A Multi-Domain Role Activation Model

Vilmar Abreu[1], Altair O. Santin[1], Eduardo K. Viegas[1], Maicon Stihler[1,2]
[1]Graduate Program in Computer Science / Pontifical Catholic University of Parana
Curitiba, Parana, Brazil
[2]Federal Center for Technological Education, Leopoldina, Minas Gerais, Brazil
{vilmar.abreu, santin, eduardo.viegas, stihler}@ppgia.pucpr.br

***Abstract*— Organizations establish partnerships in order to achieve a strategic goal. In many cases, resources in a given organization are accessed from external domains, characterizing multi-domain operations. This paper presents an approach to perform role activation in multi-domain environments. The active roles are imported in other domains from a user's home domain. Thus, a Single Role Activation (SRA) is performed, similarly to Single Sign-On (SSO) authentication. The administrative autonomy to define each role permission is kept within each local domain. We evaluated the proposal by implementing a prototype to provide support for SRA, based on RESTful web services and standardized specifications such as XACML and OpenID Connect. The prototype evaluation measured response time for simultaneous access requests, with SRA showing better results when compared to traditional role activation. Furthermore, from a security perspective, the proposal is about 15 times faster than traditional approaches.**

Keywords— ***Multi-Domain Role Activation; RBAC; XACML; OpenID Connect.***

I. Introduction

Unauthorized access always represents a risk for applications [14]. An access control is a security mechanism to prevent unauthorized access to an application by using policies to regulate user interactions [17]. The Role-Based Access Control (RBAC) has several advantages over traditional access control models (e.g. discretionary and mandatory) [9, 12]. RBAC uses roles as a mechanism to mediate the association between a user and her permissions (granted rights). In Attribute-Based Access Control (ABAC), on the other hand, permissions are defined by attributes and there is no permission assignment for users or roles [8].

XACML is an XML-based language used to write access control policies. It defines how to perform requests and receive responses, and how to evaluate policies [13]. Although XACML has a profile for RBAC, it does not support the implementation of role activation nor the Separation of Duty (SoD) from the RBAC model.

RBAC traditionally operates in a single domain. Some authors [5, 10, 11, 18] presented proposals that deal with multi-domains. The main difficulty in multi-domain operations are role semantics (i.e., a role with the same name across different domains may have different sets of permissions on each domain). In some cases, resources and operations may not be compatible between different domains, therefore requiring a taxonomy or ontology to provide interoperability [11, 18]. Moreover, the user must activate the required role on each domain to operate on its local resources.

Web multi-domain security protocols, such as OpenID Connect (OIDC), provide Single Sign-On (SSO) without fine-grained access control [1]. OAuth [6] is an open source framework for access authorization in a web environment, it limits access to a protected path (URL) without using policies to control the operations on the resource (URL). Its purpose is to provide access authorization to a relying party's (RP) application, without requiring credential sharing [6]. On the other hand, OIDC is an Identity Management (IdM) system that allows the RP to authenticate the user identity [1].

Our proposal is to create a multi-domain access control that supports different role semantics, improved with a single role activation (SRA) approach. We found that the XACML architecture and the RBAC model, integrated with the OIDC, are able to provide support for SRA in a multi-domain environment. Therefore, a user will be able to access different domains (using SSO) and, by using SRA, she will not have to activate roles for each accessed domain.

The remainder of the paper is organized as follows. Section II discusses the related works. Section III describes the proposed model. Section IV shows experimental results. Finally, the conclusion is drawn in section V.

II. Related Works

Performing authorization in multi-domain environments is a widely-discussed challenge in the literature. The work of H. K. Lee [20] reviewed a series of works that address multi-domain authorization. The authors identified the following properties present in most studies: (i) *Autonomy*: all domains aim to maintain their autonomy to define the access rules to protected resources; (ii) *Privacy*: the private user information cannot be accessed by unauthorized users; (iii) *Decentralization*: a decentralized mechanism is needed to manage security policies; (iv) *Scalability*: the authorization system needs to be scalable and simple to deploy; (v) *Standardization*: the usage of well-known patterns provides compatibility and allows interoperability between various domains.

