ELG7178D
Network Security and Cryptography

Assignment #2

XACML 2.0 Policy and SAML Assertion for Submit and Retrieve Access to SomeCompany.Com

Rich Goyette
13 Nov 2007

Table Of Contents

Introduction

This document describes the solution to the second assignment in ELG7187G. The aim of this assignment was to generate a schema-valid *policy set* to govern access by **SomeCompany** employees to the resources owned by **SomeCompany**. In addition, a schema-valid SAML assertion is required for a particular individual **Liv Tucode** making a resource request.

The following section provides an overview of the transactions occurring between the various entities within the policy framework of SomeCompany.com in order to fulfill a request by Liv Tucode to submit a code module. Assumptions are noted as appropriate.

Company Organization

SomeCompany is organized as shown in Figure 1. The corporation is partitioned into four logical divisions: Research and Development, Marketing, Sales, and IT Support. Each division is a separate policy domain.

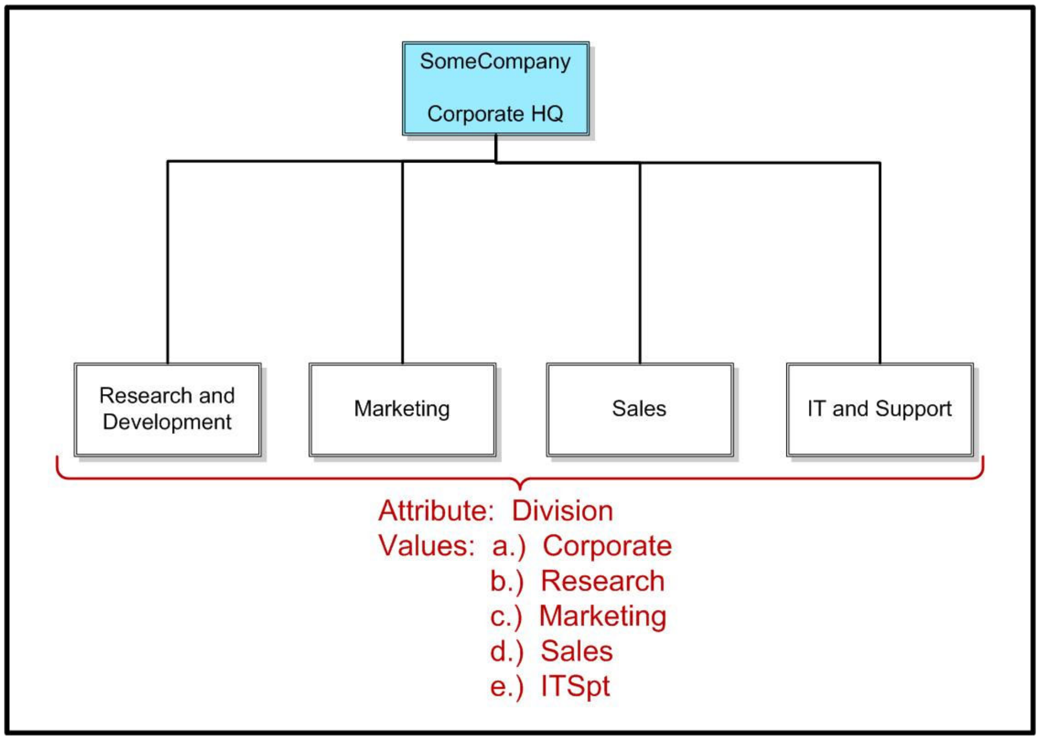

Figure 1: SomeCompany Organization

All employees of SomeCompany receive a unique RFC822 email name in the domain SomeCompnay.com. To distinguish between the divisions in which people work, each employee has a profile that includes an attribute named "Division" and whose values can take on any of Corporate, Research, Marketing, Sales, or ITSpt. An employee can be a member of only one division at a time.

SomeCompany has occasion to engage contract employees periodically to support short-term bursts of development activity or during time-to-market crunches. Because SomeCompany is sensitive to intellectual property loss, contract employees are required to do their work on the SomeCompany network. They must therefore be provisioned with account credentials and controlled access to specific resources.

