Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Security Essentials: Network, Endpoint, and Cloud (Security 401)"
at http://www.giac.org/registration/gsec

An Introduction to XACML

GIAC Security Essentials
Practical Assignment
Version 1.4b (August 2002)

Michael W Armstrong

June 29, 2003

© SANS Institute 2003, Author retains full rights.

An Introduction to XACML

Abstract

The security policies of most large organizations have many elements as well as many points of enforcement. Multiple departments within the organization may manage the elements of policy. Policies may be enforced by many systems within the organization, to include extranets, mail systems, WANs and/or remote-access systems. Currently, most organizations manage the configuration of each point of enforcement independently for each system. Attempting to change such policies is expensive and can produce inconsistent results. Additionally, it is difficult to obtain a complete view of all the policies in effect, much less test these policies for consistency across all systems. In order to manage all of these policies consistently, it is necessary to have a common language in which to express these policies.[1]

The eXtensible Access Control Markup Language (XACML) is an XML-based standard that provides a policy language and an access control decision / response language for managing access to resources.[2] It is designed to provide a universal language for authorization policy to enable interoperability with a wide range of administrative and authorization tools. By doing so, it allows for security policies to be applied consistently across different environments and across different vendor products.[3]

XML

XML (eXtensible Markup Language) is a tag-based metalanguage used to define document markup. It provides a generic syntax for marking up data with simple, human-readable tags. Data is included in XML documents as strings of text surrounded by text markup that describes the data. XML does not have a fixed set of tags; developers create their own tags as needed to define the data. XML is flexible in how it allows the tags to be defined to describe the data within the document. However, XML is strict in defining how the tags are structured within the document. The grammar defined by the XML standard is what allows XML documents to be easily read and understood by various programming languages across multiple platforms.[4] Even though XML is composed of text that is human-readable, it is meant to be read and generated by computer programs.

1 XACML ratified as OASIS Open Standard. http://www.webservices.org/index.php/article/articleview/909/1/3/

2 Sun Microsystems, Inc. http://sunxacml.sourceforge.net/guide.html

3 XACML Access Control Markup Language. http://www.webdesk.com/xacml/

4 Harold & Means, XML in a Nutshell: A Desktop Quick Reference

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

XML allows the developer to formally describe the syntax of the data being presented. This allows data to be shared between applications as well as between computer programming languages, making XML an effective choice for exchanging data in a platform-independent and language-independent manner.[5]

The tags used to delimit XML data also can be used to describe the data. Additionally, attributes can be used within the tags to further describe the data. For example:

```
<person id="111000" type="employee">
   <name>
      <last_name>Smith</last_name>
      <first_name>John</first_name>
   </name>
</person>
```

The tags used in this example (the text enclosed in <>) provide enough information to tell the reader what (or who) is being described. The attributes (id and type attributes on the person tag) provide the reader with additional information that further describes the role this person plays in an organization.

OASIS

The Organization for the Advancement of Structured Information Standards (OASIS) is a non-profit organization responsible for developing worldwide standards for e-business.[6] These include standards for security, Web Services, XML conformance, business transactions, electronic publishing, topic maps, and interoperability within and between marketplaces.[7] OASIS's mission is to drive the development, convergence, and adoption of structured information standards in order to allow products from different vendors to interoperate.[8] This will allow end-user customers greater flexibility in implementing a common security language across multiple vendor platforms.

Overview

XACML defines a standard set of XML elements with which to define an access control policy language and a request/response language for two-way communications.[9] This policy language is used to depict general access control

5 McLaughlin, Java and XML

6 OASIS. http://www.oasis-open.org/who/

7 Organization for the Advancement of Structured Information Standards (OASIS). http://www.service-architecture.com/web-services/articles/organization_for_the_advancement_of_structured_information_standards_oasis.html

8 OASIS. http://www.oasis-open.org/who/

9 McCarthy, Vance. XACML -- A No-Nonsense Developer's Guide. http://www.oetrends.com/cgi-bin/page_display.cgi?180

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

requirements.[10] It also provides standard extension points for combining individual rules and policies into policy sets and for defining new data types and flexible definitions of the procedures by which rules and policies are combined. By combining the rules and policies that apply to a request, the security application is able to render an authorization decision.

