Weighted Grid Authorization Graph (WGAG)

Weighted Grid Authorization Graph (WGAG)

Sara Namane
Networks and Systems Laboratory, Badji Mokhtar University, Annaba, Algeria
snaamane@lrs-annaba.net

Mustafa Kaiiali
Research Fellow, Queen's University Belfast, United Kingdom.
mustafa_kaiiali@ieee.org

Nacira Ghoualmi
Networks and Systems Laboratory, Badji Mokhtar University, Annaba, Algeria
ghoualmi@yahoo.fr

Rajeev Wankar
School of Computer and Information Sciences (SCIS), University of Hyderabad, Hyderabad, India.
wankarcs@uohyd.ernet.in

C.R. Rao
School of Computer and Information Sciences (SCIS), University of Hyderabad, Hyderabad, India
crrcs@uohyd.ernet.in

Sakir Sezer, Research Director,
Centre for Secure Information Technologies (CSIT), ECIT, Queen's University Belfast, United Kingdom

***Abstract*— grid 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 one large 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 lots 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 a system 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 dynamicity 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.

978-1-5090-4663-8/17/$31.00 ©2017 IEEE

II. PROPOSED ARCHITECTURE

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 U_i whose security clearance **(USC(U_i))** is less than the classification level of resource r_j **(RCL(r_j))**, that is **(USC(U_i)) < (RCL(r_j))**. Thus the developed weighted GAG with the following details:

- Let $SR= \{ sr_j \mid j=1 \ldots l \}$ be the set of all security rules.
- Let $IDSR: SR \rightarrow N: IDSR\ (sr_j)$ = the Importance Degree of security rule sr_j.
- 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 $W: E \rightarrow N: W\ (e_{ij})$ = IDSR (sr_i) be a function defines edges' weights in GAG. Where sr_i is the rule which the edge e_{ij} emerges from.
- Let $R= \{r_j|j=1 \ldots n\}$ be the set of grid resources.
- Let $SP: R \rightarrow N: SP\ (r_j)$ = weight of the shortest path to resource r_j.
- Let $RCL: R \rightarrow N: RCL\ (r_j) = SP\ (r_j)$ be a function maps each resource to its classification level.
- Let $Role=\{role_i|i=1 \ldots s\}$ bet the set of user's roles
- Let $IDR: Roles \rightarrow N: IDR\ (role_j)$ = the importance degree of $role_j$.
- Let $U = \{Ui \mid i= 1 \ldots m\}$ be the set of all users.
- Let $UR \subseteq U \times Roles$: be the set of relations of user to role assignements.
- Let $USR: U \rightarrow Roles : USR(u_i) = \{role_j \mid (u_i, role_j) \in UR\}$ be a function derived from UR mapping each user to a set of roles.
- Let $USC: U \rightarrow N: USC\ (U_i) = \sum_{roles\ \in\ USR(Ui)} IDR(role_j)$ be a function maps each user to his security clearance.

Let us consider now the following example: A grid environment has 5 resources R= {r_1, r_2, r_3, r_4, r_5}, and 3 security rules SR= {sr_1, sr_2, sr_3} with their importance degree (Illustrated in table 1)

- sr_1 requires the user to be from XYZ university
- sr_2 requires the user to have a Student role
- sr_3 requires the user to have a programmer role

The five resources have the following security policies:

- r_1, r_2 require sr_1.
- r_3 requires sr_1 and sr_2.
- r_4 requires sr_1 and sr_2 and sr_3.
- r_5 require sr_1 and sr_2 and sr_3.

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: RAP [11] (Request Analyzer and Processor), WGAG Search engine, WGAG database, XML parser [11].

TABLE 1. SECURITY RULES' IMPORTANCE DEGREE

Security rule	Importance degree
Sr1	4
Sr2	1
Sr3	9

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 r_i; the graph have to be parsed to calculate the shortest path from root to r_i, the obtained value is taken as classification level of r_i (illustrated in Fig.2). Following this it maintains the output decision graph in WGAG data base to be used by WGAG search engine.

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(U_i) 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(ri) ≤ SC(U_i)**. Finally, the group of 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 r_i, 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 $CL(U_{i)}$ then the RAP sends the request to the WGAG search engine, this last will parse the weighted grid authorization graph for the special resource r_i and gives accepted access as result if: **CL (ri) ≤ SC (U_i)** else denied, the answer is sent to RAP, then to PEP.

