Policy-Driven Access Control over a Distributed Firewall Architecture

Theo Dimitrakos[1*], Ivan Djordjevic[2*], Brian Matthews[1], Juan Bicarregui[1], Chris Phillips[2]
[1]Central Laboratory of the Research Councils, Rutherford Appleton Laboratory, OX11 0QX, UK
[2]Department of Electronic Engineering, Queen Mary, University of London, Mile End Road, London E1 4NS, UK

Abstract

Motivated by a Grid based scientific application, where a dynamic collection of individuals and institutions are required to share resources to achieve certain goals, we propose the synthesis of two lines of research. The first line is Policy-Driven Access Control which treats policies as first-class objects that can be negotiated and tailored to particular roles. The second line is Distributed Firewalls that provide a dynamic and distributed security infrastructure bringing together peer-to-peer collaboration and hierarchical administration. Through this fusion we expect to deliver a scalable, dynamic and distributed method of setting up security infrastructures which has the benefits of allowing peer-to-peer collaboration, whilst maintaining the robustness and re-configurability of systems supplied by the central administration of the security policies.

The approach undertaken in this paper can also be understood as defining an interpretation of the deployment model of the Ponder policy framework on a distributed firewall architecture that supports Closed User Groups.

1 Introduction

Conventional firewalls are a well-established technology to protect organisations from unauthorised access and malicious attack. Conventional firewalls rely on a restricted topology and controlled entry points to regulate the flow of information in and out of an organisation. They restrict the use of resources within a domain to broadly two classes of users; insiders, who have wide-ranging authorisation, and outsiders who have very limited access to resources inside the domain.

However, in many areas of business, there is an increasing need to obtain secure communication for remote clients or employees situated outside of borders protected by the firewall.[1] There is also a need to support the formation of dynamic communities of interest to facilitate greater use of the Internet for electronic, social and business transactions, overcoming obstacles of distance, time and national boundaries. Furthermore, the recent emergence of peer-to-peer (P2P) technology has shown the power of direct communication between members of a group, unmediated by a server. In order to deliver true dynamically configurable groups, we need to move towards a P2P architecture, rather than using a traditional server-based architecture. Regarding that, new architecture for distributed firewalls which support dynamic and distributed method of setting up closed-user groups (CUGs), is proposed [2]. The scheme uses a combination of secure data transport technology and distributed firewall mechanisms [3] to provide a secure and robust infrastructure, where separate teams within a business or separate businesses can form virtual organisations for the purpose of e-business transactions. In this case, the intelligence of who is in the group (and suitable security and control information) needs to be shared among all member nodes using some form of message passing. However, some degree of centralisation is beneficial for administration, to oversee a consistent security regime, and to support asynchronous services.

Further, these dynamically configurable groups have different privileges and obligations from each other and from the original domains. Thus the authorisation policies that control access to the resource have to be different from each other, within the legal and business frameworks of the collaborating organisations. Traditionally, such policies have been embedded in the firewall itself, and cannot be changed dynamically to match fluid circumstances caused by these groups.

In this paper we propose to address these problems by bringing together two current lines of research. Firstly policy-driven access control, where policies are identified as first-class data objects in their own right, which can be negotiated and tailored to particular groups of clients. . Secondly, the CUG architecture, which has the benefits of facilitating P2P collaboration, whilst allowing to maintain the integrity of systems supplied by central administration of the security policies. The approach we investigate in this paper is based on an interpretation of a deployment model for the Ponder policy framework [4] on a distributed firewall architecture that supports Closed User Groups [2].

2 Motivating scenario

The concept of the Grid has emerged as a new approach to high-performance distributed computing infrastructure within the last five years [4]. Based on the Internet, the Grid seeks to extend the scope of distributed computing to encompass large-scale resource sharing including massive data-stores and high-performance networking,

*Corresponding editors: theo.dimitrakos@rl.ac.uk at CLRC and ivan.djordjevic@elec.qmul.ac.uk at QMUL.

[1] Recent research demonstrates that approximately 70% of attacks originate inside organisations and are made by inside users [1].

and shared use of computational resources, be they supercomputers or large networks of workstations. Currently the applications that are developing this infrastructure are large-scale scientific collaborations, such as NASA's Information Power Grid [6], and the European DataGrid project [7] led by the High-Energy Physics community. Such collaborations have a clear need for the collaborative use of resources, both data and computational, and established communities which can pool their resources for common goals, and thus form an ideal test bed to develop such technology. Tools are appearing to support the Grid concept, notably Globus [8], Condor [9] and Legion [10].

The Grid concept has been generalised to cover any *virtual organisation*, defined as any dynamic collection of individuals and institutions which are required to share resources to achieve certain goals [11]. Thus the Grid will have applications in commerce and industry, supporting distributed collaborative design and engineering, or distributed supply chains. In these contexts, the emerging Web Services technologies are likely to play a key role [12]. A key feature in the emerging Grid architecture will be the establishment of trust. Existing approaches to security within distributed systems are stretched by the extreme conditions imposed by the Grid, and significant effort has been undertaken in, for example, Globus to provide support for secure use of resources. However, trust thus needs to be established at all levels of the trust hierarchy

- *Authentication*: the establishment of identity of the user.
- *Policy based management*: the provision and deployment of rules and procedures for governing the choice in the behaviour of a managed system towards users who have been successfully authenticated.
- *Business Rules*: the business and legislative framework determining the nature of policies

We illustrate the trust requirements of the Grid via a representative example. Consider the following scenario (Figure 1) for the use of the Grid.

An engineer within organisation A wants to perform an analysis on a material. Rather than generating a new set of raw data to perform the analysis, she decides to save resources by looking for a pre-existing set of data. By accessing a portal at site B, she discovers a suitable data set held by a data archive C. The analytical tools are provided at university D within her Virtual Organisation. She initiates the analysis by passing the reference to the data set from B to D, which is then accessed by the analysis tools. D then determines that it does not have enough computational resource available, determines that a computer is available at different institution E, and delegates part of the job there. Finally, D completes the job and return the results to A. D also caches the results of the analysis locally and registers the fact that the pre-computed results are available with the portal B and the data provider C. However, the analysis has taken several hours, so the engineer has established a user proxy agent to represent her, collect the results, make payments as appropriate and close down the collaboration.

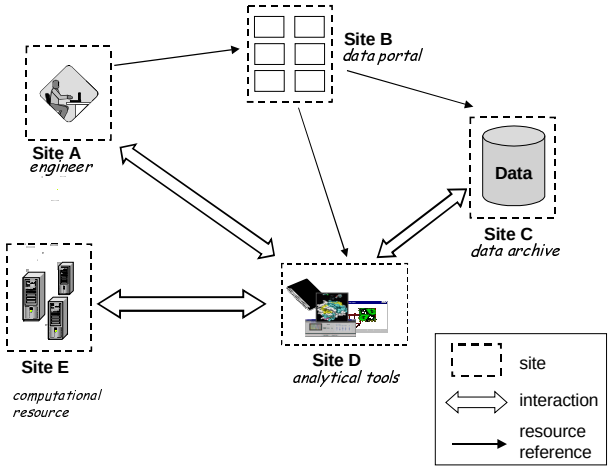

Figure 1: An e-Science scenario

This scenario highlights several features of the Grid that are relevant to trust.

Using the Grid requires the collaboration of resources which are controlled by different institutions. Each institution will have their own policies on access control and conditions of use. The allocation of resource is dynamic; the computational resource at D may not know until part way through the job that additional resource at E is required.

The user may have different identities in different domains. For example the engineer may have different Users Ids in her own institution, with the portal, and at the university.

Resources need to establish trust between each other (for example, the archive C and the portal B, the archive C and the analysis tools D, and the tools D and the compute resource E) on a P2P basis independently of the trust in the original user.

