A Heterogeneous Network Access Service based on PERMIS and SAML

Gabriel López[1], Óscar Cánovas[1], Antonio F. Gómez-Skarmeta[1], Sassa Otenko[2] and David W. Chadwick[2]

[1] University of Murcia, Spain
[2] University of Kent, United Kingdom
gabilm@dif.um.es, ocanovas@ditec.um.es, skarmeta@dif.um.es, o.otenko@kent.ac.uk, d.w.chadwick@kent.ac.uk

Abstract The expansion of inter-organizational scenarios based on different authorization schemes involves the development of integration solutions allowing different authorization domains to share, in some way, protected resources. This paper analyzes different emerging technologies. On the one hand, we have two XML-based standards, the SAML standard, which is being widely accepted as a language to express and exchange authorization data, and the XACML standard, which constitutes a promising framework for access control policies. On the other hand, PERMIS is a trust management system for X.509 attribute certificates and includes a powerful authorization decision engine governed by the PERMIS XML policy. This paper presents a sample scenario where domains using these technologies can be integrated allowing, for example, the use of attribute certificates in a SAML environment and the utilization of the PERMIS authorization engine to decide about the disclosure or concealment of attributes. In order to design this scenario we have based our work on a Credential Conversion Service (CCS) which is able to convert ACs into SAML attributes, and a User Attribute Manager (UAM) which controls the disclosure of credentials. These modules are governed by policies defining the conversion process (the Conversion Policy) and the disclosure of attributes (the Disclosure Policy).

1 Introduction and Rationale

Nowadays, authorization systems are more and more complex. They span domains of administration, depend on many different authentication sources, and the management of permissions and policies can be as complex as the system itself. Worse still, while there are many standards defining authentication mechanisms, the standards for authorization systems are less widely adopted and accepted, and tend to work only within homogeneous systems.

Today we often experience inconvenience and inefficiency like this: a professor of *University A* visiting *University B* is not allowed to use the latter's network even when there is an existing collaboration by means a research project between the two institutions. The main reason might be the use of different formats

for representing the authorization information (credentials, policies, requests, etc.), and this could occur despite an existing service level agreement (SLA), a high level collaboration, between the two universities. Therefore, heterogeneity restricts the development of standard management tools and toolkits that serve common policy needs.

Authorization in a distributed system often requires cooperation among separate and autonomous administrative domains. Maintaining a consistent authorization strategy requires each system to maintain at least some knowledge of its potential collaboration with other domains. Therefore, any authorization decision that spans several authorization domains requires each party to be able to produce, accept and interpret authorization information from a group of potentially heterogeneous peers. This property can be achieved by establishing an agreement on protocols, syntax and semantics of shared pieces of authorization data to be exchanged.

In this paper we present a case study where we demonstrate how a PERMIS [6] based administrative domain can be integrated with a network access service based on SAML [13] attributes and XACML [9] policies. Our aim is to provide a feasible scenario allowing the PERMIS users to make use of the foreign network implementing a network access control mechanism based on the AAA (Authentication, Authorization, Accounting) architecture and SAML attributes [12], hereafter the NAS-SAML domain. This integration, which will be based on existing proposals like the Credential Conversion Service [7], illustrates that we can build an interdomain environment from separate and autonomous domains, which constitutes a valuable step towards the required interoperability between multiple existing authorization environments.

As we will show, while Role Based Access Control (RBAC) is an increasingly important component in distributed systems, it is one that is often hard to support in heterogeneous environments. In our proposal, this leads to the definition of loosely coupled multidomain environments, where a predefined set of role mappings to mediate interdomain accesses is defined. This approach requires the constituent systems to indicate the level of sharing they want to allow and to establish a consistent set of mediation rules for interdomain accesses. As we stated before, SAML is one example of a protocol that provides a framework for secure assertion of authorization data across domains, and it will be used in our proposal since it constitutes a key element of the Credential Conversion Service.

The main idea behind this paper can be summarized as follows. A PERMIS user willing to make use of the NAS-SAML domain has to demonstrate that he has gained the required X.509 attribute certificates. Those certificates should be provided by the PERMIS domain, and they must be translated into SAML credentials before processing the network access request. Therefore, from the architectural point of view, we have to define the entities (pertaining to the home and target domains) that will be involved in the integration process. From an operational point of view, we have to establish the way the attribute certificates will be disclosed and exchanged, and how they will be converted into equivalent

SAML attributes. As we will see, several design alternatives (push and pull based) are provided.

One central issue that our design addresses is the management of attributes and the circumstances under which a PERMIS attribute authority should disclose the user's attributes to another site. We define some simple rules about when and to whom the attributes may be revealed. Although this idea has been widely explored in the literature [15,3], our approach tries to minimize the impact of adding an overloading privacy management system to the PERMIS home domain. As we show, that privacy system will be built using the PERMIS technology itself, adapting the PERMIS policy to produce a disclosure policy that can be enforced using the PERMIS ADF (Authorization Decision Function). In this way, we make our infrastructure resilient in defence against security attacks such as data-mining, that is, the unauthorized gathering of information for improper use. Consequently, we designed a system that allows the PERMIS domain to have full control over the private information being disclosed. That is, administrators are given the opportunity to decide the kind of information to share with the foreign network provider.

Finally, we also present a simple conversion policy based on XACML that will be used to translate X.509 attribute certificates into SAML Attribute Statements. That conversion policy will be managed at the destination site (the NAS-SAML domain) and will be expressed in terms of the object identifiers related to the attributes, attribute values, SAML attribute designators, etc. XACML constitutes an appropriate tool to express these kinds of policies, as we will show in the following sections.

