.khaleel@uokufa.edu.iq.

Abstract

Objectives: This study aimed to design an application that effectively scans, detects, and removes malware based on their signatures and behaviours. **Methods/Statistical analysis**: The rapid growth in the number and types of malware poses high security risks despite the numerous antivirus softwares with Signature-Based Detection (SBD) method. The SBD method depends on the signatures or malware names that are available in the algorithm database. **Findings:** Malware is a type of malicious software that poses security threats to the targeted system, resulting in information loss, resource abuse, or system damage. The antivirus software is one of the most commonly used security tools to detect and remove malware. However, the malware defences should focus on the malware signatures since there is no universal way of recognising all malware. Therefore, this study suggested N/A detection technique as the dynamic method (behaviour-based detection method) that depends on the Windows Registry (system database). Both static and dynamic detection methods were assessed in this study. Based on the experimental outcomes, SBD method detected and removed most of malware (only known viruses). **Application/Improvements:** Meanwhile, the N/A detection method detected and removed all injected malware (known and unknown Trojan horse) within a relatively low running time.

Keywords: Dynamic Method, Malicious Software, Malware Detection, Signature Analysis, Static Method

1. Introduction

A malware is a type of "software" with "malicious" intent which poses major security threat to Internet users today. New viruses can quietly evade the antivirus detector through code misperception where the damaging content can be hidden[1]. The Internet threats such as denial-of-service attacks and huge spam emails through botnets are increasingly extensive. In order to prevent malware infection, Internet security specialists employ various approaches and skills to display the details of malware action[2]. However, the detection of malicious software is a challenging and daunting task for Information Technology (IT) managers and system operators, especially with an ever-increasing volume of malicious programmes and tools. Antivirus software is typically used to detect and suspend malicious files. The malware defences should focus on the malware signatures since there is no single way of recognising all malware[3].

In particular, this study, which employed both static and dynamic detection methods, aimed to design an application with the capacity to effectively scan, detect, and remove malware based on specific malware signatures and behaviours. Accordingly, the static detection method focuses on malware signatures with a static detection procedure that identifies malware by inspecting the record or files for any incidences of malware without running the programme. Meanwhile, the dynamic detection method focuses on malware behaviours. Basically, this type of method evaluates whether a programme is consecutively

*Author for correspondence

infected by a malware code and subsequently detects its behaviour. With that, this study proposed an application with signature-based malware detection (specifically for viruses) and behaviour-based malware detection (specifically for Trojan horse) in scanning, detecting, and removing various types of malware.

2. Background of Study

The following subsections describe static detection method (signature-based detection method) and dynamic detection method (behaviour-based detection method).

2.1 Static Detection Method

As briefly described earlier, the static detection method identifies malware by inspecting the content or files for any incidences of malware without running the programme. It employs diverse techniques and tools to quickly discover whether a file is malicious or not, offer information about its action and collect technical pointers to create simple signatures. Also signature method is distinctive identification for a binary file, which is made by analyzing the binary file using static analysis methods.

The static detection method scans, detects, and removes malware based on their corresponding signatures. This method is commonly used to detect malware that threaten the security of computer networks and computing systems. referring to[4]. Apart from the data mining, machine learning, and other heuristic answers for malware detection, another method was recently introduced, namely the extraction of opcodes. The algorithms were applied as part of the feature determination method to minimise the number of features. Based on the experimental outcomes, the model detected malware with sensitivity of approximately 98% and precision of approximately 99%. The researchers have database that contain a recognized viruses types of regulations. Once a file is scanned the signature based detection works on compares the series of symbols that showed in files with recognized viruses' types that saved in database. If the signature based detection algorithm discover a match then the algorithm states the file is a virus. Observe that signature detection algorithm relies on the database of recognized viruses types. The database was made by analyzing recognized viruses, by taking out series of instructions showed in them and deleting any series from them that are usual of precious programs. Many kinds of malware have been detected using graph mining method utilizing static analysis, though covering the current faults. The researchers suggested Minimal Contrast Frequent Sub graph Miner (MCFSM) algorithm, as new method for taking out minimal distinctive and commonly pernicious behavioral types that can detect accurately all family of pernicious programs, in compare to another set of precious programs. A high detection values and low false affirmative values have been showed by MCFSM method and produce a finite number of behavioral virus[5]. According to[6] a number of wide security gaps were discovered through the signature-based malware detection methods of various general commercial antivirus tools. Their methods created a substantial number of obfuscated types of known viruses that were verified on numerous antivirus software which, reaffirmed that these methods encountered critical inadequacy in detecting these viruses. This highlights the pertinent need to establish an algorithm that creates malware signatures used in much antivirus software. The outcomes prove that these tools are severely lacking in their ability to detect obscured forms of known viruses.

