Research of Code Detection Theory Based on Dynamic Taint Analysis

Caiyun Xie [1, a], Ruxia Hong [2,b *]

[1,2] Department of Information Science, Nanchang Teachers College, China

[a]xiecaiyun2005@163.com, [b]545798076@qq.com

Keywords: Dynamic Taint Analysis; Code Detection; Information Safety; Fuzzing

Abstract. Nowadays, dynamic taint analysis is a hot field of information security. There is important implications for improving world's information safety to study dynamic taint analysis techniques penetrate deeply. This paper introduces the concept of dynamic taint analysis techniques, principles and realizing the principle, then it describes the status of dynamic analysis techniques mainly from different aspects of information flow, fuzzy test, vulnerability detection and types. And it compares the advantages and disadvantages of each method and the improved method.

Introduction

With the rapid development of information industry, information technology is unprecedented prosperity, it is deep into every aspect of people's lives. But at the same time, incidents of harming information security occur constantly. It's a great threat to information security of the state, society and personal. In the information security incidents, which caused by malware(including viruses, Trojans, worms, Root kit, spy ware, etc.) take a large proportion. Malicious software poses a great danger on people's daily lives, economic security and social stability. Therefore, there is important theoretical and practical significance to conduct research on malware to seek new and more effective technology to detect, analyze and defense malicious software automatically.

Detection method of malicious code

A. Method based on "Checksum"

The basic idea of method based on "checksum" is: before running any software, calculate the current hash value firstly, and then compare the calculated hash value with the expected hash value of the software. If they match, it is assumed that the software is authentic, otherwise it is not credible. This method is relatively simple and easy to implement, but its scope of appliance is limited.

B.Method based on "Software dynamic behavior"

From the perspective of a credible analysis of the behavior of the dynamic behavior of the running software, it is one of the hot researches. However, from what point to characterize the dynamic behavior of the software, and how to characterize the dynamic behavior of the software, there is no unified conclusion. Different researchers from different perspectives put forward their findings.

System call is the most common object of software modeling of dynamic behavior. Back in 1996, Forrest et al[1] proposed: the abnormal behavior of software can be detected by segment monitoring system calls. Inspired by Forrest, the researchers launched a variety of research for the system call. In addition to the open system call, Feng et al [2] used the stack information for abnormal detection. Abadi et al [3] used the control flow graph (CFG) to ensure the real-time control Flow integrity (CFI) when software was running. Petroni et al [4] improved it as state-based control flow integrity (SBCFI) on the basis of the CFI furtherly. Linn et al [5] proposed: semantic information (system call address) can be embedded into the executable code, and used the semantic approach to identify malicious system call, which prevented remote code from injecting attacks, and so on.

In China, the literature [6-9] attempted to study the behavior of the credibility of the use of information flow. Literature [10-12] also studied measurement problems of software-based dynamic integrity based on the behavior.

Security analysis of software based on behavior is currently a hot research at home and abroad. However, due to the complexity of the form of malware, as well as the diversity of means of attack, this respect is to some extent still in the initial research stage, there are many issues needed to be studied furtherly[13].

C. Method based on "static analysis"

"Static analysis" is: analyze the software source code or binary code, obtain possible dynamic behavior when software is running, thus study safety of running software indirectly.

Livshits et al [14] used static analysis method to complete vulnerability analysis of Java application. Cova et al [10] proposed a method to discovery ELF binary file format vulnerabilities through static analysis. Wang Tielei et al [15] in Peking University carried out vulnerability analysis of integer overflow for x86 binary. Lu Xicheng et al [16] in National Defense Science also used symbolic execution and taint analysis for studying high assurance software test automation.

D. Dynamic taint analysis

Dynamic taint analysis (DTA) is an analysis method of dynamic information flow. This method tracks the data processing when programs are running and records spreading of data. DTA is to identify dependencies between the result of destination data and sources[17]. Currently, there are three main dynamic taint analysis implementations. The first method is insert piles through the source code, compile and run the program after insert piles, and track the process of spreading of information flow dynamically in the program run-time, to detect exploits. The second method is to insert piles based on binary files, and track the process of spreading of taint information dynamically in the program run-time to detect exploits. This method has higher cost then other types of stain analysis methods [18]. The third method is to track spreading of information flow at the hardware level, the method has advantages of high speed and small cost, but a narrow range of applications, not easy to expand [19]. This paper describes the analysis of several recent proposed principles and methods, and compares their advantages and disadvantages.

