Contractual Access Control*

Babak Sadighi Firozabadi[1] and Marek Sergot[2]

[1] Swedish Institute of Computer Science (SICS)
babak@sics.se
[2] Imperial College of Science, Technology and Medicine
mjs@doc.ic.ac.uk

Abstract. In this position paper we discuss the issue of enforcing access policies in distributed environments where there is no central system designer/administrator, and consequently no guarantee that policies will be properly implemented by all components of the system. We argue that existing access control models, which are based on the concepts of permission and prohibition, need to be extended with the concept of entitlement. Entitlement to access a resource means not only that the access is permitted but also that the controller of the resource is obliged to grant the access when it is requested. An obligation to grant the access however does not guarantee that it will be granted: agents are capable of violating their obligations. In the proposed approach we discuss a *Community Regulation Server* that not only reasons about access permissions and obligations, but also updates the normative state of a community according to the contractual performance of its interacting agents.

1 Introduction

One of the problems that underlies emerging computing technologies with highly decentralised structures, such as Peer-to-Peer, Grid Computing and Ad-Hoc Networks, is how to manage access policies to disparate resources that are not under the control of a single system designer/administrator.

In these systems, a number of individuals and/or institutions interact in a collaborative environment to create a *Virtual Community* (VC), shaped and organised according to a set of rules and policies that define how its resources can be shared among its members. A VC is also sometimes called a *Virtual Organisation*, as in [FKT01].

A VC is usually a composition of heterogenous and independently designed subsystems with no centrally controlled enforcement of the community policies. Consequently, there is no guarantee that community policies will be followed as they are prescribed: members of a VC may fail to, or choose not to, comply with the rules of the VC. If there is no way of practical (physical) enforcement of community policies then it would be useful to have a *normative control mechanism* for their *soft enforcement*. By soft enforcement we mean that even if VC

* This work is supported by the Swedish Agency for Innovation Systems (Vinnova) as part of the Policy Based Management Project.

members are practically able to avoid complying with the community policies, they can still be subject to sanctioning and remedial action as the consequence of their behaviour.

1.1 Why traditional access control models are not sufficient

Existing access control models are originally designed for distributed applications operating on client-server architectures. A basic assumption for these models is that there is a centrally supervised management of the entire system such that access policies will be updated and enforced as they are prescribed. For example, when a new user is introduced then its identity and its access permissions will be added to the access control lists of the provided services. Given this assumption, the policy enforcement component is trusted always to comply with the prescribed policy (unless it develops faults). The question of what to do when a resource provider deliberately fails to comply with the system's policies does not arise.

In contrast, in a system of heterogeneous and independently designed subsystems this assumption no longer holds. Consider this example: agents a_1 and a_2 are participating in an application with no central enforcement mechanism. a_1 wants to access data d that is stored remotely with agent a_2. Upon an access request from a_1, a_2 has to decide whether to grant the access to a_1 or not. There are several possible cases:

– a_1 is permitted to access the resource, but there is no obligation on a_2 to grant that access. a_2 will not violate any policy regardless of whether it grants or denies the access.
– a_1 is not only permitted to access the resource, but is also *entitled* to it. This means that a_2 has an *obligation* to grant the access whenever a_1 requests it. A typical scenario is when a_1 is the owner of d and a_2 is the storage service provider for d. Another example is where d is owned by another agent a_3 and a_3 has authorised (or rather, entitled) a_1 to access d on a_2. a_2 violates the policy if it fails to grant access to the entitled agent a_1.
– a_1 has no permission to access d, and so a_2 is forbidden to grant the access. Note that a_2 may have the practical possibility to give access to a_1 even if it is not permitted to do so.

1.2 Entitlement

In the literature on computer security, and in computer science generally, the terms 'right' and 'permission' (and 'privilege', and others) are used interchangeably. We have chosen to use the term 'entitlement' to emphasise that we have in mind a concept stronger than mere permission.

Suppose that in a VC there is a community policy that member X makes available 15GB of its disk storage for use by other members (under certain other specified terms which we ignore for the sake of the example). Suppose that X also has its own *local* policies, to the effect that members from domain D will not be granted access to its resources (because of some previous experiences with domain D, for example), and files containing gif images will not be stored (because of the danger of storing pornographic materials, say). Suppose now that one of the members of the VC, Y, attempts to store a file on X's disks. X denies the access because the file contains gif images. Would we say that X has thereby violated (failed to comply with) the community policies operative in VC? If the answer is 'no' then the community policy is merely that Y has *permission* to store files on X's disks. If the answer is 'yes', then Y is not only permitted to store files on X's disks but is *entitled* to do so. The policy language used to express the community policies must be capable of making the distinction.

