Spring 1-1-2004

Access Control in a Distributed Decentralized Network: An XML Approach to Network Security using XACML and SAML

Paul J. Mazzuca
Dartmouth College

Access Control in a Distributed Decentralized Network: An XML Approach to Network Security using XACML and SAML

Paul Mazzuca
paul.j.mazzuca.04@alum.dartmouth.org
Adviser: Edward Feustel
Dartmouth Computer Science Technical Report TR2004-506
Honors Thesis

Spring 2004

Abstract

The development of eXtensible Distributed Access Control (XDAC) systems is influenced by the transference of data access and storage from the local computer to the network. In this distributed system, access control is determined by independent components which transmit requests and decisions over a network, utilizing XML signing capabilities found in the Security Assertion Markup Language (SAML). All resources in the XDAC system are protected by the first component, a Policy Enforcement Point (PEP), which acts as the main divider between the requesting entity and the requested resource. The PEP grants access to a resource only if the second component, a Policy Decision Point (PDP), returns a permit response after consulting a set of applicable policies based on the requester's attributes, the resource, the action that the requester desires to apply to that resource, and optionally the environment. With Sun's eXtensible Access Control Markup Language (XACML), the XML encoded policies can be combined among multiple nodes across a network using XACML rules and algorithms to formulate a single decision based on an XACML request. In this thesis project, I build a secure and efficient XDAC System based on XACML, implement an extension to the SAML Assertion design by including XACML Attributes and Results, describe in-detail about the many features that a XDAC System should embody, and show how a XDAC System would be effectively used in modern day computing.

Contents

List of Figures

1 Introduction

In December of 1995, an estimated 16 million users were connected to the Internet. Nearly a decade later, as computer technology continues to progress, the number of users has grown to a staggering 757 million, 12 percent of the world population [Int04a]. Computers have become the foundation upon which institutions and businesses build, combining to form various networks with which employees, students, professors, doctors, and people across the globe can come together to share data. Though having the world's information at ones fingertips seems, and often is, a great opportunity for consolidation of knowledge and easy access of data, the fact that not all users necessarily want their information to be accessed by others must be considered. What has evolved with the influx of over 700 million users is a dichotomy of reliance upon the internet for sharing data, and an overwhelming need to make sure that the sharing can be controlled. With businesses and institutions being unable to compete nor survive without having the presence of the Internet, the problem that must be addressed is how to confine accessing information on a network level to only those that are intended to have access?

There are two ways to examine the problem of network security. First the computing model must be flexible, universal, and dynamic. It must be flexible in the sense that the Internet is composed of many users, with many different needs and reasons for access control and security. The components that form the basis of access control must be able to expand with the spectrum of possibilities that encompass what criterion will be used for access control, meaning they should be specified but not limited. The computing model must also be universal, independent of platform and context. Not everyone uses the same operating system, nor is access control limited to individual platforms. And finally, it must be dynamic, evolving with the rate at which the Internet changes. The security needs of today will most certainly not resemble the security needs of the future, and the computing model must be able to change with the expectations of the computing community.

The second way to examine the problem, and the more important, is that the computing model must have the capability of distribution. The internet is not a centralized entity, but rather a collection of decentralized nodes. By conforming to the structure of how modern day computing occurs, network security can be dramatically increased. The division of a single system across nodes of a network enables a division between the enforcement process and the decision process in any access control decision. As a result, nodes in a network can be structured to specialize in decision processes, modularize large systems, consult one another across various mediums, and confer with one another for fair decisions.

The eXtensible Distributed Access Control (XDAC) computing model described above will become essential as networks continue to grow across the globe. In terms of implementations, there are none currently available that encompass the flexibility and distributed nature of the XDAC system. In this thesis, I will describe how an effective and efficient XDAC system can be built using the flexible, universal, dynamic, and distributed capabilities of the eXtensible Access Control Markup Language (XACML), and how this XDAC system can remain secure

in a distributed environment by extending the Secure Assertion Markup Language (SAML) to include XACML.

2 Related Work

In conjunction with the development of the XDAC System, two other projects are currently underway to bring distributed access control to the computing world. Shibboleth and Cardea both seek to divide the authentication/enforcement process from authorization just like the XDAC System, however there are notable distinctions.

2.1 Shibboleth

Shibboleth, a project currently in development by Internet2 and MACE, is an open source access control implementation supporting the sharing of web resources among institutions of higher learning. When a user at a given institution attempts to access a resource from a different institution, attributes about the user are sent to a remote destination, which presumably is affliated with the user in some manner. This destination entity then decides based on these attributes whether or not the user is permitted to use the resource. Throughout this decision process, Shibboleth also aims to allow a user to control what attributes are sent to the remote destination, thereby promoting user privacy [Int02]. Two limitations of the current implementation of Shibboleth is that remote institutions do not have the ability to consult other institutions or other deciding entities, and it does not use a single standardized evaluation/policy language for access control. Both of these deficits are addressed in the XDAC System by providing reference decision nodes and a flexible, powerful access control language, XACML.

2.2 Cardea

As part of NASA's developing Information Power Grid technology, Cardea is a distributed access control system much like the XDAC System that is employed and described in this thesis. Cardea implements a distributed security model with a Policy Decision Point and Policy Enforcement Point as modular components, and communication between these components uses XACML and SAML standards. Cardea's approach to authorization utilizes a SAML Policy Decision Point as a front end hub for SAML requests. To obtain attributes about a requester, the SAML PDP contacts an LDAP directory and/or an attribute authority. The attributes are evaluated by an XACML PDP, and then the decision is relayed to the PEP [Lep03]. Compared to Cardea, the XDAC System distinguishes itself by adopting a common language model for the transmission of attributes among nodes. This model is achieved by extending SAML Assertions to include XACML attributes and XACML Results. An exten-

sion of SAML Attributes allows the XDAC System nodes to efficiently communicate with Attribute Authorities without SAML Attribute conversion drawbacks. And, the extension of XACML Results enables a PDP to send all necessary XACML Response information to a PEP without following Cardea's method of using a SAMLAuthorizationDecisionStatement which has compatibility issues with XACML Results, and must then deal once again with conversion drawbacks. Unlike Cardea, the XDAC System also employs PDP to reference node communication, allowing for more accurate request evaluation and a more flexible distribution model. Despite these differences, Cardea has a promising future by taking advantage of both SAML and XACML.

3 XML-Language Components

The XML encoded languages, XACML and SAML, are the foundation of the XDAC System. Together these languages facilitate the access control process in a distributed environment. XACML provides the evaluation context and policy archtiecture whereas SAML ensures non-repudiation and effective attribute tranmission across nodes. In the following sections, the details of these languages will be discussed along with how they contribute to the XDAC Model.

3.1 XACML

3.1.1 Description

XACML was made a standard by OASIS in February of 2003. The standard specifies a request/response language and a policy language that are both encoded in XML. A request contains attribute information that will be needed for a given access control decision to be determined by a PDP. This attibute information includes a subset of subject attributes, an action attribute, a resource attribute, and an optional set of environmental attributes. A request, containing these attributes, is evaluated against a set of policies.

The policy language specifies the access control requirements that a request must demonstrate in order for a user to be granted access. A policy is composed of two sections: a target and a set of rules. The target describes the requesting parties for which the policy is intended, and the set of rules stipulates the required attributes of a requester. Because multiple policies and rules might be applied to any given evaluation, XACML rule combining algorithms consolidate a set of decisions into a into a single decision. For example, a deny-overrides algorithm states that if any rule or policy evaluates to deny, then the ultimate decision will be deny regardless of the evaluations of any other rules or policies. After a request has been evaluated against the applicable set of policies, an XACML response is returned. The response conveys an access control decision along with any obligations by which the requester must abide when using the resource. There are four types of responses: permit, deny, indeterminate, and not

applicable. The permit response demonstrates that access has been granted, while the three remaining responses deny the requester the resource. Not applicable specifies that the attribute information contained in the request does not match any policy. Indeterminate specifies that a decision could not be made due to an error in the evaluation process [Pro03].

When a user desires to take some action on a resource, the query is received by the the protector of that resource known as the Policy Enforcement Point (PEP). The PEP has the difficult task of access control, having to ensure that no requesting entity is granted access to a resource to which it is not permitted. To achieve this, the PEP relies on a Policy Decision Point (PDP) for the evaluation of a requester's information. In this scheme, the PEP authenticates the user and formulates an XACML Request based on the attributes of that user, the desired action, the resource, and properties of the environment at the time of the user query. The request is then sent to the modular PDP, that may or may not be located in the same local network. The PDP evaluates the request against the set of XACML policies which it maintains and then sends a response back to the PEP [Pro03].

3.1.2 Advantages

The design of XACML offers many advantages to the XDAC Model. First, XACML is a standard. Large communities of experts from various backgrounds have reviewed it and approved of it. The widely accepted nature of being a standard lends a universal and steadfast aspect to the language and any model that implements it [Pro03].

Second, it is generic. Because XACML is written in Java and encoded in XML, it can be used in any environment regardless of the context, making mobility among nodes in a network possible and efficient. A Linux server can run a PEP application which can contact a PDP application running on a Windows server in a different country. Using a standardized encoding language like XML and the universal capabilities of Java provides different sects of the computing world with a similar foundation, maximizing communication across different mediums. Rather than each sect having a different encoding language for storing policies and different evaluation functionality, XML and Java lend a common ground for minimizing error across nodes [Pro03].

Third, XACML is powerful and dynamic. It supports various data types, functions, and rules, with the capacity of extension. Because XACML is written in the widely deployed language of XML, it can easily be combined with other XML languages for building new implementations [Pro03].

And fourth, it is distributed. Rule and policy combining algorithms can combine multiple evaluations into a single decision. PDPs existing in different locations can utilize these algorithms as a means of ensuring a valid decision across a network of deciding nodes. Furthermore, PDPs can request policies from other PDPs on different nodes. This enables any single PDP to have access to a very large store of policies across a network. On a distributed level, this becomes highly advantageous, because it consolidates information from multiple sources. More sources of information provide for effective validation, evaluation, and decision making.

3.2 SAML

At the beginning of 2003, the OASIS group approved the Security Assertion Markup Language (SAML) specification [wo04]. Comparable to Microsoft's Passport technology, SAML was designed for Single Sign On (SSO) capabilities. In SSO, once a user has authenticated his/herself with an authority, the authority asserts certain attributes about that user or a set of permitted actions that the user can take. These assertions can later be presented to sites as a means of authentication or authorization without a user having to sign on multiple times. Besides SSO, SAML has also been suggested to reinvent the PKI-model, by taking the private key ownership from the employee level to solely the institutional level. Jamie Lewis, CEO of the Burton Group, suggests rather than issuing certificates to every employee, a company might certify itself and then sign SAML assertions on behalf of its employees who might still authenticate by name/password [Ude02].

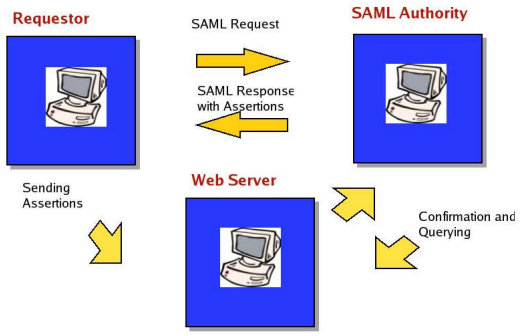

Figure 1: Diagram of Original SAML Communication Model

Much like the concept of a central authority in a network sending certificates about a subject, SAML Authorities send assertions containing attributes that apply to a particular subject or the actions which that subject desires to make. Unlike the certificate authority framework though, SAML Assertions can be generated and sent from any point in the network by anyone [Ros02]. A SAML Authority consequently has only as much authority as the degree to which other nodes in a network trust its certificate. Thus, the onus of this model is placed on whether or not the receiving node trusts the signer or authority of a particular assertion . The advantage though, is a more flexible framework for a distributed system.

As shown in the SSO and key certification examples, SAML is a very effective means of information exchange. The framework when combined with the XML signing capabilities creates a level of nonrepudiation through the endorsement of assertions, and a level of assurance through validation backing by authorities. In the XDAC Model where multiple nodes are contributing to an access decision, there must be some degree of responsibility. Because SAML Assertions are signed packages of user information, nodes can be held accountable for the endorsements they make in any decision process. If a given node N is going to contribute X to a decision, then N signs X, saying that N endorses what is contained in X. This signing

process binds X and N into a single unit, making the transmission of invalid attributes between nodes very unlikely. And, in the case where a node contributes false information, the binding will expose the guilty party. This functionality is essential in the XDAC Model especially as systems grow large and multiple nodes become active in the decision process.

When incorporating SAML into the XDAC System, the goal was to preserve the SAML paradigm so as to allow the XDAC System to be compatible with the assertion functionality described above. In a distributed environment, this meant building a XDAC System to accept external SAML Assertions from SAML Authorities as a means of conveying requester attributes, and to build internal SAML Assertions containing requester information and evaluation results. By including assertions, the XDAC Model can easily be placed into a structured attribute transmission model, giving it an applicable position and sturdy foundation in the computing community.

4 The XDAC Model

[1]The Distributed Access Control model describes the roles of the individual network components that contribute to the decision process. In this model, there are two main components: a Policy Enforcement Point (PEP) and a Policy Decision Point (PDP). The PEP symbolically is a divider or gate between the protected resources and the external environment. The design of the PEP is built on the XDAC requirement that the only way to access a resource protected by a PEP is through the PEP user interface. The PEP provides the XDAC System with the following capabilities: (1) Ensure that no entity is given access to a resource to which it is not permitted (2) Forward to the PDP any assertions or attribute information about the subject that have been presented by the requesting entity (3) Create, sign, and send an assertion containing the entity attributes (subject-id, action and resource) acquired by the PEP, and (4) Have the ability to enforce obligations specified by an XACML Result from a PDP.

The PEP sends the assertions to the PDP, the second component of the XDAC model. The task of the PDP is to decide whether or not a request should be granted. The PDP provides the XDAC system with the following capabilities: (1) Builds a request from a set of validated assertions sent by a PEP (2) Optionally consults other Attribute Authorities for more information about a subject or other PDPs for additional policies (3) Evaluates the request against a trusted set of policies (4) Securely maintains the set of policies and, (5) Sends the response to the requesting PEP concerning the subject.

[1]For detailed diagram of XDAC Model, refer to Appendix A

4.1 PEP

4.1.1 Description and Initialization

The PEP is a server application which exists at a control point, or a point at which a resource can be accessed. For a file system, the PEP would either be an application that acts as an intermediary between access calls by a user and the OS retrieving process, or alternatively in a distributed environment, the PEP would be a server application. For a door access system, the PEP would exist on the computer system which controlled the door locking mechanisms. In each of these examples of a XDAC system, the PEP is the divider between the resources and the external world.

The threaded PEP server application has the capacity of running multiple instances of itself on different ports, each instance corresponding to a separate thread and a separate session. By threading sessions, a single server application can accommodate multiple clients in an efficient manner. The options for a PEP server application are listed and described below, and can be manipulated by flags on executing the application.

- PORT: Server application port (Default: 1111)
- PDP_HOST: Host IP address of the default PDP server (Default: mazook.kiewit.dartmouth.edu)
- PDP_PORT: Port that the remote PDP will run (Default: 4444)
- SYSTEM: The type of PEP, either internal or external (Default: internal)
- KS: Location of the key store (Default: ./KEY_DIR/key_PEP)
- KSPASS: Password to access the key store (Default: password)
- TS_CLIENT: Location of the trust store for clients (Default: ./KEY_DIR/trustClient_PEP)
- TSPASS_CLIENT: Password to access the trust store for clients (Default: password)
- TS_PDP: Location of the trust store for PDPs (Default: ./KEY_DIR/trustPDP_PEP)
- TSPASS_PDP: Password to access the trust store for PDPs (Default: password)
- ALIAS: Alias for accessing the PEP key in the key store (Default: pep)
- VALID_TIME: Time is milliseconds that an assertion is valid after being signed by PEP (Default: 300000 or 5 minutes)
- CACHE_GLOBAL: Activates the global cache (Default: cache off)
- CACHE_LOCAL: Activates the local cache (Default: cache off)
- RREQUEST_OFF: Deactivates the return request option which requires the request to be returned in response (Default: on)

- LOG_OFF: Deactivates logging (Default: on)
- LOG_DIR: Directory where logs will be stored (Default: ./LOG)

4.1.2 Client/PEP Connection

Once the PEP server application has been initialized in accordance with the options above, it listens for client connections. Mutual authentication through SSL is used in this connection process to ensure that the client is connecting to the intended server, and the server is verifying a legitmate client, both necessary elements for security as will be described later. For each client connection request that a PEP server application receives, a new threaded session is made. [2] In this session, the client can optionally send signed assertions containing attributes about his/herself. Because the PEP does not evaluate attributes, these assertions are simply handed off to the PDP along with the base assertion.

