Weighted Grid Authorization Graph (WGAG)


Published in:
Proceedings of the The 6th International Conference on Communications and Networking (ComNet'2017)

Document Version:
Peer reviewed version

Queen's University Belfast - Research Portal:
Link to publication record in Queen's University Belfast Research Portal

Publisher rights
© 2017 IEEE
This work is made available online in accordance with the publisher's policies. Please refer to any applicable terms of use of the publisher.

General rights
Copyright for the publications made accessible via the Queen's University Belfast Research Portal is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The Research Portal is Queen's institutional repository that provides access to Queen's research output. Every effort has been made to ensure that content in the Research Portal does not infringe any person's rights, or applicable UK laws. If you discover content in the Research Portal that you believe breaches copyright or violates any law, please contact openaccess@qub.ac.uk.

Download date: 28. Dec. 2018
Weighted Grid Authorization Graph (WGAG)

Sara Namane  Mustafa Kaiiali  Nacira Ghoualmi
Networks and Systems  Research Fellow,  Networks and Systems
Laboratory, Badji  Queen's University  Mokhtar University,
Mokhtar University,  Belfast,  Annaba, Algeria
Annaba, Algeria  United Kingdom.  Qndaba, Algeria
snaamane@lrs-annaba.net  mustafa_kaiiali@ieee.org  ghoualmi@yahoo.fr
Rajeev Wankar  C.R. Rao  Sakir Sezer
School of Computer and  School of Computer and  Centre for Secure
Information Sciences  Information Sciences  Information Technologies
(SCIS), University of  (SCIS), University of  (CSIT), ECIT, Queen's
Hyderabad, Hyderabad,  Hyderabad, Hyderabad,  University Belfast, United
Hyderabad, India  India  Kingdom
wankarc@uohyd.ernet.in  rao@scis.uni hyderabad.ernet.in  sezer@csit.qub.ac.uk

Abstract—grid computing is a collection of distributed computing resources. These resources are available over a local or wide area network that appears to an end user or application as a virtual computing system. The vision is to create virtual dynamic organizations through secure and coordinated resource-sharing among individuals and institutions. Access control to these resources is a problem difficult to manage. How to store and manage the security policies of such a system plays a key role. If these policies are not properly stored, the response time to access control requests will be dramatically increased. Grid Authorization Graph (GAG) was proposed to improve the authorization efficiency by eliminating the redundancy in checking security rules. This article proposes the Weighted Grid Authorization Graph (WGAG) as an enhancement to GAG which further improves the authorization efficiency by avoiding a large number of security rule checking. Finally, as proof of concept, we implement the WGAG simulator where simulations were done. The obtained results show that the proposed model can effectively reduce the complexity of security rule checking and gave better results than GAG.

Keywords—Grid computing; Access control; Grid authorization; HCM; GAG.

I. INTRODUCTION

Grid computing was introduced to allow multiple institutions to share their resources across multiple locations with a large number of users in a variety of organizations [1]. The security of such systems is a crucial issue. Not only the data has to be secured but also the resources and calculations must be protected from inadequate access [2]. Furthermore, the dynamism and multi-domain characteristics of grid environments have created access control related challenges [3].

The traditional techniques of authorization are based on access control lists. This technique allows user whose name appears in the list to reach the grid based on the privileges associated with his/her name [4]. In this model, the resource providers have to maintain authorization decisions for every user, which is a very time consuming and non-scalable solution.

To avoid the disadvantages of the preceding technique, the Role Based Access Control (RBAC) [6] has been introduced and replaced the traditional identity based lists. In RBAC model, the privileges are assigned to roles and roles to users; this made the management of the security policies more flexible. However, the concept of the hierarchy of roles in the RBAC model is a little ambiguous because in general it is incorrect to consider that the hierarchy of roles corresponds to the organizational hierarchy. In recent years, organizations start adopting approaches that pass attributes for authorization instead of passing user’s credentials. This model is named as: Attribute Based Access Control (ABAC) [7]. ABAC makes it possible to represent the policies of an organization with more detailed criteria than RBAC.

