CHAPTER VI

FINE GRAINED ACCESS CONTROL FRAMEWORK

6.1 Introduction

In Chapter V, we presented a trust-based mechanism for decisions based on the degree of dynamic delegated rights. In this chapter, we use the fuzzy-based control mechanism to arrive at a flexible, trust-aware fine-grained access framework for grid access control from the context of access permissions and resources allocation. The access policy (grant/deny or partial grant), is governed by the trust relationships among the grid domains. The quantification of trustworthiness of a resource site is done using fuzzy inference systems.

The rest of the chapter is organized as follows. In section 2, we discuss the fine-grained authorization requirements of a grid computing environment. Section 3, discusses the advantages of trust-aware grid sites and quantification of trust using Fuzzy Inference Process. We consider various categories of trust such as direct trust, asymmetric trust and indirect trust. In section 4, we propose a fine-grained authorization mechanism which can map the computed trust values to the corresponding access control decisions to ensure secured fine-grained resource access. We consider varying degrees of access like full access, partial access (fine grained) or nil access. And finally we summarize the work done as part of this chapter.

6.2 Fine-grained Authorization in Grids

The heterogeneous nature and independent administration of geographically dispersed resources in grid, demand the need for an access control mechanism built on fine-grained policies. We need to investigate the problem of fine-grained access control in the context of resource allocation in grids as it is the first and key step in developing access control methods specifically tailored for grid systems. Fine-grained access control system has proved to be effective for different applications
Elisa Bertino et al., [42] proposed a security component as part of a meta-scheduler service that can find the list of nodes where a user is authorized to run his/her jobs. The security component was designed in an effort to reduce the number of rules that need to be evaluated for each user request. Similarly, grid users get a higher flexibility in choosing the resources in which their jobs must execute.

A grid domain has its own local administrative policy and follows a global access policy for inter domain access. Implementation of a grid-wide access control policy, which can comply with the local policy is a real challenge. A grid would be valuable if it can contribute resources to the ultimate user and/or application across multiple domains in a secure way. Computational grid infrastructure softwares like the Globus Toolkit enable a user or application to identify and use the available resources irrespective of location and ownership. User programs/applications may contain malicious codes, which may act as a threat to the grid resources. On the other hand, grid resources, which are not trustworthy, may pose security problems to the user applications. Therefore security assurance must precede resource access. Almost all grid applications like scientific explorations, health care, government and business services have security and privacy requirements. Virtual organizations formulate and enforce policies for its members’ use of community resources, not only in terms of who can use what resource, but how it can be used.

The resource management is a real issue in a virtual environment like a grid where heterogeneous resources are geographically distributed and are held by individual domains having their own access policies. In such an environment, both the resource and the application need to be security-aware. If a resource provider site and a resource consumer site can establish a trust relationship among themselves, then mapping a resource request as per the trust value is possible. To analyze trust and take access control decisions based on it, we need to quantify trust. Several issues need to be considered in trust computation. We consider various categories of trust to arrive at a fine-grained access control model.

The fine-grained access control and authorization requirements in grids [149], [186] can be listed as follows:

- Combining policies from different sources. While outsourcing a portion of
the policy administration to the virtual organization (VO), the policy enforcement mechanism on the resource needs to be able to combine policies from the resource owner, the local policy and the grid-wide policy.

- Fine-grain control of access rights. For the VO to express the differences between how its user groups are allowed to use resources, the VO needs to be able to express policies regarding a variety of aspects of resource usage, not just grant/deny access.

- VO-wide management of jobs and resource allocations [187]. Normally, in a virtual organization, the jobs themselves are treated as resources. This poses a challenge with the jobs being dynamic and static methods of policy management not effective. Users may also initiate jobs that are independent of the VO - e.g. a user may have allocations on a resource other than through the VO and jobs invoked under this alternate allocation should not be subject to VO policy.

- Fine-grained and dynamic enforcement mechanisms. To implement policies, we need supportive enforcement mechanisms. Most resources today are capable of policy enforcement at the user level. This implies that all jobs run by the same user may be governed by the same policy though finer policies for different resources is ideal. Also, the enforcement mechanisms are typically statically configured through file permissions, job quota and the like. Therefore, we require an enforcement mechanism which can handle dynamic, fine-grain policies.