Dynamic taint analysis

A.Dynamic taint analysis based on information flow

Over the years, buffer overflow vulnerability is the most common one of security vulnerability. Buffer overflow vulnerability is very common, it is widespread in a variety of operating systems, application software. At the same time, the buffer overflow vulnerability is very dangerous, buffer overflow attacks can cause the program to fail, and cause the system to crash and restart and other consequences. Zhou Ling[20] put forward a new network attack detection technology: Dynamic taint analysis based on information flow. The technology is based on information flow, including data flow analysis and control flow analysis, dynamic execution monitoring and inspection tasks in runtime of binary program. This technology can effectively detect various network attacks based on vulnerabilities, including buffer overflow attacks, format string attacks. Dynamic taint analysis based on information flow is a combination of dynamic data flow and control flow analysis. The main principle is to make monitoring, labeling, tracking, testing activities program data from external blemishes in memory dynamically in runtime of program. Based on this principle, combined with various considerations, the system has the following main functions:

1) Achieve dynamic monitoring when program is running, and not affect the normal operation of the program.

2) Not require the source code, dynamic monitoring in binary code execution.

3) Mark, track, test stain data and other basic functions.

4) Control information flow and track dissemination.

System flow chart of system is shown in Figure 1:

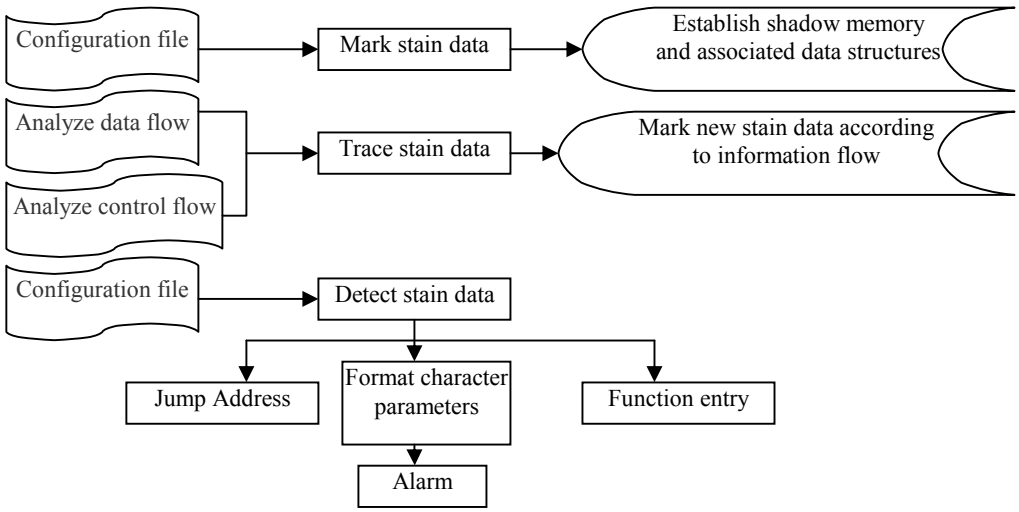

Figure 1 Flow chart of dynamic taint analysis technique based on information flow

Advantages: This method has the feature of real-time, it can complete the monitoring inspection tasks in dynamic execution when the client does not require customers to program source code.Another feature is low false alarm rate

Disadvantages: High cost, slow speed.

B. Dynamic taint analysis based on fuzzing

Fuzzing is an effective mining techniques of binary software vulnerabilities, but inefficiency of the test hampered its application in modern software testing. Currently, how to effectively improve the efficiency of fuzzing is a serious problem. Professor Miller in Wisconsin University proposed the idea of using random, unstructured data to test applications in the early 1990s, which was called fuzzing[21]. Currently, Dr. Wang Tielei[22] in Peking University has better idea, and developed a fuzzing framework named TaintScope. It introduces the main idea. As is shown in Figure2.

