Malicious Code Detection Based on Software Fingerprint

Zhimin Yin,Xiangzhan Yu and Linhua Niu
Department of Computer Science and Technology
Harbin Institute of Technology
Harbin, China
yzm0621@163.com,yxz@hit.edu.cn, xn12280000@126.com

***Abstract*—The malicious code on the network is increasingly rampant that the traditional detection method of characteristic code has been difficult to deal with malicious code, with features of various variants, deformations and other problems. In this paper we present a new static analysis model based on software fingerprint to distinguish malicious codes. Through obtaining the software call graph by disassembling the binary file and mapping it as an image, shape moments can be obtained as the software fingerprint based on the retrieval theory of content image, combined with moment theory and the image's color, texture and shape features. The idea of pattern matching is used to measure the extracted software fingerprint similarity to determine whether it is malicious code or not. Then, we analyze the collected program samples. Test and verify whether the program has good performance in uniqueness, invariability and sensibility.**

Keywords: malicious code, static analysis model, software fingerprint, shape moments.

I. INTRODUCTION

With the rapid popularization of Internet and corporate information technology acceleration, the computer is a great convenience to people's lives, whether it is shopping, leisure or work and more obvious the importance of the Internet, but due to the openness of the Internet and flexible application and operating system vulnerability so that people can enjoy the benefits brought by the computer at the same time, also is experiencing distress and abuse of all kinds of malicious code threats to network security events increased year by year. Network security incidents, the most serious harm caused by malicious code, causing huge economic losses to the country as a whole, society and the individual, information security has become a major challenge facing. Domestic and foreign researchers turned to the semantics of malicious code, trying to judge the signatures of the two deformation malicious code through the instruction-level semantics rather than program syntax, further can determine whether the deformation of the malicious code. Trying to evade detection of malicious code, malicious codes are disassembled and for standardization or stack analysis system call for the use of fuzzy transformation technology to discriminate. In recent years, the researchers also uses malicious code detection technology engineering methodology, based on feature detection based on the application of malicious code detection technology based on data mining and machine learning.

However, while foreign research scholars during the malicious code detection and anti-virus software R & D, the attacker malicious code using anti-debugging techniques, anti-Hook technology, to detect whether the code being debugged, the to find themselves debugger or analysis environment, malicious code using fuzzy transform technique and a series of anti-debugging measures show some non-anomalous behavior of the code, so as to protect their own purposes. In this process, we need manually assistive technology. Most automated virus analysis software only to capture some of the behavior, security experts need further analysis and screening of experimental results, and finally determine the extent of the harm of malicious behavior.

In summary, the malicious code on the network more and more rampant code detection method for the traditional characteristics of difficult to deal with malicious code variants, deformation problem, solve the problem of the detection of unstable deformation caused due to malicious code upgrade has become a research focus and difficulty, but also of the issues that must be resolved key issues.

II. RELATED TECHNOLOGY RESEARCH

Malicious code analysis is to extract the characteristics of malicious code and then the malicious code detection of the foundation and prerequisite. In accordance with the process of analysis, whether to execute malicious code, malicious code analysis technology is divided into two types of static analysis and dynamic analysis technology.

1) Static analysis techniques

The static analysis technical analysis method does not perform binaries, such as disassembly, source code analysis, binary statistical analysis, decompile, belong to the reverse analysis methods [1,2]. Determined by static analysis of malicious code after its general structure, the feature string, feature code section [3]. Analysis of disassembled code can also get the program execution flow chart to determine whether a piece of the system what impact. Currently, analysis of malicious code and the evaluation software security work, based on a static analysis method is one of the most mainstream ways. In essence, both the normal code and malicious code, they are composed of data structures, and computer instructions, in the static analysis process based on whether or not to consider the semantics of computer instructions constitute a malicious code, the static analysis

method is divided into two types, i.e. based on code analysis and behavior-based analysis.