2.2 Dynamic Detection Method

As briefly introduced in the earlier section, the dynamic or behaviour-based detection method determines whether the specific file or program is consecutively infected by a malware code prior to distinguishing its behaviour. Dynamic analysis essentially turns malware to detect its behavior, find technical indicators, and understand its functionality which can be utilized in detection signatures.

An automatic detection technique based on the graph mining method was previously employed[1]. The maximal periodic sub graphs in the list of code charts that represented common behaviours with accurate specifications in the implementation files were removed and used as attributes for the generation of semantic signs. Based on the experimental outcomes, the employed technique extracted an incomplete number of stimulating features and achieved active malware detection. Meanwhile, the extraction of opcodes from the decompiled runnable program can also be executed to detect malware. Nowadays, the opcode series are mined using text-based approaches, but the extracted face sequences are incapable of performing the appropriate characterisation of behaviours for a runnable (malicious) programme. Hence, a control

flow-based procedure to extract these runnable opcode behaviours was employed to address this restriction[7]. With that, the mined behaviours can fully reflect the behavioural characteristics of a runnable programme. Moreover, the study also performed two types of opcode behaviour inspection methods to validate the efficiency of the rule flow-based behaviours, which revealed higher overall accuracy and lesser false positive result for the proposed control flow-based behaviour.

Nevertheless, the extensive use of antivirus software with SBD method remains incapable to reduce the increasing number and types of malware. Although the SBD method distinctively identifies any two similar files using static analysis, there is also the dynamic or behaviour-based detection method that determines whether a runnable file is malware-infected. Both of these methods have their own strengths and limitations. With that, an integration of static and dynamic methods was recommended to scan unknown runnable files for malware[8]. Additionally, the machine learning was incorporated where recognised malware and benign platforms are used as training data. The attribute vector was elected by analysing any two similar codes of dynamic behaviour. With the strengths of both methods, the classification and efficiency results were obtained with an accuracy of 97.1% for the dynamic method and an accuracy of 95.8% for the static method.

Considering the significance of malware detection in replicated security, the proposed solutions are expected to be efficient, precise, and robust, which propelled the need to perform malware detection in two stages through a hardware-assisted manner. Accordingly, the attack model of malware was learned through the procedure of "Deterministic Finite Automaton" (DFA) in an offline phase. Meanwhile, the DFA-based detection method was employed during the execution phase to examine whether the implementation file exhibits malicious behaviour using the real world data of 168 Linux malware examples and 370 benign examples. As a result, another malware of the same family was identified with the possibility of detecting zero-day attacks. Applied in hardware, actual time detection with resource overhead, low performance and more importantly offered by their architecture. Moreover, A malware using advanced evasion techniques cannot avoid their architecture[9]. Besides that, the attempt to detect the modern, complex malware with numerous exploits was based on an out dated hierarchy of malware domains that describe the malware by a single prevalent behaviour. Considering this gap, the study recommended a core model for the new malware ontology that focused on the malware behaviours the difference between their suggested ontology and current ones is that is not connected to classical malware lessons, but to potentially risky behaviors. Hence, the researchers are capable to find unknown applications as malware[10].

