Adaptive Threat Detection Method Based on the Operating System Audit Subsystem

Valerii Simonenko
Department of Computer Engineering
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"
Kyiv, Ukraine
0000-0002-6341-6041

Anna Verner
Department of Computer Engineering
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"
Kyiv, Ukraine
0000-0001-8598-363X

***Abstract* — There is a continuous exponential increase in the total number of malicious software for the last decade leading to the computing nodes integrity breach. Existing means are incapable of effective threats countering. Therefore, the problem is the invention of the adaptive method for threat detection. The article considers an adaptive method of security threat detection based on the operating system audit subsystem. The method uses audit subsystems as a means of monitoring the processes and system calls, they make in the system. The classifier is implemented using a neural network represented as a multilayer Rosenblatt perceptron. Training of the model was performed by the usage of a dataset consisting of system calls sequences received as a result of malicious software samples execution. The suggested method effectiveness in the detection of harmful impact is proven on the results of model testing on independent samples of malware and benign software.**

Keywords — operating system audit subsystem, neural network, malware detection, information security, data integrity

I. INTRODUCTION

The rapid development of computing technologies has almost changed all areas of the present. The integration of information technology led to a significant increase in the productivity and improvement of the whole human activities. However, as a result of global digitalization, the variety and number of dangers that affect the integrity of information systems are significantly increasing [1]. Material damages caused by malicious software are estimated to be billions of dollars [2]. That leads to the need for end nodes protection strengthening and the invention of new adaptive security means.

Among the means of information protection, the most widespread are hardware, software, and hardware and software. The last is the most versatile and flexible since they are reconfigurable for any system and architectures types, which is relevant due to the active spread of the Internet of Things (IoT). It is this sector of devices that is one of the most vulnerable and is increasingly used to carry out targeted attacks [3]. However, desktop personal computers are still the leaders in the number of existing malicious software.

The paper suggests an adaptive threat detection method to improve the security of a computing node, based on the use of the operating system audit subsystem.

The intended usage of the audit framework is to monitor events in real-time (while the system is running) or periodically [4]. This component is built-in in most modern operating systems (OS). Recent scientific works [5], [6] demonstrate the effectiveness of these means used to detect the presence of external influence on the information system. However, such analysis, in general, is purely diagnostic in nature and performed only when malicious actions have already been carried out against the computing node.

The offered method for identifying the security threats of a node uses a wide range of capabilities provided by the audit system by using an integrated kernel component to control the processes occurring in the system. That allows preventing malicious software from the execution in the early steps providing the ability to keep data integrity intact.

II. THREATS ANALYSIS METHODS

Various code analysis methods are used to detect malware. By the execution, methods of threat detection are separated into two parts: dynamic and static analysis.

A. Static Analysis

Static analysis stands for the analysis of malicious software without its execution. Detection patterns used in the static analysis include string signatures, byte sequences, n-grams, library syntax calls, control flow graphs, frequency code analysis of operational codes, and more. The analysis is performed due to the pre-unpacking and decoding of the executable file to represent the malware in a different format by using reverse engineering and other means.

Disassemblers and debuggers commonly used in reverse engineering allow displaying the malicious software code in the form of assembly instructions, which provides a lot of information about what exactly malware does, and also helps to identify patterns to identify attackers. Memory dumping tools are used to retrieve protected code stored in system memory and dump it to a file. This technique is useful for analyzing packaged executables that are difficult to disassemble. Binary obfuscation methods convert malicious binary files into self-compressed and uniformly structured files. They are designed to withstand change and learning and don't allow for qualitative analysis. Besides, in the usage of binary executable files for static analysis, information such as the size of data structures or variables is lost, further complicating the analysis of software code, accordingly to [7].

Sophisticated methods of evasion of static analysis by attackers have led to the need for dynamic analysis developing. In [8] highlighted the shortcomings of the static

analysis methodology and introduced a scheme based on code obfuscation, which proves the inadequacy of static analysis for malware detection or classification. According to that research, dynamic analysis is a necessary complement to static analysis because it is less vulnerable to obfuscation.

