Policy Control Management for Web Services

Arlindo L. Marcon Jr.[1], Altair O. Santin[1], Luiz A. de Paula Lima Jr.[1], Rafael R. Obelheiro[2], Maicon Stihler[1]
[1]Pontifical Catholic University of Paraná / Graduate Program in Computer Science
[2]State University of Santa Catarina / Department of Computer Science
(almjr, santin, laplima, stihler)@ppgia.pucpr.br; rro@joinville.udesc.br

***Abstract* — The decentralization of corporate policy administration aiming to maintain the unified management of user permissions is a hard task. The heterogeneity and complexity of corporate environments burdens the security administrator with writing equally complex policies. This paper proposes an architecture based on Web Services, policy provisioning, and authorization certificates, to build up a loosely coupled unified administrative control for corporate environments. A certificate-based permission management scheme is used to derive new policies in the local domains of each branch. These new policies will update the corporate repository which, in turn, will configure the corresponding policies in the local domains of each branch. The Web Services technology provides the underlying protocols for the development of a prototype which shows the feasibility of our proposal.**

***Index Terms* — Policy Management, Web Services Security, Authorization Certificates.**

I. Introduction

THE Service Oriented Architecture (SOA) is a model that has shown fast acceptance in corporation environments, since enterprises envisage the integration of their computer systems and providing services through the Internet.

Centralized corporation policy management facilitates the imposition of rules in distributed environments, but in some cases it is unfeasible for corporate administrators to define specific rules for a large number of local resources in remote branches. Normally, the strategy adopted is to delegate the management of branches' resources to local administrators, but how to ensure that corporate policies have been applied correctly?

Cooperation amongst corporation branches in a project, for instance, requires access to resources located at different company sites. In such a case, project members need to work temporarily at other company sites. This requires an access control model that supports flexible policy configuration.

Access control policies based on provisioning provide flexibility in policy configuration, reducing the tight coupling among branches and a main site. However, a synchronization mechanism is required to maintain the branches' policy repositories consistent.

One approach to achieve a loosely coupled architecture is to decentralize security administration, but this strategy requires increased message exchanging among the entities that enforce the security controls in corporation environment. The main disadvantage of this approach is the higher network overhead.

One way to facilitate policy management in distributed and heterogeneous systems is the adoption of SOA standards, which contribute to loose coupling among the entities of the architecture. One of the most well-known and adopted technologies to implement SOA nowadays is Web Services (WS). In spite of offering interface independence, such implementations of SOA employ security mechanisms that impose a tight dependence among their policy control entities.

WS standardize the information exchanges among their entities in order to cope with eventual corporation platform heterogeneity. However, WS inherits policy control mechanisms from traditional security architectures. Thus, relationships among distributed entities of the policy control architectures depend on trusted third parties – normally, domain authentication services that are not easily scalable.

Our proposal is twofold. First, provisioning security policies lowers the coupling among policy-related entities in the architecture, reducing network overhead and making it easier to cope with component and communication failures. At the same time, asynchronous policy updates are still possible, thus retaining the main advantage of traditional policy control architectures. Second, our scheme allows branch security guardians to derive policies for local resources in accordance to the corporation policy scheme, using certificate-based authorization. An important benefit is that these derived policies can be automatically and securely incorporated to corporate policy repositories, thus alleviating the need for manual intervention in policy management.

The remaining of this paper is organized as follows: Section II reviews Web Services and some security specifications related to our proposal. Section III describes the main policy control architectures. Section IV presents the proposal. Section V presents an example scenario. Section VI shows the prototype and its evaluation. Section VII discusses related work. Finally, Section VIII draws some conclusions.

II. Web Services

The main goal of the Service Oriented Architecture (SOA) is to provide a model by which services that carry out a given task can be made available by a standard, loosely coupled, and interoperable way, in order to supply a demand [1].

Web Services technology tries to implement SOA in the best possible way [2]. A service can be implemented in any specific programming language. The service interface description can be published in a UDDI directory (Universal Description Discovery and Integration) to allow access for others [3]. The interfaces description must be written in WSDL (Web Services Description Language) [4]. In Web Services, the entities interact using SOAP messages (Simple Object Access Protocol) [5], normally, on top of HTTP (Hypertext Transport Protocol) with data serialization based on XML (Extensible Markup Language) [6].

Web Services adopt a set of specifications to provide end-to-end security and platform-neutral message exchanges.

978-1-4244-3487-9/09/$25.00 ©

The specifications more closely related to our proposal are discussed below.

WS-Security adds extensions to SOAP messages providing support for signature and encryption (i.e., XML Signature/Encryption) and for the representation of security credentials (e.g., SAML assertions) [7]. The Security Token Service (STS) assures the validity of a security credential or promotes the translation between different credentials.

WS-Trust extends WS-Security through a request-reply message model to transport credentials in a secure way [8]. Credentials are obtained from an STS, and are used to achieve intra- and inter-domains trust relationships.

The XML Key Management Service (XKMS) is a resource that aids clients to store and retrieve cryptographic keys [9]. XKMS hides the particular details in the management of each key infrastructure (e.g., X.509) by adopting a neutral scheme to operate with key storage and retrieval.

The Service Provisioning Markup Language (SPML) defines standard operations for the configuration of distributed objects [10].

