

Department of Veterans Affairs
Veterans Health Administration
Office of Information



Evaluation of XML Technologies as Applied to Access Control

Version 1.0

September 13, 2004

Prepared by:

David Staggs

Chief Systems Engineer
Science Applications International Corporation
Authentication and Authorization Infrastructure Project (AAIP) Lab



Evaluation of XML Technologies as Applied to Access Control

TABLE OF CONTENTS

1. INTRODUCTION.....	1
2. XACML	2
Introduction to XACML	2
The XACML Conceptual Model	3
3. USE OF XACML IN THE LAB	4
Requirements	4
Initial Results	5
XACML Policies in LDAP	6
Tools Supporting XACML	8
4. SUMMARY	10
APPENDIX A – USING XACML.....	1
A-1 Using XACML in the Healthcare Environment	1
A-2 Accessing Patient Records.....	1
A-2.1 Using the Decision Request Context	1
A-2.2 Allowing Patients to Read Their Records.....	3
A-2.3 Allowing Doctors Access to Patient Records	4
A-2.4 Allowing Guardians Access to Records of Minors.....	6

LIST OF FIGURES

Figure 2-1. OASIS XACML Data-Flow Diagram.....	3
Figure 3-1. Screen Shots of Testing Networked XACML Implementation.....	6
Figure 3-2. An XACML Policy Stored in LDAP.....	7
Figure 3-3. Altova Prototype Tool for writing XACML policies.....	8

Evaluation of XML Technologies as Applied to Access Control

LIST OF ACRONYMS

Acronym	Item
AAIP	Authorization Infrastructure Project
API	Application Program Interface
DIT	Directory Information Tree
DOM	Document Object Model
JAAS	Java Authentication and Authorization Service
LDAP	Lightweight Directory Access Protocol
NIST	National Institute of Standards and Technology
OASIS	Organization for the Advancement of Structured Information Standards
PAP	Policy Administration Point
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
RBAC	Role Based Access Control
SAML	Security Assertion Markup Language
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SPML	Service Provisioning Markup Language
SSL	Secure Sockets Layer
URI	Uniform Resource Identifier
XACML	eXtensible Access Control Markup Language
Xlink	XML Linking Language
XML	eXtensible Markup Language

Evaluation of XML Technologies as Applied to Access Control

Revision History

Name	Date	Reason For Changes	Version
David Staggs	7/15/04	Initial Draft	0.1
David Staggs	7/19/04	Initial Release	1.0
Craig Winter	9/02/04	QA Review	1.1

Evaluation of XML Technologies as Applied to Access Control

1. Introduction

This document summarizes efforts by the Authentication and Authorization Infrastructure Project (AAIP) lab to explore the use of emerging XML¹ (eXtensible Markup Language) based technologies for controlling access to protected information resources. Our central focus is on providing solutions to meet the VHA business requirement of implementing role based access control (RBAC). RBAC is an approach to controlling access to protected resources on an information system.² The National Institute of Standards and Technology (NIST) identified RBAC as the predominant model for advanced access control because it reduces the complexity and cost of security administration in large networked applications.³ Consequently, the AAIP lab is reviewing several XML-based technologies that may be useful in implementing RBAC.

Since XML is a relatively new technology, the AAIP has relied on the guidance of industry-recognized standards groups. Several XML standards have been promulgated by the Organization for the Advancement of Structured Information Standards⁴ (OASIS). XACML (eXtensible Access Control Markup Language) is an OASIS standard useful in expressing and evaluating access control decisions on a protected system resource, such as a patient's medical record. The OASIS XACML Technical Committee has provided guidance on using XACML with several XML-based technologies. Specifically, "XACML Profile for Role Based Access Control (RBAC),"⁵ "LDAP (Lightweight Directory Access Protocol) Profile for Distribution of XACML,"⁶ and "Service Provisioning Markup Language (SPML),"⁷ as discussed below.

Although using XML-based access control technologies in a Service Oriented Architecture (SOA) is still in its infancy, the technology shows promise in providing a non-proprietary, industry-wide methodology for RBAC implementation. Based on our initial experiences with XACML in the AAIP laboratory, we believe additional effort in developing XML-based technologies for RBAC is justified.

Evaluation of XML Technologies as Applied to Access Control

2. XACML

Introduction to XACML

XACML is based on the XML language. In a markup language, data are encoded along with information describing the data in the message. Using a markup language permits the exchange of access control information without concern for the data encoding formats or operating systems involved. The details concerning what information is exchanged are defined by an XML schema published by the OASIS XACML technical committee.⁸ The most recent description of XACML is Version 1.0.⁹

Briefly, a request to use a protected resource generates a “decision request” by a Policy Enforcement Point (PEP). The decision request is formatted into an XACML “request statement.” The request statement describes the subject making the request, the resource that is being acted upon, and the action being requested. These three elements are combined to define a request *target*. XACML describes the subject, resource, and action in sufficient detail to provide an adequate context to make an access decision.

The participants in the XACML exchange pass the request statement to a Policy Decision Point (PDP). Policies also have information about subjects, resources, and actions. This information is called the *policy target*. If the PDP finds a policy target that matches the request target, it evaluates the request against the policy. An XACML policy serves as a useful handle to a set of rules. Each rule in the policy contains information about certain subjects, resources, and actions. These three elements are combined to define a *rule target*. The results of multiple matching rules within a policy are combined and returned as a response context. The method of combining the results of each rule into a single response context is specified in the policy.