Hence, there is a requirement to efficiently distinguish between different "types" of employees including regular staff (i.e. retained and administered by SomeCompany for a generally indeterminate period of time) and contract employees (i.e. retained and administered by a specialist company but providing niche services for a specific contract period). Therefore, each employee's profile contains an attribute named "Category" whose values can be Full_Time, Part_Time, Summer_Student, or Contractor_Support.

Policy Decision/Enforcement Overview

Natural Language Policy Requirement

In this problem, only recognized employees belonging to SomeCompany can access SomeCompany resources. In addition, the Research division has a local policy whereby an employee of type contractor_support can submit or retrieve his or her own code modules from a project database and only while the active period of his/her contract is valid.

Creation of the Request Context

The policy enforcement environment of SomeCompany.com is shown in Figure 2.

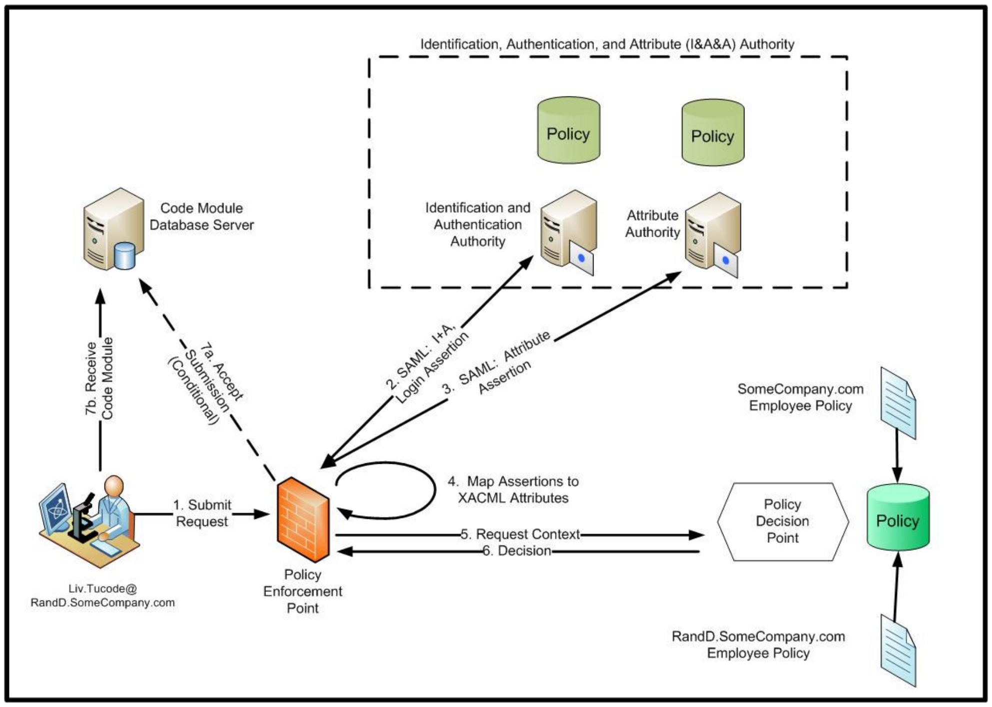
Identification, Authentication, and Attribute (I&A&A) Authority
Policy
Policy
Code Module Database Server
Identification and Authentication Authority
Attribute Authority
7b. Receive Code Module
7a. Accept Submission (Conditional)
2. SAML: I+A, Login Assertion
3. SAML: Attribute Assertion
SomeCompany.com Employee Policy
4. Map Assertions to XACML Attributes
1. Submit Request
5. Request Context
6. Decision
Policy Decision Point
Policy
Policy Enforcement Point
Liv.Tucode@ RandD.SomeCompany.com
RandD.SomeCompany.com Employee Policy

Figure 2: Policy Enforcement Environment at SomeCompany.com

In Step 1 of Figure 2, user Liv.Tucode (an employee within the Research division of SomeCompany) makes a request to submit a code module to a database server. This request is passed to a **Policy Enforcement Point (PEP)** that may or may not reside on his computing device.

