A LOGIC-BASED ACCESS CONTROL APPROACH FOR WEB SERVICES

Author and co-author

M. Coetzee, J. H. P. Eloff

Author's affiliation
Department of Computer Science, University of Pretoria

Author's contact details
mcoetzee@cs.up.ac.za, (012) 420-4035
eloff@cs.up.ac.za, (012) 420-2361

ABSTRACT

Web Services technology enables organisations to exploit software as a service. Services are accessed by method invocations. Method interfaces are described and published, and may be freely available. In Web Services environments, access control is required to cross the borders of security domains, to be implemented between heterogeneous systems. Interaction is between remotely located parties who may know little about each other. Access control generally assumes that identity is established. To overcome the limitations of identity-based solutions, domain-independent access control information is added to a message. As a Web Service endpoint is required to integrate such information into its access control decision-making process, issues arise such as; whom to accept access control information from; what the format of such information must be; how to inform the requestor of the format; and how to give access to methods based on presented access control information. To address such issues, a logic-based access control approach is defined for a Web Service endpoint. A logic-based authorization manager is described, that provides a formal foundation of logical reasoning, to enable the enforcement of consistent access control decisions over the resources of Web Services.

KEY WORDS

Web Services, access control, SOAP, assertions, logical rules, access control requirements

A LOGIC-BASED ACCESS CONTROL APPROACH

FOR WEB SERVICES

1 INTRODUCTION

Service-oriented computing (SOC) is the computing paradigm that uses services as fundamental elements for developing applications. [GEOR03] A service is a function that is well defined, self-contained, that does not depend on the context or state of other services.

Web Services technology [GOTT02] provides organizations with the ability to exploit this paradigm by enabling location and connection independent services. A Web Service is a type of service, identified by a URI that can be invoked through an Internet connection. SOAP (Simple Object Access Protocol) [BOXD00] is the primary transport mechanism used to convey method requests and responses. Services are offered by organizations that support the service implementation, and publish the service description. Requestors interact with services from distant, independent locations, as the necessary information for discovery, selection and binding is made available. Additional quality of service attributes such as performance, security, and reliability are required, although they are not currently addressed. As SOAP requires no additional ports or access mechanisms beyond those used in Web servers, each SOAP message can potentially be seen as a possible security threat. Reliable access control is therefore a fundamental requirement to the acceptance of Web Services by organisations.

Web Services demand a security framework for access control that does not impede the exchange of data essential for their success. Languages such as WSDL represent a valuable tool for the description of functionality but not for security properties. Therefore, secure interoperability requires additional mechanisms.

Assertion-based access control provide scalable and flexible access control since access conditions are expressed in terms of sets of attributes instead of users or groups. This approach scales well for the number of users that may exist, as well as the number of assertions used by the access control system. Assertions provide means to transport authorisation information to decentralized applications. In this way, authorisation information becomes mobile and interoperable, which is highly convenient for applications defined with Web Services. The need of a domain to authenticate users from other domains, who require access to local resources, is removed.

The implementation of access control within distributed environments has always been complex and difficult. Factors such as the size of a distributed user population, heterogeneity of participants and the autonomy of access control systems of participants all increase complexity. There is also no standard technique defined to enforce access control for SOAP messages in Web Services environments, although much work is in progress to address this issue. In such environments, logical methods may play an important role.

The major contribution of this paper is to present a logical approach for access control for service-oriented computing environments. For the purposes of this paper, the discussion is narrowed down to address the access control decision made by an independent Web Service endpoint. This paper is structured as follows: in the remainder of the introduction, related work is discussed. Section 2 presents the logical access control approach. An autonomous authorisation interface and authorisation manager is described. Section 3 concludes the paper.