2) Dynamic analysis techniques

Dynamic analysis method is the executable code up and running, all operations in a controlled environment to run malicious code, monitor the entire code, to observe changes in the status and implementation process, during the execution of a variety of data. Dynamic analysis tools include: network activity monitoring tool TcpView, dynamic debugging tool OllyDbg, disassemble tool IDA Pro and so on. The most widely used controlled environment is the "virtual machine" , a computer system with malicious code analysis environment isolation, specialized monitoring tools for real-time monitoring of code in the process of implementation. Dynamic analysis in accordance with whether to consider the malicious code semantic features, will be divided into two ways, namely external observation and tracking debugging method [4].

III. SOFTWARE FINGERPRINT

In the process of software static analysis, because of the software has certain files or specific sequence (such as instruction, etc.) .It can't be used in the static analysis of software from the macroscopic angle, causing not structure of fingerprint extraction software quickly. In order to meet this requirement, this paper research in this field is introduced to the theory of complex network, to study the flow diagram of software. Attempts to research software call graph obtained structure characteristics of software as the structure of software fingerprint [5].

Call graph is a description of the software program functions (methods) to call the relationship between the function of a directed graph, it reflects the static abstract structure of the program is the basis for the process control flow graph constructed. In contrast, it is a relatively simple control flow graph [6]. This article discusses the call graph is defined as follows:

Definition 1: Assume that the software call graph is a directed graph. It can be expressed as *G=(V,E).V* represents a collection points to the node and E represents a collection directed edges of the graph. Assume $u,v \in V$, directed edges can be represent as (u,v),That function *u* call function *v*. Figure 1 shows the sample code corresponding to the program call graph.

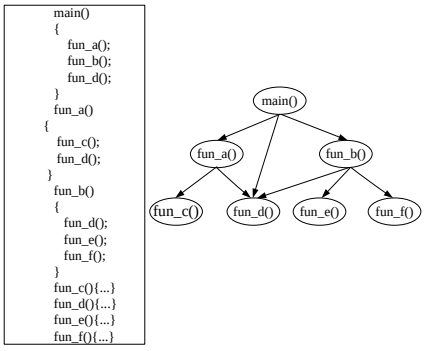

Figure 1 Example program and its procedures call graph

Definition 2: Software call graph can be expressed as $CallGraph = (V, E)$. All functions $\{f_1, f_2, f_3, ..., f_n\}$ and all nodes $V = \{v_1, v_2, v_3, ..., v_n\}$ corresponded in the program. The relationship of software calls with edges in the software call graph $E = \{v_i \to v_j, v_i, v_j \in V\}$corresponded in the program. In the software call graph, $in\deg ree_i$ represents the in-degree of the function f_i , $out\deg ree_i$ for out-degree. $call_{i \to j}$ for the relationship between f_i and f_j .If f_i calls f_j ,then $call_{i \to j} = 1$,otherwise $call_{i \to j} = 0$.

According to the above definition, software call graph can be described as an abstract ternary matrix $(in\deg ree, out\deg ree, call)$ .We call this matrix as software structure matrix. Figure 2 for the sample program call graph and its corresponding software structure matrix.

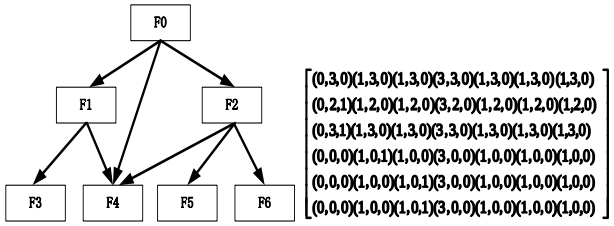

Figure 2 illustrates the program's call graph and the corresponding software structure matrix

IV. IMPLEMENTATION

1) Content-based image retrieval technology

With the rapid development of multimedia technology and the Internet, the image retrieval of information resources has become the focus. The researchers have proposed content-based image retrieval. Herein software fingerprint matches generated a large amount of data can be well combined with the technology for processing [7].

Content based image retrieval is the use of image content to achieve a comprehensive image retrieval technology, according to the characteristics of the image content as well as combinations of features directly from the image library to identify images containing specific content. Ships will feature extraction based on the characteristic of the image content, into three levels. In the original image, the first level is the low-level feature layer and content-based image retrieval technology is at this level, with characteristics such as color, texture, contour and shape; second layer is the object semantic layer that contains object classes and their spatial relationships. The third layer is abstract semantic, it will image content and more abstract concepts of reality, including emotional scenes and behavior. Their hierarchical relationship is shown in Figure 3.