The PEP will produce a *Request Context* to send to the **Policy Decision Point (PDP)** (Step 5). As indicated in [XACMLV2], the implementation of the PEP must map attribute representations in the application space to attribute representations in XACML in order to insulate the **PDP** from dependencies on the environment (the PDP deals only with XACML). Therefore, the PEP makes one or more requests to various authorities to gather information relevant to forming the *Request Context*.

In forming the *Request Context*, the PEP is interested in knowing if Liv Tucode is a valid user of the system and if he has logged in. The PEP is also concerned with collecting and validating certain attributes of Liv Tucode's profile that will support the *Request*. These include Liv's employee Category and validity dates as well as the Division in which he works.

The validity of Liv Tucode as a user of the system can be provided by an Identification and Authentication (I&A) Authority as shown in Figure 2. In SomeCompany.com, employees are issued with PKI credentials as a means of Identification and Authentication. Furthermore, these credentials are used with a SmartCard for login purposes.

User profile attributes - such as Category and Division can be obtained from an Attribute Authority.

In both cases, the I&A and Attribute Authorities can issue *Assertions* of the validity of the requested information using the Security Assertion Markup Language (SAML). A SAML assertion is a *container* with the following information:

a.) Information about the issuer including a unique identifier;
b.) Information about the subject of the assertion (including, potentially, the subject's public key);
c.) Conditions on the assertion (e.g. validity period);
d.) Advice;
e.) Attribute statements (e.g. "member of gold club," etc)
f.) Authentication statements (e.g. "so-and-so was authenticated at 1500 hrs using his smart card");

To ensure the integrity of assertion that transits between an Authority and the PEP, the assertion can be signed.

Figure 2 shows two separate requests from the PEP to the I&A and Attribute Authorities in SomeCompany (Steps 2 and 3). This was done to expose the fact that the I&A and Attribute authorities need not be collocated. However, for the purposes of this document, both the I&A and Attribute authorities are assumed to be the same entity. Therefore, a single SAML assertion will be passed back to the PEP in response to a request for authentication and attribute information.

Figure 3 shows the SAML assertion that would be returned from the I&A&A Authority in SomeCompany for Liv Tucode.

```
<saml:Assertion ID="_aaaabbb2-2327-ffff3-334f-face37656ebc"
        xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
        Version="2.0"
        IssueInstant="2007-05-31T12:00:00Z">

        <saml:Issuer>
            http://www.SomeCompany.Com
        </saml:Issuer>

      <saml:Subject>
            <!-- Identify the person or process that is the subject of this assertion
             nameid-format of emailAddress specifies RFC822 address -->
            <saml:NameID
                  Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
                  Liv.Tucode@SomeCompany.com
            </saml:NameID>
```

```
</saml:Subject>

<saml:Conditions
        <!--  This assertion must have a time limit or we very quickly run into trouble.
        We assume arbitrarily that the assertion is good for five hours-->
        NotBefore="2007-05-31T11:41:00Z"
        NotOnOrAfter="2007-05-31T16:41:00Z">
</saml:Conditions>

<!-- The PEP wants to confirm that the claimed subject is authorized and logged in.  -->
<saml:AuthnStatement
        AuthnInstant="2007-05-31T11:40:00Z"
        SessionIndex=" 34524224235">

        <!-- Here we identify the context class used to effect the authorization -->
        <saml:AuthnContext>
                <saml:AuthnContextClassRef>
                        urn:oasis:names:tc:SAML:2.0:ac:classes:SmartcardPKI
                </saml:AuthnContextClassRef>
        </saml:AuthnContext>
</saml:AuthnStatement>

<!--The PEP must check the subject against an attribute authority to ensure that the
  attributes needed for the request context are available and correct. -->
<saml:Attribute
        NameFormat=urn:oasis:names:tc:SAML:2.0:attrname-format:basic>
        Name="Category"
        <saml:AttributeValue>
                Contract_Support
        </saml:AttributeValue>
</saml:Attribute>

  <saml:Attribute
        NameFormat=urn:oasis:names:tc:SAML:2.0:attrname-format:basic>
        Name="Division"
        <saml:AttributeValue>
                Research
        </saml:AttributeValue>
</saml:Attribute>


<saml:Attribute
        NameFormat=urn:oasis:names:tc:SAML:2.0:attrname-format:basic>
        Name="start_date"
        <saml:AttributeValue>
                2007-01-01
```

```
            </saml:AttributeValue>
        </saml:Attribute>

        <saml:Attribute
            NameFormat=urn:oasis:names:tc:SAML:2.0:attrname-format:basic>
            Name="end_date"
            <saml:AttributeValue>
                2007-12-25
            </saml:AttributeValue>
        </saml:Attribute>
</saml:Assertion>
```