Principally, the run-time behaviour of running processes on an end-host dynamically identifies malware. Most of these identification schemes construct a model of run-time behaviour of a specific process based on its corresponding data flow and/or operating system calls. However, The genetic footstep contain elected parameters – preserved inside the process control block of a kernel for every running operation- that describes its corresponding behaviour and semantics. Addressing that, the study identified the discriminatory elements of a PCB through the execution traces of benign and malware processes, which shortlisted 16 out of 118 task construction elements using the time series analysis. The attributes of the generic footprint were also statistically validated. The appropriate machine learning classifiers for malware detection were also statistically selected. Based on a dataset of 105 benign processes and 114 recent malware processes for Linux, the model revealed a detection accuracy of 96% with false alarm rate of 0% within 100 ms of the initiation of the malicious activity. Furthermore, it only acquired partial information at a given time during the execution of the process, which allowed the kernel of operating system to devise mitigation solutions; thus, reaffirming the robustness of the model to evasion[11].

Overall, a substantial number of studies on malware detection and the proposed malware removal models or scanners were reviewed. Most of these studies employed static and dynamic methods. As comparison between these methods we found some studies used static method that depend on signature names of many viruses that injected in database of their algorithms. Static method has one weak in detection accuracy (only known viruses signatures names detected). While some studies used dynamic method that utilized many techniques such as opcodes, machine learning, sub graph mining and *Deterministic* Finite Automaton (DFA), all these techniques depends on malwares behaviours factors. Most of these techniques works better than signatures detection techniques in detection time and removal but also has only one weak in detection accuracy (not all malwares detected). Hence, our study is intending to design behavior-based detection and removal methods can detect and remove all malwares in low time running.

3. Methodology

In general, there are various types of malware with approximately 100,000 recognised viruses. This study combined static and dynamic detection methods to effectively scan, detect, and remove malware of various types. The static detection method or also known as signature-based detection method scans e-mails, files, messages, programmes, and other data to detect malicious files. Typically, a malware signature depends on a unique piece of code from the malware. Besides that, the dynamic detection method or also known as behaviour-based detection method, specifically the N/A (Not Available in registry) detection method for this study, detects and removes known and unknown types of malware according to their behaviour.

The method utilizes Microsoft windows operating system registry (system database). Therefore, when a malware value is injected in a system database, it will be detected by the proposed method.

```
Data: Containing All Running Processes information
Result: System Status Boolean Value
1  Begin
2    For  Each Running  Process  Info do
3      IF  Running Process File Description    Equal Null
4        Then
5      IF Running Process Main Module Name Equal      "N/A"
6        Then
7          Set System   Status  False;
8        Else
9          Set System Status True
10     end
11   end
12 end
13 end
```

Algorithm 1: N/A detection technique.

The detection Algorithm 1 reflects malware detection based on two factors, namely "File Description" and "Module Name". When the "File Description" equals to null and the "Module Name" equals to N/A, a false system is indicated. In other words, a malware is detected. On the other hand, when the "File Description" and "Module Name" do not equal to null, a true system is indicated. In other words, it indicates no malware. The quality attributes of this particular algorithm were deemed reliable, straightforward (one nested loop), reusable (can be used more than once), and high-performance.

Signature based detection is a method used to detect malicious code; the signature is commonly depends on part of the code that is taken away from the malware itself. Using this technique the scanner will scan messages, programs, files, emails, and other data. Using procedures and compare these files to the signatures which saved in its data.

```
Data: Contenting Directory      Files, Virus List Data Base
Result:  Detected   Files
1  Begin
2    For each Directory Files  do
3      If Directory Files Name not equal Null  then
4        For each Virus List Name do
5          If Directory Files Name equal Virus List Name then
6                ++ Detected Files [Name];
7          end
8        end
9      end
10   end
11 end
```

Algorithm 2: Signature detection technique.

Meanwhile, the detection Algorithm 2 involves two factors that indicate the case of malware: 1. when the "Directory File Name" does not equal to null, and 2. when the "Directory File Name" equals to "Virus List Name". Similarly, the quality attributes of this algorithm were deemed straightforward (one nested loop) and reusable.

```
Data: Control process Name, Registry Name
Result: System Status Boolean Value
1  Begin
2    While System Status equal False
3      Kill Process by Name;
4      Remove Registry by Name;
5      System Status equal True;
6    end
7  end
```

Algorithm 3: N/A removal technique.

