Web services access control architecture incorporating trust

Marijke Coetzee[1]
J H P Eloff[2]

1) Academy for Information Technology, University of Johannesburg, Northcliff, South Africa
2) J.H.P. Eloff, Information and Computer Security Architecture Research Group, Department of Computer Science, **University of Pretoria**, Pretoria, South Africa

[Fgures and tables at the bottom of the document, before the references]

Abstract

Purpose – This paper seeks to investigate how the concept of a trust level is used in the access control policy of a web services provider in conjunction with the attributes of users.

Design/methodology/approach – A literature review is presented to provide background to the progressive role that trust plays in access control architectures. The web services access control architecture is defined.

Findings – The architecture of an access control service of a web service provider consists of three components, namely an authorisation interface, an authorisation manager, and a trust manager. Access control and trust policies are selectively published according to the trust levels of web services requestors. A prototype highlights the incorporation of a trust level in the access control policy as a viable solution to the problem of web services access control, where decisions of an autonomous nature need to be made, based on information and evidence.

Research limitations/implications – The WSACT architecture addresses the selective publication of policies. The implementation of sophisticated policy-processing points at each web service endpoint, to automatically negotiate about policies, is an important element needed to complement the architecture.

Practical implications – The WSACT access control architecture illustrates how access control decisions can be made autonomously by including a trust level of web services requestors in an access control policy.

Originality/value – The WSACT architecture incorporates the trust levels of web services requestors and the attributes of users into one model. This allows web services providers to grant advanced access to the users of trusted web services requestors, in contrast with the limited access that is given to users who make requests through web services requestors with whom a minimal level of trust has been established.

1. Introduction

Today, organisations that seek a competitive advantage are adopting virtual infrastructures that share and manage computing resources. The trend is towards implementing collaborative applications that are supported by web services technology (Gottschalk *et al.*, 2002). Even though web services technology is rapidly becoming a fundamental development paradigm, adequate security remains an obstacle to its adoption as an industry solution. An important issue to address is the development of suitable access control models that are able to not only restrict access to unauthorised users, but also to discriminate between users that originate from different collaborating parties.

In web services environments, access control is required to cross the borders of security domains, in order to be implemented between heterogeneous systems (Coetzee and Eloff, 2004). Traditional access control systems that are identity-based do not provide a solution, as web services providers have to deal with unknown users, manage a large user population, collaborate with others, and at the same time be autonomous of nature. Previous research (Bertino *et al.*, 2004; Biskup and Wortmann, 2004; Bacon and Moody, 2002), point towards the adoption of attribute-based access control as a means to address some of these problems. Security Assertion Markup Language (SAML) (Cantor *et al.*, 2005) provides a mechanism for web services to transfer information about the attributes of users between collaborating domains without the need for those domains to lose ownership of that information. For autonomous collaboration, performing access control based on attributes of users is not sufficient, as it does not make it possible to treat users of trusted partners different to those with whom no substantive relationship exists. The trust in web services requestors is thus an important pre-requisite for secure collaborative interaction. Consequently, this paper investigates the inclusion of a trust level in the web services access control architecture.

The paper is structured as follows: Section 2 gives a web services example that draws attention to access control concerns. Section 3 highlights the progressive inclusion of trust in access control architecture, in order to motivate the approach followed by this research. Section 4 introduces the web services access control architecture incorporating Trust (WSACT architecture), and describes its components. Section 5 gives details of the prototype, and finally the paper is concluded by section 6.

2. Web services example

Figure 1 shows eRetailer, a provider of retail web services. Consider the following two operations that it exposes:

1. Search_academic is an operation that can be accessed by students who are in possession of a digital credential that proves that they are registered at an academic institution. It gives access to books and other products that are reserved for students at reduced prices.
2. List_specials is an operation that can be accessed by employees or clients of partners with whom an excellent business relationship exists, as it gives access to special offers on newly released products.

Assume that there is a user Sue Smith, who is both an employee and a student. She is employed by eCompany, an organisation that integrates the operations of eRetailer in its application environment as a benefit to employees. eCompany has fostered a sound relationship with eRetailer over many years. Their employees and clients are granted special offers on newly released products because of the nature of the relationship with eRetailer. Sue is also registered as a part-time student at eInstitution, an academic institution that integrates the operations of eRetailer in its application environment to enable its students to purchase academic books by using their student loans.

Implementing such web services applications requires a unique approach to access control. In the case of Sue accessing the search_academic operation from the eInstitution application environment, the focus of access control is on her, the user. The fact that she is in possession of a digital credential that consists of a set of attributes describing her status as student, will grant her access to the operation. This means that a user can be granted access to this operation if it is in possession of such a credential, irrespective of the environment from where the operation is accessed.

In the case of Sue accessing the list_specials operation, the focus of access control is the trust level of the web services requestor. Sue is granted access to the operation not because of who she is, but because eCompany makes the request on her behalf, and the trust in eCompany is high. This means that Sue will not be able to access this operation from another environment other than eCompany, or web services requestors that are highly trusted.

eRetailer also exposes operations such as place_order. In this case, a combination of the attributes of a user, and the trust level of a web services requestor are used to determine whether access is granted or not. Here, the identity of the user must be known, and the trust in the web services requestor must be of moderate level.

This scenario identifies the focus of the architecture proposed in this paper. An access control decision is not only based on the attributes of users, but also on a differentiable trust relationship with the web services requestor presenting the attributes. Next, the role of trust in access control architecture is briefly described, in order to define the context in which trust is used, in the architecture proposed in this paper.

