

Penn State College of Information Sciences and Technology IST

Pragmatic XML Access Control using Off-the-shelf RDBMS

Bo Luo, Dongwon Lee, and Peng Liu
The Pennsylvania State University

ESORICS'07
Dresden, Germany

1

Penn State College of Information Sciences and Technology IST

Where is your XML data?

Large volume of XML data vs. mature RDBMS

2

Penn State College of Information Sciences and Technology IST

Where is your XML data?

- XDB: Native XML Database Systems
 - Galax, Timber, etc
- XRDB: RDBMS-supported XML DB
 - XML-Relational conversion algorithm
 - Underlying RDBMS
 - Taking advantage of the maturity of RDBMS.
- "Native" XML support in commercial RDBMS
 - XML data → CLOB, XML Table, Object Relational etc.
 - RDBMS with "built-in" XRDB

• **We will focus on XRDB**

3

Penn State College of Information Sciences and Technology IST

XRDB Architecture

Query processing in XRDB:

$$A_X = C^{-1}(A_R) = C^{-1}(C_Q(Q_X) < C_D(D_X) >)$$

4

Penn State College of Information Sciences and Technology IST

Access control in XRDB: naïve approach

Step 1. conversion of access control rules

$$Rule R_X = \{subject, object, action, sign\}$$

$$Rule R_R = \{subject, CD(RX.object), action, sign\}$$

Step 2:

ACR_X – the set of XML access control rules
 ACR_R – the set of relational access control rules

5

Penn State College of Information Sciences and Technology IST

Surprising Finding

- The naïve approach is **NOT** secure!!
 - Problem 1: Confidentiality leaking
 - Problem 2: Denial of service effects – over filtering
- The goal of this research is two-fold:
 - Why is the naïve approach not secure?
 - How to fix the two problems?

6

PENNSYLVANIA STATE UNIVERSITY College of Information Sciences and Technology IST

XML-2-Relational Conversion

- XML-2-Relational conversion algorithm $C()$:
 - $C_Q()$, $C_D()$, and $C^{-1}()$.
 - Properties
 - Lossless node conversion
 - Lossless node set decomposition
 - Exclusive conversion
 - Correct query processing
- Existing approaches
 - Schema-based: XML schema \leftrightarrow relational schema
 - Shared-inlining
 - Schema-oblivious: XML data \leftrightarrow relational data
 - XRel, Edge

09/24/2007 Pragmatic XML Access Control using Off-the-shelf RDBMS 7

PENNSYLVANIA STATE UNIVERSITY College of Information Sciences and Technology IST

XML-2-Relational Conversion

- Example of Schema-oblivious X2R conversion
 - XRel [Yoshikawa et al, ACM TOIT 2001]
 - Each XPath is converted to a record in PATH table
 - Each node is converted to a record in ELEMENT table

DOCID	ELEMENTID	PATHID	ST	ED	
0	252	164	33996	36229	#<people>
0	293	165	35592	35826	#<person>
0	294	166	35600	35625	#<name>
0	299	165	35832	36217	#<person>
0	303	188	35989	36032	#<creditcard>

09/24/2007 Pragmatic XML Access Control using Off-the-shelf RDBMS 8

PENNSYLVANIA STATE UNIVERSITY College of Information Sciences and Technology IST

XML-2-Relational Conversion

- Example of Schema-based X2R conversion
 - Shared-inlining [Shanmugasundaram et al, VLDB 1999]
 - XML schema is converted into relational schema

Person	Person_name	Person_...	Person_Creditcard
Rens Rifaut	xxxxxxxx	8814 4441 4702 6117	
Rinli Trebbe	xxxxxxxx	4464 2718 4111 2553	
Sohell Cosar	xxxxxxxx	8116 1498 1997 5316	

09/24/2007 Pragmatic XML Access Control using Off-the-shelf RDBMS 9

PENNSYLVANIA STATE UNIVERSITY College of Information Sciences and Technology IST

Access Control?

