An Access Control Architecture for XML documents in workflow environments

R Botha[a] J Eloff[b]

[a] *Faculty of Computer Studies, Port Elizabeth Technikon, Port Elizabeth*
rbotha@computer.org
[b] *Department of Computer Science, Rand Afrikaans University, Johannesburg*
eloff@rkw.rau.ac.za

Abstract

The eXtensible Markup Language (XML) are being upheld as having the potential to change the way business is conducted. This will be effected by changing the way in which information is shared. However, with the sharing of information, information security becomes a concern. This paper presents an access control architecture that allows for the sharing of XML documents in workflow environments. The architecture addresses the issue of access control from two perspectives. On the one hand, issues regarding the confidentiality of information are addressed. On the other hand, the semantic integrity of information is attended to. The paper shows how the access control services, provided as part of the architecture, achieve these objectives.
Keywords: *systems architecture, information security, access control, workflow systems*
Computing Review Categories: *D2.11, D4.6, H4.1, I7.2, K6.3, K6.5*

1 Introduction

The eXtensible Markup Language (XML) has had a profound impact on the computer industry since its proposal by the World Wide Web Consortium (W3C) in February 1998 [15]. XML is often mistaken, by the uninformed, as a technology. However, XML presents a standard [4] with respect to information representation. As such, other tools and technologies make use of the XML standard for a uniform way to transfer and manipulate data.

XML, as its name suggests, is a markup language. It is often discussed in the context of content markup, content management, search engines and meta languages. XML differs from the HTML markup language in one fundamental way – it is extensible. With HTML, you are given a set of tags (markups) and you are bound by the given set as you markup a document. With XML, you "invent" the tags, so you can use XML to markup different types of documents for very specific purposes. Adjunct languages (e.g. XSL [1], XPointer [9] and XLink [10]) can be used to associate these elements to some rendering or linking semantics for their display on different media types. This allows a definite separation between the description of the structure of the document and the description of its representation on the respective media types. The structure of an XML document can be dictated by a specification in the form of a DTD [3] or an XML Schema [11]. The resultant document can be viewed and parsed as a hierarchical structure, referred to as the document tree.

Since sensitive content may be contained in an XML document, organizations that wish to utilize XML documents must consider the information security aspects associated with XML.

2 Information Security Aspects of XML Documents

Information Security is often described in terms of the five services (authentication, access control, confidentiality, integrity and non-repudiation) [13] that is employed to ensure that information maintains certain attributes, i.e. that the information stays available and confidential while maintaining a state of integrity.

For XML documents, as with any other information, these attributes of secure information need to be maintained. A distinction can be made between the protection of information while in transit and the protection of information while at the end-points, i.e. when it resides on the client or server machines. This paper is not concerned with the information security issues while the document is in transit – standard encryption and other techniques that are available within the communication technology field can be employed in this respect. This paper will only address the protection of XML documents while at the end-points.

At the end-points information security requirements can be found in questions such as: "Who may read this document?", "May Sue read this part of the document?" and "Should Tom be able to edit this

document? If so, which parts of it may he edit?". In workflow environments these questions can all be extended with "within this context", for example "Who may, within this context, read the document?".

Careful examination of questions such as the above, yields the realization that we are concerned with all three attributes of secure information, namely availability, confidentiality and integrity of the information. Briefly consider each in turn.

In so far as *availability* is concerned, it must be ensured that access is governed in such a way that a user's work is not hampered. Strategies to achieve this may include redundant storage techniques and other data backup and recovery techniques. However, it also implies that the access control service should not prohibit access from authorized users. At the same time, it is of prime importance that information is not disclosed to unauthorized users. In this respect the concern is *confidentiality* of information.

Finally, the *integrity* of information needs to be protected. Leyman and Roller [14] name three kinds of integrity concerns that exist. Firstly, physical integrity is concerned with ensuring that information is not altered during transmission and storage. This is achieved through techniques such as checksums. Secondly, operational integrity is concerned with issues such as concurrent updating. Thirdly, semantic integrity refers to the consistency of the information with business rules. For example, the business rule that states, "a person may not approve his own purchase order" requires the access control service to deny "approve" access to the initiator of the purchase order.

From the above, it is clear that the access control service is of paramount importance when considering the information security aspects during use of XML documents. Consequently, it forms the focus of the rest of this paper. The next section will discuss the properties expected of an access control service for XML documents in a workflow environment.