B. Dynamic Analysis

Dynamic malware analysis involves the analysis of the program during its execution [9] by running malware in a secure and controlled environment to avoid transferring the malware under investigation to other systems or networks. The main dynamic analysis includes observation of the collected sample and its interaction with the system. Snapshots of the virtual machine's initial state are taken, after which the malware runs for execution in the test system. Output and input states are compared for change detection. The changes obtained from the observations are then used as key features to remove and detect threats and its software from infected nodes.

Dynamic analysis is an important step in threats analysis, although it does not provide comprehensive information on malware [10]. Advanced dynamic analysis involves the usage of tools for studying the state of a malicious program during its launch. The usage of advanced analysis methods provides information that cannot be collected using other methods [11]. Dynamic analysis is always performed in an isolated environment to ensure that all inputs and outputs of the system are known for further consideration.

The usage of additional tools allows tracking of the APIs, system function calls, modified and deleted files, registry changes during the interaction with the system. Analyzing the parameters used during function calls allow to group the functions used semantically while analyzing the data processed and distributed in the system giving an understanding of the files used and produced by the malware [12]. Advanced dynamic malware analysis is very useful for detecting malicious software variants.

The particularities of each analysis method have to be indicated. Dynamic analysis systems execute binaries in a virtualized environment to record the behavior of the sample, looking for indicators of malicious activity. On the other hand, static parsers process executable files without running them, extracting the features used for classification directly from the binaries and their metadata. Although both approaches have positive and negative aspects, many endpoint security solutions are usually solved precisely by static analyzers due to the strict time constraints required to avoid affecting system performance.

C. Behavior-Based Approach

Dynamic analysis includes an approach based on analyzing the behavior of malicious software. Such an approach is used in the suggested paper during the development of an adaptive method of threat detection. The behavioral-based approach allows detecting a large number of existing malwares also providing the ability to detect new types and samples of malware. However, it is not universal. Therefore, there is a need to find a method that effectively detects more complex, unknown programs.

The behavioral approach for detecting malware adheres to the behavior of the program using monitoring tools and determines whether the program is malicious. Despite the change in program code, the behavior remains similar, thus allowing the application of this approach to identify most new malware [13]. However, some malware programs do not work properly in a virtual environment, and therefore, a sample of malware may become false positive. The monitoring of system calls is used as a basis of a behavioral-based approach in the proposed adaptive threads detection method. The behavioral analysis consists of a few steps. Its algorithm is shown in Figure 1.

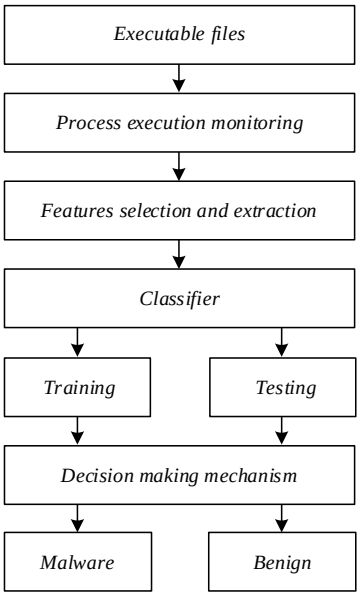

Fig. 1. Behavioral-based approach algorithm [14]

After running the executable file, monitoring of its spawned processes begins. While monitored processes are being executed the service data capturing is performed. These data are being taken to a separate set that is then filtered by extracting only the key features necessary for further classification. On the final stage, the obtained reduced dataset is passed to the inputs of the classifier, which makes the decision on its maliciousness degree according to the input sequence and pre-trained patterns.

III. Adaptive Threat Detection Method Based on the Audit Subsystem

The proposed method is based on the usage of two main components: operating system audit subsystem and deep learning model.

A. Audit Subsystem

The audit subsystem is a built-in component in modern operating systems. Thus, in the most popular desktop operating system Windows, it is implemented as Event Tracing (ETW) - a software interface for tracing at the kernel level, which allows recording not only kernel events but also applications system calls in a specialized log file in real-time and provides the ability to further usage of data for applications configuration and instant problems identification [15].