The rest of this paper is structured as follows. Section 2 provides an overview of the two different authorization scenarios involved in this paper, the PERMIS project and the SAML-based network access service. Section 3 describes the architectural elements, the disclosure policy and the conversion policy. Then, Section 4 presents the two different design alternatives based on pull and push models. Next, Section 5 includes the related work that informed our research. Finally, we conclude the paper with our remarks and some future directions.

2 Authorization systems

This section provides an overview of the two different authorization scenarios that have been integrated in this paper.

2.1 SAML-based network access service

In [12] we present a network access control approach based on X.509 identity certificates and authorization attributes, which addresses some of the challenges derived from the integration of existing authentication systems with a flexible, scalable and manageable authorization system. The proposal is based on the SAML and the XACML standards, which will be used for expressing access control policies based on attributes, authorization statements, and authorization

protocols. Authorization is mainly based on the definition of access control policies including the sets of users pertaining to different subject domains which will be able to be assigned to different roles in order to gain access to the network of a service provider, under specific circumstances. The starting point is a network scenario based on the 802.1X standard [14] and the AAA (Authentication, Authorization and Accounting) architecture [8], where we centralize all the operations related to authentication, authorization, and accounting.

The system operates as follows. Every end user belongs to a home domain, where he was given a set of attributes stating the roles he plays. When the end user requests a network connection in a particular domain by means of a 802.1X connection, the request is obtained by the AAA server, and it makes a query to obtain the attributes linked to the user from an authority responsible for managing them. Finally, the AAA server sends an authorization query to a policy decision point, and that element provides an answer indicating whether the attributes satisfy the resource access policy. Furthermore, that policy can also establish the set of obligations derived from the decision, for example some QoS properties, security options, etc. This general scheme works both in single and inter-domain scenarios, and using both push an pull based communications.

2.2 PERMIS project

PERMIS [6] is a trust management system. It uses X.509 Attribute Certificates to specify subject attributes such as roles and permissions, and defines a hierarchical role based access control (RBAC) policy language in terms of those roles and permissions. The PERMIS policy specifies who is to be granted what type of action on which targets, and under what conditions. Policy based authorization on the other hand allows the domain administrator (the SOA) to specify the policy for the whole domain, and all targets will then be controlled by the same set of rules. The policy is expressed in XML and comprises the following components:

- SubjectPolicy: this specifies the subject domains, that is only users from a subject domain may be authorized to access resources covered by the policy.
- RoleHierarchyPolicy: this specifies the different roles and their hierarchical relationships to each other.
- SOAPolicy: this specifies which SOAs are trusted to allocate roles.
- RoleAssignmentPolicy: this specifies which roles may be allocated to which subjects by which SOAs.
- TargetPolicy: this specifies the target domains covered by this policy. Each domain is specified as an LDAP subtree or using URIs (Uniform Resource Identifier).
- ActionPolicy: this specifies the actions (or methods) supported by the targets, along with the parameters that should be passed with each action.
- TargetAccessPolicy: this specifies which roles have permission to perform which actions on which targets, and under which conditions. Conditions are specified using Boolean logic and might contain constraints involving strings, integers, dates, or boolean expressions.

On the other hand, the privilege verification subsystem is responsible for authenticating and authorizing the remote user and providing access to the target. The primary component is the application gateway. The functionality of this is split into two components: an application-specific component termed the Access Control Enforcement Function (AEF), and an application-independent component termed the Access Control Decision Function (ADF). An application programmable interface (APIs) between the AEF and ADF has been defined based on the AZN API [10].

3 Architectural elements and Policies

The integration of different authorization scenarios involves the definition of new components which act as a bridge between the participating domains, hiding from the rest of the components the knowledge of different authorization mechanisms. Those new components must respect the already defined systems and their components, and they should be able to interact in the most transparent way. Beside these components, it is necessary to define the policies used for the disclosure and conversion processes. This section describes the components, their functionality, and gives an overview of the needed policies.

The next figure shows an application scenario where two domains, using different authorization mechanisms, exchange credentials to deny or grant access to the required services.

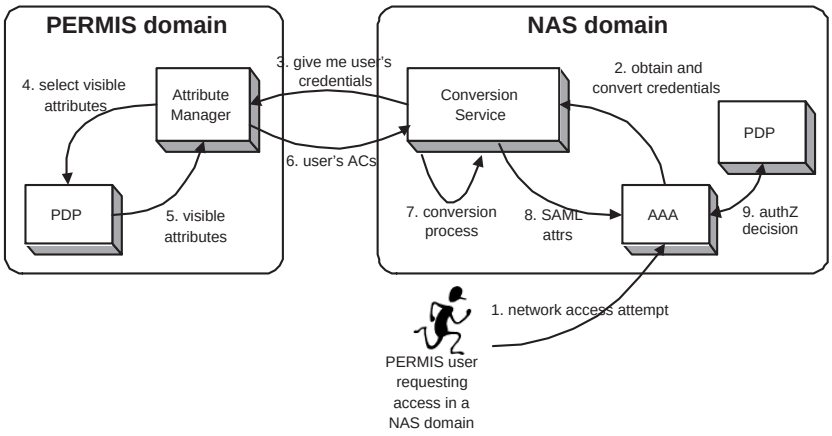

Figure 1. Credential Conversion Scenario

3.1 New components

Regarding the scenario described in Figure 1, several issues must be addressed. On the one hand, the user's home domain needs a module able to receive creden-

tial requests from an external domain, and to decide which of the user's attributes must be revealed. For example, in the proposed scenario, the PERMIS domain needs a module able to manage the ACs requests from the NAS-SAML scenario. On the other hand, the NAS-SAML domain needs a component responsible for recovering from an external domain the user attributes, which are represented in a source format (for example X.509 ACs), and for translating them into internal credentials, represented in a target format, in this case SAML.

Those modules are the UAM (User Attribute Manager), used to deal with the attribute requests received from an external domain, and the CCS (Credential Conversion Service), which is in charge of translating the authorization credentials.

