Research on Software Behavior Evaluation Method Based on System Calls

Yaling YU[1], Xiao DONG[2], Jingfeng XUE[2], Chen HE[1], Xuyang ZHOU[1]

[1]15th Research Institute of China Electronics Technology Group Corporation, Beijing, 100083, China

[2] School of Software, Beijing Institute of Technology, Beijing, 100081, China

Keywords: Software Security; Software Behaviors; Knowledge Base; System Calls; Behavioral Evaluation

Abstract. Software anomaly behavior detection depends on the software behavior modeling. The key of software behavior model based on system call is the maturity level of the model. Based on the analysis of currently used software behavior modeling method, Combined with variable-length sequence model and VT-Path virtual path model's advantages, an improved software behavior model is presented, and apply this model to software behavior evaluation system. Experimental results show that the software behavior evaluation system based on this model is able to evaluate and judgment the malicious behavior of software.

Introduction

In the current information age, information is an important resource, and it is facing increasingly grim security situation. Software is the tool to control information. Software's security is directly related to information security [1]. For software, if it is attacked by a hacker or a virus, software security will be greatly decreased. User information will be got by hackers. For software testing, within a certain range can detect the whether the behavior of the software is normal, so that we can know whether the software is under attack, if it's under attack we can deal with it immediately in order to make sure the security of information [2].

In recent years through the research, dynamic assessment software has made great progress, research includes system call sequences、software automation、system call context and parameters and so on [3], most of these studies are based on the Linux platform, lack of in-depth research for Windows programs.

This paper based on Win32 software, using open source tools to get the system calls of running software and intercepts and analyzes those information., create system call model and knowledge base for each software, through a degree of normal training, get a better knowledge, Then we combine variable-length sequence model and VT-Path virtual path model, we create an improved software behavior model. Train the model in this way, by compare the runtime system call sequence with the normal knowledge. We can easily know whether the software's behavior is normal.

The Software Behavior Model

Software behavior evaluation based on system calls is monitoring the act of software, gets information about its trajectory and system calls in the running, and use the model to analysis these sequences of system calls, finally evaluate whether the behavior of software is normal [4]. When software is attacked by hacker or virus, internal processes will change, can also load additional libraries which is not loaded before, so it may leave a mark on system calls which can be detected by system call analysis, and we can find the exception immediately [5]. The key of software behavior model based on system call is the maturity level of the model. The amount of system call is a large number. Little change on model can lead a great change on the result. Up to the present, main research directions is how to use this information to form a more perfect model.

The N-gram model mentioned by Forrest in 1998 is the first software behavior model based on system call, its main ideal is to use an N-length system call sequences to express an action in a normal behavior, and so the entire sequence can be saw as a big behavior. For one program, once its normal knowledge is recorded, we can use it to evaluate the behaviors later. The way to evaluate is to count the number of system calls that are not match to the knowledge base [6]. The model is simple and easy to understand, implement and have advantages of real-time monitoring, bug the detection capabilities are very limited and very high false positive rate.

N-gram cut fixed-length sequences of system call, but as a matter of fact different functional pieces can’t be the same length, so there is a need to find every functional piece's variable length sequences, based on this, Wepsi proposed a new model named Var-Gram which has variable length sequences [7]. It can better meet the realities and the new model has better detection capability than N-gram, but it needs more time and memory.

Because of that sequence models all have a high false alarm rate and they can't show the procedure branching and looping constructs, researchers need a better model. Sekar proposed the FSA model, this model use the finite-state automata to show behavior of software. The FSA model can be dynamic trained, it combine the system calls and PC information to create the finite-state automata, the node in automata is the PC information and the state transitions is system calls [8].

Whether abstract stack model or dynamic established FSA model, they all need to create automata, so they all got a time and memory problem and some path is unreachable. In order to face this problem Feng proposed VT-Path (Virtual Path) model. This method create normal software model by dynamic training, it use the function return address, the program counter value and system call information to create a hash table, one for store the function return address list, the other store the virtual path. Using this two hash table it can detect kinds of attacks and avoid from create an automata and unreachable path [9]. But it needs to store two hash tables. The memory may be a large number.

In general, when creating the model of software behavior, we need to pay more attention to is the completeness and reference factor, and this becomes the main research direction.

Improving the Software Behavior Model

This paper combines the ideas of several different models, and proposes a new strong comprehensive model. First of all, the system capture and analysis software system call information to be tested, save the system call information of normal behavior as a part of original knowledge, after many times of training we can cross and matching different train data, at last get a list of short sequences of unequal length, these short sequences can be seen as the label of normal behavior.

