A Mobile OS Integrity Check Using Bootloader & Storage Device of Physical Independent

Dong-yeob Oh[1*], Jae-Kyung Park[2]

[1] Cyber Security Research Center of KAIST, KOREA
[2] Cyber Security Research Center of KAIST, KOREA
[Email : oh51dy@kaist.ac.kr, wildcur@kaist.ac.kr]

Abstract

The dissemination of smartphones is rapidly progressing and there are many similarities of smartphones and PCs in terms of security risks. Recently, in mobile network environment, there is a trend of increasing damages and now, there are active researches on a system that can comprehensively respond to this. As a way to prevent these risks, integrity checking method on operation system is being researched. As most integrity checking algorithms are classified by verification from the levels before booting the OS and at the time of passing on the control to the OS, in which, there are minor differences in the definitions of integrity checking or its methods. This paper researched the verification techniques for OS integrity that can be more fatal than apps in case of security issues, and suggests the integrity verification technique of OS using a boot loader and a physically independent storing device in the terminal.

Index Terms: Android, Integrity, Kernel, Bootloader.

I. INTRODUCTION

In time, mobile terminals have changed in various ways, and the directions are changing from closed structure for applications and contents from open types. Similarly, mobile OS have changed from exclusive OS to universal OS. Recently, security risks for mobile terminals have rapidly increased. Although researches and efforts are put into responding to these mobile security risks, complete protection against malicious risks that come from all directions hasn't been found. The kernel area of OS can be exposed to various risks by attacks by hackers. This paper suggests integrity verification technique for OS using an independent storing device in terminal in order to assure safer mobile environment from risks of mobile OS.

II. RELATED RESEARCH

Currently, various researches are being progressed for integrity verification of mobile OS. Recently, a mobile security solution by Samsung called KNOX was introduced. This solution is a multi-directional security solution based on device hardware that can prevent falsification through Linux kernel and Android OS. This is a wide-ranged mobile security platform that includes app and information protection including integrity verification of Android OS, and responsive measures to risks through network infrastructure via VPN connection for each app.

The integrity verification technique of KNOX suggests integrity verifications of OS and block level. In particular, it is important to look closer at the integrity verification technique for OS in the boot level that consists of Secure Boot and Trusted Boot. As for Trusted Boot, all boot loaders and encrypted fingerprints of OS kernel are stored using ARM TrustZone technology, and integrity verification is performed using this information.

III. SYSTEM MODEL AND METHODS

The integrity verification technique suggested by this paper intends to secure independent security memory area where verification data is loaded and produce the interface accessible to relevant security memory area as the kernel module. This module shall be loaded to kernel while booting the OS at the boot loader, and the relevant data can be accessed via the integrity verification application that uses this module.

A. Securing independent security memory area

Security memory area refers to a new storage medium that doesn't exist in previous mobile devices and can be accessed from OS only when device driver is present along with other parts of the device. Likewise, the OS performs I/O to the actual device according to user request via the I/O interface defined by the device driver. The independent security memory area suggested in this paper is designed to control the access to the relevant area by directly producing the device driver as in the illustration 1. The access to the relevant memory area is designed to allow read access from both boot level and OS level. However, in case of writing function, it is designed to only allow at the boot loader area. This is because in case of falsification of OS by malicious user and extortion of the control of the OS, access to the prepared security memory area is lost. For this reason, although reading function is provided, in order to defend against falsification and counterfeit of data in the relevant area in advance, writing function is removed. In addition, on the device driver, in order to secondarily manage the access to the saved data from I/O, encrypting/decrypting functions are added. The relevant driver is produced in advance as kernel module type and managed separately from the kernel where the relevant driver is loaded to kernel after loading the kernel image at the boot level.

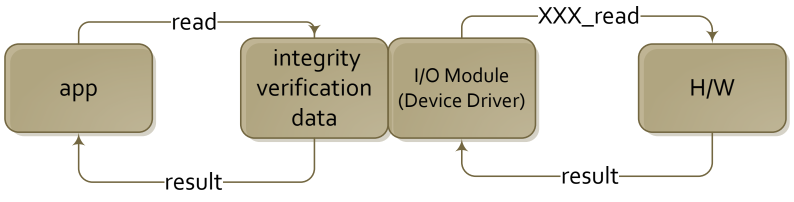

Fig 1 . Access to the data associated with integrity verification

B. Integrity verification using the bootloader

Booting to falsified OS can happen by abusing the boot area. Actually, rooting, blamed as the biggest security concern of mobile terminals, is frequently occurring not only by abusing the weak point of OS but also starting by editing the boot area. Accordingly, mobile terminal manufacturers and other researchers are processing boot level in various ways in order to upload only the selected safe OS by applying solutions such as the aforementioned KNOX.