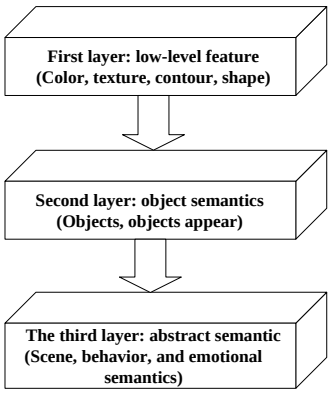

Figure 3 image content features three levels

2) Structure matrix to RGB color space mapping

RGB color space can be used to represent the color of a set of vectors (a, b, c).This software call graph abstract structure matrix consisting of a three element matrix $(in\deg ree, out\deg ree, call)$consistency in the number of elements. So the structure matrix call graph are mapped to the RGB color space transformation through certain methods, so the problem is transformed to the graphics, this will be an easy access to the software structure of fingerprint. Transformation method such as the formula :

$$r = (in\deg ree_i - \min(indgree))/(\max(in\deg ree) - \min(in\deg ree))*255$$

$$g = (out\deg ree_j - \min(out\deg ree))/(\max(out\deg ree) - \min(out\deg ree))*255$$

$$b = \begin{cases} 255 & call_{i\to j} = 1 \\ 0 & call_{i\to j} = 0 \end{cases}$$

3) The Moment Invariant software-based fingerprint extraction

Due to the expression of Hu moment invariants are available trigonometric invariance launch, As follows:

$$\phi_1 : \cos\gamma_1 + \sin\gamma_1$$

$$\phi_2 : \cos(2\gamma_1 - 2\gamma_2)$$

$$\phi_3 : \cos(3\gamma_1 - 3\gamma_2)$$

$$\phi_4 : (\cos^2\gamma_1 + \sin^2\gamma_1)(\cos^2\gamma_2 + \sin^2\gamma_2)\cos(\gamma_1 - \gamma_2)$$

$$\phi_5 : (\cos^2\gamma_2 + \sin^2\gamma_2)(\cos^2\gamma_3 + \sin^2\gamma_3)(\cos^2\gamma_4 + \sin^2\gamma_4)\cos(3\gamma_1 - \gamma_2 - \gamma_3 - \gamma_4)$$

$$\phi_6 : (\cos^2\gamma_2 + \sin^2\gamma_2)(\cos^2\gamma_3 + \sin^2\gamma_3)\cos(2\gamma_1 - \gamma_2 - \gamma_3)$$

$$\phi_7 : (\cos^2\gamma_2 + \sin^2\gamma_2)(\cos^2\gamma_3 + \sin^2\gamma_3)(\cos^2\gamma_4 + \sin^2\gamma_4)\sin(3\gamma_1 - \gamma_2 - \gamma_3 - \gamma_4)$$

Proved by the complex nature of moment invariants and discussion , our research focus on how to select the two-dimensional density distribution function $f(x,y)$. The gradation of the image (density) is defined as $f(x,y)$.

a) Image gray value

Image gray is the brightness of the image, simple to say is the image color depth. Grayscale range is generally from 0 to 255, white 255, black is 0, so the gray-scale image is a black and white image, use is widespread in the medical field of image recognition.

b) Structure matrix conversion to gray value

Using floating-point images will be converted to grayscale. Gray value is the two bit image density distribution function $f(x,y)$. RGB color space obtained in the above is seen as the size of the two-dimensional image $\{ f(x,y) | x = 0,\ldots\ldots, N; y = 0,\ldots\ldots, N \}$ of the parcel of $N \times N$ .Using discrete instead of integral to calculate the two-dimensional $p+q$ -order moment, get m_{pq} , $p,q = 0,1,2,\ldots\ldots$且$(x,y \in R)$.The we calculate and obtain the centroid x_0, y_0 . The central moments can be calculated according to the above within a third order μ_{pq} .Then, they were normalized scale normalized central moments η_{pq}. Finally calculate Hu deduced seven invariant moments $(\phi_1,\phi_2,\phi_3,\phi_4,\phi_5,\phi_6,\phi_7)$, and thus constitutes the feature vector based the Moment Invariant software architecture fingerprint.