Figure 3: SAML Assertion For Subject Liv Tucode

In the SAML assertion of Figure 3, the subject Liv Tucode is identified by his RFC822 email name Liv.Tucode@SomeCompany.com. The assertion has a unique ID and is time-stamped. Note that, the IA&A Authority may have required a separate SAML assertion from a trusted time authority in order to create the time stamp in this assertion..

Liv Tucode is of category "contractor_support". It is assumed that every employee profile will also have attributes named "start_date" and "end_date" and that the values for contractors match their contract dates while those for full-time, regular employees are matched to the lifetime of the PKI credentials.

The PEP receives the SAML assertions and, optionally, verifies their integrity (the SAML assertion in Figure 3 does not include a signature over the assertion). It must then map the SAML assertion attributes into XACLM attributes as described in Section 2 of [SAMLPRO] to fill out the Subject, Resource, Environment, and Action elements of the *Request Context*.

In the current example, Liv Tucode would like to perform a submit action on a code file in the database. However, XACML provides native mechanisms to access *content* only when the resource being requested is an XML document. Here, contractors are inserting and removing code modules from a database.

To simplify the analysis, we assume that there exists, for each code module stored in the database, an XML metadata descriptor record (metafile). We further assume some mechanism for detecting submission or retrieval of these metafiles with the corresponding actions being taken on the actual modules within the database. Finally, we assume that the only two *actions* available for code segments are submit and retrieve actions (such as in a check-out cell). Entities retrieving an object have full access to change and modify all aspects of that object and it's metadata (this may be a bit unrealistic from a strong security point of view).

An instance of a partial XML metadata record (mod_record) is shown in Figure 4. Note that the schema currently allows only one *owner* of the object. It *should* be possible for multiple contractors to share ownership of a code module. However, for the purposes of this assignment, it is assumed that only one owner exists.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--  xmlns is the xml definition of a namespace shortform:
      xmlns:namespace-prefix="namespaceURI"
      xmlsn:xsi is a reference to an XML schema definitions.
      xsi:schemaLocation is a URL to the file containing the definition -->

<mod_record xmlns="code_repository:schemas:mod_record"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.SomeCompany.com/mod_record.xsd">

      <file_details>
            <module_ID>13579</module_ID>
            <owner_ID>LivTucode@RandD.SomeCompany.com</owner_ID>
            <last_modified>2007-02-04</last_modified>
            <last_accessed>2007-02-07</last_accessed>
      </file_details>

      <dependencies>
      < ....etc

</mod_record>
```

Figure 4: XML Metadata for Code Modules

With the resource required and the action on that resource defined, the PEP is now able to construct a *Decision Request* to the PDP by providing a *Request Context*. This is sent to the PDP in Step 5. A *Request Context* for the PEP that services Liv Tucode's submission is shown in Figure 5.

```
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="urn:oasis:names:tc:xacm:2.0:context:schema:os"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:context:schema:os
http://docs.oasis1020open.org/xacml/access_control-xacml-2.0-context-schema-os.xsd">

<Subject>
        <Attribute
                AttributeId="urn:oasis:names:tc:xacml:2.0:subject-category"
                        DataType="http://www.w3.org/2001/XMLSchema#anyURI">

                <AttributeValue>
                        urn:oasis:names:tc:xacml:1.0:subject-category:access subject
                </AttributeValue>
        </Attribute>

