Information and Software Technology 51 (2009) 993–1009

Contents lists available at ScienceDirect

Information and Software Technology

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

Requirements-based Access Control Analysis and Policy Specification (ReCAPS)

Qingfeng He [a,*,1], Annie I. Antón [b]

[a] ABB Inc., US Corporate Research, 940 Main Campus Dr. Suite 300, Raleigh, NC 27606, USA
[b] Department of Computer Science, North Carolina State University, Raleigh, NC 27695-8206, USA

ARTICLE INFO

Article history:
Received 22 August 2008
Received in revised form 17 November 2008
Accepted 23 November 2008
Available online 10 January 2009

Keywords:
Requirements analysis
Security
Access control

ABSTRACT

Access control (AC) is a mechanism for achieving confidentiality and integrity in software systems. Access control policies (ACPs) express rules concerning who can access what information, and under what conditions. ACP specification is not an explicit part of the software development process and is often isolated from requirements analysis activities, leaving systems vulnerable to security breaches because policies are specified without ensuring compliance with system requirements. In this paper, we present the Requirements-based Access Control Analysis and Policy Specification (ReCAPS) method for deriving and specifying ACPs, and discuss three validation efforts. The method integrates policy specification into the software development process, ensures consistency across software artifacts, and provides prescriptive guidance for how to specify ACPs. It also improves the quality of requirements specifications and system designs by clarifying ambiguities and resolving conflicts across these artifacts during the analysis, making a significant step towards ensuring that policies are enforced in a manner consistent with a system's requirements specifications. To date, the method has been applied within the context of four operational systems. Additionally, we have conducted an empirical study to evaluate its usefulness and effectiveness. A software tool, the Security and Privacy Requirements Analysis Tool (SPRAT), was developed to support ReCAPS analysis activities.

© 2008 Elsevier B.V. All rights reserved.

1. Introduction

Data security and privacy is important in every software system, but is particularly vulnerable in critical infrastructure systems, such as medical, immigration, and financial information systems. Specifying correct policies to control users' access to a system and its resources is essential for protecting data security and privacy. Access control (AC) is a mechanism for achieving confidentiality and integrity in software systems to keep sensitive data private [1,2]. Throughout this paper, *confidentiality* means that information is not disclosed to unauthorized persons, processes or devices. *Integrity* means that unauthorized persons, processes or devices cannot modify information. *Privacy* implies that data is protected so that it is used only for authorized business purposes, based on legal requirements, corporate policies and end-user choices.

There are two major challenges in an access control system: (a) defining correct and complete policies to control users' access to the system and its resources, and (b) ensuring the resulting policies comply with the system requirements and high-level security/privacy policies. Proper access control analysis requires one to examine system requirements in tandem with organizational security and privacy policies to specify access control polices (ACPs). Defining correct and complete ACPs is both a conceptually and practically complex process because software systems can have many users performing various tasks and many resources that need to be protected [3,4]. Organizational complexity presents another challenge—it is difficult to identify and agree upon a common set of roles and associated permissions within an organization that may have hundreds of roles that must be considered.

ACP specification is not an explicit part of traditional software development processes and is often isolated from requirements analysis activities, leaving systems vulnerable to security breaches. Researchers have recognized the need to bridge the gap between requirements analysis and complex ACP specification [5]. Existing RE approaches (e.g., KAOS [6], i^* [7] and the analytical role-modeling framework [5]) provide limited support as we discuss herein. However, methodological support is needed to guide software and security engineers (referred as analysts in this paper) as they specify a system's ACPs. To this end, we have developed the Requirements-based Access Control Analysis and Policy Specification (ReCAPS) method [8] to integrate these activities, improve software quality and develop policy- and requirements-compliant systems. In this paper, we present the ReCAPS method, which

* Corresponding author. Tel.: +1 919 807 5709; fax: +1 919 856 2411.
E-mail addresses: heq@acm.org, qingfeng.he@us.abb.com (Q. He), aianton@ncsu.edu (A.I. Antón).

[1] This work was completed while the author was at North Carolina State University.

0950-5849/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2008.11.005

provides prescriptive guidance for specifying ACPs, improving the quality of a software system's artifacts (e.g., software documentation), and ensuring compliance between access policies and system requirements.

To date, the ReCAPS method has been applied within the context of four operational systems, two of which are discussed in this paper. Additionally, an empirical study was completed to evaluate its usefulness and effectiveness. Finally, a software tool, the Security and Privacy Requirements Analysis Tool (SPRAT), was developed to support ReCAPS analysis activities.

The remainder of the paper is structured as follows. Section 2 summarizes the most relevant work. Section 3 provides an overview of the ReCAPS method. Section 4 discusses application of the ReCAPS method within the context of two operational systems. Section 5 details an empirical study to evaluate the effectiveness and usefulness of the ReCAPS method. Section 6 briefly summarizes the SPRAT (Security and Privacy Requirements Analysis Tool) that supports the ReCAPS analysis activities. Finally, Section 7 provides a discussion and summarizes our plans for future work.

2. Background and related work

Requirements-based access control analysis and policy specification builds upon prior work in key areas: access control policies for security and in requirements engineering, as well as policy specification and software development. This section provides an overview of this related work.