User Attribute Manager (UAM) One issue in distributed systems that serve users from multiple communities is determining which organization a particular user is from and hence the organization whose attribute authority can provide attributes regarding the user. This is often referred to as the *Where are you from?* problem. Although this could be implemented placing a pointer to the attribute authority in the user's X.509 identity certificate, this solution requires cooperation of the CA issuing the user's identity credentials, which may not always be available and also binds attribute information to the user's identity credential, which may raise problems if the lifetimes of these two elements differ. Our initial system implementation either assumes fixed UAM locations dependent on the requester or discovery via an information service query to a trusted source.

In the proposed scenario, the UAM is a module able to understand queries expressed in SAML, and able to create authorization responses in that format. Moreover, this module should return to the NAS-SAML domain only the visible credentials specified by the disclosure policy. The UAM module is defined with this intention, but its particular behavior will depend on the communication model.

When the pull model is used, the UAM receives attribute queries from the target domain, specifically from the CCS. The UAM obtains the user's attributes, for example, from an internal repository, and asks the local PDP (Policy Decision Point) about the visibility of those attributes. This module enforces a disclosure policy establishing which attributes will be revealed for that domain. Once the decision is obtained, the UAM returns a response message to the CCS containing the user's attributes in source format, in this case, X.509 ACs.

In the push model, the UAM receives the attribute query directly from the end user. Then, the UAM returns the allowed attributes, following the same process as described before, to the end user. Once the user has validated the attributes and, depending on the communication between entities, he may request the UAM to forward the conversion query to the appropriate CCS module, or he may present these attributes directly to the target domain. Different design alternatives are described in Section 4.

Credential Conversion Service (CCS) The CCS [7] integrates external authorization schemes (non SAML-based) into authorization scenarios which make use of SAML as the main language for assertions. Our starting point is an end user requesting access to a resource secured in a SAML environment. We do not want the user authorities pertaining to non-SAML domains to issue SAML assertions, since they were not designed to perform that task. In fact, what we need is a service able to translate the external credentials into SAML assertions. CCS defines the different architectural entities involved in that process and their relationships. Moreover, it extends some standard SAML elements, such as SAML assertions and queries, to provide the needed syntax and semantics.

The CCS module is located in the NAS-SAML domain and it receives attribute conversion queries related to a foreign user.

In a pull model, where the user's attributes must be obtained by the target AAA server, it asks the CCS for those attributes, and the conversion process will be responsible for obtaining and translating them into the internal format. Following the described scenario, when the CCS receives an attribute query from the local AAA server, it has to discover the user's home domain, and the exact location of the UAM module. Then, it forwards the query to the UAM module, and waits for the user's attributes in a source format. When the CCS gets these attributes it has to use the Conversion Policy rules, which define how to translate the external credentials into SAML attributes.

In a push model, where the user requests his authorization attributes before accessing the target network, the CCS may receive the conversion query directly from the UAM, or from its local AAA server, according to the push model alternative selected. That is, when the user needs the authorization credentials to get access to a target domain, first he asks the home domain for his attributes. Then, attributes are forwarded to the CCS and hereafter the process is very similar to the one described above.

These modules allow the interaction between domains based on different authorization systems, and its corresponding behavior is completely based on two integration policies. The next section describes those policies and gives the guidelines about how they can be expressed.

3.2 Integration policies

In order to define how these new components work, it is necessary to introduce the policies involved. The UAM module needs a policy specifying which attributes can be revealed to which target domains, for example, depending on the level of trust agreed with that domain. This policy is named the *Disclosure Policy*. On the other hand, the CCS needs a policy describing how attributes from the user's home domain must be mapped into internal attributes, to be used next by the PDP to obtain an authorization decision. This policy is defined as the *Conversion Policy*.

Disclosure Policy. When two or more domains are involved in a trust relationship, where users from one domain can request access to resources in the others domains, it is necessary to define which user's attributes could be revealed to those domains. For example, if the domain requesting those attributes is a highly trusted domain, due to a previously established very restrictive security agreement, the home domain could reveal all the user's attributes. Otherwise, if the relationship established between the domains is not so trusted, the home domain could decide to conceal some of them.

The Disclosure Policy identifies the allowed external CCSs (domains), assigns specific roles to every domain based on the existing relationships between the two domains, and defines the set of attributes that can be revealed and under which conditions.

In the proposed scenario, we suppose the home domain is based on the PERMIS authorization infrastructure, that is, the use of Attribute Certificates to represent the subject-attribute pair. This Disclosure Policy uses the resource access control policy defined by PERMIS, which requires that every external domain must have, at least, an internal AC defining the role played by that domain.

The Disclosure Policy defines the following elements:

- *Subjects*: Set of external domains allowed to request internal user's attributes.
- *Roles*: The set of roles that can be played by external domains. Those roles might be organized hierarchically.
- *SOA*: Authorization Authority managing the ACs issued by the home domain for external CCSs.
- *Targets*: The set of users whose attributes are to be disclosed (the user requesting the access to the foreign network must be included here).
- *Actions*: Only *disclose* has been defined, and its parameter is the attribute that is to be disclosed.
- *TargetAccess*: Defines which attributes assigned to a particular set of users can be disclosed to which domains, and under which conditions.

Figure 2 represents a simple Disclosure Policy defined by *o=PERMISDomain, c=C*. The *SubjectPolicy* element specifies that only the external domain *cn=CCS, o=SAMLDomain, c=C* will be allowed to request attributes. As we explained before, the NAS-SAML domain must have an attribute certificate issued by the *PERMISDomain* SOA. This AC will contain the attribute type *permisRole*, with value *LongTerm-CCS*, specifying the role played by this domain, in this case, a stable and durable relationship is specified.

