Scientific Journal of Informatics
Vol. 8, No. 2, November 2021
p-ISSN 2407-7658 http://journal.unnes.ac.id/nju/index.php/sji e-ISSN 2460-0040

Utilizing Reverse Engineering Technique for A Malware Analysis Model

Amiruddin Amiruddin[1*], Putri Nur Halimah Suryani[2], Shandika Dianaji Santoso[3], Muhammad Yusuf Bambang Setiadji[4]
[1,2,3,4]Cyber-Security Engineering, Politeknik Siber dan Sandi Negara, Indonesia

Abstract.
Purpose: Malicious software or malware is a real threat to the security of computer systems or networks. Researchers made various attempts to find information and knowledge about malware, including preventing or even eliminating it. One effort to detect it is using a malware dynamic analysis model based on reverse engineering techniques. However, there are many reverse engineering techniques proposed with various stages and requirements in the literature.
Methods: This research uses an experimental method. The object of research is a malware analysis model using reverse engineering techniques. The experimental method used is qualitative, collecting data related to the advantages and disadvantages of the reverse engineering-based malware analysis models used as a reference in this study. The data is used as consideration to propose a new model of malware analysis utilizing reverse engineering techniques.
Result: In this study an analysis model of malware was proposed by synthesizing several reverse engineering-based malware analysis models.
Novelty: The proposed model was then tested in a virtual environment where it is proven to be more effective than previous models for analyzing malware.

Keywords: Analysis, Malware, Model, Reverse Engineering
Received June 2020 / **Revised** September 2021 / **Accepted** November 2021

INTRODUCTION

Malicious software (malware), such as viruses and worms, spreads far more quickly in a world where millions of users connect to the Internet and use email every day [1], including through mobile gadgets [2] on Internet of Things networks [3][4]. To combat this widespread, researchers around the world conduct malware analysis. Researchers or analysts use various models of malware analysis according to their needs and competencies. Some malware analysis models are surface analysis and runtime analysis, and the most commonly used is static analysis [5]. In addition, there is a dynamic analysis model using machine learning [6], [3], using network traffic analysis [7], using deep learning [8], isolated malware [9], malware detection on IoT [10], as well as the analysis of the malware community [11].

The surface analysis and runtime analysis model only observe malware that is thought to have infected a system and study the behavior and characteristics of the malware without executing it. The static analysis model does what the two previous models do but runs the malware to analyze the source code and the algorithm used. In such an environment, the malware can infect a system, or it can be said that the scope of the static analysis is broader and more complicated. But in practice, this static analysis model cannot be used to analyze unknown malware. Researchers who use static analysis must have a pretty good knowledge of machine language (assembly) and have a deep enough understanding of the operating system. In addition, the results obtained are also not optimal, have not yet reached the design or infrastructure of the malware [12].

The dynamic analysis model uses reverse engineering techniques to overcome weaknesses in surface analysis, runtime analysis, and static analysis models. Reverse engineering allows researchers to know the design or structure, process, and various information obtained using the previous three models. Implementing reverse engineering techniques in malware analysis becomes a problem because only a few

* Corresponding author
Email: amir@poltekssn.ac.id (Amiruddin), putri.nur@student.poltekssn.ac.id (Suryani), shandika.dianaji@student.poltekssn.ac.id (Santoso), yusuf.setiadji@poltekssn.ac.id (Setiadji)
DOI: 10.15294/sji.v8i2.24755

methods discuss this technique, and there are limitations in human resources. Dynamic analysis can observe malware behavior and its impact on the system environment but cannot get detailed information. Conversely, static analysis can provide detailed information on malware, but it is challenging to observe the behavior of malware and its effects.

Therefore, we need an analysis model based on existing models that can reduce these problems. The model is expected to help users choose which model is the most appropriate way to analyze malware using the reverse engineering technique. This study proposes a new model for malware analysis using reverse engineering techniques based on previous models.