Additional features of XML can be used by XACML policy writers, such as XPath.¹⁰ As shown in the XACML specification document, XPath can be used to extract values from XML patient records, such as patient’s physician or patient’s legal guardian. These values can be used within the XACML policy to decide whether the requestor is allowed to access the protected resource. Some examples of evaluating XACML policies within the health care domain are included in Appendix A.

An implementation of the XACML standard was recently written by members of the Sun Microsystems Laboratory.¹¹ A copy of the XACML implementation can be downloaded from the open source group SourceForge.net.¹² Operational examples of XACML in this document use version 1.1 of the SourceForge implementation.

Evaluation of XML Technologies as Applied to Access Control

The XACML Conceptual Model

XACML messages are triggered by a request to use a resource.¹³ The request could result from an application participating in a policy controlled environment. For example, a request might be generated by a Java application using JAAS (Java Authentication and Authorization Service)¹⁴ to ensure a requestor has the right to view a patient file. In this example, the application would be considered a PEP.¹⁵

The PEP need not understand XACML; it can rely on another participant in the XACML exchange called a context handler.¹⁶ The context handler does the work of creating the XACML message from properly formed XML statements. The context handler may request information in creating the XACML from the Policy Information Point (PIP).¹⁷ The PIP can be used to access information about the application environment, such as the time of day, which may be required in evaluating policy decisions.¹⁸ The PIP can also provide information on the subject or resource involved in the request.

The fully-formed XACML “decision request” is passed to the PDP¹⁹ for an XACML “authorization decision.” The PDP may request information in collecting the appropriate policies from the Policy Administration Point²⁰ (PAP). The relationships between the XACML participants are shown in Figure 2-1.²¹

As shown in the Figure, the decision to permit or deny access to a protected resource is based on a response to an access decision request. The decision request must be formatted into an XACML context request, if the PEP does not natively output XACML. The context request contains four elements: subject, resource, action, and, optionally, environment. As stated earlier, the first three elements are referred to as a “target.” Each of the elements of a request contains XML attributes. The attribute values associated with the context request elements are used by the PDP to make the access decision that is returned in a XACML response.

The PDP compares the target in the context request to the target in each of its policies. If the attributes describing each element in both targets match, the policy is evaluated. This is actually a simplification since the PDP can also contain policy sets. A policy set contains multiple policies. A policy set is evaluated when the policy set target matches a request target. Policies within the policy set are evaluated individually, and matching policies are combined using a policy-combining algorithm specified in the policy set.

Evaluation of XML Technologies as Applied to Access Control

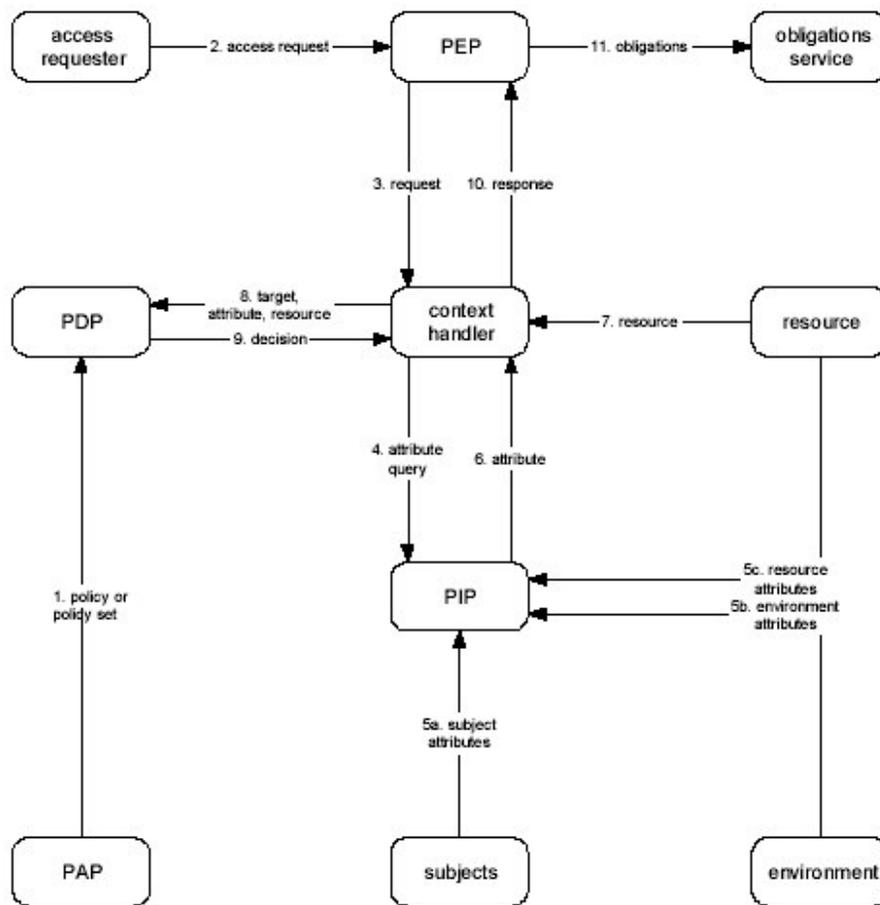


Figure 2-1. OASIS XACML Data-Flow Diagram

The PDP evaluates a policy by evaluating the policy rules therein. The target of each individual rule is separately compared to the context request target. If the rule applies to the request, the rule value is determined by the condition and action encoded in the rule.