At the same time, when evaluating the software behavior using these short sequences, we consider about every single system call, every short sequence and every train log. Using the return address of a system call and its dynamic link library entrance address to calculate the offset, and use this offset and system call to form a part of original knowledge, that is a two tuple of system call sequences and relative memory offset. Train the model many times, through continuous accumulation, we have access to each system call common memory migration and can create a relatively complete knowledge library. At last, create the model based on the library. When evaluating, judge it by the sequence and relative memory offset, if a system call memory is different from the offsets recorded, the program might be under attack.

Our system will use the method of dynamic analysis and modeling, training by using the above theory, and calculate the upper and lower limits of detection judgment threshold through the evaluation knowledge of all the training content values, finally according to the upper limit and lower limit and software's running traces to detect the running state of the software.

Design of Prototype System

Based on the improved software behavior model, this paper describes the design of evaluation system of software behavior based on system call. The system is divided into 5 modules: monitor, information analysis module, knowledge base, evaluation and detection module and the database operation module. The system flow chart is shown in Figure 1. In figure 1, monitor is based on open source tool SoftSnoop, it cans tracking and monitoring programs, and save a system call log. This module is the base of the whole system.

Train Knowledge Base

The system first loaded SoftSnoop monitoring log files, extract the module entrance, exit address, then extract all the sequences of system calls the return address, the function name, dynamic link library information and so on, after this analysis these information and learning. The steps are as follows:

a) Extraction of entrance address of each module.

b) Extraction of each system function call content, for each system function calls, and extraction of the function name, the dynamic link library name, return addresses, and other information.

c) According to the return address of each system call and entrance address of the module, to calculate the relative offset.

d) Save all learning contents into the learning record to cross each other and contrast.

e) Each learning process, learning content will be inserted after learning record table, crossing comparison the learning content and learning record already exists, extraction of cross sequence with fixed sequence, at the same time save the confirmed sequence into the confirmation of knowledge base, and according to the cross match degree evaluation factors determine the value of the sequence.

f) For the learning records, use confirmation sequence segment and its evaluation factor of knowledge, calculating the evaluation value to determine the degree of importance of this time of study.

g) A learning process is complete.

The complete process is shown in Figure 2.