Resources may be called upon to participate in the task without previous knowledge of the other participants. For example, E may not know the identity of the engineer, or even the institution she comes from. Trust thus needs to be established on the fly so a mechanism needs to negotiate conditions of use, through the delegation of trust from one party to another.

Different trust conditions may be applied for different parts of the resource, including restrictions on data. For example, the data on the material lodged at archive C may have particular access conditions set by its originators (who may well be elsewhere than at C). A negotiation with respect to the use of the results needs to take place between the archive and the engineer before the data can be sent to the analysis tools.

Users and resources may delegate their identity to other parties – for example, the user at A creates a user proxy to act her role as the user in the process. The delegation of trust should follow the delegation of identity.

Users and resources may be located in different countries under different jurisdictions and thus with as a consequence subject to different legal and business requirements.

Resource usage tracking and charging may be involved. For example, portal B and computational resource E may be provided as commercial services. They need to identify the correct party to bill and establish trust in credentials (including possibly contacting third parties such as credit agencies) to be satisfied that bills will be honoured. These credentials need to be propagated by other agencies, for example, the analysis tools provider D, who may not be involved in the monetary transaction.

A suitable Grid architecture needs to be able to provide a security infrastructure which meets these requirements.

3 Policy-driven management

We view a **policy** as *"a rule that can be used to change the behaviour of a system"* (following [13]). A policy can be understood as a constraint on how a system should work or how people should use the system. Policies therefore govern the way a system works and as such they are relatively static with well-controlled procedures for change. Typically, a policy is defined as a high level, enterprise concept (in a traditional corporate environment this would be specified and understandable by high level managers) and refined so that it is meaningful in terms of the real systems and the various locations and organisations in which they exist [14], [15].

Policies can also be used as a mechanism for enhancing trust within an e-service environment: an e-service takes on policies and must act in accordance with these policies. These policies are either associated with the result of a trust relationship between two parties or they are specified requirements given by a third party enabling trust relationships to be established. In that sense policy forms an essential component allowing the management of e-services to be tied into a trust relationship [16].

Policy based management of distributed systems promises to provide reliable and flexible solutions to the increasing complexity of managing security and trust in networks and open distributed systems, such as those required in Grid applications.

The recent trend of separating policy specification from the policy deployment, and both of them from the underpinning technology, allows for the policy to be modified at run-time. Separating system policies from the control mechanisms that interpret them allows the behaviour and strategy of the management system to be changed without re-coding the control mechanisms. The management system can then ensure continuity of operation and adaptability to changing requirements by disabling policies or replacing old policies with new ones as a part of its normal operation.

Policy enforcement mechanisms do not change between the traditional use of policy and this more distributed policy approach. There is a range of possible enforcement mechanisms from dictating the use of certain components to the configuration of systems to higher-level enforcement systems, such as for access control [18]. When the possible enforcement mechanisms rely on somebody correctly configuring the services and computer systems, as typical today, audit becomes an essential part of any policy enforcement system.

From this point on, we will refer to the description of a policy as the *"policy specification"* and to the control mechanism as the *"policy deployment architecture"*. We will also use the term *"policy enforcement agents"* to refer to the control mechanism components that cater for the enforcement of a given policy specification. These are a part of the policy deployment architecture.

The rest of this section introduces the policy deployment model. In subsequent sections we will sketch how it can be interpreted over a distributed firewall architecture which allows for the dynamic formation of closed user groups.

3.1 Policy deployment

The actions of policy enforcement should be assessed against the business objectives, the risk, and the performance cost on the system. With respect to this, security and access control policies are among the types of primary concern (although not all policies need to be enforced).

A strict interpretation of policy enforcement requires the continual monitoring of the relevant system entities' behaviour and often the interception of their interactions, so as to ensure that they comply with the policy. This adds an additional complexity (and possibly additional performance requirements) to the system, so selecting and specifying which policies or which parts of a policy should be enforced is important. This form of preventive policy enforcement is generally used when the consequences of a policy not being followed are high, or trust is low on adherence to the policy.

An alternative form of enforcement is to assume that all entities will naturally comply with a policy, provide an event-based mechanism to detect non-compliance, and only then trigger actions to record, report, and correct the action. This type of enforcement is cheaper to implement and doesn't impact the performance of the system as much. Posterior corrective actions are reasonable if the consequences of a violation of a policy are not severe. It is also a good method to use if there is little reason to suspect that a policy will be violated.

Specifying the role of policy enforcer in a way that ensures enforcing of the policy rules and taking of an appropriate obligation action if a violation occurs,

enables better policy management in the system, whilst maintaining the consistency of the policy specification of the system. Yet, the policy enforcement mechanism in most current systems is ad-hoc, does not scale and hard-coded into the programming of the systems. In part this is because most existing policy work has focussed on specification, information models and application-specific policy enforcement.

3.1.1 A general-purpose policy deployment model

There have been very few publications that address important goal of providing a general-purpose deployment model for policies. In this section we sketch an object-oriented policy deployment model inspired by [1] that addresses the instantiation, distribution and enabling of policies as well as the disabling, unloading and deletion of policies. The focus is on describing the policy deployment architecture consisting of policy objects and policy enforcement agents to respectively control and apply policy enforcement, and outlining the necessary interactions between them.

This deployment model is associated with the policy specification language Ponder [13], which supports access control by means of *authorisation, delegation, information filtering* and *refrain* policies.[2] The following key terms are most commonly used in Ponder specifications:

Subject - Refers to users, principals or automated manager components.

Target - Refers to objects (resources or service providers) which are accessed by a subject invoking methods visible on the target's interface. The granularity for access protection in Ponder is an interface method.

Domain - Refers to a means of grouping objects to which policies apply and partitioning the objects in a large system according to geographical boundaries, object type, responsibility and authority, or for the convenience of human managers [17]. References to both subject and target objects are stored within domains maintained by a domain service.

In addition to the basic policies mentioned above, Ponder provides policy composition mechanisms for the creation of groups of policies with a common semantic content or interdependency. Groups of policies referring to the same subject provide an interpretation of a role [19] understood as the specification of the policies that apply to an organisational position [20],[21]. Roles can be related to each other and thus give rise to a policy oriented view of the management structure of an enterprise.

In this model, each policy type is realised as a policy class and represented by a policy object at runtime. Policy management operations are carried out by invoking methods on the policy object. The policy object maintains the state of the policy and co-ordinates all policy operations acting as single point for managing concurrent and possibly conflicting requests from multiple policy administrators and from domain objects to which the policy applies. Policy objects may perform requests concurrently. For most operations, the policy object will invoke corresponding operations on its underlying enforcement agents. An overview of the policy deployment model is shown in Figure 2.

3.1.2 Supporting Services

As summarised in Figure 2 the policy deployment model considered includes the following three supporting services:

Domain service - Manages a distributed hierarchy of domain objects, and supports the efficient evaluation of subject and target sets at run-time. Each domain object holds references to its managed objects, but also references to the policy objects that currently apply to the domain. The service is assumed to generate events for changes to the membership of a domain and allows object to be members of more than one domain. After the policy has been distributed, each domain needs to maintain references to the policies applying to it. When a domain membership changes occurs, the domain object can notify the change to its referenced policy objects

Policy service - Acts as the interface to policy management. It stores compiled policy classes, creates and distributes new policy objects, and supports policy management actions not provided elsewhere in the model.

Event service - Collects and composes events from the underlying systems and from the managed objects in the system, and forwards them to registered policy management agents triggering obligation policies.

The instantiation of a basic policy creates and initialises a **policy object**, which can be either an **authorisation policy object** (APO), or an **obligation policy object** (OPO), or a **refrain policy object** (RPO). Composite policies map to sub-systems, which are represented by objects that elaborate the instances within them while delegation policies map to authorisation policy objects that allow grantors to invoke the delegate operation on the policy service, with respect to a specific authorisation policy.