With this distinction a server might have a local policy to the effect that access to its resource will be granted to any permitted member (including entitled ones) between certain hours, but outside those hours access will be granted only to those who are entitled.

In general, a member X of some VC will be subject to (at least) two separate sets of policies: the community policies operative in VC, and the local policies defined for X. In [PWFK02] it is assumed that the community policy in the VC must always be consistent with the local policy defined for each resource. But how could this be ensured, in a system that is composed of independently designed sub-units? It is possible to imagine applications where the assumption might hold up, for instance, when all the independent resource providers either formulate their local policies to be consistent with community policies, or specify in their own local enforcement mechanisms that in case of conflict between community policies and local policies, the community policies will take precedence. But such a remarkable degree of co-operation between all the resource providers will not be so common. Rather, it is to be expected that local policies will conflict in certain circumstances with community policies, sometimes because the resource provider is looking for a 'free ride', but also because there are some detailed local considerations (such as bad previous experiences with domain D) which would lead a resource provider to choose to violate community policies from time to time. Or suppose that the community policies require that X makes available its disk storage between 6 a.m. and midnight, but X has a local policy which restricts access to the hours of 8 p.m. and 11 p.m. How could this happen? Well, leaving aside the possibility that X is out for a free ride, perhaps when X joined the VC it was expected that accesses outside those hours would be very infrequent and therefore not worth worrying about. It may also be that a resource provider belongs to more than one VC, and finds itself in circumstances where it cannot comply with the community policies of both.

Our argument is that when dealing with access control in networks of heterogeneous systems without centralised control, the usual concepts of permission and prohibition are inadequate, and must be extended with (at least) one additional concept which we are calling *entitlement*. The question is whether a request, e.g, an access request, from an agent creates an *obligation* on the controlling agent to grant this access. There are then two further subsidiary questions. (1) What if the controlling agent ignores the request and consequently violates its obligation? (2) Under what circumstances may one agent create or pass on entitlements to another?

2 The Approach

The idea of extending access control mechanisms to deal with entitlement relations is useful for any computing environment with heterogenous and independently managed subsystems. However, we intend to investigate the idea in the setting of Grid computing [FKT01,FKNT02] because it provides a suitable infrastructure.

Any member of a VC can be a service provider as well as a service consumer. As a service provider, a member commits itself to provide a service under certain terms and conditions to other VC members. These terms and conditions may either be specified in the community policies of the VC, or they may be negotiated by each member separately as part of their conditions for participating in an interaction with other members, or in some combination of the two.

A VC member entering into an agreement agrees not only on what it is permitted to do, but also what it is obliged to do (what its duties are to other members, and what their entitlements are). Hence, to enforce access permissions according to an agreement we need to reason not only about who is permitted or prohibited to access a service and in what conditions, but also under what conditions a service provider is obliged to give access to its service.

In [PWFK02], the authors describe a *Community Authorisation Service* (CAS) server as a trusted third party component that is responsible for managing access policies to the community's resources. CAS keeps track of the permissions that the community grants to each individual member according to the agreements obtaining between the community and its members. Based on the access state of the VC CAS issues capabilities (signed attribute certificates) that the members can provide to a resource server as credentials with their access requests. The service provider, or its resource server, will first check whether the request should be granted according to its own local policy and then check the capability issued by CAS before it grants the access.

Here, we suggest an extended CAS that we call the Community Regulation Server (CRS) which is responsible for updating the normative state of a VC as well as the soft enforcement of its policies. By normative state of a VC we mean the access permissions of the VC members, and their obligations to provide access to their services. The normative state of a VC is derived from its community policies, the agreements entered into by the VC and its members, and the management structure of the VC, including the positions and roles of the members. By soft enforcement of VC policies we mean triggering remedial actions and perhaps updating the normative state of the VC in case of violations. Moreover, the CRS will update changes in the management structure of the VC, that is to say, the delegations of management powers among the VC members as described in [FSB01].

3 Open issues

In order to realise the approach above there are a number of subsidiary issues that need to be addressed. Here we sketch the main requirements.

3.1 Accountability

Any regulative system requires that the entities subject to its norms (its policies and rules) can be uniquely identified in order for them to be accountable for their behaviour.