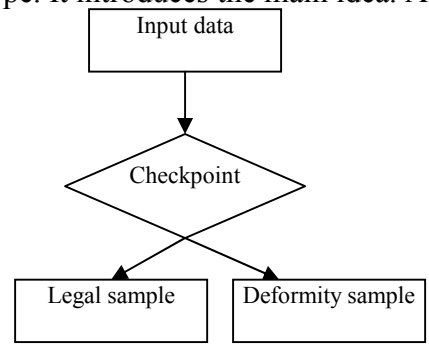

Figure 2 Checkpoint detection

TaintScope used dynamic taint analysis techniques to detect calibration point of binary software, used symbolic execution techniques to generate test cases to bypass the detection point. TaintScope was divided into four modules: dynamic taint tracking, testing and inspection check points, guiding fuzz and generate test cases to make the program crash.

This method has the following disadvantages: A failure to take full advantage of the platform provided by the previous frame. It can't improve test efficiency of known binary format specification software. And code coverage is low.

Yan Xiaodong in University of Electronic Science and Technology made improvement of TaintScope [23]. Design and implement a byte-level granularity of dynamic taint analyzer (TaintTracer), use a dynamic tag management strategies and different assembly instructions for a detailed tracking strategy stain, make collected keyword section more scientific and accurate. Built on fuzzing framework Peach, so that not only can take the mutation strategy targeted information for different data types, know part of the format specification information, but also can bypass software verification and testing. But it also has disadvantages: first, it didn't solve the problem of code coverage and can't handle the control dependencies between data leading to inaccurate tracking

blemish. Second, a large amount of repeat information collected and a waste of time. Third, TaintTracer also failed to apply in binary network application software, it needs additional work.

C. Dynamic taint analysis based on attack detection technology

Lu Kaikui[24] used excellent instrumentation platform Pin to improve system efficiency, at the meanwhile, the control flow is proposed based on CFG propagation algorithm to solve the problem of taint analysis to detect blind spots. And he get more precise definition of the corresponding stain to detect security policy based on the principle of exploits. This prototype system not only can detect buffer overflow attacks, but also can travel the entire directory. SQL injection and format string attacks were detected, greatly improved the detection range of the attack. Flow chart of the system is shown in Figure 3.

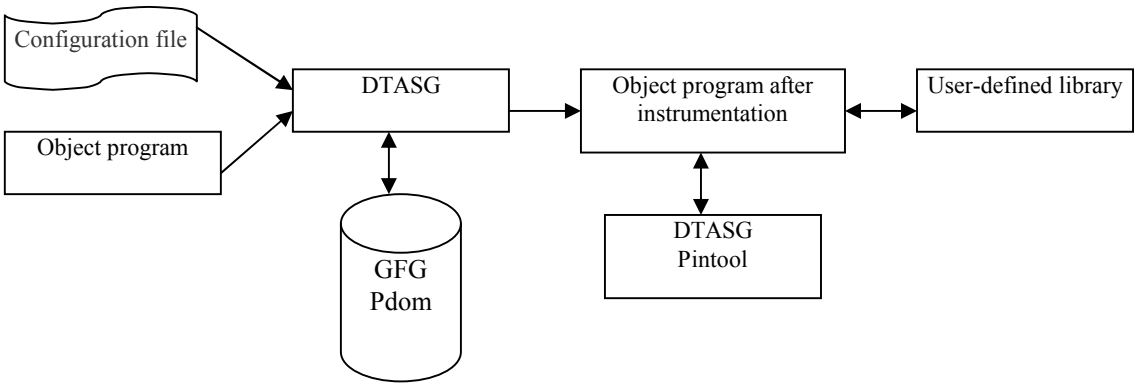

Figure 3 System frame diagram

System workflow is as follows:

(1) Set the stain analysis of relevant policies through an XML configuration file format: stain marks strategy, communication strategy and stain detection strategy.