TABLE 2. RESULT OF XML PARSER

sr_i / r_i	Sr_1	Sr_2	Sr_3
r_1	1	0	0
r_2	1	0	0
r_3	1	1	0
r_4	1	1	1
r_5	1	1	1

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 $\mathbf{r_i}$ in the graph is a structure of 2 fields:
- Label
- **CL** an integer that represent the classification level of the resource.

Begin
<u>**Step 1:**</u>

- 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.

<u>**Step 2**</u>:

- Add each resource $\mathbf{r_i}$ whose correspondent row in N'ST has 0 cells, as a child resource to **N**.
- Sum the cells of each column of N'ST and refer it as count.
- Choose the security rule $\mathbf{sr_j}$ 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 $\mathbf{sr_j}$ (each row whose **jth** cell >**0**).
    - ➢ The second table **T2** contains the rows of resources which do not demand $\mathbf{sr_j}$ (each row whose **jth** cell=**0**).
- Add a left child node **LCN** to **N** with $\mathbf{sr_j}$ as a security rule (sr) and **T1** as the security table **ST**; the weight of sr_j in IDT as weight of **LCN**. Let the correspondent edge of **LCN** refers to $\mathbf{sr'_j}$ 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 **RCN**. Let the correspondent edge of **RCN** refers to **Null**.

<u>**Step**</u> 3: Repeat step 2 for each child until a child with empty security table is reached.
<u>**Step4**</u>: for each resource $\mathbf{r_i}$ in the graph a shortest path from the root node to this resource is calculated and added to **CL** of $\mathbf{r_i}$.
<u>**Step 5**</u>: Prune the graph at nodes labeled **ROOT**. Erase all interim security tables (**ST**) to free space.

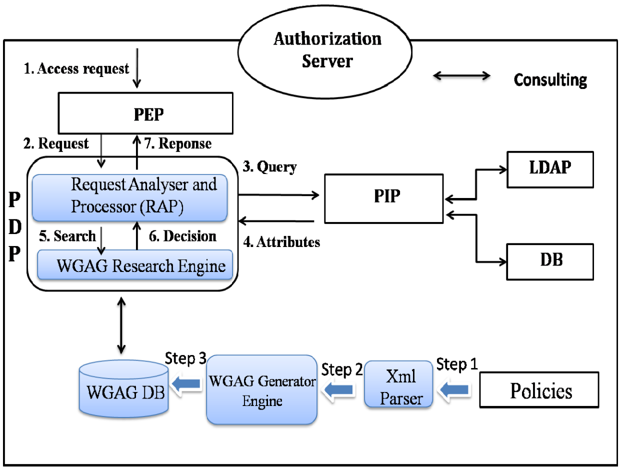

Fig. 1 Proposed architecture: an extended XACML model (Shaded components are our contribution)

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.