1.1 Related work

In order to address cross-domain movement of users, a move to attribute-based access control is a central theme found in distributed access control research [BONA02] [CHAD02] [ASHL00]. The need of a domain to authenticate users from other domains, who require access to local resources, is removed. The ability, rather than the identity of the user becomes important. Before attributes can be accepted across domains, some form of trust must be established. [FOST01] [BACO02]. In Akenti [JOHN98], a trust-management system, a combination of authenticated X.509 identity certificates, and distributed digitally signed authorization policy certificates are used to make access decisions about distributed resources. Partner's policies are collected by an authorization engine that makes an access decision. For such systems, access control rules are expressed in terms of sets of attributes. Both unsigned declarations, or credentials digitally signed by a trusted authority may be used for such purposes [BONA02]. They are collectively known as assertions, or statements of fact.

A recent access control approach that can be used for Web Services is presented in the OASIS XACML specification [ANDE03]. It is based on an extension of XML and supports user credentials and context-based privilege assignment. It does not directly support role-based access control, and thus lacks features such as separation of duty constraints and role hierarchies. A major shortcoming of the XACML architecture is that all access control policies that are referenced must be expressed in XML, with XACML syntax. An architecture is defined that separates a policy decision point (PDP) from policy enforcement points (PEPs). The specification does not address the design of the policy decision point.

2 A LOGIC-BASED ACCESS CONTROL APPROACH FOR WEB SERVICES

The service-oriented computing paradigm is based upon the interactions between Web Service providers and Web Service requestors [COYL02], as shown in figure 1. A typical Web Service access control scenario is illustrated in figure 1. The remote user is a member of a security domain that allows its users to access Web Services, exposed by other domains. The application that performs the access is referred to as a requestor. The requestor authenticates the user before access to its resources is allowed. The user selects an option displayed on a page on his browser, for the execution of a method of a Web Service, defined in another security domain. On behalf of the user, the requestor invokes the selected method. The invocation is sent in a SOAP message to the Web Service provider. The Web Service provider resides on a Web or Application server that is network accessible. Effectively protecting methods such as *Search, ExpediteOrder and PlaceOrder*, exposed by a Web Service provider in a distributed Web Service environment is difficult with current solutions. The next paragraph defines a logic-based access control approach for a Web Service provider to address the effective protection of Web Service methods. Firstly, the architecture for access control is defined; thereafter the operation of access control is discussed.

2.1 Access control architecture

A general overview of the main components of the architecture is depicted in figure 1. There are two main components: an authorisation interface, and an authorisation manager. In addition, published access control requirements of the service endpoint are necessary for interoperation between the requestor and provider.

2.1.1 Published access control requirements

Providers and requestors of Web services are independent business entities that should have complete control over their resources. Access control interoperation between such independent security domains is required. When access control rules, passed to a Web services provider by a requestor, are allowed to complement or influence access control decisions of the provider, this should not result in the loss of control over local independence. To allow interoperation, Web services providers should convey to requestors their expected access control requirements in order to invoke proper access control decisions, as shown in figure 1. A central question is determining what to publish, without compromising the service provider's access control policy.

2.1.2 Authorisation interface

A SOAP request to invoke a Web Service method is sent by a requestor. The SOAP request is accompanied by assertions in order to gain access. Assertions have been defined, based on published access control requirements. As the SOAP request reaches the provider, it is intercepted at the Web or Application Server that hosts the method that is requested. This is performed by an authorisation interface as depicted in figure 1. The authorisation interface acts as policy enforcement point. It protects the authorisation manager from malicious or malformed requests. Only valid requests are passed to the authorisation manager that acts as a policy decision point. The authorization interface contains a verification module that ensures the integrity of assertions that are received. Requestors use access control policy requirements and a service provider-defined schema to define assertions. In the verification process, the XML parser checks for well-formed and valid assertions. Further checks are made to ensure no malicious information is introduced. Accompanying identifying keys are checked to ensure their authenticity. Thereafter, a transformation module converts XML-based assertions into logical facts, by parsing the assertions in DOM, together with appropriate logic. Logical facts are created that are to be added to the policy base, defined in figure 1. Finally, the received SOAP request is inspected and an access request formulated which includes the method to be executed. The access request and logical facts are sent to the authorization manager.

