Future Generation Computer Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Design and implementation of the secure compiler and virtual machine for developing secure IoT services

YangSun Lee [a], Junho Jeong [b], Yunsik Son [b,c,*]

[a] Department of Computer Engineering, Seokyeong University, 16-1 Jungneung-Dong, Sungbuk-Ku, Seoul 136-704, Republic of Korea
[b] Department of Computer Engineering, Dongguk University, 26 3-Ga Phil-Dong, Jung-Gu, Seoul 100-715, Republic of Korea
[c] Department of Brain and Cognitive Engineering, Korea University, 145 Anam-ro, Seongbuk-ku, Seoul 136-713, Republic of Korea

HIGHLIGHTS

- Secure software for developing secure/trustworthy services for IoT was proposed.
- A secure compiler was used in the development phase to eliminate the weaknesses.
- A virtual machine was used in the operating phase to watch the abnormal behaviors.

ARTICLE INFO

Article history:
Received 15 October 2015
Received in revised form
24 February 2016
Accepted 23 March 2016
Available online xxxx

Keywords:
Secure software
IoT services
S/W weakness
Program analysis
Compiler construction
Virtual machine

ABSTRACT

Recent years have seen the development of computing environments for IoT (Internet of Things) services, which exchange large amounts of information using various heterogeneous devices that are always connected to networks. Since the data communication and services occur on a variety of devices, which not only include traditional computing environments and mobile devices such as smartphones, but also household appliances, embedded devices, and sensor nodes, the security requirements are becoming increasingly important at this point in time. Already, in the case of mobile applications, security has emerged as a new issue, as the dissemination and use of mobile applications have been rapidly expanding. This software, including IoT services and mobile applications, is continuously exposed to malicious attacks by hackers, because it exchanges data in the open Internet environment. The security weaknesses of this software are the direct cause of software breaches causing serious economic loss. In recent years, the awareness that developing secure software is intrinsically the most effective way to eliminate the software vulnerability, rather than strengthening the security system of the external environment, has increased. Therefore, methodology based on the use of secure coding rules and checking tools is attracting attention to prevent software breaches in the coding stage to eliminate the above vulnerabilities. This paper proposes a compiler and a virtual machine with secure software concepts for developing secure and trustworthy services for IoT environments. By using a compiler and virtual machine, we approach the problem in two stages: a prevention stage, in which the secure compiler removes the security weaknesses from the source code during the application development phase, and a monitoring stage, in which the secure virtual machine monitors abnormal behavior such as buffer overflow attacks or untrusted input data handling while applications are running.

1. Introduction

Recently, the expansion of computing environments to the IoT (Internet of Things), and mobile and cloud computing have resulted in privacy and system security issues becoming more important. Especially, the software included in mobile applications will always be vulnerable to possible malicious attacks by hackers, because it exchanges data in the Internet environment. These security weaknesses are the direct cause of software breaches, thereby causing serious economic loss. Moreover, in recent years the computing environment has been changing into a complicated system composed of various and heterogeneous sensors, IoT/embedded devices, mobile devices, PCs, and servers from the traditional environments.

* Corresponding author at: Department of Brain and Cognitive Engineering, Korea University, 145 Anam-ro, Seongbuk-ku, Seoul 136-713, Republic of Korea.
E-mail addresses: yslee@skuniv.ac.kr (Y. Lee), yanyenli@dongguk.edu (J. Jeong), sonbug@dongguk.edu, sonbug@korea.ac.kr (Y. Son).

http://dx.doi.org/10.1016/j.future.2016.03.014
0167-739X/

In this environment everything is connected; hence, it is difficult to apply conventional application development methods and execution environments to this complicated system. Thus, IoT services are vulnerable to serious security problems such as hacking and exploiting, because almost all the devices of IoT systems are connected to the Internet and transmit data over the network. IoT sensors or devices are more exposed to relatively serious security threats compared to a traditional server system inside a firewall or IDS (Intrusion Detection System). When such terminal devices are under external attack, the entire IoT-based services are unable to operate normally because of the abnormal behavior.

In this regard, offering a secure coding guide or static analysis tools to solve software weaknesses from the coding stage is a trend, nowadays. If weaknesses are considered and prevented from the software development stage, enormous cost can be cut, compared to the efforts to recognize and correct weaknesses in the operation stage, and also huge contribution can be made to the development of safe software from hackers [1,2].

Our research team is working to solve this problem with the aim of producing high-quality/trustworthy IoT services by developing technology for IoT secure software development and execution based on a compiler and virtual machine. In this paper, we propose the use of a secure compiler and virtual machine with a stack monitoring method to develop secure IoT applications and protect abnormal behavior in computing environments containing various IoT devices.