- Fine-grained Access Control is desired.
- It is not supported/researched in existing XRDB or commercial RDBMS with XML DB.
- The research questions:
 - Why is the naïve approach not secure?
 - How to fix the two problems?

09/24/2007 Pragmatic XML Access Control using Off-the-shelf RDBMS

PENNSYLVANIA STATE UNIVERSITY College of Information Sciences and Technology IST

XML Access Control Model

- Role based access control
- 4-tuple access control rules
 - {subject, object, action, sign}

Note: Every data object is a tree

09/24/2007 Pragmatic XML Access Control using Off-the-shelf RDBMS 11

PENNSYLVANIA STATE UNIVERSITY College of Information Sciences and Technology IST

Deep Set Operators

- Regular set operators: only consider operands' context nodes
- Deep set operators: consider context node and descendants

09/24/2007 Pragmatic XML Access Control using Off-the-shelf RDBMS 12

Deep Set Operators

09/24/2007 Pragmatic XML Access Control using Off-the-shelf RDBMS 13

Our Definition of "Security" in Native XDB

- *Safe answer* of query Q includes all the XML nodes n such that:
 - (1) $n \in \langle Q \rangle$
 - (2) the access to n is granted by positive rules
 - (3) the access to n is not denied by negative rules.

Given XML document D :

09/24/2007 Pragmatic XML Access Control using Off-the-shelf RDBMS 14

Secure XDB

- Safe answer denoted by deep set operators

$$SA_X = \langle Q \rangle \cap_X \left[\left((R_{X1}^+ \cup_X \dots \cup_X R_{Xn}^+) -_X \left((R_{X1}^- \cup_X \dots \cup_X R_{Xm}^-) \right) \right) \right]$$

09/24/2007 Pragmatic XML Access Control using Off-the-shelf RDBMS 15

The Basic Question

$$SA_X = \langle Q_X \rangle \cap_X \left[\left((R_{X1}^+ \cup_X \dots \cup_X R_{Xn}^+) -_X \left((R_{X1}^- \cup_X \dots \cup_X R_{Xm}^-) \right) \right) \right]$$

09/24/2007 Pragmatic XML Access Control using Off-the-shelf RDBMS 16

Problem Statement

- The basic finding: in many cases,

$$SA_X \neq FA_X$$

In the following, we will first explain WHY; then we will propose a fix

09/24/2007 Pragmatic XML Access Control using Off-the-shelf RDBMS 17

Naive Enforcement of R_R

- We convert an XML document using XRel [Yoshikawa et al, ACM TOIT 2001]
- We have the following two rules:
 - (user, /site/people/person, read, +)
 - (user, /site/people/person/creditcard, -)

DOCID	ELEMENTID	PATHID	ST	ED	
0	252	164	33996	36229	#-<people>
0	293	165	35592	35826	#-<person>
0	299	165	35832	36217	#-<person>
0	303	188	35989	36032	#-<creditcard>

09/24/2007 Pragmatic XML Access Control using Off-the-shelf RDBMS 18

Naive Enforcement of R_R - Problem 1

DOCID	ELEMENTID	PATHID	ST	ED	
0	252	164	33996	36229	#<people>
0	293	165	35832	35826	#<person>
0	299	165	35832	36217	#<person>
0	303	188	35989	36032	#<creditcard>

XPath Query: `//people/person`

```

SELECT e0.DOCID, e0.PATHID, e0.ELEMENTID, e0.ST, e0.ED
FROM document d, element e0, pth p0
WHERE p0.pathexp LIKE '#//people#person'
AND e0.pathid = p0.pathid AND d.docid = e0.docid
    
```

Result:

DOCID	ELEMENTID	PATHID	ST	ED	
0	293	165	35832	35826	#<person>
0	299	165	35832	36217	#<person>

