Available online at www.sciencedirect.com

ScienceDirect

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

Computers & Security

CDroid: practically implementation a formal-analyzed CIFC model on Android

Zezhi Wu[a,b], Xingyuan Chen[a,b,*], Xuehui Du[b], Zhi Yang[b]

[a] Institute of Zhengzhou Information Science and Technology, Zhengzhou, China
[b] State Key Laboratory of Cryptology, Beijing, China

ARTICLE INFO

Article history:
Received 28 August 2017
Revised 16 March 2018
Accepted 6 July 2018

Keywords:
Android
Information flow control
Security process algebra
Dynamic taint tracking
Noninterference

ABSTRACT

Decentralized information flow control (DIFC) operating systems provide mechanisms for applications to handle the secrecy and integrity of their data by themselves. DIFC adapts to the distributed systems well, but not for the centralized authorization systems where an administrator manages all the privileges. For example, Android is full of untrusted third-party applications. A phone user may want to specify what kind of application can deal with what kind of private data by enforcing information flow control. To address this, we proposed a novel formal-described and security-proofed centralized information flow control (CIFC) model. In CIFC, taint *tag* of private data and capability label of applications are designed to support fine-grained and user-defined information flow control. Differs from DIFC and classic information flow control models, CIFC model controls information flow according to the relation between *tag* and *label* rather than the relation between two *labels* of applications. We use Value-passing Security Process Algebra (VSPA) to clarify the formal semantics of CIFC model. The verification of system equivalence proves that the model guarantees the noninterference security property in virtue of Checker of Persistent Security (CoPS) tool. We also implemented CDroid, a prototype of the CIFC model which can track and control information flow at runtime. CDroid is demonstrated to be an accurate system to achieve the security goal through several function test experiments. Furthermore, CDroid has 5% lead in memory consumption and 17% overhead of runtime performance compared to Android.

1. Introduction

Modern operating system (OS) provides access control and permission mechanisms to help the users to handle their private data. However, only the releases of private data are controlled by these mechanisms, the transfers of private data are not controlled in OS. Once any of the applications gets the data, no more rules are made to guarantee the application uses the private data legally.

Information flow control technique can monitor the private data transfer within the OS and ensure that the private data are securely handled by applications. Classic information flow control was initially applied in military systems in order to protect information confidentiality (such as BLP model) and integrity (such as BIBA model). Myers and Liskov (1997) firstly presented a Decentralized Label Model (DLM) to control information flow during compiling in decentralized authority systems. Other researches (such as Asbestos (Efstathopoulos et al., 2005), HiStar (Zeldovich et al., 2011), Flume

* Corresponding author at: Institute of Zhengzhou Information Science and Technology, Zhengzhou, China.
E-mail addresses: 1141208772@qq.com (Z. Wu), chxy302@vip.sina.com (X. Chen), dxh37139@sina.com (X. Du), zynoah@163.com (Z. Yang).

https://doi.org/10.1016/j.cose.2018.07.005
0167-4048/© 2018 Elsevier Ltd. All rights reserved.

(Krohn et al., 2007), Dstar (Zeldovich et al., 2008), Laminar (Roy et al., 2009), Aeolus (Cheng et al., 2012)) have implemented varieties kind of DIFC OSs. However, these systems have not been widely used. The reasons are listed as follows:

1. Some of the information flow control models (IFCMs)(such as Asbestos, HiStar, Flume, Dstar) are too complex for the users and programmers to understand. The other IFCMs (such as BLP, BIBA, DLM) are easier to learn, but their information flow control rules are too rigid in majority of the cases.
2. DIFC systems provide diversity models with application-defined security policies for individual applications. The private data are protected in the method of allocating and declassifying labels. However, very little contributions are made to an system administrator on system-wide security policies.
3. Source codes need to be rewrite by the programmers in order to work with the new programming model. This makes the DIFC systems are not compatible with off-the-shelf applications. Meanwhile, the majority of the systems provide coarse-grained information flow control, only few of the systems provide fine-grained information flow control but with heavier runtime overhead.