Now that the general aspects of the XACML architecture have been introduced, we turn to a more detailed description of its use in the AAIP lab.

3. Use of XACML in the Lab

Requirements

Our goal in this work is to demonstrate the feasibility of using emerging standards by creating a prototype of an RBAC-controlled privilege management infrastructure in a service oriented architecture. That is, the PDP would be available on the network to respond to access control requests as a web service. Applications requiring access control decisions would communicate as a PEP or through an Intermediary PEP (see Figure 2-1). The PDP would use policies written in a manner that supports RBAC.

Evaluation of XML Technologies as Applied to Access Control

These policies would be available to all PDPs in a distributed, fault-tolerant manner. Finally, tools would be deployed to write and maintain XACML policies. These requirements are summarized below:

1. Network-based access control that supports RBAC using XACML
2. Retrieval of policy information from SPML-compliant, centralized servers
3. Commercial tools suitable for creating and maintaining XACML policies

In order to meet the first requirement, we wrote a PEP and PDP using the API (Application Program Interface) provided in the SourceForge XACML implementation. Retrieval of policies in the second requirement was done using openLDAP²² version 2.2.11, which is an “open source”²³ implementation. A review of tools identified in the third requirement is discussed at the end of this document.

Initial Results

In order to provide network-based access control, the SourceForge implementation of XACML was modified as described below. The SourceForge distribution includes a simple PDP executable that is able to read policies from text files and process one XACML request stored in another text file. The SourceForge implementation is written in the Java language and is well organized and documented. We were able to quickly change the PDP code to receive XACML requests from a network connection. Instead of processing one request, the PDP was rewritten to continually process XACML requests as they arrive from the network. Figure 3-1 shows the interaction between the PDP and two PEP process.

The screen labeled “Remote PEP” shows a PEP program connecting to the PDP process over a network connection at marker one. The screen labeled “Networked PDP” shows the PDP receiving the request from the PEP at marker two. The request is evaluated against XACML policies and the result is displayed at the PDP. After processing the XACML request and printing the result, the PDP process waits for another request at marker three.

Evaluation of XML Technologies as Applied to Access Control

Networked PDP

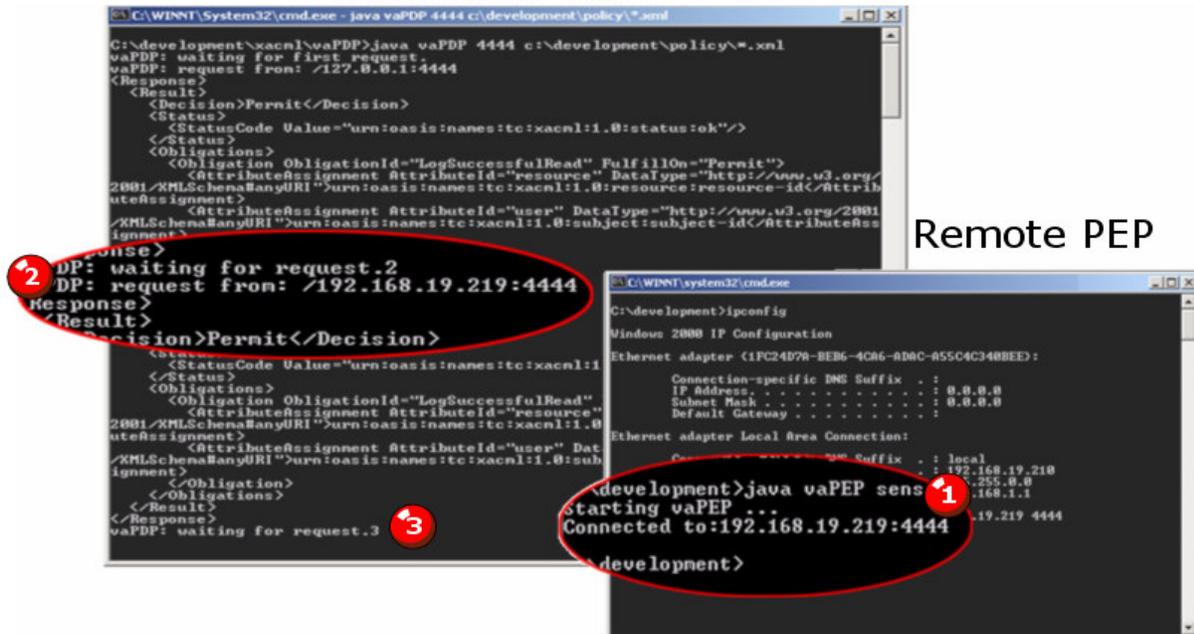


Figure 3-1. Screen Shots of Testing Networked XACML Implementation

Our ability to easily modify the SourceForge XACML implementation is a tribute to the technical quality of the implementation. The code is documented using the Javadoc tool,²⁴ which allows quick access to the classes and methods invoked within the code. There are additional implementation guides to customizing the code by addition of specialized methods. At our invitation, the author provided an outline of how to store and receive XACML policies from an LDAP directory service.

