Proposing a Secure XACML architecture ensuring privacy and trust

Yared Keleta, J.H.P Eloff, H.S Venter

[1]yared@cs.up.ac.za
[2]eloff@cs.up.ac.za
[3]hein@cs.up.ac.za

Information and Computer Security Architectures Research Group
(ICSA)
Department of Computer Science
University of Pretoria

Abstract

The Extensible Access Control Markup Language (XACML) is a platform independent standard based access control policy specification language. It defines rules on how authorization decisions from evaluating applicable access control policies are combined. However, it fails to incorporate built-in trust and privacy-enhancing mechanisms. There are some possible attacks that are identified in the specification that can potentially breach the security of a system using XACML. They are: unauthorized disclosure, message replay, message insertion, message deletion and message modification. In addition to these, there are no mechanisms in place to ensure the confidentiality and integrity of messages in transit between the components of the standard XACML architecture. This paper will briefly investigate the security loop holes in the XACML architecture and proposes an architecture that incorporates built-in trust and privacy features.

Keywords: Architecture, Possible Attacks, Access control policy

1. Introduction

There is an increasing need for information systems integration in organizations. This is because various business applications are developed in different technologies, and run on different platforms. However, it is important to examine if there are any side effects and security implications that could subsequently emerge after the integration process. For instance, departments within an organization may have different information systems, and each information system may have its own proprietary access control implementation. Thus, the integration of these information systems could have various security implications such as inconsistencies in authorization decisions due to many points of enforcement of access control policies within the same organization. Therefore, in order to avoid such inconsistencies and put an appropriate common access control mechanisms in place, the security infrastructure of each information system should be carefully understood before any integration takes place.

Due to the fact that there are various implementations of proprietary access control mechanisms, addressing the security requirements of an organization makes an integration process reasonably complicated. That is why there is a need for a common standard-based access control policy specification language that could be deployed in heterogeneous environments such as in Web services. Since Web services are mainly designed for the purpose of integration of different applications and platforms, it is very important to have a standard-based access control specification language that could be used by all the interacting parties that ensures only authorized users have access to a resource. The main idea is to find a convenient access control mechanism which can interoperate easily with any information system.

Taking the need for a standard-based access control policy specification language into consideration, the Organization for Advancement of Structured Information Standard (OASIS) ratified the Extensible Access Control Markup Language (XACML) [1]. XACML is believed to be the best candidate for an access control policy specification language in Web services. Although XACML is a powerful tool for specifying access control policies, and processing access requests and authorization decisions, it lacks certain built-in trust and privacy-enhancing mechanisms. There are some possible attacks identified in the specification that can potentially breach the security of a system using the standard XACML architecture [1]. They are: unauthorized disclosure, message replay, message insertion, message deletion and message modification. In addition to these, there are no mechanisms in place to ensure the confidentiality, integrity and availability of access control policies.

This paper gives an overview of the XACML architecture and briefly discusses the interactions between the components of the architecture. It emphasizes on the possible attack scenarios that could potentially threaten an XACML-based system. The background section gives a broad overview of the XACML architecture. Section 3 discusses the possible attacks that can take place on an XACML-based system. In Section 4, we propose an XACML architecture with built-in trust and privacy features. Finally, the conclusion section summarizes the paper by highlighting the main concepts discussed in the paper.

2. Background

XACML is a platform independent standard-based access control policy specification language which is described in Extensible Markup Language (XML) [5]. It provides an architecture that is designed to enable organizations to specify access control policies. It has several advantages, some of which are: extensibility, flexibility, reusability, scalability and a policy referencing mechanism [2]. Besides this, it supports a wide range of data types and functions that can be used in various environments, and defines rules on how authorization decisions from evaluating applicable policies are combined. It is therefore believed to be used as a replacement to various application-specific and proprietary access control policy specification languages.