V. SIMILARITY MEASUREMENT AND ANALYSIS OF EXPERIMENTAL RESULTS

1) Similarity measurement

Image similarity measure is based on the extracted image content features, the use of the degree of similarity of the image feature comparison of the degree of similarity of the image itself. Being the image's color, shape and texture, and other visual characteristics is considered to be characterized in the first hierarchy, having relatively objective characteristics. Image similarity measure difficulties are mainly concentrated in the first level, the characteristics of color, shape and texture of the image similarity measure. , We say that the two images is similar, their characteristics so similar, so we must first extract the features then determine the similarity in content-based image retrieval technology. Typically, the image visual characteristics similar method is the most commonly used measure is the vector space model. Visual characteristics as a point in the feature space, by calculating the degree of proximity between the two points to measure the degree of similarity between the image feature. In this paper, we use the Euclidean distance:

$$EUD = \sum_{i=1}^{n} (A_i - B_i)^2$$

The Euclidean distance is calculated and the square of the difference of the two vectors A and B correspond to the respective component. This distance has a good effect, the calculation is not complicated. In this paper we use Euclidean distance.

2) Experimental results and analysis

This paper selects 200 program samples as a test case, concentrated samples including 150 normal and 50 malicious programs.

a) Software uniqueness of fingerprints

Through the sampling analysis, this section of the 200 program sample to carry on the research analysis, extraction software structure of fingerprint, choose some representative

software or procedural listed below, table I lists the different program shape invariant moment values.

TABLE I. DIFFERENT PROCEDURES MOMENT INVARIANT

prog	Φ_1	Φ_2	Φ_3	Φ_4	Φ_5	Φ_6	Φ_7
1	7.11	1.57	8.83	6.83	5.26	-2.10	-4.38
2	3.00	1.04	8.41	5.64	3.77	-1.95	-9.63
3	6.45	5.10	2.40	2.67	6.77	2.68	-4.79
4	3.14	1.25	2.30	1.24	2.03	-3.83	-2.74
5	3.86	2.30	2.69	2.75	2.06	-1.48	-1.69
6	3.08	1.55	5.86	2.55	9.10	-4.00	-1.27
7	1.27	4.79	5.60	5.80	3.31	-2.77	-4.42

Program 1-7 represents 360 security anti-virus software,oicq99b2 instant messaging software,DevCpp compiler,Chrome is the browser,6to4svc.dll system dynamic link library,malware programs 003gangsir and 027gangsir.

Following are their shape invariant moments from the distance used in this paper are the Euclidean distance, such as table II shows.

TABLE II. DIFFERENT PROCEDURES MOMENT INVARIANT DISTANCE

prog	1	2	3	4	5	6	7
1	0	1.9	6764	15.3	5.4	4.7	5.4
2	1.9	0	6766	16.7	4.1	3.1	4.1
3	6764	6766	0	6749	6770	6769	6770
4	15.3	16.7	6749	0	20.7	19.7	20.7
5	5.4	4.1	6770	20.7	0	1.8	0.3
6	4.7	3.1	6769	19.7	1.8	0	1.9
7	5.4	4.1	6770	20.6	0.3	1.9	0

Analyze the data in the table can be drawn from table II program 1, 2, 3, 4, 5, 6, 7, and consistent in table I, the shape of the different procedures unchanged moments distance difference. Program Moment Invariant meet the software architecture of the uniqueness of fingerprints, namely: the software architecture of the different procedures fingerprint, and a larger difference can be drawn from tables I and II.

b) The software fingerprint of invariance

Since the software structure of the fingerprint extraction method is based on the software call graph, so the software structure of the fingerprint may have good invariance, and analysis of experimental results can be verified by the following experiment. In order to verify the invariance of the software structure fingerprint select the Tencent 1999 (OICQ era as subjects) to the 2009 calendar year QQ installer, the experimental results are shown in table III.