Several proposals tried to solve the multi-domain RBAC problem. Shafiq et al. [18] proposed an algorithm to create role mapping through match operations. Such operations perform an intersection among each role involved when merging a domain to yield the new role permissions set. This centralized merge operation demands some processing power, since it becomes necessary to map all roles among themselves. Furthermore, in the case of updates, in most cases, it is necessary to redo all intersections. Additionally, a shared resource area is necessary, so that resources are visible to all domains. Finally, the

permission semantic is domain specific, even when they have the same name. On Qi Li et al. [11] the concept of role virtualization on demand was proposed. The RBAC administrator defines the roles she wants to use for multi-domain operations purposes and creates links for those roles in a global domain. Such approach improves on the proposal of Shafiq [18] bringing the major advantage of choosing roles on demand, without the need of including all roles in the global domain. However, it still requires the centralization of role management, a shared resource area and the definition of role semantics.

Freudenthal et al. [5] adopted a credential repository called wallet, which stores authorization delegations using roles. The delegation is composed of three elements with the following format: "*[Subject -> Role] Domain*". Each domain has wallet synchronized with each of the other domains' wallets through a publish/subscribe service. When the delegation does not exist in the local wallet, a discovery service is used to locate the corresponding wallet (i.e., the one that holds the resource involved in the delegation). If the delegation is found, the entry is inserted in the local wallet for caching purposes. The search for delegation data can be exhaustive, requiring a scan through several wallets to retrieve all delegations needed to obtain an authorization.

Mouliswaran et al. [21] presented a model using Formal Concept Analysis (FCA) to represent the access permissions that a role has in different domains. The authors considered that there are global roles (cross-domain) that have local permissions on each domain. Those permissions are stored in a matrix, where global roles are organized as rows and permissions as columns. Their approach is not scalable because each operation on a particular resource is stored in a column, therefore, if a domain has several resources and operations, the matrix becomes large and sparse, thus its usage becomes unfeasible.

III. Proposal

This section describes the proposed access control model, followed by its specification. Afterwards, it shows the proposed mechanism to import the activated roles from a user's home domain and the method for policy writing, which uses local roles and imported role sessions.

A. Access Control Model

In order to enable multi-domain active role importation, it was necessary to create a model capable of supporting such a feature. Thus, support for SRA was built in a model integrating an Identity Management (IdM) system, RBAC controller, fine-grained access control and a policy language. The proposed model (Fig. 1) is based on the integration of:

- *Authentication Control*: responsible for user identification and authentication – when a user is successfully authenticated, the Authentication Control provides a short-term ticket, which proves the user's authentication to the Access Authorization Control. Besides, it is responsible for managing users in the proposal, acting as an Identity Provider (IdP) [2, 24].
- *Access Authorization Control*: responsible for issuing tokens that authorize the user to access protected services. Thus, it acts as an admission control to other Service Providers (SPs). The Access Authorization Control provides a different token for each service that the user access, each with a different access scope.
- *Role-Based Access Control*: Acting as a SP, it is responsible for role and session management, creating and assigning roles to users. It also provides the means to detect conflicts of interest between roles (used to implement SoD).
- *Attribute-based Access Control*: Acting as a SP, it is responsible for performing fine-grained access control, using roles as user attributes. It implements attribute evaluation and policy enforcement. Attribute-based Access Control and Role-based Access Control form up the core of the SRA implementation.

The user requests her authentication to the Application (Fig. 1). The Application (App) creates a session for the user and requests authentication to the Authentication Control, providing the user session (*event i*) as input. The user provides her credentials that, when valid, make the Authentication Control provide a ticket for the App (*event ii*). The App requests the token to the Access Authorization Control, providing the ticket (*event iii*). Afterwards, the Access Authorization Control provides an access token (*event iv*) that allows the user to access the Role-based Access Control (RBAC) and Attribute-based Access Control (ABAC). When holding the token, the App is able to access the RBAC to activate user roles (*events v and vi*) and to access the ABAC to request access to protected resources (*events vii and viii*).