3. Access control architecture and trust

Several web services access control proposals have been suggested in the past. Research has steadily progressed by recommending the use of mechanisms such as access control lists (Damiani *et al.*, 2001), role-based access control (Wonohoesodo and Tari, 2004), attribute based access control (Shen and Hong, 2006; Lopez *et al.*, 2005) and trust management (Olson *et al.*, 2006). Often mechanisms are used in conjunction with each other to enhance flexibility (Miao *et al.*, 2005; Koshutanski and Massacci, 2003).

In line with these developments, the progressive inclusion of trust in distributed access control architectures is reported in literature. Architectures such as the ISO 10181-3 Access Control Framework (ISO, 1996) and the Internet Engineering Task Force (IETF) policy architecture (Guerin *et al.*, 2000) identifies the separation of access control decision-making and access control enforcement, but does not identify trust in its design. Trust management systems (Rivest and Lampson, 1996), (Blaze *et al.*, 1999) enabled cross-domain movement of entities, represented by credentials. Automated trust negotiation systems (Winslett, 2002), (Hess *et al.*, 2002) establish trust over the exchange of the properties of entities, in the form of digital credentials. As cryptographic controls do not guarantee the behaviour of parties participating in Internet commerce, Barkat and Siyal (2002) have described the role of a network of trust service providers that provide services related to aspects such as quality of products, the enforcement of contracts, and assistance with terms of negotiation between parties. The secure framework architecture (Cahill *et al.*, 2004) demonstrates how trust can be formed over evaluation of trust based on past experiences and other information. These architectures represent the progression of trust in access control architectures from binary trust that is created by the verification of public keys, to incremental trust that is created by the exchange of digital credentials, to autonomous trust that reflects the trust held towards another party by processing information, evidence and recording history.

The architecture proposed in this paper is used in support of applications characterised by collaboration between unfamiliar parties. In such environments there is a lack of a central control authority. The lack of central control localises the responsibility for access control and other decisions in the domain of each participant. This requires of web services providers to be autonomous when access control decisions are made. The concept of autonomous trust can enable web services requestors and providers to reason about relevant information and evidence before making access control decisions. Following this line of thought, this research implements an autonomous approach to access control and trust by including a trust component in the proposed architecture so that the trustworthiness of requestors can continuously be computed and be re-evaluated. Previous research by the authors (Coetzee and Eloff, 2006) has defined a comprehensive web services trust formation framework. The results of this trust framework is a trust level such as “high” or “moderate” that can be used for access control and other decisions. This paper extends the previously defined trust formation framework to illustrate how trust levels of web services requestors can be used in access control decision-making. Next, the architecture is described.

4. Web services access control architecture incorporating trust (WSACT)

In order to be able to answer the question “Which request does a web services provider grant to a user – on whose behalf a web services requestor is acting – if trust in the web services requestor is high/low?”, the WSACT architecture implements a number of components, namely:

- A component positioned at the perimeter of a web services provider, which manages all interactions related to trust and access control requests.
- A component internal to a web services provider, which makes access control decisions based on access control and trust policies.
- A component internal to a web services provider, which supports trust formation by calculating a trust level for a web services requestor.

A general overview of the main components of the architecture and their relation is depicted in Figure 2. The architecture consists of an authorisation interface, an authorisation manager and a trust manager. Together, these components constitute an access control service. The access control service should be secured so that it cannot be compromised. It is important to note that the architecture does not address the administration of policies. This would be an important feature of a comprehensive solution.

The access control service intercepts all SOAP requests. As SOAP is considered the de facto standard for web services, it will be referred to in this discussion. SOAP requests are formatted by web services requestors according to the functional description as defined by the interface document and the non-functional description found in associated policy documents. The first component, the authorisation interface, inspects the header and body of the SOAP request and may optionally store information in the information database. Next, the authorisation manager is invoked to make an access control decision. Based on the latter's answer, requests are either granted or denied. In the background, the trust manager calculates a trust level for each web services requestor.
Next, each component is described in more detail.

4.1. The authorisation interface

The authorisation interface, shown in Figure 2, is an intermediary in the access path between a web services requestor and web services operation being requested. The authorisation interface is a highly specialised component that extends the functionality of a Policy Enforcement Point (PEP) (Guerin *et al.*, 2000) by controlling not only access to web services operations, but also all interactions related to the collection of trust information. Ideally, an access control service should be designed so that it functions unobtrusively in the background of the application. In order to remain unobtrusive, the access control service should preferably use the same messaging infrastructure as the web service, its interface policy should be referenced from the functional interface of the web service, it should seamlessly be invoked when web services operations are accessed, and it should use the same exchange patterns as the web service.

In order to interact seamlessly, web services requestors and providers expose requirements and capabilities to each other in XML-based policies associated with the interface of the service. An interface policy is a set of assertions about a web services provider, such as the type of encryption algorithm that is supported or the format of a credential that is required, so that web services requestors understand how to use web services operations. Similarly, an access control and trust interface policy must be published to allow interoperation.