The structure and principles of the audit subsystem can be explored on the basis of operating systems with the Linux kernel.

The general purpose of the audit subsystem is to obtain detailed information on the status of system processes that

comply with appropriate policies and to log any types of events in real-time mode (monitoring file access, system calls, recording user commands, recording security events, event search, etc.). Data acquisition is carried by passing each process through the filter system of the audit subsystem. Scheme of audit subsystem architecture and system calls from user space implementation is shown in Figure 2.

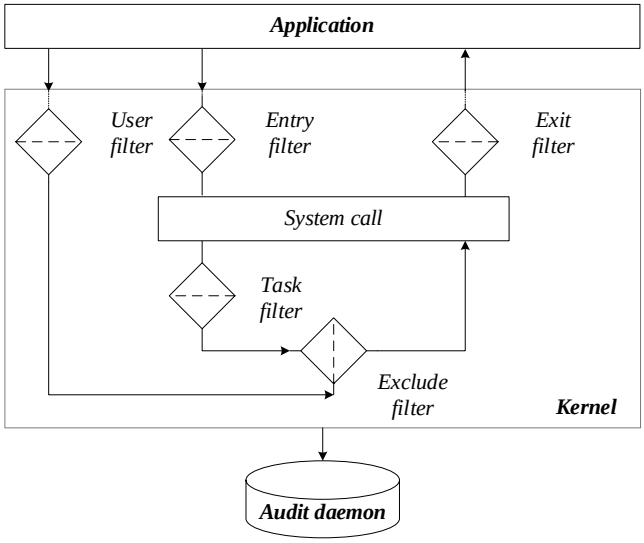

Fig. 2. Audit system architecture [16]

B. Threat Detection Method Structure

The presence of a layer in the form of an audit kernel module allows building a universal tool for intercepting and managing system calls and processes, as well as provides the ability to collect data needed for further use in the analysis of system activity for threats detection. Accordingly, it is mandatory to provide the audit subsystem with the ability to record the following data: date and time of the event, type of event, entity ID, a result of actions; association of the event with the identity of the user who caused it; all attempts to modify the audit subsystem configuration files and access the logs; use of authentication mechanisms; changes in trusted sources and databases; attempts to import and export information. It is also significant to enable or exclude events based on user ID, object labels, and other attributes.

To perform an analysis of the system work based on the captured data the use of an analyzer built with the means of neural networks is proposed. This approach allows the creation of a system capable of self-learning and its further accuracy improvement through the analysis of new threats by using data from knowledge bases and gaining experience. The advantage of this approach is the relative resistance to zero-day attacks.

The interception of all system calls that occur in the system has a significant impact on system performance. Therefore, to improve the utilization of system resources, it is proposed to perform further tracing only for calls that excite the analyzer triggers, as shown in Figure 3.

It is necessary to create a list of trusted applications that have access to the knowledge database and log files of the audit subsystem for its update and modification (among them should be singled out service applications that are included in the list of system utilities and daemons required for normal system functioning).

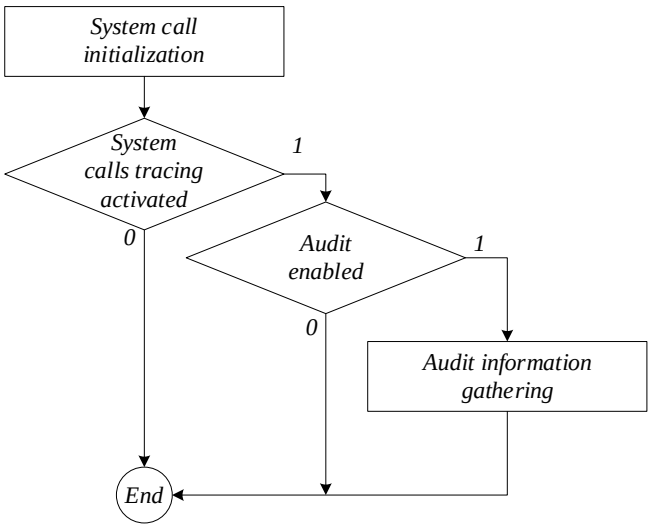

Fig. 3. Audit information gathering

According to this algorithm, the need for logging is determined only if the current entity complies with the audit policy.

