Computers & Security (2004) **23**, 559–570

ELSEVIER

Computers & Security

www.elsevier.com/locate/cose

Towards Web Service access control

M. Coetzee, J.H.P. Eloff*

Department of Computer Science, University of Pretoria, Pretoria, South Africa

Received 17 October 2003; revised 18 March 2004; accepted 4 May 2004

KEYWORDS
SOAP;
XML;
Access control;
Web Services;
Assertions;
Authorisation manager;
Logical rules;
Roles;
Trust

Abstract The Internet has revolutionised the capacity to share information and services across organisations. Web Service 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. Method requests and responses are conveyed in SOAP, which has the ability to pass unhindered through firewalls. Applications that process SOAP requests may be endangered by messages with malicious intent. Protection of methods and resources exposed by SOAP is thus a critical requirement for Web Services to be acceptable to organisations. In Web Service environments, access control is required to cross the borders of security domains, to be implemented between heterogeneous systems. New approaches are required that would address the movement of unknown users across borders so that access to resources can be granted. Specifications have been released to address access control, but are not well established. In this paper, an analysis of current approaches to Web Service access control is made, which leads to five requirements to be addressed by future access control solutions. To address such requirements, a logic-based access control approach is defined for a Web Service endpoint. The paper does not address the access control logic that is required when more than one Web Service is used in an integrated business solution.

Introduction

The Internet has revolutionised the capacity to share information and services across organisations. Web Service technology (Gottschalk et al., 2002) has enabled a major step forward in cross-domain organisational cooperation. Web Services are derived from emerging standards that describe a service-oriented architecture on the Internet, enabling electronic exchange to take place in a distributed network environment.

A Web Service is the name of an object with methods that can be invoked through an Internet connection. SOAP (Simple Object Access Protocol) (Box et al., 2000) is the primary transport mechanism used to convey method requests and

* Corresponding author.
E-mail addresses: mcoetzee@cs.up.ac.za (M. Coetzee), eloff@cs.up.ac.za (J.H.P. Eloff).

0167-4048/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cose.2004.05.006

responses. It is based on HTTP (Hyper Text Transfer Protocol) (Fielding et al., 1997) and XML (Extensible Markup Language) (Bray et al., 2000). When used with WSDL (Web Service Definition Language) (Christensen et al.), and UDDI (Universal Description, Discovery and Integration) (Atkinson et al., 2003), Web Services are easily found and integrated into the applications of other organisations.

The deployment of Web Services presents new security problems for IT departments of organisations. SOAP requires no additional ports or access mechanisms beyond those used in Web servers, which are found in almost every organisation. Each SOAP message can potentially be seen as a possible security threat, as it presents itself just as normal Web traffic would. SOAP messages may be treated by firewalls as simple HTTP requests for Web pages, resulting in possible unauthorised access to the internal applications behind the firewall. Applications that are to process the requests contained by SOAP messages may be endangered by false claims or malicious information. Organisations would require predictable and reliable operation from their applications, and the interaction of their applications with those of their business partners. Reliable access control is therefore a fundamental requirement for the acceptance of Web Services by organisations.

Current access control solutions for Web Services are not ideal, as an integrated configuration and development effort is required across security domains, in order to protect methods and resources that are exposed. A complicating factor is the independence of partners, as similar access control capabilities cannot be adopted. As Web Services are used in unexpected ways, by any number of unknown partners, access control solutions are required to be flexible and loosely coupled.

In a Web Service environment, 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, to make it self-contained. This can enable a Web Service endpoint to make an informed access control decision. 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. Custom defined solutions can be error-prone, as an omission or misinterpretation of a communication from a partner may lead to improper access to resources.