The publication of access control requirements has not been addressed by security specifications such as WS-Security (Atkinson *et al.*, 2002), and XACML (Godik *et al.*, 2003) as access control and trust requirements are domain specific, and of a sensitive nature. An unsophisticated approach would be to expose all access control, trust and other requirements and capabilities to any potential web services requestor. Instead, publication of such policies, based on trust levels of web services requestors, is proposed by this research. The publication of sensitive policies is controlled so that web services requestors are given access to sensitive policies according to the level to which they are trusted. When such policies are published, it is private of nature. Relevant endpoint information and policy URLs are sent to web services requestors. WS-MetadataExchange (Ballinger *et al.*, 2004) can for instance be used to retrieve policy and other information from an Internet address. A possible materialisation of interface policies has been described by the authors (Coetzee and Eloff, 2005). The interface policy deals with two types of publication; the publication of requirements related to access control and trust. *Publication of access control requirements*. Unknown web services requestors will initially not have access to interface descriptions of sensitive operations, or of sensitive access control requirements defined in interface policies. This means that the interface policy of eRetailer indicates that a student credential is required to be able to access the search_academic operation, shown in Figure 3. The list_specials operation, and associated interface policy are kept private.

Publication of trust requirements. The interface policy exposes information related to initial trust formation, and trust evolution to all external parties. The identity of web service requestors is first required in the registration process to form an initial level of trust. To increase trust, domain specific requirements are published relating to trusted partners, and required recommendations and references. The gathering of trust information has been described in a preceding paper by the authors (Coetzee and Eloff, 2005).

The main function of the authorisation interface is to intercept all SOAP requests for the purpose of trust formation and access control. The SOAP header is first inspected to determine if trust information such as references or recommendations is present. If so, this information is verified and stored in the information database to be used by the trust manager in trust computation. The authorisation interface next formats an access request for the authorisation manager in syntax that the authorisation manager can understand. Additional credentials or declarations that are required when the decision is made are also formatted so that the authorisation manager can use them in its access control decision-making processes. The authorisation manager is contacted to determine whether the request may be granted. Based on the answer that is returned from the authorisation manager, the authorisation interface either passes the request to the web services operation to be executed or returns an error message to the web services requestor as described by SOAP faults in the interface document.

4.2. The authorisation manager

The authorisation manager is a policy decision point (PDP) (Guerin *et al.*, 2000) that bases all its decisions on requests that it receives, as well as on access control and trust policies. As required, the authorisation manager contacts the trust manager to determine the level of trust of the web services requestor. The trust level of a web services requestor is calculated by the trust manager that continuously evaluates information and evidence. Finally, a decision of either grant or deny is returned to the authorisation interface.

The authorisation manager is defined by means of a specification that enables logical reasoning. This ensures that access control reasoning can be verified and easily understood. It consists of two parts: an access control policy and an inference engine. The access control language developed in this research is an extension of the Authorisation Specification Language (ASL) (Jajodia *et al.*, 1997) that is based on Datalog. The language is extended with constraints (Li and Mitchell, 2003) to compare trust levels. It uniquely addresses attribute-based access control and trust levels in conjunction with each other. The access control policy extends role-based access control mechanisms to grant access to web services requestors based on their trust level.

It is important to note that the focus of this discussion is not to provide a comprehensive treatment of access control specification and reasoning, but rather to highlight how trust levels and attributes are used in conjunction with each other in the access control policy. Logical facts and rules define the access control policy, and are shown in Figure 4. Next, the basic constructs of the access control language are discussed. The discussion first describes the access request, and then the conditions considered by access control reasoning and finally, access control reasoning is illustrated.

Access request. The access request is formatted by the authorisation interface and passed to the authorisation manager. The request, shown in rule 1, is of the form: do(wsobj, requestor, sa), where wsobj is the web services operation being accessed, requestor represents the identity of the web services application making the request, and sa is a signed action to be executed. An access can be granted if a permission can derived by using logical rules of inference. The logical deduction of an access decision is now considered. The following conditions are considered by a process of logical deduction.

Permission. Permission is set by an administrator to allow web services requestors access to an operation. Here, permissions are role-based to make the assignment of permissions simpler and more scalable, and are of the form cando(wsobj, role, sa), where wsobj is the web services operation being accessed, role represents the role that is granted access to the operation, and sa is a signed action to be executed.

Trust levels. Trust levels are assigned to both roles and web services requestors. Trust levels are defined as the set {ignorance, low, moderate, good, high}, where

$ignorance \subset low \subset moderate \subset good \subset high.$ Administrators assign a trust level to each role with the roleTlevel(role, tlrole) predicate that is used in rule 3, where role is the name of the role, and tlrole is the trust level assigned to the role. Roles are defined as the set {visitor, associate, client, partner, trusted_partner}. The role "visitor" is assigned a trust level of "ignorance", and has the least privilege; the role "trusted_partner" is assigned a trust level of "high" and has the most priviledge.

Next, trust levels are assigned to web services requestors. Administrators do not assign roles statically to web services requestors, but it is done dynamically, when an access decision is inferred. A predicate ask, defined in rule 4, is used to dynamically retrieve the trust level of a web services requestor when needed. It is of the form ask(reqtrustlevel, tlreq, requestor), where reqtrustlevel represents a call to the trust manager, requestor is the identity of the web services requestor for which a trust level is computed, and tlreq is the trust level of a web services requestor that is unified with the predicate.