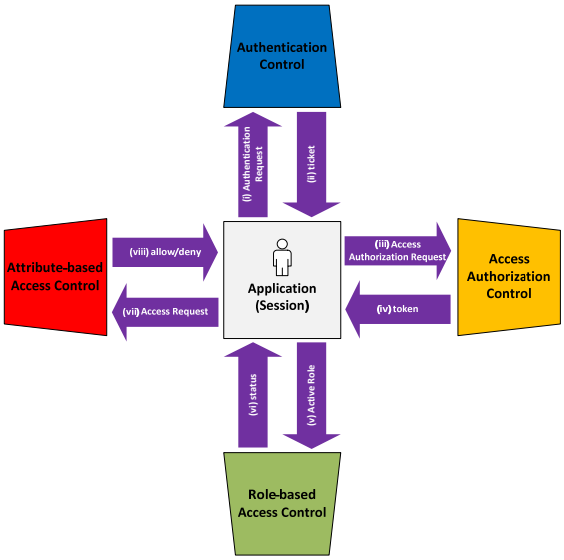

Fig. 1. Proposed Access Control Model

The access control model was conceived with each component working as a service, requiring an access token that is provided online by the Access Authorization Control. For each service, two different scopes are required: one scope allows only reading the resource, while the other allows also its update. This restriction allows access to the application only for authenticated users holding a valid access token. For example, to activate a role on RBAC, the user must have an access token within the *"rbac_home_full"* scope. However, to retrieve her active roles, a user must have a token within the *"rbac_home_read"* or *"rbac_home_full"* scope.

B. Access Control Specification

The specification of the attribute-based access control using multi-domain roles aims to use well-known standards to enable the proposed model. The OpenID Connect (OIDC), which is based in the OAuth specification, was used to specify the Authentication Control. The identity token is used to store the user information and her home domain. For the Role-based Access Control, the specifications defined in the standard of NIST [8] was used. Finally, for the Attribute-based Access Control the XACML was used.

The integration of these components can be seen in Fig. 2. The App requests the user authentication in the OIDC providing her credentials (*event i*). The OIDC validates the user credentials and returns a *nonce* to App (*event ii*). The *nonce* is used by the App to get the access token (*accessToken*) and the identity token (*idToken*), represented in *events iii* and *iv*, respectively. The App requests the roles associated to the user to RBAC (*event v*) and informs the access token. The RBAC performs an online validation of the access token, returning the roles available for the user to choose which role she wants to activate (*event vi*). The role activation is performed by sending the access token and the chosen role to activate to the RBAC (*event vii*).

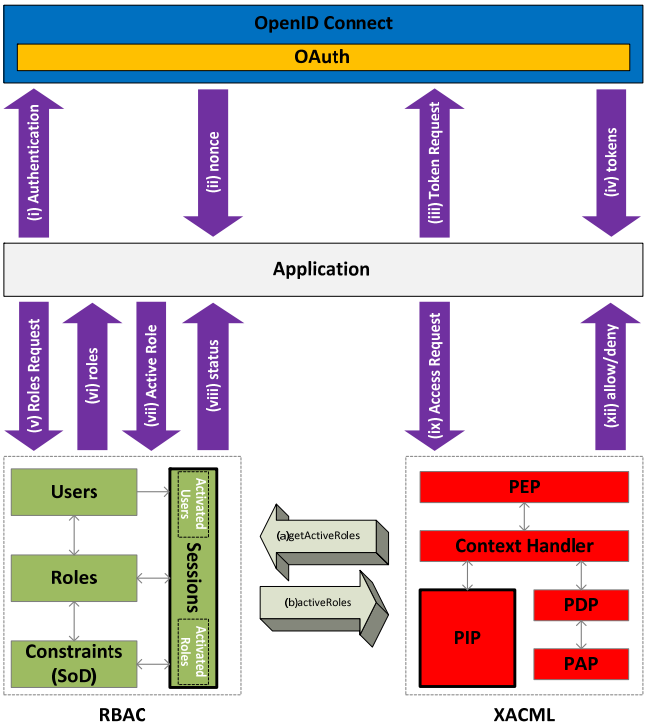

Fig. 2. Proposed model overview

The App request access to the XACML service, in behalf of the user, informing the action that she wants to perform over a given resource (*target*) along with the access token (*event ix*). This request reaches the PEP, which forwards it to the Context Handler (CH). The CH builds an XACML request to send to the PDP, which will evaluate the policy and make a decision (allow or deny). Before building the request, the CH asks the PIP for the active roles of the user. The active roles are retrieved from the RBAC service by providing the access token (*event a*). After receiving the active roles for the user (*event b*), the CH forwards the attributes (including the active roles) for the PDP, that in turn evaluates if a user is allowed to perform the desired operation on a resource. The CH receives the PDP decision and forwards it to the PEP, which allows or denies the user access to the requested resource (*event x*).

