2015 International Conference on Industrial Informatics-Computing Technology, Intelligent Technology, Industrial Information Integration

An Overview of Source Code Audit

Xiang Lingzi， Lin Zhi
National Engineering Research Center of Information Security
Beijing, China

***Abstract*—Software vulnerability reports and reports of software exploitations continue to grow at an alarming rate in recent years. Many security issues are appeared in codes. The source code audit can improve the source code quality and avoid potential vulnerabilities in application system. This paper firstly expounded the principles of code audit and the purpose of Code audit is to make sure developers strictly follow the security technology, also briefly introducing the CERT secure coding standards which provide a detailed enumeration of coding errors that have caused vulnerabilities. Next, summarized the audit methods and techniques and compared the analysis tools for source code audit, then, show the value and significance of code audit. Finally, the development trend of audit technology is estimated**

***Keywords-** source code review; static analysis; information security*

I. INTRODUCTION *(HEADING 1)*

Software vulnerability reports and reports of software exploitations continue to grow at an alarming rate in recent years, and a significant number of these reports result in technical security alerts. To address this growing threat to the government, corporations, educational institutions, and individuals, systems must be developed that are free of software vulnerabilities. Coding errors cause the majority of software vulnerabilities. For example, 64% of the nearly 2,500 vulnerabilities in the National Vulnerability Database in 2004 were caused by programming errors [1].

However, with the development of the information system, the functions of application increase accordingly, besides, the size of the program source code is also increased and the back door of source codes is easy to be used to cause security vulnerabilities. It is very difficult to detect security vulnerabilities through the traditional testing methods. So, testing and analysis the source code by source code review can ensure the safety of the software and information systems fundamentally.

This paper firstly expounds the principles of the source code auditing, secondly, summarizes the methods and techniques of source code audit, and compare the code review tools, thirdly, show the value and significance of the source code audit. Finally, make forecasts the development trend of audit technology for source code.

II. THE PRINCIPLES OF SOURCE CODE AUDIT

Process of code audit is as follows, firstly read the source codes of the target system. Secondly, checking the control that assures the safety of source codes whether set in the key position on the logic flow. The purpose of Code audit is to make sure developers strictly follow the security technology. Generally, if the audit target system after a complete source code review, it can avoid most of the loopholes in application and improve the precision of simulation of penetration test to system.

A. Do not trust all input

There are a lot of inputs in the program of source code, including the system internal and external system. Applications will process and analysis these inputs, then display. Each input can be malicious used and cause the code vulnerabilities. According to statistics, the security problem caused by input data in holes is for 90%, so the format and contents of input date must be strictly limited. It is also necessary to take the security control to input.

B. Follow security coding standards

The CERT Program, part of Carnegie Mellon University's Software Engineering Institute, takes a comprehensive approach to identifying and eliminating software vulnerabilities and other flaws [2]. CERT has compiled many security coding rules for many computer languages.

The CERT Oracle Coding Standard for Java (published in 2011) consists of rules and recommendations, collectively referred to as guidelines. Rules are meant to provide normative requirements for code, whereas recommendations are meant to provide guidance that, when followed, should improve the safety, reliability, and security of software systems [3].

The CERT C Secure Coding Standard (published in 2005) provides rules and recommendations (collectively called guidelines) for secure coding in the C programming language [3].. The goal of these rules and recommendations is to develop safe, reliable, and secure systems, for example by eliminating undefined behaviors that can lead to undefined program behaviors and exploitable vulnerabilities. Conformance to the coding rules defined in this standard are necessary (but not sufficient) to ensure the safety, reliability, and security of software systems developed in the C programming language.

The CERT C++ Secure Coding Standard (published in 2008) provides rules and recommendations for secure coding in the C++ programming language. The goal of these rules and recommendations is to eliminate insecure coding practices and undefined behaviors that can lead to exploitable vulnerabilities [4]. The application of the secure coding standard will lead to higher-quality systems that are robust and more resistant to attack.

978-1-4673-8312-7/15 $31.00 © 2015 IEEE
DOI 10.1109/ICIICII.2015.94

Secure coding standards provide a detailed enumeration of coding errors that have caused vulnerabilities, along with their mitigations for the most commonly used software development languages.