[2]Ponder is the result of research in policy-based management at Imperial College, University of London, over the past 10 years [20], [21], [22], [23], [24]. It is a declarative, object-oriented language for specifying security and management policy for distributed object systems. It places emphasis on "non-discretionary" access control, where administrators have the authority to specify security policies to be enforced by the access control system. Of course, in principle, one could use Ponder for specifying also discretionary and mandatory policies.

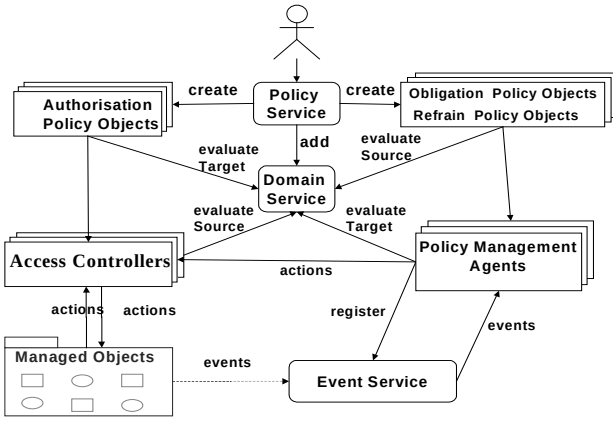

Figure 2: Example of a policy deployment model.

By having policy objects placed into one or more policy object domains one may allow policies to be applied to them and therefore use policies to control the deployment of other policies. For example, authorisation policies can be specified to control the entities that have access to the actions on the policy objects.

3.1.3 Policy Enforcement Mechanism

Policy objects entrust enforcement to separate agents. This devolution of policy enforcement increases performance and scalability allowing for the concurrent enforcement of a policy over heterogeneous system entities. Two types of policy enforcement agents are distinguished: **Policy Management Agents (PMA)** and **Access Controllers (AC)**.

Policy Management Agents (PMA) - Enforce all the enabled refrain and obligation methods for a subject. The enforcement objects for obligation policies and refrain policies are loaded from the corresponding policy objects and stored locally. When an obligation policy is enabled, its PMAs register with an **event service** to receive relevant events generated from the managed objects of the system. On receiving an event, PMA queries the domain service to determine the target objects used in the obligation method and it performs the policy actions, provided no constraint or refrain policy prevents the action.

Access Controllers (AC) - Enforce all the authorisation policies for one or more target objects. The policy object first evaluates the target set and determines the AC for each target object in the target set, and then informs to each AC which target objects the AC must protect and sends a copy of the enforcement object for the policy retaining references to each AC. ACs are normally co-located with the targets that they protect. Unlike PMAs, which can be generic, ACs require close interaction with the underlying access control mechanism. Consequently, the means they use to interact with each mechanism will vary. ACs are *not* event-driven. They invoke methods constraints and handling authorisation filters. When an action is *"intercepted"* by an AC, it calls a method to check whether the access should be permitted.

An overview of the essential dependencies between the policy objects and the policy enforcement agents are summarised in Figure 3.

The instantiation of a policy object gives rise to an enforcement object, which can be loaded into enforcement agents, and then enabled thus causing the enforcement agents to actively implement it. An enabled policy can be disabled and later re-enabled, or disabled and then unloaded, removing it from its enforcement agents. Unloaded policies can be reloaded or deleted.

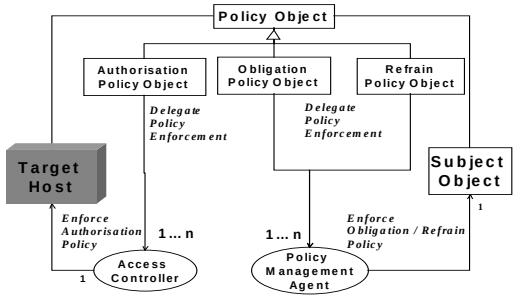

Figure 3: Dependencies between policy objects and policy enforcement agents.

4 Dynamic closed user groups (CUG) with integrated distributed firewall functionality

In this section we describe an architecture for distributed firewalls[3]. The main objective of this architecture is to develop a new distributed, secure working environment based on robust and efficient network mechanisms, allowing for dynamical closed user groups without topological constraints. Previously, Virtual Private Networks (VPNs) have only been configurable prior to use, but this scheme provides mechanisms where secure Closed User Groups (CUGs) can be dynamically altered. Aspects of both P2P and hierarchical working are included as this hybrid structure provides an architecture that is scalable, manageable and sympathetic to the different application services that will be supported.

4.1 System Architecture

The distributed firewall architecture, illustrated in Figure 4, comprises client hosts (containing CUG and firewall functionality), administrator nodes and optional Trust Authorities. Circles (on Figure 4) that represent different businesses don't have any topological implications, but only refer to logical structure of architecture. The administrator nodes are responsible for maintaining the firewall authentication policy through the creation of certificates and assigning them to users. Certificates are an attachment to an electronic message used for security

[3] Developed as a part of ongoing academic research at the Department of Electronic Engineering, Queen Mary, University of London.

purposes [25]. They accompany all messages and are used to establish the credentials of the sender. The certificate contains the sender's ID, a serial number, expiration date, a copy of the sender's "public" key for the CUG group concerned, and the digital signature of the certificate-issuing authority (typically, Administration node) so that a recipient can verify that the certificate is real. For relatively insecure usage, a certificate could be issued with a null expiration date allowing satisfactory operation for an indefinite period. Conversely, greater security could be achieved if the certificate is valid for a single transaction. The lifetime of the certificate forms part of the policy rules that are associated with it when it is issued.

Certificates are used to distinguish between members of different CUGs, meaning that only a client possessing an appropriate certificate can become member of the particular group. Also, separate certificates are used for the Administrator to client communication, for messages in relation to CUG management, and for messages used to disseminate firewall security and intrusion detection information[4].

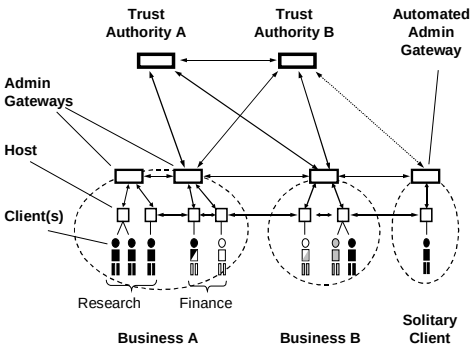

Figure 4. Distributed Firewall Architecture

Each administrator node can control many concurrent distributed firewall instantiations localised to individual host terminals. Administrator nodes may also be in touch with various Trust Authorities (third parties) who may assist them with authenticating the claims of other companies (e.g. for financial transactions or claims regarding professional expertise) [26].

The firewall functionality itself, is localised to each host, be it a personal computer or a mobile communications device. This controls personalised user access through the firewall at the given host based on the certificates in its possession for each identified user(s). All hosts then become part of a large distributed firewall providing all the features offered by traditional firewall choke point. Typically, the client(s) at each host terminal are individually assigned privileges that permit their host firewall to encrypt, send, receive and decrypt all forms of information between themselves and other client members of the same group as defined by the certificate(s) they have in common. As all communicated data is accompanied by a certificate wrapper, which is used by the host firewall CUG vetting function to determine whether to pass or block the message both at the point of origin and at the recipient(s), the architecture facilitates certificate-based policy enforcement in order to provide group confidentiality. That is, to protect against the inadvertent or malicious disclosure of group information to non-group members.

The hierarchical structure of the architecture provides the flexibility of both P2P information exchange and centralised server support. It allows different services to be supported in the most appropriate manner. For example, synchronous services (such as: chat, real-time voice, white boarding) can be handled in a P2P manner between client hosts, while, asynchronous services, (such as: discussion postings, file transfer, group calendar) can make use of the administrator as a repository and message queue. In addition, since one host can serve several users (see Figure 4), by introducing databases such as HLRs (Home Location Register) in mobile phone networks, one could extend this concept to true nomadic computing, where users can access their CUGs from every terminal, regardless of location.