C. Single Role Activation (SRA) Mechanism

The proposal aims at maintaining the autonomy of a local domain administrator for defining role permissions, and at the same time supporting cross-domain operations without requiring role activation on each visited domain. Thus, a mechanism is proposed to support importing role sessions from a user's home domain to other domains. The SRA means that a user does not need to activate roles in each local domain where there are roles enabled to used. Different role semantics are handled by the local domain administrator, who defines the role permissions for each user visiting it. This requires some trust relationship between the involved domains. We consider that this trust relationship can be motivated by the same reasons that make all involved domains share the same OIDC. Therefore, we assume the trust relationship as being external to the system and it will not be addressed in this paper.

Fig. 3 shows the SRA overview. As SRA is SSO-based (*event i*), the user must authenticate herself on OIDC and successfully obtain tokens. In order to access an application transparently, in a visited domain through the SRA, the user must have active roles in her own home domain (*Domain A*).

On the user's behalf, the home domain App requests access to XACML (*event ii*), informing the action to be performed on a given resource with the access token. The access token scope contains the permission for the user to access a visiting domain in *read* mode (*Domain B*). The PEP receives the request and forwards it to the CH. Thus, the CH requests the PIP to get the active role's session from the user's home domain. For this purpose, the PIP employs the user's access token to request the OIDC about her identity token (*idToken*). The *idToken* has a claim (attribute) identifying the home domain for the user. Finally, the PIP requests the user active roles to RBAC home domain (*event iii*).

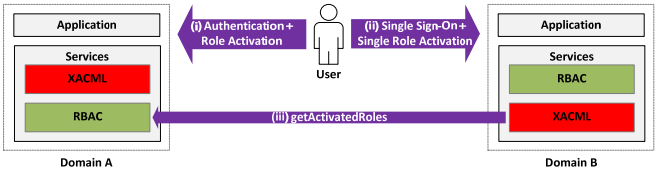

Fig. 3. Multi-domain SRA overview.

It must be said that the user's roles session, which is active in the user's home domain, is imported in the visiting domain through SRA, but RBAC can detect permission conflicts (Dynamic SoD - DSoD) with active roles. Therefore, when the user's home domain role activation is possible (i.e., there is no DSoD), the PIP returns to the CH all the user's active roles, which may be both local or active by SRA. Then, the CH sends the attributes to the PDP to be used in the policy evaluation.

This mechanism is flexible, given it keeps compatibility with the RBAC specification, by assigning permissions to roles that refer to roles from the user's home domain, which are imported through SRA mechanism. In practice, it is expected that a user, which uses SRA, will visit other domains without

activating roles in such domains, just like what occurs with SSO for authentication.

D. Policies with active roles and SRA

In order to implement SRA, the PIP must be capable of searching for a user's active roles in the RBAC service on the user's home domain. However, it was necessary to develop a method for writing policies supporting SRA without changing the XACML architecture or its data flow. First, a new attribute type was proposed to support RBAC. The administrator must use the attribute *"rbac_active_role"* to refer active roles when writing policies. When the PIP receives the request for user active roles, it searches in the local domain RBAC service.

Then, to take advantage of SRA, the local system administrator (visited domain for a user) must write policies referencing roles from the user's home domain, inserting the domain prefix followed by the role name. In such a case, the attribute *"rbac_sra_role"* must be used to write the XACML policy. When a policy that uses a SRA is added, the related roles references are stored inside the local RBAC session. Thus, it is possible to use this role reference for DSoD. Once the administrator adds a policy, it is asked if she wants to create a DSoD rule using the related role policy.

IV. Experimental Results and Discussion

This section presents the prototype that implements the proposal and the performed evaluation tests.

