Future Generation Computer Systems 29 (2013) 1909–1918

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Grid Authorization Graph

Mustafa Kaiiali [a,c,*], Rajeev Wankar [a], C.R. Rao [a], Arun Agarwal [a], Rajkumar Buyya [b]

[a] *Department of Computer and Information Sciences, University of Hyderabad, Hyderabad, India*

[b] *Cloud Computing and Distributed Systems (CLOUDS) Laboratory, Department of Computer Science and Software Engineering, The University of Melbourne, Australia*

[c] *Department of Computer Engineering, Mevlana University, Konya, Turkey*

HIGHLIGHTS

- A brief overview of access control mechanisms used in grid systems is illustrated.
- The limitations of the Hierarchical Clustering Mechanism (HCM) are highlighted.
- The Grid Authorization Graph (GAG) is introduced to encounter all HCM limitations.
- The GAG Generator Algorithm is illustrated to build GAG decision graph.
- Embedding GAG in GT4 authorization framework is finally discussed.

ARTICLE INFO

Article history:
Received 24 July 2012
Received in revised form
21 March 2013
Accepted 6 April 2013
Available online 22 April 2013

Keywords:
Grid computing
Grid authorization
Access control
Hierarchical Clustering Mechanism
Grid Authorization Graph

ABSTRACT

The heterogeneous and dynamic nature of a grid environment demands a scalable authorization system. This brings out the need for a fast fine-grained access control mechanism for authorizing grid resources. Existing grid authorization systems adopt inefficient mechanisms for storing resources' security policies. This leads to a large number of repetitions in checking security rules. One of the efficient mechanisms that handle these repetitions is the *Hierarchical Clustering Mechanism* (HCM). HCM reduces the redundancy in checking security rules compared to the *Brute Force Approach* (BFA) as well as the *Primitive Clustering Mechanism* (PCM). Further enhancement is done to HCM to increase the scalability of the authorization process. However, HCM is not totally free of repetitions and cannot easily describe the OR-based security policies. A novel *Grid Authorization Graph* (GAG) is proposed to overcome HCM limitations. GAG introduces special types of edges named "*Correspondence Edge*"/"*Discrepancy Edge*" which can be used to entirely eliminate the redundancy and handle the cases where a set of security rules are mutually exclusive. Comparative studies are made in a simulated environment using the *Grid Authorization Simulator* (GAS) developed by the authors. It simulates the authorization process of the existing mechanisms like BFA, PCM, HCM and the proposed novel GAG. It also enables a comparative analysis to be done between these approaches.

1. Introduction

Grid computing is concerned with a shared and coordinated use of heterogeneous resources belong to distributed virtual organizations to deliver nontrivial quality of services [1]. In grids, security has a major concern [2]. The heterogeneity, massiveness and dynamism of grid environments complicate and delay the authorization process. This brings out the need for a fast and scalable fine-grained access control mechanism to cater to grid requirements.

Currently, the main focus in the literature is on the way to write the resource's security policy, either using a standard specification language like SAML/XACML as used in VOMS [3] to provide the interoperability property [4], or it can be specific to a particular authorization system (as in Akenti [5]). Furthermore, they describe the authorization process, either to be centralized (push model [6] as VOMS and CAS [7,8]), or decentralized (pull model [6] as PERMIS [9] and Akenti [10]). Some systems adopt transport level security rather than message level security as the latter involves slow XML manipulations, which make adding security to grid services a performance bottleneck [11].

Current grid authorization systems seldom look at the way in which they store and organize the resources' security policies in order to work more effectively. There is no well-defined data structure to store and manage the security policies to provide a quick response to the user. There are not so many articles that

* Corresponding author.
E-mail addresses: mustafa_kaiiali@ieee.org (M. Kaiiali), wankarcs@uohyd.ernet.in (R. Wankar), crrcs@uohyd.ernet.in (C.R. Rao), aruncs@uohyd.ernet.in (A. Agarwal), raj@csse.unimelb.edu.au (R. Buyya).

0167-739X/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.future.2013.04.010

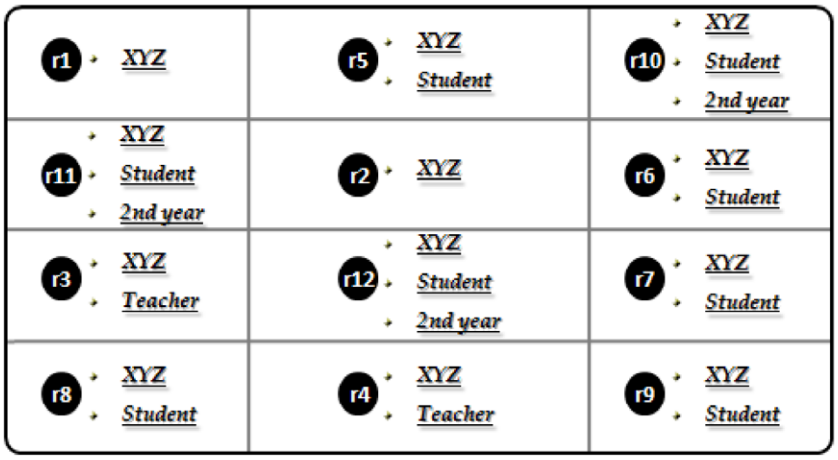

Fig. 1. Example of the Brute Force Approach access control mechanism.

have been published so far, and the most representative methods are the *Brute Force Approach* (BFA) [12] and the *Primitive Clustering Mechanism* (PCM) [13–15].