The *TargetPolicy* defines the set of PERMIS users who make use of external resources, that is, only attributes assigned to those users can be revealed by the UAM. There is only one allowed action, *disclose*, which uses an attribute type and an attribute value as parameters. Finally, the *TargetAccessPolicy* defines that only CCSs (*Subjects*) assigned to the *LongTerm-CC* attribute value will have access to the attribute *studentRole* type, with value *ERASMUS*, assigned to *Students*.

```
<X.509_PMI_RBAC_Policy OID="2.6.2004.24.1.2005">
  <SubjectPolicy>
    <SubjectDomainSpec ID="SD_International_CCSs">
          <Include LDAPDN="   cn=CCS, o=SAMLDomain,c=C    "/>
    </SubjectDomainSpec>
  </SubjectPolicy>
  <RoleHierarchyPolicy>
    <RoleSpec Type="permisRole"
          OID="1.2.826.0.1.3344810.">
          <SupRole Value="ShortTerm-CCS">
          </SupRole>
            <SupRole Value="   LongTerm-CCS  ">
                  <SubRole Value="ShortTerm-CCS"/>
          </SupRole>
          </RoleSpec>
  </RoleHierarchyPolicy>
  <SOAPolicy>
    <SOASpec ID="PERMISDomain_UAM"
          LDAPDN="  cn=UAM,o=PERMISDomain,c=C     "/>
  </SOAPolicy>
  <RoleAssignmentPolicy>
    <RoleAssignment>
          <SubjectDomain ID="SD_International_CCSs"/>
          <RoleList>
            <Role Type="   permisRole  "
                  Value="  LongTerm-CCS   "/>
          </RoleList>
          <Delegate Depth="0"/>
          <SOA ID="PERMISDomain_UAM"/>
          <Validity/>
    </RoleAssignment>
  </RoleAssignmentPolicy>
  <TargetPolicy>
    <TargetDomainSpec ID="All-Users">
          <Include LDAPDN="o=PERMISDomain, c=C"/>
    </TargetDomainSpec>
    <TargetDomainSpec ID="Students">
          <Include LDAPDN="ou=Students,
                  o=PERMISDomain, c=C"/>
    </TargetDomainSpec>
    <TargetDomainSpec ID="Professors">
          <Include LDAPDN="ou=Professors,
                  o=PERMISDomain, c=C"/>
    </TargetDomainSpec>
  </TargetPolicy>
  <ActionPolicy>
    <Action Name="disclose" Args="role value"/>
  </ActionPolicy>
  <TargetAccessPolicy>
    <TargetAccess>
          <RoleList>
            <Role Type="permisRole"
                  Value="LongTerm-CCS"/>
          </RoleList>
          <TargetList>
            <Target Actions="disclose">
                  <TargetDomain ID="Students"/>
            </Target>
          </TargetList>
          <IF>
            <AND>
                  <Substrings>
                    <Arg Name="role" Type="String"/>
                    <Constant Type="String"
                          Value="studentRole"/>
                  </Substrings>
                  <Substrings>
                    <Arg Name="value" Type="String"/>
                    <Constant Type="String"
                          Value="ERASMUS"/>
                  </Substrings>
            </AND>
          </IF>
    </TargetAccess>
  </TargetAccessPolicy>
</X.509_PMI_RBAC_Policy>
```

Figure 2. Disclosure Policy example

When the UAM module receives an attribute query it must generate a request message for each user attribute, specifying the target domain (subject and role), the attribute requested (type and value) and the required action. This request is sent to the PDP and it returns a response message indicating whether the attribute can be revealed or not.

Conversion Policy. A target domain, which has to interact with a home domain based on a different authorization system, needs both to use the CCS module and to define the conversion policy for each domain. Following the design described in [12], this policy is based on XACML, and it defines the following elements:

- *Subject*: One or more subjects specifying the related home domains.
- *Resource*: The resource elements represent the credentials issued by the home domain that need to be translated into internal credentials.
- *Action*: This policy contains only the *translate* action.
- *Obligation*: Every permitted translation will imply an obligation, which specifies how to translate the credentials.

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:cd:04"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation=" access_control-xacml-2.0-policy-schema-
cd-04.xsd"
PolicySetId="GlobalConversionPolicy" PolicyCombiningAlgId="...">
   <Target>
      <Actions>
            <Action>
               <ActionMatch MatchId="...:string-equal">
                  <AttributeValue DataType="...#string">translate
                        </AttributeValue>
                  <ActionAttributeDesignator          AttributeId="..:action"
                        DataType="...#string"/>
               </ActionMatch>
            </Action>
      </Actions>
   </Target>
   <PolicySet PolicySetId="PERMISDomainConversionPolicy"
      PolicyCombiningAlgId="...">
       <Target>
            <Subjects>
               <Subject>
                     <SubjectMatch MatchId="...:string-equal">
                        <AttributeValue DataType="...#string">
                              cn=UAM,o=PERMISDomain,c=C
                           </AttributeValue>
                     <SubjectAttributeDesignator
                        AttributeId="...:external:SOA"
                        DataType="...#string"/>
                     </SubjectMatch>
               </Subject>
            </Subjects>
      </Target>
```

