ISSN: 0975-8585

Research Journal of Pharmaceutical, Biological and Chemical Sciences

An Extended XACML Model to Secure Biological Web Services using Access Control Policies.

Nirmalrani V [1]*, Saravanan P [2], and Sakthivel P[3].

[1, 2] Department of Information Technology, Faculty of Computing, Sathyabama University, Chennai, Tamil Nadu, India.
[3] Department of Electronics and Communication Engineering, Anna University, Chennai, Tamil Nadu, India.

ABSTRACT

Web service technologies change the software industry by integrating enterprise web services and applications in order to enable the users to access them. Traditional access control solutions, which are based on preliminary identification and authentication of access requester, are not adequate for the web service applications nowadays. As a result, the eXtensible Access Control Markup Language (XACML) has established itself as a solution for controlling access in an interoperable and flexible way. But XACML still suffers from certain limitations which has an impact over its ability of supporting the actual requirements of web services. Web server needs to determine the conditions under which access can be granted, and to communicate them to the requester. In this paper, an expressive solution has been assured by introducing concepts to support access control in open web servers. Here, important features have been proposed about what an access control system should possess and how they can be deployed in the XACML standard. This involves deployment of the concepts in XACML standard which ensures interactions based on access control between web service client and server. By doing so, an efficient access control framework is assured for today's open web services applications. The ensured interaction is to be implemented in Health Care and Biological Web Services applications for service requester and provider.

Keywords: Access Control, Authentication, Authorization, Security, Web Services, XACML.

Corresponding author

ISSN: 0975-8585

INTRODUCTION

An access is to interact with a system entity in order to manipulate, use, gain knowledge of, and / or obtain a representation of some or all of a system entity's resources. Access control is the component of security systems, responsibly to evaluate if a subject can be allowed to operate in a given way on a specific resource. Access Control protects the resources against unauthorized access; a process by which use of resources is regulated according to a security policy and is permitted by only authorized system entities according to that policy. In traditional centralized environments, the focus has been usually directed to the design of solutions able to provide efficient decisions on whether an access request, usually represented as a triple (subject, object, action), should be permitted or refused. Security requirements impose that the control has to be realized for every access to a resource and even a modest increase in the evaluation time can have a great impact on general system performance.

Servers in Web service-based systems receive and process requests from remote parties. In traditional systems web services systems are considered to be open which implies that the servers may not have preliminary knowledge of the requester. As a result of which preliminary identification and authentication of the requestor is required. The requester can enforce her own policy and initiate a negotiation with the server, during which access policies, credentials, and attributes are exchanged, until access is eventually granted or denied.

In a networked environment, information needs to be protected, therefore authorization and access control systems have been studied in the field of computer security for a long time, as a result of that, many access control mechanisms have been developed. Most of these mechanisms focused on how to define users' rights in a precise way to prevent any violation for the access control policy. To some degree classical access control models are not flexible; they either permit access or deny it completely. The access control decision is made based on the assumption that all access needs are known in advance, and these needs can be expressed by machine readable code. In many situations it's hard to predefine all the access needs, or even to express them in machine readable code. One example of such situation is an emergency case which cannot be predictable.

RELATED WORKS

V. Cheng, P. Hung and D. Chiu proposed "Enabling Web Services Policy Negotiation with Privacy Preserved using XACML" [2], in today's business environment, the use of web services technology is becoming more popular. This growth has been met with an increase of security related attacks, which has caused web services providers to adopt stricter security policies. Having the ability to negotiate security contracts, web services providers will have the capability to attract more consumers while keeping the security of the web service at an acceptable level. This paper has presented a novel framework for a system that allows the consumers of web services to negotiate a new security contract with the provider of a web service by incorporating environmental considerations into its decision-making. A list of requirements for a web services security policy negotiation system was defined and analyzed. To be able to use the system with current web services technologies, a standards based implementation was preferred, which affected the choice of policy language and communication protocol. The semi-automated approach to negotiation was selected to offset the inflexibility of current policy languages. Future work aims to investigate the chosen decision model by identifying negotiation objects found in policy documents, and the environment. Their influence on each other needs to be determined to be evaluated by making use of fuzzy techniques.

