

Access Control in Distributed Object Systems: Problems With Access Control Lists

S.V.NAGARAJ

*Infosys Technologies Ltd,
Bangalore 561 229, India
subramanyamv_n@infy.com*

Abstract

In this paper we look at some drawbacks of access control lists (ACL's) that are in wide use.

Keywords: Access Control, Access Control List (ACL), Distributed Systems.

1. Introduction

Providing satisfactory access control mechanisms for distributed object systems is a challenge, as the characteristics of these mechanisms are not well understood. Distributed object technology has progressed in recent times but the lack of practically useful security mechanisms for access control hinders their deployment in application domains. CORBA from the Object Management Group is one of the most popular distributed object technologies. Despite its popularity, commercial product releases that fully confirm to its security service specification are just beginning to emerge.

Access control in CORBA or any other distributed system becomes complicated because of the fact that a target object upon receiving an invocation from an authorized client may have to become the client of other objects in the system (in order to form a response to the original request). So the basic question boils down to one of delegation.

2. Access Control Lists

Distributed computing has become very popular because of the widespread usage of the Internet. Controlling access to distributed resources such as databases, a printer etc. becomes important.

A security mechanism that has its origin in operating systems is the Access Control List. It specifies the access rights a principal has on an object (or resource). An example is an entry such as "User A can Read file X". Often such lists are distributed throughout a system instead of being confined to one physical location (because a single location could become a bottleneck). ACL's are being used in distributed systems due to their simplicity. There is a vast literature on ACL's due to their origin in operating systems. In [1]; some reasons why ACL's are inadequate for the security of distributed systems are given:

Authentication: In an operating system, the identity of a principal is well known. However this is not the case in a distributed system where some form of authentication has to be performed before a decision to grant access is made. Authentication is often accomplished via a username/password mechanism. It turns out that simple password-based protocols are inadequate in networked computing environments. Some recently developed authentication mechanisms include: one-time password and centralized ticket-based systems such as Kerberos. These systems create the need for an authentication server (and frequent communications with it).

Delegation: It is necessary for the scalability of a distributed system. It enables the decentralization of administrative tasks. Existing distributed system security mechanisms usually delegate directly to a "certified entity". In such systems, policy (or authorizations) may only be specified at the last step in the delegation chain, most commonly in the form of an ACL. This means that high-level administrative authorities cannot directly specify overall security policy; rather, all they can do is "certify" lower-level authorities. This authorization structure can easily lead to inconsistencies among locally specified sub-policies.

Expressibility and Extensibility: A generic security mechanism must be able to handle new and diverse conditions and restrictions. The traditional ACL based approach has not provided sufficient expressibility or extensibility. Thus many security policy elements that are not directly expressible in ACL form must be hard-coded into applications. This means that changes in security policy often require reconfiguration, or even rewriting of applications.

Local trust policy: The number of administrative entities in a distributed system can be quite large. Each of these entities must have a different trust model for different users and other entities. For example, system A may trust system B to authenticate its users correctly, but not system C; on the other hand, system B may trust system C. It follows that the security mechanism should not enforce uniform and implicit policies and trust relations.

Blaze et al. [1] argue that these points constitute a forceful argument that Authenticode, X.509, and, generally, the use of identity-based public-key systems in conjunction with ACL's are inadequate solutions to distributed (and programmable) system-security problems. Even modern ACL-based systems such as the Open Group's Distributed Computing Environment (DCE) seem to fall some-what short of satisfyingly addressing the issues of extensibility, expressibility and delegation, though there has been some progress in this direction. In practice, one finds to his/her dismay that insecure, inadequate or non-scalable authentication mechanisms such as username/password, one-time password, and hardware token authentication are being used in conjunction with ACL's.

Many policy problems are left unsolved by the "binary" authentication model widely used in Web security (and elsewhere): a particular Certificate Authority grants access on the condition that the requesting principal has a certificate. In [1] this is called a "binary" authentication model because it means an essentially "all-or-nothing" access. This may be sufficient in some situations (e.g. when read-only access to a Web page is the only decision that needs to be made) but this approach is neither scalable nor extensible. These problematic mechanisms are in use mainly because of the lack of alternatives that are better suited to distributed systems.

In a large object oriented system the number of objects and methods may be enormous. An attempt to associate separate access control attributes with every method of every object will cause proliferation of access control information that could result in a loss of understandability and manageability. This problem becomes worse when ACL's are used as attributes; each ACL can contain many entries (including those appear contradictory) whose combined effect can only be understood by applying complex precedence and ordering rules (see [2]). DCE uses ACL's to express authorization policy. File systems often use permission bits, owners, and groups to describe the permitted modes of access by authorized users. In these systems it is not easy to answer basic questions about the policy, such as "Which files is users X permitted to read".

In the traditional setting authorization and access control decisions are based on the use of ACL's implemented as database entries in which each resource is followed by a list of subjects who are allowed to access the resource. ACL's offer a solution suitable for centralized client/server systems but they do not scale well in distributed systems. Replicating the ACL database in multiple distributed servers will lead to difficult update and synchronization issues. ACL databases have to be protected from unauthorized changes. It is not easy to support anonymity and privacy with ACL based approaches as they favor the usage of globally unique names mapped to access rights. Modern access control approaches based on authorization certificates overcome many of the above drawbacks but introduce some new ones.

3. Conclusion

Access control lists are being used widely due to their simplicity but the emerging applications require methods that pose fewer problems.

References

- [1] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. "The Role of Trust Management in Distributed Systems Security", in *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, eds. Jan Vitek and Christian Jensen, Springer Verlag, 1999.
- [2] Tally G et al, "A Scalable Approach to Access Control in Distributed Object Systems", NAI Advanced Security Research Journal, Vol 1, No 1, 1998 pp 59-73.