The Security Assertion Markup Language (SAML) can be used for information transportation in a standard format that other Web Services may trust, based in pre-established relationships [11]. SAML defines three types of expressions (e.g., authentication, attributes, and authorization) which may be created by a trusted third party (e.g., STS) and inserted in an assertion that will be associated to a client.

The eXtensible Access Control Markup Language (XACML) defines a XML-based language for expressing access control policies and a server-based architecture for their evaluation [12]. XACML uses the Policy Enforcement Point (PEP) and the Policy Decision Point (PDP) entities, defining a context for their intercommunication.

III. Policy Control Architectures

In this section a server-based and a certificate-based policy control architecture are shown.

A. Server-Based Policy Control Architectures

Server-based policy control architectures are based on two main entities: a reference monitor and a guardian. A Policy Decision Point (PDP) acts as a reference monitor, being responsible for deciding to either allow or deny accesses. A Policy Enforcement Point (PEP) acts as a guardian, honoring the PDP decision by either releasing or blocking accesses to a resource. In such architecture, a Context Handler intermediates the intercommunication between PDP and PEP, in order to maintain the messages they exchange in a standard format. The Policy Administration Point (PAP) stores the policies for the PDP evaluation.

Depending on how the aforementioned entities interact, two distinct control models, namely outsourcing and provisioning, may be employed [13].

Outsourcing Model

In the outsourcing model (pull operation mode), every request to access a resource sent by a client (event 1, Fig. 1) is intercepted by the PEP and then forwarded to the PDP (event 2). After that, the PDP makes a decision (event 5) based on authorization policies, retrieved from its Policy Administration Point (PAP, event 3), on the identity of the client, and on the requested resource. Next, the decision is sent back to the PEP (event 6) for it either to allow (event 6.2) or to block (event 6.1) the requested access.

In this model, the PEP has always to query the PDP in order to have the authorization policy evaluated. This approach is applied in the traditional access control architecture usually adopted by Web Services.

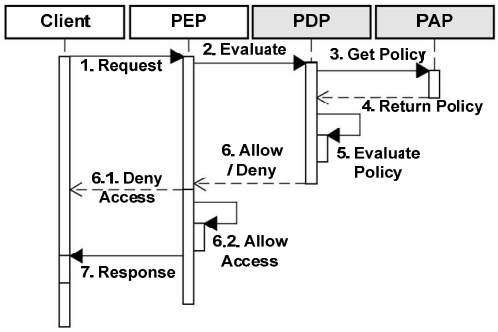

Fig. 1. Outsourcing Model

Provisioning Model

In the provisioning model (push operation mode), at initialization time, the PEP sends a message to the PDP (event 1, Fig. 2) to inform its features in order to get the corresponding policies. All retrieved policies (event 2) are stored in a Local Policy Administration Point (LPAP, event 4); local means in PEP domain. Therefore, the authorization policy evaluation can be done locally by a Local PDP (LPDP, event 11). According to such evaluation, the access request (event 7) is either allowed (event 12.2) or denied (event 12.1).

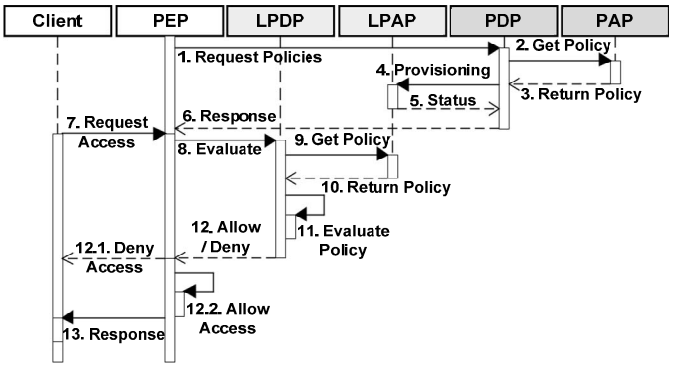

Fig. 2. Provisioning Model

In this model, there is no need for continuous message exchange between PEP and PDP. However, asynchronous interactions between PEP and PDP can still occur at any time. Such interactions can be caused by policy insertion, some update at the PDP policy repository (PAP) or by the arrival of policy evaluation requests at the LPDP whose corresponding policies are not present in LPAP.

Synchronizing policies

The provisioning model requires policy repository

synchronization to maintain the LPAP updated with respect to PAP, i.e., when an update is done in PAP it needs to be replicated to LPAP. According to RFC 3084, COPS-PR (*Common Open Policy Service protocol for support of Policy Provisioning*) is used as protocol to the perform policies synchronization in provisioning approach. Thus, COPS-PR establishes a TCP connection and uses keep-alive messages to monitor the availability of the intercommunication channel, allowing LPAP [14] to be updated when necessary. COPS-PR applies *Basic Encoding Rules* (BER) to encode the message being transported; BER also is applied to save the policies in the *Policy Information Base.*

B. Certificate-Based Policy Control Architecture

The Simple Distributed Security Infrastructure (SDSI) [15] associated to Simple Public Key Infrastructure (SPKI) [16] provide a flexible authorization scheme to build secure and scalable distributed systems.

