COMPUTERS & SECURITY 62 (2016) 317–327

Available online at www.sciencedirect.com

ScienceDirect

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

Computers & Security

Access control lists in password capability environments

Lanfranco Lopriore *

Dipartimento di Ingegneria dell'Informazione, Università di Pisa, via G. Caruso 16, 56126 Pisa, Italy

ARTICLE INFO

Article history:
Received 16 February 2016
Received in revised form 15 July 2016
Accepted 17 August 2016
Available online 24 August 2016

Keywords:
Access control list
Access right
Domain
One-way function
Password capability
Protection

ABSTRACT

With reference to a protection system featuring active subjects that attempt to access passive, typed objects, we propose a set of mechanisms supporting the distribution, verification, review and revocation of access privileges. In our approach, a protection domain is a collection of access rights for the protected objects. An access control list is associated with each object to specify the access rights in each domain. Objects are grouped into clusters. To access the objects in a given cluster, a subject presents a gate referencing this cluster. The gate is a form of password capability that identifies one or more domains. The gate grants the access rights specified for these domains by the access control lists of the objects in the cluster. A subject that holds a gate and is aimed at distributing the access privileges in this gate in restricted form can reduce the gate to eliminate domains; the gate reduction procedure requires no intervention of the protection system. A small set of protection primitives allows subjects to manage objects and access control lists. Forms of revocation of access permissions are supported, at both levels of gates and access control lists.

1. Introduction

In a classic protection system paradigm, a set of active *subjects* (users, processes) attempts to access a set of passive, typed entities called *objects* (Lopriore, 2015b; Zhang et al., 2005). The type of a given object states the set of the *operations* that can be executed on this object, and the *access rights* that are necessary to accomplish each operation successfully. A subject aimed at executing an operation on a given object must possess the access rights required by this operation, as is stated by the object type. A *protection domain* is a collection of access rights for the protected objects. At any given time, each subject is executed in a protection domain, and can take advantage of the access rights included in this domain to operate on the protected objects. In the course of execution, the subject can change domain, according to the access right requirements of the different execution phases.

1.1. Capabilities and access control lists

Let b_0, b_1,… be a set of objects, and d_0, d_1,… be a set of protection domains. In a well-known representation, the state of the protection system takes the form of a matrix, called the *access matrix*, featuring a row for each domain and a column for each object (Lampson, 1974; Samarati and De Capitani Di Vimercati, 2001). Element $AM_{i,j}$ of the access matrix, corresponding to row i and column j, contains the specification of the access rights included in domain d_i for object b_j. An important aspect is that a domain can also be a protected object, and in this case the domain corresponds to both a row and a column of the access matrix.

The access matrix tends to be large and sparse. Most elements of the access matrix are likely to be empty, so storage in matrix form is usually inadequate. Two alternative, well-known approaches to represent the access matrix are *capability lists* and *access control lists* (Sandhu and Samarati, 1994). In the

* *E-mail address:* l.lopriore@iet.unipi.it.
http://dx.doi.org/10.1016/j.cose.2016.08.005
0167-4048/

capability list approach, the access matrix is stored by rows. A capability list is a collection of capabilities, which is associated with each domain, and specifies the access rights included in this domain. A *capability* is a pair (b, ar), where ar specifies a set of access rights for object b (Levy, 1984). In the access control list approach, the access matrix is stored by columns. An access control list ACL_b is associated with each given object b. Each element of ACL_b has the form (d, ar), and specifies the set ar of access rights for b that is included in domain d.

Capabilities need to be segregated into protected memory regions (de Vivo et al., 1995; Lopriore, 2012; Wilkes, 1982). This is necessary to prevent a subject that holds a given capability from modifying this capability, e.g. the access right field, to obtain an undue amplification of access rights, or the object identifier field, to forge a capability referencing a different object. In a segmented memory environment, the capability segregation problem can be solved by reserving *ad hoc* memory segments for capability storage, the *capability segments* (in contrast, the *data segments* will be reserved for storage of ordinary information items) (England, 1974; Klein et al., 2009). Each capability segment can contain a capability list. Capability segments can only be accessed in a strictly controlled fashion, by executing special machine instructions, the *capability instructions*. In an alternative approach, in a system featuring a form of tagged memory, a tag can be associated with each memory cell to specify whether this cell contains a capability or an ordinary information item (Carter et al., 1994; Kwon et al., 2013; Neumann and Feiertag, 2003; Watson et al., 2015). A cell tagged to contain a capability can be accessed only by using the capability instructions.