In [10] the authors concentrate on how to adopt an efficient structure to store security policies in grid environments, they spoke about how security policies were represented and checked using the Brute Force Approach (BFA). In this mechanism the entire security policies are required to be checked in order to find the user’s authorized resource group (UARG) which leads to huge repetition. Furthermore, authors think to cluster the resources which have identical security policies to reduce redundancy, it was done by the Primitive Clustering Mechanism (PCM), the authors observed that PCM removes the redundancy of checking the identical security policies but it cannot remove the redundancy of checking identical security rules. To avoid this redundancy, HCM was proposed. It considers the PCM’s parent node’s information as a data to generate a hierarchical clustering of the parent nodes themselves depending on their shared security rules. In [11, 12] authors discussed the limitations of HCM and noticed that it cannot represent the or-based security policies then they propose the Grid Authorization Graph (GAG), a decision graph derived from HCM by embedding special edges named “correspondent edges”. These can be used to entirely eliminate the redundancy of rules checking. However, some security rules do not need to be verified from the beginning.

This paper proposes a novel enhancement to GAG, so called as the Weighted Grid Authorization Graph (WGAG) which further improves the authorization process. The rest of this paper is organized as follows: section II describes the proposed architecture; Section III presents the WGAG algorithm. In section IV, experiments are presented to evaluate the effectiveness of the proposed model. Conclusion and future work are given in section V.
A grid environment usually consists of more than one domain in a hierarchical fashion [5]. Each domain has an administrator that manages the whole grid scenario for that domain by adding/editing resource access policies. These policies are generally written in XML file. Previous section gave more important mechanisms proposed to store security policies and manage them. The more efficient result was given by grid authorization graph (GAG) [11]. As each edge in graph can have a weight, one can think of assigning a weight (non-negative integer) to every edge in the decision graph which reflects the importance degree of the security rule which the edge emerges from. Then an attribute named “classification level” can be assigned to every resource which is a numerical value that equals the weight of the shortest path from the root node to the resource’s node (sum of weights of edges), as shown in the (Fig.2) every user in the system has an attribute named “security clearance” derived out of user’s set of roles in the system, one can avoid parsing the decision graph for authorizing a user Ui whose security clearance (USC(Ui)) is less than the classification level of resource rj (RCL(rj)), that is (USC(Ui)) < (RCL(rj)). Thus the developed weighted GAG with the following details:

- Let SR = { srj | j=1 … l } be the set of all security rules.
- Let IDR: SR → N: IDR (srj) = the Importance Degree of security rule srj.
- Let G (V; E) be the Grid Authorization Graph GAG, where V is the set of vertices (security rules) and E is the set of edges.
- Let W: E → N: W (eij) = IDR (srj) be a function defines edges’ weights in GAG. Where srj is the rule which the edge eij emerges from.
- Let R = {rj}|j=1…n | be the set of grid resources.
- Let SP: R → N: SP (rj) = weight of the shortest path to resource rj.
- Let RCL: R → N: RCL (rj) = SP (rj) be a function maps each resource to its classification level.
- Let Role= |role| i=1…s | set the bit of user’s roles
- Let IDR: Roles → N: IDR (rolej) = the importance degree of rolej.
- Let U = {Ui | i= 1…m} be the set of all users.
- Let UR ⊆ U× Roles: be the set of relations of user to role assignments.
- Let USR: U → Roles : USR(Ui) = {rolej | (ui, rolej) ∈ UR} be a function defined from UR mapping each user to a set of roles.
- Let USC: U → N: USC (U) = \sum_{rolej} IDR(rolej) be a function maps each user to his security clearance.