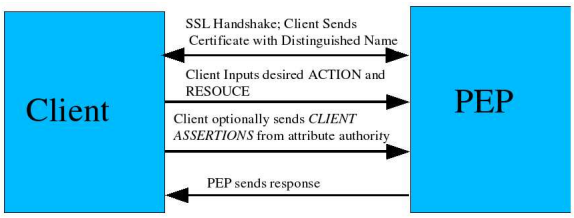

Figure 2: Client PEP Connection

4.1.3 Building the Base Assertion

In this stage, the PEP prompts the user to input an action and a resource, both of which are then converted to XACML attributes. Because the client/PEP connection uses SSL with mutual authentication, the subject-id is acquired from the certificate presented by the client in the SSL authentication process. Thus any client attempting to access a resource using the XDAC system will be bound to the name in the certificate that that client presents, though a client still has the luxury of including a different ID in the client assertions. Like the input action and resource, the subject-id is converted into an XACML attribute. To convey the environment in which the request was made, the PEP also formulates an environment attribute containing the date/time.[3] With the base attributes formulated, the PEP must endorse the validity of the client information. To achieve this, the PEP bundles the base

[2] In this XDAC System the client accesses the PEP through a client application. Additionally, a client would be able to access a PEP through a web site, though for the scope of this project the client application is a simple text-based GUI.

[3] Refer to Appendix section *PEP-local Environment Attribute Formulation* for explanation of creating environment attribute on PEP node.

attributes into a SAMLXACMLAttributeStatement which is then put into a SAMLAssertion that can be signed by the PEP. This final signed assertion containing the client base attributes is referred to as the base assertion.

4.1.4 Contacting the PDP

After the base assertion has been built, the PEP now can contact the PDP. Using an internal referencing PDP, the PEP will determine which external PDP should be contacted for the given access control query by the client.[4] The connection to the PDP occurs on the same thread as the client/PEP connection or session. At this point, the PEP has a base assertion and optionally a set of client assertions. Though not implemented in this XDAC System, assertions could come from any other source like an internal database that a company may maintain, or perhaps from some other third party Attribute Authority that has an exclusive relationship with the PEP. Together the assertions acquired by the PEP and base assertion form an assertion package. Here, the interoperability characteristic of SAML provides the advantage of using the predefined XML structure, SAMLResponses, for encapsulating multiple assertions. Obviously, the intention of using the SAMLResponse in the XDAC model is not the same as the intention in the SAML specifications. Whereas a SAMLResponse was designed to return assertions from a SAML Authority after receiving a request for the assertions, the XDAC model uses the SAMLResponse as a request to a PDP containing assertions. Though it may seem like a shortcut, and though it is a shortcut, it is exactly what is needed. SAMLResponses allow for multiple assertions to be packaged into a single collection which can also be signed if desired.

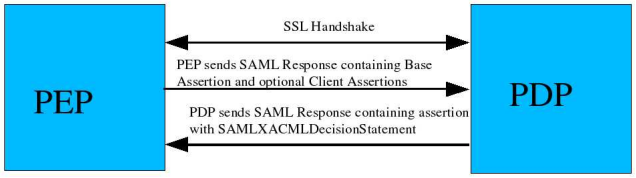

Figure 3: PDP PEP Session

4.2 PDP

4.2.1 Description and Initialization

The PDP is a server application responsible for evaluating XACML Requests in the XDAC Model. Designed as a modular component which exists separate from the PEP node, a PDP

[4]In the current version of the XDAC System, only a single predetermined PDP can be accessed by the PEP. Future versions will implement the internal PDP

can have dual roles. It can either exclusively handle requests from a PEP, returning access decisions after evaluation, or it can act as a reference to other PDPs and/or Attribute Authorities through its stored policies. In terms of the XDAC Model, the PDP departs from the authentication and enforcement duties of the PEP by exclusively managing policies and evaluating requests. Like the PEP server application, the PDP is a threaded server which creates a new thread for each connection request. The options for a PDP server application are listed and described below, and can be manipulated by flags on initialization.

- PORT: Server Application Port (Default: 4444)
- POLICY: Directory where policy files are stored (Default: ./POLICY)
- KS: Location of the key store (Default: ./KEY_DIR/key_PDP)
- KSPASS: Password to access the key store (Default: password)
- TS: Location of the trust store (Default: ./KEY_DIR/trust_PDP)
- TSPASS: Password to access the trust store (Default: password)
- LOG_DIR: Directory where logs are to be stored (Default: LOG)
- LOG_OFF: Deactivate logging (Default: logging is activated)
- ALIAS: Alias used to access key (Default: pdp)
- VALID_PERIOD: Validation period of a PDP assertion (Default 300000 milliseconds or 5 minutes)

4.2.2 Stages of Execution

In a decision session, a PDP steps through four different stages: (1) a filtering assertion stage, (2) a request formulation stage, (3) an evaluation/reference stage, and (4) a response stage.

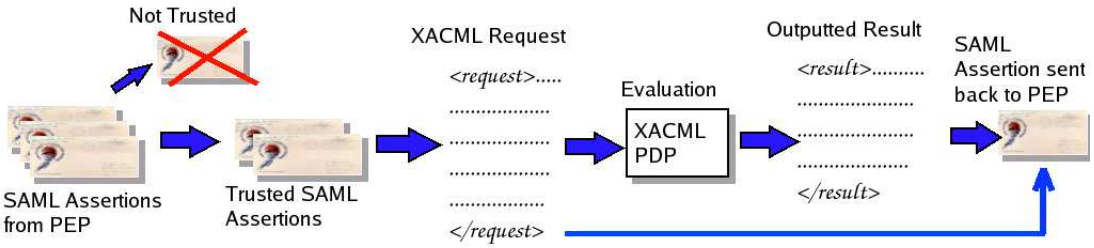

Figure 4: PDP Stages of Execution

In the first stage, the PDP receives a set of PEP signed assertions. From the perspective of the PDP, these assertions could have been originated by anyone. Thus, the PDP checks

the signatures of each assertion, disregarding those that do not make the validation process. After filtering out the assertions that were not validated, the PDP then proceeds to the second stage.

In the request formulation stage, the PDP unpacks the XACML attributes from the assertions, and constructs an XACML request. When formulating a request from a set of attributes, the PDP identifies the category (subject, action, resource, or environment) of each attribute by checking the prefixes of URI ids. For example, if the prefix of the URI is *urn:oasis:names:tc:xacml:1.0:subject*, then the attribute belongs in the subject category. Unlike the current implementation of XACML which allows a URI id to be any type of string, the XDAC system requires that a URI id has a prefix specifying the category. Without this requirement, the PDP could not determine an attribute category because the request formulation exists on the PDP node and because the input from other nodes is an unspecified set of attributes. Once the attributes have been placed in the appropriate category and an XACML Request has been formulated, the PDP then proceeds to the third stage of evaluation.

In the evaluation stage, the XACML request is evaluated against the policies of the PDP. A PDP is not limited to the local policies or attributes received from the PEP. Additionally, a PDP can utilize reference nodes which will be described in detail later. The output of the evaluation is an XACML Response containing the result.

With the result available, the PDP then enters the final stage of the decision session. In this response stage, the PDP formulates an SAMLXACMLDecisionStatment which contains the XACML Result, and optionally the request that was inputted into the evaluator. After packaging the statement into an assertion, the PDP signs the assertion and sends it in a SAML Response to the PEP, thereby closing the session with the PEP.

4.2.3 PDP Connection Terminals

The PDP node is comprised of three connection terminals: a PEP session terminal, an Attribute Authority terminal, and a reference PDP terminal. The PEP session terminal is the connection between a PEP and a PDP. When a PEP sends a set of assertions to a PDP, a threaded connection is created between the two nodes. This threaded session remains open for the remainder of the decision process until the PDP returns an assertion containing the SAMLXACMLDecisionStatement. In the Attribute Authority Terminal, the PDP contacts an Attribute Authority for additional information about a subject. This includes LDAP servers, SAML Authorities, and any other external nodes which can contribute further information about a requester. The protocol between the PDP and the Attribute Authority would ideally be SAML, utilizing SAMLXAMLAttributeStatements. Finally, when a PDP requires more policies to make an accurate decision, it can consult other external PDPs through the reference PDP terminal by requesting policies that the deciding PDP does not own, or it can send all available attributes on to an external PDP, relinquishing the entire request evaluation.

4.2.4 Receiving Attributes versus an XACML Request

One of the significant differences between the XDAC model and a non-distributed access control model using XACML is what the PDP receives from the PEP. In a non-distributed environment, the PEP can formulate the request and hand it to the PDP for evaluation. In the XDAC model though, and as described above, the PDP receives sets of attributes. The advantage of sending attributes rather than a request in a distributed environment is that it allows the PDP to accumulate data from multiple sources, decide on which data is valid, and then combine all the data into a single request. By limiting the input from other nodes to only attributes, the PDP maintains a very simple and efficient protocol which allows any sending node to only be concerned with bundling the attributes. [5]

4.2.5 Reference Node Connections

One of the advantages of a distributed system is that any given PDP node can know everything without really knowing anything. This illogical concept can be attributed to Reference Nodes. So long as a PDP knows who to trust and how to contact those that are trusted, it has a wealth of access points in a network for acquiring any information that it does not already have. With the trust model already built into the XDAC System, the only question that remains is where to implement the functionality. Because contacting a reference node is dependent on the evaluation process, the information about reference nodes should be included within the policies.

The two circumstances when a PDP would contact a reference node are either to acquire more attributes about a subject from an Attribute Authority or to use the policies of another PDP. In either case, the difficult aspect of having reference nodes in the XDAC System is that a PDP must know where to find them, which nodes contain the desired information, and under what circumstances it should contact them.

If a PDP wants to know where to find a reference node, the only information that is required is an IP address and a port. With these two items and a proper protocol for requesting information, a PDP has the ability to automatically connect to another server. Furthermore, the issue of incorporating the reference capability into the already defined evaluation process is now only an issue of extending XACML policies to include IP address and ports.

Still the question remains about which nodes will actually be beneficial for contacting in the first place. What makes a reference node A more likely to be contacted than a reference node B? The process for determining this selection must be based once again, on the policies. Attributes specified in policies would be bound to certain reference nodes. For instance, an attribute about which classes a student is taking may refer a PDP to the registrar server. Or an attribute about financial status might refer a loan agency to a bank PDP.

Besides needing to know where to look, a PDP must also know under what circumstances

[5] This is an example of the common language concept which is described later.

to look, or when is it warranted to make reference connections? There is no easy answer to this question because there are no clear cut guidelines to follow. Clearly, not all decision processes will require that a PDP contact other nodes. Similarly, a PDP cannot always make references because of efficiency issues. Ideally, the PDP should only connect to a reference node when the information gained from the connection will change the outcome of the evaluation. The problem with this scenario is that the only way to know whether or not a connection should have been made is to go ahead and make it and see what decision results. Once again, the solution is viable but incredibly inefficient. In many cases, it depends on the system and the administrators that maintain the system. Still the only automatic functionality available are the policies. A policy of the PDP could specify when is it necessary to contact a reference through the XACML functions and/or rule combining algorithms. For instance, a rule combining algorithm could specify that a permit response be returned from PDP X and Y before returning a permit. Or, if an attribute is necessary for evaluation and not present in the attributes provided by the PEP, the policy could require that a PDP consult a specific IP/port to obtain that information. Both of these examples would suggest a multi-session evaluation process. In a preliminary evaluation, either a typical XACML response would be returned from the evaluator, or in the case where further information is needed, a pending notification would be returned, containing those IP/ports of the relevant reference nodes. If the latter response is returned, then the PDP would have the option to either return a deny back to the PEP or to contact the reference nodes, gather more information, and pursue a secondary evaluation.

By requiring or stipulating for certain conditions to be met before returning a response, the XDAC Model is relieved from the detriment of uncertainty. However, there still remains a considerable amount of gray area involving those cases where contacting reference nodes can be still be useful, yet are not required. What if the original response is deny, but with more information the response would be permit? Or even worse, what if the original response is permit, but with more information the response would be deny? When does a PDP have enough information? All these questions address valid points about the distributed nature of the XDAC System. If efficiency was not a concern, then a PDP should acquire all available information about a subject. If efficiency is a concern however, certain trade offs must occur.

4.2.6 Requesting Policies or Sending Attributes?

When a PDP is referred to another PDP the issue arises of whether or not to request policies or send the client attributes. If a PDP requests for policies then the reference PDP would send the specified policies to the requesting PDP, and the requesting PDP would use the policy as if it were its own. The problem with this paradigm is that not all PDPs are willing to share their policies, especially in the case where the two PDPs are unaffiliated. But, if a PDP cannot acquire the necessary policy to complete an evaluation, then the only option that remains would be to send the client attributes to another external PDP. And, just like the former case, problems do exist. What guarantee does a PDP have that another referencing

PDP will secure all client information? What guarantee does a client have that his or her attributes will not be transmitted across the globe just to check out a book at the library? There is no right or wrong answer. Perhaps the best solution is to have a PDP really consider what it means to trust another node in the model. If a requesting PDP can trust a policy from a reference PDP, why shouldn't the requesting PDP trust the reference PDP with the client attributes? Once again, there is no clear answer. Nevertheless, this is an issue that should be considered in any given implementation.

4.2.7 The Internal PDP

One of the options that the XDAC System offers is an external or internal PDP. To this point, only the external PDP, the PDP that exists on a separate node from the PEP, has been discussed. The internal PDP, a PDP that exists on the same node as the PEP, is very effective in a local and distributed environment. First, and perhaps more obvious, an internal PDP gives the XDAC System the flexibility of being enclosed. An enclosed system localizes the policies and request evaluation. For a system that does not require a distributed environment, the XDAC system can easily be switched to a single access control system which uses XACML.

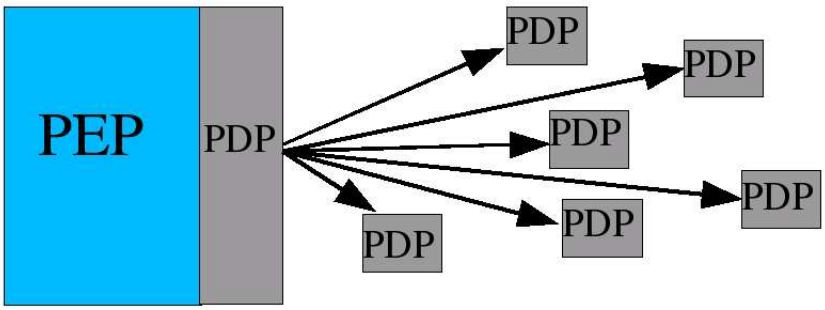

Figure 5: Internal PDP

An internal PDP is more valuable in the XDAC Model when used for PDP to PDP communication in the reference context. Notice that in this XDAC System, the PEP has an exclusive PDP for making decisions. Ideally, a PEP would have a large array of PDPs to contact for the various requests of clients. In order to determine which of the PDPs to contact, the PEP needs some type of mechanism for selection. An internal PDP would act as a referencing PDP which would have a collection of policies whose sole purpose is to examine the base attributes and attributes contained in the client assertions, and then direct the PEP to the appropriate external PDP. The internal PDP would utilize the same functionality for contacting a reference PDP as external PDPs use. The only distinction would be that in order to maintain the division between enforcement and evaluation, an internal PDP would exclusively send client attributes to external PDPs rather than obtain remote policies.

5 Extending SAML with XACML

5.1 Underlying Purpose

The XDAC Model has been described thus far as a collection of nodes utilizing SAML and XACML for access control, however the relationship between the two XML languages has yet to be discussed. With multiple nodes across a network, an XDAC system needs some way to transmit attribute and decision information from one node to another. SAML Assertions include this functionality, however the implementation of opensaml 0.9 does not include XACML Attributes nor the same decision capabilities as XACML Results. The question therefore becomes to what degree can SAML Assertions be combined with XACML Attributes and Results? To answer this, the structure of these classes must be examined.

5.2 XACML Compatibility Issues

A SAML assertion is composed of a collection of SAML Statements. There are three types of SAML Statements: SAMLAttributeStatements, SAMLAuthenticationStatements, and SAMLAuthorizationDecisionStatements. The two statements that best correlate with the XDAC model are the SAMLAttributeStatement which would correspond to sending attributes across nodes, and the SAMLAuthorizationDecisionStatement, which would correspond to sending decisions across nodes. In an exclusive SAML context, each of these would be returned by a SAML Authority in an assertion stating either that a set of attributes is associated with a subject, or that a subject is authorized to do a specified action. Ideally, both of these statements should fit into the XDAC model of communication between the PEP and PDP, however there are some discrepancies which raise compatibility issues.

5.2.1 Conveying Attributes

The SAMLAttributeStatement is designed to hold a collection of SAML Attributes. If in fact a SAML Attribute were equivalent to an XACML attribute, then attribute transmission using SAML with XACML would be inherently compatible. In the current implementation of SAML though, this is not true. A SAML Attribute is flexible in that it contains a collection of values that can be associated with any namespace so long as the value is a simple type. Unlike SAML Attributes though, XACML Attributes contain complex data which cannot be represented in a SAML Attribute context.[6] The depth and detail of the XAML Attribute exceeds the capabilities of the SAML Attribute, making the two languages incompatible at the attribute level.