Role activation. Role-activation rules specify the pre-requisite conditions and constraints that a web services requestor must satisfy in order to enter a role. Based on the evaluation of trust levels of objects, and that of web services requestors, roles are activated for the duration of the request that is processed, as shown by rule 3. The assignment of trust levels to both roles and web services requestors thus introduces another level of access control abstraction between web services requestors and operations. A web services requestor that is assigned a trust level of "moderate" creates a session in which it activates a role that is at either a trust level of "low" or a trust level of "ignorance". It dynamically activates roles in a role hierarchy that follows the structure of trust levels. Each session relates the web services requestor to many possible roles that are available according to the role/trust level hierarchy. It follows that access control – incorporating trust levels – enforces one-directional information flow in a lattice of trust levels as is required.

Attribute declaration. In order to access an operation, a user must present valid attributes. The authorisation interface presents these attributes to the authorisation manager in the form of a declaration, shown by rule 5. The predicate declaration_op($attr_i$,……$attr_j$) is used to indicate the set of attributes that are needed to satisfy operation requisites. Each web services operation may require a different set of attributes before access can be granted. In some cases no requisites exist. Attributes used in the evaluation of the access request are added to the set of facts in the access control policy before an access decision of grant or deny is derived.

Access decision. The abovementioned conditions are used to derive a permission, defined by the dercando predicate shown in rule 2. If there exists permission for a role to perform an action, and a web services requestor is active in the role, and the user on whose behalf it is acting has presented valid attributes, access is granted to the operation. The access request, defined in rule 1, finally evaluates to either true or false.

Access control reasoning. Table I illustrates how facts are materialized to answer the question “Is Sue granted access to the search_academic operation, if eInstitution makes the request on her behalf?” Here, the trust in eInstitution does not affect the decision, as the operation is assigned to a role with a trust level of “ignorance”. Trust levels are converted to a range of numerical values, where “ignorance”=1, and “high”=5.
Table II illustrates how facts are materialized to answer the question “Is Sue granted access to the list_specials operation, if eCompany makes the request on her behalf?” Here, the attributes of Sue do not affect the decision, as the operation has no attribute requisites.

4.3.The trust manager

The authorisation manager invokes the trust manager, the third component of the WSACT model. The trust manager bases its computation of a trust level on information contained in the information database. Figure 2 indicates that the information database is populated from four different sources. Records are written by the authorisation interface, automatically by application entities, semi-automatically by a combination of application entities and human intervention, and manually by human intervention.

The trust manager is invoked as the ask predicate is instantiated, shown by rule 4 in Figure 4. The trust manager is called with the identity of the web services requestor in a custom predicate. The latter inspects the information at its disposal and calculates the appropriate trust level. It replies to the request of the authorisation manager by indicating the trust level that is allocated to the web services requestor. The trust level is unified with the predicate in the access control policy, so that an access control decision can be made.

A comprehensive treatment of the design of the trust manager has been presented by the authors (Coetzee and Eloff, 2006) that describes how a trust level of “high” or “low” is determined. The trust manager presents an automated trust formation process for web services. Trust evolves gradually and includes trust in the environment and the underlying control and support mechanisms. In addition, experiences with trading partners and recommendations from trusted referees influence a trust relationship. The trust manager illustrates that much of the required information is available in machine-readable format, and demonstrates how it can be automatically gathered and assessed. The trust level is composed from explicitly defined trust types. Trust is inferred from trust in the internal environment, trust in the external environment, and trust in the other party.

5. Prototype

A request made to a target operation is described next by means of a sequence diagram, shown in Figure 5. Consider the case where a user Sue, makes a request to access the Order operation. Both her attributes and the trust in the web services requestor will play a role in determining if she is granted access to the operation. A web services requestor such as eCompany, receives her request and attributes, and makes a call on her behalf for the Order operation. A SOAP request is formatted, that contains her attributes and credentials of eCompany. The authorisation interface intercepts all credentials, adds them to the Information Database, and formats a request for the authorisation manager. When required, the authorisation manager interrogates the trust manager for the trust level of eCompany.

The trust manager calculates the trust level of eCompany. The calculated trust level and the asserted attributes of Sue are used by the authorisation manager to determine a response of either grant or deny. If the request is granted, access to the target operation is enforced. The target operation is executed, and a response returned to the web service requestor, that finally returns a response to Sue.

In order to evaluate the proposed WSACT architecture, a prototype of limited scope was implemented. The prototype was developed on the Microsoft ASP.NET (MS ASP.NET, 2005) platform, as it provides built-in support for building and consuming standards-based web services. All code is written in VB.NET (MS VB.NET, 2005) and in Amzi! Prolog (AMZI, 2005). The authorisation interface is placed in the request-processing stream between the request handler and all web services operations as a SOAP extension. An important benefit derived from this implementation is that the authorisation interface can be used in conjunction with the implementation of authentication, confidentiality and integrity mechanisms as specified by the WS-Security specification, as it makes use of the same architecture on the .NET platform.

The call from the authorisation interface to the authorisation manager is through the Logic Server of Amzi! Prolog – the Prolog runtime engine, implemented as a Dynamic Link Library. It provides the VB.NET authorisation interface with the ability to query access control policy rules of the authorisation manager, and retrieve an answer of either true or false as is required. During the execution of rules of the access control policy, the authorisation manager determines the trust level of a web services requestor by invoking a method of the trust manager class. A custom built-in Prolog predicate makes it possible for Prolog to directly access methods of the trust manager class.

Experimentation showed that the prototype performs well. It demonstrates that the incorporation of a trust level in the access control policy is a viable solution to the problem of web services access control, where decisions of an autonomous nature need to be made, based on information and evidence.