XACML Policies in LDAP

Several benefits flow from storing XACML files in an LDAP. Reliability can be improved by replicating information in an LDAP to several instances on the network. Also, information security can be improved by accessing LDAP over a SSL (Secure Sockets Layer). Unfortunately, there are no examples of XACML policies stored in LDAP presently available from the standards committees.

Members of the OASIS XACML language Technical Committee have suggested the use of LDAP for the distribution of XACML policies.²⁵ Therefore, the AAIP lab has designed a Directory Information Tree (DIT) for storing XACML policies in LDAP. This implementation is consistent with the OASIS working draft “LDAP Profile for Distribution of XACML Policies.” We have tested this implementation using OpenLDAP version 2.2.11.

Evaluation of XML Technologies as Applied to Access Control

Figure 3-2 shows the directory hierarchy we use to store XACML policies. We currently add XACML policies directly to our LDAP using Jxplorer²⁶ version 1.0. Jxplorer is an LDAP utility released under an open source license courtesy of Computer Associates.²⁷ In future work, we anticipate XACML policy information will be added to LDAP using the SPML protocol.

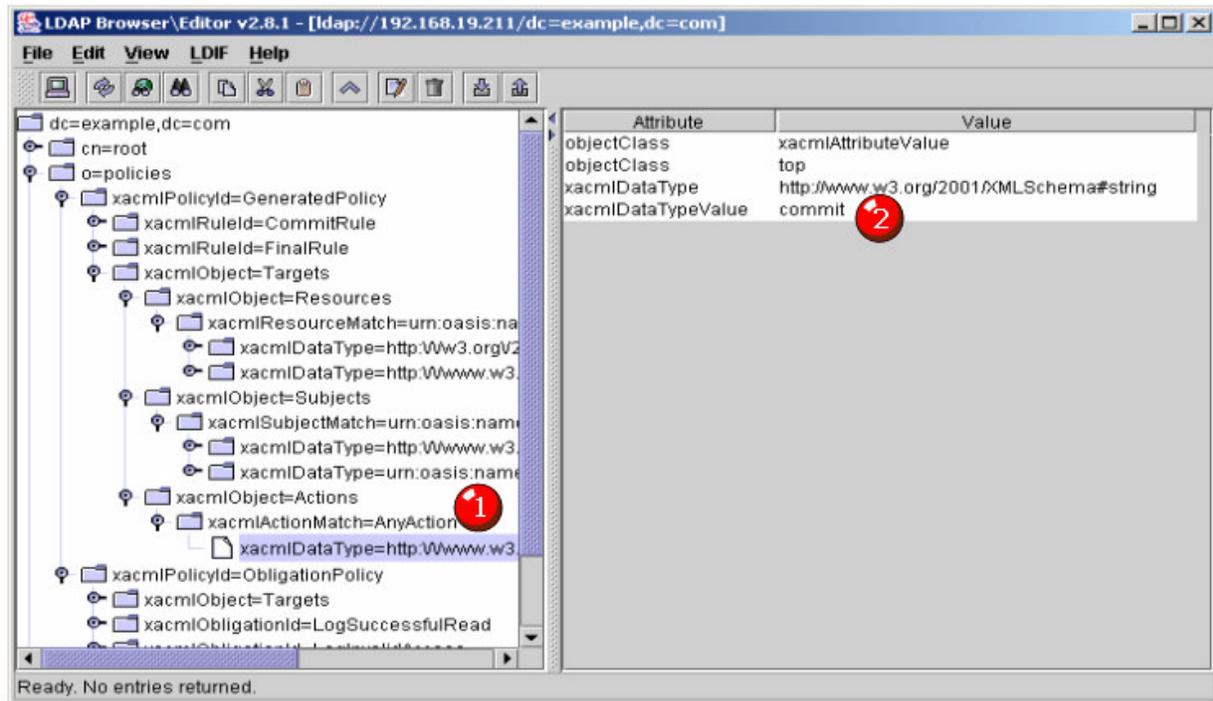


Figure 3-2. An XACML Policy Stored in LDAP

The structure of the LDAP hierarchy was designed to allow efficient searching for policy targets. Native LDAP searching operations should allow us to retrieve just the policy that is relevant to the XACML request being processed. Marker one in Figure 3-2 shows elements of an XACML policy. The corresponding values of each element are shown at marker two. LDAP appears to be appropriate for policy storage, because of the attribute-value pair format of XACML. Also, the existence of LDAP Java interfaces provides flexibility in storing and retrieving XACML policy information. Once the relevant XACML policy is identified it can be moved from the LDAP into a memory structure called the XML Document Object Model (DOM).²⁸ Elements within the DOM can be accessed using the API provided by the SourceForge XACML implementation.

We see no problem with storing XACML policies in LDAP and retrieving them for processing by the PDP process. However, we have not fully implemented and tested the XACML policies that would be required to implement RBAC. As discussed above, XACML RBAC is currently under active study by OASIS. As the technical committee provides guidance on this topic, it is likely that reference implementations of XACML support for RBAC will become available.