[6] An XACML Attribute consists of the following: (1) a URI id which identifies the attribute into a category such as group, physician-id, or email (2) An issuer field which describes the attribute issuer (3) An XACML DateTimeAttribute describing the issue instant, and (4) an XACML Attribute Value which can be one of 15 types in the normal XACML package with the capability of expansion.

5.2.2 Decisions Statements

Another discrepancy between SAML and XACML is the description level of decision statements. Through the SAML Authorization Decision Statement, SAML conveys subject attributes, a decision, a resource, a collection of actions, and evidence about the decision. Though this information is extensive and detailed, in the XDAC Model a response from a PDP needs to encompass the same range of data contained in an XACML Result.[7] Despite the SAMLAuthorizationDecisionStatement sharing the functionality of authorizing with the XACML Result, the detail of the XACML Result dismisses any notion of combining the two languages without modifications. For instance, the scope of an XACML decision exceeds the scope of allowable types for a SAML decision. An XACML decision can be either Permit, Deny, Indeterminate, or Not Applicable while a SAML decision can only be Permit, Deny, or Indeterminate. The additional decision type in XACML identifies important information about a PDP evaluation, and thus it should be present in the communication protocol. Furthermore, the SAMLAuthorizationDecisionStatement does not include XACML obligation capabilities which will be a necessary element in granting access of a resource because a policy might specify certain conditions that must be fulfilled first.

The degree to which SAML and XACML differ makes any attempt at incorporating the two languages impossible unless modifications occur. The problem, as described above, stems from the differences between SAML Attributes and XACML Attributes, and the differences between SAML Decision Statements and XACML Results. As a result, if SAML and XACML are to be incorporated, SAML must be extended.

5.3 Extending SAML

When extending a language, consideration must be given to its structure and design in order to avoid incompatibility issues or limiting the original capabilities of the language. In terms of extending SAML, this means that any modifications that are required for adopting XACML Attributes and Results should not in any way affect the original functionality of SAML. Therefore, choosing the right extension point is imperative.

The most logical, and efficient extension point was the SAML statement class because (1) it is the base component of a SAMLAssertion, and (2) it best resembles the desired functionality requiring very little modification if new subclasses were added. The goal therefore was to create subclasses of the SAMLStatement which would contain XACML Attributes rather than SAML Attributes, or in the case of decision responses, XACML Results rather than SAMLDecisionStatements. The advantage of extending on the SAML Statement level is that the template for SAML Assertions remains the same and the abstraction level is consistent with the properties of what a SAMLStatement encompasses.

[7] An XACML Result contains the decision, the status, the resource identifier, and the set of obligations that the PEP must enforce when granting access of a resource.

The two SAMLStatement subclass extensions that were made to SAML were a SAMLXACMLAttributeStatement and a SAMLXACMLDecisionStatement. The SAMLXACMLAttributeStatement contains a SAML Subject and a collection of XACML Attributes. The SAMLXACMLDecisionStatement contains a SAML Subject, XACML Result, and a optional field for an XACML Request. [8] The inclusion of the XACML Request enables a PDP to send the request used in the decision process to the PEP, thereby allowing the PEP to know exactly what attributes were used in the decision process, including those attributes from reference nodes which the PEP would not otherwise have known. In both the SAMLXACMLAttributeStatement and the SAMLXACMLDecisionStatement, a SAMLSubject field was included to allow for additionally subject information in the statement. By extending SAML, conversion techniques are avoided, efficiency is maximized, and the system as a whole shares a common language among the distributed nodes.

6 Security Issues

Despite all of the advantages of a XDAC System, the ultimate question is whether or not security can be maintained on a distributed level? If the scalability of the system increases then so will the vulnerabilities. Who can a PEP trust? Who can a PDP trust? How can communication between components remain private? Can the system still be efficient if it is distributed? Can components be held liable for how they contribute to a decision? Can the integrity of requests and responses be upheld across a network? Can the system as a whole be secure?

If the system is to be secure, then first one needs to know what it means for a XDAC System to be secure. The security of the XDAC System is reliant on the following two invariants: (1) under all circumstances, no entity shall be granted access to a resource to which that particular entity does not have access as defined by the policies of that resource. (2) All information used in the decision process shall remain private to the deciding parties.

Security in a distributed system is different than security on a localized system. In a localized system the containment of the application allows the administrator to make assumptions about how the application executes in that the administrator has the capability of maintaining that application. In a distributed system however, the system is distributed over multiple nodes, most of which do not belong to the host server. Information that originated at an LDAP server, may be sent in an assertion to a client, then on to the PEP, then on the PDP for evaluation, and then finally back to the PEP in a response. And in between each of these nodes, this information travels through the open highways of the internet where hypothetically anyone can see the data or even modify it. Who is to say that this modification will never occur, or even that the destination nodes are who they they say they are? What if the alleged IP address of a PDP is actually the IP of a devious client trying to access the bank records of a company? Then that client can formulate any decision that he or she desires. Security

[8] Returning the XACML request was suggested in the SAML Profile for XACML. Refer to [AL04]

is a major issue of the XDAC System. It relies on Public Key Infrastructure (PKI), logging, client abstraction, and principles of trust.

6.1 PKI

PKI is the combination of software, encryption technologies and services that enables enterprises to protect the security of their communications [Ver04]. By integrating digital certificates, public-key cryptography, and certificate authorities, PKI secures transmission between nodes in a network. Specifically, in the XDAC Model, PKI is responsible for authenticating entities, ensuring liability, encrypting data sent over links, and maintaining integrity in communication.

6.1.1 SSL

One of the technologies that has resulted from PKI is Secure Socket Layer (SSL) protocol. In the XDAC System, SAMLAssertions are sent over SSL between any two nodes including client-PEP, PEP-PDP, and PDP-Reference nodes. SSL can authenticate clients and servers, and provides strong encryption of the data, keeping all information within that data private to the sender and intended receiver. SSL ensures no alterations to the data providing a high level of integrity in communication [Net04]. For each of the links in the XDAC System, Java SSL has been implemented with the requirement of mutual authentication. Mutual authentication is necessary in the XDAC System because mutual trust must be verified in order to maintain the invariants. It especially is useful in the authentication of a client by the PEP because the PEP uses the subject information contained in the certificate presented through SSL as an XACML subject-id attribute which will eventually be a central component in the XACML request. With SSL, any node B receiving data from a node A in the XDAC model is ensured the following: the data really did come from A, the data has not been modified in transit, and no other entity has seen the data.

Now, the only question is whether or not A is really A? This is where the XDAC model relies on PKI. Each certificate presented in the SSL protocol is digitally signed by a certificate authority.[9] If the receiver of the certificate trusts the signer, then that public key belongs to the entity described in the certificate. If A presents a certificate during SSL, the only entity that can complete the SSL handshake with B is the owner of the corresponding private key of which should be A.

[9]Ideally the certificate authority is a trusted third party that can verify the identity of a user who requests a certificate. In this XDAC System however, all certificates are self-signed in order to preserve the simplicity of the XDAC model.

6.1.2 Digital Signatures

Outside of the SSL protocol, digital signatures provide nonrepudiation and integrity. If the first invariant is to be held true, and if the distributed property of the XDAC system is to be effective, then each node that contributes to a decision should be responsible for the data that it contributes. SAML Assertions include XML signing which, if used, bind all the information contained in that assertion to the owner of the private key used for signing. A signed assertion also guarantees that the contents of the assertion have not been modified, increasing the integrity of the data.

One might ask why signing is even necessary when all data transmitted between any two nodes uses SSL protocol which already guarantees integrity and privacy among other things? Signed assertions give the XDAC System nonrepudiation by offering proof of contribution and actions by any particular node in a session. A signed assertion can later be shown to a third party to prove the link between the signer and that assertion. For example, if A sends B attribute information which will be an active component in what type of response B returns, then B wants to make sure that A is responsible for all information it sent. If a A signs the assertion which contains the attribute information, then B can always provea to any third party C, that A vouched for that attribute information.

6.1.3 Validating Assertions

When receiving an assertion, a server application must first validate the signature. The validation process includes four steps: (1) verify that the assertion is signed (2) verify that the signature is accurate (3) verify that the current date/time falls between the validity data/time boundaries of the assertion, and (4) verify that the signer is trusted. If any of these steps fail, the validate function in the XDAC System throws an exception because no entity can be held liable for that assertion, and thus it cannot be trusted. Liability closes the gaps between nodes and puts them face to face in a closed room where everyone is aware of the decision process, and everyone is aware of each other. This awareness backed by the mathematical theory of digital signatures binds a node and a SAML Assertion into a single inseparable unit. The strength of this binding is the backbone of the XDAC Model, and a key element in distribution.

6.2 Logs

As a means of storing signed assertions along with a variety of other data, both the PEP and PDP have logging capabilities built into their functionality. All logs are stored in the log directory which defaults as ./LOG. Each time the server application is initialized, a new log file is created in the log directory. The naming conventions for log files are date-time.log (example: Sat Apr 24 12:58:43 EDT 2004.log), where the date-time corresponds to the exact date and time when the server application was initialized on the server host system. Every

24 hours a new log file is created, using the same directory and naming conventions. Because the log functionality is active at the onset of the server initialization, the logs extend across all server level and client level threads, consolidating a significant amount of information into a single manageable file. In the case where multiple threads might be logging at the same time, the XDAC System uses synchronization to ensure that only one of these threads can be writing to a file at any given time. To identify a given session, unique numerical tags are associated with each thread, and any information written to the log is preceded by that tag.

Logging in the PEP includes all of the following: initialization settings, exceptions and errors, certificates received from clients, inputs from the client, signed assertions received from the PDP, date and time stamps, PDP assertion validation, and decisions sent back to the client. Logging for the PDP includes all of the following: initialization settings, signed assertions from the PEP, certificate information received from PEP, exceptions and errors, date and time stamps, PEP assertion validation, and PDP assertions sent back to PEP. By maintaining the events that occur on a given server and storing the assertions that a server sends and receives, the logs are a key element in ensuring accountability and security.

6.3 Trust Model

If the XDAC System is to function correctly, all the nodes that contribute to a decision must be trusted and all the links that combine the nodes must be secure. This idea of trust is essential to the security of a XDAC System. If a PEP or PDP cannot trust any other node, then it has lost its ability to consult other nodes in a network. If at any point, a node is falsely trusted, then the invariants have been broken and the security of the entire XDAC System is compromised. The overall concept of maintaining trust in a distributed environment is referred to as the trust model.

The trust model is essential to the security of the XDAC System and the most important factor that will contribute to the validity and overall correctness of any decision. Since trust is based on PKI, then the manner in which the keys are handled and stored is essential. PKI specifies the use of local repositories for storing keys. Each node whether a client, PEP, or PDP in the XDAC System has a password protected trust store file which holds those public keys and certificates which that node trusts. In some cases it is only necessary to keep the public key of a root authority who signs and validates the public keys of others. In other cases it is necessary that the actual public key of each possible connecting node be kept.

6.3.1 Client Perspective

From the client's point of view, the main trust issue is whether or not the PEP it requests a resource from is actually the PEP that enforces the access on that resource. If an adversary pretends to be the PEP, then that adversary has the potential of acquiring personal information about the client and/or sending a false resource to the client, each of which breaks

the invariants of the XDAC Model. The client therefore has a truststore containing trusted root certificates and trusted PEP certificates. In order for a client application to commence communication with a PEP through SSL, the certificate sent by the PEP must be linked to a certificate in the clients trust store. Assuming that the client has the sensibility of carefully choosing which certificates it allows in his or her trust store, the client can be ensured that the PEP server that he or she connects to is the PEP server with the corresponding private key from the trust store.

6.3.2 PDP Perspective

From the PDPs point of view, trust is also important because of the service that the PDP offers to the XDAC Model in decision making based on policies. There are two possible trust issues that can occur: an adversary posing as a PEP and an adversary posing as a reference authority. If an adversary poses as a PEP, it is unlikely that a resource will be compromised, however, it is possible that the adversary could learn about the policies and thus breaking the second invariant. For instance, an adversary could send a variety of attribute sets to learn how certain attributes affect the decision of the PDP. This knowledge could then aid the adversary in selecting the attributes needed to gain access to a resource. Granted, the adversary would still need to acquire these attributes from an authority, however, giving an adversary this much of a head start is considered a breach in security.

The other trust issue, and the more important of the two, is the threat of an adversary posing as a reference node. Because a reference node impacts the decision of a PDP through the data that it sends to the PDP, it has a direct impact on whether or not a client is given access to a resource. If an adversary were to pose as a reference node, that adversary could control the decision returned by even a trustful PDP, breaking the first invariant. In order to prevent these trust threats, the PDP maintains two truststores: a trust store containing the certificates of trusted PEPs, and a truststore containing the certificates of trusted reference nodes.

6.3.3 PEP Perspective

The view of the PEP is somewhat unique because it has a vital role in the XDAC model in terms of being linked to the resource. A PEP has all the brawn to keep an unauthorized user out, yet no brains to determine whether or not a user is unauthorized. Thus, the trust model is extremely important for a PEP because a PEP relies on the PDP to give it a valid decision. The PEP has two links with which it must be concerned: the client connection and the PDP connection.

Because the PEP uses the client certificate information in the attributes that it sends to the PDP, it is essential that the client certificate is valid. Since the PEP verifies the client certificate using a client truststore, the only way an adversary can impersonate a client is by

having the private key of that client or having a trusted root authority sign a false certificate, both of which the trust model assumes is not possible.

When the PEP contacts the PDP, it inherently puts all of its trust into the decision of the PDP. Whatever response that is returned, the PEP docilely follows. If an adversary were to successfully imitate a PDP, that adversary has access to any resource for which that PDP is responsible. To prevent this, a PEP has a specific PDP trust store containing all the certificates of those PDPs that are trusted. Thus the only way that an adversary can imitate a PDP is by having the private key of that PDP which is assumed to be impossible.

6.3.4 Potential Vulnerabilities

One of the potential vulnerabilities in the trust model is how the trust stores are maintained. If a system is to be secure, each node must have individual trust stores for each of the types of nodes with which it communicates. The reason behind this philosophy of the trust model is that each of the types of nodes has a different role in the XDAC model. For instance, a trusted client should not have the same capabilities as a trusted PDP. If a PEP were to have only one trust store for all the types of entities that it trusted, a client could potentially act as a trusted PDP which breaks the first invariant because that client could return any decision to a PEP.

The ultimate vulnerability of the trust model and of PKI in general is hijacked keys. If a private key is obtained by an adversary, then that adversary has all the capabilities of the owner of that private key because the entire trust model assumes that if you have the private key, then you are the entity specified in the corresponding certificate. This potential risk is one of key management problems and PKI vulnerabilities. As a result, though XDAC Systems assume that private keys are used by their rightful owners, the overall security of the XDAC System can only be as secure as PKI technology.

6.4 Fail-safe Model

The fail-safe design of the XDAC System is based on the assumption that usability is unpredictable. If the state of the XDAC System at any given time is to fulfill both security invariants, then the system must be able to react to this unpredictable nature of the environment in which it is used. Ideally, transmission between nodes would be guaranteed, every user would have good intentions, and hardware would never fail. Unfortunately, reality does not coincide with this ideal environment, and therefore the onus is placed on the architecture of the system.

The primary manner in which the XDAC System deals with the unpredictable nature of usability is through exception handling. If I/O is occurring, it is assumed that an exception can occur and thus should be handled appropriately without causing the server application to quit. In the unlikely event that a server application does fail, it should not affect the states

of other nodes in the network.

In terms of the XDAC model, the fail-safe design operates on two different levels: within a node and between nodes. Within a node, a server application should be able to handle any error or exception that might occur and maintain the flow of execution. This is achieved by catching and properly dealing with exceptions that might occur.

Between nodes, a server application should not be affected by any unintended input from another node. For example, a PEP expects a properly formed SAMLAssertion containing a SAMLXACMLDecisionStatement from a PDP. However, if a transmission from a PDP to a PEP is terminated prematurely and as a result the PEP receives an incomplete response, the flow of execution should continue. In this scenario or any similar unintended input scenario, when the PEP's XML parser realizes that the input is incorrect it will throw an exception. The PEP then must do two actions: (1) catch the exception, and (2) respond to the client to complete the session. Still there is the question of what should be the response to the client in a case like this or even any other similar situation? Without a complete response, the PEP must abide by the invariants and assume that an incomplete response is a deny response. So, after catching the exception, the PEP responds to the client with a deny.

The concept behind the fail-safe design is to ensure that the foundation of the XDAC system is not affected by the unpredictable nature of the environment. Just like in the example above, the fail-safe design is a two part system. First, the failure must be caught or dealt with in the appropriate manner. And second, the flow of data must continue whether its within the node or between nodes. Because it is possible that the failure is connected to the flow of data, sometimes in order to maintain the communication within a session, it is necessary that the exception handler make assumptions. What these assumptions are depends on the security of the system which in this case are the invariants. In order to maintain the invariants, the fail-safe design defaults to deny under all circumstances. This fail-safe design approach makes the accurate assumption that networks and users are unpredictable and as result maintains the security of the XDAC System.