C. The relationship between the CWE and CERT

The Common Weakness Enumeration (CWE) is a unified, measurable set of software weaknesses that enables the effective discussion, description, selection, and use of software security tools and services that can find these weaknesses in source code and operational systems [5]. The CWE also enables better understanding and management of software weaknesses related to architecture and design. It enumerates design and architectural weaknesses, as well as low-level coding and design errors.

CERT is developing secure coding standards for commonly used programming languages such as C, C++, and Java through a broad-based community effort that includes members of the software development and software security communities [6].

The CWE and the CERT secure coding standards perform separate but mutually supportive roles. Simply stated, the CWE provides a comprehensive repository of know weaknesses, while CERT secure coding standards identify insecure coding constructs that, if present in code, could expose a weakness or vulnerability in the software. Not all weaknesses enumerated in the CWE are present in any particular secure coding standard, because not all weaknesses are present in each language and because CWE also includes high-level design issues [7]. Not all CERT secure coding guidelines are mapped directly to weaknesses in the CWE, because some coding errors can manifest themselves in various ways that do not directly correlate to any given weakness. Both tools are necessary in evaluating the security and safety of software systems. Guidelines in the CERT C Secure Coding Standard are cross-referenced with CWE entries. These cross-references are only created for guidelines which, if violated, directly contribute to the referenced weakness. Similar mappings will be created for other CERT coding standards once they are completed [8].

III. The Overview of Methods and Technology for Source Code Auditing

A. Methods for source code auditing

There are several commonly used methods in the processing of code audit:

1) analysis from top to bottom on logic

The code audit exists in the life cycle of applications, when application receives the user's request, the code audit start from top to bottom on logic, then user get the final results.

Top-down audit for source code tracks all the external input variables including the input of environment and user. All possible variables which may be maliciously control by user or the function which is easy to damage the internal variables must be strictly followed. Once the parameters have been accepted, which will be traversed tracking follow the code logic, until the threat security codes are found or all the inputs filtered and limited to safety [9].

2) analysis from bottom to top on logic

This method according to the keywords dictionary of sensitive function gets parameters from Application point back and follows these parameters step by step until find potential safety problems [10]. This method requires auditors have a good understanding for the internal mechanism and use method of the sensitivity functions. So as to determine some illegal input parameters are safe.

3) the coverage of logical path

This is a kind of Traverse test way based on the source code logic which can find the indiscoverable vulnerabilities than traditional test way. It starts from the life cycle and source code logic, this method uses tools to traverse the code logic on all possible paths and find the potential vulnerabilities on the path [11]. Although this method can help developers detected the dangerous loopholes in coding phase, the human costs put into covering logical path of source codes are very high and auditing process costs long time with low output.

B. Technology for source code auditing

Since 1976, source code security research has being studied. The famous article Data Flow Analysis in Software Reliability (published in ACM Computing Surveys) wrote by Lloyd D. Fosdick and Leon J.Osterweil mentioned that the commonly used technology for code auditing, such as boundary detection, data classification test, the state machine system, authentication, control flow, data flow and so on[12]..

At present, the analysis methods in software source code can be included in two classes, namely, static analysis and dynamic analysis. Researchers at home and abroad have done a lot of research in this area and put forward some different implements and designs for source code analyses.

Ye Liang presented an analysis method for source code based on safety coding rules in 2013[13]. Yuan Bing audited the source codes aiming at the malicious backdoor of the applications and achieved very good results in 2013[14]. Zhou bin put forward an test source code methods for detecting the vulnerabilities of source codes in 2015[15].

Although dynamic methods have no limit to the size of the source code, can even inspect the large scale codes, there is a deficiency that the effect of testing depends on the input methods heavily. The vulnerabilities will be found until the particular input caused performing codes reach dangerous point. The dynamic methods have high mis-reported rates.

Now, the mainstream technologies are the static analysis for source code based on safety rules, including lexical analysis, semantic analysis based on abstract syntax tree, checking rules analysis, etc. it can be done by manual or automatically tools.