- Fine-grainedness based on roles (like initiator of jobs, manager of jobs etc)

- Fine-grained access for users with resource specific account [147]

- Fine-grained access based on trust relationships

6.3 Trust Aware Grid Sites

The fundamental concept of inter domain trust relationships is that trust works as an assurance that an entity will act in exactly the way we expect. Trust, when concerned with the authentication of grid entities is called identity trust. The Grid
Security Infrastructure (GSI) of Globus provides identity trust-related services. Behavior trust of an entity implies its trustworthiness. Shared grid resources, if infected with malicious programs, may damage the user application running on the grid platform. Trust assessment of a site involves the measurement of dependability, security, reliability and performance [125].

Let us consider a grid (G) as a virtual organization of multiple domains each having its own administrative and local access policies. Let $D_j$ be any domain in a virtual organization, which may have n number of members. Let $S_{D_j}$ be the site representing the $j^{th}$ domain. Then we can represent a grid as $G = \{D_j, \text{where}, j=1 \text{ to } n \}$. The trust relationship between two sites $S_{D_1}$ and $S_{D_2}$ is a linear and directional value $t_{12}$. We consider the following categories of trust relationships namely direct trust, asymmetric trust and transitivity trust to arrive at access control decisions.

6.3.1 Direct Trust

If a domain $D_i$ trusts another domain $D_j$ based on their direct interactions, then it is called direct trust $t_{ij}$ or $\eta$. It also reflects the reputation of $D_j$ as a trusted entity. The reputation of an entity is an expectation of its behavior based on other entities’ observations or information about the entity’s past behavior at a given time.

6.3.2 Asymmetric Trust

Sites in a grid environment may be either resource sites or user/application sites. A bi-directional trust relationship is to be established between these sites to ensure secured access to resources and to provide security assurance to the applications. If $t_{ij}$ is the trust from a resource site ($D_i$) to an application site ($D_j$) and $t_{ji}$ the trust from $D_j$ to $D_i$. $t_{ij}$ is not necessarily equal to $t_{ji}$. The unequal trust between two sites is termed asymmetric trust. To create two-way trust awareness, we compute the asymmetric trust.

6.3.3 Transitivity Trust

Transitivity property of trust determines whether trust can be extended outside the two domains between which it was established. Transitivity property can be used to extend trust relationships with other domains and non-transitivity trust to deny
trust relationships with other domains [20]. Transitive trust is an automatic trust association between parent and child domains in a forest of domains. Transitive trust relationships flow upward through a domain tree as it is formed, creating transitive trusts between all domains in the domain tree. Here, we extend this idea for establishing trust between two domains.

Execution of jobs in a grid environment generally requires a site to interact with several trusted intermediaries before actually accessing the resource site. Let us consider sites $D_i$, $D_j$ and $D_k$ in the neighborhood. If $D_i$ trusts $D_j$ and $D_j$ trusts $D_k$, then by using property of transitivity we can say that $D_i$ trusts $D_k$. The site $D_i$ need not directly compute the trust value on $D_k$. So transitivity trust is an indirect trust relationship. Transitivity trust reduces the overhead involved in computing trust for another time. It is also useful in the absence of data on past interactions. $D_j$ acts as a trusted intermediary between $D_i$ and $D_k$. Figure 32 depicts various trust paths for the above categories.

![Figure 32: Trust Paths across a Virtual Organization](image-url)
6.3.4 Quantification of Trust using FIS

In the previous chapter, we have seen that the linguistic evaluation of trust value is not enough to assess the security preparedness of a grid site. We need a mechanism by which we can translate this value to its corresponding numeric form. A fuzzy inference system allows us to provide input values as vague/imprecise linguistic terms, which, after fuzzification process, is converted to the corresponding fuzzy inputs.

6.3.4.1 Quantification of Direct Trust

The basic inputs which we use for trust quantification of a site are, job success ratio ($\alpha$), defense mechanisms ($\beta$) and well-behavedness ($\gamma$). To fine-tune these inputs, we use one more parameter called timestamp $t$, which indicates the time of computation of trust. This timestamp $t$ shows whether trust values have been updated or not.

Trust computation procedure between sites $i$ and $j$ starts with measurement of the trust inputs. Job success ratio $\alpha$ is the ratio of the number of jobs successfully executed ($m$) by $j$ to the total number of jobs submitted ($n$) at $j$ by $i$ in the recent past. i.e., $\alpha = \frac{m}{n}$. $\alpha$ of a site is an indication of its performance. This ratio is a numeric value. The Grid Resource Allocation Manager (GRAM) at site $i$ maintains relevant job data as well as status of the jobs.