6.5 Client Abstraction

Client Abstraction refers to the concept of limiting the knowledge of a requester to only what the requester needs to know about a request session. In the XDAC System, a requester only needs to know whether or not the resource was granted, and perhaps obligation information. Any additional information is unnecessary and potentially a security risk. The primary concern involves a client learning about which properties are needed in order to access a resource, or in other words, what a policy contains. For example, XACML Results include three types of deny decisions: deny, indeterminate, and not applicable. If an adversary received a not applicable response rather than a deny response, that adversary now knows that no policies contained by the PDP apply to the information that was passed. It may not provide any direct aid for an attack, however indirectly an adversary could use the responses to learn about how

the system works and potentially where and how to attack it. A step taken to ensure client abstraction in the XDAC System is to limit the negative response from a PEP to only deny, rather than include the reasons for why a deny occurred.

The idea of client abstraction is more of a general philosophy than a concrete security mechanism. The only way an adversary can attack a XDAC System is by having knowledge about how that system functions, and that is why the architecture of XDAC System is designed to convey only necessary information to a client.

6.6 Compromising the XDAC System

Compromising the security of the XDAC System means compromising either of the two invariants. Assuming that all the hardware that executes the XDAC applications are physically secure, and assuming third party PDP and attribute authorities are trusted, there are two access points where an adversary could attempt to accomplish this: (1) the transmission mediums between nodes, and (2) the client gateway.

The transmission mediums consist of the travel areas where session data flows between nodes. Since all communication uses SSL protocol, this is assumed to be a secure channel and not a threat. Conversely, because the client gateway is designed to allow an honest client access to resources, the client-PEP link is an ideal pathway for an adversary. In order to access a resource through this pathway though, an adversary must do at least one of the following: gain access to the PEP or PDP server machine, deceive the PEP, or deceive the PDP.

In each of these cases an adversary must first connect to the PEP which requires client authentication through SSL. This means that even before an attack can occur, an adversary must decide on whether to authenticate his/herself, manage to hijack a private key from someone else, or manage to create a fake identity which is trusted. If the security concepts behind PKI are held true, ideally the only way that an adversary could be authenticated by a PEP would to present the valid certificate owned by the adversary. An adversary should not be able to hijack a private key, and certificate authority should not sign a certificate of someone that is not who they say they are. And because an adversary would likely want to conceal his/her identity the authentication process alone should prevent an attack using the client-PEP pathway. Nevertheless, in the case where an adversary manages to authenticate with the PEP, possible vulnerabilities should still be discussed.

6.6.1 Input Validation

Input validation is a universal vulnerability that can lead to various problems in systems including the XDAC System unless it is considered in the architecture. If an adversary were to use the client-PEP pathway, the most direct means of attacking the XDAC System would be through the request that a client would typically send to the PEP. From the PEP's perspective, a client is to send a string containing the desired resource and a string containing the desired

action. From the adversary's perspective a string can be anything. If the PEP does not consider unintended input, then an adversary could cause problems. For example, suppose an adversary were to enter the following string into the action prompt of the client application as opposed to the expected input:

```
Creditable Input:  open

XACML Request Effects:
      <Action><Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id" DataType="http://www.w3.org
      /2001/XMLSchema#string" Issuer="Dartmouth_PEP" IssueInstant="2004-05-10T20:04:56.074000000-04:00"><Attribute
      Value>open</AttributeValue></Attribute></Action>

Adversary Input:
      open</AttributeValue></Attribute></Action>...President of the Bank.... or whatever I want...<Action><Attribute
      AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id" DataType="http://www.w3.org/2001/XMLSchema#string"
      Issuer="Dartmouth_PEP" IssueInstant="2004-05-10T20:04:56.074000000-04:00"><AttributeValue>open

XACML Request Effects:
      <Action><Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"DataType="http://www.w3.org/2001/
      XMLSchema#string" Issuer="Dartmouth_PEP" IssueInstant="2004-05-10T20:04:56.074000000-04:00"><AttributeValue>open
      </AttributeValue></Attribute></Action>... President of the Bank ... or whatever I want.<Action><Attribute
      AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id" DataType="http://www.w3.org/2001/XMLSchema#string"
      Issuer="Dartmouth_PEP" IssueInstant="2004-05-10T20:04:56.074000000-04:00"><AttributeValue>open</AttributeValue>
      </Attribute></Action>
```

Under the current implementation of XACML, the parser will throw an exception because the request is not in the correct XACML request format. That is not to say that future implementations may not allow for a more flexible arrangement of attributes. As a result code inject injection should be prevented by adding input validation to the PEP. To prevent the injection example above from occurring, the PEP does not accept any type of input from the client prompt that contains the hypertext characters < or >.

6.6.2 Hijacked Assertions

Client assertions, though advantageous to the single sign on paradigm, present the issue of adversary intervention. A client who controls a set of assertions about his/herself can visit many different environments without having to continually consult an authority or present a password. In the XDAC model, the greatest advantage of the client controlled assertions is that a client can control what attributes it gives to the PEP. The problem is that if the assertions are in the control of the client, what is to stop an adversary from stealing and using these signed assertions. In order to prevent hijacked assertions, the XDAC System binds the assertions to the intended user. Any assertion that a client obtains from an attribute authority, in addition to being signed by that authority, must include a hash of the client's public key. The PEP when presented with the assertions then can verify that the assertions indeed belong to the requester by hashing the public key corresponding to the private key in the SSL authentication, and then comparing that hash with the hash in the assertion. If they match, then the PEP can be assured that the requester's public key matches the public key of the assertion.

6.6.3 Caching Environment Reliant Decisions

Due to the overlap in time characteristics between an environment attribute and cache element lifespan, caching presents a potential risk of breaking the first invariant. If the cache lifetime exceeds the environment attribute differential [10], then it is possible for an adversary to access a resource that he/she should not be permitted to access. For instance, suppose that an adversary wants to access an office building where the doors lock at 5 pm from the outside, preventing all people from entering. Assuming the adversary was permitted to enter before 5pm, he enters at 4:59:40, and then leaves once again before all the door are locked. If a cache is used by the PEP controlling the locking mechanism on the door, the adversary then has a entrance time frame equivalent to the lifespan of a cache element in the PEP. Therefore, any cache element lifespan longer that 20 seconds would cause the first invariant to be broken. Even though the log will later divulge the entrance of the adversary along with all identification information, the first invariant has been broken. To combat this vulnerability and others like it, any decision based on an environment attribute should include a do-not-cache obligation.

In the case where an environment attribute is not used in the decision process, it is assumed that the lifetime of a cache element will not interfere with the security of a resource. Nevertheless, the cache lifetime should be kept to a relatively short time so as to not exceed any possibility in which a general attribute of a subject might change, thereby possibly changing the output decision by the PDP.

6.6.4 Denial of Service Attack

A popular attack on an XDAC System would be a Denial of Service (DOS) attack in which an influx of connections would overwhelm the PEP server application. Even with multiple servers running on multiple machines, and even with incredibly large disk space, a XDAC System has a limited capacity. If a DOS attack were to occur, this limited capacity would be divulged, and the PEP would have be affected. One consequence of a DOS attack resulting from multiple connections would be an overload in the network bandwidth causing any given session to be uncharacteristically slow. Another, more severe, consequence would be a complete shutdown of either a PEP or PDP server. If logging is activated, a continuous influx of connections would eventually cause the hard disk space to fill, leading to unexpected problems in the server, the most benign being a complete shutdown.

Combating the DOS attack is a very difficult task which is shared by the security community as a whole. The problem is of course, what constitutes a legitimate session versus an illegitimate DOS session especially when many of the attacking computers belong to innocent people with compromised systems? How to combat DOS attacks is beyond the scope of this paper, however it is important to note that it is a potential vulnerability of the XDAC System. This does not mean that either of the invariants are necessarily broken, but it does mean that resources will be inaccessible which can lead to many problems.

[10] The time required to change a permit response based on environment attributes to deny

6.7 Summary of Security

If an adversary were to choose which system to hack among the many systems that exist around the world, the XDAC System would be the least likely choice. It is secure and it is effective. It utilizes the power and technology of PKI to ensure that liability and integrity is maintained, and that which is supposed to be private remains private. At every point of access it builds a wall which keeps the right people in and the wrong people out. And most importantly, the XDAC System distributes the decision making across multiple nodes in a secure manner, allowing for maximum collaboration between nodes, without relinquishing the security of the system.

7 Efficiency

Efficiency was a major consideration in the design process of the XDAC System. Even if a system is secure and able to make accurate decisions, if that system is not efficient, then in most cases it will be useless. The efficiency of the XDAC System is based on the elapsed time between when a user requests a resource from a PEP and when the user receives the corresponding response. Considering the events that occur and the extent to which data travels in any given access control session, minimizing this elapsed time to improve efficiency is not an easy task.

In terms of the XDAC System, efficiency is difficult to achieve as the result of the distribution property. Though distribution is the main benefit of the XDAC System, it is also the primary source of time consumption. For each instance that a node contacts another node, encryption, decryption, validation, encoding, decoding, and data transmission all must occur in two directions. Each of these actions takes time, suggesting that the best way to make a distributed system efficient is to make it non-distributed. This of course is not an option. Thus, the challenge is to maintain the distribution while minimizing the elapsed time of an access decision. To achieve this, two steps have been taken in the XDAC System: the use of a common language among nodes, and the development of a PEP cache.

7.1 A Common Language

Ideally, any two nodes in a XDAC System should be able to communicate in a reliable and efficient manner despite inherent differences. There are two means of achieving this: either convert data to meet the expectations of a receiving node, or to use a common language.

In the former case where conversions are made from node to node, problems of efficiency and accuracy arise. Converting data from one form to another requires computation, and computation requires time. Since one of the goals of the XDAC System is to create an efficient access control process, conversions pose an unnecessary obstacle. Additionally, a

certain degree of accuracy and validity is always lost due to either scaling down data when making assumptions about how it should fit a desired template, or scaling up data by assuming certain characteristics of how it should fill in those fields of the desired template that are not specified in the original template. In either case, a loss of accuracy can dramatically damage the reliability of the XDAC System.

The better choice, and the choice implemented in the XDAC System is the use of a common language. If a group of people who all speak different languages are expected to complete a task together, the chance that they will be able to do so efficiently would be very small considering they would have no way to communicate. The idea of a common language is similar. If all nodes in a XDAC System used different means of expressing information, completing a single access control task would take a significant amount of time, thus a common language would be an efficient mechanism for communication between nodes.

7.1.1 Choosing the Common Language

Choosing the common language of the XDAC System is based on first determining the single most important unit involved in an access control decision. Once this unit has been identified then it can be applied to the system as a whole. Because of its role as the basic unit of an XACML Request, the XACML Attribute is the best candidate for being the common language. Access control in the XDAC System revolves around XACML and the evaluation process at the PDP node. Assuming that information about a subject has to be acquired from some location, regardless of how that information is stored, or transferred, eventually it will have to be in the form of an XACML Attribute contained in an XACML Request.

7.1.2 Applying a Common Language to the XDAC System

In order for the common language to be effective, the application of the XACML Attribute to the XDAC Model should extend across the three mediums of storage, transmission, and evaluation. The storage medium includes all Attribute Authority nodes. By storing client information in the form of XACML Attributes at the Attribute Authority node level, detailed information could be stored in an organized manner while still matching the unit of evaluation at the PDP node.

In the transmission medium, when an Attribute Authority responds to a request for attributes, it can then easily retrieve the data and enclose it in a SAMLAssertion using the SAMLXACMLAttributeStatement. This can also be applied to the transmission of attributes between a PEP-PDP or client-PEP, where in each case, XACML Attributes are passed within SAML Assertions. In each case, the unit of transmission is consistent with the eventual unit, enhancing efficiency.

When the data reaches the evaluation medium, the PDP then has the luxury of selecting which SAML Assertions are trusted and then unpacking the common component in each one. These

assertions can be from a variety of different locations yet at the base level, they all contain XACML Attributes. From storage to evaluation, conversions were unnecessary and unneeded. Without conversions and with a set XACML Attributes, a PDP can then efficiently build an XACML Request for evaluation.[11]

7.2 Caching

The cache of the XDAC System stores recent decisions in PEP-local memory, making subsequent retrieval significantly more efficient than contacting a PDP once again. After any given decision is received by the PEP from a PDP, if the response is permit and there is no obligation to not cache, the PEP adds the subject, action, and resource strings of the request to a cache element which is stored in singly link list.[12] The lifetime of the element and maximum number of elements in the cache is determinant on the PEP initialization flags: cache_life, and cache_size. After an element has been placed in the cache, any requests that match the index strings of subject, action, resource will immediately be permitted to that resource, bypassing the communication with the PDP.[13]

7.2.1 Functionality

The general design of the cache consists of the following methods: add, searchCache, and update.

- *add*: Appends the new cache element to the end of the singly link list
- *searchCache*: Iterates through each element in the list until finding a match of all three index values. When comparing index values, first the subject is compared, then the resource, then the action. Short-circuiting occurs when a previous comparison is false.
- *update*: A preliminary iteration beginning at the oldest elements in the cache, removing those that have exceeded the element lifespan. The iteration through the list terminates once an element within the lifespan is found or the updater reaches the end of the list.

In order to check if a request is in the cache, before contacting a PDP, a PEP must always confer with the cache. Though this might seem inefficient, the combination of the preliminary update, and singly link list design provides for quick searches through the cache and ultimately a faster and more efficient XDAC System.

[11] Note: The common language concept is reliant on the SAML extension of including XACML Attributes

[12] There are some security risks due to limiting the index values to only subject, action, and resource. Specifically naming conflicts might result in overlapping cache indices, perhaps breaking the first invariant. In future implementations a hash of the public key will also be used as an index in order to minimize overlapping indices.

[13] Decisions based on cached items cannot be cached

7.2.2 Levels of Usage

There are three levels in which the cache can be used in the XDAC System: local, global, or no cache. On initialization the cache level is set using the flags cache_global, cache_local, or the default which is no cache. The local cache is session dependent, remaining local to the threaded session between the client and PEP. Once the socket is closed, the session ends and the cache is terminated. An example of when a local cache would be advantageous would be in the scenario of a client accessing multiple files or subdirectories in a permitted directory of a file system. It would be inefficient if for every file or subdirectory access within the directory the PEP would have to contact the PDP and then wait for a decision. Rather, the local cache would keep the decision in memory and as a result, all subsequent access requests within the directory would be answered by the cache rather than the PDP.

Unlike the local cache, the global cache is session independent. An example of when a global cache would be advantageous would be in the scenario of an employee accessing his office that requires an entrance through a PEP locked door. Considering that employee may enter and leave his office multiple times throughout the day, where each access to the door would be a separate session, having a global cache would provide for an efficient means of subsequent accesses.

8 Applications

Never before has an access control system had the flexibility provided by the XDAC System. The applications are only limited to the imagination. Because of its modular design and scalability to any network, the XDAC System can be shaped to fit the needs of any institution. In an attempt to encapsulate all the possible applications of the XDAC System, three general models based on the arrangement of the nodes of the XDAC System are described below:

8.1 One-to-One Model

The One-to-One model consists of two nodes, one PEP and one PDP. The model emphasizes the advantages of separating the enforcement process from the decision making process, and is an integral part of the XDAC System.

The PEP is the divider between the requester and the resources, and its implementation should be simple and straightforward. A permit response is the only situation in which the PEP should allow access. The PEP does not have to be concerned with the computation of decisions or any other decision dependent problems that might occur. A modular PEP can be modified to best meet the needs of the system it protects just so long as the communication front is consistent to communicate with PDPs.

The PDP provides exclusive decision making to the model. Because of its modular design

in the One-to-One model, it can be stored in a maximum security location without altering the flow of data in the XDAC System. This is advantageous considering that a PDP is the holder of the policies. A modular PDP could be stored in a 4758 security device in order to prevent most if not all physical attacks, and any attempts to access the information (policies and executable code) that a PDP would hold. The One-to-One model is the base model in the XDAC System. It can be applied to all other models.

8.2 Many-to-One Model

The Many-to-One model consists of multiple nodes, two or more PEP's and one PDP. Any PEP-PDP connection could also encompass the properties of the One-to-One model. The main advantage of the Many-to-One model is reliability. Each of the PEPs consult the same PDP which means that all requests are based on the same set of policies and same functionality embodied in that PDP. Besides reliability, this model allows for a single central authority to maintain the security of multiple resource centers each controlled by its own PEP. In this case, it would be advantageous for the PEP to be stored in a 4758 as a means to ensure that the PEPs are secure when separated from the administrator's watch. In large institutions where there are a variety of types of resources and few administrators to govern all these resources, the Many-to-One model would be ideal.

8.3 One-to-Many Model

The One-to-Many model consists of multiple nodes, one PEP and two or more PDPs. Any PEP-PDP relationship could also encompass the properties of the One-to-One model. The One-to-Many model is representative of most applications in the XDAC System. The general idea behind this model is to separate the decision making amongst many nodes. One would want to distribute the decision process for a variety of reasons: manageability, agreement checking, consulting, and building an attribute authority architecture.