(2) By dynamic binary instrumentation framework Pin start the target program, initialize DTASG Pintool.

(3) Heuristic analysis target program branch instruction, and generate CFG and control dependent (pdom) information data for the target program.

(4) After the target stub program by visiting DTASG Pintool (mainly provide stain marks and dissemination capabilities) and user-defined library files (mainly provide detection and stain tag mapping function) to complete the tag information flow of stain, dissemination and exploits testing.

Performance evaluation: DTASG detects all the 18 types of buffer overflow attacks, because overwriting the return address, function pointers, the base address or other forms of attack are all required for the transfer of data by some blemishes program control. While principle of DTASG is just to detect illegal use of tainted data, so DTASG is very effective in detecting these types of buffer overflow attacks.

Inadequate: The system is not ideal in terms of performance; it is lack of memory optimization.

D. Other types of dynamic taint analysis techniques

Liu Yu[25] et al proposed one method based on inverse dynamic taint analysis by constructing extended taint propagation graph (ETPG) of malicious process instruction and function behavior. This method analyzed a variety of protocols for different processing elements to divide the grammatical structure of the protocol data; and derived semantic information according to processing implied semantics of API functions of the relevant fields. Compared with references[4, 6, 7], it not only can get precise syntax division, and also provide semantics of each field in order to serve a variety of network security applications and support communication protocol specification aspects. Shi Dawei[26] et al proposed a dynamic taint analysis method based on particle sizes of combining binary code. This method focused on designing a new framework of first coarse-grained then fine-grained analysis. Dai Wei[27] proposed several techniques used to enhance the accuracy and efficiency of dynamic taint analysis, including more accurate instruction-level taint propagation, and function-level communication strategy stain. Zhang Bin[28] proposed fuzzing techniques oriented by

binary program based on dynamic taint analysis. This method can generate a valid test for complex module functions, improved the efficiency of fuzzing.

Conclusion

This paper lists kinds of methods about automatic detection of malicious software, analysis and defense according to the current security issues in the field of network information security deteriorating, among which the most prominent is dynamic taint analysis techniques. Because it captures the essence of software security vulnerabilities, whether the security vulnerabilities are disclosed, the detection techniques are effective in principle. This paper mainly from the concept of dynamic taint analysis techniques, principles and implementation to study, then describes the domestic situation of dynamic analysis techniques. And it lists methods of information flow, fuzzy test, vulnerability detection et al, and compares advantages and disadvantages of each approach. Now, dynamic analysis technologies are relatively mature, the general direction of the next step should be: first, study the precise positioning and back propagation paths of stain information, improve the effectiveness of large-scale analysis of the program; second, reduce overhead, optimize memory, improve the detection speed.

Acknowledgments

The project is supported by Foundation of Jiangxi Provincial Education Department (No. GJJ14791).

References

[1] S. Forrest, S. A. hofmeyr, A. Somayaji, et al. A Sense of Self for UNIX Processes. In Proceedings of the 1996 IEEE Symposium on Security and Privacy (Oakland'96), 1996: 120-128.
[2] H. H. Feng, O. M. Kolesnikov, P. Fogla, et al. Anomaly Detection Using Call Stack Information. In Proceedings of the 2003 IEEE Symposium on Security and Privacy (Oakland'03), 2003: 62-75.
[3] M. Abadi, M. Budiu, U. Erlingsson, et al. Control-flow Integrity. In Proceedings of the 2005 ACM Conference on Computer and Communications Security (CCS'05), 2005: 340-353.
[4] N. L. Petroni, M. Hicks. Automated detection of persistent kernel control-flow attacks. In Proceedings of the 2007 ACM Conference on Computer and Communications Security (CCS'07), 2007: 103-115.
[5] C. M. Linn, M. Rajagopalan, S. Baker, et al. Protecting against Unexpected System Calls. In Proceedings of the 2005 Conference on USENIX Security Symposium (USENIX Security'05), 2005.
[6] Zhao Jia, Shen Changxiang, Liu Jiqiang, et al. A Noninterference-Based Trusted Chain Model. Journal of Computer and Development, 2008, 45(6): 974-980.
[7] Zhang Xing, Chen Youlei, Shen Changxiang. Noninterference Trusted Model Based on Process. Journal on Communications, 2009, 30(3): 6-11.
[8] Zhang Xing, Huang Qiang, Shen Changxiang. A Formal Method Based on Noninterference for Analyzing Trust Chain of Trusted Computer Platform. Chinese Journal of Computer, 2010, 33(1): 74-81.
[9] Li Xiaoyong, Han Zhen, Shen Changxiang. Transitive Trust and Performance Analysis in Windows Environment, Journal of Computer Research and Development, 2007, 44(11): 1889-1895.
[10] G. J. Peng, X. C. Pan, J. M. Fu, et al. Static Extracting Method of Software Intended Behavior based on API Functions Invoking. Journal of Wuhan University (Science Edition), 2008, 13: 615-620.
[11] G. J. Peng, X. C. Pan, H. G. Zhang, et al. Dynamic Trustiness Authentication Framework based on Software's Behavior Integrity. Proceedings of the 2008 International Conference for Young Computer Scientists, 2008: 2283-2288.
[12] Peng Guojun. Research on theory and technology of software dynamic trusted behavior based on integrity. Wuhan: Wuhan University, 2008.

