Global Information Assurance Certification Paper

An Introduction to XACML

Bob Thigpen
July 12, 2003

Table of Contents

ABSTRACT

XACML eXtensible Access Control MarkupLanguage (pronounced “exact-mil”) is an OASIS standard using XML as a common language to express security policy[1]. It is one of a family of specifications that OASIS has created to support “web services”. Some of the other related family members include XML, XSLT, XML namespaces, XML schema, Xpath, Xpointer, SOAP and SAML. Information on these (and more) standards are available at the OASIS website. Familiarity with the XML family concepts facilitates review of the coding and pseudo-code examples provided in the standards specification. This paper provides an overview of the XACML standard, building blocks upon which it is formed; and provides a review of the implementation status of XACML products available at the time of this writing.
This is a developing standard and the information on this and related areas is rapidly evolving. The writer’s intent is to provide an introduction only to this potentially important web security standard.

BACKGROUND

Experience with metadirectory approaches to enterprise security policy in a large multi-platform, multi-operating system environment, such as is found in many state governments and in larger private business organizations was motivation to explore XACML. Metadirectories have a central engine (sometimes XML based) that provide software”connectors” to the operating systems and applications running out at the far ends. They often use X500 or LDAP repositories to store user credentials in a common place and format, but require the connectors to deliver these credentials to the applications and operating system security systems out at the end points for user authentication to those systems. From a centralized or corporate administrators perspective, managing security that is administered on a large number of different applications and operating system platforms is difficult, at best. The metadirectory approach does not typically prohibit local security administration on all of the platforms.
The connectors have to be upgraded every time there is a change in the security features of the operating systems or applications. This implies that there is enough knowledge of all of these systems to upgrade complex system interfaces with non-trivial coding requirements. If this isn’t the case, delays in upgrading the connectors can cause frustration with the centrally administered system and also can lead to high maintenance costs. With so many factors to consider, it can be very difficult to implement a security policy that has clearly defined rules for users to follow and management to understand.[2]
XACML promises to provide a common language and common primitives for managing all of the diverse operating systems and applications found in large

decentralized organizations and this is an area where it can provide a unified, but not simple approach to security administration. Another perhaps more lucrative market exists in the world of web services where a transaction can involve many entities such as banks, credit card companies, sellers, buyers and more for complex web transactions.

XACML is written in XML and requires a familiarity with several XML standards including SAML, XML namespaces, Xpath , Xpointer and XSLT and others. There are several web-based tutorials that cover this information. O'Reilly's web site[3]
cover much of this material and provides a search function for the site. Peter Flynn's XML FAQ [4] site is another good introductory site with pointers to many XML web based resources. The World Wide Web Consortium[5] (W3C) and OASIS[6] are definitive sources for a detailed, standards based description (also known as normative) of all of the XML family of tools and languages. Because of the need for the W3C and OASIS to be exact in the definitions, readers may prefer to begin with the aforementioned introductory sources.

THE POWER OF XML

XML namespaces can offer a security template that can be located on a local machine or found out on the web. Referencing another namespace ensures object name uniqueness because the reference becomes part of the object name. Namespaces are composed of XML schemas that provide several object-oriented functions like inheritance, where an object of type security-policy (or any other object) can be created in the image of an existing policy (object). A set or more technically, a tree of elements and their attributes are known as a schema. The policy being created inherits all of the attributes of the existing policy. This can be accomplished by declaring your image policy elements and attributes of type (existing) security-policy. [7]
XML elements are typed. In addition to standard types like integers and text, one can make up types – like security-policy. Elements can have attributes like time, from/to times, definitions of minimum and maximum occurrences, be one or more than one of a member of a set or make up your own attribute. Mathematical and set operators are available so that the number of variables that can be examined is limited only by the practicality of having readable code. It is possible to define types globally or locally and to define new elements, types and/or attributes to meet requirements that are not defined in the imported namespace schema. This is known as extending the schema.

Xlink is much more powerful than an HTML hyperlink because it can be an attribute within a XML schema document that forms a bi-directional link and allows the traversal of an XML tree structure. Since it is an attribute, it can change to reflect the addition or deletion of the contents of an XML document.[8] Xpath allows a more refined path within a XML document so that one can select a specific element or attribute within the document to display or retrieve.