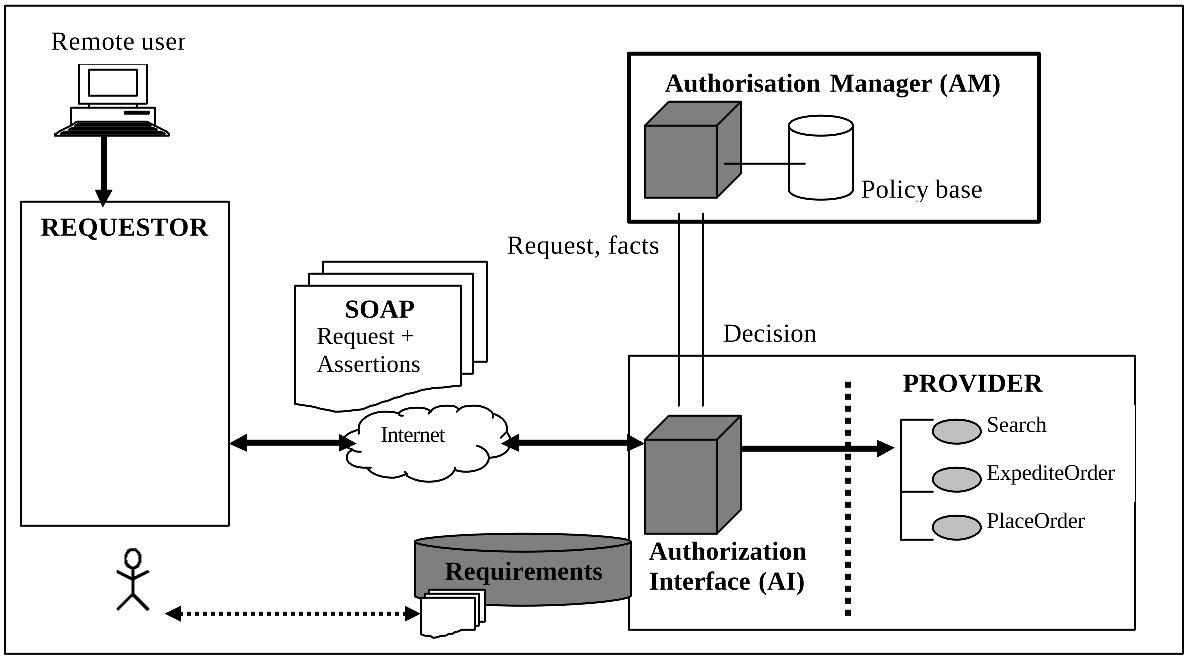

Figure 1: Authorisation architecture

2.1.3 Authorisation manager

The core component of the authorisation framework is the authorisation manager that mediates all access requests. An important feature of the authorisation manager is that its location is separated from where the access control policy is enforced. The authorization manager consists of two parts: a policy base and inference engine. Logical facts are added to the policy base. Thereafter, access permission (permit/deny) is generated, based on the access request, the access control policy of the service endpoint defined in the policy base and the composition of the access control policy with new facts that are added to the policy base at run time.

2.2 The operation of the Web Service access control system

The focus of the next paragraphs is to show how a method such as *PlaceOrder* at a service provider is protected so that only authorised remote users are given access to it. The structure of the SOAP request is defined in order to show how it can be used to convey access control information to the Web Service provider. A Web Service provider is required to integrate such access control information into its access control decision-making process. It should know whom to accept access control information from; what the format of such information must be; how to inform the requestor of the format; and how to give access to methods based on presented access control information.

2.2.1 The structure of the SOAP request

The incoming SOAP request from the requestor is routed to the *PlaceOrder* method at the provider. The message that is conveyed to the Web Service endpoint in SOAP is shown by figure 2. For the sake of clarity, not all message details are shown. The SOAP message contains three different sections. Their semantics are defined with associated XML schemas, which the receiving SOAP processor should be able to interpret. The SOAP envelope element defines the start and end of the message. The SOAP body element contains the method name and associated parameters in either the SOAP request or response. The SOAP header element, which is optional, allows application specific data, not directly associated with a method request or response to be passed to the service endpoint.

```
<soap:Envelope
     xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
      <soap:Body xmlns:m="http://www.provider.com/orders">
            <m:PlaceOrder>
                  <m:StockName>XE2234 Laptop</m:StockName>
            </m:PlaceOrder >
      </soap:Body>
</soap:Envelope>
```

Figure 2: The SOAP request for the PlaceOrder method

The *PlaceOrder* method at the provider is exposed, and can be called by any requestor. An access control system is required to protect all methods exposed by the provider.