In SPKI/SDSI principals are public keys, and it is based on authorization rather than authentication. A principal could be a client (permissions grantee) or a server (permissions grantor). Permissions are granted by delegation from one principal to another, forming a chain of authorization certificates – the subject of one certificate becomes the issuer of the next certificate in a chain (Fig. 3). The digital signature on certificates guarantees the authenticity of delegations.

SPKI/SDSI is serverless, meaning the certificates are stored by the client (*subject* of the certificate – Fig. 3) and they are issued by server (*issuer* of the certificate – Fig. 3). The permissions are coded in the *tag* field of certificate. As an example, in Fig. 3 – cert (1), the *Public-Key-A* is granting to *Public-Key-B* the permissions (*update* and *execute*) in the path *www.branch.com/researcher*. The field *propagate* means the permissions coded in the certificate may be granted by the subject to another principal. The validity (i.e.: *not-before* and *not-after*) and digital signature fields of the certificate are not shown in Fig. 3.

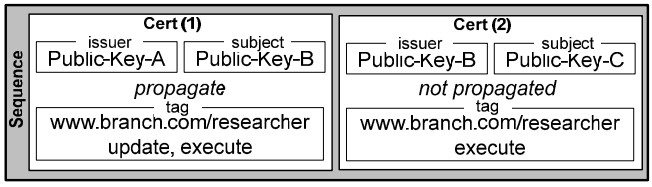

Fig. 3. Authorization certificates chain (sequence)

A certificate chain can be reduced to a single certificate that summarizes the corresponding granted permissions. To do so, the sequence of issuer and subject of each certificate in the chain is checked and an intersection among all certificates is done in the tag and validity date fields. Also, the *propagate* field must be present in all certificates, otherwise the *not propagated* certificates finishes the sequence. Therefore, the sequence of certificates in Fig. 3 can be reduced to *Public-Key-A* granting the *execute* permission to *Public-Key-C* on the path *www.branch.com/researcher*.

In order to facilitate the handling of clients' identification, SPKI/SDSI employ name certificates, which associate a local name with a principal's public key. Local names apply only to the namespace of each principal, and can be used in lieu of a public key.

IV. PROPOSED ARCHITECTURE

The proposed corporation architecture is composed by one corporate and several branch domains. The corporate domain comprises the administrative part of the corporation and the branch domains are composed by resource providers and clients.

The main goal of this proposal is to apply web services to provide the decentralization of corporation policy administration while maintaining its unified corporate policy management. Decentralization is achieved by provisioning of policies and the unified management is carrying out through SPKI/SDSI authorization certificates. The policies in the corporate domain repository (PAP) are used to provision the policies in the branches' repositories (LPAP). The policies in PAP can be stored by the corporate domain administrator or generated by a STS from a sequence of SPKI/SDSI certificates sent to PEP in a branch domain.

PAP maintains all the policies for branch resources, while LPAP (Local PAP) stores a local copy of the policies only applied to resources that a given PEP specifically controls. Administrative updates (events up_1 and up_2, Fig. 4) only are possible in the PAP repository. LPAP can receive updates only from PAP.

In the provisioning of branch policies, according to the SPML specification, the PDP represents the PSP (Provisioning Service Provider), whereas the PAP repository that stores all the corporation's policies represents the PSOs (Provisioning Service Objects). The PEP denotes the RA (Requesting Authority) and the LPAP represents the PST (Provisioning Service Target) that stores the provisioned policies.

In other words, the PEP requests the policies to PDP (event req_{pp}, Fig. 4), which in turn invokes SPML to retrieve policies from PAP and store them in LPAP (event *pp*). The provisioning of all branch policies is executed only at the PEP bootstrap, after that only updates are sent to LPAP, trigged by an alert message addressed to PEP. Based on the alert message the PEP requests the updates to PDP and the process follows as explained. If, for some reason, the LPAP cannot be updated, a notification of update pendency is generated by PSP to the corporate administrator (event *np*, Fig. 4).

SPML employs XML, which is standardized and an expressive language. Also the XML encoding, used to transport data across the network, is better than Basic Encoding Rules [17].

The proposed architecture can evaluate policies using the outsourcing (based on PDP, events *av* and ev_2, Fig. 4) or provisioning (based on LPDP, event ev_1) approaches. The PEP enforces the authorization decision (event *ac*) independently of whether the decision was done either locally (by LPDP) or remotely (through the PDP). In fact, the outsourcing approach is supported only for compatibility purposes. In other words, if the PEP works with policy provisioning, the proposed architecture can configure its LPAP with the adequate

policies. Otherwise, PEP can request to corporate PDP the evaluation of a policy. In fact, it can be assumed that only small branches with few resources, e.g. a regional office, will use the outsourcing approach.

As mentioned before, the architecture aims to reduce the administrative burden in the handling of policies. To do so in the proposed corporation environment, the first step is to establish Inter-Domain Trust Relationships (*IDTR*, Fig. 4). Thus, IDTR are trust relationships used for administrative purposes, mainly to accomplish permission granting between administrators.

IDTRs are established between STSs and are based on SPKI/SDSI mutual STS group (domain) inclusion, i.e., each STS inserts the partner STS in its local group and issues a name certificate denoting group membership. Group membership provides the basis to ensure that an entity can be trusted by others. In the proposal the IDTR is the only way for a principal to know the public key of another principal with certainty, since there are not Certificate Authorities in SPKI/SDSI.

