Observation-based Fine Grained Access Control for XML Documents

Raju Halder and Agostino Cortesi

DAIS, Università Ca' Foscari Venezia, Italy
{halder,cortesi}@unive.it

Abstract. The eXtensible Markup Language (XML) is recognized as a simple and universal standard for storing and exchanging information on the web. The risk of unauthorized leakage of this information mandates the use of access control at various levels of granularity. In this paper, we extend to the context of XML documents the notion of Observation-based Fine Grained Access Control (OFGAC) which was originally designed for the relational databases. In this setting, data are made accessible at various levels of abstractions depending on their sensitivity level. Therefore, unauthorized users are not able to infer the exact content of an attribute or element containing partial sensitive information, while they are allowed to get a relaxed view of it, according to their access rights, represented by a specific property.

Key words: Access Control, XML Documents, Abstract Interpretation

1 Introduction

With more and more information being exchanged, distributed or published through the web, it is important to ensure that sensitive information is being accessed by the authorized web-users only. Disclosure of sensitive information to unauthorized web-users may cause a huge loss to the enterprizes or organizations. Access control policies and their enforcement [2, 7, 9] emerged as a most effective solution to ensure safety of the sensitive information in an web information system. The granularity of traditional access control mechanism for XML is coarse-grained and can be applied at file or document level only. As a result, any XML file containing data with both public and private protection requirements will have to be split into two files before applying the access control. However, the need of more flexible business requirements and security policies mandates the use of Fine Grained Access Control (FGAC) mechanisms [1, 6, 10, 12, 13] that provide safety of the information in XML documents even at lower level such as individual element or attribute level.

In traditional FGAC, the notion of sensitivity of web-information is too restrictive (either public or private) and impractical in some real systems where intensional leakage of the information to some extent is allowed with the assumption that observational power of external observers is bounded. Thus, we need to weaken or downgrading the sensitivity level of web-information, hence,

consider a weaker attacker model. The weaker attacker model characterizes the observational characteristics of attackers and can be able to observe specific properties of the private data.

To cope with this situation, in our previous work [8], we introduced an Observation-based Fine Grained Access Control (OFGAC) mechanism for Relational Database Management System (RDBMS) based on the Abstract Interpretation framework.

In this paper, we extend this approach to the context of XML documents aiming at providing accessibility of sensitive information at various levels of abstractions depending on their sensitivity level. Unauthorized users are not able to infer the exact content of an attribute or element containing partial sensitive information, while they are allowed to get a relaxed view of it represented by specific property, according to their access rights. The traditional fine grained access control can be seen as a special case of the proposed OFGAC framework.

The structure of the paper is as follows: Section 2 provides a motivating example. Section 3 recalls some basic ideas about the policy specification for XML fine grained access control system and the OFGAC framework for RDBMS. In Section 4, we extend the OFGAC framework to the context of XML documents. Finally, in Section 5, we draw our conclusions.

2 A Motivating Example

Various proposals in support of fine-grained XML access control have been introduced in the literature, including View-based [1, 5, 6], Non-deterministic Finite Automata (NFA)-based [3, 12, 13], RDBMS-based [10, 11, 14] etc.

All the proposals above are binary-based, *i.e.* an access control has only two choices: either "allow" or "forbid", resulting into two extreme views to the XML information: either "public" or "private". Sensitive information are visible to the authorized people only, whereas non-sensitive information are visible to all. However, there are many application areas where some data on the web are treated as partially sensitive and a relaxed view of those data is provided to the users at various levels of sensitivity according to their access rights.

Example 1. Consider an XML document that stores customers' information of a bank. Figure 1(a) and 1(b) represent the Document Type Definition (DTD) and its instance respectively. According to the DTD, the document consists of zero or more "customer" elements with three different child elements: "PersInfo", "AccountInfo", "CreditCardInfo" for each customer. The "CreditCardInfo" for a customer is optional, whereas each customer must have at least one bank account represented by "AccountInfo". The element "PersInfo" keeps the record of personal information for the customers.