XML standards are evolving and they are extensive. The aforementioned examples cover only a small portion of the current standards. They are illustrated to give the reader an introductory appreciation for the power and flexibility of XML as it relates to writing security policies with XACML. XACML policy writers can import XML schema to provide a template for their own policies. They can use the language to quickly navigate through large XML documents and can define document content constraints that can save on the need for software parsing and type checking. While the body of XML standards is large and growing, readers with interest in writing XACML policies will find their time well spent in becoming familiar with the XML standards toolset.

XACML REQUIREMENTS

The OASIS XACML standard was published in February 2003 and the following security policy requirements that XACML addresses are derived from that 132-page document.[9] Keywords are defined along with the requirements.

1. "To provide a method for combining individual **rules** and **policies** into a single **policy set** that applies to a **decision request."**
A set of rules creates a policy and policies can be combined to make a policy set. An example of a policy set might be the combination of security policies from a police force and the criminal database owner such as a State or the FBI. A decision request typically will translate into allowing or denying access to a resource, such as a database or access to a network.

2. "To provide a method for flexible definition of the procedure by which **rules** and **policies** are combined."
There are a number of built in algorithms that support these goals. These include rule combining and policy combining algorithms that deny overrides (if any rule or policy evaluates to deny, access is denied), permit overrides (if any rule or policy evaluates to permit, access is permitted), first applicable (evaluate only first rule or policy that matches) and only one-applicable (applies to policies only). Users can create their own combining algorithms if the built-ins are insufficient.

3. " To provide a method for dealing with multiple **subjects** acting in different capacities"
The standard provides the example of a high level financial transaction where multiple subjects must provide approval. A subject is an actor that has attributes that can be evaluated for truth. An example would be user name jane_bank_executive has attributes (privileges in this example) that allow her to approve a loan only if the loan approval form already has the approval of user

john_bank_loanapprover.

4. “To provide a method for basing an **authorization decision** on **attributes** of the **subject** and **resource**”
XACML provides special named elements that point to attribute values by using an URN (Uniform Resource Name)[10] or that contain an Xpath expression in the request context that maps to a particular subject attribute value by its location. Put another way, the request XML schema maps to the subject XML schema and Xpath is used to find the corresponding value(s).
Resources (data, network access, etc.) also can have special named elements using the same URN or Xpath approaches as are used for subjects. An example of where this might be used is with access to arrest records where a juvenile attribute would restrict the set of users who would be allowed access.

5. “To provide a method for dealing with multi-valued **attributes**”
Higher order functions are provided for attributes that might contain multiple values such as may be found in LDAP and SAML. The functions are a set of list operators formally described using the Haskell functional language in the XACML specification.

6. “To provide a method for basing an authorization decision on the contents of an information **resource**”
An example policy would be where a patient is allowed to read records in a database if they belong to her. This is addressed if the database is an XML document using the URN or Xpath approaches in #4 above. If the database cannot be represented as an XML document there is a special attribute called “subject-category” that can be defined and then compared to the database value(s) – names and possibly date of birth in this instance.

7. “To provide a set of logical and mathematical operators on **attributes** of the **subject**, **resource** and **environment**”
The authorization decision can be based upon a computation or mathematical operation upon the attribute values of the requestor, the resource being requested and environmental variables like time of day. An example would be applying for a travel cash advance. Credentials might be checked to ensure one is an active employee and the database might be checked to ensure that there are no overdue balances from past advances prior to authorizing the check for the requested amount. Some organizations might permit transactions over a certain monetary limit to be conducted only during normal business hours (an example of an environmental variable).

8. “To provide a method of for handling a distributed set of **policy** components, while abstracting the method for locating, retrieving and authenticating **policy** components”
Policies can be distributed across an organization or multiple organizations allowing component updates to occur as needed. Policy elements can be

attached to a resource such as a database or can be maintained in a distant location(s) accessible over a network.

9. “To provide a method for rapidly identifying the **policy** that applies to a given action, based upon the values of **attributes** of the **subjects**, **resource** and **action**”
One approach of supporting this functionality is to store subject, resource and action attributes as part of a database and to read those database attributes as part of a decision request. Another approach is to evaluate all available policies or policy sets for matching attributes of subject, resource and action.
XACML requires that all attributes match for authorization to occur and introduces a special target element that defines the data structure of the attributes to be evaluated by the policy or policy set.