        <Attribute
                AttributeId="urn:oasis:names:tc:xacml:2.0:subject:subject-id"
                        DataType="urn:oasis:names:tc:xacml:2.0:data-type:rfc822Name">
                <AttributeValue>
                        Liv.Tucode@SomeCompany.com
                </AttributeValue>
        </Attribute>

        <Attribute
        AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:EmployeeCategory"
                        DataType="http://www.w3.org/2001/XMLSchema#string"
                        Issuer="SomeCompany.com">
                <AttributeValue>
                        contract_support
                </AttributeValue>
        </Attribute>

<Attribute
        AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:EmployeeDivision"
                        DataType="http://www.w3.org/2001/XMLSchema#string"
                        Issuer="SomeCompany.com">
                <AttributeValue>
```

```
                        Research
```

```
        </Attribute>

        <Attribute
                    AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:start_date"
                    DataType="http://www.w3.org/2001/XMLSchema#date"
                    Issuer="SomeCompany.com">
            <AttributeValue>
                    2007-01-01
            </AttributeValue>
        </Attribute>
        <Attribute
                    AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:end_date"
                    DataType="http://www.w3.org/2001/XMLSchema#date"
                    Issuer="SomeCompany.com">
            <AttributeValue>
                    2007-12-25
            </AttributeValue>
        </Attribute>
    </Subject>

    <Resource>
    <!-- Select a record to retrieve -->
        <Attribute AttributeId="urn:oasis:names:tc:xacml:2.0:resource:resourceid"
            DataType="http://www.w3.org/2001/XMLSchema#string">

            <AttributeValue>
                    http://www.SomeCompany.com/mod2334A.xml
            </AttributeValue
        </Attribute>
    </Resource>

    <Action>
         <Attribute AttributeId="urn:oasis:names:tc:xacml:2.0:action:action-id"
            DataType="http://www.w3.org/2001/XMLSchema#string">

            <AttributeValue>
                    submit
            </AttributeValue>
        </Attribute>
    </Action>
```

Figure 5: Request Context

Note that all of the appropriate values from the SAML assertions gathered by the PEP have been mapped into the request. This request is sent to the PDP (it is not yet clear how the integrity of this transmission is guaranteed unless this service is pushed off to the transport or other protocol).

Policy Decision

The Policy Decision Point must make a decision on the access request based on the Request Context provided to it by the PEP as well as one or more policy repositories to which it has access.

Because the Research and Development division operates as an independent unit, it is capable of generating it's own policies. However, the actions of all employees of SomeCompany are governed by corporate policy as well. Therefore, the Request Context will likely trigger a Policy Set within the PDP's policy database. The term "trigger" here implies that the one or more of the subject, action, or resource elements in the Request Context is identified as a "Target" element in the policy set. These targets allow for more efficient computation of decisions since a policy and its rules need not be evaluated if the targets in the request are the same as the targets in the policy.

In the current example, the policy set in Figure 6 (on page 11) triggers on *any* access to *any* resource by *any* subject. This is counter-intuitive to the principle of composing targets for efficient search since the target in the policy set is everything (guaranteeing that the policy set will be evaluated on every request). However, the corporate policy was to allow all access to all resource by any valid employee of SomeCompany so the policy set shouldn't have a more restrictive target.

This policy set could be considered an umbrella that is used to ensure that all of the applicable divisional policies are evaluated. That is, policies created locally by division may be held locally and only linked into the policy set (as in Figure 6) which is held at the PDP. At evaluation time, executing this policy set forces the PDP to fetch divisional policies thereby guaranteeing freshness.

```
<?xml version="1.0" encoding="UTF-8"?>
<PolicySet
        xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
        http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-os.xsd"
        PolicySetId="urn:oasis:names:tc:xacml:2.0:example:policysetid:SomeCompnayComboPolicy"
        PolicyCombiningAlgId="urn:oasis:names:tc:xacml:2.0:policy-combining-algorithm:deny-overrides">

        <Description>
                This policy set combines all divisional policies concerning access to any SomeCompnay
                resource.
        </Description>

        <!--For any access on any resource by any subject -->
        <Target/>
        <!--Include both of the policies we have created thus far... (by reference only) -->
        <PolicyIdReference>
                urn:oasis:names:tc:xacml:2.0:example:policyid:RandDPolicyforCodeModules
        </PolicyIdReference>