2.2.2 Publish access control requirements

If an access control system is defined by the provider, administrators of requestor applications need to understand its requirements, and how it is to be used. To enable this, the Web Service provider must publish well-defined access control requirements in policy statements. This will reduce the interdependencies between a requestor and Web Service provider. Generally, requestors may not be limited to only one Web Service provider, but may be able to select the most secure service, based on the manner in which access control and other security services are implemented.

WSDL supports the description of method interfaces, but not any access control or other security requirements. WS-Policy [BOXD03] provides a grammar for requestors and Web Service providers to communicate their requirements and capabilities in a machine-readable XML format. WS-Policy defines the security requirements related to authentication, integrity and confidentiality of SOAP messages. No expectations in terms of access control are currently described. An extension to WS-Policy is required that would provide all or some of the access control policy requirements to potential requestors so that they can make decisions regarding if and how they can use a service. An optional <AccessControlPolicy> element can be defined to publish such requirements. The requestor would need to understand the semantics of the published access control requirements. Currently, only humans can determine or decipher published access control semantics through a static inspection at design time as depicted in figure 1. Careful consideration should be given to the requirements that are exposed, as they may lead to a security domain being exploited.

The provider service may have specific access control requirements for each exposed method that must be complied with. The access control system of the provider is not interested in the real identity of a remote user, but rather in his ability, as asserted by the requestor, who must be a trusted authority. For the *PlaceOrder* method to be invoked successfully, it may require of requestors to make assertions about remote users such as their credit cards details, and employee or customer identification number. The definition of such access control requirements is shown in figure 3.

The provider service may provide an alternative to the normal ordering process. An expedited order process may be made available to employees of requesting companies that are at management level. This benefit is given to the management of a requestor, in order to improve the relationship with the service provider. For this method to be invoked, an additional assertion on the seniority of an employee must be provided by the requestor. Based on provided assertions, dynamic access is granted to either the *PlaceOrder* or *ExpediteOrder* methods.

```
<AccessControlPolicy>
  <DeclarationPlaceOrder>
     <!-- Credit card details of user -->
        <CreditCard>
            <CreditCardNumber> </CreditCardNumber>
            <ExpiryDate> </ExpiryDate>
            <Issuer> </Issuer>
        </CreditCard>
     <!-- ID of user, eg Customer Number or Employeed ID -->
        <IDNumber> </IDNumber>
     <!-- Public key of requestor to verify authenticity of request -->
        <PublicKey> </PublicKey>
  </DeclarationPlaceOrder>
</AccessControlPolicy>
```

Figure 3: An extension of a WS-Policy requirements description in XML

2.3 Formulate assertions

If assertions were passed with each method invocation, it would require of each method to perform an authorisation check before it renders its service. If assertions are passed in the SOAP header, the flexibility of the design is increased, as authorisation logic is separated from application logic. Authorisation logic can be modularised for all methods that are exposed. The semantics of the header containing the assertions may be defined by the Web Service provider at http://schemas.provider.com/orderHeader, in order to allow requestors to formulate valid requests.

```
<soap:Envelope
     xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
      <soap:Header
      <h:AssertionInfo
         xmlns:h=http://schemas.provider.com/orderHeader
           <!-- assertions -->
            <h:CreditCard>
                <h:CreditCardNumber>9987334566785</h:CreditCardNumber>
                <h:ExpiryDate>0506</h:ExpiryDate>
                <h:Issuer>VISA</h:Issuer>
            </h:CreditCard>
             <h:IDNumber>8894</h:IDNumber>
           <!—public key of requestor -->
             <h:publickey> . . . <h:/publickey>
      </h:AssertionInfo>
      </soap:Header>
      <soap:Body xmlns:m="http://www.provider.com/orders">
            <m:PlaceOrder>
                 <m:StockName>XE2234 Laptop</m:StockName>
           </m: PlaceOrder >
      </soap:Body>
</soap:Envelope>
```

Figure 4: A request with a header containing required assertions.