There are several main technologies of static analysis of source codes:

1) Analysis of data flow and control flow

The analysis of data flow and control flow aims at logic paths in source codes and sequences of operations on these paths. It collects semantic information from the program source code under the static environment (don't run the program) and tests where are any safety problems in operation and in the definition of a variable.

This method, aiming at detecting assignment, reference and memory errors, has some defects, such as excessive information, easy to cause the combination explosion. Therefore, it often used as an auxiliary method of other detection methods.

2) Analysis of lexical grammar

Besides analyzing the basic grammar, this method also matches the database of security vulnerabilities, such as CWE, CVE, efficiently detecting dangerous function and the number and position of vulnerabilities in source codes.

Due to lacking the semantic information of source codes and the quantity of vulnerabilities in the database being limited; this detection can't match undisclosed vulnerabilities, which can cause high rate of false alarm and reduce the accuracy of detection. So this method always is used in the early period of the audit work.

3) Semantic analysis based on the abstract syntax tree

As the abstract syntax tree in source code audit is built through scanning code program and extract the main information in source code after removing redundant syntax structure, Analyzing semantic information of source code through the abstract syntax tree which grasps the global information or module information of source code can efficiently detect hidden vulnerabilities. But there are many 0day vulnerabilities and complicated vulnerabilities, the capability and accuracy of detection of this method needs to be improved.

4) Analysis of rule checking

First of all, Rule checking uses specific grammar to describe the rules of vulnerabilities, and transfer analysis results of the rules by parsers into internal expression identified by system, when matching the code rules to complete the vulnerabilities detection.

This method is perfect in the process; it is easy to detect the codes which are inconsistent with security. Because the characteristics of vulnerabilities is just one of the observed secure coding rules , achieving good detection effect only in the known vulnerabilities library, for the 0day vulnerabilities , effect is poor.

IV. THE TOOLS FOR SOURCE CODE AUDITING

The security problems should be solved from the primitive point-the source code. Analyzing source code security depends on the automation code auditing tools with combined artificial analysis. The tools first scans the vulnerabilities in codes, next, complete the analysis, rectification and confirmation of vulnerabilities by human.

With the development of computer language, the analysis technologies of source code are being perfected. There are lots of analysis tools for source code appearing in information security field. Such as Klocwork Insight, Coverity, Parasoft, Rational Software Analyzer, Fortify SCA, Checkmarx Suit, Code Secure and so on.

Briefly introducing common used analysis tools for source codes as follow:

A. Fortify SCA

The Fortify Source Code Analyzer Engine (SCA) renders a variety of programming languages (Java, C, and C++, for example) into an intermediate form that is then processed with a set of algorithms used to identify and flag dangerous coding constructs. The Fortify Source Code Analyzer (SCA) consists of five analysis engines. The data flow analyzer, the control flow analyzer, the semantic analyzer, the structural analyzer, and the configuration analyzer combine to identify vulnerabilities in source code. Each of these analysis engines uses a different approach to detect vulnerabilities.

TABLE I. TABLE TYPE STYLES

Summary of Rules Implemented for Fortify SCA			
	C	***C++***	***Total***
Total number of secure coding standard rules	***180***	99	279
Rules that fortify SCA implements by default	***16***	6	22
Rules implement for Fortify SCA	***27***	17	44
Rules partially implemented for Fortify SCA	***14***	5	19
Total number of rules at least partially implemented	***57***	28	85

B. FindBugs

FindBugs is an open source program created by Bill Pugh and David Hovemeyer which looks for bugs in Java. It uses static analysis to identify hundreds of different potential types of errors in Java programs. Potential errors are classified in four ranks: (i) scariest, (ii) scary, (iii) troubling and (iv) of concern. This is a hint to the developer about their possible impact or severity. FindBugs operates on Java bytecode, rather than source code.

C. PMD

PMD is a source code analyzer. It finds common programming flaws like unused variables, empty catch blocks, unnecessary object creation, and so forth. It supports Java, JavaScript, XML, XSL.

D. CheckStyle

Checkstyle is a static code analysis tool used in software development for checking if Java source code complies with coding rules. It defines a set of available modules, each of which provides rules checking with a configurable level of strictness (mandatory, optional...). Each rule can raise notifications, warnings, and errors. It also helps developers to comply with good programming practices which improve the code quality, readability, re-usability, and reduce the cost of development.