BASIC CONCEPT AND RELATED WORKS

Reverse Engineering

In simple terms, reverse engineering malware can be defined as engineering malware to obtain information about malware behavior. Several tools such as a disassembler, debugger, PE Viewer, and network analyzer are required to reverse engineer malware. A disassembler is a tool used to unload an application to generate assembly code. It is also possible to convert binary code to native code even though it is not always available for all architectures. An example of disassembler is IDAPro. Debuggers manipulate the implementation of programs to get information about what is happening and how it impacts when malware is running. Examples of debuggers are X64dbg, Wind River, Windbg, The GNU Project Debugger (GDB), PEiD, PEStudio, and PE32. PE Viewer is needed to extract important information from executable files to display the dependencies. Examples of PE viewer are CFF Explorer and PE Explorer. A network analyzer is used to get information on how a program interacts with other programs, including what connections the program makes and what data it wants to send. An example of a well-known network analyzer is Wireshark.

Several analytical techniques in reverse engineering malware that can be used are static, dynamic, and automatic analysis. Static analysis is the process of analyzing malware or binaries without actually running them. It can be as simple as seeing the metadata from a file. Dynamic analysis is the process of analyzing malware by running it in an environment and then observing the behavior of malware and the impacts that might occur. Automatic analysis is used to automate activities to speed up the process and save time.

Related Works

There are several reverse engineering-based malware analysis models that were used as a reference in this study. Dynamic analysis model of ransomware by Zimba et al. [13] was conducted on the client-side with virtualization using multiple virtual machines (VMs), to avoid damage due to the effects of running malware and virtual networks to connect with servers. The servers run Cuckoo Sandbox and Volatility on Linux to collect the behavior of the analyzed malware. Reverse engineering was done to strip the ransomware code. Setia et al. [14] performed the process of identifying Flawed Ammyy Remote Access Trojan (RAT) malware using a descriptive methodology and malware dynamic analysis method. They proceed with the reverse engineering method in the form of disassembly using IDAPro tool to translate from machine language into an assembly language so that the malware code commands can be known.

Basic analysis techniques are challenging to find new-technique malware. Setia et al. [15] used reverse engineering to conduct malware analysis because the code on malware can be known. They performed a RAT malware analysis using several stages of reverse engineering that consist of identifying malware samples to find out the type of malware file; calculating the hash value; creating a system environment to be used; dynamic analysis; string analysis; and disassembly or debugging. Megira et al. [16] analyzed malware by using malware samples to better understand how they can infect computers and devices, the level of threat they pose, and how to protect devices against them. The reverse engineering malware process was implemented with the following steps: assembly, disassembly, debugging, X86 architecture, instruction, hashing, and string analysis.

Assembly language was used for microprocessors and other devices that can be programmed with any low-level programming language. Assembly languages cannot recognize high-level languages such as Java and Pascal. Disassembly was used to change assembly language into machine code. Disassembly is the reverse assembly process. Debugging is a method that developers can apply to find bugs, reduce bugs, and isolate the source of the problem. Debugging was used to run tests of each core process in malware. The X86 architecture is a complex computer instruction set design with varied instruction lengths. Internally, most modern computer architectures, including x86 follow the Von Neumann architecture. In a reconfigurable

system design, interoperability can also be a problem in architecture. Instruction is the construction of an assembly program. The x86 instruction assembly consists of mnemonic and zero or operands. The hashing process was run for verification before and after the malware analysis process. Verification was carried out to determine the absence of hash changes in the malware sample after the analysis process. String analysis is one step in static analysis. This analysis can find information in string values stored in a variable at specific points in a program. This information is beneficial for understanding how the program works, detecting programming errors and correcting them, detecting security vulnerabilities, and solving specific program verification problems [17].

A model by Uppal et al. [12] removes obfuscation code from programs to improve malware detection. During the process, the malware passes through the normalizer and is then matched with an existing legacy database. If it proves to be a match, it will be a new signature stored in the database. PE code was passed through the decompression application, and then the decompressed code would pass through the disassembler and then pass through the normalizer. The code to be obtained was a normalized code, which was then sent to the malware detector. This detector then extracted the malware and compared it to the signature on the repository. If the signature did not match, it would be categorized as a new signature. The analysis model's stages are as follows: PE code, decompression, disassembly, normalizing, malware detection, and signature comparison.

