Advanced Science and Technology Letters
Vol.93 (Security, Reliability and Safety 2015), pp.5-8
http://dx.doi.org/10.14257/astl.2015.93.02

# A Software Weakness Analysis Technique for Secure Software

Yunsik Son[1], Yangsun Lee[2], Seman Oh[1*]

[1] Dept. of Computer Engineering, Dongguk University
26 3-Ga Phil-Dong, Jung-Gu, Seoul 100-715, KOREA
{sonbug, smoh}@dongguk.edu
*Corresponding Author
[2] Dept. of Computer Engineering, Seokyeong University
16-1 Jungneung-Dong, Sungbuk-Ku, Seoul 136-704, KOREA
yslee@skuniv.ac.kr

**Abstract.** The recent computing environment is developing to the IoT services which exchange tremendous amount of information as the variety of devices are connected to network all the time. Since the data communication and services take places on the variety of devices including not only traditional computing environment and mobile devices such as smartphone but also embedded devices and sensor nodes, the security requirements is getting more important under this environment. This paper presents the weakness inspection mechanism to implement secure software on the IoT environment.

**Keywords:** Software weakness, Static analysis, Dynamic analysis, Secure software

## 1 Introduction

Recent computing environment exchanges tremendous amount of information on the variety of devices as shown on IoT service and big data processing. Based on this, it provides various services. The environmental characteristic highly enhanced the security requirements of the system structure, the data network and applications on devices and the increase of the devices has lifted the importance of security more. This paper presents the software secure mechanism for the applications run on this system.

## 2 Previous Software Weakness Analysis

As the 75% of recent software security incidents were caused by the applications containing vulnerable points in accordance with the Gartner's report [1], detecting and eliminating the possible weaknesses in program effectively from the application development stage became an very important issue. For the weakness inspection

ISSN: 2287-1233 ASTL

feature, there are two mechanism: static analysis which utilizes the tools which analysis the weakness of source code level such as Fortify 360 and Coverity Prevent and the dynamic analysis which utilizes the traditional software testing methodology and the testing tools [2, 3]. The static analysis is a technique to analyze without program execution, it utilizes token, abstract syntax tree, control flow, data flow graph and so on. The dynamic analysis is a technique which analyses programs by executing step by step by inserting the specific code at the execution time or mapping libraries. There are few similar analysis tools, for instance, MOPS which is the model inspection tool developed by UC Berkley [4], Safe-Secure C/C++ which is specialized in buffer overflow developed by Plum Hall and Sparrow which is specialized in buffer overflow and inspecting memory usage and errors by the Korean security corporation Pasoo.com. The development of various test tools suit to applications on mobile devices has been proceeding recently. Google provides Monkey test tool to test android applications and Wind-River developed the automatic test solution for android device development task. The most of the researches related to mobile devices, however, are focusing on the way to test the functions of the software only.

## 3 Proposed Weakness Analysis Model

The technique proposed in this study statically analyzes source codes, categorizes CWE weakness [5] into four groups and then detects the weaknesses. Fig. 1 is a depiction of the weaknesses detected through static analysis and the detected weaknesses are grouped into groups; Precision Weakness, Bypassed Weakness, Imprecision Weakness and Non-decision Weakness.