U. Mbanaso, G. Cooper, D. Chadwick and S. Proctor proposed "Privacy Preserving Trust Authorization Framework using XACML" [4], nowadays many organizations share sensitive services through open network systems and this raises the need for an authorization framework that can interoperate even when the parties have no pre-existing relationships. This paper proposes a way that the widely accepted XACML standard can address three essential security components: trust, privacy and service control in a synchronized manner. It identifies some of the necessary additions to core XACML model that allow for progressive bilateral exchange of policies and credentials between two parties in a way that privacy and trust are sufficiently preserved while protecting access to sensitive services. They have leveraged trust concepts already proposed by researchers and demonstrated how their model can protect hierarchical resources at the same time. Though policy disclosure during trust negotiation is vulnerable to probing attacks, they have demonstrated how their

progressive policy and credential disclosures can minimize this threat. That is, the progressive and incremental paradigm allows the risk to be managed such that exposure to risk at any particular point is limited. Again, since the Trust level X determines what each party can disclose, it is arguable that not much sensitive information is lost during establishment of trust using their framework. The system presented in their paper can be adapted to an existing suite of negotiation strategy to allow participants the option of selecting how fast they can establish trust. Some of the common issues in many of the existing work are

- The XACML (eXtensible Access Control Markup Language) based on preliminary identification and authentication of access requesters are not adequate for web services.
- Security requirements impose even a modest increase in the evaluation time can have an impact on general system performance.
- Low level access control models with limited flexibility have been used.

When privacy is an issue, these works assume that the requester can enforce her own policy and initiate a negotiation with the server, during which access policies, credentials, and attributes are exchanged, until access is eventually granted or denied.

OBJECTIVES OF THE PROPOSED SYSTEM

- In this proposed system an extension to XACML is proposed that enables the server to determine the conditions under which access can be granted, and to communicate them to the requester.
- An enhanced access control framework is achieved with audit management for open web services.
- The XACML architecture is used in order to provide a higher degree of access control solution.
- The components like policy auditor are added in this system, which provide more security and privacy over web services.

ARCHITECTURE OF THE PROPOSED SYSTEM

XACML is an OASIS standard that describes both a policy language and an access control decision request/response language (both written in XML). The policy language is used to describe general access control requirements, and has standard extension points for defining new functions, data types, combining logic, etc. The request/response language lets you form a query to ask whether or not a given action should be allowed, and interpret the result. The response always includes an answer about whether the request should be allowed using one of four values: Permit, Deny, Indeterminate (an error occurred or some required value was missing, so a decision cannot be made) or Not Applicable (the request can't be answered by this service).

XACML engine takes two inputs and gives a single output. Inputs are “XACML Policies” and also called “XACML Request”. Output per request can be one of the following

- Permit – Request is evaluated against all applicable policies given to XACML engine and request is authorized to carry on the operation/actions.
- Deny – Request is evaluated against all applicable policies given to XACML engine and request is not authorized to carry on the operation/actions.
- Not Applicable – XACML engine didn’t find any applicable policy for given request and request is not evaluated in XACML engine.

A typical access control and authorization scenario includes three main entities -- a subject, a resource, and an action -- and their attributes. A subject makes a request for permission to perform an action on a resource. For example, in the access request, “Allow the finance manager to create files in the invoice folder on the finance server,” the subject is the “finance manager,” the target resource is the “invoice folder on the finance server,” and the action is “create files”.