4.1.1 Description of Protocol

In order to become a member of the overall environment, a client has to first register by sending a register request message to the Local Administrator (LA) node, where the client presents its identity through a digital signature or some similar form of authentication. This information may be optionally passed to a Trust Authority, for validation (different Trust Authorities can be contacted as appropriate). If the registration is approved, the client can choose either to join[5] existing group(s), or to create new one(s).

Typically, once created, a group exists until the last client leaves it, be it the creator of the group or any other member. This also means that creator normally doesn't have to have any special privileges; it is only the entity that initiated creation of a particular group. However, depending upon the nature of the group, the firewall policy rules may be either set solely by the Administrator, or by the Administrator in consultation with the originating client. Once the policy is in place, such as which port the secure communication should take place over, it is enforced by the firewall software on the client host.

[4] An essential extension to this scheme is for each host to support a genetic learning firewall mechanism. If a particular machine is attacked, it could send a message to its parent Administrator node (with a certificate wrapping), informing it of the nature of the suspicious activity. The Administrator could then optionally modify the behaviour of the other hosts it is responsible for, implementing proactive protection.

[5] As the Administrator node provides management of the CUGs, its role during the registration phase could be extended to provide the new client with information concerning existing groups that it may be of interest.

Clients are free to leave the group whenever they want. Furthermore, the Administrator node can expel them at any point in time if they break the rules of the behaviour within the group. Exceptionally, an Administrator node can deregister a client, e.g. exclude it from overall environment if it commits a serious violation. Under these circumstances, the client would not be allowed to join any group administered by this Administrator node.

If a client decides to join an existing group, it sends a *join request* message to its Local Administrator (LA). If the request is approved, the Administrator provides the client with the appropriate CUG certificate (for the P2P client communication), informing the other CUG members of the presence of the new member. Also, a client can initiate the invitation of specific users (of his choice) to join the group.

In an environment with dozens of administration nodes and thousands of clients, it is very likely that clients controlled by different administration nodes may wish to become members of same CUG, particularly when extranets are to be formed. The administration node responsible for initially creating the CUG typically retains responsibility for it throughout the CUG's lifetime. However, remote clients from other administration domains are able to join this group.

4.1.2 Example: *Joining a remote user group*

We elaborate this via an example, shown in . Normally, a client X is prevented from contacting a Remote Administrator (RA) directly, due to port restrictions imposed by Local Administrator (LA)[6]. Instead, it is able to contact RA via LA. If LA approves this, then LA acts as a proxy client and attempts to join the remote group on the local client's behalf. If RA accepts this request, LA will receive a certificate for peer-to-peer client (C2C) communication within the scope the intended group, that is to be forwarded to and used by the appropriate client.

The possession of a C2C certificate is sufficient for the local client to actively participate in group communications with the remotely hosted CUGs. Still the local client is unable to contact the RA directly, allowing LA to retain a greater degree of control over its client. When new members join the group, for example, LA (acting as the proxy) will be informed by RA. LA must then forward this information to the appropriate local client(s). As depicted in , the client X sees only his own administrator (LA) and the entire client–members of the CUG B. From RA's point of view, LA is simply a client. The figure also shows CUG A, which has high security demands (or low privileges), and whose members are not allowed under any conditions to contact any of Remote Administrators. They will be refused at LA node. Exceptionally, LA may maintain a list of RAs with which local clients can join CUG groups.

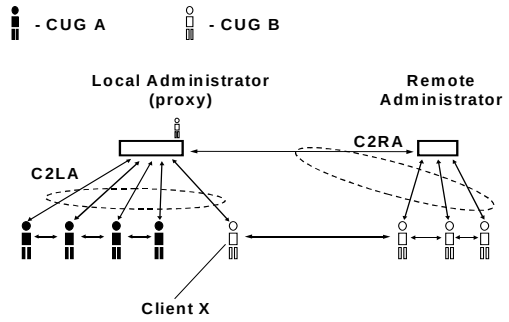

Figure 6: Local Administrator / Proxy Client Operation

4.2 Policy and Firewall Updates

Clients can ask to have the policy updated but the Local Administrator node makes the decision. Under exceptional circumstances, the LA could accommodate requests for policy updates from the group members or members with particular privileges, such as moderators. However, the Administrator is always responsible for forming, regulating and removing CUGs. It retains responsibility for issuing the policy certificates and for the programming of the client firewall rules.

During runtime, different types of network activity will be seen and the distributed firewalls can observe this traffic and also observe the "health" of the system in terms of both network and host performance. This can be measured in terms of average packet throughput, packet loss and packet delay. If these parameters are deemed to fall outside thresholds of normal behaviour, the client host can inform its LA. Additionally the client may instigate local protective algorithms to try and rectify the situation and inform it of the outcome. This is the mechanism by which successful distributed firewall configurations are identified and then cached for future use. Also, if any type of network attack is experienced and defence is successful, that information will be stored. The successful configuration could then be propagated to other local client firewalls and beyond, to protect other hosts from attacks that the local firewall has seen, but they have not. This process can be improved by applying intelligent reasoning and machine learning for the advancement of network security mechanisms (as in [27]) in each node so as to continuously improve the robustness and adaptability of the distributed firewall. This process is illustrated in Figure 7.

[6] Depending upon the firewall policy of particular clients, they may be able to contact Remote Administrators (RA) directly, in order to join CUGs administered by them. In such scenario client can choose to join any of the groups administered by RA (there is no LA vetting of RA groups).

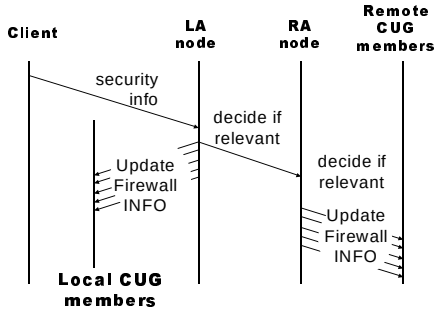

Figure 7: Example Firewall Update Message Sequence

The client initiates this process. It sends the Administrator information regarding any abnormal (unexpected) behaviour. The Administrator then examines received information and makes decision about its significance. If relevant, the Administrator can choose to disseminate this information to other clients within its domain, and possibly to inform Remote Administration nodes.

4.2.1 Underlying agent technology

In a typical organisational environment, individuals are not necessarily the administrators of the computers they use. Instead, to simplify system administration and to permit the necessary central control, an administrator agent is used to administer individual machines. As proper firewall configuration requires a moderate level of network and security expertise, it is inappropriate to require these skills of stand-alone individuals. Is this case one may use an automated administrator agent server.

The distributed firewall architecture can be viewed as a multi-agent system with various types of agents. In general terms, an agent can be defined as a software program capable of executing a complex task on behalf of a user. Both individuals and companies have administrative agents that communicate in the system to maintain the firewall according to the certification process. These administrative agents are responsible for managing the creation of new certificates and assigning them to staff (in the case of a company) or individuals. Administrative agents are also in touch with various Trusted Authority Agents (third parties) who may assist them with authenticating the claims of other parties. A separate client firewall agent instantiation is also present on each host system. This controls access through the firewall at the given host based on the certificates in its possession.

In order to communicate throughout this system, it is envisaged that these agents may use an open standard Agent Communication Language.[7] The basic multi-agent architecture is illustrated in Figure 8.

[7] The forerunner in this field is FIPA ACL [28] (Foundation of Intelligent Physical Agents), though KQML is an alternative (Knowledge Query Manipulation Language) that could be considered.