Based on inspected requirements, developers of the requestor portal create application code that would enable the remote user to formulate a request to invoke the *PlaceOrder* method. A valid SOAP header is added to the message, containing his required assertions. Assertions may be sourced directly from the remote user, or may be kept in for instance, a corporate LDAP directory.

The modified message, that includes the SOAP header with assertions for the remote user, is shown in figure 4.

Message confidentiality and integrity can be assured by sending the SOAP message with SSL. If required, the credit card number of a remote user can be encrypted with XML encryption to further protect it. The public key of the requestor accompanies the assertions, in order to verify their authenticity.

2.4 Mediate an access request

When SOAP requests for methods are mediated by the authorisation manager, a logic-based access control system provides a formal foundation of logical reasoning, to enable the enforcement of consistent access control decisions over the resources of Web Services. For the purposes of this research project, the authorisation manager is defined in Prolog, a logical programming environment. The Prolog inference engine has the potential to provide a mechanism to derive consistent access control decisions at runtime. It may also be used to analyse the correctness and consistency of access control and other rules. A policy base representing the access control policy of the Web Service provider is defined. Access control rules, defined in Prolog, can be more expressive than the traditional (subject, object, action) tuple. The access control policy is machine-readable and directly under the control of the administrator. It can be modified or replaced, without affecting Web Service methods or other application code. It also possesses dynamic updating capabilities. New facts from independent policy sources can thus be added to the policy base before decisions are made, ensuring dynamic decisions at runtime.

The authorisation manager has the task of determining whether access to a Web Service method must be granted or denied. An access control decision is made, based on four aspects: the formulated request, the relationship of trust with the requestor, the assertions of the remote user that accompany the request and the access control rules that have been defined. Each of these aspects will now be defined in more detail.

2.4.1 A formulated request

To invoke the access control logic of the authorisation manager, an access request is formulated. The SOAP request from the requestor is intercepted by the authorisation interface, before it reaches the method it is destined for at the Web Service provider. The SOAP message is inspected and its parameters extracted. The method to be accessed is *PlaceOrder*. A logical access request is formulated with the **access** predicated, as shown below.

access(PlaceOrder)**?**

2.4.2 The relationship of trust with the requestor

An access decision is made, based on assertions that accompany a request. As assertions can only be accepted from trusted requestors, the authorisation manager needs to establish whether a relationship of trust exists with the requestor in its policy base. A relationship of trust is created when the initial requestor registration process is performed. The requestor submits a credential in the form of an identifying digital certificate to the provider. The public key, $K_{Requestor}$, is used as a means to verify its identity and to facilitate a relationship of trust. This is an additional fact that is added to the policy base, which is depicted in figure 1.

trust(Requestor**,** $K_{Requestor}$).

If the provider loses trust in the requestor, the statement can immediately be removed from the set of facts, and no further permissions would be derived for them. If, on the other hand, such a fact can be found, the logical decision-making process can proceed.

2.4.3 The assertions of the remote user that accompany the request

If a relationship of trust exists, assertions from a requestor can be imported into the policy base. The authorisation interface translates the XML assertions found in the SOAP header of the request into a logical statement. The **request** predicate is used to distinguish imported facts from local facts. R **request** F means that the requestor R requests formula F to be added as a new fact. The assertions are identified by the **assert** predicate. The public key of the requestor that accompanies these assertions is also used in the rule to ensure the authenticity of the request. If credentials are used from trusted authorities other than the requestor, additional verification of the key of each issuing authority must be performed. The next clause shows how the credit card and employee identification number of the remote user, defined in figure 4, are imported as facts into the policy base.

request(requestor(Requestor**,** $K_{Requestor}$), **assert**(**cc**(9987334566785, 0506, VISA), **id**(8894))).