3 Access Control Requirements for XML Documents

XML documents have a structure as specified by an XML Schema [11] or a DTD [3]. An XML Schema provides a richer way of expressing structure than a DTD, but, in essence, both ways specify rules regarding the relation between parts of the document. In the light of the fact that an XML document can be perceived as a tree structure [19], we can see this as restrictions on the child-nodes of parent-nodes.

Consider, for example, the representation of a personnel record, depicted as an XML document and its accompanying DTD in Figure 1. The DTD show that elements consist out of other elements. For example,

```
<!ELEMENT pers_details(surname,first_name,
       other_inits?,home_address)>
```

(a) A DTD for a personnel record

```
<!ELEMENT staff_member(pers_details,slary_details,
                       old_leave_details)>
<!ATTLIST staff_member personnel_number ID #REQUIRED>
<!ELEMENT pers_details(surname,first_name,
                       other_inits?,home_address)>
<!ELEMENT home_address(street, city, zip)>
<!ELEMENT salary_details(basic_pay, bonus_pay,
                         OTE_claim*, salary_history)>
<!ELEMENT salary_history(from_date,to_date,total_pay)>
<!ELEMENT old_leavedetails(leave_period*)>
<!ELEMENT leave_period(from_date, to_date, workdays)>
<!ELEMENT surname (#PCDATA)>
<!ELEMENT first_name (#PCDATA)>
<!ELEMENT other_inits (#PCDATA)>
<!ELEMENT home_address (#PCDATA)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT zip (#PCDATA)>
<!ELEMENT basic_pay (#PCDATA)>
<!ELEMENT bonus_pay (#PCDATA)>
<!ELEMENT OTE_claim (#PCDATA)>
<!ELEMENT from_date (#PCDATA)>
<!ELEMENT to_date (#PCDATA)>
<!ELEMENT total_pay (#PCDATA)>
<!ELEMENT workdays (#PCDATA)>
```

(b) Personnel number as an XML document

```
<?xml version="1.0"?>
 <staffmember personnel_number="emp1">
 <pers_details>
   <surname>Jones</surname>
   <first_name>Ben</first_name>
   <other_inits>GF</other_inits>
   <home_address>
     <street> 123 Street </street>
     <city> Big City </city>
     <zip> 4356 </zip>
   </home_address>
 </pers_details>
 <salary_details>
    <basic_pay> 100000 </basic_pay>
    <bonus_pay> 10000 </bonus_pay>
    <OTE_aim> 200000 </OTE_aim>
    <salary_history> </salary_history>
 </salary_details>
 <old_leave_details>
    <leave_period>
       <from_date>10-May-2000</from_date>
       <to_date>12-May-2000</to_date>
       <workdays>3</workdays>
    </leave_period>
    <leave_period>
       <from_date>16-Dec-2000</from_date>
       <to_date>7-Jan-2001</to_date>
       <workdays>13</workdays>
    </leave_period>
 </old_leave_details>
 </staffmember>
```

Figure 1: A DTD and XML representation of a personnel record

shows that `pers_details` consists of a `surname`, `first_name`, `other_initials` and a `home_address`. Note that `home_address`, in turn, consist out of several elements. Elements can also have attributes, such as `staff_member` who has a `personnel_number`. The DTD furthermore controls the number of instances of an element. For example, `other_inits?` indicates that it is an optional element, after all, not everybody has second names. Similarly `leave_period*` shows that the element may occur zero, one or many times, while a `+` would indicate that the element could occur one or more times.

A document, such as the personnel record, contains parts which are more sensitive than other parts. An employee's salary can be considered more sensitive than the employee's address, which, in turn, could be more sensitive than his personnel number. Different parts of the document must thus be protected in different ways. We can therefore state that XML documents require a fine-grained access control mechanism.

The permissions associated with access control documents will be dependent on the semantics of the document (or the component of the document) to which it applies. However, permissions can be seen as abstractions of the general actions that can be observed with XML documents. All of these will be connected to the creation, inspection and modification of nodes and elements in the XML document tree. In the personnel record example, the permission "update salary" reflects the ability to replace the values in `/salary_details//@basic_pay` as well as writing historic information in the relevant part of the documents, say (among others)

`{/salary_details/salary_history//@basic_pay}`.