A. Implementation

The App was implemented in Java (www.docs.oracle.com/javase/8) using the Vaadin framework (www.vaadin.com), due to its rich user experience. The OIDC server was implemented using the Nimbus [23]. Nimbus API (*Application Programming Interface*) follows the OpenID Connect and OAuth 2.0 specification. The XACML evaluation engine was implemented using the Java-based WSO2 Balana (www.github.com/wso2/balana), an API based on the well-known sun-xacml (www.sunxacml.sourceforge.net), which supports the XACML version 3.0.

Two RESTful (www.tools.ietf.org/html/rfc6690) web services were implemented using the JAX-RS API [22]. The first web service implements the following RBAC features: role management, users and roles assignment, active users, active roles, and separation of duty. The second web service implements the PEP features, honoring the PDP access decisions defined in WSO2 Balana. All available functions in both services requires an access token issued by the OAuth 2.0. Besides the need of a valid access token, the services check if the token access scope is compatible with the requested function. All communication happens using HTTPS and self-signed certificates, generated by Java *keytool* for dealing with certificates.

In order to interact with architectural entities in different domains, the PIP was extended to search for the user's active roles in RBAC service. By default, the WSO2 Balana performs caching of attribute values provided by the PIP to optimize the process. However, as the active roles of users are volatile, it was necessary to clean the PIP cache every time a user request to activate/deactivate a role. Thus, the PIP gets and provides only the roles that are currently active.

B. Performance Evaluation

A controlled environment was developed to prevent possible interferences in the time measurements. A total of four hosts were connected in a Gigabit local network, all with the same configuration: core i7 processor and 8 GB of RAM. Each host executed the Ubuntu Linux x64 operating system v. 14.10 with Java 1.7 and tomcat 7 to run the web services.

The architecture's components were developed as web services. In the test, two hosts ran the RBAC and XACML services, representing two different domains. A third host ran the OIDC server while the fourth host was a test application automating the entire process (i.e., from user authentication to the resource access request). The role activation process mimic the user's choice (always using the first role available), given that it is not possible to predict the user behavior.

The test application implemented the scenario of Fig. 2 (composed by 11 facilities), grouped into four groups (*i*) *OpenID Group* comprises the session management, local authentication, home user authentication and logout processes; (ii) *OAuth Group* covers the local and home user token retrieving processes; (iii) *RBAC Group* includes the local domain role retrieving and activation, and the home user role retrieving and deactivation processes, and (iv) *XACML Group* comprises the step of requesting access to a protected resource.

The tests aimed at evaluating the access control performance in the home domain *versus* the visited domain while using the SRA. For this purpose, a number of several requests and a number of users were evaluated. Initially, 10 roles were registered on each RBAC domain. To test SRA performance, it was created 10 policies linked to the local RBAC domain and 10 policies linked to the home user RBAC domain. On the OIDC, it was created 1,000 users; each user was linked to a role in each domain, randomly chosen. The first test was deployed using 10 clients performing from 1 to 100 simultaneous requests. Each iteration randomly chooses 10 clients. Two scenarios were tested: on the first scenario, the clients make each request on a single domain; on the second scenario, the requests are made on a multi-domain with SRA support. All requests are equally evaluated, given the caches for PDP/PIP were disabled.

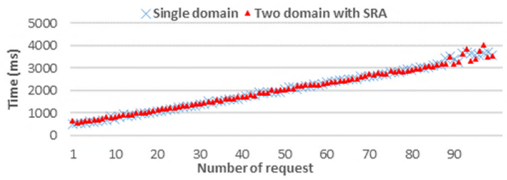

Fig. 4. Effect of increasing simultaneous requests on domains.

The Fig. 4 shows the time spent for accessing a single and multi-domain considering the number of simultaneous requests. One can be noticed that an equivalent performance can be observed for local and home RBAC user access.

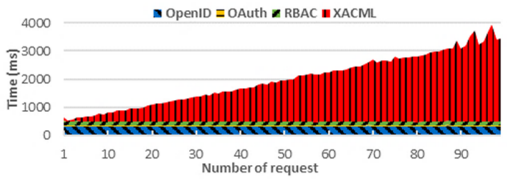

Fig. 5. Effects of increasing simultaneous requests in the proposal entities.