The certificate-based permissions granting is done through the IDTR trust relationship between the STS in corporate and client domain. The administrator of STS_2 (provider) delegates permissions to the administrator of STS_1 (corporate), that in its turn grants permissions to STS_3 (client administrator). An administrator that is grantee of permissions always delegates them to the principals in its domain (event DR_3, Fig. 4), i.e., the administrator never enjoys the permission granted by a certificate. Client administrators can only derive permissions from those coded in the chain of authorization certificates delegated by the corporate administrator. The permissions granted by the branch administrator, through certificates, to clients in its domain serve only to create policies to grant client-specific access to provider's resources.

When the client request cannot be evaluated by either LPDP and or PDP (outsourcing), this means that a policy for such subject (client) concerning the requested resource does not exist. Thus, the client sends the chain of SPKI/SDSI certificates to PEP, that in its turn sends them to STS invoking XKMS (event *ce*, Fig. 4). After obtaining the reduced certificate, STS_2 creates an XACML policy and encapsulates it in an SAML envelope, sending it to the corporate repository (event up_2). Moreover, $BCPM_p$ sends back an alert message to PEP requesting an update of LPAP. After the update, the evaluation is done by LPDP.

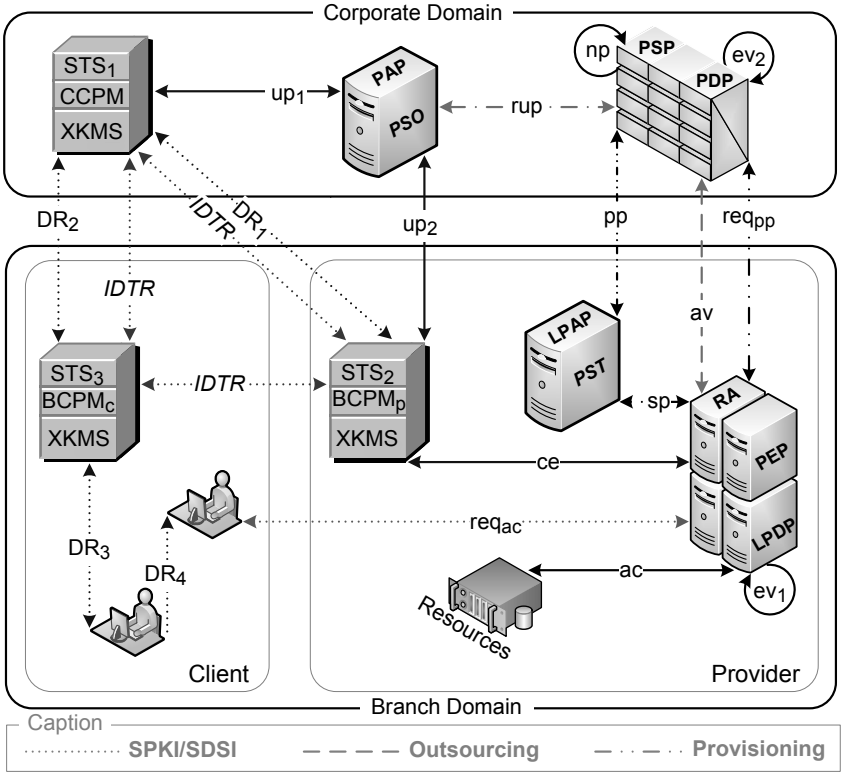

Fig. 4. Overview of proposed architecture

Chains of authorization certificates grant permissions to clients without the need of administrative intervention; the Branch Credentials and Policies Management ($BCPM_C$, Fig. 4) could be a software agent responsible for interacting with a client. Thus, $BCPM_C$ can substitute the client administrator, guided by a set of rules, in order to grant permissions automatically. $BCPM_C$ is an interface for administrators to handle certificates and trust relationships and for clients to interact with STSs in order to obtain permissions. Also, a properly authorized client may forward permissions to other clients (event DR_4, Fig. 4).

$BCPM_P$ is an interface for administrator handling certificates and trust relationship and for send alerts to PEP. Corporate Credentials and Policies Management (CCPM, Fig.

4) is the interface for corporate administrator deals with certificates, policies, provisioning pendency and trust relationship.

SPKI/SDSI, which is independent of the underlying technology, forwards permissions by delegation via a chain of authorization certificates. The advantage of SPKI/SDSI is that through a chain the information about who delegated permissions to whom is maintained within the system. Moreover, the corporate administrator specifies which permissions can be delegated (propagated), thus preventing branch administrators from intentionally or accidentally violating the corporate policy.

Our distributed architecture provides support for certificate-based policy control and for provisioning server-based architectures. This hybrid policy control system aims at low coupling, high interoperability and easier distributed policy management in service-oriented architecture environments. Our scheme favors the decentralization of policy management together with the unification of information regarding which clients can access which resources in the whole corporation environment.