Figure 1 XACML Architecture

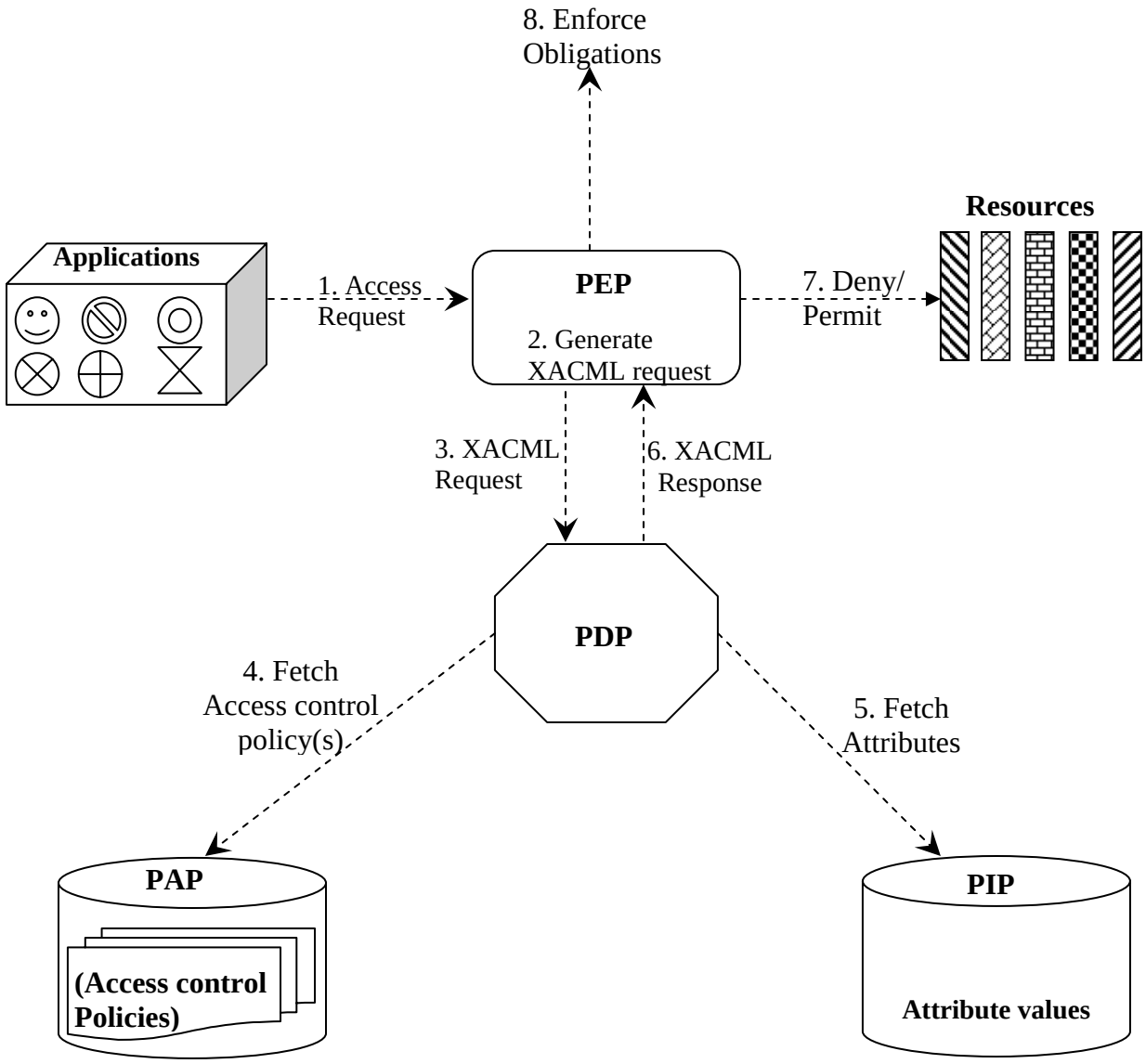


Figure 1 depicts the standard XACML architecture which is the author's interpretation. It illustrates the interaction between the components in the architecture. The main components of the XACML architecture as shown in figure 1 are discussed below.

a) **The Policy decision point (PDP)**

The PDP receives an XACML request, fetches the applicable policy(s) from the policy administration point, retrieves the attribute values from the policy enforcement point, evaluates the request against the applicable access control policies and returns an authorization decision to the PEP.

b) **The Policy enforcement point (PEP)**

The PEP receives an access request, extracts the attributes in the request, generates an XACML request and sends it to the PDP for evaluation. It also makes sure that all the obligations with an authorization decision are executed. An obligation is an action that should be performed together with the enforcement of an authorization decision, and is specified in an access control policy.

c) **The Policy administration point (PAP)**

The PAP creates an XACML access control policy(s) and stores it in a policy database server. It addition to this, it sets a restriction in order to prevent unauthorized access to the access control policies. Beside this, it conducts a regular check in order to maintain the uniqueness of policy identifiers.

d) **The Policy information point (PIP)**

The PIP is a component that acts as a directory server that stores the attribute values and makes it available to the PDP. Attributes values are the data that describe the characteristics of a subject, resource, action and environment. Examples of attributes include: the subject's name, ID and login ID, the time of the day, the name of the resource, and what action (read, write, execute) is required.