A secure compiler was designed for preventing software weaknesses in the source code during the application development phase, and it is combined with a traditional compiler and weakness analyzer to generate the target code and remove the weaknesses. The secure compiler is implemented in conjunction with a virtual machine, which monitors abnormal behavior such as buffer overflow attacks or untrusted input data handling to protect the system while the applications are running.

The contents of this paper are as follows. First, in Section 2, secure coding, weakness analysis tools, and a smart cross platform are examined. Next, in Section 3, the technique proposed in this paper is introduced. In Section 4, the results obtained by applying the proposed method are analyzed and evaluated. Lastly, in Section 5, the conclusion and future direction of research are discussed.

2. Related studies

2.1. Secure coding

The software of today exchanges data in the Internet environment, thereby making it difficult to secure the validity of the data input and output. The possibility of being maliciously attacked by unknown and random invaders exists. 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 systems, etc. However, according to a Gartner report [2], 75% of software security incidents occur because of weaknesses in the application programs. Therefore, rather than strengthening the security systems for the external environment, the creation of more secure software code by programmers is a 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.

Recently, there has been recognition of this problem and therefore research on secure coding, that is, writing secure codes from the development stage [3,4] onwards, is being carried out actively. Especially, CWE (Common Weakness Enumeration) [5], an organization that 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 of different languages. Also, CERT (Computer Emergency Response Team) [6] has defined secure coding rules to ensure secure source code creation. In Cigital [7], the weaknesses can be eliminated by using the 61 rules classified according to the Seven Pernicious Kingdoms [8] classification method proposed by Katrina Tsipenyuk, Brian Chess, and Gary McGraw. The coding rule suggested by Cigital is defined in XML form and can be used as an input in weakness analyzers and other programs. Industries prone to fatal mistakes due to software defects, such as the airplane and car industry, have implemented coding rules, such as JSF and MISRA Coding Rule [9], to contribute towards high quality software development.

2.2. Source code weakness analyzer

According to a report by Gartner [2], 75% of recent software security incidents were caused by applications containing vulnerable points; thus, the effective detection and elimination of possible weaknesses in a program from the application development stage has become a very important issue.

The source code weakness analyzer is a tool which has been developed to automatically examine the weaknesses within source code after it has been created by a programmer. Programmers aspire to have weaknesses within their programs to be entirely eliminated. However, it is difficult to acquire expert knowledge about weaknesses and it is difficult to recognize how to alter such weaknesses. Therefore, there is a need for a tool capable of automatically analyzing weaknesses at the source code level. There exists a suitable weakness analysis method depending on each weakness and these are broadly classified into static and dynamic analysis methods. The static method uses technology that does not require the subject program to run and uses methods such as token, AST (Abstract Syntax Tree), CGF (Control Flow Graph), DFG (Data Flow Graph). The dynamic method uses technology that performs a level-by-level analysis of programs while they are running and it uses certain codes that can either be used during execution time or by library mapping to carry out the analysis.

MOPS (MOdel Checking Programs for Security properties) [10] is a model testing machine developed at the University of California, Berkeley. MOPS defines the properties of security weakness factors, and has been standardized using limited automata. Accordingly, weaknesses that have been modeled can all be examined at low analysis costs. However, since it does not analyze the flow of data, there is a limit to the weaknesses that can be analyzed. Safe-Secure C/C++ by Plum Hall [11] is a type of compiler that has combined a compiler with a software analysis tool. Safe-Secure C/C++ only focuses on eliminating buffer overflow. Execution programs created using this software are capable of eliminating buffer over-flows 100% and have less than a 5% decrease in function compared to execution files created by ordinary compilers. Coverity's Coverity SAVE [12], is a static analysis tool for source codes. Coverity SAVE shows all weaknesses discovered in codes as a list. Each list includes details on the location of and reason for weaknesses discovered within each list. Fortify Static Code Analyzer (SCA) [13] is a weakness detection tool. Fortify SCA supports C/C++, Java, and other languages, and uses both static and dynamic analysis to detect weaknesses in source codes. The detected weaknesses are given to the user along with statistical data. Compass [14] is an open source static analysis tool for C/C++ based on ROSE [15]. Rule-based Compass uses the source-to-source framework ROSE for source code transformations, allowing users to modify the domain-specific rules sets of this tool. Sparrow [16] is a tool that carries out

Please cite this article in press as: Y. Lee, et al., Design and implementation of the secure compiler and virtual machine for developing secure IoT services, Future Generation Computer Systems (2016), http://dx.doi.org/10.1016/j.future.2016.03.014