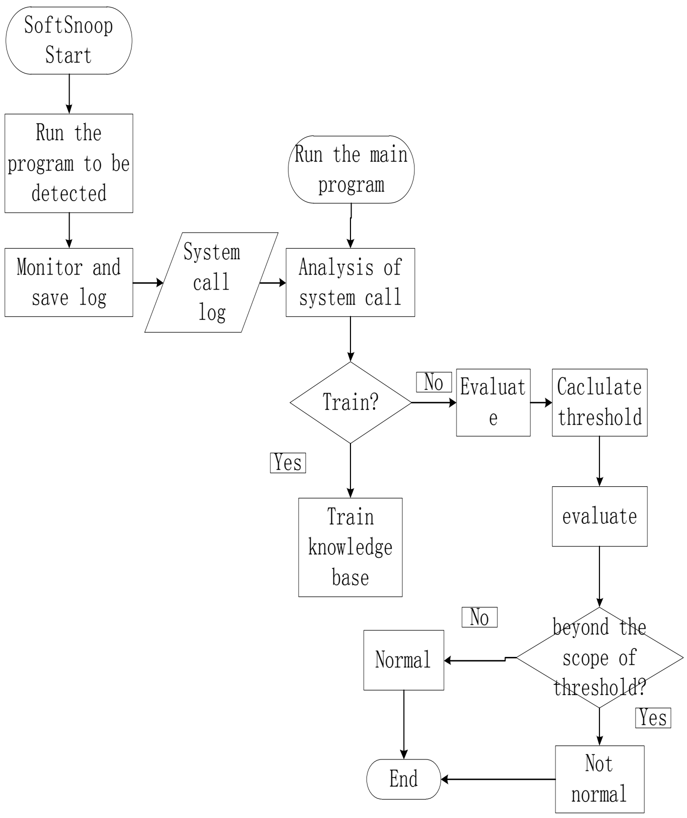

Fig.1. System flow chart

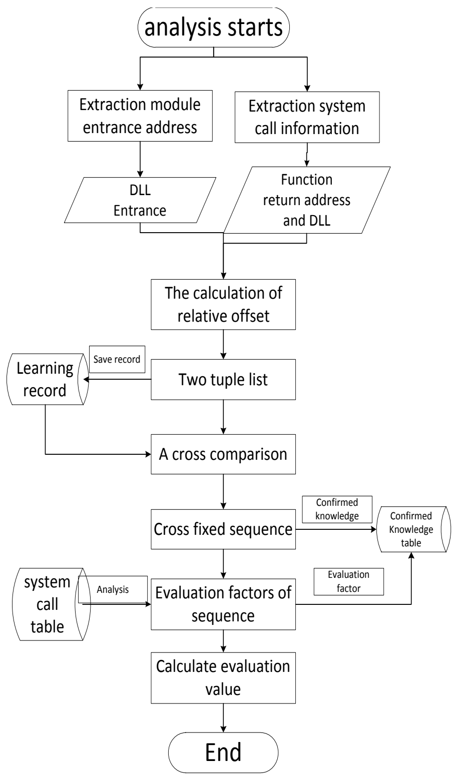

Fig.2. Learning process

Calculate the Evaluation Value

System for each confirmed knowledge have different evaluation factor, according to the matching degree of the evaluation factors and the detection process we can calculate the evaluate value. At the same time the system for each learning record will automatically calculate the learning evaluation value, the calculation method and the detection process are the same, the value of evaluation offers an effectiveness reference of the learning process, and we can consider deleting some learning record with low evaluation value to keep knowledge correct.

(1) Determine evaluation factor

For each study process, the learning records will be cross compared with records already there, extract all common subsequence and save them into the confirmed knowledge base. If the sequence is already present in the confirmed knowledge base, then the evaluation factor will increase. Otherwise add one new confirmed knowledge and set the evaluation factor to be 1.0. The process is shown in Figure 3.

(2) Evaluation value of behavior

When the confirmed knowledge is already existed, the system can use confirm knowledge to evaluate a behavior of the software. Here will use the relevant pattern matching algorithm, first get all confirmed knowledge from the confirmed knowledge base, then doing Pattern matching between these knowledge and the system call sequences to be detected.

Evaluation module set of two relatively independent evaluation value, they together they determine the final evaluation value, for each detected sequence, D represents the total evaluation values derived by knowledge matching, Y represents the evaluation by comparing single system call and relative memory offset values derived, T represents the number of training of the test program, S represents the number of system calls that sequences to be detected contained. The final evaluation value is calculated by the following ways:

$$K = D + \frac{Y}{S}$$

The system calls the evaluation process as shown in Figure 4.

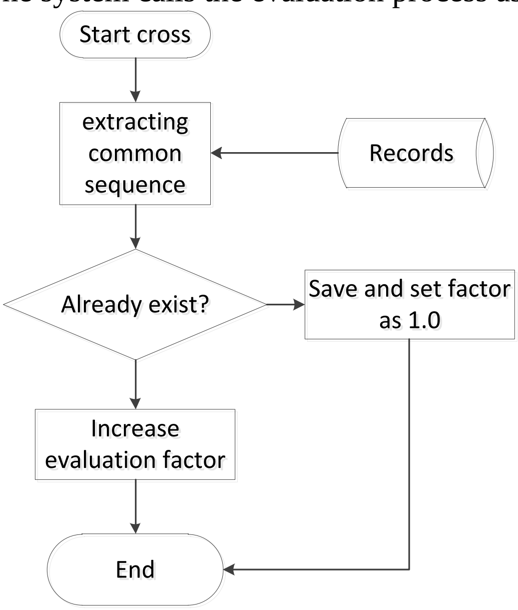

Fig.3. Process of determine evaluation factor

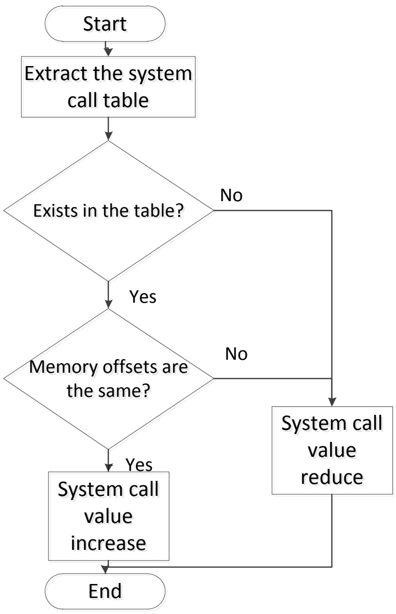

Fig.4. System calls the evaluation process

The evaluation process is as follows:

a) Once monitoring module monitored software behavior save a system call log.

b) Monitor analysis module analyzes the information, extracting all modules loaded information and system call information to form a "offset-system calls" two tuple list.

c) Based on the above list, training module will get all confirmed knowledge of this program and their evaluation factor, according to the train time T, using the following formula to calculate the average effect factor of each short sequence knowledge Di:

$$Di = \frac{1}{T}$$

Then matching the knowledge sequence in all short sequences to be detected, add the evaluation factor to the final evaluation value if match success. Use the following formula to calculate the total evaluation value D through knowledge matching:

$$D = \sum_{k=1}^{n} Di * Ei$$

Ei represents the knowledge sequence matching result is 0 or 1.

d) At the same time, analysis the system call sequences two tuple list to be detected, using the system call table data, compare whit the data in list, increase or decrease the evaluation value according to the matching result.