Password capabilities are an alternative, effective solution to the capability segregation problem (Chase et al., 1992; Heiser et al., 1998; Lopriore, 2015a; Pose, 2001). In a password capability environment, one or more passwords are associated with each protected object. Each password corresponds to a set of access rights for this object. A password capability is a pair (b, w), where b identifies an object, and w is a password. If w matches one of the passwords associated with object b, the password capability grants the corresponding access rights on b. If passwords are large and sparse, the probability that a malevolent subject guesses a valid password to forge a password capability is vanishingly low (Castro et al., 2008). It follows that password capabilities can be mixed in memory with ordinary information items, and can be manipulated by using standard machine instructions.

In this paper, we propose the organization of a protection system that takes advantage of both access control lists, for the protection of ordinary objects, and a form of password capability, called *gate*, for the protection of domains.

1.2. Clusters

We group objects into *clusters*. For each object, the cluster includes the corresponding access control list. To access the objects in a given cluster, a subject must demonstrate the right to take advantage of all, or part of, the domains specified by the access control lists in that cluster. This is a problem of certified identity whose solution is the main contribution of this paper, and is based on gates. A gate referencing a given cluster identifies one or more domains in this cluster. A subject that possesses the gate can access the objects in the cluster with the access rights specified for these domains by the access control lists of these objects.

The rest of this paper is organized as follows. Section 2 introduces the gate concept with special reference to the relation existing between a gate for a given cluster and a *base gate* generated when the cluster is created. Gate validation and reduction, to eliminate access permissions, are analyzed in special depth. Section 3 presents a small set of primitives, the *protection primitives*, which allows subjects to access the objects and to manage their access control lists. Section 4 considers the problems related to the review of access permissions, with special reference to gate revocation. Section 5 discusses the motivations of the gate paradigm in reference to a number of important viewpoints, which include fraudulent gate forging, gate equivalency, and the grouping of objects into clusters. A few considerations concerning performance are presented, in both terms of the memory requirements for gate storage and the execution times for gate validation. Section 6 discusses the relation of our work to previous work. Section 7 gives concluding remarks.

2. Gates

Let C be a cluster, and let b_0, b_1,... be the protected objects included in C. The cluster implements an object protection technique based on domains and access control lists. Let d_0, d_1, ..., d_{n-1} be the protection domains for b_0, b_1,.... An access control list ACL_b is associated with each given object b. Each element of ACL_b has the form (d,ar), where ar denotes the set of access rights for b that is included in domain d.

A subject S, aimed at accessing cluster C to execute operation *op* on object b, must present a gate referencing C. This gate gives the right to take advantage of one or more domains in C. The access control list ACL_b of b specifies a set of access rights for each of these domains. The access will be accomplished successfully only if the *union* of these sets of access rights includes the access rights that are required to execute *op*.

The gate concept is a variant of the classical password capability concept. A gate G for cluster C has the form $G = (W, R)$. Quantity W is a password; as will be made clear shortly, this password univocally identifies the cluster. Quantity R, called the *domain selector*, identifies one or more domains in C, as follows. R is partitioned into $n - 1$ subfields, called *primary selectors* and denoted by r. Thus, $R = (r_{n-2}, r_{n-3}, \ldots, r_0)$. The size of a primary selector is n bits, one bit for each domain (the least significant bit, bit 0, corresponds to the first domain, d_0). For each bit that is asserted in a primary selector, the corresponding domain is eliminated from the gate. It follows that the domains referenced by G are those corresponding to cleared bits in quantity $r_{n-2} \vee r_{n-3} \vee \ldots \vee r_0$. A primary selector whose value is 0 is called *null*. All null primary selectors are placed in the most significant positions of domain selector R, at the highest order numbers. In a given gate, if all primary selectors are null, $R = 0$ and the gate references all the domains in the corresponding cluster. We wish to point out that $n - 1$ primary selectors allow us to specify any combination of active