The removal Algorithm 3 is related to only one factor, which is the "System Status". When the "System Status"

reveals false sign, it indicates that the removal algorithm detects and removes the malware in less than two seconds. The quality attributes of this algorithm were also deemed reliable, straightforward (one nested loop), reusable, and high-performance.

```
Data: contenting Directory Files, Virus List Data Base
Result: Detected-Files
1  Begin
2    For each  Directory Files do
3      If Directory Files Name not equal Null  then
4        For each Virus List Name do
5          If Directory Files Name equal Virus List Name then
6            ++ Detected Files [Name];
7            Remove File by Name;
8          end
9        end
10     end
11   end
12 end
```

Algorithm 4: Signature removal technique.

As for the removal Algorithm 4, it has similar factors with the factors of the detection Algorithm 2. When the "Directory File Name" does not equal to null and the "Directory File Name" equals to "Virus List Name", the removal algorithm detects and removes the malware in less than four seconds. Likewise, the quality attributes of this algorithm were deemed straightforward (one nested loop) and reusable.

4. Experiment Setting and Results

This study first installed scanner software with two different interfaces (each interface operated under a single algorithm) in HP desktop (Intel® Core™ i5 processor with 64-bit operating system and 4.0 GB of RAM). Both algorithms operated individually in the scanner software. The SBD algorithm testing compared the viruses that were injected in partitions to the signatures or virus names in the algorithm database. Basically, a virus is detected when there is a match. Besides that, the behaviours of malware were also analysed for the N/A algorithm testing. For this, a Trojan horse was used because it poses the highest security threat (hackers are able to steal all files from infected computer) among the various types of malware. Furthermore, a Trojan horse is associated with running time given its behaviour to enter the Windows Registry after the users double-click on the malware-infected file (sent by hackers). Moreover, malware detection algorithm in this study was designed based on the behaviors of a Trojan horse.

Table 1 shows the results of malware (virus) detection and removal time using the SBD method. This study prepared between 200 and 700 lines in a file as a database for the SBD algorithm then proceeded to inject signatures or virus names into the SBD algorithm database. Besides that, 25 viruses were injected in different partitions. Overall, it was revealed that most of the viruses (1–23) were detected within different timeframes (1.5 – 5 seconds) and removed between 0.5 seconds and 4.0 seconds:

1. One virus was injected—SBD method detected one virus within 1.5 seconds and removed the virus within 0.5 seconds.
2. Five viruses were injected—SBD method detected five viruses within 3.5 seconds and removed the viruses within one second.
3. 10 viruses were injected—SBD method detected 10 viruses within four seconds and removed the viruses within 1.5 seconds.

Table 1. SBD results

No of Files as database	No of malware in partitions	No of Detectedmalwares	Detection Time/ second	Removal Time/ second
200	1	1	1.5	0.5
300	5	5	3.5	1
400	10	10	4	1.5
500	15	14	4.5	2
600	20	18	5	3
700	25	23	5	4

4. 15 viruses were injected—SBD method detected 14 viruses within 4.5 seconds and removed the viruses within two seconds.
5. 20 viruses were injected—SBD method detected 18 viruses within five seconds and removed the viruses within three seconds.
6. 25 viruses were injected—SBD method detected 23 viruses within five seconds and removed the viruses within four seconds.

Table 2. N/A results

No of processes	No of injected malwares	No of detected malwares	Detection time (Second)	Removal time (Second)
38	2	2	1	1
45	3	3	1.5	1
58	4	4	1.5	1
68	5	5	2	2
72	6	6	2	2
88	7	7	2	2

Table 2 shows the results of malware (Trojan horse) detection and removal time using the N/A detection method. The processes in Windows Registry (system database) for the Windows operating system were operated within different timeframes, which involved between 38 and 88 processes. The malwares (Trojan horse) were also injected in six different partitions. Overall, it was revealed that the malware were detected within different timeframes (between one second and four seconds) and removed between one second and two seconds:

1. Two malwares were injected—N/A detection method detected two malware within one second and removed the malware within one second.
2. Three malwares were injected—N/A detection method detected three malware within 1.5 seconds and removed the malware within one second.
3. Four malwares were injected— N/A detection method detected four malware within 1.5 seconds and removed the malware within one second.
4. Five malwares were injected—N/A detection method detected five malware within 2 seconds and removed the malware within two seconds.
5. Six malwares were injected—N/A detection method detected six malware within 2 seconds and removed the malware within two seconds.
6. Seven malware were injected—N/A detection method detected seven malware within 3 seconds and removed the malware within two seconds.