[13] Shen Changxiang, Zhang Huanguo, Wang Huaimin et al. Research and Development of the Trusted Computing, China Science: Information Science, 2010, 40(2):139-166.
[14]V. B. Livshits, M. S. Lam. Finding Security Vulnerabilities in Java Applications with Static Analysis. In Proceedings of the 2005 Conference on USENIX Security Symposium (USENIX Security'05), 2005.
[15]T. L. Wang, T. Wei, Z. Q. Lin, et al. IntScope：Automatically Detecting Integer Overflow Vulnerability in X86 Binary Using Symbolic Execution. In Proceedings of the 2009 Annual Network & Distributed System Security Symposium (NDSS'09), 2009.
[16] Lu Xicheng, Li Gen, Lu Kai et al. High-Trusted-Software-Oriented Automatic Testing for Integer Overflow Bugs, Journal of Software, 2009, 21(2): 179-193.
[17] D.E.Denning: A lattice model of secure information flow. Communications of the ACM, 19(5): 236-243, 1976
[18] John C.Dolak. The Code Red worm. http://rr.sans.org/malicious/code_red8.php, 2007
[19] Lu Kaikui. Research and Realization of detection technology based on vulnerability dynamic Taint Analysis. University of Electronic Science and Technology, 2012.
[20] Zhou Ling. Research of dynamic Taint Analysis technology based on information flow. University of Electronic Science and Technology, 2010.
[21] Chen Yanling, Wang Zheng. Advancement of the Study on Fuzzy Testing. Computer Applications and Software. 2011,28(7): 291-293.
[22] Wang Tielei, Wei Tao, Zou Wei. RoBDD- Based Fine-Grained Dynamic Taint Analysis. Journal of Peking University. 2011, 47(6): 1003-1008.
[23] Yan Xiaodong. Research on Fuzzing Based on Dynamic Taint Analysis Technology. Xidian University. 2013
[24] Lu Kaikui. Research and Realization of Vulnerability Detection Based onDynamic Taint Analysis. University of Electronic Science and Technology. 2012.
[25] Liu Yu, Wang Minghua, Su Purui et al. Communication Protocol Reverse Engineering of Malware Using Dynamic Taint Analysis. Chinese Journal of Electronics, 2012, 40(4): 661-668
[26] Shi Dawei, Yuan Tianwei. A Dynamic Taint Analysis Method Combined with Coarse-grained and Fine-grained. Computer Engineering. 2014, 40(3): 12-17.
[27] Dai Wei, Liu Zhi, Liu Yihe. Binary Code-based Dynamic Taint Analysis. Application Research of Computers. 2014, 31(8): 2497-2501.
[28] Zhang Bin, Li Mengjun, Wu Bo et al. Method of Binary Oriented Fuzzy Testing Based on Dynamic Taint Analysis. Modern Electronics Technique. 2014, 37(19): 89-94.