        <PolicyIdReference>
                urn:oasis:names:tc:xacml:2.0:example:policyid:SomeCompanyResourceAccessPolicy
        </PolicyIdReference>
</PolicySet>
```

Figure 6: Policy Set for SomeCompany

The policy set in Figure 6 invokes the *RandDPolicyforCodeModules* policy (which belongs to the Research division) and the S*omeCompanyResourceAccessPolicy* which is a corporate policy for all valid users within the SomeCompany.com namespace. The policy combining algorithm specifies that any DENY result by any rule invoked by the group of policies contained in the set will cause a DENY response to the request.

Attachment A shows both the *SomeCompanyResourceAccessPolicy* and *RandDPolicyCodeModules* policies. The target for *SomeCompanyResourceAccess-Policy* is any action on any resource by any subject (<Target/>). The single rule attached to this policy will PERMIT if the RFC822 email name assigned to the subject contains the SomeCompany.com namespace. This implies that the user is a valid entity within SomeCompany and is therefore granted access on all resources.

The *RandDPolicyCodeModules* is slightly more complex. The approach in this policy is summarized in natural language pseudocode in Figure 7 (on page 12).

```
<Policy>
    Target:
    Subject:  match anyone from R+D
    Action:  performing any action
    Resource:  on any resource

    Rule1
        Effect:  Permit
        Target:
            Subject:  match any contractor
            Action:  retrieve and submit
            Resource:  on any resource
            Condition:
                subject is owner of the requested resource
                AND
                current date is greater than contract start date
                AND
                current date is less than contract end date
</Policy>
```

Figure 7: Natural Language Pseudocode for *RandDPolicyCodeModules*

A <SubjectMatch> on the attributes provided as part of the <Subject> (in the *Request Context*) is first performed in order to apply this policy against anyone having a Division attribute of "Research". If that is the case, then the policy applies for any action on any resource.

A single rule is then defined for any contractor (determined by a match from the EmployeeCategory attribute) with valid contract start and end dates and who is making a *submit* or *retrieve* request on a resource that he or she owns. To check the ownership, the contractor's RFC822 email identity must match the file owner identity. The validity of the start and end dates are computed using variable definitions that evaluate to boolean values. Each variable compares the start or end dates extracted from the *Request Context* to the current date (using **less-than** or **greater-than** functions). In the condition portion of the rule, these boolean variables are ANDed and result in a PERMIT if both are TRUE (in addition to the requestor being the owner as well).

The allowable actions for the requestor are *retrieve* and *submit* and these are compared to the action request submitted as part of the *Request Context*.

Decision Response

To complete the interaction, the PDP, after evaluating the appropriate policies against the request context, submits a PERMIT/DENY response to the PEP (Step 6 of Figure 2). The PEP then conditionally provides access to the requested resource in Step 7a and 7b.

References

[XACMLV2]	OASIS eXtensible Access Control Markup Language (XACML) Version 2.0, 1 Feb 2005.
[SAMLPRO]	Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) Version 2.0, 15 March 2005.
[SAMLBAS]	E. Maler, "SAML V2.0 Basics," Presentation Slides, May 2005, accessed at www.oasis-open.org/committees/download.php/12958/**SAML**V2.0-**basics**.pdf.
[SAMLXACML]	H. Lockhart, "SAML 2.0 and Related Work in XACML and WS-Security," Presentation Slides (undated), accessed at http://www.oasis-open.org/committees/download.php/18104/SAML-Overview.pdf

The following links were referenced to develop sufficient background in XML and ?? in order to complete the assignment:

http://www.w3schools.com/xml/xml_syntax.asp
http://www.w3schools.com/xml/xml_elements.asp
http://www.w3schools.com/xml/xml_attributes.asp
http://www.w3schools.com/xml/xml_dtd.asp
http://www.w3schools.com/xml/xml_namespaces.asp
http://www.w3schools.com/schema/default.asp
http://www.w3schools.com/schema/schema_intro.asp
http://www.w3schools.com/schema/schema_howto.asp
http://www.w3schools.com/schema/schema_schema.asp
http://www.w3schools.com/schema/schema_simple.asp
http://www.w3schools.com/schema/schema_simple_attributes.asp
http://www.w3schools.com/schema/schema_complex_any.asp
http://www.w3schools.com/schema/schema_complex_anyattribute.asp
http://www.w3schools.com/schema/schema_example.asp
http://www.w3schools.com/xpath/xpath_syntax.asp
http://www.w3schools.com/xpath/xpath_nodes.asp
http://www.w3schools.com/xpath/xpath_intro.asp
http://www.oasis-open.org/committees/security/faq.php
http://identitymeme.org/doc/draft-hodges-learning-saml-00.html

SomeCompany Corporate Access Policy

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy
    <!-- The default namespace pointing to the xacml policy schema -->
    xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    <!-- Define a namespace shortcut to point to the schema for the modification record metadata -->
    xmlns:mr="http://www.SomeCompany.com/schemas/mod_record.xsd"

    PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:SomeCompanyResourceAccessPolicy"

    RuleCombiningAlgId="identifier:rule-combining-algorithm:deny-overrides">

    <Description>
        This policy restricts access to all SomeCompany resources to SomeCompany employees only
    </Description>

    <!--  The form of the target specification below means apply to all.... -->
    <Target/>
    <Rule
        RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:SomeCompanyWideAccess"
        Effect="Permit">

        <Description>
            Any authorized user within SomeCompany is allowed to perform any action on any resource.
        </Description>

        <Target>
            <Subjects>
                <Subject>
```

```
                        <SubjectMatch
                            MatchId="urn:oasis:names:tc:xacml:2.0:function:rfc822Name-match">
                            <AttributeValue
                                DataType="http://www.w3.org/2001/XMLSchema#string">
                            </AttributeValue>

                            <SubjectAttributeDesignator
                                AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
                                DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"/>
                        </SubjectMatch>
                    </Subject>
                </Subjects>
            </Target>
        </Rule>
</Policy>
```