Evaluation of XML Technologies as Applied to Access Control

Tools Supporting XACML

Successful use of XML-based technologies in an enterprise requires that computer security personnel be given the proper tools. It seems unlikely that a large enterprise could create and manage XACML policy writing “by hand.” Fortunately, tools are available for writing, managing, and interpreting XML documents. At our invitation, the Altova Corporation provided an example of an XACML policy writer tool.

The prototype XACML policy writer tool is based on Altova Stylevision.²⁹ Stylevision is capable of using an XML schema to anticipate properly formatted user input. For example, if the user selects an rfc822Name option, the program understands that the input format should be “name@company.com” based on the schema definition. Therefore, the tool is able to predict user input from the published XACML schema and offer options in pull-down menus on the screen. Figure 3-3 shows a screen shot of the Altova prototype tool.

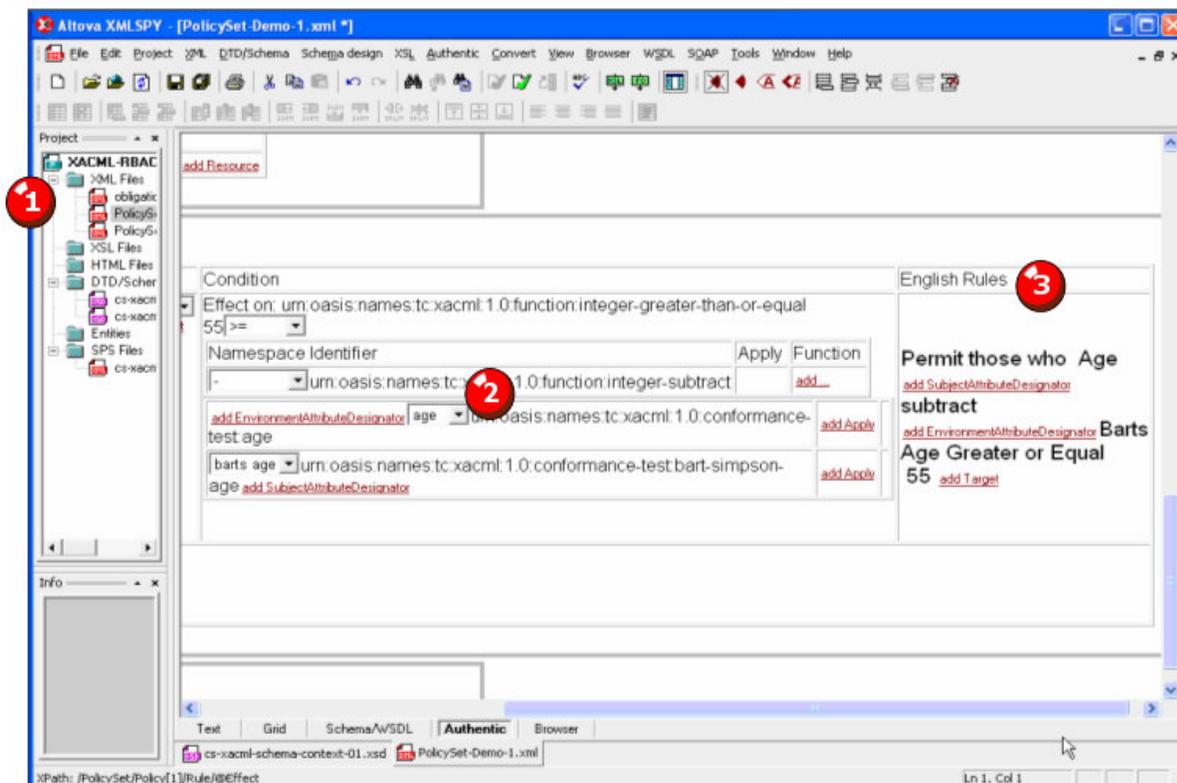


Figure 3-3. Altova Prototype Tool for writing XACML policies

Options for the XACML RBAC tool are organized in a hierarchy at marker one in Figure 3-3. As the policy is entered by the user, pull-down menus are provided as shown at marker two. Marker three displays a natural language representation of the policy as it is entered to allow the user to check their work. The prototype tool can interface with the

Evaluation of XML Technologies as Applied to Access Control

other tools available from Altova providing auditing and review of policies before they are deployed. The Altova prototype demonstrates the ability to incorporate XML support tools in an RBAC deployment as well as the level of interest vendors have in supporting these new tools.

Evaluation of XML Technologies as Applied to Access Control

4. Summary

Our investigation has demonstrated that an XML-based implementation of RBAC in the healthcare sector is technically feasible. In addition, there are several other recent technologies that can be leveraged using XACML. The XML-based Security Assertion Markup Language (SAML) protocol can provide access control information between security domains.³⁰ The SPML offers a system for populating XACML policies in LDAP.³¹ In addition, XACML requests and responses can be transmitted using the Simple Object Access Protocol (SOAP), providing end-to-end security.³²

It appears that this confluence of standards will bring forward the technology to support the adoption of XML-based applications. Therefore, we see the use of XACML as a viable technology in providing an XML-based RBAC implementation for the healthcare sector. Although still in its infancy, XACML and related technologies may provide the means to simplify the complexities inherent in managing individual user access permissions in large organizations.