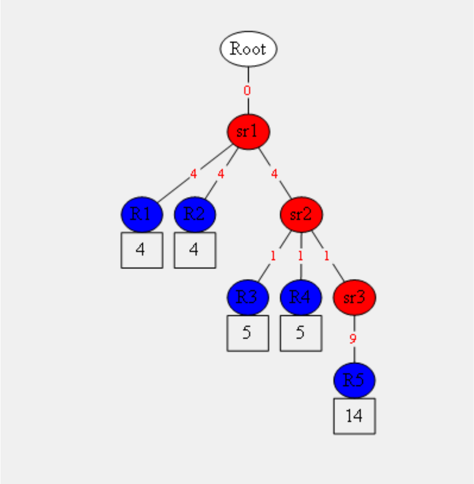

Fig. 2 weighted grid authorization graph generated by WGAG

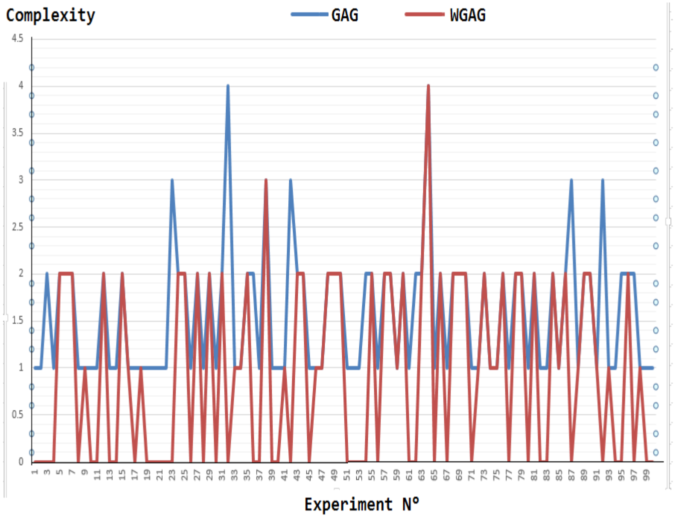

Fig. 3 Simulation result

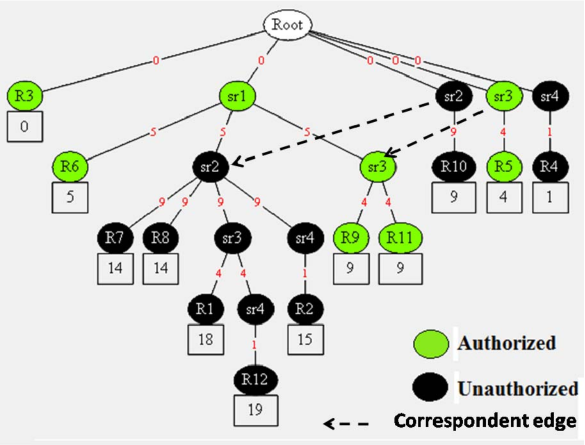

Fig. 4 : group of authorized resources for a user with sr1(importance degree=5) and sr3 (importance degree=4).

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

[1] I. Foster, What is the grid? A three point checklist, GRID Today (2002).

[2] A. Chakrabarti, A. Damodaran, S. Sengupta, Grid computing security: a taxonomy, IEEE Security & Privacy 6 (1) (2008) 44–51.

[3] Rashmi Bhatia; “Grid Computing and Security Issues”, International Journal of Scientific and Research Publications, Volume 3; ISSN 2250-3153 (2013).

[4] Husam Khider ; Taha Osman ; Nasser Sherkat “Attribute-Based Authorization for Grid Computing” International Conference on Intelligent Systems, Modelling and Simulation (2010).

[5] Kaustav Roy, Avijit Bhowmick “A Proposed Mechanism for Cross-Domain Authorization in Grid Computing Environment ”, International Journal of Emerging Technology and Advanced Engineering ISSN 2250-2459, Volume 2, Issue 4, (2012)

[6] J. Barkley, K. Beznosoz et J. Uppal. Supporting Relationships in Access Control Using Role Based Access Control. *Proceeding of the ACM workshop on RBAC*, Fairfax, Virginia, USA, 28-29 October (1999).

[7] E. Yuan, J. Tong, “Attributed Based Access Control (ABAC) for Web Service”, The 2005 IEEE International conference on web service (ICWS’05), **(**2005**)**.

[8] Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V1.1,‖ Organization for the Advancement of Structured Information Standards (OASIS), http://www.oasisopen. org/committees/tc_home.php?wg_abbrev=security, 2003.

[9] eXtensible Access Control Markup Language (XACML) Version 2.0,‖ Organization for the Advancement of Structured Information Standards (OASIS), http://docs.oasisopen. org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf, (2005).

[10] M. Kaiiali, R. Wankar, C.R. Rao, A. Agarwal, New efficient tree-building algorithms for creating HCM decision tree in a grid authorization system, in: The 2nd International Conference on Network Applications Protocols andServices, NETAPPS, Malaysia, 22–23 September 2010, pp. 1–6.(2010).

[11] Mustafa Kaiiali, Rajeev Wankara, C.R. Raoa, Arun Agarwal, Rajkumar Buyyab, Grid Authorization Graph, Future Generation Computer Systems 29 1909–1918 (2013)

[12] Mustafa Kaiiali, C. R. Rao, Rajeev Wankar, Arun Agarwal, Cross-Domain, Single Resource Authorization using HCM. International Technology Management Conference, Antalya, Turkey, (**2015**)