<u>Research Code Module Access Policy</u>

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy
    <!-- The default namespace pointing to the xacml policy schema -->
    xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    <!-- Define a namespace shortcut to point to the schema for the modification record metadata -->
    xmlns:mr="http://www.SomeCompany.com/schemas/mod_record.xsd"

    PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:RandDPolicyforCodeModules"

    RuleCombiningAlgId="identifier:rule-combining-algorithm:deny-overrides">

    <Description>
    This policy allows contracted support employees within the R+D department to submit or retrieve code modules from
    a central store.
    </Description>

    <Target>
        <Subjects>
            <!--This policy only affects personnel employed in Research and Development>
            <Subject>
                <SubjectMatch MatchId="urn:oasis:names:tc:xacml:2.0:function:string-equal">
                    <AttributeValue
                        DataType="http://www.w3.org/2001/XMLSchema#string">
                        Research
                    </AttributeValue>

                    <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:Division"
                        DataType="http://www.w3.org/2001/XMLSchema#string"/>
                        </SubjectMatch>
                </Subject>
        </Subjects>


        <Resources>
                <AnyResource/>
        </Resources>

        <Actions>
                <AnyAction/>
        </Actions>

</Target>

<!--The following two variable definitions evaluate the current date against the start and end dates that should have been
provided as part of the request context.  The current date must exist between both the start and end dates.  -->

<!--This variable definition compares the current date with the start date>
<VariableDefinition VariableId="12345">
        <Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:date-greater-than-or-equal">
                <Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:date-one-and-only">
                        <EnvironmentAttributeDesignator
                                AttributeId= "urn:oasis:names:tc:xacml:2.0:environment:current-date"
                                DataType="http://www.w3.org/2001/XMLSchema#date"/>
                </Apply>

                <Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:date-one-and-only">
                        <SubjectAttributeDesignator
                                AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:start_date"
                                DataType="http://www.w3.org/2001/XMLSchema#date"/>
                </Apply>
        </Apply>
</VariableDefinition>