Every resource in a grid has its own security policy, which may be identical or quite similar to other security policies of some other resources. This fact motivated us to cluster the resources which have similar security policies in a hierarchical manner based on their shared security rules. The authorization system can built a hierarchical decision tree to find User Authorization Resource Group (*UARG*). The *Hierarchical Clustering Mechanism* (HCM) [16–19] was a step in that direction to provide a more fine-grained clustering at multi-levels.

This paper highlights the limitations of HCM and introduces the *Grid Authorization Graph* (GAG) to overcome these limitations and to further enhance the authorization process by adopting new tools which cannot be adopted in HCM.

Rest of the paper is organized as follows: Section 2 gives a brief description of HCM. Section 3 discusses the proposed GAG and shows how the drawbacks of HCM are addressed. GAG Generator Algorithm is proposed in Section 4. Section 5 explains how GAG components can be embedded in current authorization architecture like GT4. Experiments with results are discussed in Section 6. Section 7 concludes and suggests future work.

2. A brief description of the Hierarchical Clustering Mechanism (HCM)

Consider the following definition:

- Let $\mathbf{R} = \{r_j | j = 1, \ldots, k\}$ be the set of grid resources.
- Let $\mathbf{SR} = \{sr_j | j = 1, \ldots, l\}$ be the set of security rules.
- Then for each resource $r_j \in \mathbf{R}$ there will be a corresponding security policy $\mathbf{SP_j} \subseteq \mathbf{SR}$.

If a user wants to access resource r_j then he has to satisfy all the security rules of $\mathbf{SP_j}$. Let us now consider the following example:

A grid environment has 12 resources $\mathbf{R} = \{r_1, r_2, \ldots, r_{12}\}$ and four security rules $\mathbf{SR} = \{sr_1, sr_2, sr_3, sr_4\}$ where:

- sr_1 requires the user to be from <u>***XYZ***</u> *University*.
- sr_2 requires the user to have a <u>***teacher***</u> role.
- sr_3 requires the user to have a <u>***student***</u> role.
- sr_4 requires the user to be in <u>***2nd year***</u>.

All the 12 resources are deployed with the following security policies:

- r_1, r_2 require the user to be from <u>***XYZ***</u> *University* to be able to access them. So $\mathbf{SP_1} = \mathbf{SP_2} = \{sr_1\}$.
- r_3, r_4 require the user to be from <u>***XYZ***</u> *University* and to have a <u>***teacher***</u> role in order to access them. So $\mathbf{SP_3} = \mathbf{SP_4} = \{sr_1, sr_2\}$.
- r_5, r_6, r_7, r_8, r_9 require the user to be a <u>***student***</u> in <u>***XYZ***</u> *University*. So $\mathbf{SP_5} = \mathbf{SP_6} = \mathbf{SP_7} = \mathbf{SP_8} = \mathbf{SP_9} = \{sr_1, sr_3\}$.
- r_{10}, r_{11}, r_{12} require the user to be a <u>***2nd year***</u> <u>***student***</u> in <u>***XYZ***</u> *University*. So $\mathbf{SP_{10}} = \mathbf{SP_{11}} = \mathbf{SP_{12}} = \{sr_1, sr_3, sr_4\}$.

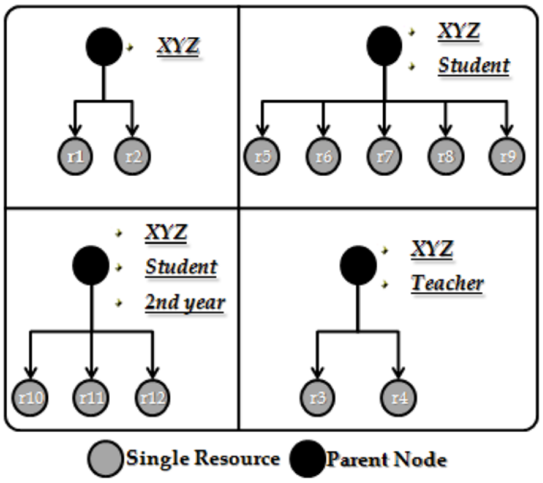

Fig. 2. Example of the Primitive Clustering Mechanism.

BFA for the proposed example stores the security policies as shown in Fig. 1. We have 12 security policies each of them consists of a set of security rules, all together need to be checked to find the *UARG*. Redundancy of BFA is obvious as we have many redundant security policies like $\mathbf{SP_5}$, $\mathbf{SP_6}$, $\mathbf{SP_7}$, $\mathbf{SP_8}$ and $\mathbf{SP_9}$.

PCM reduces BFA redundancy by clustering the resources which have identical security policies. Fig. 2 shows how PCM stores the security policies of the proposed example. It is obvious that the number of security policies to be checked is reduced from 12 security policies to only four security policies.

PCM removes the redundancy of checking identical security policies, but it cannot remove the redundancy of checking identical security rules. In other words, it avoids checking identical security policies **SP**s more than once; since each security policy **SP** is a set of security rules, the security rule (sr) level of redundancy is still prevailing in PCM. As an example, <u>***XYZ***</u> security rule has to be checked four times.

HCM [16] clusters the resources in parent nodes based on their shared security policies, as in PCM. However, it also achieves a hierarchical clustering of these parent nodes based on their shared