A common theme shared in each of the following applications is dividing the computational load and decision making in order to achieve some purpose. The distributed characteristic of the XDAC System is as much a tool as it is a description of how the components are arranged. The tool can be used on many different levels and in many different ways, and is by no means limited to the examples presented below.

8.3.1 Manageability

Manageability involves taking a larger access control system and dividing it into smaller, more manageable parts. Each part would have its own PDP with its own set of policies where a PDP would still have the capability to contact other PDPs. In a large institution like Dartmouth, it would be beneficial to have a PDP for each department on campus. This would allow those

departments to change their policies as the policies of that department change. Furthermore, it would prevent putting the tremendous work load of maintaining multiple PDPs and policies on one administrator.

8.3.2 Agreement Checking

Agreement checking is based on the concept of checks and balances. With a distributed set of PDPs conferring on the access of a given set of resources through a single PEP, an internal PDP (connected to the PEP) could require that n specific PDPs return permit in order for a resource to be accessed. For example, suppose a resource was the root password of a system maintained by three different groups, yet not known by any individual group. In case of an emergency where the system fails, it might be necessary to use the root password in order to restart the system, making it is necessary to have it accessible. In order for the root password to be read though, the PDP from each group must return a permit response. If each PDP were to utilize logging, then each group would be able to use the root password in case of emergency, but that group would not be able to do so without alerting the other three groups.

An extreme example would be the the process required to use a nuclear warhead. Ideally the nuclear warhead PEP would have to confer with the PDPs controlled by the White House, each senator's office, and the chief military officer. Only if all of the PDPs return permit is the warhead allowed to be used. Furthermore the internal PDP would limit the use of this resource to the only the parties above, which would create a two tier access control process. Agreement checking utilizes the distributed nature of the XDAC System in a very practical and effective manner. By separating the deciding groups that are involved in the decision process, the result is legitimate, fair, unbiased, and hopefully right, preventing mistakes from occurring.

8.3.3 Consulting

Consulting is a very specific One-to-Many model type in which the deciding PDP is linked to the requester, and therefore has a better idea about how to evaluate attributes about the requester. Typically, the PEP would be controlled by a third party that knows very little about the requester. For example, a student at Dartmouth might want to access the JSTOR web site. Dartmouth subscribes to JSTOR and so the student has the right to access the database. When going to the website the student presents some proof of identity and tells the JSTOR PEP (perhaps using XACML Attributes or through information in the certificate) that Dartmouth will vouch for him. The JSTOR PEP then consults the Dartmouth PDP, sending the attributes of the student. The Dartmouth PDP then can (1) confirm that the requester is in fact a Dartmouth Student by retrieving attributes from the Dartmouth LDAP server, in addition to the attributes sent from JSTOR PEP, and (2) refer to its policies to verify that that particular student is allowed to access the JSTOR database. If the Dartmouth PDP decides that the student is allowed to use JSTOR, then it sends a permit response to the

JSTOR PEP which then grants access to the student. As seen in the example, the primary advantage of the One-to-Many consulting type is that the PEP can pass the access control evaluation to subscribing parties while still being the main hub for all client access. Therefore, any given PDP can have its own stipulations for access control to a third party resource shared by many different nodes.

8.3.4 XACML based SAML Authority

One interesting application of the One-To-Many Model is an Attribute Authority. Throughout this paper, Attribute Authorities have been shown to be an essential component in the XDAC Model by asserting client characteristics. How these assertions are acquired is actually no different than the model that has been presented thus far. The desired resource is a set of attributes. The action is to read or download. And, the client is either a user or a PDP seeking information about a user. As a result, the PEP would be the enforcement mechanism protecting the attributes, or in other words, the PEP would be the actual attribute authority. This attribute authority would refer to either an external or internal PDP to determine whether or not attributes should be given to requesters. And, the policies would specify the guidelines for what is required to allow access to the attributes. In this application, the policies could also be the mechanism for specifying privacy by allowing attribute owners to modify their own policies.

9 Recommendations for Future Implementations

The current implementation of the XDAC System is a basic example of what a XDAC System can achieve. Much of what has been mentioned thus far is a design model for future implementations and a recommendation for which direction newer versions should head. For anyone who wishes to continue the XDAC System, the following recommendations should be considered.

9.1 Internal Reference PDP

Perhaps the most important element that has not been included in the current version is the internal referencing PDP. If the XDAC System is to be distributed, it needs to have reference functionality for selecting a PDP. This capability will require an extension to XACML in two ways: (1) enable XACML policies to specify IP addresses of possible reference connections such as PDPs and Attribute Authorities, and (2) include a pending XACML response and a special obligation which would suggest a location for acquiring additional subject information and/or policies to aid a subsequent evaluation. This capability would also be applicable to external PDPs referencing other PDPs and/or Attribute Authorities.

9.2 Secure Platform Adaptation

One task that would increase the security of the XDAC System would be to store the PDP/PEP on a secure platform like the 4758. In its current implementation, the XDAC System would exceed the capacity of the 4758 however by using scaled down libraries, it would be possible to reduce the size of the application. Another option would be to wait for the computer industry to create newer secure platforms with more memory. In either case, by running the PEP/PDP on a secure platform, many potential vulnerabilities will be prevented.

9.3 PDP to Reference Node Communication

Though mentioned in this paper, the PDP to reference node communication is not implemented in the XDAC System. The description and design has been discussed at length, and now the only step that must be taken is to code the capability into the current PDP framework. Similar to the reference PDP, the communication will rely primarily on an extension of XACML policies and Responses.

9.4 Privacy Issues

Privacy and security are equally important in modern day computing. Which is more important is a current issue in government and institutions around the world. In terms of the XDAC System where subject attributes may be transmitted across the internet for any given access control decision, the question of who sees what is one that must be considered. One step taken in the current implementation is giving the client the ability to acquire and send his/her own assertion directly to the PEP, choosing which attributes are appropriate. Unfortunately, in many cases, the client will not know what attributes are needed for an access control decision therefore deciding on which attributes to send becomes more of a guessing game.

Ideally the requester would be able to specify privacy levels for each of the attributes that belong to him/her, however in an automated environment like the XDAC System, controlling the direction in which these attributes travel is a difficult task. There are two methods that should be implemented in order to ensure privacy in the XDAC System.

First, the client should have access to all attributes that might be used in the decision making process and thereby specify a numerical privacy level for each attribute and/or exemptions of who can see which attributes. Assuming that attributes would be stored on an Attribute Authority server, this would require a client having an idea about who knows what about him/her, which might not always be clear to the common user. By specifying privacy levels and exceptions, the Attribute Authority would be able to release attributes according to the client's specifications.

Secondly, with the privacy levels and viewer exceptions for attributes, a negotiation process

could be developed whenever attributes are requested from a particular node. This negotiation process might be based on a progressive model of continually checking if what has been given is enough for a valid decision, or it might be based on some other algorithm.

9.5 Obligations

Implementing obligations is necessary for specifying restrictions on how resources are used, or what must be fulfilled by a PEP before granting access to a requester. Obligations should be directly affiliated with the access of a particular resource. If at any time they are not met, the PEP should be able to discontinue access.

9.6 Upgrading to Newer Versions of SAML and XACML

At the beginning of this project, the current version of SAML was 0.9, and the current version of XACML was 1.1. Now six months later, SAML is at version 1.0, and XACML is soon to be at version 2.0. Upgrading the XDAC System to the new versions of these XML languages would be necessary for security reasons, and beneficial for using some of the newer functionality.

9.7 Policy Formation

As the development of XACML continues and as XACML slowly establishes itself in the computing world, one of the biggest challenges that users will face will be maintaining and creating policies. REVA, a new project which can be found at sourceforge.org, aims at accomplishing just this. The success or failure of XACML and the XDAC System will somewhat rely on the success of REVA as a bridge to allowing administrators fully take advantage of and understand XACML's capabilities. This project has just begun and will be a very important development in the future.

10 Problems and Obstacles

In a large scale project, what is learned from the first night of research to the last line of code, is a tremendous amount. Building the XDAC System required patience, intuition, determination, and desire. And from the first web site that I researched to the last line of code that I wrote concerning a bug that took me three days to find, I have learned a great deal. In this section I hope to explain some of the difficulties that I encountered and how I overcame them in hopes of making future endeavors by others easier.

I would first like to discuss the research process involved in building the XDAC System. The foundation of the XDAC System was built on SAML and XACML, and knowing exactly what

these languages could do, and how to use them was imperative. When researching SAML and XACML, the question that I immediately found myself asking was how do I learn about languages that are still in development? In general third party descriptions were nonexistent primary because the novelty of SAML and XACML. Perhaps the closest outside sources that I found were brief articles which generally described this new *thing* coming out, or an occasional message board archive with past emails sent from people as confused as I was. After scouring the internet, the best sources of information on SAML and XACML were the development websites and the developers themselves who surprisingly responded to my emails usually within two hours. Overall, communicating with these developers, Scott Cantor of Ohio State (SAML) and Seth Proctor of Sun (XACML), and familiarizing myself with the code were the two actions that best helped me learn about SAML and XACML.

Another difficulty of using languages in development is that the languages are in development. Finding a bug is not limited to the scope of the code one just wrote, but rather applies to developmental languages as well, which can of course complicate debugging immensely. Thus, it is very important to periodically check the CVS repositories at the development sites.

A specific problem that had me at a standstill for two weeks was getting the XML signatures to function correctly for SAMLAssertion signing. After consulting Scott Cantor, I finally discovered that the difficultly was not one of coding, but rather including a missing package. Due to export regulations on encryption libraries, the xmlsec.jar file used for XML signing did not include the Java Cryptographic Extension (JCE). Therefore, each time I attempted to sign a SAMLAssertion, I received an Algorithm not found exception. To fix this, I had to acquire the JCE from bouncycastle.org, [Bou04] include the JCE in the xmlsec.jar, and then rebuild the jar. Additionally, it was necessary that the version of xmlsec was 1.1 or greater, though I do not know the reason.

Finally, the most pervasive challenge that I faced was one of familiarizing myself with the software development world. Even with a rather extensive programming and security background, I found that a substantial amount must be learned just to get a project like the XDAC System off the ground. To a developer novice, which I consider myself even still, there are many tools and concepts that exist that no one would have any way of knowing or understanding until getting ones feet wet with a large scale project. The range of knowledge about these tools that I refer to exceeds the current scope of discussion, but the advice that I would give to the next person would be to find the good sites, find the good tutorials, and check the message boards. Chances are that if you are having a problem, someone else has already encountered it, asked about it, and solved it.

11 Summary

Network security is an ever evolving and necessary component of the computing world. If it is to be maintained now and in the future, flexible, dynamic, and universal systems must be used as a means of ensuring the security of resources. In this thesis, I have developed

a Distributed Access Control system which meets these specifications by using SAML and XACML, two upcoming XML-encoded languages for the transmission of client attributes and access control evaluation of requests based on policies. The XDAC System consists of two main elements: a protocol for secure and accountable node to node communication, and server applications. In developing the protocol, I extended the SAMLStatement class to include both the SAMLXACMLDecisionStatement class and the SAMLXACMLAttributeStatement class in order to apply the nonrepudiation aspect to node communication, as well as to incorporate a popular and effective attribute transmission language like SAML with XACML Attributes and Results. All communication between nodes uses SSL and PKI for secure and trusted communication.

In addition to the protocol, I built a PEP server application and a PDP server application, both of which are the basic building blocks of the XDAC System. The PEP and PDP have threading capabilities to allow for large scale access control. To expedite access control decisions, I programmed a caching system that runs in conjunction with the PEP. And, as a means of accounting for the decisions and actions of nodes in the XDAC System, the server applications contain logging functionality. Finally, I have provided a detailed description of reference PDPs, security issues, and possible applications of the XDAC System.

The capabilities of the XDAC System distinguish it from other systems like Shibboleth and Cardea in that its modular design and effective node to node to protocol create a distributed environment with a wealth of advantages. Whereas Shibboleth is a specific instance of distributed access control for higher education, the XDAC System provides a flexible blueprint for any type of access control system. And, unlike Cardea, the XDAC System has an XACML extended SAML language which enables for efficient communication across nodes and an accurate packaging mechanism for XACML Attributes and Responses.

Overall, by separating the enforcement process from the decision making process, the XDAC System has lent itself to be an incredibly flexible application. Its modular architecture scales to the internet and allows individual entities to play a large role in the decision process of a universal resource. It is platform independent and is not limited to any given environment that may require access control. It embodies the universal, widely adaptable, and powerful characteristics of XACML and the accountable and cross-node capable communication aspects of SAML. And finally, even with all of the innovations, the XDAC System is secure, efficient, and at the forefront of ensuring access control in the future.

12 Glossary

- **access control**: controlling access in accordance with a policy.
- **assertion package**: a package of assertions that include the base assertion and the client assertions.
- **attribute authority**: an authority which manages a repository containing client information, and a component of the XDAC Model. An LDAP server would be an example of an attribute authority.
- **base assertion**: The assertion composed of the base attributes acquired by a PEP from the client.
- **base attributes**: The XACML attributes formulated from the DN name of the client certificate, the inputted action and resource by the client, and the environment. The base attributes are compiled by the PEP and are bundled into the base assertion.
- **client**: the requester of a resource.
- **control point**: the point at which a resource can be accessed. The PEP runs on a control point.
- **XDAC Model**: Distributed Access Control Model; a representation including the component structure of nodes and the communication path between nodes in the XDAC System.
- **XDAC System**: Distributed Access Control System; the classification used to describe a collection of nodes that together achieve access control for a given set of resources.
- **institution**: a business, college, government organization, or affiliated group.
- **Internet**: the specific, historic, ubiquitous worldwide digital communications network.
- **internet**: any interconnection among or between private, industrial or governmental computer networks.
- **Link**: In the XDAC System, a connection between any two nodes. Links include client-PEP, PEP-PDP, and PDP-Reference Node.
- **node**: The component of a distributed system which in the case of the XDAC System can be a client, PEP, PDP, or Attribute Authority.
- **PDP**: Policy Decision Point; a component of the XDAC Model which evaluates requests from a PEP and returns a response to that PEP of Permit, Not Applicable, Indeterminate, or Deny. A PDP can optionally refer to an attribute authority or other PDP through a reference connection.

- **PEP**: Policy Enforcement Point: a component of the XDAC Model which enforces the access decision returned by a PDP.
- **XACML Policy**: a set of rules written in XML. A PDP evaluates an XACML Request against XACML Policies.
- **reference node**: Either an attribute authority or another PDP. Reference nodes are contacted by a PDP who seeks further information about a client.
- **session**: the period from when a client is authenticated by a PEP to when a client disconnects from the PEP.
- **SAML**: Security Assertion Markup Language; an XML framework for exchanging authentication and authorization information.
- **trust model**: The description of what is necessary to maintain trust in the XDAC System.
- **XACML**: eXtensible Access Control Markup Language; an XML framework.

References

[AL04] Anne Anderson and Hal Lockhart. Xacml profile for saml 2.0: Working draft. *wd-xacml-saml-profile-02.pdf*, March 2004.

[Bou04] BouncyCastle. Jce jar file. *http://cvs.internet2.edu/cgi-bin/viewcvs.cgi/opensaml/java/lib/Attic/bcprov-jdk14-117.jar*, April 2004.

[Int02] Internet2. Shibboleth faq. *http://shibboleth.internet2.edu/docs/draft-internet2-shibboleth-faq-v07.txt*, 2002.

[Int04a] Miniwatts International. Internet world stats, usage and population statistics. *http://www.internetworldstats.com/emarketing.htm*, May 2004.

[Int04b] Internet2. Saml api. *http://wayf.internet2.edu/opensaml/java/doc/api/*, February 2004.

[Lep03] Rebekah Lepro. Cardea: Dynamic access control in distributed systems. *http://www.nas.nasa.gov/Research/Reports/Techreports/2003/PDF/nas-03-020.pdf*, November 2003.

[Mic03] Sun Microsystems. Xacml api: Sun's xacml implementation. *http://sunxacml.sourceforge.net/javadoc/index.html*, 2003.

[Net04] Layer N Networks. Ssl advantages. *http://www.layern.com/html/ssl_advantages.html*, March 2004.

[Oas03] Oasis. Oasis: extensible access control markup language. *http://www.oasis-open.org/committees/xacml/repository/cs-xacml-specification-1.1.pdf*, July 2003.

[Par04] Ian Parkinson. Custom ssl for advanced jsse developers. *http://www-106.ibm.com/developerworks/java/library/j-customssl*, March 2004.

[Pro03] Seth Proctor. Sun's xacml implementation: Programmer's guide for version 1.1. *http://sunxacml.sourceforge.net/guide.html*, November 2003.

[Ros02] L. Rosencrance. Saml secures web services. *http://www.computerworld.com/*, August 2002.

[Ude02] J. Udell. Finessing pki. *http://www.infoworld.com/article/02/10/15/021017opwebserv_1.html*, October 2002.