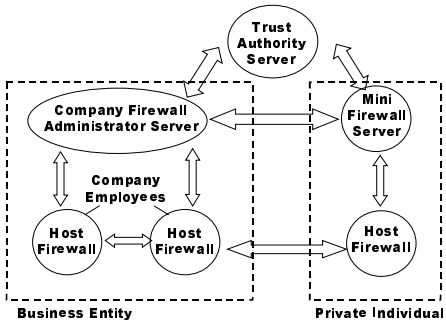

Figure 8: Distributed Multi-Agent System

5 Policy Deployment over a Distributed Firewall Architecture

In this section we focus on the realisation of access control policy deployment over a distributed firewall architecture with closed user groups. The aim is to provide an integrated architecture, which aims to deliver the flexibility, intelligence and scalability of policy-driven management for access control combined with the decentralisation, distribution and efficiency of the CUG distributed firewall architecture.

The section is structured as follows: In section 5.1 we discuss the impact of policy deployment over a distributed firewall. In section 5.2 we elaborate a realisation of the policy model described in section 3.1 on the CUG distributed firewall architecture describe in section 4.1. We first explain how the supporting services of the policy deployment model are realised over a CUG distributed firewall architecture (paragraph 5.2.1) and then elaborate the role of the policy control and enforcement agents in this realisation (paragraph 5.2.2). An illustrative example (paragraph 5.2.3) explains how the proposed realisation works when a local client host joins a remote CUG, hence redrawing the example.

For the purposes of the architecture presented in this paper, we have chosen to keep the concepts of a CUG and a domain distinct. We see domains as referring to logical groups of system objects reflecting geographical boundaries or enterprise structure. We see CUG as referring to dynamically formed contained coalitions of actors and resources - engineers, tools, data storage, etc - with a common objective. For example, actors from different enterprises may agree to form a CUG for a limited period for the purpose of distributed scientific experiment. Once the experiment is finished, the CUGs may dissolve. Changes to domains, on the other hand usually reflect changes to the structure of an enterprise.

5.1 The Role of Policies within a Distributed Firewall Architecture

A policy is enforced by the policy enforcement agents residing at each individual host that participates in a distributed firewall. The (security) administrator, whose location is immaterial, defines the security policy and correlates it to certificates. This effectively corresponds to *establishing and maintaining a mapping between certificates and roles* [19][20][21] (i.e. the collection of policies that apply to a client). By informing the policy enforcement agents about the correspondence between certificates and roles, P2P interaction between client hosts becomes feasible. The certificate is consulted before processing incoming or outgoing messages, or granting access to verify their compliance. By using certificates and relating them to roles the local host has a much stronger assurance of its identity and role than in a traditional firewall.[8]

Furthermore, as all CUG communication takes place with firewall awareness, the dynamic CUG services can be regulated. Furthermore, it is possible within the scheme for the clients to renegotiate privileges with the Administrator. Updating of firewall access policy is integrated with the CUG architecture and can be managed by suitable obligation polices.

The scheme is scalable and provides "openness" as anyone can normally initiate a group. Membership is vetted by the Administrator of the CUG concerned. However, members of a CUG can read and send data securely with other members using encryption keys unique to each group, subject to the policy constraints set by the group's initiating user or their elected representative. This P2P activity takes place without Administrator intervention, affording flexible communication without the loss of security across a publicly available infrastructure.

By managing the certificates, policies may also be used to ensure confidentiality and privacy, as well as to regulate the operation of the firewall at multiple protocol layers.

5.2 Method Integration

From a CUG perspective, is worth distinguishing two logically different classes of policies: *local policies* and *CUG policies*. The former are owned and managed by the Local Administrator (LA) of a collective of client hosts and apply to the clients associated with this LA. The latter are defined for each CUG when CUGs are established and then maintained by the administrator. CUG administrators compile policies, and they control and maintain policy deployment of the policies applying to the CUG they administer. Controlling the policy deployment involves the ability to interpret and maintain meta-policies (constraints about which policies apply[9]) to their CUG and to their local clients. Administrators also register and maintain references to their client base and policies for their clients.

As the architecture described in section 4.1 allows for P2P communication between administration nodes, CUG administrators have the ability to be informed by the LA of each remote client involved in their group about its local policies and should be able to check if the local policies of the LA for the client who joints a remote CUG are violated by the policies of the remote CUG and vice versa.

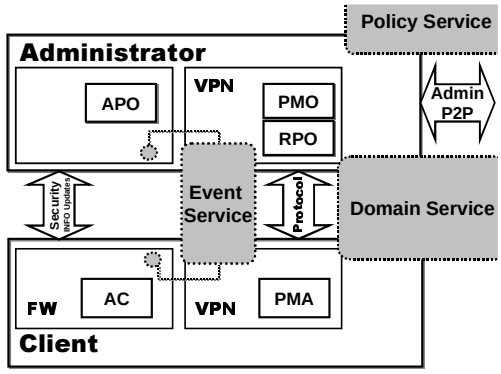

Figure 9. Policy enforcement in CUG architecture

Figure 9 provides a pictorial overview of the relationship and interactions between the concepts defined in the policy deployment model and CUG architecture described in sections 3.1 and 4.1 respectively. In the following, we elaborate the relationships between these basic concepts.

5.2.1 Supporting Services

Services are collectively realised by the CUG architecture through a combination of local instantiations and P2P communication between administration nodes. The realisation of each of the supporting services of the policy deployment model presented in section 3.1.1 is discussed in turn.

Policy Service - Each administrator has its own local implementation of the policy service, which is complemented with additional P2P communication among different administration nodes. P2P communication between administration nodes is initiated for a variety of purposes. These include: negotiation of a local administration node on behalf of its clients (for example when a client wishes to join a remote CUG in order to perform a task), policy updates, and updates of security intelligence.

[8] Certificates also benefit from being independent of topology. For example, if a machine is granted certain privileges based on its certificate, those privileges can apply regardless of where the machine is located physically.

[9] Meta-policies are semantic constraints which restrict the set of permissible policies thus defining policies about which policies can coexist in the system or what are permitted attribute values for a valid policy. They are particularly useful for specifying various precedence relationships that can be established between policies in order to allow inconsistent policies to coexist within the system [20].

Clearly, such an approach adds administration overhead (as one may need to exchange policy specifications between administrators and generate policy objects locally), and necessitates the definition of a machine-readable policy exchange language. However, it allows decentralised policy deployment by providing the essential functionality for the exchange of local policies. More importantly, incorporating peer-to-peer communication between administration nodes in the realisation of the policy service facilitates exchanging security intelligence (such as firewall updates) and imparting this intelligence into the deployment of policies.

Event Service - We assume an event service per Virtual Private Network (VPN) accompanied by interfaces implemented locally on each of the administration nodes. The event service collects and composes events from the component systems and from the managed objects in the VPN. The local interface on each of the administration nodes incorporates intelligence implemented in each node for security purposes, e.g. Case-Based Reasoning algorithms (and databases related to them) that examine data logs of suspicious behaviour, recorded on client side. Event service interfaces can exist on client side as well, though in more primitive form, since main intelligence of the system is localized in admin node. Although exchange of events between different VPNs is not prohibited, this can be viewed as a part of interaction within a CUG that invokes the corresponding administration nodes as members of the group.

Domain Service - We make the simplifying assumption that a conceptually unique and global domain service is realised by the underlying communication protocol. Supported by this domain service, local administrators resolve domain references to their local clients and act as proxies to remote administrators of CUG to which their local clients may belong. Clearly, this abstraction applies merely to the logical structure of internal CUG protocol within overall environment, rather than the physical implementation of VPNs that have to be dynamically established during runtime.

5.2.2 Policy Deployment and Enforcement Agents

Following the deployment model described in section 3.1, we distinguished two classes of agents that contribute to the deployment and enforcement of policies. The first, **policy objects**, are generated by the policy service; they are owned by, and reside with, administration nodes. The second, **policy enforcement agents**, reside with each client host.

In the following paragraphs we discuss how these classes of agents operate within the CUG distributed firewall architecture presented in section 4.1.

Policy Objects - generated by the local instance of the policy service; they are owned by and co-located with the administration nodes. As already elaborated, the update and exchange of policy specifications is managed by the administration nodes. The situation where all members of a CUG have the same LA does not present any particular challenges as the CUG and local administrations coincide: all policy objects are produced by and reside with the same administration node.

