A Method of Tainted Data Detection Based on Static Code Analysis

Yifei Xiao[1, a*], Dahai Jin[1, b] and Dalin Zhang[3, c]

[1] Institute of Network Technology, Beijing University of Posts and Telecommunications, Beijing, 100876, China;

[2] Beijing Jiaotong University, Beijing, 100044, China.

[a]xiaoyifeibupt@bupt.edu.cn, [b]jindh@bupt.edu.cn, [c]dalin@bjtu.edu.cn.

Corresponding Author: Yifei Xiao

Keywords: security, static analysis, tainted data.

Abstract. In the modern society, with the high-speed development of computer science and technology, the wave of the Internet economy is around the world, the use of computer software in every corner of our lives, various applications emerge in endlessly, people pay more and more attention on software security. Tainted data which comes from the external input variables and has been used by some function without detection of legitimacy is a kind of code security defect. In this paper, the author provides a detailed analysis and classification on the cause of the security defect, and introduces a method of tainted data detection based on static code analysis. The detection method preprocesses the code firstly to create abstract syntax tree, symbol table, control flow graph and function call graph. To analysis the relationship between the function calls, the author uses the function summary instead of expansion of the functions. In the last, by using this method to detect some open source projects, the experiment shows that this method has both lower positive rate and negative rate.

1. Introduction

At present, to build a reliable software in the process of software development, we must find the defect as early as possible, we should try to be, early detection, early treatment [1]. The first principle of software testing is to dig holes earlier, the better way can be involved in reliability testing in the requirements analysis phase. Static code analysis is the most convenient, the most direct method, defects can be found in the early stages of software development, and in the various stages of software development static testing can be used to minimize the defects of each phase [2].

Through the analysis of the software's source code, to summarize, about the system security, resource injection, tainted data is the first to be affected. As long as a software program to obtain a data from outside, there is a potential danger which may contain deceptive or no need data, to threaten the integrity of the program. As long as the program to access external data, this data will be in accordance with the design specifications of legitimacy, if the data is illegal, the data is tainted [3]. The program does not as planned by using the tainted data, such as confidential documents leaked to hackers. If the data comes from the functions offered by C\C++ lib or system calls, such as “getenv()”, “ gethostbyname()”, can be used by hackers, it will create tainted data.

This paper according to the principle of resource injection process and produce, provides a detailed analysis and classification on the cause of the security defect, and introduces a method of tainted data detection based on static code analysis. The detection method preprocesses the code firstly to create abstract syntax tree, symbol table, control flow graph and function call graph. To analysis the relationship between the function calls, the author uses the function summary instead of expansion of the functions. In the last, by using this method to detect some open source projects, the experiment shows that this method has both lower positive rate and negative rate.

2. Classification of tainted data

The security vulnerability model of tainted data is mainly that some input variables without valid detection are used directly as a function of the parameters. If these functions, involves the system function of the high level of security, or involve the operating system resources, so these malicious input, it may cause system information or system resource leaks [5].

2.1 Buffer overflow

If the amount of data which we want to write to a buffer is more than buffer capacity, the buffer overflow errors will occur. Additional data will be stored into the memory unit of adjacent, it will damage the system previously stored in the data. It is the most common problems in software security problems.

When the application attacked by buffer overflow, these extra data may contain malicious code, it could damage the user files, data, steal the user's privacy. The reason that we can use the buffer overflow attack is that C/C + + compiler doesn't inspect the scope automatically, and many programmers also do not pay enough attention to the problem.

2.2 Format string

The attacker will pass a control function that can format string to create a tainted data.

- The lack of control format string
- Format string vulnerability

2.3 Process control

Executing the command in the untrusted environment or loading the library files from untrusted source may cause application to perform malicious command.

2.4 Resource injection

Allows the user to enter and control resource identification may make the attacker has access and modify the protected system resources.

2.5 Not validate input for the system level functions

- Call an unsafe external application
- System resources are malicious modification
- Load incorrect library function

2.6 Others

- Denial of service
- Lure security risks
- Use unsafe shell command
- Fake log

3. A method of tainted data detection

3.1 Overview

This method is based on static code analysis, which is the process of detecting errors and defects in software's source code. The basic work flow is preprocessing the source code to create intermediate file, than creating an abstract syntax tree by using JavaCC, which is a parser generator is a tool that reads a grammar specification and converts it to a Java program that can recognize matches to the grammar.

Next, we can create symbol tree, control flow graph, function call graph, define-use chain and use-define chain. If the unreliable input is passed to other function or getting some data from the value of function return, we need not to expend the function [6], we can use the function summary instead. In other function calls to the function of each point, according to its front constraint information for judgment.