A typical scenario for using XACML would be for an application to check to see if a user (an actor) has access to a particular function within a certain module. For example, suppose a bank employee wanted to access a customer's account for inquiry only. In order to do so, the employee would log on to the bank's account system, enter the account number, and retrieve the account information. Behind the scenes, the account application would make a request to the service that protects the account inquiry function. This service (the Policy Enforcement Point, or PEP) would then create a request based on the employee's identity and attributes, the module and function (account/inquiry) requested, and any other information needed. The PEP then sends its request to the Policy Decision Point (PDP), which evaluates the request and applies the applicable policies to the request. The PDP then returns a response indicating whether the user has access.[11]

The employee from the previous scenario is referred to as a Subject in XACML. A Subject is an actor in the system with attributes that may be referenced by a predicate. The module to be accessed (account system) is called the Resource and the function to be performed (inquiry) is referred to as an Action. Attributes refer to the characteristics of the Subject, Resource, or Action and a Predicate is a statement about attributes, which can be evaluated as either true or false.[12] Some request may have more than one Subject. For example, if a customer comes into the bank to cash a check and the account on which the check is drawn indicates that two signatures are required, the authorization request could potentially contain four subjects: the customer trying to cash the check, both signatories to the check, and the teller performing the transaction.

Once the PDP receives a request, it must search through its information and find the particular policy or set of policies that apply to the request. Once a policy has been found to apply to the request, its rules are evaluated. A policy can have multiple rules, made up of multiple conditions. conditions evaluate to True, False, or Indeterminate. If it evaluates to Indeterminate, it means that there was an error in processing the Rules, such as there was not enough information in the request to make a decision. Once all of the Conditions have been evaluated, the Rule returns an Effect of Permit, Deny, or Indeterminate.[13]

10 Sun Microsystems, Inc. http://sunxacml.sourceforge.net/guide.html

11 Sun Microsystems, Inc. http://sunxacml.sourceforge.net/guide.html

12 Godik & Moses, ed., OASIS eXtensible Access Control Markup Language (XACML) Version 1.0

13 Godik & Moses, ed., OASIS eXtensible Access Control Markup Language (XACML) Version 1.0

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

XACML defines four layers to access policy control:[14]

1. The Policy Administration Point (PAP) creates security policies and stores these policies in the appropriate repository.

2. The Policy Enforcement Point (PEP) performs access control by making decision requests and enforcing authorization decisions.

3. Policy Information Point (PIP) serves as the source of attribute values, or the data required for policy evaluation.

4. The Policy Decision Point (PDP) evaluates the applicable policy and renders an authorization decision.

As defined in, and quoted from, the OASIS standard[15], data flows through an XACML model by the following steps:

1. PAPs write policies and policy sets and make them available to the PDP. These policies or policy sets represent the complete policy for a specified target.

2. The access requester sends a request for access to the PEP.

3. The PEP sends the request for access to the context handler in its native request format, optionally including attributes of the subjects, resource and action. The context handler constructs an XACML request context in accordance with steps 4, 5, 6 and 7.

4. Subject, resource and environment attributes may be requested from a PIP.

5. The PIP obtains the requested attributes.

6. The PIP returns the requested attributes to the context handler.

7. Optionally, the context handler includes the resource in the context.

8. The context handler sends a decision request, including the target, to the PDP. The PDP identifies the applicable policy and retrieves the required attributes and (optionally) the resource from the context handler. The PDP evaluates the policy.

14 Godik & Moses, ed., OASIS eXtensible Access Control Markup Language (XACML) Version 1.0

15 Godik & Moses, ed., OASIS eXtensible Access Control Markup Language (XACML) Version 1.0

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

9. The PDP returns the response context (including the authorization decision) to the context handler.

10. The context handler translates the response context to the native response format of the PEP. The context handler returns the response to the PEP.

11. The PEP fulfills the obligations.

12. If access is permitted, then the PEP permits access to the resource; otherwise, it denies access.

Policy

An XACML policy document has as its root element either a Policy or a PolicySet.[16] A Policy is the set of the rules and obligations that apply to a request. XACML also provides a PolicySet, which provides a container that can hold other Policies, PolicySets, and optionally Obligations. A Policy consists of a Target, one or more Rules, and an optional set of Obligations. Target specifies which conditions must be met in order for the Policy to apply to a Subject, Resource, and Action. A Target uses boolean evaluations to determine if the values in a request meet the conditions of the Target. If all of the conditions are met, the Target's associated Policy, PolicySet, or Rule applies to the request.[17]

Each Policy defines a rule-combining algorithm that is used to combine its contained rules and obligations in order to render a decision. A PolicySet defines a policy-combining algorithm to do the same with its included Policies. XACML defines seven common Combining Algorithms and implementers can define additional ones as well. These Combining Algorithms can be nested to provide increasingly complex Policies. Examples of Combining Algorithms are:[18]

1. Deny-overrides – this algorithm states that of any evaluation returns Deny, then the result must be Deny. If all rules evaluate to Permit, then the result is Permit.