The scenario where some member of a CUG has a different local administrator (LA) than the CUG administrator (RA) presents some challenges both in terms of the location of the CUG policy objects and their interaction with the policy enforcement agents: When client joins a remote CUG, the new CUG administrator (RA) generates the policy objects for the CUG.[10] This gives rise to three distinct interaction scenarios depending on the credentials of the RA as seen by the LA. (Recall that, as explained in paragraph 4.1.2, from the point of view of the local client, the LA acts as a proxy for the RA.)

LA does not trust the RA: In this scenario LA requires a copy of the relevant *CUG policy specification*, which has to be expressed in a machine readable, standardised and mutually understandable policy exchange language.[11] The LA then generates a local copy of (or interface to) the policy object and synchronise with the archetype residing with the RA.

The client doesn't have sufficient credentials for direct communication with RA: In this scenario the LA accepts to host (a copy of or an interface to) the policy object sent by the RA. The policy is then enforced on client side for purposes of peer-to-peer communication between clients. Still, the local control regarding group management resides then on LA, but the LA doesn't take any actions (policy enforcement regarding that particular group), unless instructed by the RA or LA decides that it has to override the enforcement of the particular CUG policy reacting to a conflict between the CUG policy and local policies.

The client has sufficient credentials for direct communication with RA: In this scenario the LA allows direct communication between the RA and the local client for issues regarding the CUG. For example the policy object of the RA is allowed to communicate remotely with the access controller of the client in order to enforce a CUG authorisation policy, while LA still maintains some responsibility for the security status and credentials of its client. (E.g. LA is able to intercept the communication so as to override the enforced CUG policy, if necessary.)

---

[10] We assume that the policies of the new CUG have been agreed upon between LA and RA before the formation of the CUG. Notably, this requires that the administration nodes have the ability to relate credentials (presented by means of certificates) sent by another administrator to their own local policies and are equipped with mutually agreed meta-policies to cater for any conflicts between remote and local policies.

[11] The syntax and sematnics definition of such a policy exchange language falls beyond the scope of this paper. See [29] and [30], [31] for preliminary results in this direction.

Regardless of the scenario, the instantiation of a policy object gives rise to an instance (enforcement object) which can be loaded into its enforcement agents, and then enabled causing the agents to implement it. An enabled policy can be disabled, re-enabled, unloaded, reloaded or deleted as in described in section 3.1.3.

Policy enforcement agents - low-level counterparts of policy objects, so their role and deployment in the system (normally in client node) is always in relation to the policy objects residing at the corresponding administration nodes.

Access Controllers (AC) are generated by the LAs and accommodated on the client hosts. Their primary role (in a CUG context) is *preventive and proactive security policy enforcement*. That is, to enforce the authorisations dictated by each policy. They also have the required functionality for intercepting policy enforcement or overriding CUG policies if requested by their LA. They communicate with the APO of the CUGs in which their host belongs to and with the APO of their LA, which caters for their local policies. They enforce access control in peer-to-peer communication between client hosts (C2C) by associating group certificates to the execution of access control rules.

Policy Management Agents (PMA) are hosted at client node, similarly to AC, but communicate with the RPO or the OPO, for different purposes (see paragraph 3.1.3). Their role is to implement *event-driven security policy management* and *reactive security enforcement*. They enforce policy-based management in C2C communication by associating group certificates to the execution of CUG policy management rules.

5.2.3 Illustrative Scenario of Policy Enforcement

Some interesting aspects of the method integration elaborated in this section can be illustrated by revisiting the scenario where a client joins a remote CUG introduced in paragraph 4.1.2.

First, a client requests to join a new (remote) CUG. Becoming a member of a CUG *is endorsed by possessing a certificate* issued by the CUG administration node and verifying that one becomes a member of the CUG. The client's local administrator (LA), acting as a proxy, contacts the remote CUG administrator (RA) negotiating for a certificate in favour of its client.

This negotiation involves comparing the local and CUG policies that may apply to the client and reaching a consensus. That is, agreeing on a group of basic policies that apply to the client and define its role in the CUG. The policies are owned by the RA but are agreed upon with the LA who maintains the responsibility for (mutually agreed) meta-policies addressing potential conflicts between local and remote CUG policies for its client.

Each administration node is responsible for maintaining references mapping each local client or CUG member to the local or CUG policies that apply to this member. In the CUG case, this involves *maintaining a mapping between certificates and roles* (i.e. the collection of CUG policies that apply to a CUG member). In this scenario LA is responsible for maintaining the knowledge base about the local policies that apply to its client and RA is responsible for maintaining a knowledge base about the CUG members and policies that apply to them.

If an update of existing policies is necessary then the administration node responsible for these policies should update the policy object (that is owned by this administration node) and broadcast a description of the update. Since the policy object resides with the administration node, the policy enforcement agents on the client hosts are the system entities that actually enforce the policy. The updated policy object reloads and reactivates the new executable form of the policy to the policy enforcement agents ensuring that:

– the certificate of the new member will be recognised by the collective so as to allow peer-to-peer communication between CUG members to happen
– the CUG policies relating the new member's role will be enforced once the certificate is presented.

Once the certificate is delivered (from RA to client, via LA), peer-to-peer client communication can be established. Of course, in order to avoid unnecessary temporal conflicts policy updates must be completed before the new member's certificate is activated. Hence, all necessary policy and domain knowledge has been updated and all the clients (already members of particular CUG) are aware of the presence of new member. RA, as the CUG administration node, is responsible for bringing about these updates.

6 Motivating Scenario Revisited

The proposed realisation of policy management on a distributed firewall architecture supporting CUGs provides a highly suitable architecture for supporting Grid applications. The following features are particularly notable.

– Each site can enforce a local policy controlling the activity of the located at that site and the access to the resources of that site.
– CUGs can be established within and across domains. CUGs have separate identities and can enforce their own policies. Thus they are a suitable realisation of the concept of virtual organisation from [42].
– System entities in different roles (and therefore with different rights) can coexist within the same CUG, policy enforcement agents are now able to correlate certificates to roles.
– The application of CUG policies can be managed within a framework of meta-policies. Meta-policies

are particularly for specifying various precedence relationships that can be established between policies in order to allow inconsistent policies to coexist within the system. As [20] elaborates, they provide the means for conflict analysis that has to be part of a Role-based Management framework.

- The local domain policy can override the CUG policy to enforce local rules.
- Access rights of the member and the CUG security management rules can be changed by policy updates, without the need to issue new certificates.
- CUGs can be established and extended dynamically through negotiation between administrators.
- Once CUGs have been established, clients can participate on a P2P basis, with policies enforced by agents within their own local firewalls, without the need to involve the administrators.
- By using different certificates for different CUGs, users can be members of several CUGs at once without compromising their security.
- CUGs can by established dynamically, and by issuing certificates for that CUG which have limited lifetime, the CUG can expire once the actions it has been established for have been executed. Thus on-the-fly CUGs can be established.
- Local administrators can act as proxies for clients within their domains.
- The use of obligation and refrain policies can enable resource charging mechanisms.

