Malware Detection Based on Suspicious Behavior Identification*

Cheng Wang Jianmin Pang Rongcai Zhao Wen Fu Xiaoxian Liu
China National Digital Switching System Engineering & Technology Research Center
Postbox 1001 No. 718, Zhengzhou, Henan 450002, China
Cheng28624113@yahoo.com.cn

Abstract

Along with the popularization of computers, especially the wide use of Internet, malicious code in recent years has presented a serious threat to our world. In this paper, through the analysis against the suspicious behaviors of vicious program by function calls, we present an approach of malware detection which is based on analysis and distilling of representative characteristic and systemic description of the suspicious behaviors indicated by the sequences of APIs called under Windows. Based on function calls and control flow analysis, according to the identification of suspicious behavior, the technique implements a strategy of detection from malicious binary executables.

1. Introduction

Malware is code designed for a malicious purpose, antivirus(AV) tools primarily use signatures to detect known malware. This signature is typically created by disassembling the existing samples of malware, and selecting some pieces of unique code. However,the technique of malware is updating constantly, so that the conventional AV technique based on signature detection could not be able to detect mutation from already known or unknown malicious codes effectively yet. In order to settle this problem, some AV researchers apply with the techniques such as data mining and machine learning to detect unknown malware[1]. In paper[2],the information of character string in PE file head, the number of DLL, the number of API calls and so on are distilled, and normal program and virus are classified by naive bayes algorithm, and the detection result is quite good. However, the information in PE file head could be modified easily, and it has some difference from the true calls of program.

*This work is supported by the National High-Tech Research and Development Plan of China under Grant No.2006AA01Z408.

Under these conditions this paper makes two amendments. Firstly, the true calls of object program of malware are obtained by static analysis of its procedure; secondly, the sequence of function calls[3] in program is built and matching identification from suspicious behavior-base is implemented based on the control flow. Hereunder suspicious behavior are throughly analysed, and the malware detection technique under Windows environment is presented and implemented.

The technique mentioned in this paper has been implemented in a prototype system, named with RADUX(Reverse Analysis for Detecting Unsafe eXecutables),which is composed by Decompilation(as figure 1-① shows), Analysis of program behavior, Database of suspicious behavior models, Detection, annotation, removal and so on. Based on decompilation model of our prototype Radux, this technique mainly fulfills the design of two following modules:

1) Expand the module②--Behavior Identification: distill API function calls sequence known as malware and add it to suspicious behavior-database based on API function calls for expansion, so rules for identifying models is obtained. Then build API function calls sequence of object program by control flow analysis to suspicious behavior identification.

2) Build the module③--Program Detection: use bayes algorithm determined by abundant sample space statistics to find malicious degree of unknown object program, and classify unknown program by changing threshold, so the detecting of malicious executables is completed. The framework of our prototype system is shown in figure 1.

2. Related works

There are two main approaches for the detection of malware: static analysis and dynamic analysis. Static analysis examines the binary code to determine properties of this program without running it. This technique was first used by compiler developers to

978-0-7695-3557-9/09 $25.00 © 2009 IEEE
DOI 10.1109/ETCS.2009.306

optimize the code[4]. It is also used in reverse engineering and for program understanding[5]. It is not long since it was used for the malware detection. Dynamic analysis mainly consists in monitoring the execution of a program to detect malicious behaviour. Compared with static analysis, dynamic analysis has some disadvantages in the detection of malware[6].

This paper [6] approaches the problem of detection of malicious code in executable programs using static analysis. It involves three steps: generation of intermediate representation, analyzing the control and data flows, and then doing static verification. Static verification consists of comparing a security policy to the output of the analysis phase. A brief description of a prototype tool is presented as well.