Nugroho et al. [18] proposed a model of Malware Analysis Environment and Requirements (MAER). The earliest step in reverse engineering malware was to determine the standard operational procedure (SOP) that would be performed. Defining malware was the next step that could be done using an automated scan. However, this service has the disadvantage that if the scanned malware is a new type of malware while the scanner has not updated, the malware file cannot be detected. After that, researchers determined the purpose of malware analysis to choose the tool or method to be used. The next step was establishing a malware analysis environment and requirements. MAER is a unique laboratory used to conduct research related to malware analysis. MAER was needed in malware analysis to maintain the security of the system used by researchers. When executing malware, it was feared that the malware would spread or infect the system. Some examples of MAER commonly used are Malware Repository using virus share, Virtual Machine Environment using virtualbox, and using "host only adapter" network mode.

Hashing malware was used to ensure that the sample of malware analyzed is the same malware as the original. The results of hashing malware samples can be compared with the original hashing of malware by using hashing tools such as CFFExplore. Searching strings in malware files were done in ASCII format, not in hexadecimal format. This process aimed to determine the association of malware with a separate program, the signature of the malware, and some other information. Before packing, the malware will first detect the packer. If malware uses a packer, the next step is to unpack the malware to get the actual malware (unprotected packer) and find out more in-depth information. Monitoring the malware process requires researchers to execute malware and monitor the process carried out by malware. It will get behavior of the malware.

METHODS

Research Method

This research uses an experimental method. The object of research is a malware analysis model using reverse engineering techniques. The experimental method used is qualitative, collecting data related to the advantages and disadvantages of the reverse engineering-based malware analysis models used as a reference in this study. The data is used as consideration to propose a new model of malware analysis utilizing reverse engineering techniques. The proposed model is then simulated in a virtual environment.

Proposed Model

The proposed malware analysis model is synthesized from data extracted from several research references discussed in the previous section. To make it easier to present data, the stages of the proposed malware analysis model and its comparison with the reference model are given in Table 1.

Table 1. Stages of reverse engineering-based malware analysis models

Uppal et al. (2014)	Nugroho et al. (2015)	Zimba et al. (2017)	Setia et al. (2018)	Setia et al. (2019)	Megira et al. (2018)	Proposed
Establish secure environment	Establish secure environment	Establish secure environment	Establish secure environment	Define malware	Assembly	Establish secure environment
PE code	Hash	Analyze malware (client)	Set process explorer	Establish secure environment	Dis-assembly	Define malware
Decompress	Analyze string		Snapshot registry	Dynamic analysis	Debug	Detect packer
Disassembly	Unpack		Run malware	String analysis	Set archi-tecture x86	Unpack
Normalize	Debug		Set network traffic	Disassembly or debug	Instruction	Debug
Detect malware	Monitor	Gather malware behaviors (server)	Disassembly	-	Hashing	Monitor
Compare signature	-		-	-	String analysis	Compare signature

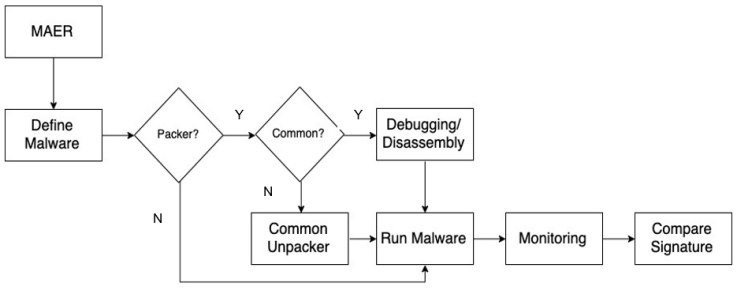

Figure 1. The flowchart of the proposed model

Based on Table 1, a flowchart of the proposed model was created and is given in Figure 1. Explanation of the stages of the proposed model is as follows:

a. Establish a secure system environment (MAER)
   The first step that researchers must do is to build a secure computer network as a research environment to prevent damage from malware activity being analyzed. In building this research environment, researchers can utilize virtual machines such as Virtualbox or VMware. In this paper, the Ubuntu 16.04 and Windows XP operating systems are used.
b. Defining malware or identifying malware
   The process of identifying malware is a process to find out the essential characteristics of malware. In general, check the characteristics of malware is done by searching for file types, creating a fingerprint malware by performing a hash process, and scanning multi-antivirus. The data which is the result of checking these characteristics will then be called a malware signature.
c. Packer detection
   Malware that uses a packer can be detected using the ExeInfo tools. The use of packers in malware aims to disguise the malware signature to avoid the scanning process carried out by signature-based antivirus.
d. Unpacking
   Unpacking is done if the malware is indicated using a packer. This process could use UPX tools if the malware uses a standard packer. However, if malware uses a non-standard packer, marked by the word "Unknown Packer" in the ExeInfo tool, debugging is needed to get the malware's core.
e. Debugging / Disassembler
   Debugging or disassembler is the process of getting the structure of the malware source code. By knowing the source code that forms malware, researchers can find out the process and activity of malware on a computer or network. IDAPro can be used for debugging.
f. Monitoring
   Monitoring is an activity to observe malware activities in a computer or a network. Tools used to monitor malware activities, inter alia, are Process Explorer, Process Monitoring, and Process Hacker whereas for monitoring them on network, we can use Wireshark or Inetsim used DNS servers.
g. Comparing database

After getting information related to the structure and the activities of the malware, we can compare the malware signature with the one saved in the database. If it matches, it will be classified as old malware, whereas if it does not match, it will be classified as new malware.

RESULT AND DISCUSSION

Proposed Model Simulation

Proposed models formed based on the synthesis of existing models need to be tested to prove whether the model is more effective than previous models for analyzing malware based on reverse engineering techniques and is suitable for further research.

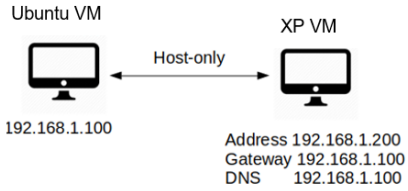

Figure 2 Experiment configuration

The experiment follows the stages of the proposed model given in Figure 1.

a. Establishing MAER
   The MAER configuration built in our experiment is given in Figure 2.

b. Identifying malware
   The malware identification process uses several tools such as Hash My File to provide fingerprints in the form of hash values for malware and PEStudio to find strings in malware. If the strings are low, it can be assumed that malware uses packers to avoid antivirus detection. The process of identifying malware using Hash My File can be seen in Figure 3. Visible information such as file names, MD5 values, SHA1, CRC32, SHA-256, SHA-512, SHA-384, Full Path, until the date the malware was created and modified and the size the file.

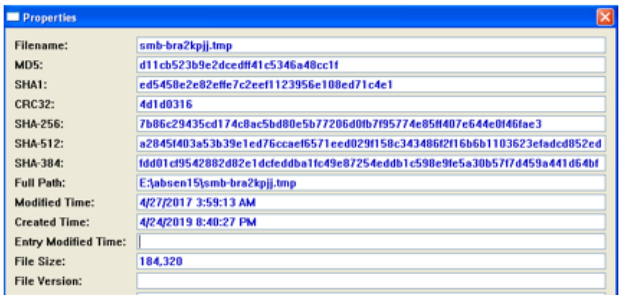

Figure 3 Malware identification

c. Detecting packer
   This aims to find out whether malware uses a packer to avoid anti-virus detection. This can be done using tools such as EXEinfo and PEStudio.

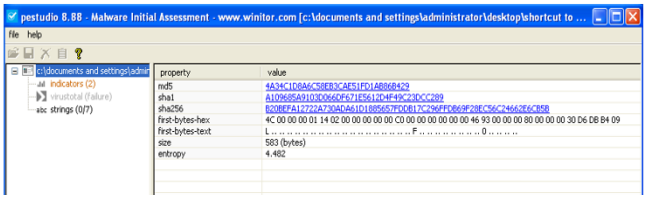

Figure 4 Packer detection using tool of PEStudio

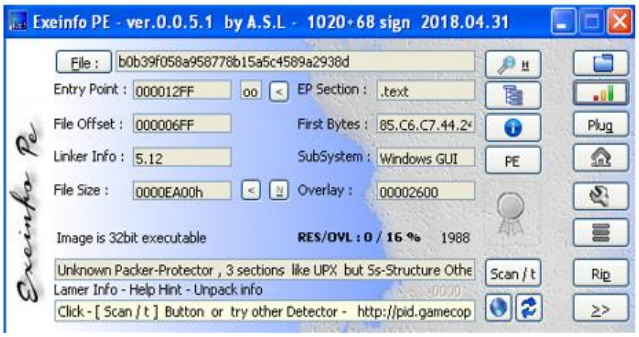

Figure 5 Packer detection using tool of "Exeinfo PE"