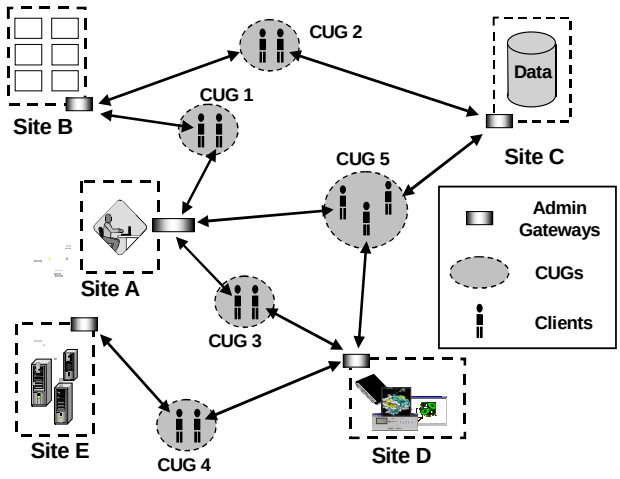

Figure 10: The motivating scenario revisited

Thus the example (simplified for brevity) given in section 2 can be reformulated in this architecture as follows, and illustrated in Figure 10. Recall that each actor (A-E) in the original scenario is within a site administered locally by its own administrator (LA-LE). We assume that there are the following four pre-existing CUGs within the example:

CUG 1 which includes the engineer at A and the portal at B. This would be the group of all users of the portal at B. A user joins this group the user registers with the portal and is provided with a certificate allowing the user access to the portal. This may include the obligation policy imposed by the CUG administrator on A to provide payment for usage.

CUG 2 which includes the portal at B, the data source at C. This is the CUG of all data sources for which B mediates. The access control policies of this CUG would allow the portal to search B's archive, and also to negotiate as a broker on its behalf with the portal's users. We note that the ability to include system entities in different roles (and therefore different rights) within the same CUG, is supported by the architecture described in section 5. This is a significant extension to the CUG functionality.

CUG 3 which includes the engineer A and the analysis tools at D. This is the CUG of the virtual organization to which A and D both belong. This CUG would have a policy that allows A to perform certain operations (upload data from a user specified place, run specified programs, use temporary storage) with the obligation on A that the data supplied should match certain constraints or be in specified formats.

CUG 4 which includes the university at D and the computational resource at E. This would have the policy that D could call on the resources of E in return for payment.

Thus when user A accesses the portal B she presents her certificate within the CUG 1, on a P2P basis, without involving the administrators. Similarly, B searches the catalogues of C with a certificate from CUG 2.

However, when the user selects a suitable dataset, it has to be downloaded to location of her choice (normally, site with tools and resources). A new CUG 5 may be dynamically established for this purpose. The group formation is initiated by A, who subsequently invites C and D. Since A and C may not have prior knowledge of each other, site B acts as a broker for this negotiation, informing C about A. Local Administrator of A then initiates issuing of short-lived certificates so that A can access C's data, and C can bill A as appropriate. In this negotiation, the local administrators are involved to ensure that the negotiated policy does not contravene local policies; for example, user A may have an upper limit set on the price which she is allowed to pay for data and the administrator will have to oversee the negotiation to ensure that this is upheld.

When D requires the assistance of resources at E, then CUG 4 is used. (However, depending on the nature of the agreed policy negotiated between A and E on which party will bear the cost of additional resources called upon in the use of D's facilities, a new CUG many need to be established between A and E to handle any necessary payment).

7 Conclusion and Further Work

The traditional Grid infrastructure, such as the Grid Security Infrastructure (GSI) from Globus [41], using the X.509 certificates as its authentication mechanism, depends on interfaces at the protocol level to provide the security infrastructure. However, this approach has concentrated on authentication and not provided a sufficient infrastructure for the rest of the trust hierarchy exemplified in section 2, especially with respect to authorisation and the statement and implementation of policies. Further, this mechanism does not appear to have considered the case where the collaborating resources have no prior knowledge of each other (or their certifying authorities). Thus, unless all the resources accessed in the example are in the same virtual organisation, and with all users having the same rights, the existing infrastructure is not adequate to fully support the requirements of the example presented in section 2.

In the present paper, we focused on the realisation of authentication and access control management over a distributed firewall architecture that is based on the concept of Closed User Groups (CUGs) [2] and can be easily adapted to Grid based implementation. Of course this constitutes only part of the realisation of a security infrastructure for the Grid.

In [35] we identified the need to supplement this infrastructure by raising the level of the trust within a Grid architecture. This would built upon the established literature in trust analysis, such as [36], [37], [38], [39], [40] which provides a framework for analysing how trust should be transmitted between agents in distributed systems, especially dealing with how to propagate trust between agents with little or no prior knowledge of each other. The basis of this infrastructure is the explicit declaration and publication of trust policies by participating resources on the Grid using an appropriate policy specification exchange language. Agents wishing to utilise resources would be able to present their credentials, policies and requirements to the participating resources and an automated process would verify the credentials, possibly referring to trusted third parties, to establish identity, deduce authorisation based upon supplier and consumer policies and to authorise (or revoke) access under the specified the conditions of use. Trust management policies can be used to control the authorisation of previously unknown users via negotiation with certification authorities. To support this one would need to add at least the following.

- Resource brokerage services to facilitate resource discovery and allocation in compliance with a contractual realisation of QoS requirements. This can be supported by incorporating a generalisation of the existing mechanism for sharing firewall-updates (paragraph 4.2) which may involve resource usage tracking and charging.
- A means of publishing, negotiating and exchanging policy statements. See [29] and [30],[31] for preliminary results in this direction.[12]
- Appropriate trust services, such networks of trust authorities, and an infrastructure allowing for the dynamic formation of certification chains.
- A trust management framework able to cope with the complexity and uncertainty underpinning most interactions in open dynamic systems such as the Grid architectures. This will need to draw a distinction between perceived and actual security, relate trust to enterprise objectives and weigh it against transaction risk. For preliminary results in this direction see [39][40].

In terms of further work, a major milestone is to test the applicability of the architecture presented in this paper within a Grid test-bed in areas such as e-Science[13] or e-Business[14].

In the near future, we plan to relate the formation of CUGs to the goals of the initiating participants, to investigate policy negotiations as understood in [43] and to experiment with incorporating policy information specifying allowable behaviour as a range of options in the certificates, in a similar spirit to [44].

Finally, in recognition of the fact that appropriate analysis of security risk within the system should begin at the conception of the system, we intend to assess the integrated architecture presented in section 5 on basis of interaction scenarios similar to the one discussed in this paper. The aim of this task will be to look assess alternative variations of this architecture in relation to specific security threats. For this, we aim to deploy analytical risk analysis methods for the evaluation of the architecture. Such methods are being developed within the European project CORAS[15] [45],[46],[47], involving the authors. As a first step in this direction, an adaptation of the CORAS approach to cover the evaluation of the network architecture model for highly dynamic, distributed and secure business-to-business environment that underpins CUGs is currently on-going [48].

---

[12] [29] discusses the shortcomings of XML extensions for this prupose and valuates the suitability of the W3C standard RDF [32] as an alternative means of encoding a semantically sound exchange language for access control and management policy specifications. [30][31] focus on the provision of an architecture for publishing data protection and privacy policies based on a combination of the W3C standard P3P [33] (for describing privacy policies) and the XML extension CPExchange [34] (for describing personal profile data).

[13] E.g. within CLRC e-Science program http://www.escience.clrc.ac.uk.

[14] E.g. within GRASP a forthcoming European project aiming to explore an advanced infrastructure for Application Service Provision based on Grid technology. See http://www.bitd.clrc.ac.uk/Person/T.Dimitrakos for an update.

[15] CORAS is a current European project developing a tool-supported framework for model-based security risk analsyis, by integrating methods for risk analysis of critical systems withn a semi-formal instantiation of a RM-ODP inspired modelling framework.

References

[1] C.M. Bemardes, E.S. Moreira, “Implementation of an Intrusion Detection System Based on Mobile Agents”, IEEE, 2000.

[2] I. Djordjevic, C. Phillips, “Certificate-Based Distributed Firewalls for Secure E-Commerce Transactions”, Proc. of FITCE Congress, Barcelona 2001; also, to appear in the Journal of the Institution of British Telecommunications Engineers (IBTE)

[3] M.S Bellovin, “Distributed Firewalls”, AT&T, November 1999, www.research.att.com/~smb/papers/distfw.pdf

[4] N. Dulay, E. Lupu, M Sloman, N. Damianou, “A Policy Deployment Model for the Ponder Language”, In Proc. IEEE/IFIP International Symposium on Integrated Network Management (IM'2001), Seattle, IEEE Press, May 2001.