The Fig. 5 illustrates a test scenario to evaluate the proposal's entities in a multi-domain environment with support of SRA. We observed that by increasing the number of simultaneous requests only the XACML is affected, while for other entities the performance remains almost unchanged.

For a second test, the number of users is varied from 1 to 100, each user performs 10 access requests. The users are randomly chosen for each test iteration. It is possible to note a small performance impact while requesting user from home RBAC domain (Fig. 6). The increase on the number of users directly affects the architecture's entities (Fig. 7).

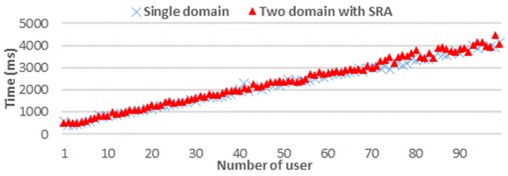

Fig. 6. Effects of increasing users requests on the domains.

Both tests significantly affect the XACML response time. This behavior was expected in the first test, due to the growth in the number of access requests. However, in the second test, it occurs because the application performs 10 access requests by user. Thus, increasing the number of users increases the number of access requests. Note that a user is identified by its access token in the services (RBAC and XACML). At each iteration test, the user creates a session and, at the end of the iteration, a session logout is performed. Thus, every iteration uses a different access token.

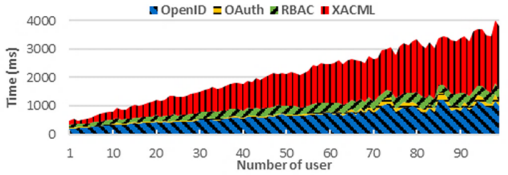

Fig. 7. Effect of increasing simultaneous users on the proposal entities.

C. Security Analysis

This section performs a security analysis of the proposed model. Figure 8 illustrates two scenarios which highlights the benefits of usage the Access Authorization Control. Scenario A does not use the access token. Thus, it is necessary to go through all XML rules to conclude that a user does not have access to a specific resource. This scenario shows an unauthorized user requesting access to a protected resource *(event i)*. The PEP intercepts the request and forwards it to the Context Handler, requesting the user attributes to PIP *(event ii e iii)*. The PIP requests the active roles from RBAC *(event iv)*. RBAC, in its turn, does not return any role because the user is not authorized (*event v)*. Next, PIP informs the Context Handler that the user has no active role *(event vi)*. The Context Handler creates a XACML request to PDP *(event vii)*. The PDP concludes that the user is not authorized and replies to the Context Handler *(event viii),* which informs the PEP *(event ix)*. The PEP denies access to the protected resource *(event x)*.

In scenario B, the unauthorized user requests access to a protected resource *(event i)*. The PEP intercepts the request and forwards to OAuth, which tries to validate the access token *(event ii)*. As it is not authorized, OAuth rejects the access token and informs the PEP *(event iii)*. The PEP denies access to the protected resource *(event iv)*. Scenarios A and B were evaluated by using the same hosts as the section IV.B. The tests aimed to measure the execution response time for each scenario. The result of the evaluation showed that the scenario B, which uses OAuth, was about 15 times faster than the scenario A. This advantage is because the number of operations is reduced in scenario B, given that a user without necessary authorizations are denied before evolve in the evaluation process. In a possible Denial of Service (DoS) attack, Scenario A, which does not use OAuth, the resources would be exhausted faster, while Scenario B would not be affected by the attack as Scenario A.

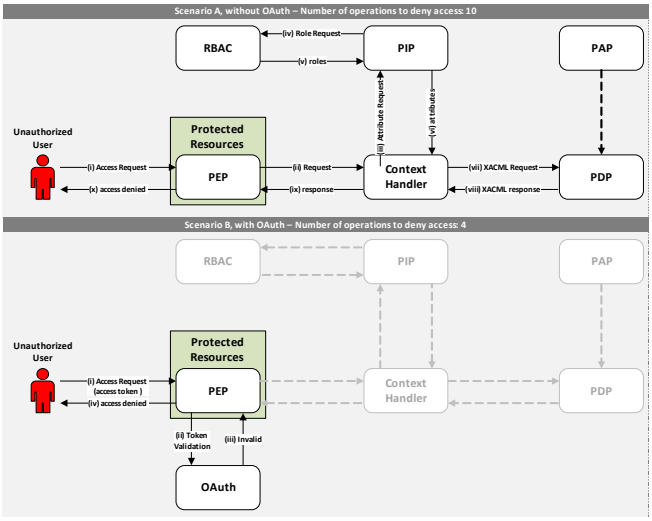

Fig. 8. Effect of increasing simultaneous users on the proposal entities.