Ensuring accountability in applications where there is no identity requirement of any kind is not possible. These application types cannot be extended with a soft control mechanism and fall outside the scope of this work. Applications that require a pseudonym identity for their users will give a weak handle for accountability ensurance. The main problem with such applications is that a user can have several identities in the system at the same time. It can also disappear after a misbehaviour and reappear with another identity. One way of achieving accountability in such systems is to introduce reputation and scoring mechanisms which give the users economic incentive to retain the same identity for a longer period of time, e.g., by increasing the quality of service for those who have been lawful citizens of the virtual community for a longer period of time.

Most applications in which the users are involved in some kind of enterprise activity will require that users participate in the business with their real identities. For example, users must often identify themselves with their public key certificates which guarantees binding between their public key and their legal identities. These system can still be very dynamic in the sense that the set of users, resources, and the relations between them change frequently.

3.2 Delegation of Entitlements

We have an existing framework [FSB01,FS02,BDF02] for the delegation of authorisations and authorities via attribute certificates, together with an implementation of the reasoning required to determine, given a record of time-stamped certificates and revocations, what are the privileges (authorisations, delegation authorities) of a given entity of the system at any time. We see no difficulty in extending the existing framework to deal with delegation of entitlements as just another kind of privilege.

There is one additional factor. In human societies it is commonplace to find organisational structures in which an individual can create an obligation on another individual, possibly its superior, to exercise one of its privileges, for example to delegate some of its power. In the present context this would correspond to an entitlement of X to oblige Y to delegate one of Y's privileges. Whether this is an important or useful feature in a computational virtual organisation remains to be investigated. Our view is that it will turn out to be important.

3.3 Violation Detection and Complaint Procedure

A basic component of the infrastructure is to monitor that services are provided in accordance with the community policies and individual agreements made between members, and to detect violations (non-compliance) as they occur. The monitoring can be active, or it can be left to the members of the VC to initiate complaint proceedings against other members when agreements are unfulfilled. Here, we need to devise protocols, and associated cryptographic mechanisms, for ensuring that proper evidence is collected of both the actions and also the lack of actions of the agents.

There is a need for designing security (cryptographic) protocols to prevent false claims by agents. This is a question of guaranteeing evidence of actions (or lack of actions) on both the requester's and the service provider's side. For example, it should not be possible for a requester to claim without justification that its request was not answered properly, and it should not be possible for the service provider to claim that it has fulfilled an obligation if it has not done so.

We believe that, as a starting point, existing cryptographic protocols can be adapted for this purpose.

3.4 Sanctioning mechanisms

It is necessary to devise effective sanctioning mechanisms in order both to encourage agents to comply with the rules of the VC and fulfill their obligations, and to provide implementable sanctions in cases where members fail, or choose not, to comply. Sanction mechanisms can be quite simple: temporary suspension of entitlements and privileges, for example, is easily implemented, as is decrease in the level of quality of service provided. More elaborate forms of sanctioning, such as the use of 'marginal accounts' or 'bonds' can also be devised. A systematic exploration of the possibilities and their effectiveness remains to be done.

References

[BDF02] Olav Bandmann, Mads Dam, and B. Sadighi Firozabadi. Constrained Delegations. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 131–140, 2002.

[FKNT02] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration. http://www.globus.org/research/papers/ogsa.pdf, January 2002.

[FKT01] Ian Foster, Carl Kesselman, and Steven Tuecke. The Anatomy of the Grid – Enabling Scalable Virtual Organisations. *International Journal of Supercomputer Applications*, 15(3), 2001.

[FS02] B. Sadighi Firozabadi and Marek Sergot. Revocation Schemes for Delegated Authorities. In *Proceedings of the Third International Workshop on Policies for Distributed Systems and Networks*, pages 210–213, Monterey, California, USA, June 2002. IEEE.

[FSB01] B. Sadighi Firozabadi, M. Sergot, and O. Bandmann. Using Authority Certificates to Create Management Structures, April 2001. To appear in Proceedings of the 9th International Workshop on Security Protocols, Cambridge, UK.

[PWFK02] L. Pearlman, V. Welch, I. Foster, and C. Kesselman. A Community Authorisation Service for Group Collaboration. In *Proceedings of the Third International Workshop on Policies for Distributed Systems and Networks*, pages 50–59, Monterey, California, USA, June 2002. IEEE.