Appendix A – Using XACML

A-1 *Using XACML in the Healthcare Environment*

The following sections cover examples that concern situations commonly found in the healthcare sector. The details of the XACML exchange in the following examples will be minimized to conserve space. The reader can review the XACML code in detail in the source document.³³

A-2 *Accessing Patient Records*

As stated above, the flexibility inherent in XML-based technologies allows us to provide access control decisions based on a wide variety of input. Access control decisions can even depend on the content of each patient medical record. In the following sections we focus on various examples useful in the healthcare environment.

A-2.1 *Using the Decision Request Context*

In an XACML policy a resource can be specified as a ResourceAttribute of DataType of “anyURI.” A Uniform Resource Identifier (URI) is a compact string of characters for identifying an abstract or physical resource.³⁴ URI addressing can be used within XML through the XML Linking Language (Xlink).³⁵ Xlink provides a way to specify any abstract or physical resource, or parts thereof. For example, an XLink can create a link to a portion of an XML document. This allows, in the context of a healthcare application, access control over specific parts of a patient record.

The XML Linking Language may be used with the XPointer Language.³⁶ The XPointer language is used to address internal structures within an XML document. This section discusses how to use XACML with an XPointer to access parts of a patient record. Subsequent examples will use the concept of an XPointer in scenarios relevant to the healthcare industry.

The Resource element in an Authorization request in the XACML v1.0 OASIS Context Schema is defined as having ResourceContent and Attribute(s). These elements may hold an XML namespace and an attribute that specifies the content being requested. For example, the content could be defined within the fictional “Medi Corp.” namespace and point to a specific instance of a medical reference.

Xpointers are typically associated to AttributeValue elements. Additional AttributeValue elements may hold references to information needed to evaluate the access request. An AttributeValue element in a request context, for example, to pass the rfc822Name (i.e. “patient@medico.com.” The name used in the subsequent OASIS examples shown below is the fictional character “Bart Simpson.”

Evaluation of XML Technologies as Applied to Access Control

The AttributeValue element can also pass the XPointer to Mr. Simpson's Medi Corp. medical record. The XPointer provides a reference that can be examined at the PDP when evaluating the request against the Policy, as shown in the request fragment on line 52 of Example A1. For our analysis in subsequent sections, the AttributeId element name referring to the XPointer is formatted in bold print in several examples below. The AttributeId element names in the decision request can be used by the PDP when evaluating the request against a rule. The element names are not emboldened in the source material.

```
[28] <Resource>
[29] <ResourceContent>
[30] <md:record
[31] xmlns:md="//http:www.medico.com/schemas/record.xsd">
[32] <md:patient>
[33] <md:patientDoB>1992-03-21</md:patientDoB>
[34] </md:patient>
[35] <!-- other fields -->
[36] </md:record>
[37] </ResourceContent>
[38] <Attribute AttributeId=
[39] "urn:oasis:names:tc:xacml:1.0:resource:resource-id"
[40] DataType="http://www.w3.org/2001/XMLSchema#string">
[41] <AttributeValue>
[42] //medico.com/records/bart-simpson.xml#
[43] xmlns(md="//http:www.medico.com/schemas/record.xsd)
[44] xpointer(/md:record/md:patient/md:patientDoB)
[45] </AttributeValue>
[46] </Attribute>
[47] <Attribute AttributeId=
[48] "urn:oasis:names:tc:xacml:1.0:resource:xpath"
[49] DataType="http://www.w3.org/2001/XMLSchema#string">
[50] <AttributeValue>
[51] xmlns(md=http:www.medico.com/schemas/record.xsd)
[52] xpointer(/md:record/md:patient/md:patientDoB)
[53] </AttributeValue>
[54] </Attribute>
[55] <Attribute AttributeId=
[56] "urn:oasis:names:tc:xacml:1.0:resource:target-namespace"
[57] DataType="http://www.w3.org/2001/XMLSchema#string">
[58] <AttributeValue>
[59] http://www.medico.com/schemas/record.xsd
[60] </AttributeValue>
[61] </Attribute>
[62] </Resource>
```

Example A1: Resource Details Passed in an Access Request.

Evaluation of XML Technologies as Applied to Access Control

A-2.2 Allowing Patients to Read Their Records

We can use the information passed in the decision request discussed above to enforce security policies in the healthcare environment. For example, a healthcare provider might have established a security policy that allows patients to read their medical records. One approach to enforcing this security policy with XACML is shown in the rule fragment shown in Example A2. The rule target is written to match the AttributeValue that corresponds to a medical record (record.xsd) to the ResourceAttributeDesignator that corresponds to the resource passed in by a reference in the request context. As shown before in Example A1 at line 56, “target-namespace” has the value of a medical record local to the requestor.

```
[18] <Resources>
[20] <Resource>
[21] <!-- match document target namespace -->
[22] <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[23] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
[24] http://www.medico.com/schemas/record.xsd
[25] </AttributeValue>
[26] <ResourceAttributeDesignator AttributeId=
[27] "urn:oasis:names:tc:xacml:1.0:resource:target-namespace"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
[28] </ResourceMatch>
[29] <!-- match requested xml element -->
[30] <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:xpath-node-match">
[31] <AttributeValue
      DataType="http://www.w3.org/2001/XMLSchema#string">/md:record</AttributeVal
      ue>
[32] <ResourceAttributeDesignator AttributeId=
[33] "urn:oasis:names:tc:xacml:1.0:resource:xpath"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
[34] </ResourceMatch>
[35] </Resource>
[36] </Resources>
```

Example A2: Resource Details Used to Evaluate a Rule.