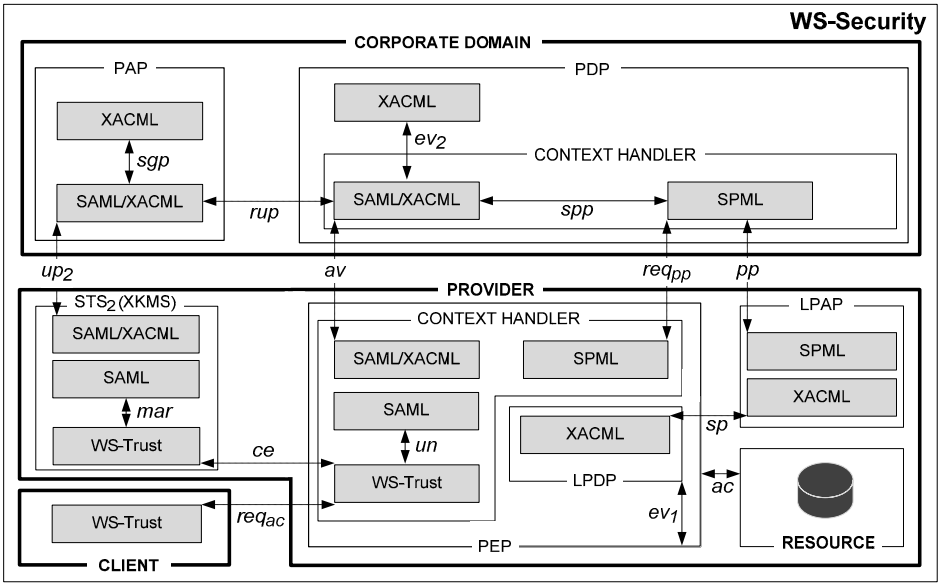

Fig. 5. Proposal overview of WS Protocols and specifications

V. Application Scenario

Corporate access control systems are usually heterogeneous, so they must support branch-specific requirements. Thus, the management of access control policies in a corporation should deal with different needs for many providers and users (clients) distributed across several branches.

Fig. 5 shows some details about the interactions between entities in the deployment of the scenario. The scenario applies Web Services, SPKI/SDSI, and the provisioning approach to implement the proposal. In this scenario, SPKI/SDSI applies digital signatures to all certificates and request messages.

Let us consider that, at any given moment, a client requests access to a given resource (event req_{ac}, Fig. 5). The PEP (Context Handler) receives the request and forwards it to the LPDP (event ev_1), which queries its repository (event *sp*) and does not find any policy applicable to the requested access. Then, the LPDP notifies the PEP that it is not able to evaluate the request.

The PEP sends the evaluation request to the PDP (event *av*, Fig. 5), since the client may be in transit, i.e., it can be external to the domain of the resource provider – a branch of the corporation, for example. Hence, no policy regarding this client was received in the LPAP provisioning. In this case, the evaluation can follow the outsourcing approach; PDP at corporate domain decides (event ev_2) and PEP at the provider's domain does the enforcement (event *ac*).

Now suppose that PDP, after querying PAP (events *rup* and *sgp*, Fig 5), still does not have the policies concerning that client. PDP reports it to PEP, which offers an alternative to the client. Using a challenge-response protocol [18], PEP replies back the client with a message advising the permissions required to access the resource.

The client then asks STS_3 (Fig. 4) for a SPKI/SDSI chain of certificates that grants the required permissions. When such chain is finally obtained, the client sends the access request and the chain of certificates (event req_{ac}) to PEP. In turn, PEP forwards the chain of certificates to STS_2 (event *ce*) so that the STS can reduce the chain to a single certificate.

STS_2 invokes the XKMS module, which reduces the chain and returns a single certificate to STS_2 that generates a XACML policy, encapsulates it in SAML [19] and updates PAP with the new policies (event up_2, Fig. 5). Then, STS_2 through $BCPM_P$ (Fig. 4) returns a notification to PEP (event *ce*, Fig. 5), which requests an LPAP update to PDP (in this case acting as SPML PST). After the LPAP update, the LPDP evaluates the client request allowing the access that is granted by PEP (event *ac*, Fig. 5).

Figure 6 shows the message SOAP sent in the event up_2 (Fig. 5), containing the XACML policy generated from the

reduced SPKI/SDSI certificate (Fig. 3). The XACML policy has as subject the *Public-Key-C*, which represents the client user, and the path of target resource (*www.branch.com/researcher*). As a valid certificate chain grants permission to its last subject, if the policy is generated, then the rule effect in XACML must be the default value "Permit". The *action* field in XACML means the permission *execute* in SPKI/SDSI. An authorization certificate that carries more than one permission for the same resource can generate policy rules with more than one <Action> element. Given that, only one XACML policy rule is generated from each SPKI/SDSI certificate. Moreover, the rule combination algorithm should be "first-applicable" in XACML.

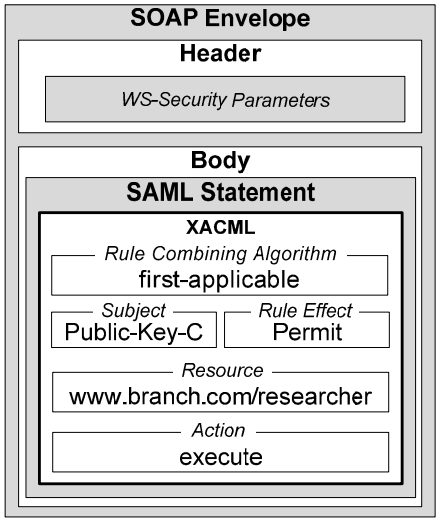

Fig. 6. XACML policy from a reduced certificate in a SOAP packet

VI. PROTOTYPE

The prototype implements the architecture proposed in Fig. 4. It was developed in JAVA (using JDK 1.6) with the support of the following APIs: OpenSPML (www.openspml.org); SAML and the SAML Profile of XACML (www.opensaml.org), and SUN XACML (sunxacml.sourceforge.net). Also, from the Apache Software Foundation it is used, the TomCat application server (tomcat.apache.org), the Axis2 SOAP engine (ws.apache.org/axis2), and the Rampart module for WS-Security and WS-Trust support. The policy repositories use Oracle Berkeley DB XML (www.oracle.com) and the SPKI/SDSI infrastructure was implemented by Morcos [20].