6. Conclusion

Current research in access control and trust architectures identifies two important themes that need to be addressed namely policy-based management, and the inclusion of autonomous, decentralised trust to allow autonomous decision-making. The WSACT architecture aims to address both these themes by first extending the policy-based approach by means of an interface policy that publishes access control and trust requirements and capabilities, to seamlessly integrate access control functionality between web services providers and requestors. The selective publication of policies, based on the trust levels of web services requestors is proposed so that untrusted parties are not granted access to policies of sensitive nature.

Second, the WSACT architecture employs a trust component that creates and manages a trust level for each requestor in order to make better access control decisions. Normally trust is created incrementally by the exchange and verification of sets of attributes defined in credentials. As trust increases, more and more sensitive resources are exposed. There is a high administrative burden to verify the credentials of each and every user for each request that is made, and to create trust that exists for a single session between the user and web service provider. Such an approach does not mirror the real world, where very often one is not only trusted as an individual, but also as a member of a community or organisation.

The WSACT architecture incorporates the trust levels of web services requestors and the attributes of users into one model. This allows web services providers to grant advanced access to the users of trusted web services requestors, in contrast to the limited access that is given to users who make requests through web services requestors with whom a minimal level of trust has been established. Web services requestors must activate a role before the attributes of users are verified. In some instances only the attributes of the user, or the trust level of a web services requestor, is required to be granted access to web services operations.