<VariableDefinition VariableId="67890">
        <Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:date-less-than-or-equal">
                <Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:date-one-and-only">
                        <EnvironmentAttributeDesignator
                                AttributeId= "urn:oasis:names:tc:xacml:2.0:environment:current-date"
                                DataType="http://www.w3.org/2001/XMLSchema#date"/>
                </Apply>

                <Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:date-one-and-only">
                        <SubjectAttributeDesignator
                                AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:end_date"
                                DataType="http://www.w3.org/2001/XMLSchema#date"/>
                </Apply>
        </Apply>
</VariableDefinition>

<Rule
        RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:CodeModAccess"
        Effect="Permit">

        <Description>
        Any contractor within R+D is allowed to submit and retrieve code modules anywhere within the R+D.SomeCompnay.com
        namespace (to allow freedom of movement for the database guys).
        </Description>

        <Target>
                <Subjects>
        <!--We have a match on a person from R+D.  Is he/she a contractor?>
        <Subject>
                <SubjectMatch MatchId="urn:oasis:names:tc:xacml:2.0:function:string-equal">
                        <AttributeValue
                                DataType="http://www.w3.org/2001/XMLSchema#string">
                                contract_support
                        </AttributeValue>

                        <SubjectAttributeDesignator
                        AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:EmployeeCategory"
                                DataType="http://www.w3.org/2001/XMLSchema#string"/>
                        </SubjectMatch>
        </Subject>
</Subjects>

<Resources>
        <!--Is the thing he/she requested in the correct namespace?>
        <Resource>
                <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                                http://www.SomeCompany.com/
                        </AttributeValue>

                        <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-
                        namespace"
                                DataType="http://www.w3.org/2001/XMLSchema#string"/>
                </ResourceMatch>
        </Resource>
</Resources>

<Actions>
```

```
        <Action>
            <ActionMatch
                MatchId="urn:oasis:names:tc:xacml:2.0:function:string-equal">
                <AttributeValue
                    DataType="http://www.w3.org/2001/XMLSchema#string">
                        submit
                </AttributeValue>
                <ActionAttributeDesignator
                    AttributeId="urn:oasis:names:tc:xacml:2.0:action:action-id"
                        DataType="http://www.w3.org/2001/XMLSchema#string"/>
            </ActionMatch>
        </Action>

        <Action>
            <ActionMatch
                MatchId="urn:oasis:names:tc:xacml:2.0:function:string-equal">
                <AttributeValue
                    DataType="http://www.w3.org/2001/XMLSchema#string">
                        retrieve
                </AttributeValue>
                <ActionAttributeDesignator
                    AttributeId="urn:oasis:names:tc:xacml:2.0:action:action-id"
                        DataType="http://www.w3.org/2001/XMLSchema#string"/>
            </ActionMatch>
        </Action>
    </Actions>
</Target>

<Condition>
    <Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:and">
        <!-- The two conditions levied against the start and end dates.  Both must evaluate TRUE >
```

```
                <VariableReference VariableId="12345"/>
                <VariableReference VariableId="67890"/>
                <!-- Now we must verify that the subject making the request is the owner
                    We compare the rfc822 formated ID pulled from the subject attributes to the rfc822 formated ID
                    that forms part of the metadata for the requested code segment-->
                <Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:string-equal">
                    <!-- Select the subject identity ( rfc822 format.) sent as part of the request context.
                        There must only be one of these in the request context or this will fail -->
                    <Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:string-one-and-only">
                        <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
                    </Apply>
                    <!-- Now select the owner identity (rfc822 format) from the requested xml metadata file.
                        This selection uses XPath to pull the owner id out of the requested file.  Again, there should only be
                        one owner ID on the metafile.  Alternatively, if we wanted to have more than one owner on a file,
                        we would select a bag of values in the owner_ID and perform a Match operation at this point.>
                    <Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:string-one-and-only">
                        <AttributeSelector RequestContextPath="//xacml-context:Resource/xacml-context:ResourceContent
                            /mr:mod_record/mr:file_details/mr:owner_ID/text()"
                            DataType="urn:oasis:names:tc:xacml:2.0:data-type:rfc822Name"/>
                    </Apply>
                </Apply>
            </Apply>
        </Condition>
    </Rule>
</Policy>
```