Two scenarios were considered for the prototype evaluation. In the first, we evaluated the influence of central architectural entities on the overall system performance, when security is not applied at the level of SOAP messages. In the second scenario, the same architectural entities under the same conditions were evaluated, but in this time considering the use of digital signatures, encryption, and timestamps, to provide end-to-end message security in the level of WS-Security. During the evaluation the scenario is kept the same, but the approach (provisioning, outsourcing and SPKI/SDSI) employed changes, as well as the use of WS-Security.

In the evaluated Server-based Policy Control Architectures (provisioning and outsourcing), the size of the request message that is sent from the client to the provider is always 247 bytes (event req_{ac}, Fig. 4). The messages exchanged between the PEP and the PDP have a size of 976 bytes (event *av*, Fig. 4).

In the Certificate-Based Policy Control Architecture the access request sent from the client to the PEP (event req_{ac}, Fig. 4), has a size of 4.30 Kbytes, and each certificate that composes this request amounts for 1.78 Kbytes. This request carries the SPKI/SDSI certificate chain (two certificates and three keys) that will provide the access permissions for a client.

Fig. 7 shows that the usage of SPKI/SDSI certificates demands more processing time than the evaluation based on provisioning or outsourcing approach. However, this delay only happens in the first time the client presents the certificate chain to the PEP. After that, the corresponding derived policy is automatically created and the evaluation occurs by the provisioning approach. In such a scheme, the policies are created dynamically, without human administrator interventions.

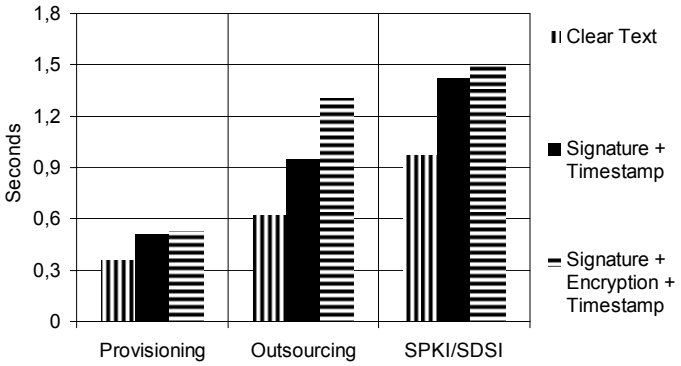

Fig. 7. Response Time for the Client in each approach

The percentages in Fig. 8 are corresponding to the results of the evaluation related to the Response Time for the Client in Fig. 7 (RTC). Analyzing the evaluation of the provisioning approach, we observed that most of the time is spent in the PEP, dealing with XML messages and enforcing the decisions received from the LPDP, which is the entity that spends the least of the total time in the proposed architecture.

In the outsourcing approach, every request sent by the client to the PEP is forwarded to and evaluated by the PDP (events events *av*, *rup*, *sgp* and *ev2*), requiring the computer hosting PEP to wait for the reply be received from the PDP. The client, in its turn, has to wait for message exchanges between the PEP and the PDP.

With the access evaluation using only SPKI/SDSI certificates (Fig. 8) the PEP spent 20% of the RTC with marshalling/unmarshalling of messages and enforcing the LPDP decision. The STS_2 spent 60% of the RTC, 40% of this corresponding to certificate chain validation and SAML assertion creation and 5% corresponding to sending the SAML/XACML policies to PAP. The remainder time is spent on internal marshalling/unmarshalling of messages on the STS_2 (15%). In this operational mode, the PEP is not a critical failure point on the system, since the client and the PEP were

only affected by the time spent on the STS_2 when evaluating the first SPKI/SDSI access request.

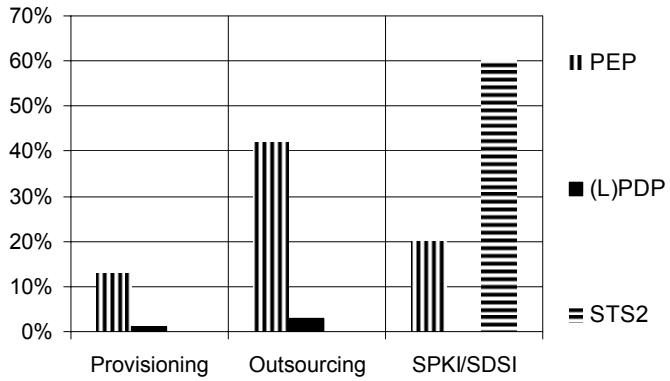

Fig. 8. Percentage of time spent in each entity of each approach

For each new certificate inserted on the chain, the average increase on the response time was 11% without WS security, and 15% with security. This increase in time is related to the additional message size (1.78 Kbytes for each new certificate), and specially related to the increased complexity involved in certificate chain validation, given that more parameters need to be verified (e.g. signatures, authorizations, expiration dates). Thereby, we simulated the delegation of permissions that occurs internally between administrator and clients in branch domains.