Suppose, according to the access control policy, that employees in the bank's customer-care section are not permitted to view the exact content of IBAN and credit-card numbers of the customers, while they are allowed to view only the first two digits of IBAN numbers and the last four digits of credit card numbers,

Fig. 1: A Document Type Definition (DTD) and its instance

(a) DTD

```
<?xml version="1.0"? >
<! DOCTYPE BankCusomers [ >
<! ELEMENT BankCusomers(Customer*) >
<! ELEMENT Customer(PersInfo, AccountInfo+, CreditCardInfo?) >
<! ELEMENT PersInfo(Cid, Name, Address, PhoneNo) >
<! ELEMENT Cid (# PCDATA) >
<! ELEMENT Name (# PCDATA) >
<! ELEMENT Address (street, city, country, pin) >
<! ELEMENT street (# PCDATA) >
<! ELEMENT city (# PCDATA) >
<! ELEMENT country (# PCDATA) >
<! ELEMENT pin (# PCDATA) >
<! ELEMENT PhoneNo (# PCDATA) >
<! ELEMENT AccountInfo (IBAN, type, amount) >
<! ELEMENT IBAN (# PCDATA) >
<! ELEMENT type (# PCDATA) >
<! ELEMENT amount (# PCDATA) >
<! ELEMENT CreditCardInfo (CardNo, ExpiryDate, SecretNo) >
<! ELEMENT CardNo (# PCDATA) >
<! ELEMENT ExpiryDate (# PCDATA) >
<! ELEMENT SecretNo (# PCDATA) >
<! ATTLIST Cid IBAN CDATA #REQUIRED ]>
```

(b) XML document

```
<?xml version="1.0"? >
<BankCusomers>
<Customer>
<PersInfo>
<Cid> 140062 </Cid>
<Name> John Smith </Name>
<Address>
<street> Via Pasini 62 </street>
<city> Venezia </city>
<country> Italy </country>
<pin> 30175 </pin>
</Address>
<PhoneNo> +39 3897745774 </PhoneNo>
</PersInfo>
<AccountInfo>
<IBAN> IT10G 02006 02003 000011115996 </IBAN>
<type> Savings </type>
<amount> 50000 </amount>
</AccountInfo >
<CreditCardInfo>
<CardNo> 4023 4581 8419 7835 </CardNo>
<ExpiryDate> 12/15 </ExpiryDate>
<SecretNo> 165 </SecretNo>
</CreditCardInfo>
</Customer>
</BankCusomers>
```

keeping other sensitive digits hidden. For instance, in case of the 12 digits credit card number "4023 4581 8419 7835" and the IBAN number "IT10G 02006 02003 000011115996", a customer-care personnel is allowed to see them as "**** **** **** 7835" and "IT*** ***** ***** **********" respectively, just to facilitate the searching of credit card number and to redirect the account related issues to the corresponding country (*viz*, "IT" stands for "Italy"). In addition, suppose the policy specifies that the expiry dates and secret numbers of credit cards and the deposited amounts in the accounts are fully-sensitive and completely hidden to them.

The traditional FGAC mechanisms are unable to implement this scenario as the IBAN numbers or credit card numbers are neither private nor public as a whole. To implement traditional FGAC, the only possibility is to split the

partial sensitive element into two sub-elements: one with private privilege and other with public. For example, the credit-card numbers can be split into two sub-elements: one with first 12 digits which is made private and the other with last 4 digits which is made public. However, practically this is not feasible in all cases, as the sensitivity level and the access-privilege of the same element might be different for different users, and the integrity of data is compromised. For instance, when an integer data (say, 10) is partially viewed as an interval (say, [5, 25]), we can not split it.

The Observation-based Fine Grained Access Control (OFGAC) mechanism in [8] provides a solution of such scenario in case of RDBMS, and is based on the Abstract Interpretation framework [4]. We will extend this approach to the context of XML documents, giving rise to partial accessibility of the information on the web.

3 Observation-based Access Control Policies

In this section, we recall some basic ideas from [4, 6, 8].