The typical setup is that someone wants to take some action on a resource. They will make a request to Policy Enforcement Point (PEP) which actually protects that resources (like a file system or a web server), the request for authorization lands at the PEP. The PEP will form a request based on the requester's attributes,

ISSN: 0975-8585

the resource in question, the action, and other information pertaining to the request. The PEP creates an XACML request and sends it to the Policy Decision Point (PDP), which evaluates the request and sends back a response. The response can be either access permitted or denied, is returned to the PEP, which can then allow or deny access to the requester. Note that the PEP and PDP might both be contained within a single application, or might be distributed across several servers. In addition to providing request/response and policy languages, XACML also provides the other pieces of this relationship, namely finding a policy that applies to a given request and evaluating the request against that policy to come up with a yes or no answer.

To get into the policies, the PDP uses the Policy Access Point (PAP), which writes policies and policy sets, and makes them available to the PDP. The PDP may also invoke the Policy Information Point (PIP) service to retrieve the attribute values related to the subject, the resource, or the environment. The authorization decision arrived at by the PDP is sent to the PEP. The PEP fulfills the obligations and, based on the authorization decision sent by PDP, either permits or denies access.

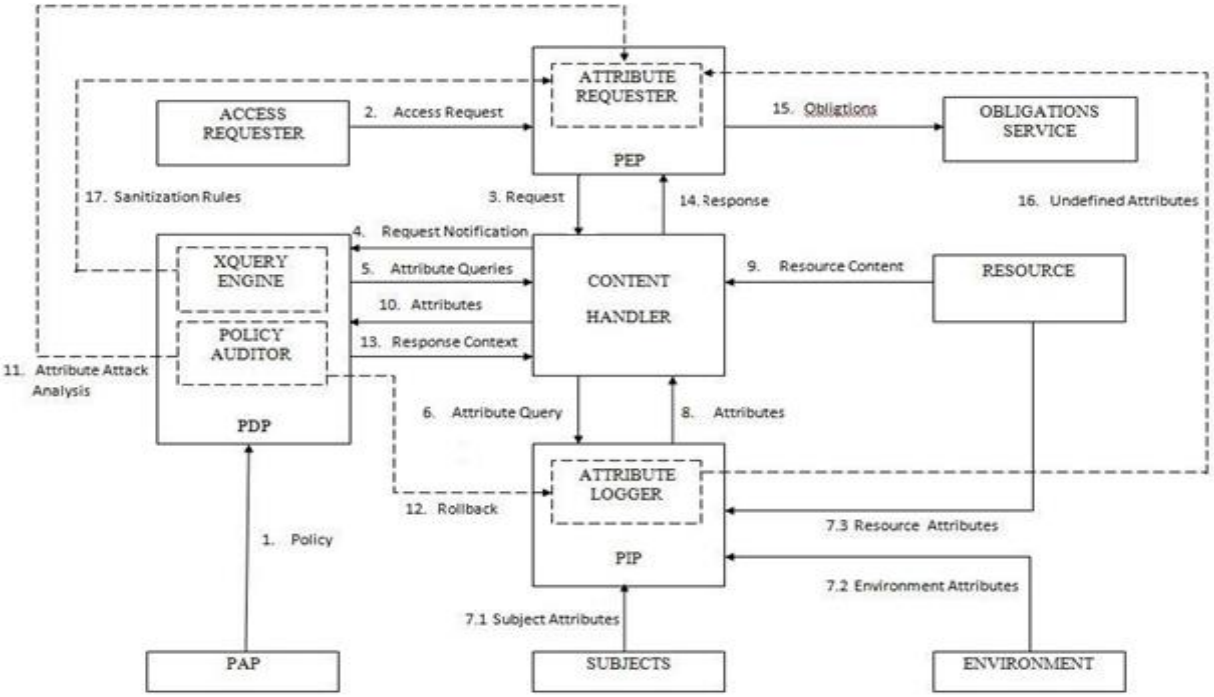

Fig. 1 Proposed System Architecture

XACML provides a standard virtual interface between the policy decision point and the service requester. Because XACML is a policy language set, it is difficult to define a specific technical architecture. XACML can support a variety of underlying infrastructures for policy stores. In summary, XACML has the following key logical architecture components:

- **Context Handler:** The Context Handler transforms service request formats into a format that XACML understands. Architects and developers may need to custom-build processing logic in the XACML context handler to handle the conversion between different service request formats.
- **Policy Decision Point (PDP):** The PDP evaluates the resource access request against the relevant rules, policies, and policy sets. Architects and developers may customize their PDP by building a PDP application as reusable Java components (such as EJB) or simple servlets. Sun's XACML implementation provides a sample PDP.
- **Policy Repository:** The Policy Repository stores the rules, policies, and policy sets that XACML accesses. XACML does not define any standard interface, and leaves it to the implementation to provide an interface to create or retrieve policy objects from the policy repository. Architects and developers may use an existing directory server infrastructure to store all policy objects using LDAP, or they may opt to implement the policy repository using a relational database if their current

entitlement application architecture is database-centric.

- The **Policy Administrator** defines policies and policy sets at the policy administration point. <VariableDefinition> and <VariableReference> elements support reuse of portions of a policy, which provides a macro capability.
- The **Service Requester** issues a request to the policy enforcement point to access the specified resource. This requires fetching the attributes and policies associated with the resource, the action, the environment, and the service requester.
- The **Policy Enforcement Point (PEP)** sends the request for access to the XACML context handler in native request format. This may include the details of attributes of the subjects, resources, actions, and environment.
- The context handler creates an XACML request context and sends a policy evaluation request to the policy decision point.
- The Policy Decision Point queries the context handler for attributes of the subject, resource, action, and environment needed to evaluate the policies.
- The context handler obtains the attributes either from the request context created in Step 4, or it queries a Policy Information Point for the attributes.
- The **Policy Information Point (PIP)** returns the requested attributes to the context handler.
- Optionally, the context handler includes the resource in the context.
- The context handler returns the requested attributes to the PDP. The PDP continues evaluating the policy as attributes are made available.
- The policy sends the response context (including the authorization decision) to the context handler.
- The context handler responds to the PEP, after translating the response context to the native response format of the PEP.
- The PEP executes any relevant obligations. Once the policy is evaluated successfully, the PEP will either grant access to the service requester for the targeted resource or deny the access.

IMPLEMENTATION PHASES OF THE PROPOSES SYSTEM

Identification

XACML enforces attribute based access control to the requester depending on the properties the requester presents and the conditions that the requester satisfies. The PDP component is extended to support the evaluation of conditions that express restrictions on the certification mechanisms. To implement this, restrictions on the credential metadata are defined using ad-hoc XACML profile. And an XSLT style-sheet is defined to translate the ad-hoc syntax to XACML condition syntax.

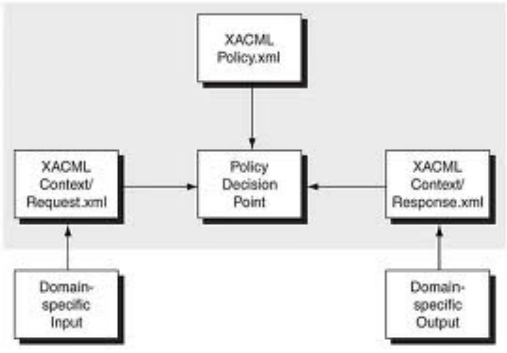

Fig. 2 Identification

Abstraction

Abstractions represent a short hand for expressing a more complex definition using a single concept. The integration of XQuery based approach, for defining and evaluating complex and recursive condition calls and for the development of XQuery engine in PDP component. Id_document can be defined as an abstraction for any element in the set of credentials (identity_card, driver_license, and passport). With the use of XPath

ISSN: 0975-8585

URI, the XQuery engine gets all the relevant attributes from the Context Handler for policy evaluation and expands the XQuery function for recursive conditions.