D. Compliance Analysis

The XACML defines three top-level elements: Rule, Policy and PolicySet. In a simplified way, the Rule is expressed as a predicate (using role) that is individually evaluated. The Policy element contains one or more Rules. Finally, the Policy Set element contains one or more Policies. The Policy and PolicySet elements adopt a combination algorithm that is used when there are conflicting rules. For example, a policy contains two rules, if evaluation of one rule returns true and the other false, PDP must use a combination algorithm to decide the evaluation's outcome. The most well-known combination algorithms are: *Deny-overrides, Permit-overrides* and *First-applicable* [13].

The RBAC standard [3] defines an administrative specification that guide RBAC implementation. Among these, the specification proposes a Permission Assignment (PA) function that associates the roles with all their permissions. Our proposal considers that the roles are user attributes. In such way, the permissions assigned to a role are defined in the policies stored in the PAP repository. Thus, the PA function is performed every time the PDP evaluates an access request.

It was necessary to adopt a strategy for writing the XACML policies using the user's active roles to prevent a possible inconsistency in the PA function. For example, the Engineer role must be allowed to read a project and write a technical report. If these rights were written in different policies, it is possible that role Engineer has only one of the permissions,

causing a role semantic inconsistency. The strategy adopted to solve this problem associates all permissions of the role (PA) in a single Policy. Thus, when a role is active, the user is assigned to all the permissions of an active role or none in the case of SoD. Such issue is not addressed neither by related works [4, 7] nor on the XACML profile.

V. Conclusion

An attribute-based and multi-domain role activation model was presented, which considers the different semantics of a role and allows the single role activation (SRA). We use OIDC to provide Single Sign-On among domains and OAuth acting as admission control for the services, because each service has different context. The RBAC controls the roles session, users and separation of duty. The XACML performs fine-grained access control based on attributes, from roles that are in fact user attributes.

The proposal keeps the autonomy of each local domain administrator, allowing the administrator to set permissions for each role (local or imported role – from a RBAC session). Thus, the local domain administrator can set XACML policies referencing also roles from a RBAC user-home domain. When a user visits a domain and requests access to a protected resource, the active role's session on her home domain is considered. If the permissions of an active roles in the home domain do not conflict with local roles, the RBAC imports the role session into the visited domain. As a result, a user transparently accesses other domains, without activating role in the visited domains, as happens with SSO for authentication.

The prototype showed the model's feasibility. The tests related to multi-domains with SRA showed that the access control has a similar performance to a single domain, with slight increase in the response times. Considering the qualitative advantages that the SRA provides to the user and the reasonable performance, we consider that the proposal shown is feasible.

Acknowledgements

This work was partially sponsored by the Brazilian National Council for Scientific and Technological Development (CNPq), grants 307346/2015-3 and 404963/2013-7. Vilmar Abreu Junior wishes to thanks to CNPq for scholarship granting, process 381612/2014-7.

References

[1] Y. C. Y. Cao e L. Y. L. Yang, "A survey of Identity Management technology," Proc. of IEEE Int. Conf. Inf. Theory Inf. Secur., p. 287–293, 2010.

[2] M. Hansen, P. Berlich, J. Camenisch, S. Clauβ, A. Pfitzmann, M. Waidner, "Privacy-enhancing identity management," Information Security Technical Report, vol. 9, no. 1, p. 35-44, 2004.

[3] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli. "Proposed NIST standard for role-based access control.," ACM Transactions on Information and System Security vol. 4, no. 3, p. 224-274, 2001.

[4] R. Ferrini e E. Bertino, "Supporting RBAC with XACML+OWL," Proc. of ACM symposium on Access control models and technologies, p. 145–154, 2009.