The focus of this paper is to define an access control approach for Web Services that addresses the mentioned issues. The paper is structured as follows. Next section provides a brief overview of recent research on access control in distributed computing environments. Subsequent section shows how access control is currently implemented for Web Service environments. An analysis of current approaches leads to five requirements to be addressed by future access control solutions. Generally, access control for Web Services can be addressed on two levels of abstraction. Each independent Web Service endpoint receives a request, and makes a decision over access to its methods and resources. When more than one Web Service is used in an integrated business transaction, a next level of access control logic is required. As a decision made by a Web Service endpoint influences the flow of control of a transaction, independent partner decisions must be orchestrated. For the purpose of this paper, the discussion is narrowed down to address the case of the access control decision made by an independent Web Service endpoint. A logic-based access control approach for a Web Service endpoint is presented next. The five requirements that have been identified in the previous section are addressed in the approach that is presented. Final section concludes the paper.

Distributed access control

Access control is a core requirement for any information system, in order to prevent malicious attacks. Access should only be allowed to authorised users. This is presently ensured by the joint use of authentication and access control mechanisms, in both centralised and distributed systems. The development of service-oriented architectures requires cross-domain access control between heterogeneous systems. Access control architectures for similar endeavors have been the focus of recent research, with results that may be useful for Web Service access control.

In distributed access control architectures, it is generally found that access control logic is separated from application logic (Bacon and Moody, 2002; Beznosov et al., 1999; Damiani et al., 2001; Lam and Woo, 1993). This simplifies access control logic and reduces the cost of administration. For instance, in a distributed object-oriented architecture such as CORBA, a user is first authenticated.

The result of authentication is a set of security-related data that is stored in a `Credentials` object. When the user makes a method invocation, the request is intercepted by an `AccessDecision` object, which grants or denies permission for the object invocation. The decision is based on evaluation of the `Credentials` object, which indicates the user's permissions. All participants in this environment are CORBA enabled.

In order to address cross-domain movement of users, a move to attribute-based access control is another central theme found in distributed access control research (Ashley et al., 2000; Bonatti and Samarati, 2002; Chadwick and Otenko, 2002). 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 (Bacon and Moody, 2002; Foster et al.). In Akenti (Johnston et al., 1998), a trust-management system, a combination of authenticated X.509 identity certificates, and distributed digitally signed authorisation policy certificates are used to make access decisions about distributed resources. Partner's policies are collected by an authorisation engine that makes an access decision. KeyNote (Blaze et al., 1999) is another trust-management system, where a credential, signed by a trusted authority, asserts attributes about a user. X.509 certificates are deliberately not required, removing the need for maintaining a PKI. 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 (Bonatti and Samarati, 2002). They are collectively known as assertions, or statements of fact. An XML standard specification for security assertions named Security Assertion Markup Language (SAML) (Hallam-Baker et al., 2003) has recently been defined. SAML defines both XML protocols and assertion structures.

A recent access control approach that can be used for Web Services is presented in the OASIS XACML specification (Anderson et al., 2003). 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.

Before an access control approach is defined that may incorporate some of these re-occurring themes, an analysis of current platform-dependent access control solutions for Web Services will be made in the next section.

Existing access control for Web Services

The service-oriented computing paradigm is based on the interactions between Web Service providers, Web Service brokers and Web Service requestors (Coyle, 2002). Web Service providers define simple business-to-business interfaces, in order to allow exchange of SOAP messages with Web Service requestors. Interfaces can be made available through central Web Service brokers.

There is currently no standard, agreed-upon method for exposing Web Service methods over the Internet in such a way that only authorised users can call them. A typical Web Service access control scenario is illustrated in Fig. 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. To be able to make access control decisions, the Web Service provider may require information about the user.

The Web Service provider resides on a Web or Application server that is network accessible. Platform-dependent access control mechanisms exist that can be used to protect resources. Identity-based access control can be implemented for SOAP requests by using the existing infrastructure of the underlying HTTP protocol. A Web Service provider can use basic authentication over HTTPS, where the requestor adds a username and password to the request. Access to methods is granted to authorised users belonging to groups or roles with defined permissions. In addition, the requestor making the request on behalf of the remote user is authenticated by verification of its digital certificate. As the number of remote users to be authenticated may be large, substantial administration problems are created.