Let us consider now the following example: A grid environment has 5 resources R = {r1, r2, r3, r4, r5}, and 3 security rules SR = {sr1, sr2, sr3} with their importance degree (Illustrated in table 1)

<table>
<thead>
<tr>
<th>Security rule</th>
<th>Importance degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sr1</td>
<td>4</td>
</tr>
<tr>
<td>Sr2</td>
<td>1</td>
</tr>
<tr>
<td>Sr3</td>
<td>9</td>
</tr>
</tbody>
</table>

The five resources have the following security policies:

- r1, r2 require sr1.
- r3 requires sr1 and sr2.
- r4 requires sr1 and sr2 and sr3.
- r5 require sr1 and sr2 and sr3.

The proposed architecture is illustrated in (Fig.1), the authorization server contains the following components: policy information point (PIP) [4], the Policy Enforcement point (PEP) [4], the Policy decision point (PDP) [4] and some others modules were added to make the framework compatible with the Weighted Grid authorization Graph (WGAG) like:

**A. Resource group authorization mode**

When a user raises an access request, the PEP intercepts the request and propagates it to the PDP; the request is kept in a queue in PDP. The RAP is a simple action listener which listens on the PDP queue [11]. Once the request is enrolled into the queue, RAP picks up the request; fetches the authorization attributes of correspondent subject from PIP as the user security clearance CL(Ui) afterwards the RAP sends the request to the WGAG search engine, this last will parse the weighted grid authorization graph and gives the group of resources that satisfy : CL(rj) ≤ SC(Ui). Finally, the group of

**II. PROPOSED ARCHITECTURE**


Security policies of resources are submitted by the administrator using the SAML specification language [8] or XACML [9], an XML parser [11] is required to browse the XML file and give as a result a security table, as shows the table 2 is the result of security policies previously cited. The WGAG generator engine is responsible to build the proposed weighted grid authorization graph (WGAG) out of the security table generated by the XML Parser. Practically, it is a direct implementation of WGAG generator algorithm whose pseudo code is shown in section III. the vertices of the graph represent the security rules, but the weighted edges will have the importance degree of security rules like value, the leaves of the graph represent the resources, then for each resource ri the graph have to be parsed to calculate the shortest path from root to ri, the obtained value is taken as classification level of ri (Illustrated in Fig.2). Following this it maintains the output decision graph in WGAG data base to be used by WGAG search engine.

TABLE 1. SECURITY RULES’ IMPORTANCE DEGREE

<table>
<thead>
<tr>
<th>Security rule</th>
<th>Importance degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sr1</td>
<td>4</td>
</tr>
<tr>
<td>Sr2</td>
<td>1</td>
</tr>
<tr>
<td>Sr3</td>
<td>9</td>
</tr>
</tbody>
</table>


Security policies of resources are submitted by the administrator using the SAML specification language [8] or XACML [9], an XML parser [11] is required to browse the XML file and give as a result a security table, as shows the table 2 is the result of security policies previously cited. The WGAG generator engine is responsible to build the proposed weighted grid authorization graph (WGAG) out of the security table generated by the XML Parser. Practically, it is a direct implementation of WGAG generator algorithm whose pseudo code is shown in section III. the vertices of the graph represent the security rules, but the weighted edges will have the importance degree of security rules like value, the leaves of the graph represent the resources, then for each resource ri the graph have to be parsed to calculate the shortest path from root to ri, the obtained value is taken as classification level of ri (Illustrated in Fig.2). Following this it maintains the output decision graph in WGAG data base to be used by WGAG search engine.
resources is sent to RAP, then to PEP (example shown in Fig.4).

B. One resource authorization mode

When a user Ui raises an access request to a special resource ri, the PEP intercepts the request and propagates it to the PDP, the request is kept in a queue in PDP. The RAP picks the request, fetches the authorization attributes of correspondent subject from PIP, as user security clearance CL(Ui) then the RAP sends the request to the WGAG search engine, this last will parse the weighted grid authorization graph for the special resource ri and gives accepted access as result if: CL (ri) ≤ SC (Ui) else denied, the answer is sent to RAP, then to PEP.