Figures and tables

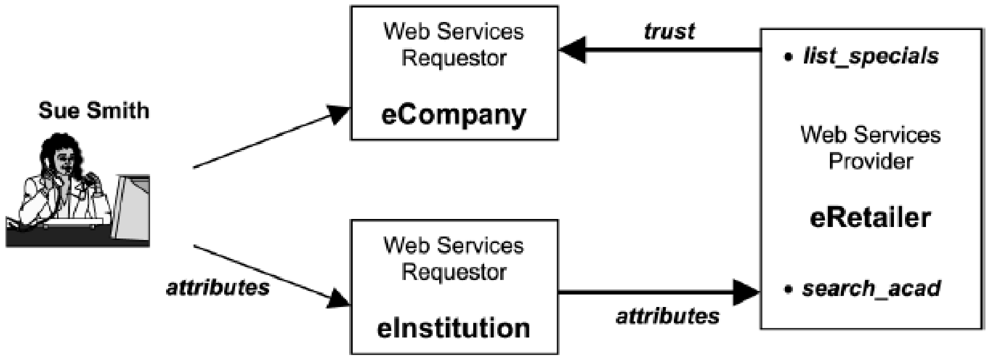


Figure 1 *Access control considerations for web services operations*

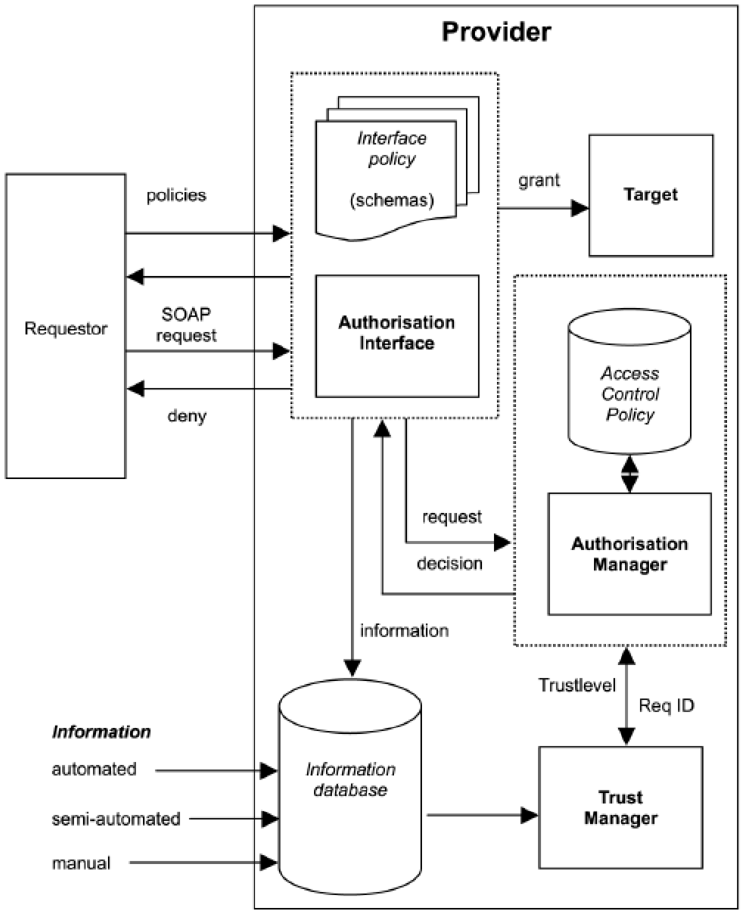


Figure 2 *WSACT architecture*

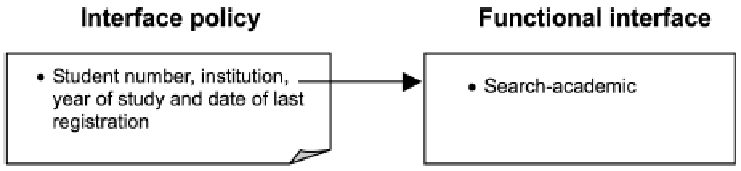
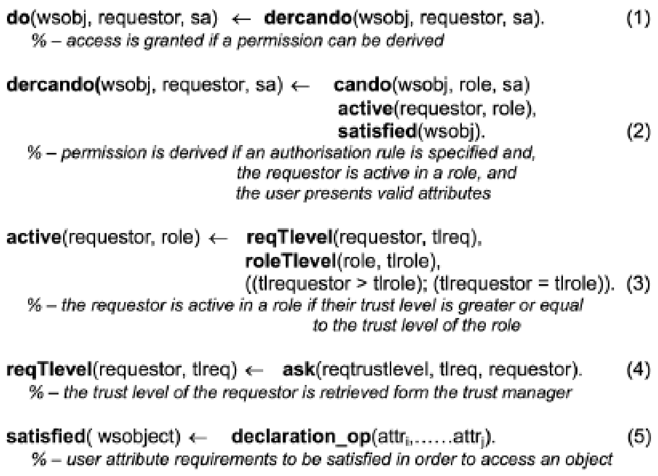

Interface policy
Functional interface
• Student number, institution, year of study and date of last registration
• Search-academic

Figure 3 *WSACT interface policy for access control requirements*



```
do(wsobj, requestor, sa)  ←  dercando(wsobj, requestor, sa).              (1)
   % – access is granted if a permission can be derived

dercando(wsobj, requestor, sa) ←   cando(wsobj, role, sa)
                                    active(requestor, role),
                                    satisfied(wsobj).                      (2)
  % – permission is derived if an authorisation rule is specified and,
                               the requestor is active in a role, and
                               the user presents valid attributes

active(requestor, role) ←   reqTlevel(requestor, tlreq),
                            roleTlevel(role, tlrole),
                            ((tlrequestor > tlrole); (tlrequestor = tlrole)).  (3)
  % – the requestor is active in a role if their trust level is greater or equal
                                          to the trust level of the role

reqTlevel(requestor, tlreq) ←   ask(reqtrustlevel, tlreq, requestor).     (4)
   % – the trust level of the requestor is retrieved form the trust manager

satisfied( wsobject) ←   declaration_op(attr_i,……attr_j).                 (5)
   % – user attribute requirements to be satisfied in order to access an object
```

Figure 4 *WSACT access control policy rules*

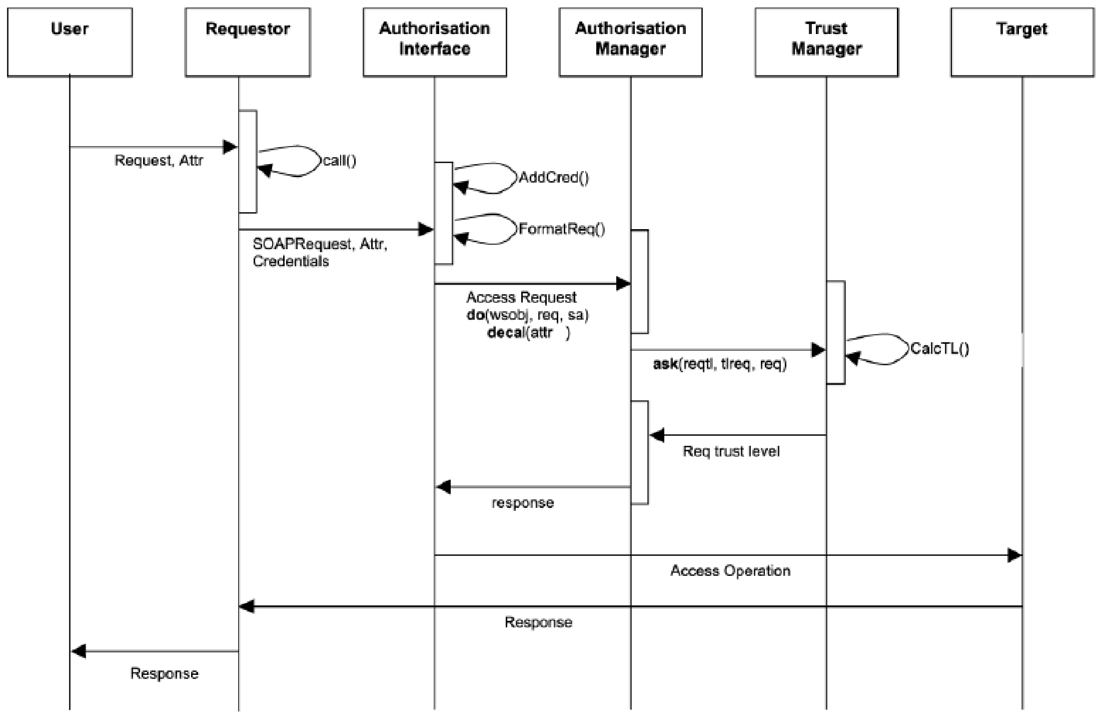
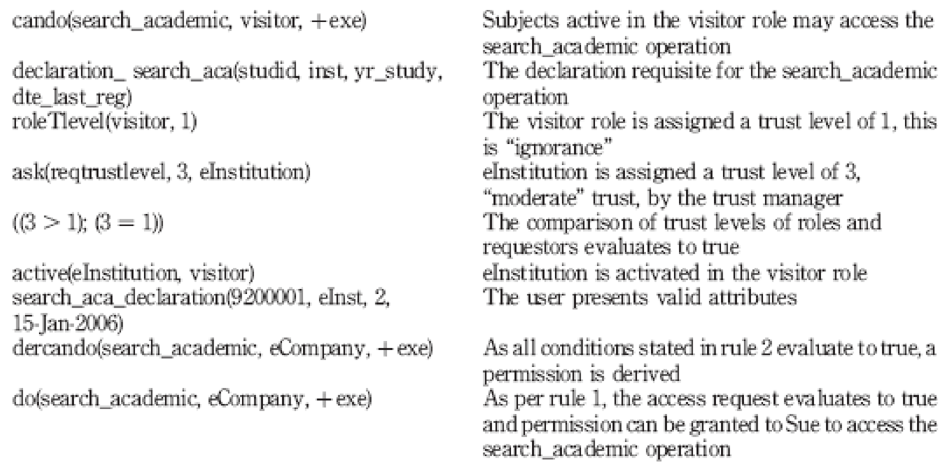