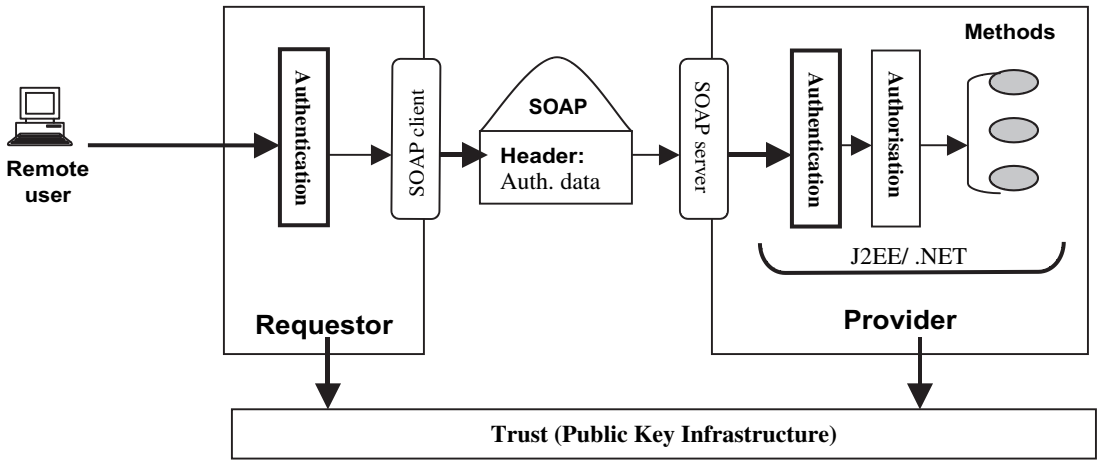

Figure 1 Current Web Service access control.

In order to alleviate cross-domain administration problems, new approaches to authentication are being developed. Requestors and Web Service providers each have their own authentication mechanisms. Costly investments are avoided by allowing each domain to do its own authentication, but to provide loose coupling between authentication systems with assertions. For instance, the requestor can send an identity assertion such as ''John was successfully authenticated with his password at 12:00 h on 4 April 2004 by www.ABC.com'' to the Web Service provider. John is granted access to methods of the Web Service provider because the requestor previously authenticated him. For the Web Service provider to accept this assertion, a relationship of trust with the requestor is required. A Web Service provider may not be interested in the real identities of the remote users, but rather in their ability or role requested on their behalf by the requestor. The requestor can make an assertion such as ''the remote user is an employee'' or ''the remote user is female'' to the Web Service provider.

Current Web Service access control integration is performed in a platform-dependent manner. A credential, presented on behalf of the user, is mapped to identities internal to the Web Service provider. Both the J2EE (Java 2 platform) and .NET (Microsoft.NET platform) platforms use role-based access control (RBAC) as a means for access control. Usernames and passwords, certificates and assertions can all be considered as sources of role-assignment. Generally, a static identity-to-role lookup is used to perform role-assignment. Assertions, which enable loose coupling between requestors and providers, are not currently supported by such assignments. Assertion-based access control integration would thus require an application-oriented approach where access control is performed by applications that are developed over platforms.

Effectively enforcing access control policies in a distributed Web Service environment is difficult with current solutions. An access control policy for each individual Web Service is statically configured with platform-dependent mechanisms. With time, the access control policy and associated configurations become unsynchronised. The dynamic context in which access requests are made is also not taken into account when access control rules are defined, and decisions taken. Access control rules of a Web Service provider ought to be more expressive in terms of the manner in which they are defined. The nature and degree of access control enforcement between requestors and Web Service providers are also more complex to define.

As the nature of a Web Service environment is highly distributed and heterogeneous, it is important to establish access control of high assurance over resources exposed by insecure SOAP connections. In order to do this, future solutions for Web Service access control should address the following requirements:

1. *Assertion-based access control*: interacting remote users and Web Service providers may know little about each other. The ability, rather than the identity, of such a user must be determined by a requestor, and passed to the Web Service provider as assertions.
2. *Mechanism-independent access control policy*: an access control policy must be defined so that it is available for real-time, dynamic decisions.
3. *Policy management*: access control functions such as decision-making and enforcement must

be clearly defined for a Web Service provider. Support must be provided for more than one language in which access control policies can be defined.
4. *Access control integration*: the authorisation decision-making process of a Web Service provider must be able to compose assertions from trusted requestors with its access control policy in a consistent manner.
5. *Standards-based implementation*: to allow any number of requestors to interoperate with a Web Service provider, a standards-based solution is required.