In a workflow environment these actions may not be applicable to the same user all of the time. The context of the task in the bigger business process will influence the access control decision [2, 17]. For example, after a leave application has been approved the dates may not be changed.

We need to consider the role that XML documents can play in a workflow environment. The next section thus describes the environment in which we operate.

4 XML documents in the workflow environment

XML documents may traverse the organization for numerous reasons. However, in order to provide a working context, this paper concentrates on XML documents that are propagated through a workflow management system.

Figure 2 graphically depicts the execution of a leave application process for employee 1. It assumes that workflow definition tools were used to define the leave application process as input to the workflow enactment service. The workflow enactment service will, on request of a user, start a new instance of a specific workflow. Thereafter the request must be approved by employee 1's Manager and the Human Resource department. In practice this would be more complex as provision must be made for negotiating the leave or rejecting the leave. The example is sufficient, however, to show the principles that will be described in this paper.

For the purposes of this paper we will furthermore assume that the workflow environment subscribes to the general idea behind role-based access control [16]. Therefore users are associated with roles, which, in turn, are associated with permissions. A role may also be related to other roles according to a partial order. Roles inherit the permissions of the roles that are inferior to them in the partial order, whereas users may also assume the roles that are inferior in the partial order to their assigned roles. The partial order is referred to as the role hierarchy.

Each of the tasks may involve a number of XML documents. The task definition will specify which role may perform the task. In environments where strict least privilege is a requirement [2, 5] this may involve the introduction of special roles in the role hierarchy. Since access to these documents is restricted at a fine granularity, permissions will be defined in terms of the constituent components of the XML document.

Figure 2 shows how the personnel record defined in Figure 1 is propogated, together with a leave form during a leave application process, in order to assist the Manager and Human Resources department with their decision. The access control requirements to the two XML documents change as their role in the workflow changes. This is considered in more detail in the following section.

5 Controlling access to XML documents

In the leave application example, two documents were encountered: a leave application and a personnel record. Consider the access control requirements of the two documents in turn.

Firstly, the personnel record. At the "Manager Approval" task in the workflow the personnel record should not be edited. Even so, not all of the information is relevant to making the decision and should, therefore, not be viewed by the manager. An example of this is the salary – the salary details should not play a role in the leave decision. Since it is sensitive information it should not be displayed at inappropriate times. A similar argument holds at the "HR Approval" task in the workflow. However, at the "HR Approval" task a portion of the personnel record that deals with leave recording might need to be updated[1].

[1] In practice, this would probably only happen once both approval tasks have been completed. Furthermore, it will most

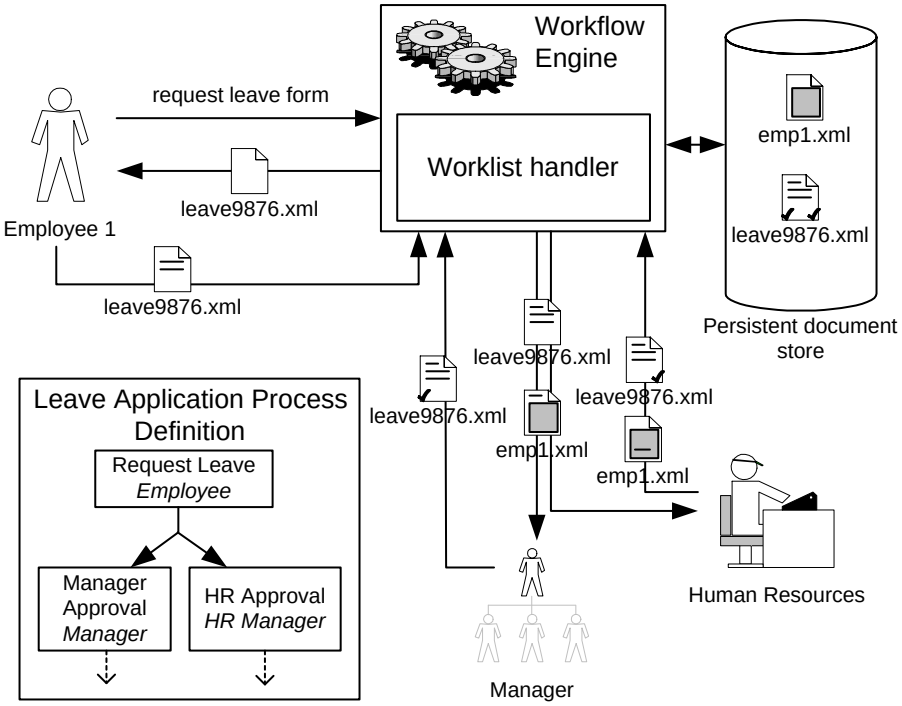

Figure 2: Involving XML documents in a typical workflow environment

At that stage, however, other details such as the salary may not be updated.