The first match, starting at line 22, ensures that the medical record is from the Medi Corp. namespace, by comparing it with the values passed in the decision request of Example A2. The second match starting at line 30 compares the record (defined elsewhere) to the instance of a medical record identified by an XPath value passed in the decision request of Example A2.

Evaluation of XML Technologies as Applied to Access Control

We mentioned the Condition element when we first introduced the Rule element. A Condition further defines a rule and evaluates to a true or false. The effect specified by the rule will only take place if the rule's condition evaluates to true. The rule fragment in Example A3 demonstrates use of a Condition.

```
[47] <!-- compare policy number in the document with
[48] policy-number attribute -->
[49] <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[50] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
[51] <!-- policy-number attribute -->
[52] <SubjectAttributeDesignator AttributeId=
[53] "urn:oasis:names:tc:xacml:1.0:examples:attribute:policy-
      number"DataType="http://www.w3.org/2001/XMLSchema#string"/>
[54] </Apply>
[55] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
[56] <!-- policy number in the document -->
[57] <AttributeSelector RequestContextPath=
[58] "//md:record/md:patient/md:patient-
      number/text()"DataType="http://www.w3.org/2001/XMLSchema#string">
[59] </AttributeSelector>
[60] </Apply>
[61] </Condition>
```

Example A3: Ensuring that the Request is by the Patient

The SubjectAttributeDesignator on line 52 and AttributeSelector on line 57 are used to match the identification on the medical record to the identity of the requestor. If the values do not match the Condition will evaluate to false, ensuring the requirements of the local security policy are met.

If the target of the rule matches and the condition evaluates to True, the effect of the rule is passed to the policy. The policy would be written to send the result of "Allow" to the PEP.

A-2.3 Allowing Doctors Access to Patient Records

Our example can be extended to the case where the security policy permits doctors access to their patient's records. The request context, shown in Figure 4, focused on the AttributeValue elements associated with the Resource element, with the intent of discussing XPath values. Similar statements associating Subject attributes to values were not shown to conserve space. The additional statements that were not shown concerning Subject attributes would not have influenced the previous example. That is, the same request context can be used both cases.

Evaluation of XML Technologies as Applied to Access Control

```
[75] <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[76] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
[77] <!-- physician-id subject attribute -->
[78] <SubjectAttributeDesignator AttributeId=
[79] "urn:oasis:names:tc:xacml:1.0:example:
[80] attribute:physician-id"DataType="http://www.w3.org/2001/XMLSchema #string"/>
[81] </Apply>
[82] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
[83] <AttributeSelector RequestContextPath=
[84] "//md:record/md:primaryCarePhysician/md:registrationID/text()"
[85] DataType="http://www.w3.org/2001/XMLSchema#string"/>
[86] </Apply>
[87] </Condition>
```

Example A4: Allowing Access to Patient Records by Physicians

The Subject attributes that were not shown associated several values to the Subject element. For example: subject-id, role, and physician-id. These values, if present, are passed to the PDP by the PEP. The PDP can use these values to enforce a rule permitting doctors access to patient's records. One approach would be to write a rule that matches values associated with a Medi Corp. physician ID to a similar field in the incoming request context. That is, the rule requires the physician ID element to match before considering permitting access to a medical record, an example is shown in Example A4.

As in the previous rule, the rule target is written to ensure that the resource is a Medi Corp. medical record. For this rule to match, the target must additionally match the physician attribute. However, before the rule generates its effect, the Condition shown in Example A4 must evaluate to True.

In order for the Condition to evaluate to True, line 75 requires that the two subsequent string Attribute values must be equal. On line 75 the physician-id attribute is drawn from the request context. Line 84 takes the physician-id from the registrationID field in the medical record. If these values match the condition is True, then the Rule will evaluate to the value of Effect, presumably "Permit."

Evaluation of XML Technologies as Applied to Access Control

A-2.4 Allowing Guardians Access to Records of Minors

Another example from the XACML standard involves permitting access to the medical records of a minor by his or her legal guardian. The security policy requires that the guardian have access to the medical records only if the patient is less than 16 years old. We will focus on the fragment of the rule that enforces this security policy without concern for the details of the rest of the rule.

As in the previous example, the security policy is honored in Example A5 by the Condition element of the rule. For our analysis, we assume that the policy target has properly matched with the request context. We also assume that the rule target is written to match requests that concern a bona fide Medi Corp. medical record. Assuming the rule target has been satisfied, we examine the Condition element from the Rule, duplicated in Example A5.