Based on Table 1 and 2, the result of N/A detection algorithm and SBD algorithm depends on three factors. First is detected malwares, second detection time and third removals time. However, N/A detection algorithm (dynamic method) show better results than SBD algorithm (static method) in quality attributes, detection accuracy (unknown and known malwares were detected), detection time and removal time.

5. Conclusion

This study employed two types of malware detection methods, specifically the static or signature-based detection method and the N/A detection method (dynamic or behaviour-based detection method). The signature-based detection method revealed a drawback, which is the need to regularly update the signatures or malware names in its algorithm database for effective malware detection. Otherwise, the algorithm is unable to detect most of the malware. Meanwhile, the N/A detection method in this study exhibited better quality attributes, detection time, and removal time compared to the signature-based detection method.

6. Reference

1. Hellal A, Ben Romdhane L. Maximal Frequent Sub-Graph Mining for Malware Detection. 2015 15th International Conference on Intelligent Systems Design and Applications (ISDA), Marrakech; 2015. p. 31–39. https://doi.org/10.1109/ISDA.2015.7489265.
2. Kawakoya Y, Iwamura M, Shioji E, Hariu T. API Chaser: Anti-Analysis Resistant Malware Analyzer. In: Stolfo S.J., Stavrou A., Wright C.V. (eds) Research in Attacks, Intrusions, and Defenses. RAID 2013. Lecture Notes in Computer Science, vol 8145. Springer, Berlin, Heidelberg; 2013. p. 123–43. https://doi.org/10.1007/978-3-642-41284-4_7.
3. Han L, Liu S, Han S, Jia W, Lei J. Owner based malware discrimination, Future Generation Computer Systems. 2018; 80:496–504. https://doi.org/10.1016/j.future.2016.05.020.
4. Divandari H, Pechaz B, Jahan MV. Malware detection using Markov Blanket based on opcode sequences. 2015

International Congress on Technology, Communication and Knowledge (ICTCK), Mashhad; 2015. p. 564–69. https://doi.org/10.1109/ICTCK.2015.7582730.

5. Hellal A, Ben Romdhane L. Maximal Frequent Sub-Graph Mining for Malware Detection. 2015 15th International Conference on Intelligent Systems Design and Applications (ISDA), Marrakech; 2015. p. 31–39. https://doi.org/10.1109/ISDA.2015.7489265.
6. Preda MD, Maggi F. Testing android malware detectors against code obfuscation: A systematization of knowledge and unified methodology, Journal of Computer Virology and Hacking Techniques. 2017; 13(3):209–32. https://doi.org/10.1007/s11416-016-0282-2.
7. Ding Y. Dai W, Yan S, Zhang Y. Control flow-based opcode behavior analysis for Malware detection, Computers and Security. 2014; 44:65–74. https://doi.org/10.1016/j.cose.2014.04.003.
8. Shijo PV, Salim A. Integrated static and dynamic analysis for malware detection, Procedia Computer Science. 2015; 46: 804–11. https://doi.org/10.1016/j.procs.2015.02.149.
9. Das S, Xiao H, Liu Y, Zhang W. Online Malware Defense Using Attack Behavior Model. In: Int'l Symposium on Circuits and Systems (ISCAS 2016), Montreal, Canada; May 22-26, 2016. p. 1322–25.
10. Paulista CL. Ontology for Malware Behavior : A Core Model Proposal. In: IEEE 23rd International WETICE Conference; 2014. https://www.lasca.ic.unicamp.br/paulo/papers/2014-WETICE.Web2Touch-gregio-afonso-ontology.malware.pdf.
11. Shahzad F. Shahzad M, Farooq M. In-execution dynamic malware analysis and detection by mining information in process control blocks of Linux OS, Information Sciences. 2013; 231:45–63. https://doi.org/10.1016/j.ins.2011.09.016.