The boot loader in this paper is designed to allow developers to arbitrarily add additional functions. This means, they can load external OS downloaded from network communication to main memory or control specific devices. By doing this, they can attempt to verify integrity of OS by accessing to the security memory area. The series of processes are as in illustration 2.

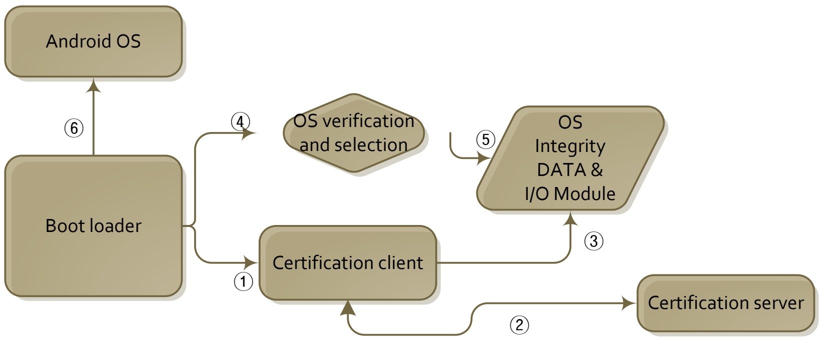

Fig 2 . Verify the integrity of the boot level

- At the time of initial terminal production, it is distributed in the state of recording the integrity verification data of the verified OS image in the relevant memory area. The procedures are as follows.

Stages (1)(2)(3): When booting the terminal, booting process is progressed in sequential order, and then the integrity verification data of OS in the security memory area performs interface and processing in the OS falsification and counterfeit testing stage to enable renewal of data by communication with terminal manufacturer.
Stage (4): Perform verification function to check presence of OS falsification and counterfeit
Stage (5): To verify falsification and counterfeit, determine whether data in the security memory area and actual OS value are the same
Stage (6): If verification is complete, load relevant OS to the memory, progress copying I/O module included in the security memory are to the file system to be used by OS and deliver control to the OS.

By the procedures above, the kernel I/O module to the security memory area managed by boot loader is loaded to the OS and control is transferred.

C. Verify the integrity of the operating system level

The relevant module is the device driver to the security storing device, and the functions of I/O of the relevant driver exclude writing function. In other words, only open, read and close functions are performed to allow only access to the relevant data on the OS and disable additional editing. This is to defend against falsification and counterfeit of integrity verification data of OS from malicious attacks. By using the device driver inserted to the OS in the module type, the verified integrity testing application regularly checks the status of integrity of system area of the OS. When doing this, the reliability of the integrity checking application is extremely crucial.

IV. DISCUSSION AND CONCLUSIONS

The suggested integrity verification method of mobile OS managed a separate storing device to perform verification on the boot level before the OS level, utilizes the same verification data even after the control is transferred to the OS and performs consistent integrity verification.

The device driver in charge of the I/O to the relevant storing device is produced in advance as the kernel module and separately managed from the kernel. The relevant can be safe from outflow of firmwares by loading when the boot loader loads the OS to the memory from boot level instead of distributing as contained in the OS.

REFERENCES

[1] Survey of Security Threats and Contermeasures on Android Environment. Joonhyouk Jang 2013.12

[2] Integrity Protection Solutions for Embedded Systems, FOSDEM 2014 Brussels, Belgium, February , 2014

[3] Sangorrin, Daniel, Shinya Honda, Hiroaki Takada. "Dual operating system architecture for real-time embedded systems." In Proceedings of the 6th International Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT), Brussels, Belgium, pp. 6-15. 2010.

[4] The Implementation of Recording and Replaying System and Its Device Driver Programming. Hyo-Jung Choi, Joong-Ho Lee, and Dae Jin Kim. 2003.11

[5] Measuring Integrity on Mobile Phone Systems. Divya Muthukumaran, Anuj Sawani, Joshua Schiffman, Brian M. Jung, Trent Jaeger. SACMAT '08 Proceedings of the 13th ACM symposium on Access control models and technologies Pages 155-164

[6] Building Efficient Integrity Measurement and Attestation for Mobile Phone Platforms. Xinwen Zhang, Onur Acıiçmez, Jean-Pierre Seifert. Security and Privacy in Mobile Information and Communication SystemsLecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering Volume 17, 2009, pp 71-82