In the provisioning approach, the computer hosting the PEP handled 200 concurrent clients without WS security. The same computer was able to serve 150 clients concurrently when applying security. When using only digital signature and timestamp, the same computer served 200 concurrent clients.

In the outsourcing approach, the computer hosting the PEP served 150 clients concurrently without security. The same computer was capable of serving 50 clients concurrently when using WS security. Using only digital signature and timestamp, the host was able to serve 100 clients concurrently.

When evaluating access requests using SPKI/SDSI certificate chains (two certificates and three keys), the computer hosting the PEP was able to handle 75 clients concurrently, without security. This number drops to 50 concurrent clients when applying WS-Security. Remember that, in addition to a timestamp, signature, and encryption applied in the SOAP message, each SPKI/SDSI certificate in the chain has its own signature, which increases the overhead.

Considering the performance when using WS security, one can observe that if confidentiality (WS encryption) is not a requirement, the security can be improved using digital signatures (which provide authenticity, integrity and non-repudiation) and timestamps (which protect against message-replay vulnerability) without unreasonably affecting the overall performance.

The outsourcing approach not only creates a tight coupling with the PDP, but also requires more computing resources, negatively affecting scalability. On the other hand, the approach based on SPKI/SDSI and provisioning creates loosely-coupled security control, given its operational autonomy and reduced dependence of message exchanges between the architectural entities. Additionally, this combination shows that it is possible to reduce human and computing costs involved in policy management while improving web services scalability.

VII. Related Work

Karp [21] considers that Identity Based Access Control (IBAC), which follows a traditional access control approach, is very limited to be applied in SOA. The author argues that Authorization Based Access Control (ABAC) is more adequate to SOA needs. An example of ABAC is SPKI/SDSI. However, the author only discusses the model and does not present an implementation to support his claims. In our proposal we show that SPKI/SDSI (ABAC) is better if combined with provisioning, for example. Applying only SPKI/SDSI in SOA brings an important cost in computing that should not be neglected.

Mello and Fraga [22] propose the use of a trusted intermediary that should be responsible for dealing with distinct security technologies (e.g., SPKI/SDSI and X.509). All the evaluation process uses message exchanges based on the SAML assertion format. In other words, SPKI/SDSI and X.509 credentials are replaced by SAML assertions to obtain a common format and a functional scheme. The use of a mediator implies an overhead, due to the increase in the number of messages exchanged and the additional time to obtain the access credential (e.g., SAML assertion), requested in each authorization evaluation. We consider the usage of the outsourcing approach increases the number of messages exchanged and the dependence on entities from the client domain during the authorization decision process; that works against the principle of loose coupling employed in SOA/Web Services.

Hai-bo and H. Fan [23] propose an access control system, based on attributes, that can cross security domains and be implemented in heterogeneous systems, enabling interactions between different parties who know little of each other. The model provides access to Web Services based on a signed digital credential and enhanced with attributes provided by trusted authorities. The work focused on the description of the entities and in the function of each one in the model. The proposed approach is based on the outsourcing model, allowing that authorization decisions be carried out based on attributes, service parameters and pre-established policies. We consider that the proposed model has the same tight coupling principles mentioned the outsourcing-based previous work.

VIII. Conclusion

This work presented a proposal for the unified management of access control policies in corporation environments. However, policy writing, evaluation, and enforcement are done in a distributed way. Branch domain security entities derive permissions from SPKI/SDSI authorization certificate chains to write new policies. The PAP stores the new policies that are automatically provisioned in branch domains. Thus, policy evaluation can be done in the provisioning approach, which has the smallest computing cost.

The proposed scheme is an alternative means for a client to

obtain required permissions in cases when it is not registered with the Corporate Credentials and Policies Management – distinctly from the traditional approach that in such a case simply would deny the access.

Transposition of client and provider authorization domains is easily obtained through the use of SPKI/SDSI authorization certificates. SPKI/SDSI is independent of the underlying technology, which permits the transport of permissions and the establishment of trust relationships across security domains.

SPKI/SDSI certificates carry all the attributes needed to evaluate the access request, not being required to contact security authorities (administrators) in the client domain to obtain additional attributes to take an authorization decision. Policy provisioning and certificate chains favor loose coupling and interoperability among policy control entities.

The proposed architecture can automatically operate in both server-based policy control approaches: provisioning and outsourcing. Such feature is not present in any other related work. However, in general, our proposal operates in the provisioning approach, because it requires less message exchanges and together with SPKI/SDSI is better fitted to organizations having branches with local autonomy.

The proposed approach is compliant with the server-based approach of Web Services. Nevertheless, it does not depend on an infrastructure based on authentication and authorization servers to transpose security domains, thus favoring the loose coupling of the system. The developed prototype shows the feasibility of the proposal. However, the scenario explored in this paper can be extended to more complex ones.