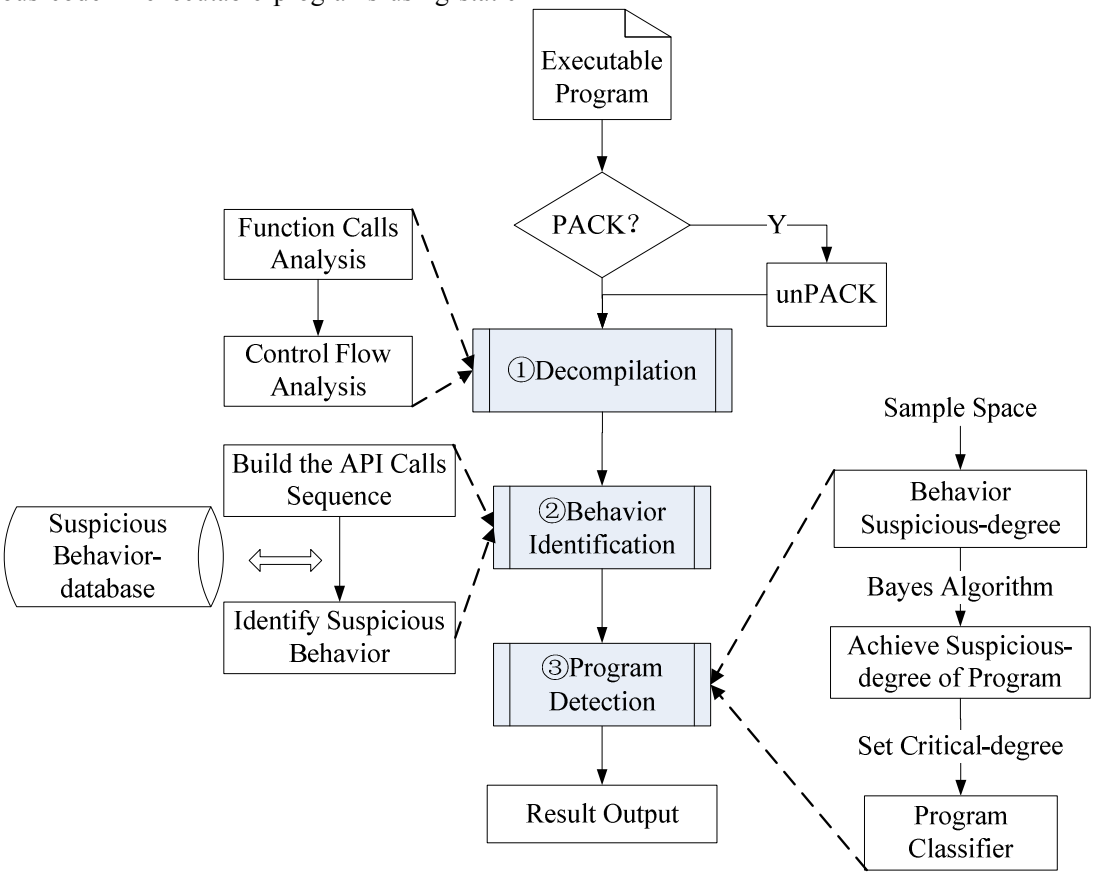

Figure 1: The project for framework of prototype system

3. Testing methodology

Bayes algorithm is a sort of method to compute probability, which calculate the posterior probability according to prior probability. Evidently it is suitable for approximate determinant of malicious code. The task of system of malware detection based on bayes algorithm: determine whether the program is malicious or not by suspicious behavior of program and so whether it is a malicious program or not. Bayes expressions—the foundation of bayes theory is：

$$P(C/F)=\frac{P(F/C)\times P(C)}{P(F)}$$

1） Establish sample space and calculate the appearance frequency of behavior in the training sets

Sample space established by collecting a large number of executable codes are divided into the training sets $S_{training}$ and the test sets S_{test}（the training sets and test sets in sample space are not intersectant, $S_{test}\bigcap S_{training}=\varnothing$）. The training sets are divided into malicious program sets $S_{malicious}$ and benign program sets S_{benign}（$S_{malicious}\bigcup S_{benign}=S_{training}$，$S_{malicious}\bigcap S_{benign}=\varnothing$）, then the tool for distilling behavior is constructed according to malware type(as virus,trojan,etc) to calculate the frequency of every suspicious behavior in the training sets.