[Ver04] Verisign. Understanding pki. *http://verisign.netscape.com/security/pki/understanding.html, visited 2004-05-17*, May 2004.

[wo04] webmaster@oasis open.org. Oasis security services tc. *http://www.oasis-open.org/committees/tc_home.php?w_abbrev=security*, May 2004.

Programming Citations

- SSL Protocol examples [Par04]
- JCE Package [Bou04]
- SAML API [Int04b]
- XACML API [Mic03]
- XACML Specification [Oas03]

APPENDIX

A The XDAC Model Flow of Execution

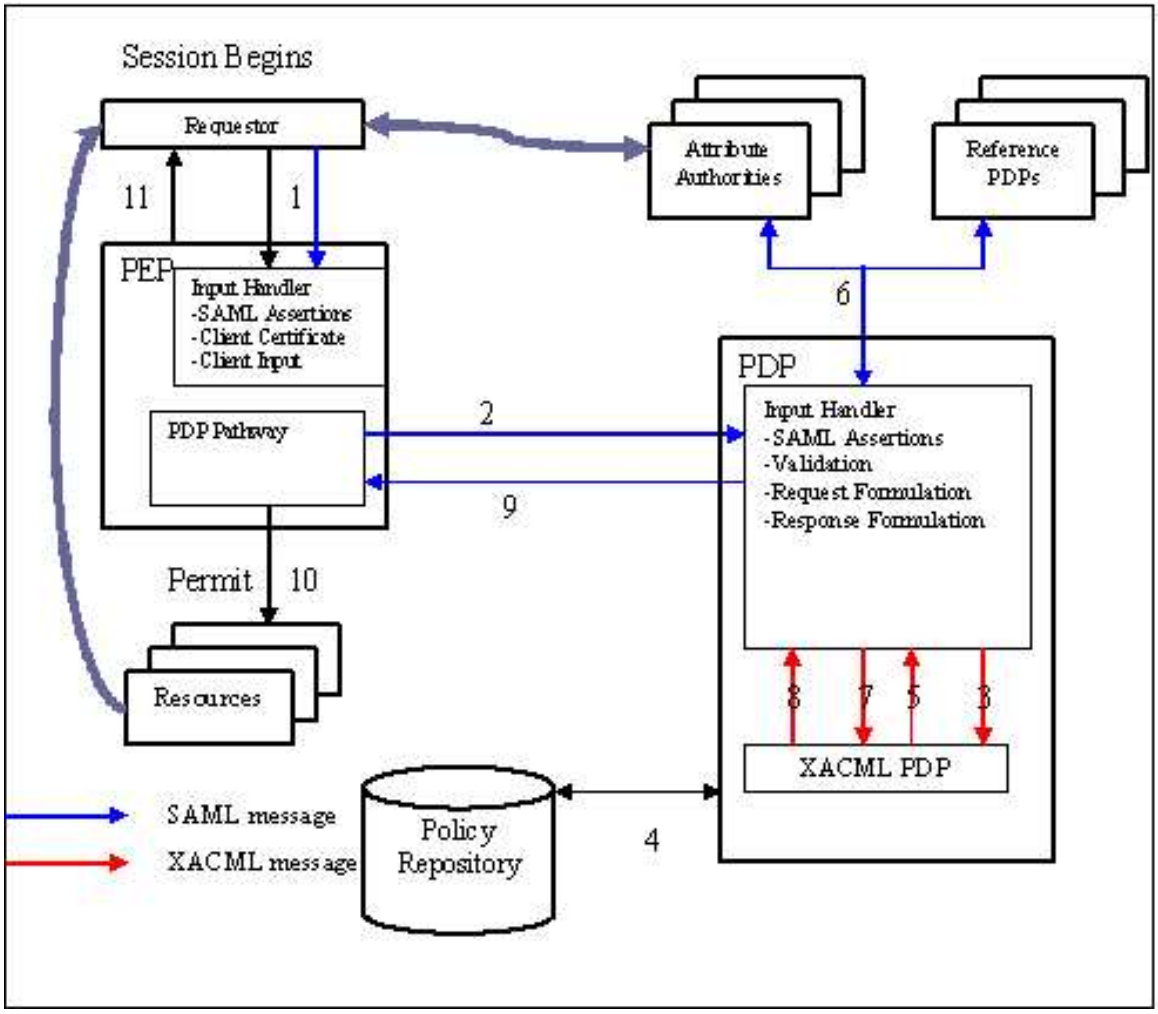

Figure 6: Diagram of the XDAC Model (based on diagram by Fang Pei fang.pei@dartmouth.edu)

1. **The Requestor Connects to the PEP over SSL**. The requestor optionally sends client assertions acquired from the attribute authority. During this connection, the requestor inputs an action and a resource to the input handler. The subject attribute is obtained through the distingushed name of client certificate and environment attribute is built by the PEP based on the current environment. Together these attributes are compiled to form the base assertion.

2. **PEP Sends Assertions to PDP over SSL**. If no cache match is found matching the base

attributes, the PEP signs the base assertion and includes it along with any client assertions into a SAML Response. This package of data is set to the PDP. On receiving these assertions, the PDP validates the assertions, keeping only those that it trusts. From these remaining assertions, the PDP forms an XACML request.

3. **PDP Sends XACML Request to XACML PDP** With the XACML Request, the PDP consults the evaulating XACML PDP.

4. **XACML PDP Evaluates Request Against Policies** The XACML PDP refers to the locally stored policies in the Policy Repository, evaluating the XACML Request.

5. **XACML PDP Returns XACML Response to PDP.** The decision in the response can be either Permit, Deny, Not Applicable, Indeterminite, or Pending. The response can optionally include an obligation. The PDP has the option of skipping to step 9 and responding to the PEP, or continuing to step 6 for further evaluation. The PDP would continue to step 6 if a Pending response was returned along with an obligation specifying which reference nodes are to be contacted for further information.

6. **PDP Contacts Reference Nodes.** The PDP contacts the reference nodes specified in the obligation. A reference node can be either an Attribute Authority or another PDP. In the case of the attribute authority, the request/response process for client attributes can utilize SAML Requests, thereby having SAMLXACMLAttributeStatements returned in SAML Assertions. In the case of reference PDPs, either policies could be requested for, or the PDP could send the assertions from the PEP. The response would either be a set of policies or a SAMLXACMLDecisionStatement, both contained within signed assertions. The PDP then compiles this newly acquired information along with the assertions from the PEP into a new XACML Request.

7. **PDP Sends Updated XACML Request to XACML PDP.** The XACML PDP evaluates the updated XACML Request, referring to the Policy Repository once again, and then formulates an XACML Response.

8. **XACML PDP Returns Updated XACML Response to PDP.** Same as step 5.

9. **PDP Responds to the PEP.** PDP extracts XACML Result from XACML Response. The XACML Result is included in a SAMLXACMLDecisionStatement along with the XACML Request used by the XACML PDP. The SAMLXACMLDecsionStatement is enclosed within a SAMLAssertion which is signed by the PDP. The signed assertion is sent to the PEP in a SAML Response. On receiving the response, if the decision is not permit, then the PEP moves to step 11. If the response is permit, then the PEP continues on to step 10.

10. **PEP Prepares a Resource Access Handle.** If the decision returned by the PDP is permit, the PEP creates a handle which will allow the client to access the resource with the requested action, under the specification of any obligations.

11. **PEP Responds to Client.** If the decision is deny, indeterminite, or not applicable, the PEP returns a deny response to the Client. If the decision is permit, the PEP returns a

permit response along with the handle to the resource.

B PEP-local Environment Attribute Formulation

In the distributed environment, the PEP assumes the PDP will use environment attributes, even though an XACML request does not require environment attributes. This consideration was made on the basis that giving more information can only aid the evaluation process, while including one extra line of XML will not really effect efficiency. One question that did arise when using environment attributes on the distributed level is which environment should be considered? In a XDAC system where the client may be in location X, the PEP is in location Y, and the PDP is in location Z, which environment should be correct in the decision making process? There are many arguments that can be made for each of the sides. The best answer is consistency. As long as everyone plays by the same rules, then the system will function correctly. However in a distributed environment, getting all the nodes in the internet to agree would be near impossible, and thus in the scenario where a policy specifies that a user can access a building between 8am and 5pm, as long as the request is made in the right time zone, there is the chance that a user can have access to the building at any time, an obvious security problem. This scenario argues against having the client application be the determiner of the environment. But depending on the implementation, similar arguments can be made against the PEP and PDP. In this system, the PEP's environment is used based on the belief that the environment of the resources best applies to request, assuming that the PEP and the resources are on the same machine. Thus, similar to the subject-id, action, and resource, the PEP creates an XACML attribute for the environment of the PEP.

C Example

In the following example, a Dartmouth Student Bluto requests to read from the website http://www.jstor.org/. He requests the resource from the JSTOR PEP. The JSTOR PEP consults the Dartmouth PDP, who evaluates the attributes sent from the PEP. The Dartmouth PDP returns a response to the JSTOR PEP who then permits Bluto to access the resource. Included below are the following (1) the intialization settings of the PEP (2) the initialization settings of the PDP (3) The client assertions attained from an attribute authority (4) A snapshot of the text interface for a client (5) the XACML Request formulated by the PDP based on the assertions from PEP (6)the policy used for JSTOR requests (7) the PEP log after the access control session (8) the PDP log after the access control session.

C.1 Initializing PEP

```
>> java XPEP -external -cache_global

*****************Welcome to the PEP server************************

PEP Initialized to the following settings...
PDP TYPE..................External
```

```
PDP LOCATION..............mazook.kiewit.dartmouth.edu:4444
PEP-PDP Protocol..........SAML over SSL
ASSERTION VALIDITY.........300000 milliseconds
PEP-CLIENT PROTOCOL........SSL
CLIENT AUTHENTICATION......Enabled
LOGGING...................Enabled
LOG DIRECTORY.............../LOG
RETURN REQUEST............Enabled
CACHE SETTING.............off

Initialized on Mon May 24 19:34:36 EDT 2004

Listening for connections...
```

C.2 Initializing PDP

```
>>java XPDP

****************Welcome to the PDP server************************

PDP Initialized to the following settings...
Port......................4444
POLICY DIRECTORY............/POLICY_DIR
LOGGING...................Enabled
LOG DIRECTORY.............../LOG

Initialized on Mon May 24 19:35:01 EDT 2004


PDP running on port 4444 in SSL mode
Waiting for request connections...
```

C.3 Client Assertions from Attribute Authority

```
<Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion" xmlns:saml="urn:oasis:names:tc:SAML:
    1.0:assertion" xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol" AssertionID="e5e85c3a7
    c1f6d663877fef90aae3144" IssueInstant="2004-05-24T19:29:03Z" Issuer="LDAP" MajorVersion=
    "1" MinorVersion="1">

<Conditions NotBefore="2004-05-24T19:29:03Z" NotOnOrAfter="2004-05-25T19:29:03Z"></Conditions>

<XACMLAttributeStatement>
           <Subject><NameIdentifier NameQualifier="1269287">Bluto</NameIdentifier></Subject>
           <XACMLAttributeSet>
                <Attribute xmlns="urn:oasis:names:tc:xacml:1.0:context" AttributeId="urn:
                   oasis:names:tc:xacml:1.0:subject:email"DataType="urn:oasis:names:tc:xacml
                   :1.0:data-type:rfc822Name" IssueInstant="2004-05-24T15:29:03.684000000-04:00"
                    Issuer="null">
                <AttributeValue>bluto@dartmouth.edu</AttributeValue>
                </Attribute>
                <Attribute xmlns="urn:oasis:names:tc:xacml:1.0:context" AttributeId="urn:oasis
                    :names:tc:xacml:1.0:subject:group" DataType="http://www.w3.org/2001/XMLSch
                    ema#string" IssueInstant="2004-05-24T15:29:03.684000000-04:00" Issuer="sean
                    .smith@dartmouth.edu">
                <AttributeValue>PKI Group</AttributeValue>
                 </Attribute>
          </XACMLAttributeSet>
</XACMLAttributeStatement>
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
```

```
<ds:SignedInfo>
<ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
    </ds:CanonicalizationMethod>
<ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"></ds:SignatureMethod>
<ds:Reference URI="#e5e85c3a7c1f6d663877fef90aae3144">
<ds:Transforms>
<ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"></ds:Transform>
<ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"><ec:InclusiveNamespaces
          xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="#default code ds
          kind rw saml samlp typens"></ec:InclusiveNamespaces>
</ds:Transform>
</ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"></ds:DigestMethod>
<ds:DigestValue>nwSz9gHO/JzzwfaF3aqak//x1o8=</ds:DigestValue>
</ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>cbcOsqnMVYNkOuJ4oRj+iRjGfTUA8Cm/A+ij4fhlvbZ/HylpbjS43Q==</ds:SignatureValue>
<ds:KeyInfo>
<ds:X509Data>
<ds:X509Certificate>
MIIDGjCCAtgCBEB6szwwCwYHKoZIzjgEAwUAMHMxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJNQTEP
MA0GA1UEBxMGQm9zdG9uMRcwFQYDVQQKEw5MREFQIFVubGltaXRlZDEXMBUGA1UECxMOVHJ1c3Rl
ZCBTb3VyY2UxFDASBgNVBAMTC2xkYXA6U2VydmVyMB4XDTA0MDQxMjE1MTgyMFoXDTA0MDcxMTE1
MTgyMFowczELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAk1BMQ8wDQYDVQQHEwZCb3NOb24xFzAVBgNV
BAoTDkxEQVAgVW5saW1pdGVkMRcwFQYDVQQLEw5UcnVzdGVkIFNvdXJjZTEUMBIGA1UEAxMLbGRh
cDpTZXJ2ZXIwggG3MIIBLAYHKoZIzjgEATCCAR8CgYEA/X9TgR11EilS30qcLuzk5/YRt1I870QA
wx4/gLZRJmlFXUAiUftZPY1Y+r/F9bow9subVWzXgTuAHTRv8mZgt2uZUKWkn5/oBHsQIsJPu6nX
/rfGG/g7V+fGqKYVDwT7g/bTxR7DAjVUE1oWkTL2dfOuK2HXKu/yIgMZndFIAccCFQCXYFCPFSML
zLKSuYKi64QL8Fgc9QKBgQD34aCF1ps93su8q1w2uFe5eZSvu/o66oL5VOwLPQeCZ1FZV4661FlP
5nEHEIGAtEkWcSPoTCgWE7fPCTKMyKbhPBZ6i1R8jSjgo64eK7OmdZFuo38L+iE1YvH7YnoBJDvM
pPG+qFGQiaiD3+Fa5Z8GkotmXoB7VSVkAUw7/s9JKgOBhAACgYABg/s+2bZi93niEF4mLgcb7qD9
PnMPXiTVdasmRlj9tYTD1ltYzSq15z/kPawy0+TGwD/LjVngSu7jiEKt6AY9O8ZJxo4VSyaRI38/
itPPCVnzU+sXF/swu15dbLB62kQ9yPaRBtP3nHrI1tI7VehbqGpeWGVZZKSBrvNy56wNQDALBgcq
hkjOOAQDBQADLwAwLAIUN5ZrVodfJ1E3IOPy93knhMbupOECFC0dbEm8HTXQoJHHxx+bxs/BFd1X
</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo></ds:Signature></Assertion>
```

C.4 Client Connection to PEP

```
>> java SSLClient -client Bluto -ks KEY_DIR/key_bluto

Connecting...

Sending available attributes from file Bluto.assert...

Welcome Bluto to XACML access control terminal

Please enter resource: http://www.jstor.org/
Please enter action: read
Decision: Access Permitted
Please enter resource:
```

C.5 XACML Request

```
<Request xmlns="urn:oasis:names:tc:xacml:1.0:context">
               <Subject SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject-category:
                     access-subject">
```

```
                <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:email" DataType
                   ="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name" IssueInstant="2004-05
                   -24T15:29:03.684000000-04:00" Issuer="null">
                      <AttributeValue>bluto@dartmouth.edu</AttributeValue>
                </Attribute>
                <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
                  DataType="http://www.w3.org/2001/XMLSchema#string" IssueInstant="2004-05-24
                  T19:38:31.814000000-04:00" Issuer="Dartmouth_PEP">
                         <AttributeValue>Bluto</AttributeValue>
                </Attribute>
                <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:group" DataType=
                    "http://www.w3.org/2001/XMLSchema#string" IssueInstant="2004-05-24T15:29:
                     03.684000000-04:00" Issuer="sean.smith@dartmouth.edu">
                        <AttributeValue>PKI Group</AttributeValue>
                </Attribute>
           </Subject>
           <Resource>
                <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
                   DataType="http://www.w3.org/2001/XMLSchema#string" IssueInstant="2004-05-
                   24T19:38:31.814000000-04:00" Issuer="Dartmouth_PEP">
                         <AttributeValue>http://www.jstor.org/</AttributeValue></Attribute>
           </Resource>
           <Action>
                <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id" DataType
                   ="http://www.w3.org/2001/XMLSchema#string" IssueInstant="2004-05-24T19:38:31.
                   814000000-04:00" Issuer="Dartmouth_PEP">
                         <AttributeValue>read</AttributeValue>
                </Attribute>
           </Action>
           <Environment>
                <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-date
                    Time" DataType="http://www.w3.org/2001/XMLSchema#dateTime" IssueInstant="
                    2004-05-24T19:38:31.814000000-04:00" Issuer="Dartmouth_PEP">
                          <AttributeValue>2004-05-24T19:38:31.814000000-04:00</AttributeValue>
                </Attribute>
            </Environment>
      </Request>
```

C.6 JSTOR Policy

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy
      xmlns="urn:oasis:names:tc:xacml:1.0:policy"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy
        cs-xacml-schema-policy-01.xsd"
      PolicyId="GeneratedPolicy"
      RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:ordered-permit-
      overrides">

  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">dartmouth.edu
         </AttributeValue>
          <SubjectAttributeDesignator DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"
                                      AttributeId="urn:oasis:names:tc:xacml:1.0:subject:email"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
    <Resources>
      <Resource>
```

```
      <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">http://www.jstor.org/
              </AttributeValue>
        <ResourceAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
                                     AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
      </ResourceMatch>
    </Resource>
  </Resources>
  <Actions>
    <AnyAction/>
  </Actions>
</Target>

<Rule RuleId="CommitRule" Effect="Permit">

  <Target>
    <Subjects>
      <AnySubject/>
    </Subjects>
    <Resources>
      <AnyResource/>
    </Resources>
    <Actions>
      <Action>
        <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>
          <ActionAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
                                     AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"/>
        </ActionMatch>
      </Action>
    </Actions>
  </Target>

  <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
      <SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
                                  AttributeId="urn:oasis:names:tc:xacml:1.0:subject:group"
                                  Issuer="sean.smith@dartmouth.edu"/>
    </Apply>
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">PKI Group</AttributeValue>
  </Condition>

</Rule>

<Rule RuleId="FinalRule" Effect="Deny"/>

</Policy>
```

C.7 PEP Log

```
File --> Mon May 24 19:34:35 EDT 2004.log


Begin Log...