C. Classifier model

The system calls classifier model is based on the usage of deep learning methods that allow identifying the features according to the statistical structure of the input data. The problem of determining system calls is solved by the application of an approach commonly used in the text classification preventing the influence of the data order and allowing to determine the harmfulness of the sample according to the existing context. The feedforward neural network, namely the Rosenblatt perceptron, is chosen as a network model. Learning in such a network is carried out by applying the backpropagation method to minimize the standard error of the network in the training sample. The main building blocks are layers of data processing modules that can be considered as data filters. The inputs of the layers receive data (tensors). The network structure is shown in the Figure 4.

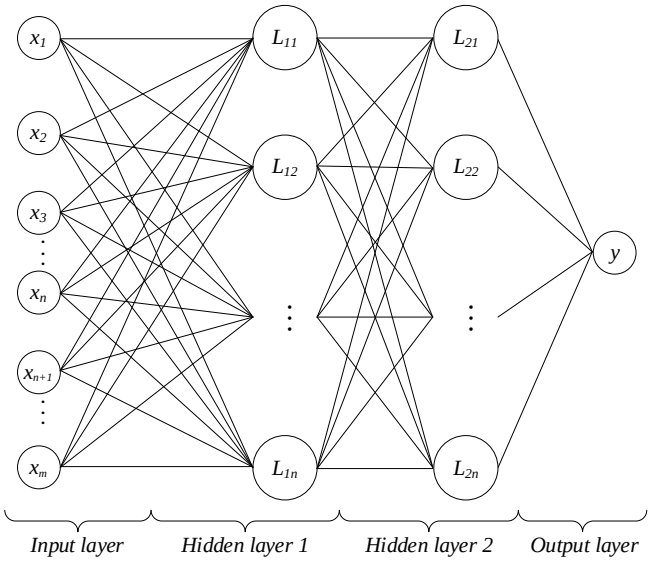

Fig. 4. Neural network architecture

The selected neural network model contains three sequentially fully connected (Dense) layers: the first two contain the same number of neurons and the output layer consists only of one neuron for binary classification implementation.

IV. MODELING

A. Preconditioning

The malware samples were collected on the Internet to create training and test data sets. As no permitted databases are containing malicious software, the search was carried out on public platforms. The bash-script was developed for automation data gathering during the execution of malicious and benign software in an isolated environment. The script was also used to clean captured log files from redundant data. The total number of system calls gathered during automated processes execution is given in table 1.

TABLE I. THE NUMBER OF SYSTEM CALLS IN THE GATHERED DATA

Total number	*Benign system calls*	*Malware system calls*
6 233 791	917 272	5 316 519

The classifier model was launched to test the adaptive threat detection method based on the operating system audit subsystem data capturing mechanism. The modeling results are shown in Figure 5.

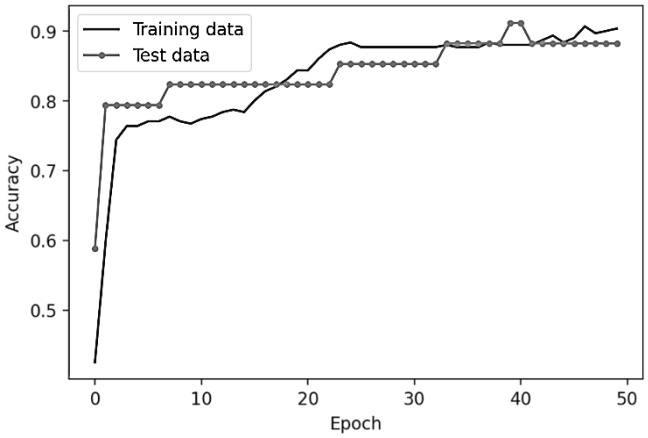

Fig. 5. Modeling accuracy plot

The graph obtained in the process of work demonstrates the correctness of the proposed method. According to the results gained on the test set, the accuracy of malware detection increases from 43% to 90%.