The second trust variable is defense mechanisms $\beta$. $\beta$ represents the security preparedness of a grid site. Preferred defense mechanisms at a site are distributed firewalls, encrypted tunnels, packet filters, intrusion detection systems and traffic monitors. If site $S_{Di}$ has to compute trust on $S_{Dj}$, it sends a query to site $S_{Dj}$ asking for the availability of these mechanisms.

Let $DM^i (S_{Dj})$ represent the defense mechanism at a site ($S_{Dj}$), where $i = 1$ to $l$, and the value of $DM^i (S_{Dj})$ is a Boolean variable. We attach certain weights $w_i$ to each of them, based on their effectiveness. Then $\beta$ can be represented as,

$$\beta = \sum w_i \cdot DM^i (S_{Dj})$$

where $i=1$ to $l$ and we consider $l = 5$, which can be increased.

The weight $w_i$ is such that $l.b \leq w_i \leq u.b$, where we assume that $l.b = 0.3$ and $u.b = 0.9$, (the lower bound and upper bound values) depending on the defense
mechanisms. Incorporating additional attributes like security levels and efficiency adds to the accuracy of measurement of $\beta$. Various defense mechanisms possible in the grid environment include distributed firewalls, encrypted tunnels, packet filters and intrusion detection mechanisms.

The third trust input well-behavedness $\gamma$ of a site during past interactions is a pointer to its access intensions. A well-behaved site never intends for unauthorized tampering of resources through its applications/task requests. Since $\gamma$ is a natural term, we quantify it and analyze it through fuzzy approach. It can be measured with the help of data from intrusion detection systems at site $i$. The data of reported intrusion attempts at $i$ from $j$ can be analyzed using data mining techniques. These patterns indicate the well-behavedness of $j$ with respect to $i$.

We consider three varying degrees of inputs namely low, medium and high (i.e. from 0 to 1 on a numeric scale). Similarly, for the output variable direct trust, we consider three levels low, medium and high (from 0 to 1). An inference rule consists of at least one antecedent and one consequent. For example, an inference rule can be: if direct trust is low then access-deny. Here, direct trust is low is the antecedent and access-deny is the consequent. We associate a weight attribute with the rule. Each antecedent has one linguistic variable, one linguistic term, and one logical operator. Similarly, each consequent also has a linguistic variable, a linguistic term and a weight attribute. We have computed and quantified the trust values using a two-stage fuzzy inference process.

For the domains $D_i$ and $D_j$, the trust value may increase or decrease over a period of time based on the interactions between the sites. This means that periodic update of trust is necessary to reflect the dynamic access environments. We update trust values based on the time stamp $t$ attached to it. Thus direct trust $\gamma$ is a function of $t$ expressed as $\eta_t = f(\alpha_t, \beta_t, \gamma_t)$.

6.3.4.2 Quantification of Asymmetric Trust

We have seen earlier that the trust values $t_{ij}$ and $t_{ji}$ between two sites $i$ and $j$ need not be the same due to the differences in the performance of sites resulting in different job success ratio values, non-uniformity in the defense mechanisms set up at the two domains and different degrees of behavioral patterns or well-behavedness
from one site to another. Thus trust is a vector with a magnitude and direction.

6.3.4.3 Quantification of Transitive Trust

In the previous two categories of trust, the trust level that a resource/application site holds on another is based on the direct relationship/interactions between them. In wide area systems such as grids, a site needs to interact with more than one site to complete a job execution. Since quantification of trust between every pair of domains is not practical, we can consider transitive trust. Scalable trust allows certain level of trust to travel to large number of nodes (or entities) and still be able to maintain maximum level of trust during its travel in a specific time frame. This propagation of trust is called transitivity trust.

A virtual organization comprises multiple grid domains. It follows a certification path. A “root” CA (RCA) issues certificates to its subordinate CAs and these CAs issue certificates to their next level subordinate CAs and so on. The “root” CA is regarded as the trustworsthesi in the hierarchical certification path. Every subordinate must know the root CA’s public key. Any subordinate’s certificate may be verified by following the certification path till the verifier reaches the root CA (e.g. a common trusted CA or the CA who issued the subordinate’s certificate). Figure 33 shows the scalability of trust in a virtual organization.