To address these, we present a novel formal-described and security-proofed centralized information flow control (CIFC) model. CIFC is a centralized authority model which provides security guarantee to a monolithic user who fully controls the system. The flow of their private data is controlled by themselves which is much safer than controlled by the program or application in OS. In CIFC, taint *tag* of private data and capability *label* of application are designed to support fine-grained and user-defined information flow control. Differs from traditional DIFC and classic IFC models, the information flow controlled by CIFC model are based on the relation between taint *tag* and capability *label* rather than the relation between the two *label*s of applications. For example, there are two applications, one is labeled as “secret” and the other is labeled as “public”. The information flow from applications “secret” to application “public” is not allowed in DIFC and classic IFC models in order to protect data confidentiality. But in our model, the information flow from application “secret” to application “public” is not allowed only when the transferred data contain “secret” information. This makes our model is more flexible and available compares to the previous models. What’s more, *label*s of capability are classified into *send* and *receive* in our model instead of *secrecy* and *integrity* in the DIFC models which offers a better description on information flow characteristics. our model also offers a better implementation on the separation of duties and the principle of least privilege via the label constraints. The formal semantics of CIFC model is clarified by using the Value-passing Security Process Algebra. The verification of system equivalence proves that our model guarantees noninterference security property of Strong Bisimulation-based Non Deducibility on Compositions in virtue of CoPS (Focardi and Gorrieri, 2000; Piazza et al., 2004) tool. We also implemented CDroid, a prototype of CIFC model. With the aid of TaintDroid (Enck et al., 2014), CDroid tracks and controls information flow dynamically at the granularity of individual variables. Meanwhile, any third-party application can successfully run in CDroid without any modification. CDroid also can be packaged as an upgrade package in zip format and installed from SD card. What’s more, several applications are provided to make the usage of CDroid more easier. Application CPolicy is designed for the user to define CIFC policy. Application CNotify is designed to notify the user when a violation of CIFC policy occurs. Application FAdmin is designed for the user to set or declassify the taint *tag* for a file.

CDroid is demonstrated to be a practical and accurate system that can achieve the security goals which are presented in section 2 easily. To the best of our knowledge, these goals are not supported by existing systems or researches. We also evaluated our prototype on memory consumption and runtime performance. CDroid has 5% lead in memory usage and 17% runtime overhead compared to Android. The results of function and performance tests prove that CDroid is a fine-grained, accurate, flexible and available system which tracks and controls information flow at runtime. What’s more, Source codes of CDroid can be download at https://pan.baidu.com/s/1pLAYlXx. We also provide system images for emulator of CDroid at https://pan.baidu.com/s/1nvTZfRV.

The contribution of this paper can be summarized as: 1. A novel CIFC model is presented for centralized authorization operating systems to enforce user-defined and system-wide information flow control. This CIFC model is simpler than previous DIFC models. 2. The SBNDC (Focardi and Gorrieri, 2000) noninterference security property of CIFC model is formal-analyzed and automatic-proofed. 3. The very initial design and implementation of VM-level and OS-level CIFC for smartphone operating systems.

2. Motivating example

One of the prominent features of Android OS is allowing users to download third-party applications. These applications might be developed by untrusted parties, which may lead to some uncertainties on abuse of user’s privacy. Fig. 1 shows an ordinary example of user’s privacy requirements in Android OS. In order to ensure the functional availability, application WPS is allowed to read sensitive file and send data out via the network interface. Nevertheless, in order to ensure the privacy and security, WPS is not allowed to send the sensitive file (even a word or a sentence in the file) out via the network interface or receive contacts information from application Message. Similarly, Message is allowed to read sensitive file or contacts data and send messages information out via the message interface, but Message is not allowed to send the sensitive file or contacts information out via the message interface. However, These common and practical demands are not supported by Android or any existing researches.

Allowing an application to read sensitive file or send data out via the network interface can be achieved by access control and permission mechanisms, but these mechanisms cannot fully support the further demand well: if the user gives WPS reading and sending permissions, then he has no idea to prevent WPS from sending sensitive file out via the network interface. If the user only gives WPS reading permission and do not gives the sending permission, this can prevent WPS