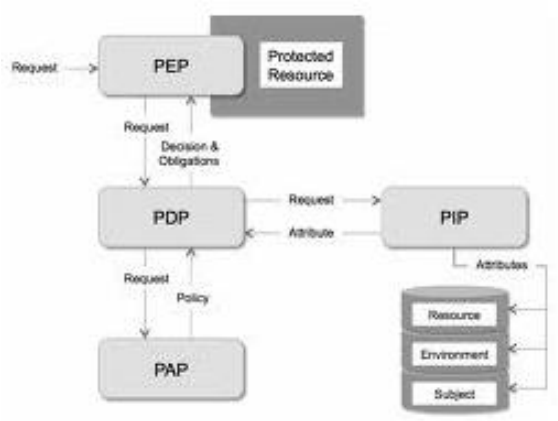

Fig. 3 Abstraction

Recursion

Recursive reasoning is needed on credentials or their properties for providing authorizations to the requester. It is used for expressing policies based in chains of credentials. XQuery engine is used to manage recursion as in abstraction. When PDP needs to reason on abstractions, it extracts the definition of abstractions stored in XACML policies and evaluates them. Figure 4 explains the procedure of recursion.

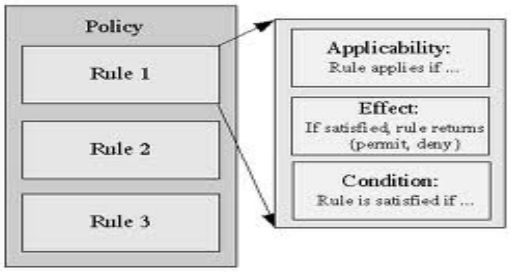

Fig. 4 Recursion

Dialog Management

XACML with a dialog management by which the server can communicate to the requester in time to provide it and acquire access to the service. To manage indeterminate evaluation returned by PDP to the PEP, the standard PEP must have direct communication channel to the PIP to retrieve the missing attributes. The evaluation of a policy can result therefore in four possibilities: permit, deny, not applicable, or indeterminate.

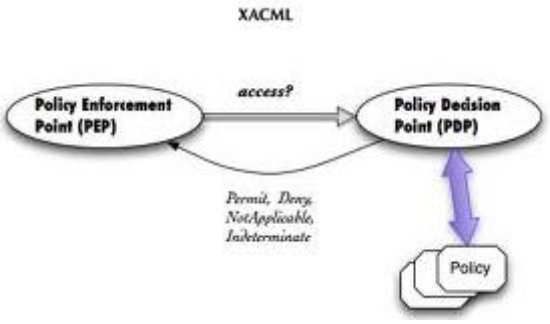

Fig. 5 Dialog Management

ISSN: 0975-8585

Audit Management

To perform a post audit after the process of getting the requested service is done by enhancing the PDP with a Policy Auditor. It performs an audit to check for any attack performed by the intruder and to rollback the accessed services by that intruder by checking the attributes of him.

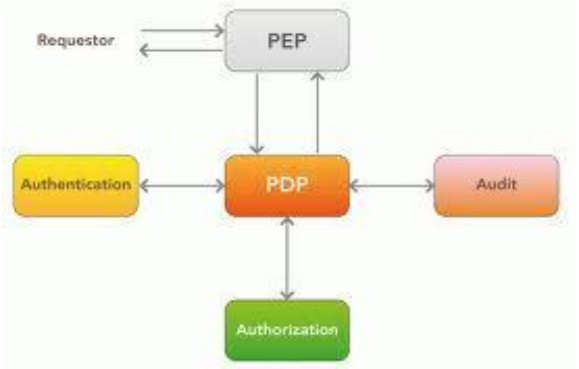

Fig. 6 Audit Management

CONCLUSION

The specific focus of the project is toward the definition of a novel policy language and architecture able to support the privacy requirements of individuals interested in exploiting the services of the web, keeping control on their data. In this scenario, XACML has been identified as a promising solution thanks to its technical features and the widespread support. However, its application in complex advanced scenarios, like the one considered in our project, requires overcoming the limitations discussed, and addressed, in this paper. The project is committed to building a robust prototype of a system supporting the management of privacy policies.