Moreover, in figure 1, applications refer to any machine, program or any client that may request an access to a resource in an XACML-based system. Resources refer to any document, information, file or data that resides in an XACML-based system. The next section discusses the possible attack scenarios on a system using XACML.

3. Possible trust and privacy attacks on an XACML-based system

Although XACML is believed to bring an integrated solution to access control requirements in heterogeneous environments, it has some trust and privacy threats. This is because XACML does not have built-in mechanisms that contribute to preserving the trust between the components and privacy of messages in transit in the architecture. Some of the potential trust and privacy threats that could lead into a compromise of an XACML-based system are explained below.

3.1 Confidentiality of access requests and authorization decisions

It is important to put appropriate safeguards in place to protect decision requests and authorization decisions from several attacks. Some of which could be: unauthorized disclosure, message replay, message insertion, message deletion and modification. For instance, consider a typical data flow scenario in an XACML-based system where the PEP sends an XACML request to the PDP. There are no mechanisms in place that ensure whether or not messages in transit are safe from attacks. For example, if an adversary manages to gain access to the communication channel between the PDP and the PEP, he would then be able to intercept XACML requests and authorization decisions easily which in turn enable him/her to insert, modify, interpret or delete messages. This unauthorized disclosure of information causes a compromise to the privacy of the users in the system. In addition to these, the adversary can effectively observe and record legitimate messages which could potentially equip him/her for other attacks such as message replay. Message replay is an attack in which an adversary can easily forge decision requests and authorization decisions using previously recorded legitimate messages.

3.2 Integrity of messages (request or response) in transit

There are no mechanisms put in place to ensure the integrity of messages in transit. For example, when the PDP receives an XACML request, how does it know that the request has not been modified while it was in transit?

3.3 Trust between the PDP and PEP

The main question that may arise here is: how does the PDP ensure whether the XACML request it received was indeed sent from the PEP? Similarly, how does the PEP know whether the authorization decision was indeed sent from the PDP? Therefore it is very important to establish trust between the PDP and the PEP, because if there is an appropriate trust enforcing mechanism in place, the PDP and PEP do not need to be concerned about the identity of senders of a message.

3.4 Trust between the PDP and PAP

Similarly, the correctness of the result of evaluation of the XACML request by the PDP depends mainly on the integrity of the access control policies that are created and supplied by the PAP. In addition to this, access control policies that are stored in a policy server should be protected from any threats such as policy modification, deletion and insertion. The PAP has to check the uniqueness of the policies on a regular base.

3.5 Privacy of users

Disclosure of information such as the requestor's identity in the decision request has a huge impact to the privacy of the users in the system. Appropriate safeguards should be adequately put into force to prevent the communication channel between the PDP and the PEP from being intercepted by an adversary. In addition to this, if the policy server is not secure enough from any kind of unauthorized access, an adversary can gain access to private

information of users (service requestors) and misuse it accordingly. Besides this, the communication channel of the access requests and authorization decisions, including all the communications that may occur between the components in the XACML architecture, should be secure enough to prevent from unauthorized disclosure of the information stored in the messages.

4. A proposed XACML architecture with inherent trust and privacy features

4.1 Requirements

Secure connection between the four components of the architecture is very important, because it is the basis for enhancing the integrity and confidentiality of the messages that are sent to and fro in the XACML-based system. Moreover, it enhances privacy by protecting private information of users from unauthorized disclosure. There are several ways of achieving secure connection. However, it depends not only on the type of connection that is required to secure the link between each component, but also on the physical location of the components. For instance, if the components are all located in a small area, it would be possible to achieve secure connection by protecting the wires between the components physically [3]. However, due to the fact that the components may not necessarily be located on the same machine, it is not practical to achieve a secure connection without employing message integrity and confidentiality-enhancing mechanisms such as encryption. In cases where the components are distributed in the network, End-to-End Encryption [3] could be used provided that all the nodes where the components are situated are secure. It is therefore of utmost importance for the architecture to incorporate inherent security, privacy and trust enhancing features using appropriate existing technologies.