Evaluation of XML Technologies as Applied to Access Control

```
[48] <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
[49] <!-- compare parent-guardian-id subject attribute with
[50] the value in the document -->
[51] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[52] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
[53] <!-- parent-guardian-id subject attribute -->
[54] <SubjectAttributeDesignator AttributeId=
[55] "urn:oasis:names:tc:xacml:1.0:examples:attribute:
[56] parent-guardian-id"DataType="http://www.w3.org/2001/XMLSchema#string"/>
[57] </Apply>
[58] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
[59] <!-- parent-guardian-id element in the document -->
[60] <AttributeSelector RequestContextPath=
[61] "//md:record/md:parentGuardian/md:parentGuardianId/text()"
[62] DataType="http://www.w3.org/2001/XMLSchema#string">
[63] </AttributeSelector>
[64] </Apply>
[65] </Apply>
[66] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-less-or-equal">
[67] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-only">
[68] <EnvironmentAttributeDesignator AttributeId=
[69] "urn:oasis:names:tc:xacml:1.0:environment:current-date"DataType="http://
    www.w3.org/2001/XMLSchema#date"/>
[70] </Apply>
[71] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration">
[73] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-only">
[74] <!-- patient dob recorded in the document -->
[75] <AttributeSelector RequestContextPath=
[76] "//md:record/md:patient/md:patientDoB/text()"DataType="http://www.w3.org
    /2001/XMLSchema#date">
[77] </AttributeSelector>
[78] </Apply>
[79] <AttributeValue DataType="http://www.w3.org/TR/2002/WD-xquery-operators-
    20020816#yearMonthDuration">
[80] P16Y
[81] </AttributeValue>
[82] </Apply>
[83] </Apply>
[84] </Condition>
```

Example A5: Allowing Guardians Access to Records of a Minor Patient

Evaluation of XML Technologies as Applied to Access Control

The statements in the Condition should now be familiar. Line 48 specifies that the Condition will be determined by the Boolean “and” operation between lines 51-65 and lines 67-83. Lines 52-57 represent the values we expect, lines 58-64 represent the value of the parent-guardian-id element in the request context. If the subject is a parent or guardian, the result is True.

The second part of the Boolean “and” function in Example A5 uses the date-less-or-equal function on line 67. The function has two arguments, shown in lines 67-70 and 71-82. First the Environment element of the request context and retrieves the current date from the Environment at line 69. Lines 71-82 extract the patient’s date of birth from a field on the medical record being requested using the XPath and adds 16 years. If the patient is less than 16 years old, the result is True.

The enclosing statement takes the result of both sets of operations. If both are true, the Condition evaluates to True, and the rule effect (presumably Permit, in this case) is sent to the PDP.

Evaluation of XML Technologies as Applied to Access Control

-
- 1 <http://www.xml.org/>
 - 2 Role-Based Access Control (RBAC) Role Engineering Process, March 24, 2004.
 - 3 <http://csrc.nist.gov/rbac/>
 - 4 <http://www.oasis-open.org/home/index.php>
 - 5 <http://docs.oasis-open.org/xacml/cd-xacml-rbac-profile-01.pdf>
 - 6 <http://lists.oasis-open.org/archives/xacml/200310/bin00000.bin>
 - 7 <http://www.oasis-open.org/committees/download.php/4137/os-pstc-spml-core-1.0.pdf>
 - 8 <http://www.oasis-open.org/specs/index.php#xacmlv1.0>
 - 9 <http://www.oasis-open.org/committees/download.php/2406/oasis-xacml-1.0.pdf>
 - 10 <http://www.w3.org/TR/xpath20/>
 - 11 <http://research.sun.com/projects/xacml/>
 - 12 <http://sunxacml.sourceforge.net>
 - 13 Resource is defined as Data, service or system component.
 - 14 <http://java.sun.com/products/jaas/>
 - 15 Policy enforcement point (PEP) is defined as the system entity that performs access control, by making decision requests and enforcing authorization decisions.
 - 16 Context handler is defined as the system entity that converts decision requests in the native request format to the XACML canonical form and converts authorization decisions in the XACML canonical form to the native response format.
 - 17 The policy information point is defined as the system entity that acts as a source of attribute values.
 - 18 Permitted environment attributes are listed in Section B.8 of the XACML specification.
 - 19 The policy decision point is defined as the system entity that evaluates applicable policy and renders an authorization decision.
 - 20 Policy administration point is defined as the system entity that creates a policy or policy set.
 - 21 From “eXtensible Access Control Markup Language (XACML) Version 1.0,” that can be found at:
<http://www.oasis-open.org/committees/download.php/2406/oasis-xacml-1.0.pdf>
 - 22 <http://www.openldap.org/>
 - 23 <http://www.opensource.org/>
 - 24 <http://java.sun.com/j2se/javadoc/index.jsp>
 - 25 <http://lists.oasis-open.org/archives/xacml/200310/msg00059.html>
 - 26 <http://www.pegacat.com/jxplorer/>
 - 27 <http://www.pegacat.com/jxplorer/license.html>
 - 28 <http://www.java.sun.com/xml/jaxp/dist/1.1/docs/tutorial/dom/index.html>
 - 29 http://www.altova.com/products_xsl.html
 - 30 http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security
 - 31 http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=provision
 - 32 <http://www.w3.org/TR/soap12-part1/>
 - 33 <http://www.oasis-open.org/committees/download.php/2406/oasis-xacml-1.0.pdf>
 - 34 <http://www.ietf.org/rfc/rfc2396.txt>
 - 35 <http://www.w3.org/TR/2000/PR-xlink-20001220/>
 - 36 <http://www.w3.org/TR/xptr-framework/>