The leave form is subject to similar arguments. When the employee requests leave, the leave form can be updated by him. However, when the document arrives at the Manager or HR, they should not be able to edit the leave application details. They should only be able to complete the portions that they need to complete to grant approval. Once the leave is approved the leave details should not be edited at all.

The question of how we specify what is allowed, i.e. how we specify the permissions then comes to mind.

5.1 Specifying permissions

A permission indicates the ability to perform some action. With XML documents these abilities relate to the actions that may be performed on the nodes of the document tree. A number of observations regarding the specification of access permissions are thus in order.

Firstly note that due to the hierarchical nature of the document tree, permissions can be filtered down the document tree. For example, the fact that someone must have read permission on the `<home_address>` node of the XML document, implies that the person must have read access on the child nodes of `<home_address>`. However we must be able to specify that a person may read `<home_address>` but may not read the `<street>` node. We call this permission the "view person living area" permission. Explicit negative permissions must therefore be included.

A permission can therefore be generalized to be a set of actions that can be performed by a role. The "view person living area" permission can thus be expressed as:

```
{(/pers_details/home_address,read,+),
   (/pers_details/home_address//@street,read,-)\}
```

In general, a permission P is $P = \bigcup(node, action, sign)$ where *node* is the XPath expression [7] identifying any node in the document tree, *action* is an action that is allowable when the *sign* is positive and disallowable when *sign* is negative. The actions represented by *action* are interdependent. For example, the action to change implies the ability to read the current value. Table 1 summarizes possible actions on nodes in the document tree and their relationships. Damiana et al. [8] refined this concept to include eight authorization types which is based on whether there is inheritance and the type of inheritance that that permission have.

The permissions are specified in terms of the node of the resultant tree. However, the specification will happen once for every type of document, i.e. in XML terms, it will happen once for each DTD or Schema definition. XML schemas may allow the further refinement of these permissions by allowing the specification of "types" of nodes, e.g. an address type. Investigation as to the specification of permission based on

probably be automated and not done manually. However, in a bid to make the example more illustrative, we assume this updating is the responsibility of HR.

Action	Description	Implied
Read	The content of the node can be viewed	
Edit	Change the content of node	Read
Add	Create a node	Edit, Read
Append	Create child nodes	Read, Add(children nodes)
Delete	Delete the node	Edit, Read

Table 1: Actions on XML documents

the type of the node, rather than its position in the document tree warrants further investigation. For the purpose of this paper, however, this issue is considered future work.

5.2 Administration of Access Control

If permissions are generalized as above then the access control administration problem can easily be addressed through standard role-based access control mechanisms such as those described in [16].

This would then involve the association of permissions with roles and roles with users. Since the permissions to a document change in terms of the context of the document, the fact that a user is associated with a permission is not a sufficient condition for access control. Specific tasks will also be associated with roles [5]. A user will have to belong, therefore, to a role equal or superior to the role required for the task to be able to perform the task. The user will only receive the permissions associated with the task.

The granting of permissions will be governed by several architectural components in the workflow environment. The next section discusses the role that the various architectural components play in securing XML documents.

6 Architectural components

Figure 3 depicts the components present in the proposed architecture. In order to see what the different components' purpose is, it is best to look at the various activities that can take place. The activities falls in one of two categories: reading or changing. These two categories are discussed in paragraphs 6.2 and 6.3 respectively.

Within the architecture we can observe that XML documents can reside either on the client or on the server. The security mechanisms devised are based on the premise that we cannot trust the client [18].

6.1 Building the worklist

Within a workflow environment the worklist handler would be responsible for determining which users should do what. The workflow server will, based on the process definition and the completed tasks, determine the tasks that should occur next. These tasks will be handed off to the worklist handler who has the responsibility of determining who may perform those task. The Access Control Service must, therefore, interpret the defined Access Control Rules. These include separation of duty specifications [2] such as that "an employee may not approve his own leave".