The next section defines a logic-based access control approach for a Web Service provider that includes the above-mentioned access control requirements in its design.

A logic-based access control approach for Web Services

Web Services may be composed to create complex services, which are supported by independent business partners. A complex service is a combination of Web Services that collectively provide a value-added service, such as integrated travel planning and insurance brokering. In complex services, a specific Web Service may take on the role of provider as well as requestor of a service. Access control for Web Services is addressed on two levels. Firstly, access control for the interaction between a requestor and independent Web Service is defined. Secondly, when complex services are processed, decisions of independent Web Services may influence the flow of control of the complex service. Informed access control decisions need to be made, based on results of previous actions. A Web Service endpoint that participates in a complex service may prefer to keep its access control policy private. As its access control policy cannot be composed with those of others, its access control decisions are conveyed to a controller. The controller orchestrates independent ''grant'' or ''deny'' decisions of participating Web Services, in order to control the flow of the complex service.

The focus of this paper is narrowed down to the first consideration. Each Web Service endpoint that is accessed by requestors must be protected and made secure by its own environment, as it would be unrealistic to expect from Web Service providers to modify their access control policies and enforcement to comply with the requirements of partners.

As an example, consider the following. Any_Company, an organisational portal, enables its customers and employees to purchase goods through the Computer_Order service. Computer_Order is the provider of a retail service that sells computer and computer-related products. The Computer_Order service exposes several methods that can be used by requestors, some of which are shown in Fig. 2. A *Register_Business* method allows all requestors to register, in order to establish an initial level of trust between requestors and the Computer_Order service. Thereafter, methods are made available to the employees and customers of Any_Company such as searching through products and special offers on to-be-released products, adding selected products to a shopping basket and placing of orders. The resulting transaction crosses the administrative domains of both Any_Company and Computer_Order that are administered independently from each other.

If John, an employee of Any_Company, makes use of such additional functionality exposed by his customised portal, he searches for and finds a product he would like to purchase. The *PlaceOrder* method of the Computer_Order service is invoked by the Any_Company requestor application. To simplify the example, the *PlaceOrder* method has only one input parameter, the stock name to be ordered, as shown in Fig. 3.

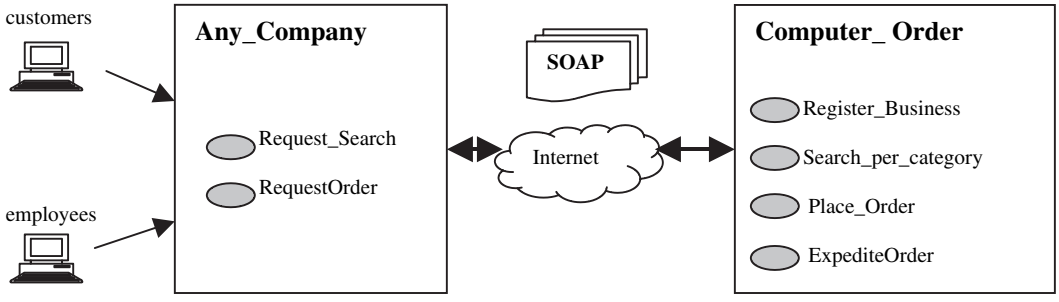

Figure 2 A requestor application invokes methods at a Web Service provider.

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

Figure 3 The SOAP request for the PlaceOrder method.

The focus of the next sections is to show how the PlaceOrder method of Computer_Order is protected so that only authorised remote users are given access to it. Firstly, 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. To enable a requestor to formulate valid access control information, its format must be understood, and made available to requestors. Finally, the design of the authorisation manager is presented, that will mediate access requests.