```
<Policy PolicyId="urn:ccs:PERMISDomainSimplePolicy1"
                  RuleCombiningAlgId="...">
      <Target/>
      <Rule RuleId="PERMISDomainSimpleRule1" Effect="Permit">
         <Target>
            <Resources>
                  <Resource>
                     <ResourceMatch MatchId="...:string-equal">
                        <AttributeValue DataType="...#string">
                              studentRole
                        </AttributeValue>
                        <ResourceAttributeDesignator
                              AttributeId="...:resource-id"
                              DataType="...#string"/>
                     </ResourceMatch>
                     <ResourceMatch MatchId="...:string-equal">
                        <AttributeValue DataType="...#string">
                              ERASMUS
                        </AttributeValue>
                        <ResourceAttributeDesignator
                              AttributeId="...:value"
                              DataType="...#string"/>
                     </ResourceMatch>
                  </Resource>
            </Resources>
         </Target>
      </Rule>
      <Obligations>
         <Obligation ObligationId="urn:ccs:Obligation1"
            FulfillOn="Permit">
            <AttributeAssignment DataType="...#string"
                  AttributeId="urn:saml:attribute:role:student">
                  ERASMUS
            </AttributeAssignment>
         </Obligation>
      </Obligations>
   </Policy>
</PolicySet>
</PolicySet>
```

Figure 3. Conversion Policy example

Figure 3 shows a simple Conversion Policy composed by the set of policies related to every home domain. For example, *PERMISDomainConversionPolicy* defines the whole set of attributes that can be translated from the domain *o=PERMISDomain,c=C*. There is only one allowed action, *translate*. The home domain is specified using the *Subject* element, and for each attribute of that domain it is necessary to define a conversion policy. This specific policy, for example the *PERMISDomainSimplePolicy1*, defines the *Rule* specifying the attribute to be translated (type and value), and a *Obligation* element specifying the internal target attribute. For example, the previous example defines that the *PERMISDomain* attribute type *studentRole* with value *ERASMUS* must be translated into the internal attribute type *urn:saml:attr:role:student*, with value *ERASMUS*. It is worth noting that the addition of home domains involves additional *PolicySet* elements, and more attributes per domain requires more *Policy* elements.

When the CCS module receives a conversion query it must generate a policy request message, specifying the home domain subject, the source user's attributes and the required action. This request is sent to the policy manager and it returns a policy response message including whether that action over that resource has been allowed, and the target user's attributes as obligations elements. Then the CCS module returns to the petitioner a conversion response in the internal domain format.

4 Design Alternatives

Interactions among the different components described in the paper depend on the requirements imposed by the user to gain access to the network. On the one hand, the end user can follow a pull approach, which requires the minimum overload and is more suitable for limited terminals, such as PDAs or mobile phones. In this way, all the authorization tasks are performed by the system. On the other hand, following a push model, the user can present a particular set of attributes. The push model involves support for selecting and transporting attributes from the end user terminal, representing a more intrusive approach. In consequence, we provide solutions to these two different environments, including three different alternatives. Beside the pull model approach, we present to alternatives for the push model. In the first one the user obtains his credentials using SAML, once the user AC's are translated. In the second one, he directly obtains his Attribute Certificates, which will be translated during the network access attempt. In both ways, before request access to the AAA server, the user selects which of them wants to present.

Independently of the selected approach, when the end user requests access in a target domain, he should be authenticated, before starting the authorization process, as described in [12], but it is out of the scope of this paper.

4.1 Design alternative 1: Pull model

This alternative provides to the user an authenticated and authorized connection in a transparent way. The management of the authorization data, that is, the conversion and validation process, are performed using a pull approach.

In this way, the first step is the authentication of the user, following the process required by the network access technology. We suppose this authentication is based on public key certificates, therefore, the user must present one of these during the authentication process and it must be validated by the target AAA server. In this scenario, the authentication is delegated, and might be based on the existence of a previously generated cross-certification relationship between both domains.

Once the user is authenticated, the authorization process starts. The AAA server has to discover that the user belongs to a home domain which is not based on SAML/XACML, and therefore his attributes must be converted. The user's home domain can be discovered using the DN attribute held in his certificate.

The AAA server sends the attribute query requesting the user's attributes to the CCS module. This request is formed by a *SAMLRequest* object containing an *AttributeQuery* and indicating that the response must be encoded using the *AttributeStatement* sentence. It also includes the name of the user (subject) requesting the access and, optionally, the type of attributes expected.