2.1. Access control policies in security

An access control system is typically described in three ways: access control policies, access control models and access control mechanisms [2]. *Access control policies* define rules concerning who can access what information, and under what conditions [1]. These policies are enforced via a mechanism that mediates access requests and makes grant/deny decisions. The *access control mechanism* defines the low-level functions that implement the controls imposed by the policies. It must work as a reference monitor [2], a trusted component intercepting each and every request to the system. *Access control models* provide a formal representation of an access control system. They provide ways to reason about the policies they support and prove the security properties of the access control system. Access control models provide a level of abstraction between policies and mechanisms, enabling the design of implementation mechanisms to enforce multiple policies in various computing environments.

ACPs can be broadly grouped into three main policy categories: Discretionary Access Control (DAC), Mandatory Access Control (MAC), and Role-Based Access Control (RBAC). *DAC policies* enforce access control based on the identity of the requestor and the explicit rules specifying who can or cannot perform specific actions on specific objects. An example DAC policy describes user Alice is allowed to read file A. Early discretionary access control models, such as the access control matrix model [9] and the HRU (Harrison–Ruzzo–Ullman) model [10], provide a basic framework for describing DAC policies. In these models, it is the users' discretion to pass their privileges on to other users, leaving DAC policies vulnerable to Trojan Horse attacks [2].

MAC policies enforce access control based on the security classifications of subjects and objects. For example, the lattice-based multilevel security policy [11], policies represented by the Bell–LaPadula model [12,13] and the Biba model [14] are MAC policies. A specific example of a lattice-based security policy could be as follows: the sensitivity of all information is classified as either Top Secret, Secret, Confidential, or Unclassified. These four levels form a lattice. A user Bob who possesses a classification of Secret can access information that is classified Secret, Confidential or Unclassified, but not Top Secret. MAC policies prevent indirect information leakages (e.g., Trojan Horse attacks), but are still vulnerable to covert channel attacks [2,15].

RBAC policies employ roles to simplify authorization management for enforcing enterprise-specific security policies [1,16]. The RBAC model is an alternative to traditional DAC and MAC models and has received increased attention in commercial applications, such as the Oracle 9i DBMS [17]. RBAC is now an American National Standard (ANSI INCITS 359-2004).

Instead of considering ACP specification from a holistic, real-systems perspective as we advocate in this paper, ACP specification research to date has been narrow in its focus (e.g., uniform or flexible ways to specify ACPs [18], specifying ACPs for XML documents [19], etc.). There are few reported methods and experiences that [4] describe ACP specification in real software systems. In the RBAC literature, researchers are investigating role engineering, the process of defining roles and privileges as well as assigning privileges to roles [20]. Several role-engineering approaches employ requirements engineering concepts. For example, Fernandez and Hawkins suggest deriving the needed rights for roles from use cases [21]. Neumann and Strembeck propose a scenario-driven approach for engineering functional roles in RBAC [22]. Role engineering is specific to RBAC, whereas the ReCAPS method is a more general approach for specifying ACPs.

2.2. Elements of access control policies

An access control *policy* is comprised of a set of access control rules. A *rule* can have various modes (e.g., allow/deny/oblige/refrain). Since allow and deny rules are the most common ones, this paper focuses on these two kinds of rules. *Allow* rules authorize a subject to access a particular object. *Deny* rules explicitly prohibit a subject from accessing a particular object. When a subject requests to perform an action on an object, the corresponding rules are evaluated by the enforcement engine for that request. A typical access control *rule* is expressed as a 3-tuple ⟨*subject*, *object*, *action*⟩, such that a *subject* can perform some *action* on an *object* [23]. A *subject* is a user or a program agent, or any entity that may access objects. An *object* is a data field, a table, a procedure, an application or any entity to which access is restricted. An *action* is a simple operation (e.g., read or write) or an abstract operation (e.g., deposit or withdraw). The ReCAPS method extends the typical AC rule 3-tuple to include conditions and obligations as we now discuss.

An ACP may express *conditions* that must be satisfied before an access request can be granted. For example, in healthcare applications, the location from which the access request originates might affect the grant/deny decision [24]. If an access request is from the emergency room, then (according to the requirements) the request may need to be granted. In this case, we can specify *the location of the request is emergency room* as a condition for the AC rule. *Obligations* [25] are actions that must be fulfilled if a request to access an object is granted. For example, consider the following policy statement: *require affiliates to destroy customer data after service is completed*. In this case, "destroy customer data" is an obligation that must be satisfied by affiliates. Obligation-based security policies can be enforced if they can be completely resolved within an atomic execution [26]. If the obligation is not an immediate action (e.g., it is a task that will be executed in the future), monitoring and auditing its execution might be sufficient for enforcement [25].

In requirements specification, we are concerned with the *actions* for which each actor (*subject*) is responsible, and the conditions under which each action can occur (constraints and *preconditions*). Thus, in the ReCAPS method, AC elements may be

Download English Version:

https://daneshyari.com/en/article/549901

Download Persian Version:

https://daneshyari.com/article/549901

Daneshyari.com