2. Permit-overrides - In the entire set of rules in the policy, if any rule evaluates to Permit, then the result of is Permit. If any rule evaluates to Deny and all other rules evaluate to NotApplicable, then the result is Deny. If all rules are found to be NotApplicable, then the result is NotApplicable.

16 Sun Microsystems, Inc. http://sunxacml.sourceforge.net/guide.html

17 Godik & Moses, ed., OASIS eXtensible Access Control Markup Language (XACML) Version 1.0

18 Godik & Moses, ed., OASIS eXtensible Access Control Markup Language (XACML) Version 1.0

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

3. First applicable - each rule is evaluated in the order in which it is listed in the policy. For each rule, if the target matches and the condition evaluates to True, then the result of that rule will be the evaluation of the policy (either Permit, Deny, or Indeterminate). Otherwise, the algorithm goes to the next rule. If no rule applies, then the result is NotApplicable.

4. Only-one-applicable – For all of policies in the policy set, if no policy applies, then the result is NotApplicable. If more than one policy applies, then the result is Indeterminate. If only one policy applies, then the result is the result of evaluating that policy.

A Policy or PolicySet can define Obligations, which will be passed back in the response from the PDP. An Obligation is operation that should be performed in conjunction with the enforcement of an authorization decision. An example of an Obligation would be to provide notification to a customer after a wire-transfer operation has been performed by the bank. Only Policy's which are evaluated and have returned a response of Permit or Deny will return Obligations and, of the Obligations returned from the PDP, only the Policy's which have returned the same result as the PDP's result will have their corresponding Obligations returned to the PEP. The PEP is responsible for fulfilling the Obligations it receives.[19]

Rules

XACML defines rules as targets, effects and conditions. Targets (discussed above under Policy) include resources, subjects, and actions. A Condition is a function that evaluates to either True, False or Indeterminate, with Indeterminate indicating that there was not enough information to determine an exact answer.[20]

A Rule's Target set identifies the resources, subjects, and/or actions to which the rule applies. Conditions can be placed on these Targets to further refine the definition of to what the rule applies. If a rule applies to all entities of a particular data type, then an empty XML element named <AnySubject/>, <AnyResource/> or <AnyAction/> is used to denote that. If the Rule has no Target element, then the Policy Target element applies to the Rule.[21]

The Effect of the Rule is the result the Rule returns if the Rule evaluates to True. Two values can be returned: Permit or Deny.[22]

19 Godik & Moses, ed., OASIS eXtensible Access Control Markup Language (XACML) Version 1.0

20 Godik & Moses, ed., OASIS eXtensible Access Control Markup Language (XACML) Version 1.0

21 Godik & Moses, ed., OASIS eXtensible Access Control Markup Language (XACML) Version 1.0

22 Godik & Moses, ed., OASIS eXtensible Access Control Markup Language (XACML) Version 1.0

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Attributes

The primary pieces of data with which Rules work are attributes, or the characteristics of a subject, resource, action or environment. Examples are a user's name, login ID, or department, or the time of day or the account number to be accessed. Requests sent from the PEP to the PDP are composed almost entirely of attributes. These attributes are what are used to make the security decision.[23]

Attributes describe. For example, an employee at a bank may have the attribute of being a teller, a loan officer, an operations specialist, or many other job roles. Based on this attribute, that employee would be able to perform different functions within the bank. A teller can enter transactions on accounts into a queue for processing, but cannot approve a loan. A loan officer can approve loans, but may only be able to read account values and not change them. The operations specialist may actually process the account transactions.

As stated earlier, attributes apply not only to the Subject requesting access. Attributes can apply to the object being accessed, the action being taken, and the environment in which the system is running. For example, a particular function may be dependent on the time of day. In our banking example, transferring funds may be limited to the hours of 9:00 AM to 2:00 PM. The current time would be an attribute of the system rather than of the requestor. And if a transfer request is entered, the account's attributes would have to show that there is enough money available to effect the transfer. In this case, multiple attributes may have to be summed in order to arrive at an account balance.