[5] E. Freudenthal, T. Pesin, L. Port, E. Keenan, e V. Karamcheti, "dRBAC: Distributed Role-based Access Control for Dynamic Coalition Environments," Proc. of Int. Conf. on Distributed Computing Systems, p. 411–420, 2002.

[6] D. Hardt, "The OAuth 2.0 Authorization Framework," in Internet Engineering Task Force (IETF), p. 1–76, 2012.

[7] N. Helil e K. Rahman, "RBAC Constraints Specification and Enforcement in Extended XACML," Proc. of Int. Conf. on Multimedia Information Networking and Security, MINES, p. 546–550, 2010.

[8] V. Hu, D. Ferraiolo, e R. Kuhn, "Guide to Attribute Based Access Control (ABAC) Definition and Considerations," NIST Special Publication, p. 1-162, 2014.

[9] J. B. D. Joshi, R. Bhatti, E. Bertino, e A. Ghafoor, "Access-Control Language for Multidomain Environments," IEEE Internet Computing, vol. 8, no 6, p. 40–50, 2004.

[10] H. K. Lee e H. Luedemann, "Lightweight Decentralized Authorization Model for Inter-Domain Collaborations," Proc. of ACM workshop on Secure web services, p. 83–89, 2007.

[11] Q. Li, X. Zhangt, S. Qing, e M. Xut, "Supporting Ad-hoc Collaboration with Group-based RBAC Model," Proc. of Int. Conf. on Collaborative Computing, p. 1–8, 2006.

[12] S. Osborn, R. Sandhu, e Q. Munawer, "Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies," ACM Trans. Inf. Syst. Secur., vol. 3, no 2, p. 85–106, 2000.

[13] B. Parducci e H. Lockhart, "eXtensible Access Control Markup Language (XACML) Version 3.0," OASIS Standard, p. 1–154, 2013.

[14] R. Power, "Tangled Web: Tales of Digital Crime from the Shadows of Cyberspace," Macmillan Press, p. 1-8, 2000.

[15] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, e C. Mortimore, "OpenID Connect Core 1.0," 2014. Available at: http://openid.net/specs/openid-connect-core-1_0.html. [Accessed: March 2017].

[16] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, e C. E. Youman, "Role-Based Access Control Models," IEEE Comput., vol. 29, no 2, p. 38–47, 1995.

[17] R. S. Sandhu e P. Samarati, "Access Control: Principles and Practice", IEEE Communications Magazine, p. Samarati, "Access Control: Principles and Practice," IEEE Communications Magazine, p. 1–21, 1994.

[18] B. Shafiq, J. B. D. Joshi, E. Bertino, e A. Ghafoor, "Secure Interoperation in a Multidomain Environment Employing RBAC Policies," IEEE Transactions on Knowledge and Data Engineering, vol. 17, no 11, p. 1557–1577, 2005.

[19] R. Sinnema e E. Wilde, "eXtensible Access Control Markup Language (XACML) XML Media Type," Internet Engineering Task Force (IETF), p. 1–8, 2013.

[20] H. K. Lee, "Unraveling decentralized authorization for multi-domain collaborations," Proc. of Int. Conf. Collab. Comput. Networking, Appl. Work, p. 33–40, 2007.

[21] S. C. Mouliswaran and C. A. Kumar, "Inter-domain Role Based Access Control using Ontology," Proc. of International Conference on Advances in Computing, Communications and Informatics, p. 2027–2032, 2015.

[22] Ribeiro, R. C. , Santin, A. O., Abreu, V., Marynowski, J. E., Viegas, E. K., "Providing Security and Privacy in Smart House Through Mobile Cloud Computing," Proc. of IEEE Latin-American Conf. on Communications, p. 1-6, 2016.

[23] A. Witkovski, A. Santin, V. Abreu, and J. Marynowski, "An IdM and Key-based Authentication Method for providing Single Sign-On in IoT," Proc. of IEEE GLOBECOM, p. 1–6, 2015.

[24] M. Stihler, A. Santin, A. Marcon, J. Fraga, "Integral Federated Identity Management for Cloud Computing," Proc. of Int. Conf. on New Technologies, Mobility and Security (NTMS), p. 1-5, 2012.