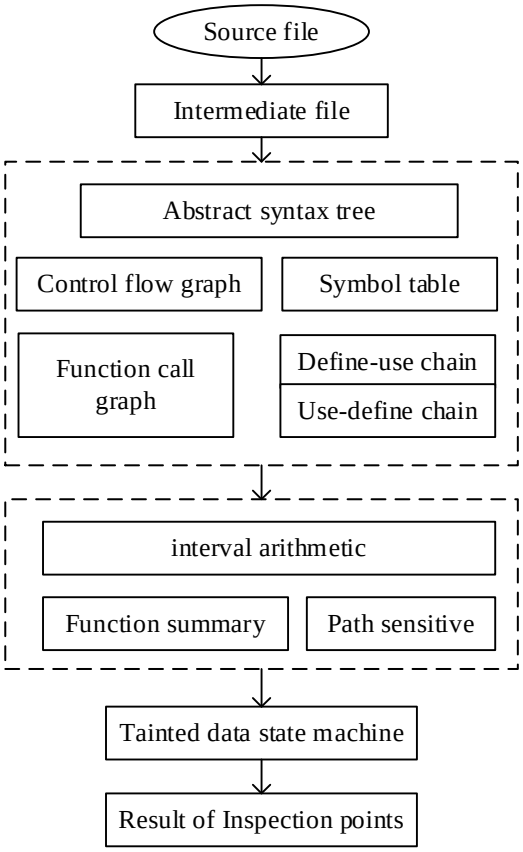

Fig. 1 Work flow of this method

3.2 Work flow

This method includes pretreatment and analysis phase. Preprocessor directive is generally used to make the source code to be convenient to modify or compile in different execution environment. And according to the pretreatment of intermediate files do preliminary analysis; Get engineering global relations of function calls, and engineering of the source file.

The abstract syntax tree data structure is a tree structure, support the basic function of a tree structure. Such as getting the parent node, acquiring a child node, accumulating child nodes and so on. Through these basic operations can find step by step in the syntax tree. Another better way is through the Xpath search [7].

The symbol table [8] are used to record the scope of markers or names and the statement using information, it will be mapping each identifier and its position statement. Each record in the symbol table represents an identifier.

For most application analysis and structure test control flow graph is indispensable basic data structure. The control flow graph is a kind of response directed graph of the program logic control process.

3.3 Function summary

Function summary is an abstract of the actual meaning of function, because in the process of the actual analysis, we don't care about the function of every detail [9], so we abstracts some characteristics of the function, when we analysis this function point, there is no need for the call will function expansion, not only improves the accuracy of cross function analysis, but also improves the efficiency, reduce the memory overhead.

- Pre-condition refers to before this function is called the context needed to satisfy the constraint conditions of the function
- Post-conditions refers to after the function call to the effects of context variables, etc.
- Function side effects associated with the defect mode of concrete, such as the use of external incoming data sources in a particular function, etc.

Function summary is related to the defect mode, we must write the corresponding visitors according to the type of the fault. The basic rule of the function summary is to determine the specific type of defect analysis, based on the analysis of defect action decided to register file generated which function in this paper, and then to monitor each function in the process of analysis of interval arithmetic analysis of the content to generate specific function. The function summary may include these parts, the function name, and the library which includes the function, the signature of the function, the type and value.

3.4 Detect the tainted data

According to the workflow above, the procedure to detect the tainted data is like this in Fig.2

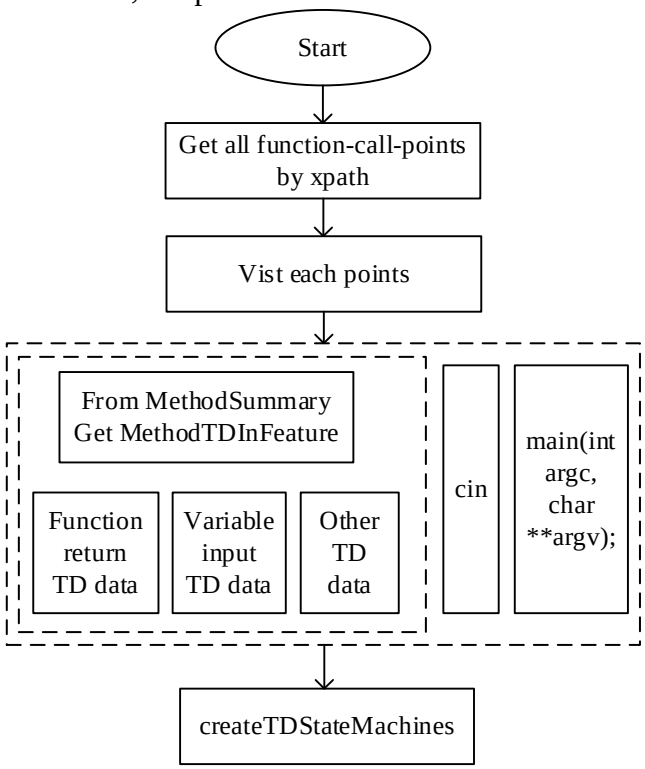

Fig. 2 Procedure to detect the tainted data

When we detect this security defect, we focus on external input parameters in C/C++ language, after defected this input parameters, if we don't verify the legitimacy, we think it is a fault. Due to data may be infected by copying operations such as transmission in the process, is the spread of the variables also need to test before use, so we by maintaining an infected data set to record the process of data, this collection combines data flow operation is needed to record all the data flow information.