2.4.4 The access control rules that are defined

Access control rules are required that would protect methods and other resources exposed by a Web Service interface. The specific subjects, objects and actions to be defined are:

Objects: Objects are the service and its methods to be accessed. Access control is required over input and output parameters, methods, services and collections of services and other applications [KRAF02].

Subjects: The requestor of the service is an active subject that acts on behalf of the remote user. Consideration should be given to grouping subjects to facilitate administration.

Signed actions*:* Remote users or applications would generally be allowed to execute the methods of a service. Permission to access a service is either granted (+) or denied (-).

When defining an access control policy, role-based access control (RBAC) can be useful as it reduces the administration burden [SAND96]. RBAC requires access permissions to be assigned to roles, rather than individual users, and users obtain permissions by virtue of being assigned appropriate roles. Possible authorisation rules are defined by administrators with the following clauses:

cando(PlaceOrder, general, +exe).

cando(ExpediteOrder, management, +exe).

These two rules are added to the policy base, shown in figure 1. The first rule assigns to the “general” role permission to execute the *PlaceOrder* method. The second rule assigns to the “management” role permission to execute the *ExpediteOrder* method.

Here, role-based access control mechanisms are extended, to map unknown remote users to roles defined by a Web Service endpoint. Access control decisions cannot be based on the identity of the remote user, but rather on the ability, through assertions that are presented. Assertions presented on behalf of a user may also vary from request to request. Access control is thus dynamic, as it is context-dependent.

In order to achieve role mapping, a logical rule is defined for each role, which includes all assertions that must be presented. A role is dynamically activated for requestors based on imported assertions with the **active** predicate. The access control policy of the Web Service provider is given a measure of protection, as roles are kept private. Requestors understand how access to a Web Service method, but do not know how access control rules are evaluated.

For instance, to activate a trusted requestor in the “general” role requires that the credit card details and employee-id be presented on behalf of the remote user. A rule is defined as follows:

active(Requestor, "general")**:-**

trust(Requestor, $K_{Requestor}$)**,**
request(**requestor**(Requestor**,** $K_{Requestor}$), **assert**(**cc**(ccnumber, exdate, issuer), **id**(id)))**.**

To activate a trusted requestor in the “management” role, an additional assertion on the seniority of the employee must be presented on behalf of the remote user.

active(Requestor, "management")**:-**
trust(Requestor, $K_{Requestor}$)**,**
request(**requestor**(Requestor, $K_{Requestor}$), **assert**(**cc**(ccnumber, exdate, issuer), **id**(id), **sen**(sen)))**.**

Finally, a decision is inferred, based on permissions that have been assigned to the activated role as follows:

access(Object)**:-**
active(Requestor, Role)**, cando**(Object, Role, SignA)**.**

Figure 5 summarises all predicates used in the definition of the policy base. .

cando(Object, Role, SignA).--1

trust(Requestor, K_R).--2

request(requestor(Requestor, K_R), **assert**(attr1, attr2, attr3)). -------------------3

active(Requestor, Role)**:-**
trust(Requestor, Key)**,**
request(**requestor**(Requestor, K_R), **assert**(attr1, attr2, attr3))**.**-------------4

access(Object)**:-**
active(Requestor, Role)**,**
cando(Object, Role, SignA)**.**---5

Figure 5: Predicates used by the policy language

2.5 Infer a decision for access(PlaceOrder)?

Based on the facts and rules defined in the policy base, a decision is inferred for **access**(PlaceOrder)? A relationship of trust is found with the requestor and the presented assertions are evaluated. Based on these assertions, the requestor is activated in the “general” role. As the PlaceOrder method may be accessed by any requestor in the “general” role, the access request evaluates to TRUE. The authorisation manager returns a “permit” decision to the authorisation interface, to allow the SOAP request to invoke the PlaceOrder method. Otherwise, a “deny” decision is returned. The authorisation interface returns a fault to the requestor.