As a result of method work modeling, positive results were obtained on the recognition of threats by performing an analysis of the records of system calls captured during the execution of malicious and benign samples.

CONCLUSIONS

The presence of malicious software high amounts leads to material damages to both, large corporations and society as a whole. There is a strong need for new analysis methods invention to protect information systems.

The results of the suggested study indicate the possibility of the proposed method application as a part of security means in modern protection systems.

The adaptive threat detection method has its advantages and disadvantages. The main benefit is the usage of a built-in tool in the form of an audit subsystem as an interception system for controlling events, so there is no need to develop an additional mechanism for interaction with system calls. As disadvantage should be considered the increment of influence on the system's productivity. Therefore, further researches are needed to eliminate this shortcoming and optimize algorithm work.

REFERENCES

[1] 2020 State of Malware. Last accessed: 12-Sep-2020. [Online]. Available: https://resources.malwarebytes.com/files/2020/02/2020_State-of-Malware-Report. pdf

[2] 110 Must-Know Cybersecurity Statistics for 2020. Last accessed: 28-Sep-2020. [Online]. Available: https://www.varonis.com/blog/cybersecurity-statistics/

[3] Internet of Things statistics for 2020 – Taking things apart. Last accessed: 28-Sep-2020. [Online]. Available: https://dataprot.net/statistics/iot-statistics/

[4] Sobers R. 10 Must-Know Cybersecurity Statistics for 2020. Last accessed: 30-Sep-2020. [Online]. Available: https://blogvaronis2.wpengine.com/cybersecurity-statistics/.

[5] Yu H. Needle in a Haystack: Attack Detection from Large-Scale System Audit / H. Yu, A. Li, R. Jiang. // IEEE 19th International Conference on Communication Technology (ICCT). – 2019. – С. 1418–1426.

[6] Bates J. Analysis of Computer Audit Data to Create Indicators of Compromise for Intrusion Detection / J. Bates, S. Millett, M. Toolin. // DATASCIENCEREVIEW. – 2019. – №2.

[7] Raghuraman C. Static and dynamic malware analysis using machine learning / C. Raghuraman, S. Suresh, S. Shivshankar. // First International Conference on Sustainable Technologies for Computational Intelligence. – 2020. – С. 793–806

[8] Arora A. Permpair: Android malware detection using permission pairs / A. Arora, K. Peddoju, M. Conti. // IEEE Transactions on Information Forensics and Security. – 2019.

[9] Casey E. Malware Forensics: Investigating and Analyzing Malicious Code / E. Casey, J. Aquilina, C. Malin. // Syngress. – 2008.

[10] Eilam E. Reversing: secrets of reverse engineering / E. Eilam, E. Chikofsky. – Indianapolis: Wiley, 2005

[11] Sikorski M. Practical Malware Analysis / M. Sikorski, A. Honig. // Network Security. – 2012. – №12. – С. 4–12.

[12] A survey on automated dynamic malware-analysis techniques and tools / M.Egele, T. Scholte, E. Kirda, C. Kruegel. // ACM Comput. Surv. CSUR. – 2012. – №44. – С. 6.

[13] Aslan Ö. Investigation of possibilities to detect malware using existing tools / Ö. Aslan, R. Samet. // IEEE/ACS 14th Int. Conf. Comput. Syst. Appl. (AICCSA). – 2017.

[14] ASLAN Ö. A Comprehensive Review on Malware Detection Approaches / Ö. ASLAN, R. SAMET. // IEEE Access. – 2020. – №8. – С. 6249–6271.

[15] Event Tracing. Last accessed: 5-Oct-2020. [Online]. Available: https://docs.microsoft.com/en-us/windows/win32/etw/event-tracing-portal.

[16] CHAPTER 7. SYSTEM AUDITING. Last accessed: 30-Sep-2020. [Online]. Available: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/security_guide/chap-system_auditing