The result of this access decision must be communicated to the users. If a "push" paradigm is followed, the worklist handler will decide who must do the task and inform the specific user through the worklist. If a "pull" paradigm is followed then the worklist handler will inform all users that may possibly do the task of the presence of the task in their worklists. As soon as one of the users commits to perform the task, the notification of the task will be removed from the other users' worklists.

The first level of access control is thus inherent to the workflow environment in that only users who should perform the task will know of the existence of the task. However, when a user performs the task, i.e. acts on the item in the worklist a further level of access control is required. The decisions that must be made can be stated as the questions "What may the user see?" and "What may the user change?". These two questions are addressed in paragraphs 6.2 and 6.3 respectively.

6.2 Restricting what the user may see

Since an XML document may contain information of various degrees of sensitivity, the complete document will seldomly be used as is. For example, in the leave application cited, the salary details may not play a role in granting/disallowing the leave.

Figure 3 shows that the Access Control Service within the worklist handler is going to be reponsible for retrieving the XML document from the XML document store. The Access Control Service will dynamically create an XSL style sheet [6] (based on the task definition) that would allow for the pruning of the document tree by a parser application. Damiani et al. [8] explain the "pruning" of an XML document tree in detail. The pruned document will only contain the information which is relevant to that step in the workflow. The document can then be transferred to the client, whereafter the client will display the document in its browser interface, possibly based on an existing style sheet. At that stage no sensitive information resides on the client as the document was pruned appropriately on the server. In this respect

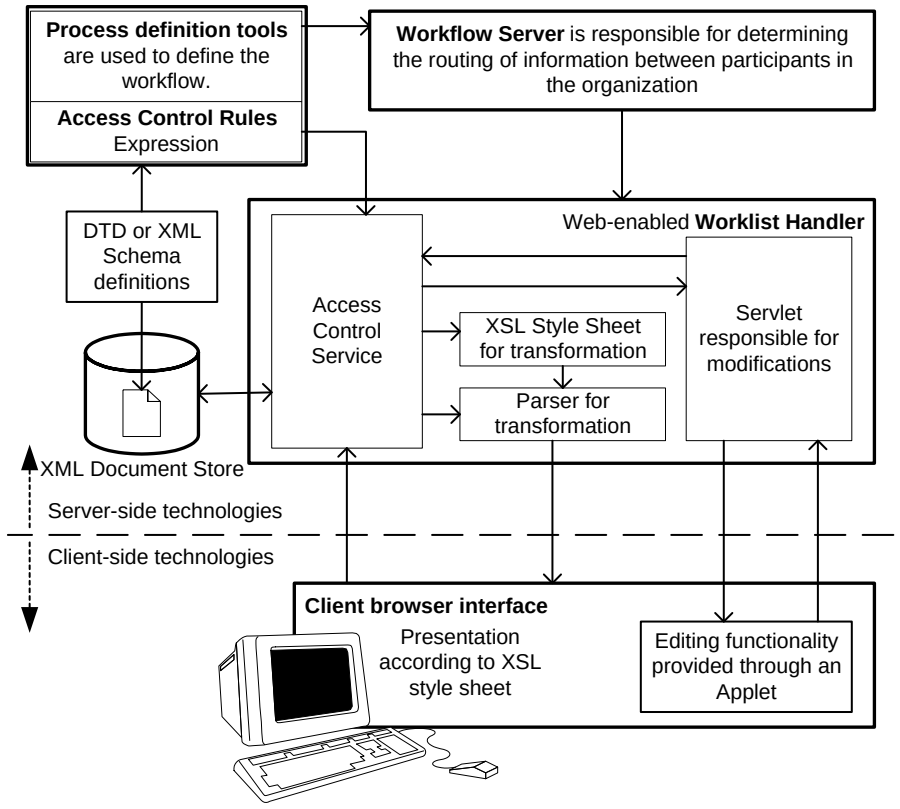

Figure 3: Architecture for controlling access to XML documents

```
<?xml version="1.0"?>
<staffmember personnel_number="emp1">
<pers_details>
  <surname>Jones</surname>
  <first_name>Ben</first_name>
  <other_inits>GF</other_inits>
</pers_details>
<old_leave_details>
   <leave_period>
      <start>10-May-2000</start>
      <end>12-May-2000</end>
      <workdays>3</workdays>
   </leave_period>
   <leave_period>
      <start>16-Dec-2000</start>
      <end>7-Jan-2001</end>
      <workdays>13</workdays>
   </leave_period>
</old_leave_details>
</staffmember>
```

Figure 4: The XML personnel record after transformation

we therefore don't have to worry about trusting the client application and execution environment for not making sensitive information available.