The structure of the SOAP request

The incoming SOAP request from Any_Company is routed to the *PlaceOrder* method at Computer_Order. The message that is conveyed to the Web Service endpoint in SOAP is shown in Fig. 3. 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.

Computer_Order's *PlaceOrder* method is exposed, and can be called by any requestor. An access control system is required to protect all methods exposed by the Computer_Order service. The access control system to be used by the Computer_Order security domain will be unique, as individual Web Service endpoints are autonomously defined and administrated.

Publish access control requirements

If an access control system is defined by Computer_Order, administrators of Any_Company need to understand its requirements, and how it is to be used. To enable this, the Computer_Order 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 (Box et al., 2003) provides a grammar for requestors and Web Service providers to communicate their requirements and capabilities in a machine-readable XML format. A policy is a collection of one or more policy assertions, and is bound to a Web Service provider through a policy attachment. 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 Any_Company portal 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 Fig. 6. Careful consideration should be given to the requirements that are exposed, as they may lead to a security domain being exploited.

The Computer_Order service may have specific access control requirements for each exposed method that must be complied with. The access control system of Computer_Order is not interested in the real identity of John, but rather in his ability, as asserted by Any_Company, who must be a trusted authority. For the *PlaceOrder* method to

be invoked successfully, it may require requestors to make assertions about remote users such as their credit card details, and employee or customer identification number. The definition of such access control requirements is shown in Fig. 4.

The Computer_Order 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 Computer_Order 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.

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.CompOrder.com/orderHeader, in order to allow requestors to formulate valid requests. Based on inspected requirements, developers of the Any_Company portal create application code that would enable John 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 John, or may be kept in, for instance, a corporate LDAP directory. The modified message, that includes the SOAP header with assertions for John, is shown in Fig. 5.

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

Mediate an access request

The access control approach relies on an independent authorisation manager that decouples access control logic from application logic. An important feature of the authorisation manager is that its location is separated from where the access control policy is enforced. Assertions sent by remote users are inspected at the Web or Application server that hosts the method that is requested. Thereafter, an access request is formulated for the authorisation manager by inspecting the SOAP request. The authorisation manager, shown in Fig. 6, is not directly exposed to users and requestors, to keep it safe from tampering. The functions of policy decision-making and policy enforcement are separated, as defined by the IETF framework for policy-based admission control (IETF Policy Framework Working Group, 2003).

When SOAP requests for methods are mediated, 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 purpose 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

```
.
<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 4 An extension of a WS-Policy requirement description in XML.

```
<soap:Envelope
     xmlns:soap="http://www.w3.org/2001/12/soap-envelope"