Once the CCS obtains the request, it has to discover how to contact the user's home domain. This information is stored in the *Resource Access Policy*, as described in [12], as additional information about the domains able to gain

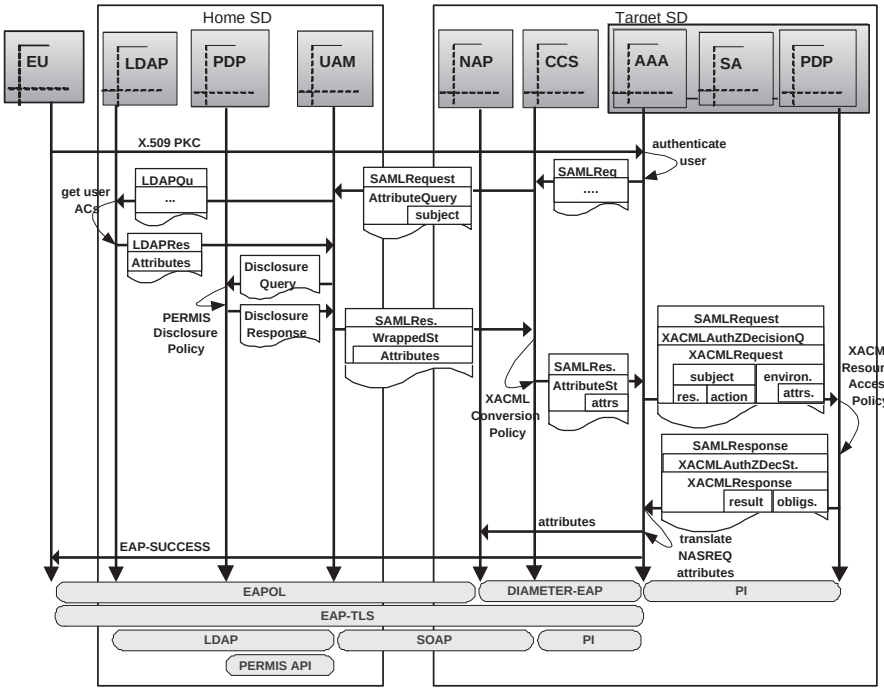

Figure 4. Pull model

network access. The contact point specified in that policy is the UAM module, located in the user's home domain.

The CCS signs and forwards the attribute query, changing the expected response sentence to *WrappedStatement*. In this way, the CCS module indicates to the UAM that the user's attributes must be returned in the original format, but encapsulated in this statement, to be translated at the target domain.

When the UAM receives the attribute query, it has to return only the attributes allowed by the Disclosure Policy. To check this policy, the UAM obtains from the internal LDAP repository all the ACs issued to this user, and the ACs issued to the target domain. The first set of certificates contain all the user's attributes in his home domain, whilst the second one contains the roles played by the target domain. The CCS public key certificate, included in the XML Signature object of the request message, can be used to authenticate the CCS and obtain the target domain.

At this point, the UAM asks the PDP component, the decision point, for the attributes that can be revealed. Using the above ACs, the UAM issues a request per user's attribute, specifying: the target domain (*subject*), the attributes assigned to that domain, the user attribute and its value, and the requested action (*disclose*). These requests are sent using the PERMIS API.

Once the UAM knows the set of attributes that can be returned to the target domain, it generates a *SAMLResponse* including a *WrappedStatement* sentence, following the schema defined in [7]. This statement includes:

- *WrappedData*: The allowed user's ACs.
- *StatementType*: URI describing the type of the wrapped data, for example, *x509ac*.
- *Encoding*: URI describing the encoded format, for example, *Base64*.

Once the CCS module receives the response message, it must convert the received attributes into internal understandable SAML attributes. To obtain these, the CCS uses the Conversion Policy, described in section 3.2. The CCS obtains the SOA identifier and each attribute type and value pairs needed to be converted. All this information is included in the ACs. For each attribute type and value pair, the CCS checks the policy and obtains the associated attribute designator and value in internal format (SAML).

Once all the attributes are translated, the CCS sends a *SAMLResponse* message including all of them as *AttributeStatement* sentences. When the AAA server receives the user's attribute it checks the Resource Access Policy as described in [12], to grant or deny the required service.

The pull model provides strong authentication of users, and a transparent authorization service based on ACs and SAML statements, avoiding the client software being modified to support these high level authorization schemes. This alternative has the disadvantage of providing no control to the user about the type of service required, that is, the user can not select the set of attributes to be presented during the network access request. In our opinion, this should not be seen as a disadvantage in most of the existing environments where default access is being provided or where the users do not want to get involved in authorization issues.

4.2 Design alternative 2: Push model based on SAML Attributes

Using the push model, end users are able to present their authorization credentials during the network access request. In this alternative, those authorization credentials are expressed using SAML attribute statements containing the roles played by the user. This model is based on two steps: first, the user has to request his attributes from his home domain, specifying the desired target domain and service. This step involves the conversion process ending with the user holding the converted attributes. Then, the user presents those converted attributes to the target domain, requesting, for example, network access. This steps involves the authentication and the authorization decision processes. This section describes the first step since the last stage is fully described in [12].

In this model, the user requests his attributes from his home domain, specifying the desired target domain and service. The user directly accesses the UAM module, for example, through a web browser. The UAM, once the user is authenticated, follows the same procedure described in the pull model, only in this

case the user acts as a proxy for the target domain. First, the UAM retrieves the user's ACs from the LDAP repository, and then asks the PDP module about the attributes that can be revealed to the target domain.

In this way, and following the push model described in [7], the UAM sends to the CCS a *ConversionQuery* sentence including the *WrappedStatement* described above. The *ConversionQuery* sentence contains the following elements:

- *Assertion*: Assertion including the *WrappedStatement* object.
- *Recipient*: Attribute defining the entity that will receive the assertions, for example, the CCS.
- *RespondWith*: This element, including in the *SAMLRequest* message, is used to specify type of SAML assertion that is expected to be generated, for example, the *AttributeStatement*.

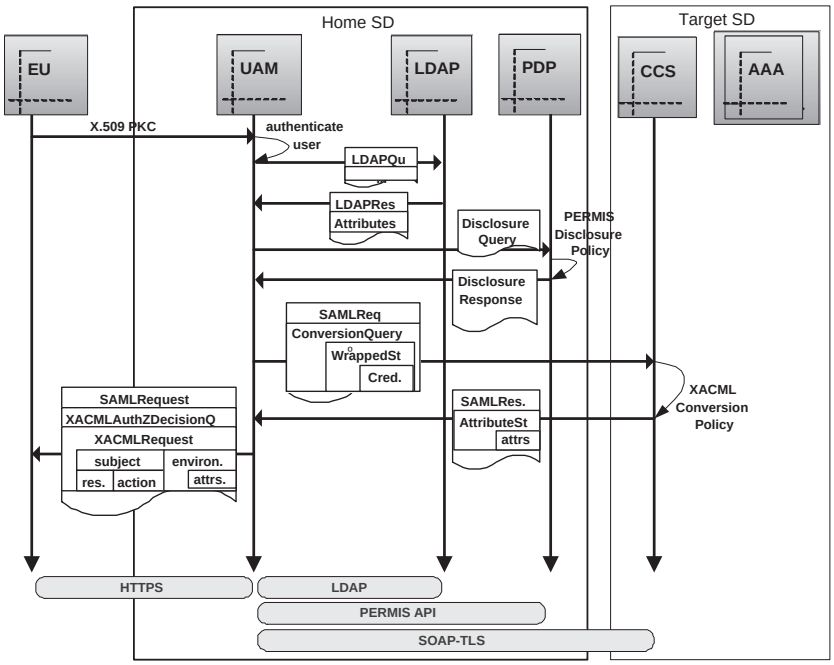

Figure 5. Push model based on SAML Attributes