6.3 Updating the documents

The contents of the XML document need to change during the workflow. In the discussed example the leave form must be completed by the employee, then it must be approved (or not) by the Manager and HR. All of these activities require the documents to be updated. However, different parts of the document are being edited at different times. For example, when the document is being approved by the Manager, he should not be able to edit the leave application dates.

Updating the information on the server requires the client to send the data to the server. The client will render the HTML either as a HTML form [12] or display it through an applet. The applet option is depicted in Figure 3. The applet approach allows client side validations to occur. However, due to the untrustworthy nature of clients, these checks will have to be repeated on the server. Either way, the information will be sent to the server.

As mentioned the server will have to check that the XML document that arrives adheres to the DTD or Schema definition. Not only should it conform to the structure, but the resultant document should be compared with the original document to confirm that only nodes which should have been updated were updated, i.e. that the client application sent expected information. Experienced hackers will find it easy to alter messages on the client side [18]. Once this have been confirmed, the XML document that arrived at the server (which represents a pruned version of the original) must be merged with the original document

and the updated version stored in the persistent document store for future use. This logic is embedded in the architecture in the "Servlet responsible for modifications" component. Although the term servlet is used, it may also be achieved through other server side technologies such as CGI scripting or ASP scripting.

The updating of information requires careful consideration as to the likelihood of information being altered on the client. The server side component that is responsible for validating the updates is thus of principle importance.

7 Conclusion

The architecture that was proposed here aims to ensure the availability, confidentiality and integrity of the XML documents. In particular, it addresses these needs from an access control perspective.

Availability is addressed through the inclusion of a persistent XML document store. However, the access control service will limit access to the documents based on the task to be performed. From a practical perspective it will occasionally be difficult to predict which document will be referred to during a specific task. It is, however, reasonable to assume that those documents will only be read and not updated.

It was shown that confidentiality can be achieved through pruning the XML document tree on the server. This ensures that no unnecessary and possibly sensitive information is displayed. For documents which must be available for continual reference at unpredictable times, a default pruning process that removes all possible sensitive information may thus take place. The architecture is therefore quite capable of ensuring that information is readily available, yet in a confidential manner.

Information will retain its semantic integrity. This is partially due to the reliance on the workflow systems to make the information available for update only when it should be updated. The server side checking that updates were in line with expectations adds a further degree of confidence that semantic integrity will remain intact.

Semantic integrity can still be at risk if the data that were expected to be edited is edited inappropriately. Technology solutions to this is not possible as human error will always play a role where information is being captured. This of course allows for experienced hackers to introduce errors that appear to be human errors. This is possible since messages can still be altered by client-based methods. However, the likelihood of this is greatly reduced through applying the proposed architectural principles.

Consider, for example, the inappropriate changes of salaries. A client-side attack is only possible when the salary must actually be edited, for example, during a salary review. At all other times any message that contains an updated XML document will be validated by the server against what should have been edited and inappropriate changes will be rejected.

In the proposed architecture the worklist handler has more functionality than what would typically be the case in a workflow environment where the documents are non-XML. This is due to the fact that the worklist handler does not (and, in most cases, cannot) concern itself with the contents of the document objects. However, considering the much finer granularity at which protection occurs the extra cost incurred is not significant.

The architecture presents several opportunities for further investigation and refinement of techniques employed. The following aspects require further attention.

The work reported on in this paper only relied on DTDs to express structure. The influence of the more powerful XML Schemas on the specification of access permissions must be considered. XML Schemas allow, for example, the specification of custom types which may influence how we wish to express access permissions.

The development of tools to support this type of architecture must also receive attention. As future work a generic forms handler that subscribes to the techniques required will be developed. Furthermore, a toolset to allow for the easy specification of access permissions is required.