<table>
<thead>
<tr>
<th>ri</th>
<th>Sr1</th>
<th>Sr2</th>
<th>Sr3</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>r2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>r3</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r4</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>r5</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

III. WEIGHTED GRID AUTHORIZATION GRAPH GENERATOR ALGORITHM

**Inputs**: Resources’ Security Table, Security Rules Importance Degree Table

**Outputs**: Weighted Grid Authorization Graph (WGAG)

**Variables**:
1. **SRV**: a vector of all security rules.
2. Each node (N) in the graph is a structure of 3 fields:
   - The security rule sr
   - An interim security table ST
   - Undirected correspondent edge from the node to the representative sr cell in SRV.
3. Each edge (e) in the graph has a weight w that is an integer representing the degree of importance of the security rule sr in security rules importance degree table IDT.
4. Each resource ri in the graph is a structure of 2 fields:
   - Label
   - CL, an integer that represent the classification level of the resource.

**Begin**

**Step 1**:
- Initialize the decision tree by root node (N) with “ROOT” as label of security rule.
- Build the security table ST which represents the entire Security policy of the system; assign it as a security table (ST) property of the root node.
- Build the importance degree table IDT which represents the importance degree of each security rule. Execute the step 2 for the root node.

**Step 2**:
- Add each resource ri whose correspondent row in N’SST has 0 cells, as a child resource to N.
- Sum the cells of each column of N’SST and refer it as count.
- Choose the security rule srj with the highest count.
- Divide ST into two tables, excluding the jth column as the following:
  - The first table T1 contains the rows of resources which demand srj (each row whose jth cell >0).
  - The second table T2 contains the rows of resources which do not demand srj (each row whose jth cell =0).
- Add a left child node LCN to N with srj as a security rule (sr) and T1 as the security table ST; the weight of srj in IDT as weight of LCN. Let the correspondent edge of LCN refers to srj cell in SRV
- Add a right child RCN to N with NULL as security rule sr, T2 as security table ST and 0 as weight of RNC. Let the correspondent edge of RNC refers to Null.

**Step 3**: Repeat step 2 for each child until a child with empty security table is reached.

**Step4**: for each resource ri in the graph a shortest path from the root node to this resource is calculated and added to CL of ri.

**Step 5**: Prune the graph at nodes labeled ROOT. Erase all interim security tables (ST) to free space.

IV. SIMULATIONS AND RESULT

For grid environment of 100 resources and 8 security rules, 100 different authorization processes have been initiated. For each authorization process, the posterior analysis of GAG and WGAG has been done and depicted in Fig.3 (X axis is for the authorization process number (experiment N°) and Y axis is for the authorization complexity (number of checked security rules)).
As shown in Fig. 3, we can find that WGAG complexity is always lower than GAG complexity, this is due to the use of classification level and security clearance attributes in WGAG model, because when one user has a security clearance inferior than resource classification level, the graph will not be parsed at all in case of WGAG, however GAG parse all the graph to check if resource security policy is satisfied by user roles.

One of most remarkable points of the result is that WGAG complexity reaches zero sometimes while GAG complexity is always superior than zero, this is due to the use of one resource authorization mode, because when user security clearance is inferior than resource classification level, there is no security rule to be checked in WGAG case, but using GAG implies checking one security rule at least.

So the analyze and simulations indicate that WGAG can effectively reduce the complexity of security rules checking, besides we can say that WGAG improves the access control process and gives better results than GAG.

V. CONCLUSION AND FUTURE WORKS

In this article, various mechanisms used for security policies storage and management in grid computing environment were compared, to enhance grid access control process, weighted Grid Authorization Graph (WGAG) is proposed, it is derived from Grid Authorization Graph (GAG). Summing up the results, it can be concluded that while GAG eliminates the redundancy in checking security rules, WGAG security rules checking can be equal to zero. In our future research we intend to concentrate on cross domain access control architecture based on WGAG.

REFERENCES