V. THE VALUE AND SIGNIFICANT OF THE SOURCE CODE AUDITING

Source code audit work is very important, on the one hand, it can eliminate safety problems form the source of application and control the risk of system security on the root, reducing the costing at the later period of life cycle of applications for maintenance. On the other hands, it can solve the hidden troubles in source codes , besides, strong the protection of whole security system from the core level.

The significance of source code auditing is to improve the quality of source code of software, and avoid dangers from the potential back door of application system, prevent the important date in information system from being leaked and improve the security of the system architecture at the same time. Source code audit lets enterprise avoid passive defensive position, and obviously save the fund investment of enterprise on security, significantly improve the efficiency safety management.

VI. THE TECHNOLOGY DEVELOPMENT TREND OF SOURCE CODE REVIEW

Due to the situation of network security becoming more and more complicated and the increase of security problems with malicious attacking change with each passing day, so as source code audit. In the recent year, the most significant trend of security technology field is the integration of source code auditing tools and manual service.

According to the Common Vulnerabilities & Exposures (CVE), Open Web Application Security Project (OWASP) 2015 and other Vulnerability Library published by equipment and software vendors, using the professional source code scanning tools to audit source code wrote by varieties programming language, the integration combine security coding standards advisory with assessment of code security situation, position security vulnerabilities that exist in the code and analyze the risks and recommend the modify suggestion.

ACKNOWLEDGEMENT

This paper is support by National High Technology Research and Development Program of China (863 Program, No. 2013AA01A214).

REFERENCES

[1] Xiangyan LU School of Computer and Electronics and Information, Guangxi University, Nanning 530004, China. Security Code Review of Web Application[A].

[2] Qin Xiaojun, Gan Shuitao,Chen Zuoning. A security flaws static testing technology software based on first-order logic code [J]. Chinese science, information science, 2014,01:108-129.

[3] li Jun. White box code security audit method analyses [J]. Journal of mudanjiang university, 2014, 10: 142-145.

[4] Zhou Kuan long, etc. Code security flaw detection method based on control flow section [J]. Computer engineering and design, 2012, 06:2265-2271+2304.

[5] Wang lei, etc. the detection methods of code security vulnerabilities based on constraint analysis and model checking. 2011,09:1659-1666.

[6] Liang Yeyu Xu Tan, build a better, Yang Ming. Code audit work play an important role in the whole security system [J]. Computer security, 2013, 12:32-35.

[7] Jianmin Bao,Shu Zhang,Junjie Zhang,Kun Wang,Haifeng Hu. Secure Efficient Routing Based on Network Coding in the Delay Tolerant Networks[A]. IEEE Beijing Section.Proceedings of 2014 IEEE 5th International Conference on Software Engineering and Service Science[C].IEEE Beijing Section:,2014:4.

[8] Zhou Kuanlong, etc. Research of software security static detection methods based on the XML [J]. Computer engineering and application, 2010, 28:64-69.

[9] Meng Yunxiu. The static test of software based on source code of software [J]. Journal of hebei academy of sciences, 2013, 11:16-21.

[10] Fang Kaibin, The methods of code security testing for mobile Internet application [J]. China inspection and quarantine, 2013, 11:31-32.

[11] LI Zhongyuan,QU Chengqin,ZHOU Xueguang,ZHUO Lifeng. Review of Public-Key Cryptosystem Based on the Error Correcting Code[J]. Wuhan University Journal of Natural Sciences,2014,06:489-496.

[12] LI Zhongyuan,QU Chengqin,ZHOU Xueguang,ZHUO Lifeng. Review of Public-Key Cryptosystem Based on the Error Correcting Code[J]. Wuhan University Journal of Natural Sciences,2014,06:489-496.

[13] Ye Liang. Safety research of Source code analysis method based on the security rules [D]. Huazhong science and technology university , 2013.

[14] Yuan Bing, etc. Analysis technology of code audit for malicious backdoor[J]. Computer security, 2013, 10:47-49.

[15] you, zhang, Ma Yuanyuan, wei-wei li. An efficient detection source code security vulnerabilities of code review method [J]. Journal of modern electronic technology, 2015, 12:83-86.