For static analysis based on the pattern of defects, the core of its ultimate job is to define and detect defect modes. System can deal with defect mode varied the stronger the ability to find potential defects. The state machine is an abstract kind of common and easy to understand the program semantics. In this method, the author extends the state machine to unite the states description model. Fault describe state machine includes a series of states and conditions.

Defect model state machine is a kind of finite state automaton, including the set of state D, state transfer collection and transfer condition set Conditions. Among them, D={$Start，$End}U Dother; D×Conditions D ; $Start, Send are states in every fault state machine, these two kinds states are unique, respectively the starting state, end state. Dother is a collection of state, in the collection in addition to the start and end state of his condition, and the state machine set can be null set.

Through the above process has been created after the state machine, you can call the status detection automaton method to detect fault occurs, the state machine of the state transition diagram as shown in Table1 and Fig.3:

Table 1 State transition and conditions

Numble	State transition	conditions
0	Start→Inputed	Nodes in the control flow graph is an external data input operation
1	Inputed →Error	The external infection in the collection of data is used in the system function
2	Inputed →End	The external data for security verification
3	Error→End	No condition

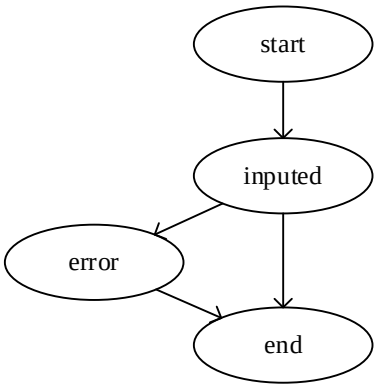

Fig. 3 State transition

4. Experiment and result analysis

In this section, we introduce the experimental environment and result. The test tool DTS (Defect Testing System) is a static code analysis software developed by our lab, the tainted data model in this system used the method in this paper.

4.1 Experimental Environment

Hardware environment:

Processor: Intel(R) Xeon(R) CPU E5620 @2.4GHz.

Memory: 4.00GB.

Software environment:

Operating system: Microsoft Windows Server 2003 Enterprise Edition Service pack 2.

JDK: 1.7.0_02.

DTS: 7.0

4.2 Result and analysis

The experimental result shows in table 2, including the size of the project, the analysis time, inspection point, defect and false alarm.

Table 2 Experimental result

Numble	Project name	Code lines	Analysis time/s	inspection point /defect/false alarm
1	avisynth	51853	174	3/30
2	BWChess	6738	20	4/3/1
3	dgvideo.	23678	70	3/3/0

The efficiency of the method in this paper is very high, it takes almost 30s to detect more than 10,000 code lines project. There are also false alarm in the result, analysis of the causes are as follows

There is incomplete context information in function summary, in the process of the computing function, this paper introduces the detection algorithm does not consider the path information.

5. Conclusion and future work

In this paper, the author provides a detailed analysis and classification on the cause of the security defect, and introduces a method of tainted data detection based on static code analysis. At last, the author uses this method detect five open source project. The experimental results show that this algorithm has a good practicability, high detection efficiency, the rate of false positives, non-response rates low.

In the next step, the author will add context information into function summary to improve the accuracy of the result.

Supported by the National Natural Science Foundation of China (Grant No. 61502029)

References

[1]. Yunzhan Gong, Ruilian Wei, “Software Testing”, Beijing: China Machine Press, 2008.

[2]. Dandan Zhou, Xianguo Li, “Research on software defect analysis model based on static analysis tool”, Computer and modernization, vol 11, pp. 55-58, 2012.

[3]. Li Liu, “Research and application of document based on static analysis”, MS thesis. Beijing University Of Posts And Telecommunications, 2012

[4]. Yunshan Zhao, “Static defect analysis based on symbol analysis”, MS thesis, Beijing University of Posts and Telecommunications, 2012.

[5]. Brian W. Kernighan and Dennis M. Ritchie. “The C Programming Language”, Prentice-Hall, Englewood Cli_s, New Jersey, 2nd edition, 1988.

[6]. Louridas P. Static code analysis [J]. Software, IEEE, 2006, 23(4): 58-61.

[7]. Chandra S, Dhoolia P, Gowri III M, et al. Static code analysis: U.S. Patent 8,806,441[P]. 2014-8-12.

[8]. Moser R, Pedrycz W, Succi G. A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction[C]//Software Engineering, 2008. ICSE'08. ACM/IEEE 30th International Conference on. IEEE, 2008: 181-190.

[9]. German A. Software static code analysis lessons learned [J]. Crosstalk, 2003, 16(11).