TABLE III. QQ DIFFERENT VERSIONS OF THE SAME SHAPE MOMENTS COMPARISON

prog	Φ_1	Φ_2	Φ_3	Φ_4	Φ_5	Φ_6	Φ_7
oicqb	3.00	1.04	8.41	5.64	3.77	-1.95	-9.63
oicqc	3.00	1.04	8.41	5.64	3.77	-1.95	-9.63
qqb	3.00	1.04	8.41	5.64	3.77	-1.95	-9.63
qqc	3.00	1.04	8.41	5.64	3.77	-1.95	-9.63
qqc	3.00	1.04	8.41	5.64	3.77	-1.95	-9.63
qq2004	6.08	1.05	2.26	2.17	4.88	3.65	-3.46
qq2005	6.08	1.05	2.26	2.17	4.88	3.65	-3.46
qq2006	6.08	1.05	2.26	2.17	4.88	3.65	-3.46
qq2007	6.08	6.00	2.41	2.28	5.36	1.26	-1.59
qq2008	6.08	6.00	2.41	2.28	5.36	1.26	-1.59
qq2009	6.08	5.11	5.94	5.80	3.41	1.06	-3.48

Analyze the data in the table can be drawn from table III, oicq99b2 oicq99c, qq2000b0425 qq2000c0630 and qq2000c1230b the software structure exactly the same fingerprint, the software oicq beginning in 2000 renamed qq, but the structure of the five software version is exactly the same fingerprint, indicating Although the software upgrade after the instruction-level changes such as inserting garbage instruction, the register rename code exchange, instruction substitution and change, will not change as long as the program's function call relationship does not change its software architecture fingerprint.

c) The software fingerprint of sensitivity

In order to verify the software structure of fingerprint is sensitive, by selecting a program sample, minor changes to the software, in the call graph, to take a random cut an edge, random add a edge and random change one side, then the structure of fingerprint software call graph by extracting after the change, between calculation and the original call graph structure of fingerprint distance, to verify its better sensitivity. The results are as shown in table IV.

TABLE IV. CHANGED SHAPE INVARIANT MOMENTS FROM THE TABLE

prog	reduce an edge	add an edge	change an edge
BridgeWAN	0.052	0.081	0.128
IE9	0.100	0.265	0.290
Thunder	0.343	0.041	0.165
075gangsir.cn	0.300	0.441	0.949
085gangsir.cn	0.712	0.362	0.794

From table IV, with good sensitivity, based on the software structure of the call graph fingerprint function call level changes in the relationship will lead to Changed in the software architecture fingerprint.

ACKNOWLEDGMENT

This research was partially supported by the National Basic Research Program of China (973 Program) under grant No. 2011CB302605, the National High Technology Research and Development Program of China (863 Program) under grant No. 2011AA010705 and No. 2012AA012502, the National Science Foundation of China (NSF) under grants No. 61100188 and No. 61173144, the National Key

Technology R&D Program of China under grant No. 2012BAH37B01.

REFERENCES

[1] Christina Cifuentes and K.John Grough.Decompilation of Binary Programs.Software Practice and Experience,25(7):811-829,July,1995.

[2] Cristina Cifuentes and Antoine Fraboulet.Intraprocedural Static of Binary Executables.In Proceedings of the International Conference on Software maintenance,Bari,Italy,October 1997,pages188-195,IEEE-CS Press.

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

[4] Ilfak Guilfanov. An Advanced Interantive Multiprocessor Disassembler.http://www.datarescue.com,2000.

[5] H.S.Kahn, “ Enhanced collection of fingerprints and ridge counting”,American J Human Biology,Wiley-Liss,Inc,USA,2005,17.

[6] Halvar Flake.Structural Comparison of Executable Objects.Detection of Intrusions and Malware&Vulnerability Assessment[C].Dortmund:GI SIG SIDAR Workshop.2004.161-173.

[7] Horst Bunke. Recent Developments in Graph Matching. IEEE Trans.SMC,2:117-124,2000.