2.6 Management of imported facts

Real-time access control decisions made by the inference engine are processed in parallel. As assertions are imported for specific requests, request identifiers must be added to assertions, in order to distinguish them from each other. Another important consideration would be the management of the lifespan of imported facts. Imported facts must be removed from the policy base after they have been used. Access control decisions become dynamic, as facts are added and removed from the policy base.

3 CONCLUSION

In this paper, a logic-based access control approach was defined, to create a flexible, loosely coupled access control solution for a Web Service endpoint. As interactions with services may change frequently, or may exist for very limited periods with a limited number of transactions, they should be composed quickly, with embedded flexibility.

The approach defined here standardised interactions between requestors and Web Service providers, in order to decouple them from each other. This was done by adding assertions defined in XML, to SOAP headers. The dynamic access control policy of the autonomous authorization manager was logically defined with Prolog, allowing it to be available for real time access control decisions. It is independent of the access control system of the requestor, as its expectations are published. The authorization manager assigns roles to requestors with changing user profiles, based on trusted assertions. Access to service methods is only granted if permission can be derived for it, where the derivation step forms a formal proof. To ensure the safety of the authorization manager, it is located away from the authorisation interface.

A very simple example of the composition of requestor assertions with the access control policy of the Web Service provider was illustrated. Future research would include the composition of the Web Service provider access control policy, with complex declarations and credentials that may allow the access request to be upgraded.

4 ACKNOWLEDGEMENT

The financial assistance of the Department of Labour (DoL) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the DoL.

5 REFERENCES

[ANDE03] Anderson A. et. al., XACML 1.0 Specification, 2003, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml

[ASHL00] Ashley. P, Au, R., Looi, M., and. Cross-domain one-shot authorization using smart cards. In *Proceedings of the 7th ACM conference on Computer and communications security* (2000), ACM Press, pp. 220–227.

[BACO02] Bacon, J., and Moody, K. Toward open, secure, widely distributed services. *Communicationsof the ACM 45*, 6 (2002), 59–64.

[BONA02] Bonatti P., Samarati P., A unified framework for regulating access and information release on the Web. *Journal of Computer Security*, vol. 10, n. 3, 2002, pp. 241-272

[BOXD00] Box D., Ehnebuske D., Kakivaya G., Layman A., Mendelsohn N., Nielsen H. F, Thatte S. and Winer D., “Simple Object Access Protocol (SOAP) 1.1”, http://www.w3.org/TR/SOAP/, May 2000

[BOXD03] Box D., Web Services Policy Framework (WS-Policy), http://www.ibm.com/developerworks/library/ws-policy/index.html

[CHAD02] Chadwick, D. W., and Otenko, A. The PERMIS X.509 role-based Privilege management infrastructure. In *Seventh ACM Symposium on Access Control Models and Technologies* (2002), ACM Press, pp. 135–140.

[COYL02] Coyle F. P., XML, Web Services and the data revolution, Addison-Wesley, 2002

[FOST01] Foster I., Kesselman C., Pearlman L., Tuecke S., Welch V., A community authorization service for group collaboration, www.globus.org/research/papers/CAS_2002_Revised.pdf

GEOR03] Georgakopoulos D., Papazoglou M.P., Service-oriented computing, *Communications of the ACM*, Oct 2003, Vol 46, no 10, pp 25-28.

[GOTT02] K. Gottschalk, S. Graham, H.Kreger and J.Snell, Introduction to Web services architecture, IBM Systems Journal, Volume 41, Number 2, 2002

[JOHN98] Johnston, W., Mudumbai, S., and Thompson, M. Authorization and attribute certificates for widely distributed access control. In *Proceedings of Seventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises,* 1998, IEEE Press, pp. 340–345.

[KRAF02] R. Kraft, A Model for Network Services on the Web, Proceedings of the 3rd International Conference on Internet Computing, IC 2002, 3:536:541, June 2002, Las Vegas

[SAND96] Sandu R., Access control: The neglected frontier, Proceedings of the 1st Australian conference on Information Security and Privacy, Wollongong, Australia, June 23-36, 1996