10. “To provide an abstraction-layer that insulates the policy-writer from the details of the application environment”
All policies are built with syntax defined in XML schema. For authorization entities that do not speak XML there is an intermediate step to convert requests and responses to the format specified in the XACML schema. In this way, the policy reader only needs to know the XML schema syntax, not all of the variations of the authorization entities native syntax. For authorization entities that support XML (those that support SAML[11],[12])
XSLT can be used to transform between native and the XACML schema or context.

11. “To provide a method for specifying a set of actions that must be performed in conjunction with policy enforcement”
XACML includes the concept of an obligation that is passed to the authorization entity called a PEP. If a policy or policy set evaluates to permit and the policy or policy set matches the value of the XACML FulfillOn attribute of the obligation, the PEP is responsible for fulfilling all of the obligations that it understands.
XACML version 1.0 does not define obligations. It is a user responsibility to match obligations that a PEP understands. PEP and other actors in the XACML domain are covered in the following section.

XACML ACTORS

This information is excerpted from the XACML standard document. See page 19 of the PDF version of the OASIS XACML 1.0 standard for a flow diagram showing the interaction between the actors.
The functional building blocks of the XACML language, called “actors” in the standard specification, follow:

PAP – Policy Administration Point is the interface where policies are referenced. While the standard doesn’t discuss a user interface for writing policies, the PAP might be a logical place to include this interface. Policies can be widely

disbursed in a number of corporate servers or centrally located on the server that houses the PAP.
When several policies pertain to the same resource (target) it is necessary to join them all into a policy set. Rules are the building blocks of a policy. The PAP matches resources, subjects and actions (an operation on a resource) that apply to authorization decisions to be made upon a target.

PDP – Policy Decision Point is the part of the system that evaluates policies and provides authorization decisions. It receives requests and provides replies in a XML context specific format with a XML schema defined for requests and another for replies (decisions) and communicates with the CONTEXT HANDLER.

PEP – Policy Enforcement Point is the part of the system that receives the access request in native format and sends it to the CONTEXT HANDLER to be transformed into the XML schema request format sent to the PDP. A PEP is a system entity that provides access control such as /etc/password on Unix systems. There is an implication for the need of a software shim such as PAM (pluggable authentication module used with Unix systems to redirect the authorization requests to another source such as a lightweight directory authentication program (LDAP) database or for the OS or NOS provider to themselves provide a XACML software shim.
The CONTEXT HANDLER receives the decision from the PDP and translates it from the XACML decisions schema format (context) into a native response understood by the PEP.

The CONTEXT HANDLER sends the request to the PDP along with the target identification. The CONTEXT HANDLER also obtains any attributes that the PDP needs to make the authorization decision(s).

The PIP – policy information point receives requests for attribute values from the context handler and returns subject, resource and environment attribute values back to the CONTEXT HANDLER.

Requests that enter the XACML actor domain through the PEP may be in native form or may already be in XML format such as with XACML's sister specification for Security Assertion Markup Language (SAML), which is an OASIS working draft.[13]
Responses that exit the XACML actor domain (back to the PEP) get converted back to the native format.
XACML schemas exist for request and response and all requests get converted to the request schema upon entering the XACML actor domain. Responses get converted from
XACML response schema to native format before being sent to the PEP.

XACML and SAML

The SAML specification defines XML schema for making security assertions of authentication, attribute and authorization decision types. SAML defines a request-response protocol structure schema and currently defines a binding to SOAP over HTTP.
SOAP is an XML envelope that describes message type, message processing instructions and remote procedures[14] and is a W3C standard. SAML authorities can sign assertions with a XML digital signature and can use a variety of authentication and message encryption methods. An authentication statement can be used between a SAML authenticating agent and a PDP or PEP.[15]
The transformation between the SAML and XML schemas can be accomplished with XSLT (XML Stylesheet Language Transform) as many of the attributes map or are very closely related. SAML capable applications may be able to converse directly with a XACML PDP without having to transform attributes into the XACML schema form known as XACML context. The OASIS standard indicates that "The PDP may be impkemented such that it uses a processed form of the XML files."