REFERENCES

[1] C. M. MacKenzie, K. Laskey, F. McCabe, P. F. Brown, R. Metz and B. A. Hamilton, *Reference Model for Service Oriented Architecture*, v. 1.0, OASIS *Std.*, Oct. 2006; http://docs.oasis-open.org/soa-rm/v1.0/.

[2] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris. and D. Orchard, *Web Services Architecture*, W3C note, Feb 2004; http://www.w3.org/TR/ws-arch/.

[3] L. Clement, A. Hately, C. V. Riegen and T. Rogers, *Universal Description Discovery & Integration (UDDI)* v. 3.0.2, OASIS Draft, Oct. 2004; http://uddi.org/pubs/uddi_v3.htm.

[4] R. Chinnici, J. J. Moreau, A. Ryman, S. Weerawarana, *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*, W3C recommendation, June 2007; http://www.w3.org/TR/wsdl20/.

[5] N. Mitra, Y. Lafon, *SOAP Version 1.2 Part 0: Primer (Second Edition)*, W3C recommendation, Apr. 2007; http://www.w3.org/TR/soap12-part0/.

[6] T. Bray, J. Paoli, C. M. S. McQueen, E. Maler, F. Yergeau and J. Cowan, *Extensible Markup Language (XML) 1.1 (Second Edition)*, W3C recommendation, Sep. 2007; http://www.w3.org/TR/xml11/.

[7] A. Nadalin, C. Kaler, R. Monzillo and P. H. Baker, *Web Services Security: SOAP Message Security 1.1 (WS-Security)*, OASIS *Std.*, Feb. 2006; http://docs.oasis-open.org/wss/v1.1/.

[8] A. Nadalin, M. Goodner, M. Gudgin, A. Barbir and H. Granqvist, *WS-Trust 1.3*, OASIS *Std.*, Mar. 2007; http://www.oasis-open.org/specs/index.php#wstrustv1.3.

[9] P. H. Baker and S. H. Mysore, *XML Key Management Specification (XKMS 2.0)*, W3C recommendation, June 2005; http://www.w3.org/TR/xkms2/.

[10] G. Cole, *Service Provisioning Markup Language (SPML) v. 2*, OASIS *Std.*, Apr. 2006; http://www.oasis-open.org/specs/index.php#spmlv2.0.

[11] S. Cantor, J. Kemp, R. Philpott, and E. Maler, *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML)*, v. 2.0, OASIS *Std.*, Mar. 2005; http://www.oasis-open.org/specs/index.php#samlv2.0.

[12] T. Moses, *eXtensible Access Control Markup Language (XACML)*, v. 2.0, OASIS *Std.*, Feb. 2005; http://www.oasis-open.org/specs/index.php#xacmlv2.0.

[13] B. Moore, E. Ellesson, J. Strassner, and A. Westerinen. *Policy Core Information Model*, IETF RFC 3060, Feb. 2001; http://www.ietf.org/rfc/rfc3060.txt.

[14] K. Chan, J. Seligson, D. Durham, S. Gai, K. McCloghrie, S. Herzog, F. Reichmeyer, R. Yavatkar and A. Smith, *COPS Usage for Policy Provisioning (COPS-PR)*, IETF RFC 3084. Mar. 2001; http://www.ietf.org/rfc/rfc3084.txt.

[15] R. L. Rivest and B. Lampson, "*SDSI - A Simple Distributed Security Infrastructure,"* Sep. 1996; http://theory.lcs.mit.edu/~rivest/sdsi10.html, Access: Jan. 2009.

[16] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen, *SPKI Certificate Theory*, IETF RFC 2693, Sep. 1999; http://www.ietf.org/rfc/rfc2693.txt.

[17] T. F. Franco, W. Q. Lima, G. Silvestrin, R. C. Pereira, M. J. B. Almeida, L. M. R. Tarouco, L. Z. Granville, A. Beller, E. Jamhour, and M. Fonseca, "Substituting COPS-PR: An Evaluation of NETCONF and SOAP for Policy Provisioning," *Proc. 7th IEEE Int'l Workshop on Policies for Distributed Systems and Networks* (POLICY'06), IEEE, 2006, pp. 195-204.

[18] NIST FIPS PUB 196, *Entity Authentication Using Public Key Cryptography*, Feb. 1997; http://csrc.nist.gov/publications/fips/fips196/fips196.pdf.

[19] A. Anderson and H. Lockhart, *SAML 2.0 profile of XACML v2.0*, OASIS *Std.*, Feb. 2005; http://docs.oasis-open.org/security/saml/v2.0/.

[20] A. Morcos, *"A Java Implementation of Simple Distributed Security Infrastructure",* master's thesis, Dept. EECS, Massachusetts Institute of Technology, 1998.

[21] A. H. Karp, "Authorization-Based Access Control for the Services Oriented Architecture," *4th Int'l Conf. on Creating, Connecting, and Collaborating through Computing* (C5'06), IEEE, 2006.

[22] E. R. Mello and J. S. Fraga, "Mediation of Trust across Web Services," *Proc. Int'l Conf. on Web Services* (ICWS'05), IEEE, 2005, pp. 515-522 .

[23] S. Hai-bo and H. Fan, "An Attribute-Based Access Control Model for Web Services," *Proc. 7th Int'l Conf. on Parallel and Distributed Computing, Applications and Technologies* (PDCAT'06), IEEE, 2006, pp. 74-79.