*****************Welcome to the PEP server************************

PEP Initialized to the following settings...
PDP TYPE...................External
PDP LOCATION...............mazook.kiewit.dartmouth.edu:4444
PEP-PDP Protocol...........SAML over SSL
```

```
ASSERTION VALIDITY.........300000 milliseconds
PEP-CLIENT PROTOCOL........SSL
CLIENT AUTHENTICATION......Enabled
LOGGING....................Enabled
LOG DIRECTORY................/LOG
RETURN REQUEST.............Enabled
CACHE SETTING..............Global

Initialized on Mon May 24 19:34:36 EDT 2004
1> ........Incoming Request at Mon May 24 19:37:50 EDT 2004
1> [
[
  Version: V1
  Subject: CN=Bluto, OU=Dartmouth College, O=Delta House, L=Hanover, ST=NH, C=US
  Signature Algorithm: SHA1withDSA, OID = 1.2.840.10040.4.3

  Key:  Sun DSA Public Key
    Parameters:DSA
p:     fd7f5381 1d751229 52df4a9c 2eece4e7 f611b752 3cef4400 c31e3f80 b6512669
    455d4022 51fb593d 8d58fabf c5f5ba30 f6cb9b55 6cd7813b 801d346f f26660b7
    6b9950a5 a49f9fe8 047b1022 c24fbba9 d7feb7c6 1bf83b57 e7c6a8a6 150f04fb
    83f6d3c5 1ec30235 54135a16 9132f675 f3ae2b61 d72aeff2 2203199d d14801c7
q:     9760508f 15230bcc b292b982 a2eb840b f0581cf5
g:     f7e1a085 d69b3dde cbbcab5c 36b857b9 7994afbb fa3aea82 f9574c0b 3d078267
    5159578e bad4594f e6710710 8180b449 167123e8 4c281613 b7cf0932 8cc8a6e1
    3c167a8b 547c8d28 e0a3ae1e 2bb3a675 916ea37f 0bfa2135 62f1fb62 7a01243b
    cca4f1be a8519089 a883dfe1 5ae59f06 928b665e 807b5525 64014c3b fecf492a

  y:
    3fa5e96b d1a0e889 e80c29ba 8ef91137 53160d53 369a9231 a8ebd034 6940ed6e
    f569bc1b ef5b8752 2a105b42 9b630d76 47415b35 ae5bf90b 04897922 3e0403e9
    9824323d da4c907f 9ed9f568 da39a9b8 e38b9e21 a0f30e19 88db3e3e 5cdde557
    9c08b4d6 af2e84d6 4dce89e7 8ebd4f35 469db993 4c342161 898a9390 5e449084

  Validity: [From: Wed May 05 22:45:43 EDT 2004,
              To: Tue Aug 03 22:45:43 EDT 2004]
  Issuer: CN=Bluto, OU=Dartmouth College, O=Delta House, L=Hanover, ST=NH, C=US
  SerialNumber: [    4099a6d7]

]
  Algorithm: [SHA1withDSA]
  Signature:
0000: 30 2C 02 14 63 93 1E 3B   64 E5 E4 B1 BC 9A 00 6A  0,..c..;d......j
0010: 65 AC 69 7E 53 2E 7E BC   02 14 33 57 6F B4 45 06  e.i.S.....3Wo.E.
0020: 88 11 80 A4 E9 99 A4 27   5C C4 CE F4 19 72        .......'\....r

]
1> Assertion from client belongs to client: Valid
1> Client has been identified as:  Bluto
Client has requested the resource: http://www.jstor.org/
Client has requested the following action:  read

1> Assertions are being sent to the PDP on Mon May 24 19:38:30 EDT 2004
1> SAMLResponse from Dartmouth.College.PDPServer received at Mon May 24 19:38:36 EDT 2004
<Response xmlns="urn:oasis:names:tc:SAML:1.0:protocol" xmlns:saml="urn:oasis:names:tc:SAML
     :1.0:assertion" xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol" InResponseTo=
     "d2b5c016ab8a819f0025a72a301ee81a" IssueInstant="2004-05-24T23:38:36Z" MajorVersion="1"
      MinorVersion="1" Recipient="mazook.Kiewit.dartmouth.edu" ResponseID="e1ef39595e1f69bbe
      51ceaae408a7492"><Status><StatusCode Value="samlp:Success"></StatusCode></Status>

      <Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion" AssertionID=
               "b8421259d112beb7fbffe04e3fa232aa" IssueInstant="2004-05-24T23:38:35Z" Issuer=
               "Dartmouth.College.PDPServer" MajorVersion="1" MinorVersion="1">
          <Conditions NotBefore="2004-05-24T23:38:35Z" NotOnOrAfter="2004-05-24T23:43:35Z">
                 </Conditions>
```

```
          <XACMLDecisionStatement>
               <Subject><NameIdentifier>Unused</NameIdentifier></Subject>
               <Result xmlns="urn:oasis:names:tc:xacml:1.0:context">
                    <Decision>Permit</Decision>
                    <Status>
                      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"></StatusCode>
                    </Status>
               </Result>
               <Request xmlns="urn:oasis:names:tc:xacml:1.0:context">
               <Subject SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject-category:
                     access-subject">
                    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:email" DataType
                       ="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name" IssueInstant="2004-05
                       -24T15:29:03.684000000-04:00" Issuer="null">
                          <AttributeValue>bluto@dartmouth.edu</AttributeValue>
                    </Attribute>
                    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
                      DataType="http://www.w3.org/2001/XMLSchema#string" IssueInstant="2004-05-24
                      T19:38:31.814000000-04:00" Issuer="Dartmouth_PEP">
                             <AttributeValue>Bluto</AttributeValue>
                    </Attribute>
                    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:group" DataType=
                        "http://www.w3.org/2001/XMLSchema#string" IssueInstant="2004-05-24T15:29:
                         03.684000000-04:00" Issuer="sean.smith@dartmouth.edu">
                            <AttributeValue>PKI Group</AttributeValue>
                    </Attribute>
               </Subject>
               <Resource>
                    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
                       DataType="http://www.w3.org/2001/XMLSchema#string" IssueInstant="2004-05-
                       24T19:38:31.814000000-04:00" Issuer="Dartmouth_PEP">
                             <AttributeValue>http://www.jstor.org/</AttributeValue></Attribute>
               </Resource>
               <Action>
                    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id" DataType
                       ="http://www.w3.org/2001/XMLSchema#string" IssueInstant="2004-05-24T19:38:31.
                       814000000-04:00" Issuer="Dartmouth_PEP">
                             <AttributeValue>read</AttributeValue>
                    </Attribute>
               </Action>
               <Environment>
                    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-date
                        Time" DataType="http://www.w3.org/2001/XMLSchema#dateTime" IssueInstant="
                        2004-05-24T19:38:31.814000000-04:00" Issuer="Dartmouth_PEP">
                              <AttributeValue>2004-05-24T19:38:31.814000000-04:00</AttributeValue>
                    </Attribute>
                </Environment>
          </Request>
      </XACMLDecisionStatement>

<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:SignedInfo>
<ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"></ds:Canonicalization
     Method>
<ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"></ds:SignatureMethod>
<ds:Reference URI="#b8421259d112beb7fbffe04e3fa232aa">
<ds:Transforms>
<ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"></ds:Transform>
<ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"><ec:InclusiveNamespaces xmlns:ec
     ="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="#default code ds kind rw saml samlp
     typens"></ec:InclusiveNamespaces></ds:Transform>
</ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"></ds:DigestMethod>
<ds:DigestValue>gmxsBR7imIpzwdyw6i3qo49R1Es=</ds:DigestValue>
</ds:Reference>
```

```
</ds:SignedInfo>
<ds:SignatureValue>HdiPHqPpec9LejcT+B62BA8Y7+FTXPUHQi6XWYiyO9PxgDLrQJQNjg==</ds:SignatureValue>
<ds:KeyInfo>
<ds:X509Data>
<ds:X509Certificate>
MIIDZDCCAyICBEBkwu8wCwYHKoZIzjgEAwUAMIGXMQswCQYDVQQGEwJVUzEQMA4GA1UECBMHTW9u
dGFuYTEQMA4GA1UEBxMHTm93aGVyZTEsMCoGA1UEChMjWUFTQyAoWWVOIEFub3RoZXIgU2VjdXJp
dHkgQ29tcGFueSkxEzARBgNVBAsTClBEUCBTZXJ2ZXIxITAfBgNVBAMTGFBEUCBTZXJ2ZXIgQWRt
aW5pc3RyYXRvcjAeFw0wNDAzMjYyMzU1MjdaFw0wNDA2MjQyMzU1MjdaMIGXMQswCQYDVQQGEwJV
UzEQMA4GA1UECBMHTW9udGFuYTEQMA4GA1UEBxMHTm93aGVyZTEsMCoGA1UEChMjWUFTQyAoWWVO
IEFub3RoZXIgU2VjdXJpdHkgQ29tcGFueSkxEzARBgNVBAsTClBEUCBTZXJ2ZXIxITAfBgNVBAMT
GFBEUCBTZXJ2ZXIgQWRtaW5pc3RyYXRvcjCCAbcwggEsBgcqhkjOOAQBMIIBHwKBgQD9f1OBHXUS
KVLfSpwu7OTn9hG3UjzvRADDHj+AtlEmaUVdQCJR+1k9jVj6v8X1ujD2y5tVbNeBO4AdNG/yZmC3
a5lQpaSfn+gEexAiwk+7qdf+t8Yb+DtX58aophUPBPuD9tPFHsMCNVQTWhaRMvZ1864rYdcq7/Ii
AxmdOUgBxwIVAJdgUI8VIwvMspK5gqLrhAvwWBz1AoGBAPfhoIXWmz3ey7yrXDa4V7l5lK+7+jrq
gvlXTAs9B4JnUVlXjrrUWU/mcQcQgYCOSRZxI+hMKBYTt88JMozIpuE8FnqLVHyNKOCjrh4rs6Z1
kW6jfwv6ITVi8ftiegEkO8yk8b6oUZCJqIPf4VrlnwaSi2ZegHtVJWQBTDv+z0kqA4GEAAKBgHi8
CvZA/AOD7M7ff/G24IFX5K3acees4qH6QBZ5fayaxWMsY39sF/SKofZNLqL7ULy4dqO9JhXcvktJ
SrgCZD/pLXBTcym3IK2fBpBclDbW62Y6q2Yi7om2n4C13WG/u/78gSfEJtSYyHgx2Pnwau7SuybU
w/nAs2bAZpIxFLuqMAsGByqGSM44BAMFAAMvADAsAhR8qy/xIxvrK3339grj+YeZo46WhgIUW1th
koIOXU1hQVpBOzbEGNVRMjg=
</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo></ds:Signature></Assertion></Response>
1> Assertion signature from Dartmouth.College.PDPServer verified
1> Assertion from Dartmouth.College.PDPServer is valid according to validity period
1> Assertion certificate from Dartmouth.College.PDPServer  verified to be trusted
1> Decision: Access Permitted
1> Time: Mon May 24 19:38:37 EDT 2004
1> ***********************************
```

C.8 PDP Log

```
Log file --> Mon May 24 19:35:00 EDT 2004.log


*****************Welcome to the PDP server************************

PDP Initialized to the following settings...
Port.......................4444
POLICY DIRECTORY............/POLICY_DIR
LOGGING....................Enabled
LOG DIRECTORY................/LOG

Initialized on Mon May 24 19:35:01 EDT 2004

1> Request received on Mon May 24 19:38:31 EDT 2004
1> Request from the following certificate...
[
[
  Version: V1
  Subject: CN=PEP Server Administrator, OU=PEP Server, O=Security Inc., L=Hanover, ST=NH, C=US
  Signature Algorithm: SHA1withDSA, OID = 1.2.840.10040.4.3

  Key:  Sun DSA Public Key
    Parameters:DSA
p:     fd7f5381 1d751229 52df4a9c 2eece4e7 f611b752 3cef4400 c31e3f80 b6512669
    455d4022 51fb593d 8d58fabf c5f5ba30 f6cb9b55 6cd7813b 801d346f f26660b7
    6b9950a5 a49f9fe8 047b1022 c24fbba9 d7feb7c6 1bf83b57 e7c6a8a6 150f04fb
    83f6d3c5 1ec30235 54135a16 9132f675 f3ae2b61 d72aeff2 2203199d d14801c7
q:     9760508f 15230bcc b292b982 a2eb840b f0581cf5
```

```
g:     f7e1a085 d69b3dde cbbcab5c 36b857b9 7994afbb fa3aea82 f9574c0b 3d078267
    5159578e bad4594f e6710710 8180b449 167123e8 4c281613 b7cf0932 8cc8a6e1
    3c167a8b 547c8d28 e0a3ae1e 2bb3a675 916ea37f 0bfa2135 62f1fb62 7a01243b
    cca4f1be a8519089 a883dfe1 5ae59f06 928b665e 807b5525 64014c3b fecf492a

  y:
    6b8d1cfb c2219806 da7a39fe cf289755 21076f38 b086dfe0 2c995a70 238bd45b
    d3a7ec15 914933c6 5913b64e c55960bf 45e66bff b5148895 8cc6a7e2 2589283c
    4b43845c 99cb2855 506621dd 1242228c f853e958 fe3b5a55 d2985d21 43fb11af
    8ad8d6c3 b47a3089 2284f0f1 43d53174 930deb8d 47f94dfc ef396842 675162cf

  Validity: [From: Fri Mar 26 18:48:06 EST 2004,
              To: Thu Jun 24 19:48:06 EDT 2004]
  Issuer: CN=PEP Server Administrator, OU=PEP Server, O=Security Inc., L=Hanover, ST=NH, C=US
  SerialNumber: [    4064c136]

]
  Algorithm: [SHA1withDSA]
  Signature:
0000: 30 2C 02 14 28 06 C4 EC   18 DE 5D 12 88 38 7C 18  0,..(.....]..8..
0010: 66 B9 D9 11 9D E5 59 25   02 14 19 F4 11 78 82 0B  f.....Y\%.....x..
0020: 2F E0 C8 15 AE 33 1B 22   EB C0 CE 06 E0 D2        /....3."......

]
1> Incoming request ......