2）Set up a hash table for the training sets and compute suspicious-degree of behavior

The detection of computer malware is a problem of classification, which are malicious program and benign program. Define C to be classify sets in form of {malicious, benign}, and C denotes malicious program while $\overline{C}$ denotes benign program as it is a variable. Set up hash tables for malicious program and benign program sets with the name hashtable_malicious and hashtable_benign to store the mapping from suspicious behavior to behavior probability. The appearance probability of a certain behavior characteristic ω_i （ $\omega_i \in \omega$ ） in hashtable is:

$P(\omega_i / C)$ = the frequency of ω_i in hashtable_malicious / the length of the corresponding hashtable;

$P(\omega_i / \overline{C})$ = the frequency of ω_i in hashtable_benign / the length of the corresponding hashtable, what showed in table 1 is an example.

Table 1. Suspicious probability of one behavior

P("Search files to infect"/$\overline{C}$)=19/289
P("Search files to infect"/C)=127/282
P("Distribute virtual memory "/$\overline{C}$)=55/289
P("Distribute virtual memory "/C)=72/282

3) Compute the program malicious degree using bayes algorithm

Distill a set of behavior vector sets ω from each program in sample space, which contains ω_1, $\omega_2 \ldots \omega_n$ (in this paper n=9). ω_i is independent from each other in malicious or benign program sets， but it is not in the whole test sets. We define malicious degree of behavior $P(C/\omega)$ to express the probability of being malware when ω is suspicious behavior sets of program to represent, so bayes expressions can be rewritten using probability expressions mentioned above:

$$P(C/\omega) = \frac{\prod_{i=1}^{n} P(\omega_i/C) \times P(C)}{\prod_{i=1}^{n} P(\omega_i/C) \times P(C) + \prod_{i=1}^{n} P(\omega_i/\overline{C}) \times P(\overline{C})}$$

By the analysis of the formula above we know: when there is a greater probability of suspicious behavior we choose in malicious program than in benign program, the value of $P(C/\omega)$ is between 0.5 and 1. In order to represent more precisely the probability in which a program is malicious, we use $f(x) = e^x$ to map from the function for adjusting result $P*(C/\omega)$ into 0~1, which felicitously reflects the probability in which a program is a malware.

4) Load the testing sets and set threshold to validate the veracity of the algorithm

Load the testing sets $S_{training}$ and compute the malicious degree of program one by one using the above formula and suspicious degree of every behavior, then classify the computed malicious degree by threshold. Compare the classification with the former one to validate the veracity of the algorithm. Finally,the result from program detection is exported by prototype system.

4. Malware detection

4.1.Describing suspicious behavior

Every program which wants to achieve its goal always takes action. No matter how crafty the malicious code is in disguise, it always has some different, relatively peculiar action which is called suspicious behavior. Behavior identification is becoming the direction of anti-virus. As Windows operating system is widely used, it rapidly catches the malware' eye and becomes the mainly growing environment and attacking object of computer vicious code. Currently most of the malicious programs are under Win32 environment. The popular vicious code for the nonce always use API function provided by Windows operating system to implement their functions, aiming at the size of code predigestion and the effect mightiness. The computer vicious program always infect normal program, and carry out their malicious purpose when the infected program is running. Some examples of suspicious behavior based on API-calling sequence are as follows:

1） Search files to infect

Describing suspicious behavior: find the types of relative files going to be infected, which are for operation of file infection.

API function calls sequence:

①FindFirstFile

②FindNextFile

③FindClose

2） Create mapping of file

Describing suspicious behavior: map files in disk to virtual address space of process to improve the accessing speed.

API function calls sequence:

①CreateFileMapping

②MapViewOfFile

③UnMapViewOfFile

As two suspicious behaviors above, the detailed description of behavior known as malware based on API function calls is shown table2.

Table 2. The description of suspicious behavior

Behavior number	Suspicious behavior description	API function calls
Behavior 1	Obtain the system directory	GetWindowsDirectory,GetSystemDirectory
Behavior 2	Search files to infect	FindFirstFile,FindNextFile,FindClose
Behavior 3	Create mapping of file	CreateFileMapping,MapViewOfFile,UnMapViewOfFile
Behavior 4	File write	CreateFile,OpenFile,WriteFile,CloseHandle
Behavior 5	Modify file attributes	GetFileAttributes,SetFileAttributes
Behavior 6	Modify time of file	GetFileTime,SetFileTime
Behavior 7	Distribute global memory	GlobalAlloc,GlobalFree
Behavior 8	Distribute virtual memory	VirtualAlloc,VirtualFree
Behavior 9	Load register	RegOpenKey,RegCreateKey,RegSetValue,RegCloseKey

4.2. Implementation

The implementation of behavior identification is described below: distill the relation of each function calls from binary executables in training set and suspicious behavior identification from function calls sequence by finite automaton(figure 2). Nodes denote state sets, the Bad state predicating identify the suspicious behavior is representative the final state, λ is any action except RegCreateKey, RegSetValue, RegCloseKey. If API function calls distilled from program will arrive at the Bad state of the suspicious behavior, the program is considered to have the suspicious behavior.