The web services profile of XACML (WS-XACML) aims at defining the aspects related to authorization, access control, and privacy policy in a web service environment. This profile is meant to be used in combination with the WSPolicy proposal. WS-XACML is supposed to wrap XACML policies in a standardized format that can be put in a WSPolicy framework in alternative to policies from various other domains. Another relevant WS-* effort is the Web Service Security (WSS) specification, which prescribes a standard set of SOAP extensions to build secure web services ensuring message integrity and confidentiality. The authorization language we propose is expressive enough to include references to the security domain currently dealt with by the WSS proposal, which our XACML extension is not meant to counter, but to integrate. The Enterprise Privacy Authorization Language (EPAL) is a formal language for writing enterprise privacy policies to govern data handling practices in IT systems. EPAL presents some relevant differences from XACML since it is specifically enterprise-oriented and proprietary.

ACKNOWLEDGEMENT

We thank Sathyabama University for providing us with various resources and unconditional support for carrying out this work.

REFERENCES

[1] C. Ardagna, M. Cremonini, S. De Capitani di Vimercati, and P.Samarati, “A Privacy-Aware Access Control System,” Journal of Computer Security, Vol. 16, No. 4, pp. 369, 2008.

[2] V. Cheng, P. Hung, and D. Chiu, “Enabling Web Services Policy Negotiation with Privacy Preserved Using XACML,” Proc. Ann. Hawaii Int’l Conf. System Sciences (HICSS ’07), Jan. 2007.

[3] Claudio A. Ardagna, Sabrina De Capitani di Vimercati, Stefano Paraboschi, Eros Pedrini, Pierangela Samarati, Mario Verdicchio, “Expressive and Deployable Access Control in Open Web Service Applications”, IEEE Transactions On Services Computing, IEEE Computer Society, Vol. 4 , No.2, April – June 2011.

[4] S.Gowri, G.S.Anandha Mala, and G.Mathivanan, "Classification of Breast Cancer Cells using Novel DPSC Algorithm", Journal of Pure and Applied Microbiology, Vol. 9, No. 2, pp. 1395 – 1400, June 2015.

[5] U. Mbanaso, G. Cooper, D. Chadwick, and S. Proctor, "Privacy Preserving Trust Authorization Framework Using XACML," Proceedings of International Symposium, World of Wireless, Mobile and Multimedia Networks (WOWMOM '06), June 2006.

[6] T. Moses, eXtensible Access Control Markup Language (XACML) Version 2.0, OASIS, 2005.

[7] Nirmalrani V and Sakthivel P, "Design and Implementation of A-RBAC Model for Services in Distributed SOA", in the proceedings of National Conference on Emerging Trends in Information and Communication Technologies, SRM University, Chennai, September 2012, pp. 101 – 107.

[8] Nirmalrani V and Sakthivel P, "Protection of Resources using Role Based Access Control with Multilevel Authentication", International Reviews on Computers and Softwares (I.RE.CO.S.), Vol. 9, No. 11, pp. 1867 – 1874, November 2014.

[9] Nirmalrani V and Sakthivel P, "Framework for Providing Access to Web Data Bases using Budget Aware Role Based Access Control", Journal of Theoretical and Applied Information Technology (JATIT), Vol. 76, No. 3, pp. 296 – 308, June 2015.

[10] P.Saravanan and P.Sailakshmi, "Missing Value Imputation using Fuzzy Possibilistic C Means Optimized with Support Vector Regression and Genetic Algorithm", Journal of Theoretical and Applied Information Technology (JATIT), Vol. 72, No.1, pp. 34 – 39, February 2015.