![Figure 33: Scalability of Trust in a Virtual Organization](image-url)
Considering the domains $D_i$, $D_j$ and $D_k$ in a virtual domain with RCA as the root CA and having their own local CAs (subordinate CAs), then “$D_j < D_i >, D_k < D_j >$” means $D_j$ signed the trust credential of $D_i$, and $D_j$ signed the certificate of $D_k$. Similar to the X.509 Public Key Infrastructure (PKI), the level of trust provided by certificates “$D_j < D_i >, D_k < D_j >$” is the same as “$D_k < D_i >$”. In other words, possession of “$D_j < D_i >, D_k < D_j >$” provides the same capability as “$D_k < D_i >$” (i.e. $D_j < D_i >, D_k < D_j >= D_k < D_i >$). Figure 34 shows the transitivity trust path in a V.O. We quantify transitivity trust using the relation, Figure 34: Transitivity of Trust

$\text{Trust}(i, j) \land \text{Trust}(j, k) \Rightarrow \text{Trust}(i, k)$ provided the sites $i$, $j$, $k$ have a common root CA. Here, $\land$ implies the composite relation AND between the trust values Trust $(i, j)$ and Trust $(j, k)$. Site $S_{Di}$ establishes a trust relationship with $S_{Dk}$ through a trusted intermediary. As direct trust relationship is more authentic, we consider a degrade value for transitivity trust. Transitivity trust evaluation $t_{ik}$ requires the trust values $t_{ij}$ and $t_{jk}$ as its inputs.

Figure 35 shows the computation of transitivity trust from the trust values trust $(i, j)$ and trust $(j, k)$. As transitivity trust is an indirect security assessment, we apply a reduction factor $w$ on trust$(i, k)$. i.e., $t_{ik} = \text{trust}(i, k) * (1 - w)$

In Figure 35, $t_{ij}$ is 0.794 and $t_{jk}$ is 0.832. The composite operation $\land$ provides
the value of $t_{ik}$ as 0.744. We assume $w$ based on the application and the security level of the resource accessed. For security critical applications/resources a large $w$ such as 0.9 can be adopted. For stable and relatively low security-sensitive applications/resources, a small $w$ such as 0.3 is adopted. In general cases one can set $w$ in the range (0.7-0.8). There is another way in which we could arrive at the transitivity trust value. Instead of site $D_i$ trusting $D_k$ through only $D_j$, we can have $D_{j1}, D_{j2}, \ldots, D_{jp}$. There can be various possibilities like $t_{ik} = \text{Min}\{t_{ij} \wedge t_{ik}\}$, where $j=j_1, j_2, \ldots, j_p$ or we can also have $\text{Max}$, $\text{Avg}$ relations depending on the trust requirement. This is depicted in Figure 36.

Transitivity trust values help in trust integration and propagation across multiple grid domains. Domain $D_j$ acts as the trusted third party or intermediary between $D_i$ and $D_k$.

**6.4 Fine-Grained Access Control Framework using Trust Values**

Fine-grained access control of grid resources is important for grid security. We need to express and enforce fine-grain policies on the usage of resources. These can no longer be expressed by simple access control as we have to specify exactly what fractions or configurations of resource may be used by a given entity. The targets of a fine-grained policy can be resource allocation and usage, job management, application services etc. The fine-grained access specifications for a resource among the VO participants is provided by the VO. The VO has two primary classifications of its members: one group has the role of developing, installing and
Figure 36: Constructing Transitive Trust
debugging the application services used by the virtual organization to perform their scientific computations and another group which does analysis using the application services. The first group may need a large degree of freedom for the applications they need to run (e.g. compilers, debuggers, the applications themselves) in order to debug and deploy the VO application services, but would only be consuming small amounts of traditional computing resources (e.g. CPU, disk and bandwidth) in doing so.

The second group may need the ability to consume large amounts of resources in order to perform scientific computations, but should only be doing so using application services approved by the VO. Also to specify policies such that certain users may use more/less resources than others and that certain applications may consume more or less resources than others, we need a fine grained grid access control mechanism.

In traditional access control mechanisms, we have only the permit - deny kind of access decisions. In trust aware grid systems, we can incorporate certain degree of fine grained access based on trust values, the access control being full, nil or partial. Though the Grid Security Infrastructure of GT4 security framework supports various authorization schemes such as self authorization, gridmap file, identity-based authorization, host-based authorization etc, there is no provision made for fine-grained authorization in any of these schemes. The client identity is obtained from the credential used by the application to contact the service. The resource owner identity is obtained from the credentials configured for the resource, service, or container, depending on availability, in that order of precedence.

6.4.1 Fine-grained Authorization Engine

The VO may wish to specify finer-grain policies that certain users may use more or less resources than others. These policies may be dynamic and change over time. In addition to the policy on the resource utilization, the VO may wish to manage jobs running on VO resources. For example, users often have long-running computational jobs using VO resources and the VO often has short-notice high-priority jobs that require immediate access to resources. This requires suspending existing jobs to free up resources; something that normally only the user that submitted the
job has the right to do. Since going through the user who submitted the original job may not always be an option, the VO may give a group of its members the ability to manage any job using VO resources so that they can instantiate high-priority jobs on short notice.