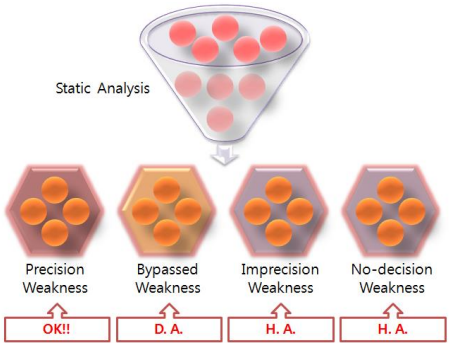


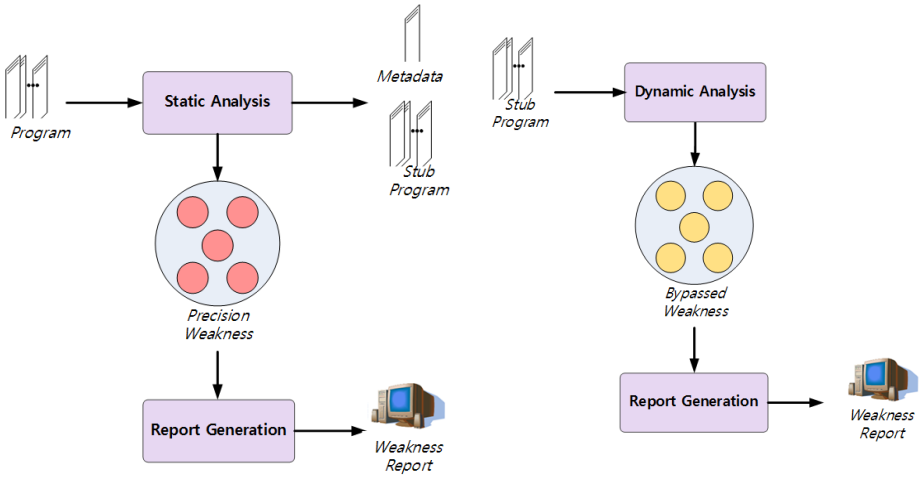

**Fig. 1.** Weakness classification of proposed method

In this paper, we deal with the static analysis and dynamic analysis except hybrid analysis.

### 3.1 Static Analysis

Precision Weakness refers to weaknesses that can be definitely detected only using static analysis. For such weaknesses, dynamic analysis is unnecessary and therefore omitted. Bypassed Weakness refers to weaknesses that cannot be detected only through static analysis and thus needs dynamic analysis to be carried out to be detected. Imprecision Weakness refers to weaknesses that have been detected through static analysis but are false positive or false positive. These require in-depth analysis during the dynamic analysis stage. Non-decision Weakness is similar to Bypassed Weakness as it can be detected through the static analysis stage however it is a weakness which has had information related to it collected during the analysis process. Fig. 2(a) shows weakness analysis through static analysis.




(a) (b)

**Fig. 2.** Model of weakness analysis by (a) static analysis and (b) dynamic analysis

After applications' source programs have passed the static analysis process, a weakness report on Precision Weaknesses is created. Also, the data collected during static analysis is created into Metadata form and source codes are altered and made into a stub program to allow further dynamic analysis to be carried out.

### 3.2 Dynamic Analysis

In the dynamic analysis process, only Bypassed Weaknesses are analyzed from the four groups of weaknesses. Bypassed Weaknesses are weaknesses that cannot be handled in the static analysis stage and since no information related to it is collected during previous stages existing dynamic analysis processes are used to examine the weaknesses. However the dynamic analysis technique proposed in the study excludes the weaknesses detected in the static analysis process and as a result the cost for this

process is much lower than the existing process. Fig. 2(b) is a depiction of a dynamic analysis model.

Dynamic analysis modules receive the stub program from the static analysis module. Stub programs are created by inserting a code for dynamic analysis to an application's source code. The inserted code can be divided into a code that is positioned in between the mobile platform and the analysis tool to dynamically transmit and receive information to and from them and a code test code to carry out dynamic analysis.

The input stub program is executed through the mobile platform. In the dynamic analysis process, analysis is carried out for weaknesses which have not been analyzed during the static analysis process, using the various interfaces provided by the platform. The weaknesses detected here are weaknesses that fall under the category Bypassed Weakness and another report is created in this analysis module.

Static weakness analyzer receive the source code of applications by as inputs and print out the inspected weaknesses by the stuff source file, meta data and static analysis. Dynamic weakness analyzer exchanges the stuff source files and information by utilizing the printed metadata and the weakness from the static analyzer and it prints the analyzed results.

## 4 Conclusions and further researches

This paper presents the mechanism to develop the software for mobile, embedded and IoT in secure. It suggests the weakness analysis technique based on the hybrid analysis, which combined the both strengths and weaknesses of the existing static analysis and the dynamic analysis. It isn't only enabling the development of strong applications against the external attacks but also reducing the huge amount of cost by preventing the problems that possibly occurs on service operational stage.

## References

1. Roberta Cozza, Carolina Milanesi, Anshul Gupta, Hugues J. De La Vergne, Annette Zimmermann, CK Lu, Atsuro Sato, and Tuong Huy Nguyen, Competitive Landscape: Mobile Devices, Worldwide, 3Q10, Gartner Inc., USA, 2010.
2. J. Viega, G. MaGraw, Software Security, How to Avoid Security Problems the Right Way, Addison-Wesley, USA, 2006.
3. J. McManus and D. Mohindra, The CERT Sun Microsystems Secure Coding Standard for Java, CERT, USA, 2009.
4. H. Chen and D. Wagner, "MOPS: an infrastructure for examining security properties of software", Proceedings of the 9th ACM Conference on Computer and Communications Security, USA, 2002, pp.235-244.
5. Common Weakness Enumeration (CWE): A community-Developed Dictionary of Software Weakness Types, http://cwe.mitre.org/.
6. Y.S. Son, S.M. Oh, "Design and Implementation of a Compiler with Secure Coding Rules for Secure Mobile Applications", International Journal of Security and Its Applications, SERSC, Vol.6, No.4, 2012, Australia, pp.201-206.