If the strings displayed are small, it can be assumed that the malware uses a packer. The malware information found using PEStudio (given in Figure 4) and ExeInfo PE (given in Figure 5) shows that the malware is detected using an unrecognized packer, so we need to conduct a debugging process.

d. Debugging

The debugging process aims to determine the structure and workings of malware in the system when the malware has been successfully executed. The debugging process can be done using Ollydbg or IDA Pro tools. Figure 6 is the result of debugging using IDAPro.

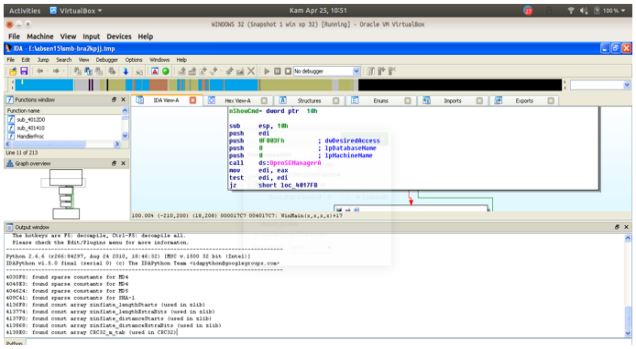

Figure 6 Result of debugging

e. Running Malware

At this stage, the malware samples are run in a secure environment that has been created by making sure that important data has been backed up.

f. Monitoring

The malware monitoring process can be done using logs on the DNS server (Figure 7) that have been created or using Wireshark to monitor data traffic on the network. Thus, researchers can find out whether malware is in contact with outside or not. System-level engineering works by running various tools and utilizing existing services in the operating system to get information, check the type of executable file, find out the input and output of programs, and others.

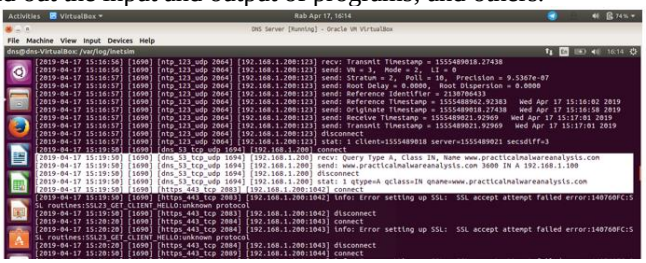

Figure 7 Monitoring using DNS log server

In the proposed method, the system-level reversing session starts in the Malware Analysis Environment and Requirements (MAER) section to identify malware and detect packers. This stage generates information in the outline of the malware object under study. Code-level reversing is the process of extracting programs to the binary level to get details from objects. In this research, the code-level reversing session is carried out at the debugging / disassembler stage of malware. This stage produces detailed information related to malware based on the binary code obtained.

Simulation Result

The results achieved in this study are summarized in Table 2. Making a secure system successfully created using virtualization with Ubuntu and Windows XP systems. The environment can protect the device from damage that malware can cause during the malware analysis process. Packer detection manages to find out the type of packer used by malware. The unpacking process successfully unlocks the packer used by malware.

Furthermore, the debugging process can show the structure of the file and the path of the malware program being analyzed. Malware execution was carried out successfully in a secure environment, as mentioned earlier. The process of monitoring malware behavior was carried out on a computer system in a virtual environment at the time of malware analysis.

Table 2. Summary of simulation results

No	Proposed stages	Achieved results
1	Establish MAER	Protect your device from damage that malware can cause
2	Detecting packer	Know the type of packer used by malware.
3	Unpacking	Open packer which is used to get more information on malware
4	Debugging	Look at the file structure and program flow carried out by malware
5	Running malware	Running malware on the system
6	Monitoring	Monitor malware behavior on computer systems

CONCLUSION

The proposed model created by synthesizing previous reverse engineering-based models is proved to be more effective for analyzing malware than the previous models. It can be reached because all stages can be controlled and observed not to cause damage to the system, and malware behavior can be observed during malware analysis. The proposed model can implement static stages to detect malware without running it. However, when it is known that the malware is a new type, dynamic steps is carried out instead of static step to study the behavior of the malware, but this analysis is carried out under controlled conditions on the virtual machine so that the adverse effects of running the malware can be avoided. This model is more effective because not all steps must be carried out depending on the detection stage of the packer used.