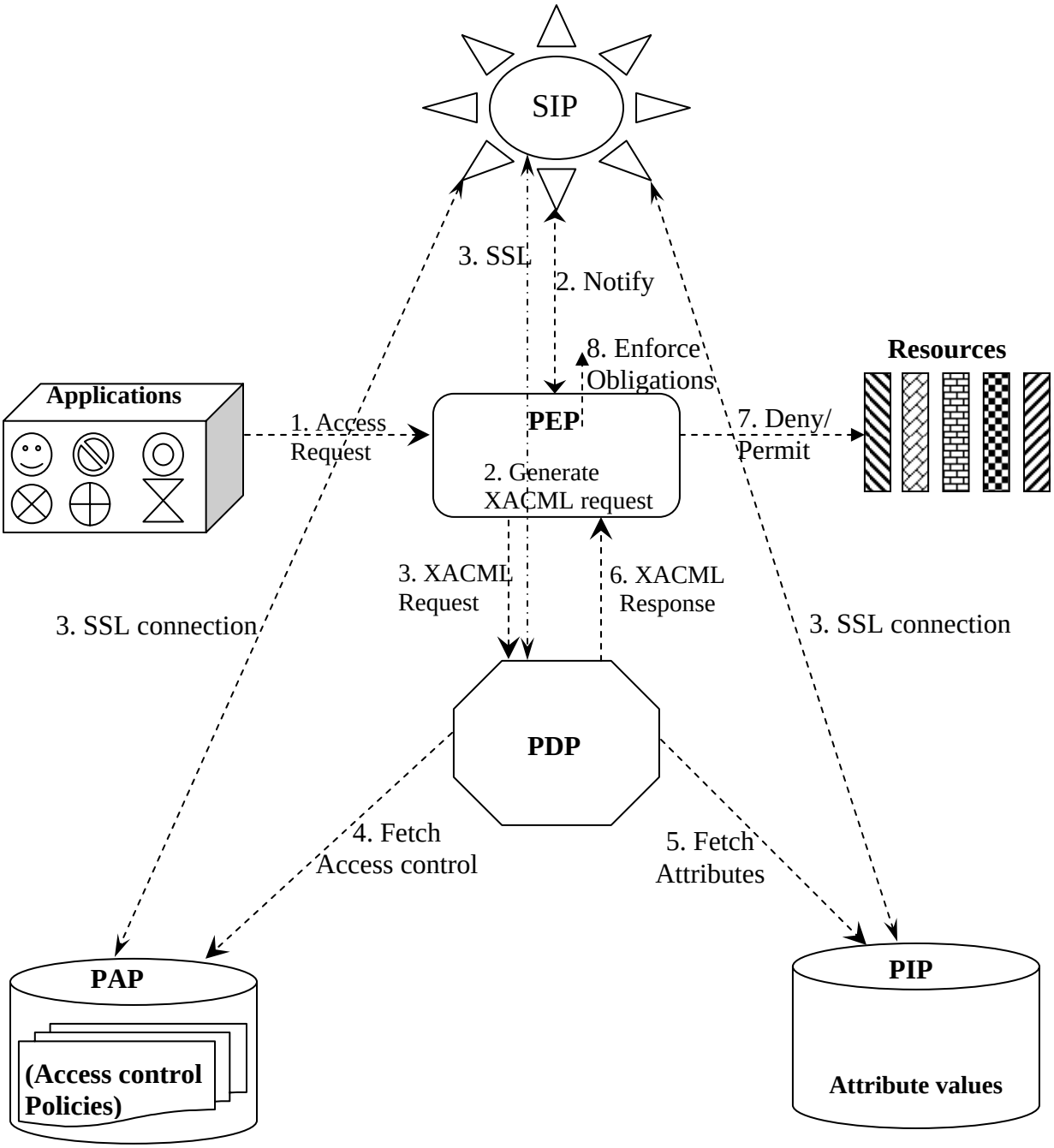
Achieving the desired level of security, privacy and trust is not an easy task, because maintenance and configuration of each component independently is not only an administrative burden, but also is expensive. Therefore, we believe that this could be solved by adding a new component that handles all security-related matters to the architecture. The next subsection discusses the proposed architecture that incorporates all the requirements discussed above.