We propose a fine grained access control system which consists of an authorization filter in the form of fine-grained access control engine. Based on the quantified trust values, this engine intercepts every access request and evaluates it against the restrictions to service/resource accessibility. Thus the resource request may be rejected; be allowed as it is or may be filtered and executed in a modified form. The filtering of a request may involve elimination of some of its attributes that the client is not allowed to specify or additionally needs to specify. Figure 37 shows the proposed fine-grained authorization mechanism.

Any of the above authorization schemes can use the quantified trust value (for different contexts) to generate access decisions. We integrate our fine-grained authorization engine the XACML framework. The implementation of a similar integration (of Role-Based Model with XACML framework) has been explained in Chapter VII of this thesis. At the Policy Decision Point or PDP [34], the authorization or access decisions are implemented. Rules form the most important sub-component of any access policy. Rules are essentially conditions that evaluate to
permit, deny or partially permit access.

6.4.2 Rule - Creation for Access Decisions

A fine-grained policy framework is formulated in three stages namely, creation of the rule target, definition of the effect and creation of the condition. A rule target is an entity on which the rule works. In a grid environment, a rule target is a grid resource/application. Once the rule target is created, it needs to be authenticated. In a trust aware grid environment, the trust levels are used as credentials. The effect of the rule implies the type of access decision namely permit, partial permit or deny. If the associated condition of a rule is true, then the effect of the rule is positive.

6.4.3 Trust Credential

The quantified trust value is used as an attribute for constructing the trust credential. The general format of the trust credential as per the XML syntax is as given below. Every trust update is reflected by incrementing the version number of the trust credential. The trust credential contains the policy decision which governs the fine-grained access of grid resources. The elements of this format include the granter name, the grantee, the trust type, trust level (trust value), validity period, the access type and the resource being accessed.

```
<trustCredential>
  <version>1.0</version>
  <grantor>grantorDN</grantor>
  <grantee>granteeDN</grantee>
  <trustType>type</trustType>
  <trustLevel>level</trustLevel>
  <validity>
    <notBefore>time</notBefore>
    <notAfter>time</notAfter>
  </validity>
  <accessType>accessval</accessType>
  <resourceName>name</resourceName>
```
6.4.4 Mapping Trust Values to Access Decisions

The trust value can be at any of the levels namely low, medium or high. We suggest an access control model, which maps the trust value to its corresponding access decision based on the trust values. The following are some of the sample if-then-elseif-else rules used for fuzzy inferencing. The different categories of trust are taken care of by weighted trust quantification and the trust input variables in each case.

- *if direct trust is low then denyAccess elseif direct trust is medium then partialAccess else grantAccess*
- *if asymmetric trust is low then denyAccess elseif asymmetric trust is medium then partialAccess else grantAccess*
- *if transitivity trust is low then denyAccess elseif transitive trust is medium then partialAccess else grantAccess*

Figure 38 shows the initiation process of resource access after mapping of trust values with the access decisions.

![Figure 38: Initiation of Access](image-url)
The overall trust aware access architecture for a virtual organization is as in Figure 39. We present the framework for any two grid domains, which could be extended to any number of domains $n$. Domains $i$ and $j$ each contain a trust management module (TMM). This module establishes trust relationships between the two domains by interacting on a set of trust inputs and then quantifies trust. After this process, the trust value is passed on to the PDP (Policy Decision Point) module, where the trust values are mapped with access privileges. PEP or the Policy Enforcement Point, acts as the access control enforcement mechanism, which implements the access decision taken at the PDP. Once the access is granted, the resource can be accessed. The working of PDP and PEP have been discussed in Chapter III, section 3.3.

![Figure 39: Trust Aware Access Control](image.png)

6.5 Chapter Summary

The main contribution of this chapter is the proposed trust-aware fine-grained access control framework for grid environment. We quantify various categories of trust namely direct trust, transitive trust and asymmetric trust using fuzzy inference process and map these values to access control decisions. This framework can be incorporated with the legacy access control mechanisms. Use of transitivity trust reduces the trust computation time and is best suited for scalable virtual
environments like grids. A trust aware grid environment can deal with dynamic access requests to resources more effectively. Ascertaining that an entity with a particular identity or set of attributes has the permission to perform a particular action on a particular resource is a key issue. Trust aware grid sites can secure their resources and applications in an efficient way.