<Response xmlns="urn:oasis:names:tc:SAML:1.0:protocol" xmlns:saml="urn:oasis:names:tc:SAML:
      1.0:assertion" xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol" InResponseTo="true"
      IssueInstant="2004-05-24T23:38:32Z" MajorVersion="1" MinorVersion="1" Recipient="Dart
      mouth_PDP" ResponseID="d2b5c016ab8a819f0025a72a301ee81a"><Status><StatusCode Value="saml
      p:Success"></StatusCode></Status>

   <Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion" AssertionID="a501bb006280ae975e8
      e29a95d1c3f9c" IssueInstant="2004-05-24T23:38:31Z" Issuer="Dartmouth_PEP" MajorVersion=
      "1" MinorVersion="1">

    <Conditions NotBefore="2004-05-24T23:38:31Z" NotOnOrAfter="2004-05-24T23:43:31Z">
          </Conditions>
    <XACMLAttributeStatement>
         <Subject><NameIdentifier>Bluto</NameIdentifier></Subject>
         <XACMLAttributeSet>
            <Attribute xmlns="urn:oasis:names:tc:xacml:1.0:context" AttributeId="urn:oasis:
              names:tc:xacml:1.0:subject:subject-id" DataType="http://www.w3.org/2001/XMLSchema
              #string" IssueInstant="2004-05-24T19:38:31.814000000-04:00" Issuer="Dartmouth_PEP">
                    <AttributeValue>Bluto</AttributeValue>
            </Attribute>
            <Attribute xmlns="urn:oasis:names:tc:xacml:1.0:context" AttributeId="urn:oasis:names:tc
                :xacml:1.0:action:action-id" DataType="http://www.w3.org/2001/XMLSchema#string"
                  IssueInstant="2004-05-24T19:38:31.814000000-04:00" Issuer="Dartmouth_PEP">
                     <AttributeValue>read</AttributeValue>
            </Attribute>
            <Attribute xmlns="urn:oasis:names:tc:xacml:1.0:context" AttributeId="urn:oasis:names:tc:
                xacml:1.0:resource:resource-id" DataType="http://www.w3.org/2001/XMLSchema#string"
                IssueInstant="2004-05-24T19:38:31.814000000-04:00" Issuer="Dartmouth_PEP">
                    <AttributeValue>http://www.jstor.org/</AttributeValue>
             </Attribute>
             <Attribute xmlns="urn:oasis:names:tc:xacml:1.0:context" AttributeId="urn:oasis:names:tc:
              xacml:1.0:environment:current-dateTime" DataType="http://www.w3.org/2001/XMLSchema#date
               Time" IssueInstant="2004-05-24T19:38:31.814000000-04:00" Issuer="Dartmouth_PEP">
                 <AttributeValue>2004-05-24T19:38:31.814000000-04:00</AttributeValue>
             </Attribute>
          </XACMLAttributeSet>
   </XACMLAttributeStatement>
```

```
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:SignedInfo>
<ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"></ds:Canonical
    izationMethod>
<ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"></ds:SignatureMethod>
<ds:Reference URI="#a501bb006280ae975e8e29a95d1c3f9c">
<ds:Transforms>
<ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"></ds:Transform>
<ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"><ec:InclusiveNamespaces xmlns
      :ec="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="#default code ds kind rw saml samlp
       typens"></ec:InclusiveNamespaces></ds:Transform>
</ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"></ds:DigestMethod>
<ds:DigestValue>CYLsDeLt6Gj6JspklCSNuxz9JCk=</ds:DigestValue>
</ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>NKp5491Q5A0tp94wrzFa5cs0+pmDYOF+IrBNH/NUh34JMdATECPfQg==</ds:SignatureValue>
<ds:KeyInfo>
<ds:X509Data>
<ds:X509Certificate>
MIIDLDCCAuoCBEBkwTYwCwYHKoZIzjgEAwUAMHwxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJOSDEQ
MA4GA1UEBxMHSGFub3ZlcjEWMBQGA1UEChMNU2VjdXJpdHkgSW5jLjETMBEGA1UECxMKUEVQIFNl
cnZlcjEhMB8GA1UEAxMYUEVQIFNlcnZlciBBZG1pbmlzdHJhdG9yMB4XDTA0MDMyNjIzNDgwNloX
DTA0MDYyNDIzNDgwNlowfDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAk5IMRAwDgYDVQQHEwdIYW5v
dmVyMRYwFAYDVQQKEw1TZWN1cml0eSBJbmMuMRMwEQYDVQQLEwpQRVAgU2VydmVyMSEwHwYDVQQD
ExhQRVAgU2VydmVyIEFkbWluaXN0cmF0b3IwggG3MIIBLAYHKoZIzjgEATCCAR8CgYEA/X9TgR11
EilS3OqcLuzk5/YRt1I87OQAwx4/gLZRJmlFXUAiUftZPY1Y+r/F9bow9subVWzXgTuAHTRv8mZg
t2uZUKWkn5/oBHsQIsJPu6nX/rfGG/g7V+fGqKYVDwT7g/bTxR7DAjVUE1oWkTL2df0uK2HXKu/y
IgMZndFIAccCFQCXYFCPFSMLzLKSuYKi64QL8Fgc9QKBgQD34aCF1ps93su8q1w2uFe5eZSvu/o6
6oL5V0wLPQeCZ1FZV4661FlP5nEHEIGAtEkWcSPoTCgWE7fPCTKMyKbhPBZ6i1R8jSjgo64eK7Om
dZFuo38L+iE1YvH7YnoBJDvMpPG+qFGQiaiD3+Fa5Z8GkotmXoB7VSVkAUw7/s9JKgOBhAACgYBr
jRz7wiGYBtp6Of7PKJdVIQdvOLCG3+AsmVpwI4vUW9On7BWRSTPGWRO2TsVZYL9F5mv/tRSIlYzG
p+IliSg8SOOEXJnLKFVQZiHdEkIijPhT6Vj+O1pVOphdIUP7Ea+K2NbDtHowiSKE8PFD1TFOkw3r
jUf5TfzvOWhCZ1FizzALBgcqhkjOOAQDBQADLwAwLAIUKAbE7BjeXRKIOHwYZrnZEZ3lWSUCFBn0
EXiCCy/gyBWuMxsi68DOBuDS
</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo></ds:Signature>

</Assertion>

<Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion" AssertionID="e5e85c3a7c1f6d663877fef
    90aae3144" IssueInstant="2004-05-24T19:29:03Z" Issuer="LDAP" MajorVersion="1" MinorVersio
    n="1">
      <Conditions NotBefore="2004-05-24T19:29:03Z" NotOnOrAfter="2004-05-25T19:29:03Z">
          </Conditions>
      <XACMLAttributeStatement>
         <Subject><NameIdentifier NameQualifier="1269287">Bluto</NameIdentifier></Subject>
         <XACMLAttributeSet>
             <Attribute xmlns="urn:oasis:names:tc:xacml:1.0:context" AttributeId="urn:oasis:nam
               es:tc:xacml:1.0:subject:email" DataType="urn:oasis:names:tc:xacml:1.0:data-type:
               rfc822Name" IssueInstant="2004-05-24T15:29:03.684000000-04:00" Issuer="null">
                   <AttributeValue>bluto@dartmouth.edu</AttributeValue>
             </Attribute>
             <Attribute xmlns="urn:oasis:names:tc:xacml:1.0:context" AttributeId="urn:oasis:na
                 mes:tc:xacml:1.0:subject:group" DataType="http://www.w3.org/2001/XMLSchema#string
                 " IssueInstant="2004-05-24T15:29:03.684000000-04:00" Issuer="sean.smith@dartmouth
                 .edu">
                       <AttributeValue>PKI Group</AttributeValue>
             </Attribute>
        </XACMLAttributeSet>
     </XACMLAttributeStatement>

<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:SignedInfo>
```

<ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
      </ds:CanonicalizationMethod>
<ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"></ds:SignatureMethod>
<ds:Reference URI="#e5e85c3a7c1f6d663877fef90aae3144">
<ds:Transforms>
<ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"></ds:Transform>
<ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"><ec:InclusiveNamespaces xmlns:
         ec="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="#default code ds kind rw saml
         samlp typens"></ec:InclusiveNamespaces></ds:Transform>
</ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"></ds:DigestMethod>
<ds:DigestValue>nwSz9gHO/JzzwfaF3aqak//x1o8=</ds:DigestValue>
</ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>cbcOsqnMVYNk0uJ4oRj+iRjGfTUA8Cm/A+ij4fhlvbZ/HylpbjS43Q==</ds:SignatureValue>
<ds:KeyInfo>
<ds:X509Data>
<ds:X509Certificate>
MIIDGjCCAtgCBEB6szwwCwYHKoZIzjgEAwUAMHMxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJNQTEP
MA0GA1UEBxMGQm9zdG9uMRcwFQYDVQQKEw5MREFQIFVubGltaXRlZDEXMBUGA1UECxMOVHJ1c3Rl
ZCBTb3VyY2UxFDASBgNVBAMTC2xkYXA6U2VydmVyMB4XDTA0MDQxMjE1MTgyMFoXDTA0MDcxMTE1
MTgyMFowczELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAk1BMQ8wDQYDVQQHEwZCb3N0b24xFzAVBgNV
BAoTDkxEQVAgVW5saW1pdGVkMRcwFQYDVQQLEw5UcnVzdGVkIFNvdXJjZTEUMBIGA1UEAxMLbGRh
cDpTZXJ2ZXIwggG3MIIBLAYHKoZIzjgEATCCAR8CgYEA/X9TgR11EilS30qcLuzk5/YRt1I870QA
wx4/gLZRJmlFXUAiUftZPY1Y+r/F9bow9subVWzXgTuAHTRv8mZgt2uZUKWkn5/oBHsQIsJPu6nX
/rfGG/g7V+fGqKYVDwT7g/bTxR7DAjVUE1oWkTL2dfOuK2HXKu/yIgMZndFIAccCFQCXYFCPFSML
zLKSuYKi64QL8Fgc9QKBgQD34aCF1ps93su8q1w2uFe5eZSvu/o66oL5VOwLPQeCZ1FZV4661FlP
5nEHEIGAtEkWcSPoTCgWE7fPCTKMyKbhPBZ6i1R8jSjgo64eK7OmdZFuo38L+iE1YvH7YnoBJDvM
pPG+qFGQiaiD3+Fa5Z8GkotmXoB7VSVkAUw7/s9JKgOBhAACgYABg/s+2bZi93niEF4mLgcb7qD9
PnMPXiTVdasmRlj9tYTD1ltYzSq15z/kPawy0+TGwD/LjVngSu7jiEKt6AY9O8ZJxo4VSyaRI38/
itPPCVnzU+sXF/swu15dbLB62kQ9yPaRBtP3nHrI1tI7VehbqGpeWGVZZKSBrvNy56wNQDALBgcq
hkjOOAQDBQADLwAwLAIUN5ZrVodfJ1E3IOPy93knhMbupOECFC0dbEm8HTXQoJHHxx+bxs/BFd1X
</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo></ds:Signature></Assertion></Response>
1> Assertion signature from LDAP verified
1> Assertion from LDAP is valid according to validity period
1> PDP Assertion certificate verified to be trusted on Mon May 24 19:38:35 EDT 2004
1> Assertion from LDAP validated.
1> Assertion signature from Dartmouth_PEP verified
1> Assertion from Dartmouth_PEP is valid according to validity period
1> PDP Assertion certificate verified to be trusted on Mon May 24 19:38:35 EDT 2004
1> Assertion from Dartmouth_PEP validated.
1> PDP Decision.....

<Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion" AssertionID=
               "b8421259d112beb7fbffe04e3fa232aa" IssueInstant="2004-05-24T23:38:35Z" Issuer=
               "Dartmouth.College.PDPServer" MajorVersion="1" MinorVersion="1">
          <Conditions NotBefore="2004-05-24T23:38:35Z" NotOnOrAfter="2004-05-24T23:43:35Z">
                </Conditions>
          <XACMLDecisionStatement>
               <Subject><NameIdentifier>Unused</NameIdentifier></Subject>
               <Result xmlns="urn:oasis:names:tc:xacml:1.0:context">
                    <Decision>Permit</Decision>
                    <Status>
                      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"></StatusCode>
                    </Status>
               </Result>
               <Request xmlns="urn:oasis:names:tc:xacml:1.0:context">
               <Subject SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject-category:
                     access-subject">
                    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:email" DataType
                       ="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name" IssueInstant="2004-05

```
                    -24T15:29:03.684000000-04:00" Issuer="null">
                       <AttributeValue>bluto@dartmouth.edu</AttributeValue>
                 </Attribute>
                 <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
                   DataType="http://www.w3.org/2001/XMLSchema#string" IssueInstant="2004-05-24
                   T19:38:31.814000000-04:00" Issuer="Dartmouth_PEP">
                          <AttributeValue>Bluto</AttributeValue>
                 </Attribute>
                 <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:group" DataType=
                     "http://www.w3.org/2001/XMLSchema#string" IssueInstant="2004-05-24T15:29:
                      03.684000000-04:00" Issuer="sean.smith@dartmouth.edu">
                         <AttributeValue>PKI Group</AttributeValue>
                 </Attribute>
            </Subject>
            <Resource>
                 <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
                    DataType="http://www.w3.org/2001/XMLSchema#string" IssueInstant="2004-05-
                    24T19:38:31.814000000-04:00" Issuer="Dartmouth_PEP">
                          <AttributeValue>http://www.jstor.org/</AttributeValue></Attribute>
            </Resource>
            <Action>
                 <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id" DataType
                    ="http://www.w3.org/2001/XMLSchema#string" IssueInstant="2004-05-24T19:38:31.
                    814000000-04:00" Issuer="Dartmouth_PEP">
                          <AttributeValue>read</AttributeValue>
                 </Attribute>
            </Action>
            <Environment>
                 <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-date
                     Time" DataType="http://www.w3.org/2001/XMLSchema#dateTime" IssueInstant="
                     2004-05-24T19:38:31.814000000-04:00" Issuer="Dartmouth_PEP">
                           <AttributeValue>2004-05-24T19:38:31.814000000-04:00</AttributeValue>
                 </Attribute>
             </Environment>
        </Request>
    </XACMLDecisionStatement>

<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:SignedInfo>
<ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"></ds:Canonicalization
     Method>
<ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"></ds:SignatureMethod>
<ds:Reference URI="#b8421259d112beb7fbffe04e3fa232aa">
<ds:Transforms>
<ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"></ds:Transform>
<ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"><ec:InclusiveNamespaces xmlns:ec
     ="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="#default code ds kind rw saml samlp
     typens"></ec:InclusiveNamespaces></ds:Transform>
</ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"></ds:DigestMethod>
<ds:DigestValue>gmxsBR7imIpzwdyw6i3qo49R1Es=</ds:DigestValue>
</ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>HdiPHqPpec9LejcT+B62BA8Y7+FTXPUHQi6XWYiyO9PxgDLrQJQNjg==</ds:SignatureValue>
<ds:KeyInfo>
<ds:X509Data>
<ds:X509Certificate>
MIIDZDCCAyICBEBkwu8wCwYHKoZIzjgEAwUAMIGXMQswCQYDVQQGEwJVUzEQMA4GA1UECBMHTW9u
dGFuYTEQMA4GA1UEBxMHTm93aGVyZTEsMCoGA1UEChMjWUFTQyAoWWV0IEFub3RoZXIgU2VjdXJp
dHkgQ29tcGFueSkxEzARBgNVBAsTClBEUCBTZXJ2ZXIxITAfBgNVBAMTGFBEUCBTZXJ2ZXIgQWRt
aW5pc3RyYXRvcjAeFw0wNDAzMjYyMzU1MjdaFw0wNDA2MjQyMzU1MjdaMIGXMQswCQYDVQQGEwJV
UzEQMA4GA1UECBMHTW9udGFuYTEQMA4GA1UEBxMHTm93aGVyZTEsMCoGA1UEChMjWUFTQyAoWWV0
IEFub3RoZXIgU2VjdXJpdHkgQ29tcGFueSkxEzARBgNVBAsTClBEUCBTZXJ2ZXIxITAfBgNVBAMT
GFBEUCBTZXJ2ZXIgQWRtaW5pc3RyYXRvcjCCAbcwggEsBgcqhkjOOAQBMIIBHwKBgQD9f1OBHXUS
KVLfSpwu7OTn9hG3UjzvRADDHj+AtlEmaUVdQCJR+1k9jVj6v8X1ujD2y5tVbNeBO4AdNG/yZmC3
```

```
a5lQpaSfn+gExAiwk+7qdf+t8Yb+DtX58aophUPBPuD9tPFHsMCNVQTWhaRMvZ1864rYdcq7/Ii
AxmdOUgBxwIVAJdgUI8VIwvMspK5gqLrhAvwWBz1AoGBAPfhoIXWmz3ey7yrXDa4V7l5lK+7+jrq
gvlXTAs9B4JnUVlXjrrUWU/mcQcQgYCOSRZxI+hMKBYTt88JMozIpuE8FnqLVHyNKOCjrh4rs6Z1
kW6jfwv6ITVi8ftiegEkO8yk8b6oUZCJqIPf4VrlnwaSi2ZegHtVJWQBTDv+zOkqA4GEAAKBgHi8
CvZA/AOD7M7ff/G24IFX5K3acees4qH6QBZ5fayaxWMsY39sF/SKofZNLqL7ULy4dqO9JhXcvktJ
SrgCZD/pLXBTcym3IK2fBpBclDbW62Y6q2Yi7om2n4C13WG/u/78gSfEJtSYyHgx2Pnwau7SuybU
w/nAs2bAZpIxFLuqMAsGByqGSM44BAMFAAMvADAsAhR8qy/xIxvrK3339grj+YeZo46WhgIUW1th
koIOXU1hQVpBOzbEGNVRMjg=
</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo></ds:Signature></Assertion>


1> Decision sent on Mon May 24 19:38:36 EDT 2004
1> ********************************
```