Advanced Science and Technology Letters
Vol.110 (ISI 2015), pp.67-70
http://dx.doi.org/10.14257/astl.2015.110.14

Design of the Secure Compiler for the IoT Services

Yunsik Son*, Junho Jeong*, YangSun Lee**

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

Abstract. Recently, the computing environments are developing to the IoT services which exchange a lot of information using various and heterogeneous devices that always connected on networks. Since the data communication and services take places on the various devices including not only traditional computing environments and mobile devices such as smartphone but also household appliances, embedded devices, and sensor nodes, the security requirements is getting more important at this point in time. In this paper, the compiler with secure software concept was proposed to develop the secure applications for IoT services.

Keywords: Secure Software, S/W Weakness, Compiler Construction, Program Analysis, IoT Services

1 Introduction

Recently, the computing environments are changing to the IoT, but the IoT services have high security problems such as hacking and exploiting because almost devices of IoT systems connected on the internet and transmit data over the network. IoT sensors or devices are exposed to relatively high security threat than the traditional server system inside firewall or IDS, if such terminal devices are under attack from the outside, the whole IoT based services can't operate normally or work with abnormal behaviors.

In this paper, we propose the secure compiler to develop the secure IoT applications on the computing environments with various IoT devices. For the proposed compiler, we apply the secure software concept to the compiler construction phase and add the 2 modules that designed for secure software on our compiler model of smart cross platform.

ISSN: 2287-1233 ASTL

2 Secure Software

The software of today exchanges data in the internet environment making it difficult to secure validity of the data input and output. There exists the possibility of being maliciously attacked by unknown and random invaders. This weakness has been the direct cause of software security incidents which generate significant economic losses or social problems [1].

Security systems installed to prevent security incidents from occurring, mostly consist of firewalls, user authentication system and etc. However, according Gartner's report 75% of software security incidents occur due to application programs including weaknesses. Therefore rather than making security systems for the external environment more firm, programmers creating software codes more firm is the more fundamental and effective method of increasing the security levels. However, efforts to reduce the weaknesses of a computer system are still mainly biased to network servers [1, 2].

Recently, there has been recognition of this problem and therefore research on secure coding, writing secure codes from the development stage, is being carried out actively. Especially, CWE(Common Weakness Enumeration), an organization which analyzes the weaknesses that can arise from programming language, has analyzed and specified the various weaknesses that can occur in the source code creation stage by the different languages [3]. Also, CERT(Computer Emergency Response Team) defines secure coding rules to ensure secure source code creation [2].

3 Secure Compiler for IoT Services

In order to design the secure compiler, a secure coding rule checker and a static weakness analyzer are added on the compiler model for smart cross platform [4]. In this study, the secure compiler was designed by 8 parts as can be seen in Fig. 1.

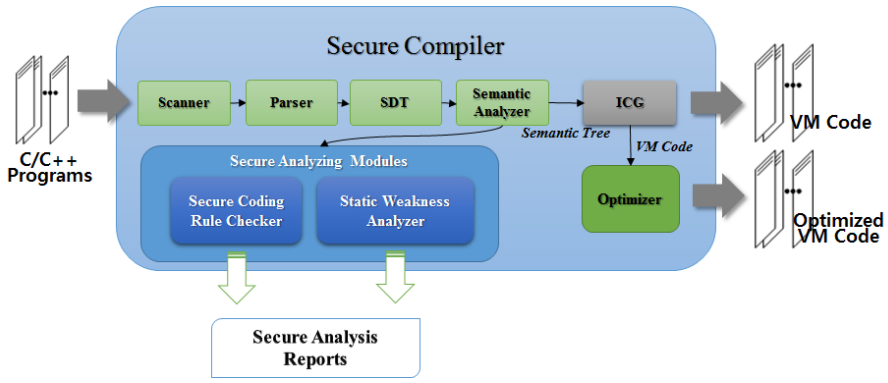

Fig. 1. Proposed Secure Compiler Model for C/C++ Languages

The secure compiler provides secure features to prevent the software weaknesses of the input program source code in C/C++. It was designed with six general compiler parts - Scanner (lexical analysis), Parser (syntax analysis), SDT (Syntax Directed Translation), Semantic analyzer, ICG (Intermediate Code Generator), and Optimizer – and 2 kinds of the secure part; Secure coding rule checker, Static weakness analyzer. The detailed information for each part is as follows.

Scanner, Parser, and SDT modules are easily can be group as a processor to analyze the input C/C++ programs and generate analyzed AST (Abstract Syntax Tree) for the input programs.

Semantic analyzer checks the process of collecting symbol information on the AST level, to verify cases which are grammatically correct but semantically incorrect. And it uses the AST and symbol table to carry out semantic analysis of statements and creates a semantic tree as a result. A semantic tree is a data structure which has semantic information added to it from an AST. It is used for not only generating the VM (Virtual Machine) code but also analyzing software weaknesses.

The code generation module receives the semantic tree as an input after all analysis is complete and it generates a VM code which is semantically equal to the input program in C/C++.

Secure coding rule checker is the module to find the rule violations of the input programs. The coding rules defined by meta-language that was designed for describe the secure policy of the target programming languages. The defined rules are interpreted by rule checker, and the checker analyze the violations using the input semantic tree with interpreted rule information.

Static weakness analysis module analyzes the control flow and data flow of a source program by using the symbol information and semantic tree generated by the front end of the compiler. Some weaknesses are too difficult to figure it out precisely by rule checker. In that case, the rule checker generates too many false alarms for target weaknesses [5]. For the case of these weaknesses, we are using the static weakness analysis module that has 1:1 mapping routines for specific weakness analysis.

The software weaknesses that analyzed by proposed compiler are collected and categorized from the CWE and OWASP's defined top level weaknesses about embedded system, network, IoT, and C/C++ programming languages.

4 Conclusions and Further Researches

In this paper, we have designed a new compiler to support secure software for IoT services. We defined 8 modules to construct the proposed compiler and to generate a VM code for use on the IoT VM which is independent of platforms with secure software features. We hope to the compiler will be implemented that is enhance the secure features of the IoT services. Also, we expect that expand the coverage of previous IoT service developmental platforms and reduce the cost of developing secure services by the proposed compiler. The compiler for weakness analysis proposed in this study examines the weaknesses that can exist within programs at the

beginning of application development. It also enables safe applications development and a differentiated function from existing developing/testing tools.

In the future, there is need for implementation phase of a secure compiler to analyze the software weaknesses on IoT applications. Further research on VM optimization for IoT services is also needed.

Acknowledgments. This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT and Future Planning (No.2013R1A2A2A01067205).

References

1. J. Viega, G. MaGraw, Software Security, How to Avoid Security Problems the Right Way, Addison-Wesley, USA (2006)
2. J. McManus and D. Mohindra, The CERT Sun Microsystems Secure Coding Standard for Java, CERT, USA (2009)
3. Common Weakness Enumeration (CWE): A community-Developed Dictionary of Software Weakness Types, http://cwe.mitre.org/.
4. Y.S. Lee, Y.S. Son, “A Study on the Smart Virtual Machine for Executing Virtual Machine Codes on Smart Platforms”, International Journal of Smart Home, SERSC, vol. 6, no. 4, pp. 93--105 Australia (2012)
5. 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, pp. 201--206 Australia (2012)