Basis of Fine Grained Access Control Policy Specification for XML. Most of the existing proposals on fine grained access control for XML are based on the basic policy specification introduced by Damiani et al. [6] that specifies the access authorization by a 5-tuple of the form $\langle Subject, Object, Action, Sign, Type \rangle$. The "*Subject*" represents the identifiers or locations of the access requests to be granted or rejected. It is denoted by a 3-tuple $\langle UG, IP, SN \rangle$ where UG, IP and SN are the set of user-groups/user-identifiers, the set of completely-specified/patterns-of IP addresses and the set of completely-specified/patterns-of symbolic names respectively. For instance, ⟨ Physicians, 159.101.*.*, *.hospital.com ⟩ represents a subject belonging to the group physicians, issuing queries from the IP address matching with the pattern 159.101.*.* in the domain matching with symbolic name pattern *.hospital.com. The "*Object*" represents the Uniform Resource Identifier (URI) of the elements or attributes in the documents. The URI is specified by the conditional or unconditional path expressions. The "*Action*" is either "read" or "write" or both being authorized or forbidden. The "$Sign$" $\in \{+, -\}$ is the sign of authorization. Sign "+" indicates "allow access", whereas sign "-" indicates "forbid access". The "*Type*" of the access represents the level of access (DTD level or instance level), whether access is applicable only to the local element or applicable recursively to all sub-elements, hard or soft etc. The priority of the type of accesses from highest to lowest are: LDH (Local Hard Authorization), RDH (Recursive Hard Authorization), L (Local Authorization), R (Recursive Authorization), LD (Local Authorization specified at DTD level), RD (Recursive Authorization specified at DTD level), LS (Local Soft Authorization), RS (Recursive Soft Authorization). Since this specification provides users only two choices in accessing the information: either "allow" or "forbid", we call it *Binary-based FGAC Policy* for XML.

Galois Connection and Abstract Representation of Databases. In general, data contained in any database are *concrete* as they belong to concrete domains of integers, strings, etc, whereas *abstract* representation of these data are obtained by replacing concrete values by the elements from abstract domains representing specific properties of interests. For instance, addresses of the patients in a "Patient" database can be abstracted by the provinces they belong. Here, province is the abstract representation of all the exact locations that are covered by that province. We may distinguish partial abstract database in contrast to fully abstract one, as in the former case only a subset of the data in the database is abstracted. The values of the attribute x are abstracted by following the Galois Connection $(\wp(D_x^{con}), \alpha_x, \gamma_x, D_x^{abs})$, where $\wp(D_x^{con})$ and D_x^{abs} represent the powerset of concrete domain of x and the abstract domain of x respectively, whereas α_x and γ_x represent the corresponding abstraction and concretization functions (denoted $\alpha_x : \wp(D_x^{con}) \rightarrow D_x^{abs}$ and $\gamma_x : D_x^{abs} \rightarrow \wp(D_x^{con})$) respectively. In particular, partial abstract databases are special case of fully abstract databases where for some attributes x the abstraction and concretization functions are identity functions id, and thus, follow the Galois Connection $(\wp(D_x^{con}), id, id, \wp(D_x^{con}))$.

The Observation-based Fine Grained Access Control policy and its enforcement to RDBMS. In OFGAC framework [8], users are permitted to view the sensitive information at various levels of abstractions according to their authorization level. Highly sensitive information are forbidden completely, while partial-sensitive and non-sensitive information are disclosed in an intermediate form (represented by specific properties according to the access control policies) and in its exact form respectively.

Definition 1 (Observation-based Disclosure Policy). *Given a domain of observable properties D, and an abstraction function $\alpha_D : \wp(val) \rightarrow D$, an observation-based disclosure policy op assigned to the observer O is a tagging that assigns each value v in the database state σ a tag $\alpha_D(v) \in D$, meaning that O is allowed to access the value $\alpha_D(v)$ instead of its actual value v.*

Given an observation-based disclosure policy "*op*", the OFGAC framework for RDBMS consists of the following steps:

- Transform the concrete database into an (partial) abstract database by providing an abstract representation of the sensitive data in the database according to "*op*".
- Convert users' queries into the corresponding abstract versions and execute them on the resulting (partial) abstract database.

Observe that the abstraction of foreign/primary key are achieved by using special variable (type-2) in order to maintain integrity constraint. Also, the aggregate functions and set operations are treated differently so as to preserve the soundness.

4 OFGAC for XML

We are now in position to introduce the notion of access control policy specification for XML under OFGAC framework. Then, we apply the OFGAC approach in two directions: view-based and RDBMS-based.

Observation-based Access Control Policy Specification for XML. It is specified by a 5-tuple of the form $\langle Subject, Object, Action, Abstraction, Type\rangle$. The components "*Subject*", "*Object*", "*Action*" and "*Type*" are defined exactly in the same way as in case of FGAC policy specification. The component "*Abstraction*" is defined by the Galois Connection $(\wp(D_x^{con}), \alpha_x, \gamma_x, D_x^{abs})$, where $\wp(D_x^{con})$ and D_x^{abs} represent the powerset of concrete domain of x and the abstract domain of x respectively, and α_x and γ_x represent the corresponding abstraction and concretization functions.

