The Journal of Systems and Software 120 (2016) 133–153

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

Privacy protection by typing in ubiquitous computing systems

François Siewe[a,*], Hongji Yang[b]

[a] School of Computer Science and Informatics, De Montfort University, Leicester, LE1 9BH, United Kingdom
[b] Centre for Creative Computing, Bath Spa University, Bath, BA2 9BN, United Kingdom

ARTICLE INFO

Article history:
Received 2 November 2015
Revised 4 May 2016
Accepted 24 July 2016
Available online 26 July 2016

Keywords:
Privacy
Type system
Ubiquitous computing
Pervasive systems
Security
Type-checking
Simulation

ABSTRACT

Ubiquitous computing systems collect and share a great deal of information upon the users and their environment; including private or highly sensitive personal information. Unless users are confident enough that their privacy is protected, many will be deterred from using such systems. This paper proposes a privacy type system that controls the behaviour of concurrent, context-aware and mobile processes to ensure that private information is not accidentally disclosed. We prove the subject reduction property and the soundness of the proposed type system; which guarantee that a well-typed process cannot accidentally disclose private information. We demonstrate the pragmatics of our approach with a case study.

1. Introduction

Thanks to the advances in technologies, the vision of ubiquitous computing (ubicomp, for short) (Weiser, 1991) is increasingly becoming a reality with the proliferation of smart handheld devices such as smart phones and tablet computers capable of providing the user with relevant information and services anytime and anywhere. These smart devices are equipped with a variety of sensors to collect information upon the user context and share this information via a network to enable timely adaptation to changes in the user context. These may include private or highly sensitive personal information such as the user's location, activity, or personal health information that must be protected from falling into the wrong hands. For example, a smart phone equipped with a GPS (Global Positioning System) receiver can sense the user location and activity (e.g. walking or driving) and eventually share them with other devices or services via the Internet. According to eMarketer's prediction (eMa, 2015), by 2017 the number of mobile phone users will surpass 5 billion worldwide; making it more urgent than ever to develop mechanisms for protecting user privacy in ubicomp systems.

However, privacy is a subjective concept based on personal perceptions of risk and benefit (Schilit et al., 2003). In general, people are likely to disclose personal information in exchange of services if they believe the benefit outweighs the potential cost of this information being misused (Schilit et al., 2003; Hong et al., 2004). This is not to say that because ubicomp carries many benefits to the user –in term of calm, invisible, context-aware and adaptive computing– privacy is not an issue in ubicomp. Surely, the threat of violating individual's privacy is more severe than ubicomp systems not being used or accepted. Ubicomp is so intrusive to our everyday life; collects all kind of information about us in a completely unobtrusive manner, anywhere and anytime; and shares this information with the environment without our knowledge most of the time. Surely, there is a high risk that private information be disclosed accidentally; and more so in large-scale distributed ubicomp systems. Unless a mechanism is put in place that gives users enough confidence that their privacy is protected, many will be deterred from using such systems. To address this problem, type systems are a powerful technique to constrain the behaviours of a system so that certain errors and security violations do not occur at run-time. A modelling language with a well-defined formal semantics will help to understand the behaviours of ubicomp systems and to reason about the system requirements prior to implementation.

The Calculus of Context-aware Ambients (CCA, for short) (Siewe et al., 2011) is a formal language for modelling ubicomp systems. The main features of the calculus include *context-awareness*, *mobility* and *concurrency*. The concept of *ambient*, inherited from (Cardelli and Gordon, 2000), represents an abstraction of a place where computation may happen. An ambient may contain other ambients called child ambients organised into a tree structure. Such a hierarchy can be used to model any entity in a ubicomp

* Corresponding author.
E-mail addresses: fsiewe@dmu.ac.uk (F. Siewe), h.yang@bathspa.ac.uk (H. Yang).

http://dx.doi.org/10.1016/j.jss.2016.07.037
0164-1212/

system –whether physical, logical, mobile or immobile– as well as the environment (or context) of that entity (Siewe et al., 2011). For example the user location, the user profile, the mobile devices carried by the user, and the nearby resources can be modelled using the concept ambient (see Al-Doori et al., Al-Sammarraie et al. (2010); Almutairi and Siewe (2013)). In addition, CCA models are fully executable (using the CCA interpreter ccaPL), which is very useful for analysing the behaviour of ubicomp systems. Hence the reasons why CCA is chosen for this work. In CCA ambients communicate by message passing, can be mobile (or immobile), and can be aware of the presence of other ambients. These interactions among ambients, if not properly regulated, may lead to unwanted disclosure of private information, whether directly or indirectly.