4.2 The proposed architecture

We propose an XACML architecture (refer to figure 2) that incorporates inherent security features. In order to achieve this, we introduce a new component called the Security Information Point (SIP) to the XACML architecture. The SIP is a component that is used mainly to handle all security-related requirements in the XACML-based system. The SIP component, acting as a security administration center, sets up and maintains a secure communication session, and performs all the management of security within the XACML system. Some of the main functions of the SIP are discussed in the subsections that follow.

4.2.1 Generate random secret keys for encryption: When the SIP component receives a notification from the PEP that an access request has been sent to the PEP, it generates a random secret key and establishes an SSL connection with all the other components and distributes the randomly generated secret key. Diffie-Hellman key exchange is used to randomly generate and distribute the secret keys, because, keys generated with Diffie-Hellman key exchange are proved to be random and secure [6]. For every access request, new secret keys are generated and distributed to the components replacing the old one. By doing so, every component uses this secret key to encrypt the messages before it is sent to other components.

Figure 2 proposed XACML Architecture



This approach has several advantages, some of which are:

- The adversary can not have an opportunity to find the secret key.
- It ensures the confidentiality of the messages in transit.
- More importantly, this helps for establishing and maintaining dynamic trust between the components.
- It will be difficult for an adversary to intercept messages.
- Privacy of information in the messages is protected from unauthorized disclosure.
- A new secret key is generated for every access request.

4.2.2 Verification of the Integrity of access control policies

The PAP will have a mechanism to regularly verify the uniqueness and integrity of the access control policies regularly. It will have a feature that notifies the owner of the policy if any sort of policy modification is noticed.

4.2.3 Management of security-related information and coordination between Components

Here the SIP component plays an important administrative role in the architecture. It has a well-defined infrastructure to store all the security-related information of the other components in the entire architecture. It conducts a regular check whether the components are communicating securely. All the components will have a feature that enables to send a report to the SIP component in cases where the components identify a suspicious incident. Furthermore, it provides a platform to keep track of all the security-related incidents and store it in a location that is only known to the administrator of the system under consideration. Keeping track of users' behavior and recording it, is also one of the tasks of the SIP.

4.3 Data flow in the proposed architecture

The data flow in the proposed architecture works as follows: the PEP receives an access request, and then it immediately notifies the SIP. The SIP component then establishes a secure socket layer (SSL) connection with the other four components and distributes to each of them a randomly generated secret key alone with other security related information. Note that the SIP component generates a secret key for every access request. The other components then use this randomly generated secret key that is obtained from the SIP to encrypt the data in order to verify the origin of the messages. After a successful secure communication session between components, the PEP finally notifies the SIP component the authorization decision that is reached from evaluating a specific access request.

5. Conclusion

The approach we followed in this paper has several advantages; this is due to the fact that all the security-related information is handled by the SIP component, so that the other components do not have to be concerned about the identity of the other components they are interacting with. The proposed architecture does not only guarantee a secure connection, but also establishes dynamic trust between the components based on a run-time interaction. In addition to this, it preserves the privacy of the users by establishing and maintaining secure communication channels between the components. We believe our approach is a good starting point in achieving a well-integrated and secure XACML architecture.

References

[1] eXtensible Access Control Markup Language (XACML) Version 1.1Commeette specification Aug 07, 2003 Retrieved from http://www.oasis-open.org/committees/download.php/2406/oasis-xacml-1.0.pdf Accessed on March 10, 2005

[2] Sun's XACML Implementation, Programmer's Guide for Version 1.2 July 11, 2004, Retrieved from http://sunxacml.sourceforge.net/guide.html Accessed on Feb 26, 2005

[3] Barry M. Leiner, Matt Bishop, Access control and Privacy in Large Distributed Systems, AIAA/ASIS/DODCI Second Aerospace Computer Security Conference: A collection of technical papers pp. 95-98 (Dec. 1986)

[4] Advanced encryption standard (AES), November 26, 2001, Federal Information Processing Standards Publication 197 Retrieved from http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf Accessed on April 6, 2005

[5] XML, Extensible Markup Language (XML) Retrieved from www.w3.org/XML/ Accessed on March29, 2005

[6] Secret Key Distribution, Diffie-Hellman Key Exchange, Retrieved from http://csrc.nist.gov/publications/nistpubs/800-7/node209.html, March 19, 2005