Since the "*Object*" represents either XML element or attribute, the following two cases may arise when "*Abstraction*" is applied on them:

- The "*Object*" represents an intermediate element and "*Type*" is "Recursive" (denoted by "R"). In this case, the abstraction defined in the rule for an element is propagated downwards and applied to all its sub-elements and attributes recursively.
- The "*Object*" represents an attribute and "*Type*" is "Local" (denoted by "L"). In this case, only the attribute value is abstracted by following the Galois Connection specified in the rule.

Table 1: Observation-based Access Control Policy Specification for XML code

Rule	*Subject*	*Object*	*Action*	*Abstraction*	*Type*
R1	customer-care, 159.56.*.*, *.Unicredit.it	/BankCustomers/ Customer/ PersInfo	read	$(\wp(D_x^{con}), id, id, \wp(D_x^{con}))$	R
R2	customer-care, 159.56.*.*, *.Unicredit.it	/BankCustomers/ Customer/ AccountInfo/ IBAN	read	$(\wp(D_{iban}^{con}), \alpha_{iban}, \gamma_{iban}, D_{iban}^{abs})$	L
R3	customer-care, 159.56.*.*, *.Unicredit.it	/BankCustomers/ Customer/ AccountInfo/ type	read	$(\wp(D_{type}^{con}), id, id, \wp(D_{type}^{con}))$	L
R4	customer-care, 159.56.*.*, *.Unicredit.it	/BankCustomers/ Customer/ AccountInfo/ amount	read	$(\wp(D_{amount}^{con}), \alpha_\top, \gamma_\top, \{\top\})$	L
R5	customer-care, 159.56.*.*, *.Unicredit.it	/BankCustomers/ Customer/ CreditCardInfo/ CardNo	read	$(\wp(D_{CardNo}^{con}), \alpha_{CardNo}, \gamma_{CardNo}, D_{CardNo}^{abs})$	L
R6	customer-care, 159.56.*.*, *.Unicredit.it	/BankCustomers/ Customer/ CreditCardInfo/ ExpiryDate	read	$(\wp(D_{ExDate}^{con}), \alpha_\top, \gamma_\top, \{\top\})$	L
R7	customer-care, 159.56.*.*, *.Unicredit.it	/BankCustomers/ Customer/ CreditCardInfo/ SecretNo	read	$(\wp(D_{SecNo}^{con}), \alpha_\top, \gamma_\top, \{\top\})$	L

Example 2. Consider the XML code in Figure 1. The observation-based access control policy under OFGAC framework can be specified as shown in Table 1,

where the abstraction functions are defined as follows:

$$\alpha_{CardNo}(\{d_i \ : \ i \in [1 \ldots 16]\}) = **** \ **** \ **** \ d_{13}d_{14}d_{15}d_{16}$$

$$\alpha_{\top}(X) = \top$$

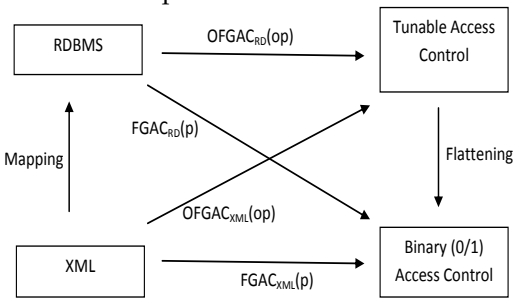
where X is a set of concrete values and $\top$ is the top most element of the corresponding abstract lattice. The functions α_{iban}, γ_{iban}, γ_{CardNo}, $\gamma_{\top}$ are also defined in this way depending on the corresponding domains. Observe that the identity function *id* is used to provide the public accessibility of non-sensitive information, whereas the functions $\alpha_{\top}$ and $\gamma_{\top}$ are used to provide private accessibility of highly sensitive information by abstracting them with top most element $\top$ of the corresponding abstract lattice.

Given a binary-based access control policy p and an observation-based access control policy op in XML format, the FGAC and OFGAC can be implemented in two ways:

- By applying p or op directly to the XML documents (view-based) or by rewriting users' XML queries by pruning the unauthorized part in it (NFA-based).
- By taking the support of RDBMS, where the XML documents and the XML policies (p or op) are first mapped into the underlying relational databases and the policy SQL respectively, and then the users' XML queries are mapped into equivalent SQL queries and evaluated on those relational databases by satisfying the policy SQL.