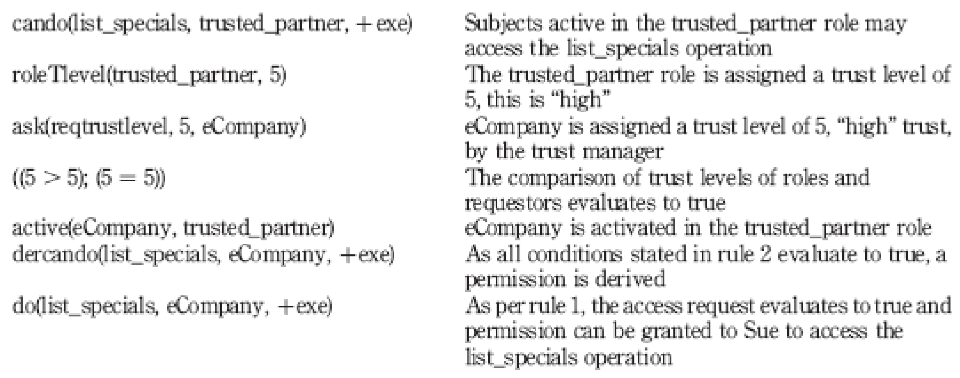
Figure 5 *Sequence diagram – access request for a target operation*



cando(search_academic, visitor, +exe)	Subjects active in the visitor role may access the search_academic operation
declaration_ search_aca(studid, inst, yr_study, dte_last_reg)	The declaration requisite for the search_academic operation
roleTlevel(visitor, 1)	The visitor role is assigned a trust level of 1, this is "ignorance"
ask(reqtrustlevel, 3, eInstitution)	eInstitution is assigned a trust level of 3, "moderate" trust, by the trust manager
$((3 > 1); (3 = 1))$	The comparison of trust levels of roles and requestors evaluates to true
active(eInstitution, visitor)	eInstitution is activated in the visitor role
search_aca_declaration(9200001, eInst, 2, 15-Jan-2006)	The user presents valid attributes
dercando(search_academic, eCompany, +exe)	As all conditions stated in rule 2 evaluate to true, a permission is derived
do(search_academic, eCompany, +exe)	As per rule 1, the access request evaluates to true and permission can be granted to Sue to access the search_academic operation

Table I

Materialisation of facts



cando(list_specials, trusted_partner, +exe)	Subjects active in the trusted_partner role may access the list_specials operation
roleTlevel(trusted_partner, 5)	The trusted_partner role is assigned a trust level of 5, this is “high”
ask(reqtrustlevel, 5, eCompany)	eCompany is assigned a trust level of 5, “high” trust, by the trust manager
((5 > 5); (5 = 5))	The comparison of trust levels of roles and requestors evaluates to true
active(eCompany, trusted_partner)	eCompany is activated in the trusted_partner role
dercando(list_specials, eCompany, +exe)	As all conditions stated in rule 2 evaluate to true, a permission is derived
do(list_specials, eCompany, +exe)	As per rule 1, the access request evaluates to true and permission can be granted to Sue to access the list_specials operation

Table II

Materialisation of facts

References

AMZI (2005), AMZI product page, available at: www.amzi.com/ (accessed 10 September 2005).

Atkinson, B., Della-Libera, G., Hada, S., Hondo, M., Hallam-Baker, P., Kaler, C., Klein, J., LaMacchia, B., Leach, P., Manferdelli, J., Maruyama, H., Nadalin, A., Nagaratnam, N., Prafullchandra, H., Shewchuk, J., Simon, D. (2002), Web Services Security (WS-Security), Version 1.0, 5 April, available at: www.verisign.com/wss/wss.pdf (accessed 10 March 2003).

Bacon, J., Moody, K. (2002), "Toward open, secure, widely distributed services", *Communications of the ACM*, Vol. 45 No.6, pp.59-64.

Ballinger, K., Box, D., Curbera, F., Davanum, S., Ferguson, D., Graham, S., Liu, K. (2004), Web Services Metadata Exchange (WS-MetadataExchange), available at: ftp://www6.software.ibm.com/software/developer/library/WS-MetadataExchange.pdf.

Barkat, B., Siyal, M.Y. (2002), "A novel trust service provider for Internet based commerce applications", *Internet Research*, Vol. 12 No.1, pp.55-65.

Bertino, E., Mevi, D., Squicciarini, A. (2004), “A fine-grained access control model for web services”, Proceedings of the 2004 IEEE International Conference on Services Computing (SCC'04), pp. 33-40.