      <soap:Header
      <h:AssertionInfo
         xmlns:h=http://schemas.CompOrder.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.CompOrder.com/orders">
            <m:PlaceOrder>
                 <m:StockName>XE2234 Laptop</m:StockName>
           </m: PlaceOrder >
      </soap:Body>
</soap:Envelope>
```

Figure 5 A request with a header containing required assertions.

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.

Other access control approaches that have been defined for Web Services (Bhatti et al., 2003; Damiani et al., 2001) use XML-based policy specification languages, which are supported by predefined types and functions. This requires an algebraic approach that uses policy specifications in executable programs for runtime checking. Such approaches may be considered

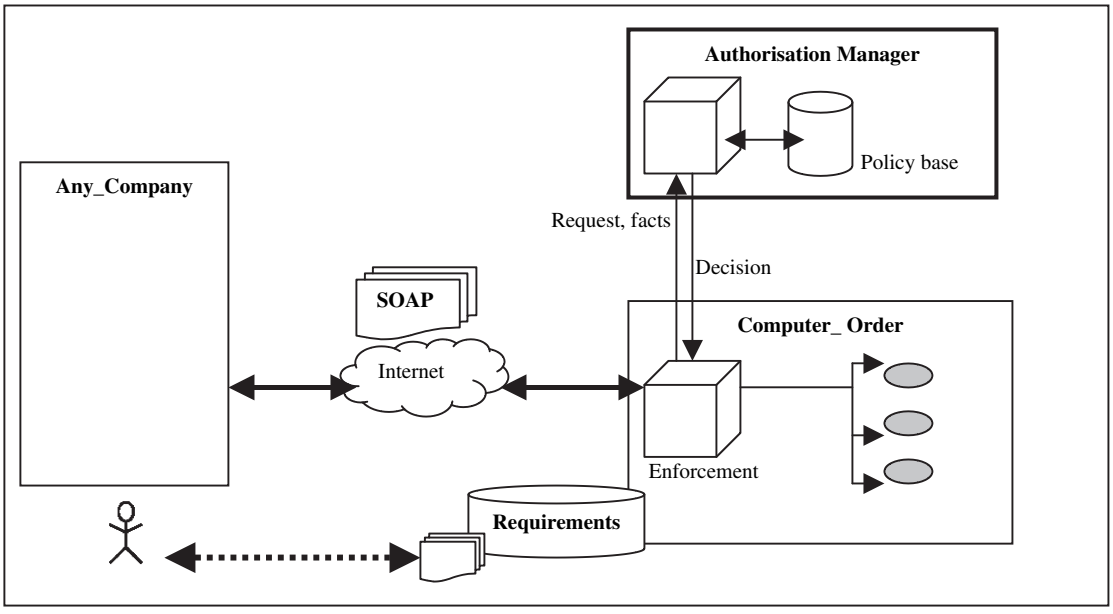

Figure 6 The authorisation manager mediates a request.

simpler to implement, as XML and its related technologies are well established, but the chances of error may increase as developers implement application code.

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: a 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.

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, 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. *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. Any_Company submits a credential in the form of an identifying digital certificate to Computer_Order. 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 Fig. 6.
   trust(Any_Company, $K_{Any\_Company}$).
   If Computer_Order 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.
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. This is done by translating 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 John, defined in Fig. 5, are imported as facts into the policy base.
   request(**requestor**(Any_Company, K_R), **assert**(**cc**(9987334566785, 0506, VISA), **id**(8894))).
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 (Kraft, 2002).
   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 (Sandu, 1996). 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, as shown in Fig. 6. 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 to access 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_R),
request(**requestor**(Requestor, K_R), **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_R),
request(**requestor**(Requestor, K_R),**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).

Fig. 7 summarises all predicates used in the definition of the policy base.

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 Any_Company and the presented assertions are evaluated. Based on these assertions, Any_Company 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 policy enforcement point, to allow the SOAP request to invoke the PlaceOrder method. Otherwise, a ''deny'' decision is returned. The policy enforcement point returns a fault to the requestor.

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.

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

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, K_R),
request(**requestor**(Requestor, K_R), **assert**(attr1, attr2, attr3)).----------4

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

Figure 7 Predicates used by the policy language.

endpoint. As interactions with services may change frequently, or may exist for very limited time periods with a limited number of transactions, they should be composed quickly, with embedded flexibility.

Five access control requirements were identified in this paper. The approach defined here addressed these requirements by standardising 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 authorisation 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 authorisation 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 authorisation manager, it is located away from policy enforcement points.

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.

Acknowledgement

The financial assistance of the Department of Labour (DoL) towards this research is hereby acknowledged. This material is based upon work supported by the National Research Foundation (NRF) in South Africa under Grant number 2054024 as well as by Telkom and IST through THRIP. Any opinion, findings and conclusions or recommendations expressed in this material are those of the authors and therefore the DoL, NRF, Telkom and IST do not accept any liability thereto.