SECURITY CONSIDERATIONS

The specification provides a number of recommendations to implementers. Some of the more obvious precautions involve protection of the PAP. Distribution of responsibilities
is allowed so there can be multiple PAPs. Access to the security policies within or accessible to the PAP(s) should be protected by access controls. Depending upon the environment it may be a good idea to partially or totally encrypt the policies. If the required access attributes are readable in the security policy, an adversary then has valuable information with which to plan an attack. Policies need to be checked to ensure that they have not been erased or changed.

Through DNS attack methods and/or internet address spoofing portions of policies or policies that are part of a policy set can be changed to give an attacker access to protected resources. It may even be possible to insert access for an attacker to masquerade as http://www.w3.org/2001/XMLschema-instance, the address for the policy schema instance. An unwary policy developer could be writing policy on top of a compromised template. To protect against these types of attacks and to keep security tight between all XACML actor points (PAP, PEP, PDP, PIP) the OASIS XACML DSig Profile. The proper use of the DSig Profile can provide authentication and integrity protection for XACML schemas[16]
A less obvious security consideration is a policy that returns indeterminate or not applicable results inste[17]ad of deny or allow. This situation does not exist when the PEP intercepts the request at the point of execution, but is frequently an issue if the PEP is implemented as a proxy or filter by different developers than the resource server. A common example of this is a web server that permit a

variety of syntaxes to be treated equivalently. The "%" can be used to represent hex values and multiple character sets may be permitted.

XACML IMPLEMENTATIONS

Sun Microsystems XACML implementation is a set of JAVA ™ classes that read, write, and process the XACML language. This implementation is "Open Source" and Sun has an open invitation policy to developers who want to contribute.[18]

Jiffy XACML is available for Linux and Windows users in binary only. It supports a command line toolset which will check for syntax and type check a policy or request , execute a request against a policy and output the response to a console and execute a test of a policy by providing a request and expected response[19]

XACML COMING ATTRACTIONS

One of the contributors to the XML specification who also was on the development team that implemented SUN's XACML , made the following observation from a policy writer and system programmer's perspective – "Working with XACML is really hard. The tradeoff to building flexible, expressive generic languages is that they can be hard to work with, and XACML is definitely that." [20] His comment was made in the context of discussions of extending the language versus providing abstraction tools to make it easier for users. In another thread of the discussion this contributor wrote "I'm already hearing that they (the average coder, sysadmin, or IT person) need a lot of handholding just understanding the ideas, regardless of whether they are looking at the syntax."[21]
There is some discussion on whether non-programmers will need to actually look at the rules that are stored in a PAP. They most certainly will. Security professionals will need to be able to understand the code to ensure it satisfies management's security objectives and auditors will need the same level of understanding.
The many XML components that are the building blocks of XACML are not trivial and require considerable time investment. XACML is a very powerful security policy language and the investment is worthwhile. When compared to having to learn the security syntax in a multi-platform environment with many operating system and application security syntaxes, XACML may be less complex and is a more exacting method with which to define enterprise-wide security policies.

To make XACML a commercial success, the development of a graphical logic flow mechanism and a translator that could provide a written translation of the policy or policy set would be very helpful. Work appears to be underway to map an XACML attribute model to LDAP. Interfaces to other PEPs such as popular

© SANS Institute 2000 - 2005 Author retains full rights.

directories, databases, network devices, and operating system security mechanisms would facilitate use of XACML.

THE WEB OF STANDARDS

There are many pieces to the web security protocol stack, some of which have been discussed in this paper and others that are briefly touched upon in this section .

XML Digital Signature (previously referenced) is a W3C activity that defines signatures for soap envelopes, signed attributes such as timestamps and network (URI) data.

XML Encryption is a W3C activity that deals with encrypting all or portions of an XML document.

XML Key Management is a W3C working group that specifies issues of key cryptography

OASIS Security Services also known as SAML (previously referenced) that is a language for security assertions and that may provide SSO (single sign-on) applications.

OASIS Digital Signature Services technical committee that is working on definitions for
obtaining signature services when local facilities don't exist for providing these services

OASIS Web Services Security technical committee is building on WS-Security (IBM and Microsoft) which defines how to sign a SOAP message. [22]

Canonical XML is a W3C standard and defines a set of algorithms for translating XML documents into a single precise form for the purposes of testing whether the information content of a document has changed. Earlier implementations of XML allowed multiple syntaxes. Canonical XML can be used to ensure that signed XML documents have not been changed by running the documents through the canonical process at sender and receiver points to ensure the same message digest value.