It is believed that this architecture provides an excellent foundation for further work. The architecture contributes to the understanding of information security and, in particular, the dynamic access control needs in an internet-based workflow environment.

References

[1] S. Adler et al. *Extensible Stylesheet Language (XSL) Version 1.0.* W3C Candidate Recommendation, http://www.w3.org/TR/2000/CR-xsl-20001121, 2000.

[2] R. A. Botha and J. H. P. Eloff. Separation of duties for access control enforcement in workflow environments. *IBM Systems Journal*, 40(3), 2001. to appear.

[3] T. Bray, J. Paoli, and C. Sperberg-McQueen, editors. *Extensible Markup Language (XML) 1.0.* W3C Recommendation, http://www.w3.org/TR/1998/REC-xml-19980210, 1998.

[4] T. Bray, J. Paoli, C. Sperberg-McQueen, and E. Maler, editors. *Extensible Markup Language (XML) 1.0 (Second Edition).* W3C Recommendation, http://www.w3.org/TR/2000/REC-xml-20001006, 2000.

[5] D. G. Cholewka, R. A. Botha, and J. H. P. Eloff. A context-sensitive access control model and prototype implementation. In S. Qing and J. H. P. Eloff, editors, *Information Security for Global Information Infrastructures: IFIP TC 11 Sixteenth Annual Working Conference on Information Security*, pages 341–350, Beijing, China, 22–24 Aug 2000. Kluwer Academic Publishers.

[6] J. Clark, editor. *XSL Transformations (XSLT) Version 1.0*. W3C Recommendation, http://www.w3.org/TR/1999/REC-xslt-19991116, 1999.

[7] J. Clark and S. DeRose, editors. *XML Path Language (XPath) Version 1.0*. W3C Recommendation, http://www.w3.org/TR/1999/REC-xpath-19991116, 1999.

[8] E. Damiani, S. de Capitani di Vimercati, S. Paraboschi, and P. Samarati. Securing XML documents. In C. Zaniolo, P. C. Lockemann, M. H. Scholl, and T. Grust, editors, *Proceedings of the 6th International Conference on Extending Database Technology: Advances in Database Technology - EDBT 2000*, volume 1777 of *Lecture Notes in Computer Science*, pages 121–135, Konstanz, Germany, 27–31 March 2000. Springer.

[9] S. DeRose, E. Maler, and R. Daniel, Jr, editors. *XML Pointer Language (XPointer) Version 1.0*. W3C Last Call Working Draft, http://www.w3.org/TR/2001/WD-xptr-20010108, 2001.

[10] S. DeRose, E. Maller, and D. Orchard, editors. *XML Linking Language (XLink) Version 1.0*. W3C Proposed Recommendation, http://www.w3.org/TR/2000/PR-xlink-20001220, 2000.

[11] D. C. Fallside, editor. *XML Schema Part 0: Primer*. W3C Candidate Recommendation, http://www.w3.org/TR/2000/CR-xmlschema-0-20001024, 2000.

[12] M. Floyd. *Building Web Sites with XML*. The Charles F. Goldfarb Series on Open Information Management. Prentice Hall PTR, Upper Saddle River, NJ, 2000.

[13] ISO 7498-2: Information Processing Systems — Open System Interconnection — Basic Reference Model – Part 2: Security Architecture, 1989.

[14] F. Leyman and D. Roller. *Production Workflow: Concepts and Techniques*. Prentice–Hall, 2000.

[15] J. Roy and A. Ramanujan. XML: Data's Universal Language. *IT Pro*, 2(3):32–36, May 2000.

[16] R. S. Sandhu, E. J. Coyne, H. L. Fenstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, Feb 1996.

[17] R. Thomas and R. Sandhu. Task-based authorization controls (TBAC): A family of models for active and enterprise-oriented authorization management. In T. Lin and S. Qian, editors, *Database Secutity, XI: Status and Prospects – Results of the IFIP WG11.3 Workshop on Database Security*, pages 166–181. Chapman and Hall, 1997.

[18] J. Viega, T. Kohno, and B. Potter. Trust (and mistrust) in secure applications. *Communications of the ACM*, 44(2):31 – 36, Feb 2001.

[19] L. Wood et al., editors. *Document Object Model (DOM) Level 1 Specification - Version 1.0*. W3C Recommendation, http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001, 1998.