Figure 2 depicts a pictorial representation of these approaches. Observe that the application of FGAC *w.r.t.* p results into a binary-based access control system that yields two extreme views to the information: either "allow" or "forbid", whereas the application of OFGAC *w.r.t.* op, on the other hand, results into a tunable access control system where partial view of the information at various levels of abstractions is provided.

Fig. 2: Pictorial Representation of FGAC Vs. OFGAC



View-based OFGAC for XML. Consider the XML code in Figure 1 and the associated observation-based access control policy specification depicted in Table 1. We know that in view-based approaches for each subject interacting with the system, separate views are generated with respect to the access rules associated with the subject [6]. Therefore, in our example, the XML view corresponding to the users belonging to "customer-care" section of the bank is depicted in Figure 3.

Fig. 3: View generated for the employees in bank's customer-care section

```
<?xml version="1.0"? >
<BankCusomers>
<Customer>
<PersInfo>
<Cid> 140062 </Cid>
<Name> John Smith </Name>
<Address>
<street> Via Pasini 62 </street>
<city> Venezia </city>
<country> Italy </country>
<pin> 30175 </pin>
</Address>
<PhoneNo> +39 3897745774 </PhoneNo>
</PersInfo>
<AccountInfo>
<IBAN> IT*** ***** ***** ********** </IBAN>
<type> Savings </type>
<amount> ⊤ </amount>
</AccountInfo >
<CreditCardInfo>
<CardNo> **** **** **** 7835 </CardNo>
<ExpiryDate> ⊤ </ExpiryDate>
<SecretNo> ⊤ </SecretNo>
</CreditCardInfo>
</Customer>
</BankCusomers>
```

Consider now the following XML query Q_{xml} issued by a personnel in the customer-care section:

$$Q_{xml} = /BankCusomers/Customer/AccountInfo[@type = \text{"}Savings\text{"}]$$

The execution of Q_{xml} on the view of Figure 3 returns the following results:

```
<AccountInfo>
<IBAN> IT*** ***** ***** ********** </IBAN>
<type> Savings </type>
<amount> ⊤ </amount>
</AccountInfo>
```

RDBMS-based OFGAC for XML. Consider the XML document in Figure 1 and the observation-based policy specification in Table 1. By following [10], we first map the XML document into relational database representation, partially shown in Table 2. Observe that we do not translate the XML policies into the equivalent SQL statements, rather we put the rules into the relational database itself by associating them with the corresponding elements or attributes. The empty rule in a row specifies that the corresponding element (and its sub-elements and child-attributes) or attribute has public authorization. If any access-conflict occurs for any sub-element, it is resolved simply by adopting *abstraction-take-precedence* policy according to which authorization corresponding to more abstract view overrides the authorization corresponding to less abstract view. The users' XML

queries are then mapped into SQL representation and are evaluated on this relational database under OFGAC framework as reported in [8].

Table 2: The equivalent relational database representation of the XML code

(a) "*BankCustomers*"

id	*pid*	*rule*
BC1	null	-

(b) "*Customer*"

id	*pid*	*rule*
C1	BC1	-

(c) "*PersInfo*"

id	*pid*	*rule*
PI1	C1	R1

(d) "*AccountInfo*"

id	*pid*	*rule*
AI1	C1	-

(e) "*CreditCardInfo*"

id	*pid*	*rule*
CI1	C1	-

(f) "*IBAN*"

id	*pid*	*val*	*rule*
IB1	AI1	IT10G 02006 02003 000011115996	R2

(g) "*type*"

id	*pid*	*val*	*rule*
TP1	AI1	Savings	R3

(h) "*amount*"

id	*pid*	*val*	*rule*
AM1	AI1	5000	R4

(i) "*CardNo*"

id	*pid*	*val*	*rule*
CN1	CI1	4023 4581 8419 7835	R5

(j) "*ExpiryDate*"

id	*pid*	*val*	*rule*
EX1	CI1	12/15	R6

Suppose the following XML query Q_{xml} is issued by an employee from customer-care section of the bank:

$$Q_{xml} = /BankCusomers/Customer/AccountInfo[@type = \text{“}Savings\text{”}]/IBAN$$

Since the OFGAC Policies and XML documents are now in the form of relational database, the system translates Q_{xml} into an equivalent SQL query Q_{rdb} as follows:

Q_{rdb} =`SELECT` *Ch_No.val* `FROM` *IBAN Ch_No, type Ch_Tp, AccountInfo P_AccInfo, Customer P_Cust, BankCustomers P_BCust* `WHERE` (*Ch_No.pid = P_AccInfo.id* `AND` *Ch_Tp.pid = P_AccInfo.id* `AND` *Ch_Tp.val* = “*Savings*”) `AND` *P_AccInfo.pid = P_Cust.id* `AND` *P_Cust.pid = P_BCust.id*

The execution of Q_{rdb} on the database of Table 2, by following the OFGAC technique in [8], yields the following result:

val
IT*** ***** ***** **********

Observe that RDBMS-based approaches suffer from time-inefficiency, whereas view-based approaches, on the other hand, suffer from space-inefficiency. The robustness of the proposed OFGAC system depends on the ability of the external observers to extract sensitive information based on the observable properties of the query results. The possibility of collusion attacks for XML documents under OFGAC framework is same as that of relational databases as described in [8].

5 Conclusions

In this paper, we discussed the extension of the notion of observation-based fine grained access control to the case of XML documents. The traditional FGAC can

be seen as a special case of the proposed OFGAC framework, where the sensitive information are abstracted by the top element $\top$ of their corresponding abstract lattices.

Acknowledgement

Work partially supported by RAS L.R. 7/2007 Project TESLA.

References

1. Bertino, E., Ferrari, E.: Secure and selective dissemination of xml documents. ACM Trans. on Information and System Security 5(3), 290–331 (2002)
2. Bertino, E., Jajodia, S., Samarati, P.: A flexible authorization mechanism for relational data management systems. ACM Trans. on Information Systems 17(2), 101–140 (1999)
3. Bouganim, L., Ngoc, F.D., Pucheral, P.: Client-based access control management for xml documents. In: Proc. of the 13th Int. Conf. on Very Large Data Bases (VLDB '04). pp. 84–95. VLDB Endowment, Toronto, Canada (2004)
4. Cousot, P., Cousot, R.: Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Conf. Record of the 6th Annual ACM POPL. pp. 238–252. ACM Press, Los Angeles, CA, USA (1977)
5. Damiani, E., de Capitani di Vimercati, S., Paraboschi, S., Samarati, P.: Design and implementation of an access control processor for xml documents. Journal of computer and telecommunications netowrking 33(1–6), 59–75 (2000)
6. Damiani, E., De Capitani di Vimercati, S., Paraboschi, S., Samarati, P.: A fine-grained access control system for xml documents. ACM Trans. on Information and System Security 5(2), 169–202 (2002)
7. Griffiths, P.P., Wade, B.W.: An authorization mechanism for a relational database system. ACM Trans. on Database Systems 1(3), 242–255 (1976)
8. Halder, R., Cortesi, A.: Observation-based fine grained access control for relational databases. In: Proc. of the 5th Int. Conf. on Software and Data Technologies (ICSOFT '10). pp. 254–265. INSTICC Press, Athens, Greece (2010)
9. Jajodia, S., Samarati, P., Subrahmanian, V.S., Bertino, E.: A unified framework for enforcing multiple access control policies. SIGMOD Record 26(2), 474–485 (1997)
10. Koromilas, L., Chinis, G., Fundulaki, I., Ioannidis, S.: Controlling access to xml documents over xml native and relational databases. In: Proc. of the 6th VLDB Workshop on Secure Data Management (SDM '09). pp. 122–141. Springer LNCS, Volume 5776, Lyon, France (2009)
11. Lee, D., Lee, W.C., Liu, P.: Supporting xml security models using relational databases: A vision. In: Proc. of the 1st Int. XML Database Symposium (Xsym '03). pp. 267–281. Springer LNCS, Volume 2824, Berlin, Germany (2003)
12. Luo, B., Lee, D., Lee, W.C., Liu, P.: Qfilter: fine-grained run-time xml access control via nfa-based query rewriting. In: Proc. of the 13th ACM Int. Conf. on Information and knowledge management (CIKM '04). pp. 543–552. ACM Press, Washington D.C., USA (2004)
13. Murata, M., Tozawa, A., Kudo, M., Hada, S.: Xml access control using static analysis. ACM Trans. on Information and System Security 9(3), 292–324 (2006)
14. Tan, K.L., Lee, M.L., Wang, Y.: Access control of xml documents in relational database systems. In: Proc. of the Int. Conf. on Internet Computing (IC '01). pp. 185–191. CSREA Press, Las Vegas, Nevada, USA (2001)