This paper proposes a novel type system that constrains the behaviour of CCA processes to ensure that private information is not accidentally disclosed. In this type system, ambients are assigned to groups and have control over who can access their context information, who can share them (through message passing) and with whom. This enables the users to control how their context information is collected, stored and shared. Type checking guarantees that a well-typed process cannot violate the privacy of any ambient. The main contributions of this work are fivefold:

- A syntax of the types is proposed (Section 5.1); its innovative features include notations for describing privacy types. Mobility types and exchange types can also be specified. Privacy types are used to specify the *privacy requirements* of ambients. The privacy type of an ambient specifies the groups of the ambients that can sense that ambient context information, the groups of the ambients that can share that information with a third party, and the groups that third party must belong to. In this way, the flow of context information can be controlled to avoid accidental disclosure of private context information. Type annotations are added to the syntax of CCA in Section 3 and to its semantics in Section 4.
- A formalisation of the proposed type system using typing rules (Section 5.2); only processes that can be typed using these rules are *well-typed*. These rules are used to check statically (i.e. at compile time) the well-typedness of processes.
- The subject reduction property of the proposed type system is formally established (Section 6). This property states that a well-typed process can only reduce to well-typed processes; i.e. well-typedness is preserved by the reduction relation.
- The type soundness (aka safety) property is also formally established (Section 7); therefore it is guaranteed that well-typed processes do not violate the privacy requirements of any ambient in the system, nor give rise to run-time errors during reduction.
- The pragmatics of the proposed type system is illustrated using a case study of an infostation-based mobile communication (IMC) system where the identity and the location information of the sender must not be disclosed (Section 8). The privacy requirements of this system are captured using the type system (Section 8.1). The full specification of the IMC system in CCA is formally proved to be well-typed (Section 8.2). Finally the *reliability* of the IMC system is demonstrated through simulations in ccaPL (the CCA interpreter) which show that the users can communicate anonymously without the risk of revealing their location information (Section 8.3).

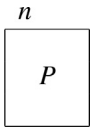
2. Motivation and examples

This section illustrates how the concept of *privacy type* proposed in this paper can be used to capture and analyse privacy requirements in ubicomp. Smart homes are a typical example of ubicomp systems where household furniture and appliances interact with each other via a home network to provide the inhabitants personalised services and comfort. The main services a smart home can provide include the intelligent climate control service that controls the heating, ventilation and air-conditioning (HVAC) system to assure good thermal comfort and appropriate indoor air quality; the lighting control service to help save energy; the multi-room audi-visual entertainment service; and the security service which controls the CCTV cameras installed in the smart home. At the tip of a smart phone and miles away from home, one can switch on or off light in the living room; access live CCTV cameras video footage or simply be notified automatically of any intrusion into her property; or attend remotely to an elderly person under her care. Protecting the privacy of the users in such complex and heterogeneous systems is a difficult task.

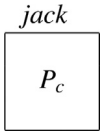
Consider the following scenario. Jack lives with his brother Peter and their grandad Jacob in a smart home. Jacob is an elderly person and suffers from dementia; he often forgets taking his medication and so needs assistance most of the time. The smart home has a number of rooms, each equipped with a location system that enables it to determine the identity of anyone present in the room in order to provide personalised services. The rooms are connected to a home network and to the Internet; and can share with other devices in the network the identity of anyone present in the room. This makes it easy to Jack and Peter to monitor at any time the location of Jacob. However, Jack and Peter would like their own location information to be private; especially not to be disclosed over the Internet. To enforce this privacy requirement in the smart home system, the flow of context information must be controlled across the home network to ensure that private information are not accidentally disclosed. The following shows how this can be done using the proposed privacy type system in the Calculus of Context-aware Ambients (CCA).

System Modelling in CCA

In CCA, the concept of ambient is used to conceptualise any entity in a ubicomp system. An ambient has a name and its behaviour is modelled as a process. For example, an ambient of name n and behaviour P is denoted by the process $n[P]$ (textual representation). This ambient can also be represented graphically as follows:



It is assumed that the user carries a mobile device that can be used to identify her. Depending on the location determination system in the rooms, this may be as simple as an RFID tag, or the blue-tooth of the user's smart phone. Any user of a smart home can be represented as an ambient, e.g.



An ambient may contain other ambients which are its *child ambients*. A parent ambient represents the *location* of its child ambients. Ambients that have the same parent are called *sibling* ambients. Such a tree structure can be used to model any ubicomp system. Suppose Jack's smart home has one living room with a TV set, one kitchen, one bath room and three bedrooms. Fig. 1 depicts the smart home model in CCA when Jack is in the living room, Peter in the Kitchen and Jacob in his bedroom. The corresponding textual representation is the following process, where the symbol '|' denotes the parallel composition of processes and each P_i describes