Biskup, J., Wortmann, S. (2004), “Towards a credential-based implementation of compound access control policies, SACMAT 2004”, paper presented at 9th ACM Symposium on Access Control Models and Technologies, Yorktown Heights, New York, USA, 2-4 June.

Blaze, M., Feigenbaum, J., Ioannidis, J., Keromytis, A. (1999), The KeyNote Trust management System, version 2, IETF, RFC 3704, September.

Cahill, V., Jensen, C.D., Chen, Y., Gray, E., Seigneur, J. (2004), SECURE Framework Architecture (Beta), available at: www.cs.tcd.ie/publications/tech-reports/reports.04/TCD-CS-2004-07.pdf.

(2005), in Cantor, S., Kemp, J., Maler, E., Philpott, R. (Eds),SAML 2.0, available at: http://docs.oasis-open.org/security/saml/v2.0/.

Coetzee, M., Eloff, J.H.P. (2004), "Towards web services access control", *Computers and Security*, Vol. 23 No.7.

Coetzee, M., Eloff, J.H.P. (2005), "Autonomous trust for web services", *Internet Research*, Vol. 15 No.5, pp.498-507.

Coetzee, M., Eloff, J.H.P. (2006), “A framework for web services trust, SEC2006”, paper presented at 21st IFIP International Information Security Conference “Security and privacy in dynamic environments”, Karlstad University, Karlstad, Sweden, 22-24 May 2006.

Damiani, E., De Capitani Di Vimercati, S., Paraboschi, S., Samarati, P. (2001), “Fine-grained access control for SOAP e-services”, Proceedings of the 10th International World Wide Web Conference (WWW10), Hong Kong, 1-5 May.

Godik, S., Moses, T., Anderson, A., Parducci, B., Adams, C., Flinn, D., Brose, G., Lockhart, H., Beznosov, K., Kudo, M., Humenn, P., Andersen, S., Crocker, S. (2003), XACML 1.0 Specification, available at: www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml (accessed 10 February 2004).

Gottschalk, K., Graham, S., Kreger, H., Snell, J. (2002), "Introduction to web services architecture", *IBM Systems Journal*, Vol. 41 No.2.

Guerin, R., Pendarakis, D., Yavatkar, R. (2000), RFC 2753 – a framework for policy-based admission control, available at: www.faqs.org/rfcs/rfc2753.html (accessed 6 June 2005).

Hess, A., Jacobson, J., Mills, H., Wamsley, R., Seamons, K.E., Smith, B. (2002), “Advanced client/server authentication in TLS”, Proceedings: Network and Distributed System Security Symposium, San Diego, California, 6-8 February 2002.

ISO (1996), ISO 10181-3 Access Control Framework, available at: http://iso.nocrew.org/iso/en/CatalogueDetailPage.CatalogueDetail? csnumber=18199 (accessed 10 November 2006).

Jajodia, S., Samarati, P., Subramanian, V.S. (1997), “A logical language for expressing authorisations”, Proceedings of the 1997 IEEE Symposium on Security and Privacy, Oakland, CA.

Koshutanski, H., Massacci, F. (2003), “An access control framework for business processes for web services”, Proceedings of the 2003 ACM Workshop on XML Security, Fairfax, VA.

Li, N., Mitchell, J.C. (2003), “Datalog with constraints: a foundation for trust-management languages”, Proceedings of the Fifth International Symposium on Practical Aspects of Declarative Languages (PADL 2003), Vol. 2562 of Lecture Notes in Computer Science, Springer-Verlag, New York, NY, pp. 58-73.

Lopez, J., Maña, A., Yagüe, M. (2005), "A metadata-based access control model for web services", *Internet Research*, Vol. 15 No.1, pp.99-116.

Miao, L., He-Qing, G., Jin-Dian, S. (2005), “An attribute and role based access control model for web services”, International Conference on Machine Learning and Cybernetics, Vol. 2, pp 1302-6. MS ASP.NET (2005), ASP.NET resources, available at: http://msdn.microsoft.com/asp.net (accessed 21 September 2005).

MS VB.NET (2005), Visual Basic resource, available at: http://msdn.microsoft.com/vbasic/default.aspx (accessed 21 September 2005).

Olson, L., Winslett, M., Tonti, G., Seeley, N., Uszok, A., Bradshaw, J.M. (2006), “Trust negotiation as an authorization service for web services”, ICDE Workshops 2006, p. 21.

Rivest, R., Lampson, B. (1996), “SDSI – a simple distributed security infrastructure”, October, available at: http://research.microsoft.com/lampson/59-SDSI/Webpage.html (accessed 21 September 2006).

Shen, H., Hong, F. (2006), “An attribute-based access control model for web services”, Proceedings of the 7th International Conference on Parallel and Distributed Computing, Applications and Technologies, pp. 74-9.

Winslett, M. (2002), "An introduction to trust negotiation", in Nixon, P., Terzis, S. (Eds),*Proceedings of the First International Conference, iTrust Heraklion, Crete, Greece, 28-30 May*, Springer-Verlag, New York, NY.

Wonohoesodo, R., Tari, Z. (2004), “A role based access control for web services”, Services Computing, IEEE International Conference on (SCC'04), pp. 49-56.

Corresponding author

Marijke Coetzee can be contacted at: marijkec@uj.ac.za