(3) Evaluation value of learning record

For each study, we save all the learning content record to learning record table. After train, evaluate all learn record through the updated new confirmed knowledge and calculate every learn record's new evaluation value. We can also decide whether one learn record is effective. The significance of the evaluation value of learning record is that we can remove unreasonable learning, to ensure the completeness and correctness of knowledge.

Result

To test the effectiveness of the evaluation method based on system call, this paper design and implement a prototype system and conducted related experiments. System runs on Win32 platform, using C# language development, knowledge database based on SQL Server. In the process of verification, this paper uses a specific example of a complete detection. The system will first carry out a large amount of training on the knowledge base, get relative perfect knowledge, then simulate the situation that software is under attack, at last, using system detecting and judging the behavior of the software.

We run the program to be tested 10 times, and we already have more than 2500 knowledge, all test data are shown in Table 1.

Tab.1. Test data

No	Content	size(KB)	number	result
1	Normal	1391	5010	First train
2	Normal	1411	5085	Confirm 100 seq
3	Normal	1422	5125	Confirm 271 seq
4	Normal	1454	5228	Confirm 502 seq
5	Normal	1492	5372	Confirm 800 seq
6	Normal	1444	5200	Confirm 1124 seq
7	Normal	1382	4976	Confirm 1452 seq
8	Normal	1425	5129	Confirm 1806 seq
9	Normal	1386	5007	Confirm 2157 seq
10	Normal	1396	5028	Confirm 2511 seq
11	tampered one function	1332	4855	Detect abnormal
12	tampered two function	1324	4802	Detect abnormal
13	tampered three function	1247	4554	Detect abnormal

In order to imitate the program has been attacked, use OllyDBG to disassembly the program, modify the program modules process, then save the new program and run it with SoftSnoop. This test modified multiple places of the program, and tested separately, the test results all shows abnormal. It is indicated that the system can correctly detect abnormal changes after training.

We just modify part of the function of the program and the initialization and most functions are not changed, so system matched some sequences successfully bug fail in some. The final evaluation value is 1.255 which is above the threshold limit, so system determined this behavior is abnormal.

Conclusion

This paper made a comparative analysis of several current software behaviors modeling method based on system call, combining the respective advantages, proposes an improved software behavior modeling method, using the modeling method to design the software behavior evaluation system. The prototype system monitoring the system call when software running, save system call information as log, through analysis the log file, extract useful information to form the original knowledge, and then create model by these knowledge. Combine the thought of Var-Gram model and VT-Path model, evaluate software behavior in two ways and get final evaluation value. Evaluate the behavior according to the final evaluation value and threshold limit, so as to achieve the purpose of detecting software abnormal behavior.

Acknowledgement

In this paper, the research was supported by the Key Project of National Defense Basic Research Program of China under Grant No. B1120132031 and the Ph.D. Programs Foundation of Ministry of Education of China under Grant No. 20131101120043.

References

[1]Yang Xiao-hui. Research on software behavior dynamic trusted theories and models. University of Science and Technology of China [D], 2010.

[2]Sun Di, Li Jian, Wang Zhi-yong. A software behavior dynamic trusted research method. Network Security Technology & Application [J], 2013: (4): 14-17.

[3]Yang Xiao-hui, Zhou Xue-hai, Tian Jun-feng, Li Zhen. Novel dynamic trusted evaluation model of software behavior. Journal of Chinese Computer Systems [J], 2010: 31(11): 2113-2120.

[4]Alessandro F, Federico M, et al. Selecting and improving system call models for anomaly detection. Proceedings of the 6th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment [C], Como, Italy, 2009: 206-223.

[5]Federico M, Matteo M, Stefano Z, Member. Detecting intrusions through system call sequence and argument analysis. IEEE T DEPEND SECURE [J], 2010: 381-394.

[6]Han Jin-e. Creditability evaluation model of software based on simplified behavior trace. Hebei University [D], 2011.

[7]Wang Fu-hong, Peng Qin-ke, Li Nai-jie. Intrusion detection using variable-length system calls patterns. Computer Engineering [J], 2006: 32(20): 143-146.

[8]Wang Kou-wu, Zhang Jun-ming, Long Shi-Gong, Dong Fang. Algorithm research based on FSA model checking. Journal of Guizhou University (Natural Sciences) [J], 2012: 29(5): 58-62.

[9]Tao Fen, Yin Zhi-yi, Fu Jian-ming. Software behavior model based on system calls. Computer Science [J], 2010: 37(4): 151-157.