REFERENCES

[1] Eilam, E. (2005). "Reversing: secrets of reverse engineering," Canada, Wiley Publishing, Inc.

[2] Sen, S; Aydogan, E. ; Aysan. A.I. (2018). "Coevolution of mobile malware and anti-malware," IEEE Trans. Inf. Forensics Secur. Vol. 13 Issue 10 pp. 2563 - 2574

[3] Jeon, J ; Park, J.H.; Jeong, Y-S. (2020). "Dynamic analysis for IoT malware detection with convolution neural network model," IEEE Access, Vol. 8 pp. 96899 - 96911

[4] Wazid, M. ; Das, A.K.; Rodrigues, J.P.C.; Shetty. S; Park, Y. (2019). "IoMT malware detection approaches: analysis and research challenges," IEEE Access, Vol. 7, pp. 182459 - 182476

[5] Indrajit, R. E. (unknown). Analisa Malware, ID-SIRTI. Available on http://www.idsirtii.or.id.

[6] Mangialardo, R. J. and Duarte, J. C. (2015). "Integrating static and dynamic malware analysis using machine learning," in IEEE Lat. Am. Trans, vol. 13, no. 9, pp. 3080-3087.

[7] Zhao, G., Xu, K., Xu, L., and Wu, B. (2015). "Detecting APT malware infections based on malicious DNS and traffic analysis," in IEEE Access, vol. 3, pp. 1132-1142.

[8] Vinayakumar, R., Alazab, M. , Soman, K. P., Poornachandran, P., and Venkatraman, S. (2019). "Robust intelligent malware detection using deep learning," in IEEE Access, vol. 7, pp. 46717-46738.

[9] Rodriguez, R. J., Gaston, I. R., and Alonso, J. (2016). "Towards the detection of isolation-aware malware," in IEEE Lat. Am. Trans, vol. 14, no. 2, pp. 1024-1036.

[10] Wazid, M., Das, A. K., Rodrigues, J. J. P. C., Shetty, S., and Park, Y. (2019). "IoMT malware detection approaches: analysis and research challenges," in IEEE Access, vol. 7, pp. 182459-182476.

[11] Cruickshank, I.J. and Carley, K. M. (2020). "Analysis of malware communities using multi-modal features," in IEEE Access, vol. 8, pp. 77435-77448, 2020.

[12] Uppal, D., Mehra, V., and Verma, V. (2014). "Basic survey on malware analysis, tools and techniques," JJCSA, vol. 4.

[13] Zimba, A., Simukonda, L., dan Chishimba, M. (2017). "Demystifying ransomware attacks: reverse engineering and dynamic malware analysis of wannacry for network and information security," Zambia Information Communication Technology (ICT) Journal, vol. 1, no. 1, pp. 35-40.

[14] Setia, T.P., Widiyasono, N., dan Aldya, A. P. (2018). "Analisis malware flawed ammyy rat dengan metode reverse engineering," Jurnal Pengembangan IT (JPIT), pp. 371-380.

[15] Setia, T. P., Aldya, A. P. , dan Widiyasono, N. (2019). “Reverse engineering untuk analisis malware remote access trojan,” Jurnal Edukasi & Penelitian Informatika, vol.5 No.1.
[16] Megira, S., Pangesti, A. R., dan Wibowo, F. W. (2018). “Malware analysis and detection using reverse engineering technique,” *IOP Conf. Series: Journal of Physics: Conf. Series* 1140 (2018) 012042, pp 12.
[17] Bultan, T., Yu, F., Alkhalaf, M., and Aydin, A. (2017). String analysis for software verification and security, Springer International Publishing.
[18] Nugroho, H.A., Prayudi, Y. (2015). "Penggunaan teknik reverse engineering pada malware analysis untuk identifikasi serangan malware," KNSI.