[5] I. Foster, C. Kesselman, (Eds) “The Grid: Blueprint for a New Computing Infrastructure”, Morgan Kaufmann Publishers, 1998.

[6] NASA's Information Power Grid (lPG) home page, www.ipg.nasa.gov

[7] European DataGrid Project, www.eu-datagrid.org

[8] The Globus Project home page, www.globus.org

[9] The Condor Project home page, www.cs.wisc.edu/condor

[10] The Legion Project home page, www.cs.virginia.edu/~legion

[11] I. Foster, C. Kesselman, S. Tuecke, “The Anatomy of the Grid: Enabling Scalable Virtual Organizations”, (to be published in Intl. J. Supercomputer Applications, 2001).

[12] Workshop on Web Services, San Jose, 11-12 April 2001, www.w3.org/2001/01/WSWS

[13] N. Damianou, N. Dulay, E. Lupu, M. Sloman, "The Ponder Policy Specification Language" Proc. Policy 2001: Workshop on Policies for Distributed Systems and Networks, Bristol, UK, 29-31 Jan. 2001, Springer-Verlag LNCS 1995, pp. 18-39

[14] R. Wies, “Policies in Integrated Network and Systems Management: Methodologies for the Definition, Transformation, and Application of Management Policies”, PhD thesis, Univ. of Munich, Germany, 1995

[15] G. Goh, “Policy Management Requirements”, System Management Department, HP Laboratories Bristol, April, 1998.

[16] M. C. Mont, A. Baldwin, C. Goh, “Role of Policies in a Distributed Trust Framework”, HP Laboratories Bristol HPL-1999-104, September 1999

[17] M. Sloman and K. Twidle, *Domains: A Framework for Structuring Management Policy*. Chapter 16 in Network and Distributed Systems Management (Sloman, 1994ed), 1994a: p.433-453.

[18] R.S. Sandhu and P. Samarati, "Authentication, Access Control, and Intrusion Detection". Part of the paper appeared under the title "Access Control: Principles and Practice" in IEEE Communications, 1994. 32(9): p.40-48.

[19] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman, Role-Based Access Control Models. IEEE Computer, 1996. 29(2): p. 38-47.

[20] E.C. Lupu, M. Sloman, “Conflicts in Policy-Based Distributed Systems Management”, IEEE Trans. on Software Engineering, 25(6): 852-869 Nov.1999.

[21] E.C. Lupu, “A Role-Based Framework for Distributed Systems Management”, Ph.D. Thesis, Imperial College, London, U. K.

[22] E.C. Lupu, M.S. Sloman, “Towards a Role Based Framework for Distributed Systems Management”, Journal of Network and Systems Management, 1997b. 5(1): p. 5-30.

[23] D.A. Marriott, “Policy Service for Distributed Systems”, Ph.D. Thesis, Imperial College, London, U.K.

[24] M.S. Sloman, “Policy Driven Management for Distributed Systems”, Journal of Network and Systems Management, 1994b. 2(4): p. 333-360.

[25] S. Kent, 'Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management”, IETF IAB RFC 1422.

[26] Z. Liu, P. Naldurg, S. Yi, T. Qian, H.R. Campbell, M.D. Mickunas, “An Agent Based Architecture for Supporting Application Level Security”, IEEE, 0-7695-0490-6/99, 1999.

[27] W. Lee, S.J. Stolfo, K.W. Mok, “A Data Mining Framework for Building Intrusion Detection Models”, Proceedings of the IEEE Symposium on Security and Privacy, Oakland, California, 1999.

[28] FIPA home page, www.fipa.org

[29] T. Dimitrakos, B. Matthews, J. Bicarregui, “Towards supporting security and trust management policies on the Web”, Presented at the ERCIM workshop 'The Role of Trust in e-Business' in October 2001, Zurich. (Proceedings to appear.)

[30] K. Bohrer, X. Liu, D. Kesdogan, E. Schonberg, M. Singh, S.L. Sparagen. "Personal Information Management and Distribution". Proceedings of the 4th International Conference on Electronic Commerce Research, November 2001, ISBN 0-9716253-0-1.

[31] K. Bohrer, D. Kesdogan, X. Liu, M. Podlaseck, E. Schonberg, M. Singh, S.L. Sparagen. "How to go shopping on the World Wide Web without having your privacy violated". Proceedings of the 4th International Conference on Electronic Commerce Research, November 2001. ISBN 0-9716253-0-1.

[32] O. Lassila, R.R. Swick, (Editors) “Resource Description Framework (RDF) Model and Syntax Specification”, W3C Recommendation 22 February 1999

[33] P3P – www.w3.org/TR/P3P

[34] CPE – www.cpexchange.org/standard

[35] B. Matthews, J. Bicarregui, T. Dimitrakos, “Building Trust on the GRID - Trust Issues Underpinning Scalable Virtual Organisations” Position paper presented at the ERCIM workshop 'The Role of Trust in e-Business' in conjunction with IFIP I3E conference on October 3, 2001, Zurich. (Proceedings to appear.)

[36] M. Blaze, J. Feigenbaum, J. Lacy "Decentralized Trust Management". *Proc. IEEE Conference on Security and Privacy*, Oakland, CA. May 1996.

[37] T. Grandison, M. Sloman "A Survey of Trust in Internet Applications" In *IEEE Communications Surveys and Tutorials*, Fourth Quarter 2000.

[38] A. Jøsang, “The right type of trust for distributed systems”, Proc. of the New Security Paradigms Workshop, ACM, 1996.

[39] T. Dimitrakos, J. Bicarregui, "Towards A Framework for Managing Trust in e-Services", Proc. of the 4th International Conference on Electronic Commerce Research, ATSMA, IFIP, INFORMS, ISBN 0-9716253-0-1, November 2001.

[40] T. Dimitrakos, “System Models, e-Risk and e-Trust. Towards bridging the gap?", Towards the E-Society: E-Business, E-Commerce, and E-Government, (Proceedings of the First IFIP conference on E-Business, E-Commerce, E-Grovernment.), Kluwer Academic Publishers, ISBN-0-7923-75297, 2001.

[41] R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, V. Welch, “A National-Scale Authentication Infrastructure”, IEEE Computer, 33(12): 60-66, 2000.

[42] I. Foster, C. Kesselman, G. Tsudik, S. Tuecke, “A Security Architecture for Computational Grids”, Proc. 5th ACM Conference on Computer and Communications Security Conference, pg. 83-92, 1998.

[43] V. Gligor, "On the Negotiation of Access Control Policies". Invited talk at the Policy Workshop 2001.

[44] T.P. Dinsmore, M.D, Balenson, M. Heyman, S.P. Kruus, D.C. Scace, T.A. Sherman, “Policy-Based Security Management for Large Dynamic Groups: An Overview of the DCCM Project”, IEEE, 0-7695-0490-6/99, 1999.

[45] CORAS web page, http://www.nr.no/coras/ See also http://www.bitd.clrc.ac.uk/Activity/CORAS

[46] Ketil Stølen: "*A framework for risk analysis of security critical systems"*. In supplement of the 2001 International Conference on Dependable Systems and Networks, pages D4 - D11, July 2-4, 2001, Gothenburg, Sweden

[47] T. Dimitrakos, J.C. Bicarregui, K. Boge, B.A. Gran, T.A.Opperud, D. Raptis, E. Skipenes, K. Stølen. "CORAS - A framework for Risk Analysis of Security Critical Systems". In Proc. of the ERCIM Workshop "The role of trust in e-business", Zurich, October 2001.

[48] I. Djordjevic, E. Scharf, “Suitability of Risk Analysis Methods for Security Assessment of Large-Scale Distributed Computer Systems”; PSAM 6 Conference, 2002 (submitted).