References

Anderson A, Anderson S, Adams C, Beznosov K, Brose G, Crocker S, et al. XACML 1.0 specification, <http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml>; 2003.

Ashley P, Au R, Looi M. Cross-domain one-shot authorization using smart cards. In: Proceedings of the seventh ACM conference on computer and communications security. ACM Press; 2000. p. 220–7.

Atkinson B, Bellwood T, Cahuzac M, Clément L, Colgrave J, Corda U, et al. UDDI version 3.0.1, <http://uddi.org/pubs/uddi-v3.0.1-20031014.htm>; 14 Oct 2003.

Bacon J, Moody K. Toward open, secure, widely distributed services. Commun ACM 2002;45(6):59–64.

Beznosov K, Deng Y, Blakley B, Burt C, Barkley J. A resource access decision service for CORBA-based distributed systems. In: Proceedings of 15th IEEE annual computer security applications conference (ACSAC '99). IEEE Press; 1999. p. 310–9.

Bhatti R, Joshi JBD, Bertino E, Ghafoor A. Access control in dynamic XML-based Web-services with X-RBAC. The first international conference on Web services, Las Vegas; June 23–26, 2003.

Blaze M, Feigenbaum J, Ioannidis J, Keromytis A. The KeyNote trust management system, version 2. IETF RFC-2704, <http://www.crypto.com/papers/rfc2704.txt>; 1999.

Bonatti P, Samarati P. A unified framework for regulating access and information release on the Web. J Comput Secur 2002; 10(3):241–72.

Box D, Curbera F, Hondo M, Kale C, Langworthy D, Nadalin A, et al. Web Services Policy Framework (WS-Policy). <http://www-106.ibm.com/developerworks/library/ws-polfram/>; 28 May 2003.

Box D, Ehnebuske D, Kakivaya G, Layman A, Mendelsohn N, Nielsen HF, et al. Simple Object Access Protocol (SOAP) 1.1, <http://www.w3.org/TR/SOAP/>; May 2000.

Bray T, Paoli J, Sperberg-McQueen CM, Maler E. Extensible Markup Language (XML) 1.0. W3C recommendation. 2nd ed. 6 October 2000.

Chadwick DW, Otenko A. The PERMIS X.509 role-based privilege management infrastructure. In: Seventh ACM symposium on access control models and technologies. ACM Press; 2002. p. 135–40.

Christensen E, Curbera F, Meredith G, Weerawarana S. Web Services Description Language (WSDL) 1.1, <http://www.w3.org/TR/wsdl>.

Coyle FP. XML, Web services and the data revolution. Addison-Wesley; 2002.

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

Fielding R, Gettys J, Mogul J, Frystyk H. Hypertext transfer protocol — HTTP/1.1. Network Writing Group, Request for Comments, no. 2068; January 1997.

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>.

Gottschalk K, Graham S, Kreger H, Snell J. Introduction to Web services architecture. IBM Syst J 2002.

Hallam-Baker P, Hodges J, Maler E, McLaren C, Irving R. SAML 1.0 specification, <http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security>; 2003.

IETF Policy Framework Working Group. A framework for policy-based admission control, <http://www.ietf.org/rfc/rfc2753.txt>; 2003.

Java 2 platform, enterprise edition, <http://java.sun.com/j2ee/>.

Johnston W, Mudumbai S, 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. IEEE Press; 1998. p. 340–5.

Kraft R. A model for network services on the Web. Proceedings of the third international conference on Internet computing, IC 2002, 3:536:541, Las Vegas; June 2002.
Lam SS, Woo TYC. A framework for distributed authorization. In: Proceedings of the first ACM conference on computer and communications security. ACM Press; 1993. p. 112–8.
Microsoft.NET platform, <http://www.microsoft.com/net/>.
Sandu R. Access control: the neglected frontier. Proceedings of the first Australian conference on information security and privacy. Wollongong, Australia; June 23–26, 1996.

Ms. Coetzee is a doctoral candidate at the University of Pretoria. Her main research interests include computer security, access control composition and Web Services. She received an M.Sc. degree from the Rand Afrikaans University in 2001.

Dr. Eloff is the head of the Computer Science department at the University of Pretoria. Since 1988, he has been a full professor in the Department of Computer Science at the Rand Afrikaans University. Many acclaimed international and national conferences were organised and conducted under his leadership. He is an evaluated researcher from The National Research Foundation (NRF), South Africa, and an advisor to industry on various information security projects.