On any request made, the PDP may find that there are multiple values for a given attribute. When retrieving attributes, the PDP receives what is referred to as a Bag of attributes. This Bag may contain none, one, or many values for a particular attribute.[24] The Bag is an unordered collection and it also may contain duplicate values for a particular attribute.[25] The function that acts on the Bag may accept these duplicate values or it may recognize duplicate values as an error depending on the implementation. XACML provides the functionality to limit the number of values returned to one. These [type]-one-and-only functions will return a Bag containing one attribute value if only one exists, or an error if there or no values or multiple values.[26]

XACML defines certain standard value types of attributes. These include string, boolean, integer, double, time, and date values. Along with these types, XACML defines operations to be performed on the different types. Examples include

23 Sun Microsystems, Inc. http://sunxacml.sourceforge.net/guide.html

24 Godik & Moses, ed., OASIS eXtensible Access Control Markup Language (XACML) Version 1.0

25 Sun Microsystems, Inc. http://sunxacml.sourceforge.net/guide.html

26 Sun Microsystems, Inc. http://sunxacml.sourceforge.net/guide.html

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

equality and comparison operations; string manipulation; numerical and set functions; and date, time, and duration operations. XACML allows implementers to define other attribute types within their applications and schemas. Implementers also can define custom functions to work on the pre-defined types or on custom types. Functions may be nested in order to build more complex functions.[27]

Attributes are identified by their corresponding SubjectAttributeDesignator, ResourceAttributeDesignator, ActionAttributeDesignator, and EnvironmentAttributeDesignator elements. All use an AttributeValue element, which is used to define the values of a particular attribute. Alternatively, an AttributeSelector element can be used to define where the values for a particular entity reside. This allows an external source to provide the attributes for one of the data type elements.[28] For example, an AttributeSelector could programmatically point to a database in which a user's roles are stored. By passing the user ID, the application could then look up the roles rather than having to pass the roles with each request. This allows the PEP to make request for a particular Subject with no need to define who or what the subject is.

Context

A Context is the canonical representation of a decision request and an authorization decision. Canonical XML refers to XML that is in canonical form; in other words, the XML is formatted by a particular standard. Canonical XML is a syntax designed to interpret the logical structure of XML documents in order to determine if the XML documents are equivalent.[29] Representing the request in canonical form allows applications to more easily parse the data and provides methods for more easily locating specific items of data within the document. If an application makes a request of the PDP that is not in canonical form, then a Context Handler can be defined to convert the requests in its native format to the XACML canonical form and to convert the authorization decisions in the XACML canonical form to the native format.[30]

Summary

Though XACML is a relatively new standard, adoption by corporate users could potentially simplify security administration across the enterprise. Security administrators can use one language to define access control for a variety of applications and systems. Additionally, once tools are available for building and administering XACML-compliant applications, developers will be able to more

27 Godik & Moses, ed., OASIS eXtensible Access Control Markup Language (XACML) Version 1.0

28 Godik & Moses, ed., OASIS eXtensible Access Control Markup Language (XACML) Version 1.0

29 Wilde's WWW Online Glossary. http://wildesweb.com/glossary/canonicalxml

30 Godik & Moses, ed., OASIS eXtensible Access Control Markup Language (XACML) Version 1.0

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

easily add authentication-level security to their applications and corporations can reduce the costs of implementing security by using a standard access control mechanism across multiple platforms and multiple applications. Having a standard also will allow developers to build products that can rely on a variety of vendors' tools. With a common access control policy mechanism, companies are more likely to be able to develop systems that can interoperate; increasing users' confidence and satisfaction with the system and also making users more productive.

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

References

1. Anderson, Birbeck, et al., *Professional XML*, Wrox Press, 2000
2. Godik, Simon and Moses, Tim, ed., *OASIS* eXtensible Access Control Markup Language (XACML) Version 1.0 3 OASIS Standard, 18 February 2003
3. Harold, Elliote Rusty and Means, W. Scott, *XML in a Nutshell: A Desktop Quick Reference*, O'Reilly and Associates, Inc January 2001
4. McCarthy, Vance. *XACML -- A No-Nonsense Developer's Guide*. Open Enterprise Trends. March 3, 2003. http://www.oetrends.com/cgi-bin/page_display.cgi?180
5. McLaughlin, Brett, *Java and XML*, O'Reilly and Associates, June 2000
6. *OASIS – Who We Are – Mission*. OASIS. June 2003. http://www.oasis-open.org/who/
7. *Organization for the Advancement of Structured Information Standards (OASIS).* Web Services and Service-Oriented Architectures. June 2003. http://www.service-architecture.com/web-services/articles/organization_for_the_advancement_of_structured_information_standards_oasis.html
8. Wilde's WWW Online Glossary. June 2003. XACML Access Control Markup Language. Web News and Product Reviews. June 2003. http://www.webdesk.com/xacml/
9. XACML ratified as OASIS Open Standard. WebServices.org. http://www.webservices.org/index.php/article/articleview/909/1/3/
10. http://wildesweb.com/glossary/canonicalxml
11. XACML Implementation Programmer's guide. Sun Microsystems, Inc. June 2003. http://sunxacml.sourceforge.net/guide.html

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.