NEXT STEPS

The next steps for this writer will be to install an XACML implementation and begin to use it to write XACML policies. It is a rich language that offers users an opportunity to sharpen security skills and become more familiar with the

large and growing body of XML based services, protocols and tools.
It will be interesting to see how this proposed standard matures and to see the degree to which it is adopted by the larger players in the web marketplace.
There are other standards that may overlap or eventually compete with XACML and it will be important for security administrators to stay current with W3C and OASIS security developments and those of other groups such as WS-Interoperability[23]
While the XML world is expanding, many (if not most) web applications today also employ or are augmented by other language such as CGI scripts. And then there are all of the web browsers that interact with the application. I've got to get back to work.

---

[1] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml **(XACML 1.0 Specification Set (18 Feb. 2003): *OASIS Standard* as of 6 Feb. 2003)**
[2] http://www.webservices.org/index.php/article/articleview/909/1/3 (Web Services.Org "XACML Ratified as an OASIS standard")
[3] http://webservices.xml.com/pub/a/ws/2003/01/15/ends.html (O'Reilly Web Services, "Securing Web Services", Rich Salz, January 15, 2003)
[4] http://www.ucc.ie/xml/faq.html#schemas (The XML FAQ Version 3.0, Peter Flynn, January 1, 2002
[5] http://www.w3.org/ (World Wide Web Consortium)
[6]http://www.oasis-open.org/ (OASIS)
[7] http://msdn.microsoft.com/msdnmag/issues/01/07/xml/default.aspx (Microsoft Developer Network, "Understanding XML Namespaces", Aaron Skonnard, July 2001)
[8] http://xml.coverpages.org/xlinkMaler980402.html (XML CoverPages.Org, "Xlink and Xpointer Overview", Eve Maler, April 2. 1998)
[9] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml (OASIS XACML 1.0 Specification Set, February 4, 2003)
[10] http://www.ietf.org/rfc/rfc3406.txt (IETF Uniform Resource Names (URN) Namespace Definition Mechanisms, October 2002)
[11] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security (OASIS SAML Version 1.1 specification, July 1, 2003)
[12] http://www.capeclear.com/support/manuals/help/security/overview_of_saml.htm (Cape Clear Support Manuals, Overview of SAML)
[13] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security (OASIS Security Services Technical Committee)
[14] http://www.w3.org/TR/SOAP/ (W3C Simple Object Access Protocol (SOAP) version 1.1, May 8, 2000)
[15] http://www-unix.gridforum.org/mail_archive/authz-wg/msg00021.html (OASIS XACML list server, "Summary of XML Security Languages", Mary Thompson, February 12, 2003)
[16] http://xml.coverpages.org/ni2003-03-28-b.html (OASIS Cover Pages, "XACML XML DSig Profile Supports Authentication of XACML Instances")
[17] http://lists.oasis-open.org/archives/xacml/200208/msg00109.html (OASIS XACML List Server, "XACML Threat Model". Don Flinn, August, 22, 2002)
[18] http://java.sun.com/features/2003/06/xacml.html (Java.Sun.Com, "XACML: A New Standard Protects Content in Enterprise Data Exchange", June 24, 2003)

[19] http://www.jiffysoftware.com/xacml/readme.html#overview (Jiffy Software, "jiffy XACML Alpha1 ReadMe")
[20] http://lists.oasis-open.org/archives/xacml/200304/msg00046.html (OASIS XACML list server, "Complexity Going Forward", Seth Proctor, April 22, 2003)
[21] http://lists.oasis-open.org/archives/xacml/200304/msg00049.html (OASIS XACML list server, reply to "Complexity Going Forward", Seth Proctor, April 22, 2003)
[22] http://webservices.xml.com/pub/a/ws/2003/01/15/ends.html http://lists.oasis-open.org/archives/xacml/200304/msg00049.html (O'Reilly WebServices, "Securing Web Services", Rich Salz, January 15, 2003)
[23] http://www.eweek.com/article2/0,3959,7267,00.asp (eWeek, "Web Services Security: A Political Battlefield", Anne Chen, May 27, 2002)