The UAM waits for a *SAMLResponse* message including the converted attributes. The location of the CCS could be previously configured in the home domain, as part of the Disclosure Policy, or could be obtained from the CCS public key certificate, as an *authorizationInformationAccess* X.509 extension. The CCS module, after enforcing the Conversion Policy, returns the converted

attributes as an *AttributeStatement* to the UAM, which generates the *XACMLAuthZDecisionQuery* following the push model described in [12]. The user then forwards this to the AAA server.

It it worth noting that absence of revocation mechanisms for SAML statements, and its recommended usage for short-term sessions, suggests that the SAML documents should not be cached in intermediate entities, like a certificate repository.

The main advantage of this alternative is that it provides to the end user complete visibility and control of the authorization process, since he can select the type of connection, security properties, quality of service, etc. Moreover, he can provide personal information by means of references to some of his attributes. On the other hand, the software used by the client (usually referenced as supplicant) must be modified in order to deal with SAML statements during the second step, as we can find in other existing proposals [12].

4.3 Design alternative 3: Push model based on Attribute Certificates

In this alternative, authorization credentials, presented by the end user to the AAA server, will be the user's Attribute Certificates, obtained from the UAM in the user's home domain. This model is based on two steps: first, the user has to request his ACs from his home domain, specifying the desired target domain and service. Then, the user, after filtering those according to his privacy policy, presents them to the target domain, requesting, for example, network access. This last step involves the authentication, the credentials conversion, and the authorization decision processes.

In this model, the user requests his attributes from his home domain, specifying the desired target domain and service. The user directly accesses the UAM module, for example, through a web browser. The UAM, once the user is authenticated, retrieves the user's ACs from the LDAP repository, and then asks the PDP module about the attributes that can be revealed to the target domain. The UAM returns the selected ACs to the end user.

Once obtained the user's ACs, he requests network access connection in the target domain, pushing those ACs according to his internal policy. The user initiates the 802.1X authentication process with the foreign AAA server. In this case, we are going to use the PEAP (Protected EAP) protocol [2], which defines how to establish a TLS channel that can be used to authenticate the communicating parties and to protect further messages related to the authorization process. For example, the user's ACs would be base64 encoded and sent as normal attributes.

The AAA server that receives the network connection detects the use of non-SAML credentials, and sends a credential conversion request to the local CCS. That is, a *ConversionQuery* sentence including the *WrappedStatement* described above.

The AAA server waits for a *SAMLResponse* message including the converted attributes. The CCS module, after enforcing the Conversion Policy, returns the

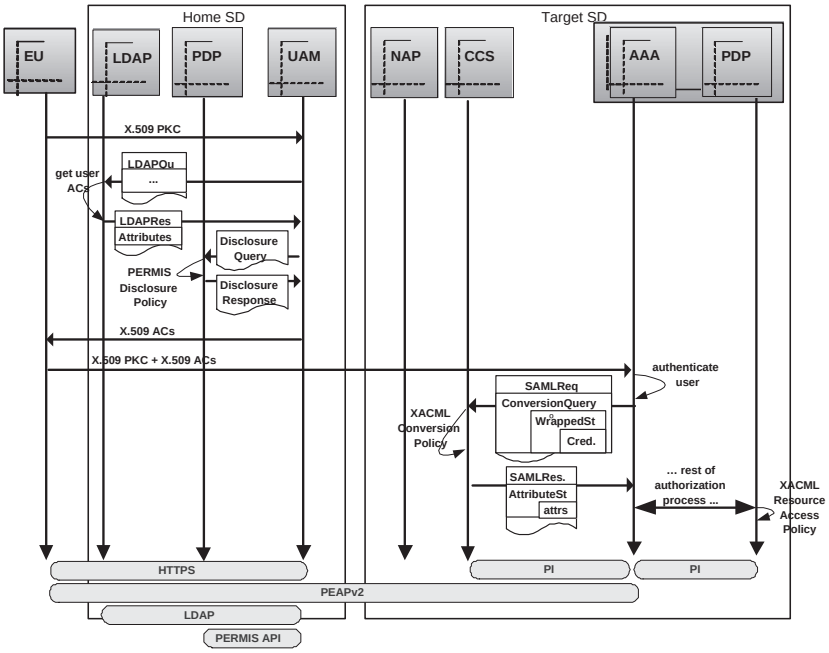

Figure 6. Push model based on Attribute Certificates

converted attributes as an *AttributeStatement* to the AAA server, which generates the *XACMLAuthZDecisionQuery* following the push model described in [12].

This alternative presents the same advantages and disadvantages that the alternative 2 previously described. The main difference is the transport of ACs from the home domain to the target domain, where they need to be translated by the CCS module.

5 Related work

Nowadays, due to the existence of several authorization schemes, their integration is becoming a common requirement in multi-domain scenarios. This section describes some of the main current integration solutions which have informed our work, including environments such as SAML, PERMIS or X-RBAC [11].

PERMIS is being used by other authorization schemes to improve the authorization decision process and to allow users holding an Attribute Certificate to interact with other environments, such as Shibboleth [4] or Grid Services [1].

Shibboleth defines an access control approach scenario for web environments, which is composed of three main entities: service providers offering target resources; identity providers maintaining the user's identities; and the end users. It offers user authentication and attribute-based authorization services based on

SAML, as well as a SSO service. It is based on a single Attribute Authority, and has no generalized decision engine. One solution to improve Shibboleth is the integration with PERMIS. During the SIPS project [16], PERMIS was integrated to work with Shibboleth. There are two modes of operation: in one mode PERMIS uses X.509 Attribute Certificates to make authorization decisions, and in the other mode PERMIS uses plain Shibboleth attributes. The PERMIS team had to develop a module similar to the conversion service discussed in this paper, so that PERMIS would accept the plain-text attributes. The module that was developed as the result is not a standalone service, rather it is an extension to PERMIS that is invoked via the API. This only once again proves the necessity for credential conversion for efficient interoperation of Privilege Management Infrastructures.