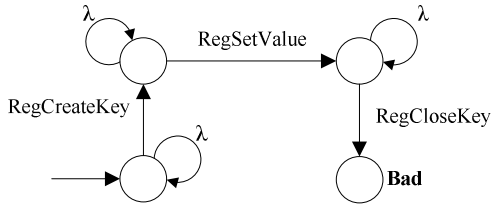

Figure 2: Automaton for Behavior Detection

5. Experimental results

Sample data for experiment is shown in table 3. The total number of sample in space is 914, which are divided into benign program and malware. Benign program are selected from operating system. The programs we choose are all 461 of the PE files under Windows directory after the installation of Windows XP for the first time. Besides, 453 malicious programs, 80% of which are training set and 20% are testing set，are downloaded from the VX Heavens[7] website.

In order to obtain function calls of program in sample space, we write an API call tracer to implement all blocks of API function calls under Windows environment.

Table 3.Sample data of experience

	Sample Data	Training	Testing
Benign	461	369	92
Malicious	453	362	91
Total	914	731	183

There are 4 results after the algorithm: (1)consider a benign program to be a benign program, which is called TN-True Negative; (2)consider a benign program to be a malicious program, which is called FN-False Negative; (3)consider a malicious program to be a benign program, which is called FP-False Positive; (4)consider a malicious program to be a malicious program, which is called TP-True Positive. In order to review the veracity of the classification of the algorithm, the detection precision (DP) are showed using detection rate and false positive rate.

$$DP = \frac{TP + TN}{TP + TN + FP + FN}$$

The relationship between setting threshold and the detection precision is showed in table 4, using the detection results of testing set. As we see in table 4, the detection precision of unknown malware in testing set using bayes algorithm achieves 93.98% when threshold is 0.7, which are higher than 89.07% in paper[3]. It indicates that the detection technique in this paper has a better effect for detection of unknown malware.

Table 4.The compare of threshold setting

Threshold	TP	TN	FP	FN	DP
0.75	87	83	9	4	92.89%
0.8	86	86	6	5	93.98%

0.85	83	88	4	8	93.44%

6. Conclusions and future work

The detection technique presented in this paper is based on identifying API-calling sequences under Windows environment. The technique involves a bayes algorithm, which is used to detect flow of suspicious behavior through the analysis of API functions invoked by malware. The precision of detection of the algorithm has been validated by the training and testing of abundant sample space. The technique is a promising method to detect the win32 virus.

References

[1] M. Christodorescu and S. Jha. Static Analysis of Executables to Detect Malicious Patterns. In: Proceedings of the 12th USENIX Security Symposium, USENIX Association, BerKeley, CA, USA, 2003, 169-186.

[2] M. G. Schultz, E. Eskin, E. Zadok. Data Mining Methods for Detection of New Malicious Executables. In: Proceedings of the 2001 IEEE Symposium on Security and Privacy. Washington: IEEE Computer Society, 2001, 38-49.

[3] B. Zhang, J. Yin, J. Hao, D. Zhang, and S.Wang. Using support vector machine to detect unknown computer viruses. International Journal of Computational Intelligence Research, 2(1):100–104, 2006.

[4] Cristina Cifuentes and Antoine Fraboulet. Intraprocedural static slicing of binary executables. In Proceedings of the International Conference on Software Maintenance, Bari, Italy, October 1997, pages188–195, IEEE-CS Press.

[5] S. S. Muchnick. Advanced Compiler Design Implementation. Morgan Kaufman Publishers, San Francisco, CA, 1997.

[6] J. Bergeron, et al. Static Detection of Malicious Code in Executable Programs, in Symposium on Requirements Engineering for Information Security (SREIS'01), 2001.

[7] VX.heavens. http://vx.netlux.org.