PERMIS has also been extended to support the SAML standard for Grid Services, as defined in [5]. This describes how PERMIS has been adapted in order to integrate its authorization engine into the Globus Toolkit. This integration is based only on the exchange of authorization decision queries and responses between an authorization service acting as the Policy Decision Point (PDP), based on X.509 ACs, and the Grid infrastructure. In this scenario, the decision about the resource is taken by the PERMIS ADF module, following the PERMIS policy syntax. In this way, it takes the advantage of the SAML standard for integration purposes. It is also able to use the SAML extensions proposed by the OGSA-Authz working group [17] for efficiency purposes.

Besides PERMIS, other scenarios describing the integration between different authorization schemes are being defined. An interesting solution for the authorization process in multi-domain environments, based on the RBAC model, is described in [11]. Although this model does not propose an integration scenario, because both domains are based on the same authorization scheme (X-RBAC), it does propose a policy mapping users' attributes from one domain to another. This proposal does not need a conversion service but it clearly shows that the relationship between the user's attributes or roles from different domains must be mapped in some way. In order to support mapping between different authorization schemes, the proposed policy should be extended allowing the definition of the source and destination authorization formats.

6 Conclusions

This paper proposes a solution to integrate two different authorization schemes, PERMIS which is being widely used due to its powerful authorization engine, and SAML which is becoming a *de facto* standard for authorization environments. We have presented a particular application scenario, the network access service, to demonstrate how the integration addresses, for example, authorization between domains based on a RBAC scheme.

Beside the architectural elements needed by the integration process, this paper presents the guidelines of the policies controlling the integration scenario, the Disclosure and Conversion policies. These policies are defined using the different

authorization languages proposed by the two domains, the PERMIS XML authorization policy and XACML. In this way, there is no need to include additional authorization technologies to accomplish the integration process.

In order to offer a versatile solution, this paper presents three different RBAC designs, which can be individually selected in order to implement the access control service that is best suited for a particular environment. Authorization can be performed in a transparent way, from the user's point of view, using the pull model. The two push model approach slightly overloads the system in relation to the previous model, but it provides more options to the users.

As a statement of direction, although the proposed scenario is based on a PERMIS home domain and a NAS-SAML target domain, it could be easily adapted to work in reverse order, that is, based on a SAML home domain and a PERMIS target domain.

7 Acknowledgements

Partially supported by IST-2001-32161 (Euro6ix) and IST-2002-001929 (SEINIT) projects.

References

1. *Globus Toolkit*, February 2005. http://www.globus.org.
2. H. Anderson, S. Josefson, G. Zorn, D. Simon, and A. Palekar. *Protected EAP Protocol (PEAP)*, 2004. IETF Draft.
3. P. Ashley, S. Hada, G. Karjoth, and M. Schunter. E-P3P privacy policies and privacy authorization. In *WPES '02: Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society*, pages 103–109. ACM Press, 2002.
4. S. Cantor. *Shibboleth Architecture. Protocols and Profiles*, February 2005. http://shibboleth.internet2.edu/docs/draft-mace-shibboleth-arch-protocols-06.pdf.
5. D. W. Chadwick, S. Otenko, and V. Welch. Using SAML to link the GLOBUS toolkit to the PERMIS authorisation infrastructure. In *Proceedings of Eighth Annual IFIP TC-6 TC-11 Conference on Communications and Multimedia Security*, 2004.
6. David W. Chadwick, Alexander Otenko, and Ed Ball. Role-based access control with x.509 attribute certificates. *IEEE Internet Computing*, 7(2):62–69, 2003.
7. O. Cánovas, G. López, and A. F. Gómez. A Credential Conversion Service for SAML-based scenarios. In *Proceedings of 1st European PKI Workshop*, pages 297–305, June 2004.
8. C. de Laat, G. Gross, L. Gommans, J. Vollbrecht, and D. Spence. *Generic AAA Architecture*. Internet Engineering Task Force, August 2000. Request for Comments (RFC) 2903.
9. S. Godik and T. Moses. *OASIS eXtensible Access Control Markup Language (XACML) Version 2.0*, February 2005. OASIS Standard.
10. The Open Group. *Authorization (AZN) API*, January 2000.
11. E. Bertino J. B. D. Joshi, R. Bhatti and Arif Ghafoor. Access-Control Language for Multidomain Environments. *IEEE Internet Computing*, 8(6):40–50, November 2004.

12. G. López, O. Cánovas, A. F. Gómez, and R. Marín. A Network Access Control Approach based on the AAA Architecture and Authorization Attributes. In *Proceedings of International Workshop on Systems and Security Networks (SSN05)*, April 2005. To be published.
13. E. Maler, P. Mishra, and R. Philpott. *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1*, September 2003. OASIS Standard.
14. LAN MAN Standards Committee of the IEEE Computer Society. *IEEE Draft P802.1X/D11: Standard for Port based Network Access Control*. IEEE, March 2001.
15. K. Seamons, M. Winslett, and T. Yu. Limiting the Disclosure of Access Control Policies during Automated Trust Negotiation. In *Proceedings of Network and Distributed System Security Symposium*, April 2001.
16. SIPS. *Seamlessly Integrating PERMIS and Shibboleth*, February 2005. http://www.jisc.ac.uk.
17. V. Welch, R. Ananthakrishnan, F. Siebenlist, D. Chadwick, S. Meder, and L. Pearlman. *Use of SAML for OGSA Authorization*, February 2005. GGF Draft (OGSA-Authz WG).