- Confidentiality Leak!
- Should use **DEEP EXCEPT** to remove descendants
- Only **EXCEPT** is implemented

09/24/2007 Pragmatic XML Access Control using Off-the-shelf RDBMS 19

Naive Enforcement of R_R - Problem 2

DOCID	ELEMENTID	PATHID	ST	ED	
0	252	164	33996	36229	#<people>
0	293	165	35832	35826	#<person>
0	299	165	35832	36217	#<person>
0	303	188	35989	36032	#<creditcard>

XPath Query: `//people`

```

SELECT e0.DOCID, e0.PATHID, e0.ELEMENTID, e0.ST, e0.ED
FROM document d, element e0, pth p0
WHERE p0.pathexp LIKE '#//people'
AND e0.pathid = p0.pathid AND d.docid = e0.docid
    
```

Result: NULL

- Over filtering: two person nodes are anticipated.
- Should use **DEEP INTERSECT** to obtain descendant nodes
- Only **INTERSECT** is implemented

09/24/2007 Pragmatic XML Access Control using Off-the-shelf RDBMS 20

How to Fix the Two Problems?

- Naive enforcement of R_R generates incorrect answer.
- How can we fix it?

09/24/2007 Pragmatic XML Access Control using Off-the-shelf RDBMS 21

Object and Operation Equivalency

- Object equivalency**
When both $R = C(X)$ and $X = C^{-1}(R)$ hold for XML node set X and relation R , we consider both X and R equivalent w.r.t. C/C^{-1} , and denote as $X \equiv R$.
- Operation equivalency**
Suppose $X_1 \equiv R_1$ and $X_2 \equiv R_2$ w.r.t. C/C^{-1} . Then, an XML operation OP_X is equivalent to a relational operation OP_R (denoted as $OP_X \equiv OP_R$) w.r.t. C and C^{-1} if:
For all possible pairs of X and R
$$C(X_1, OP_X X_2) = C(X_1) OP_R C(X_2) = R_1 OP_R R_2$$

09/24/2007 Pragmatic XML Access Control using Off-the-shelf RDBMS 22

Secure XRDB: The Idea

Lemma 1: In $XRDB(C)$, if we can find relational operators, \cup_R^D , \cap_R^D and $-_R^D$ which are **equivalent** to XML deep set operators, \cup_X^D , \cap_X^D and $-_X^D$ w.r.t. the X2R conversion algorithm C , we are able to enforce XML access control in $XRDB(C)$ such that $C^{-1}(C_Q(Q_X) < C_D(D_X)) = SA_X$ always holds.

09/24/2007 Pragmatic XML Access Control using Off-the-shelf RDBMS 23

Equivalent Conversion of Deep Set Operators

- Deep Union and Deep Intersect:**
 - At present, all X2R conversion algorithms (we are aware of) are able to support deep union and deep intersect.

Sufficient Conditions:

Lemma 2. To implement deep-union and deep-intersect operators in $XRDB(C)$, the X2R conversion algorithm C should: (1) fulfill the soundness requirement; and (2) for given node n and node set P , it should be able to check the containment condition: $C(n) \in C(P//*)$, e.g., it should recognize if $C(n)$ is a descendant of any node $C(p)$;

09/24/2007 Pragmatic XML Access Control using Off-the-shelf RDBMS 24

Equivalent Conversion of Deep Union

- Example – deep union
 - Rule: {user, /site/people/person, read, +}
 - Query: //people
 - Safe answer: //people deep-union /site/people/person
 - Now implemented as:

```

SELECT docID, pathID, elementID, st, ed FROM element
WHERE (elementID IN
  ( SELECT e.elementID FROM document d, element e, path p0
    WHERE p0.pathExp LIKE '%people%' AND e0.pathID = p0.pathID AND d.docID = e0.docID )
AND elementID IN
  (SELECT e0.elementID FROM document d, element e0, path p0
   WHERE p0.pathExp LIKE '%people/person%' AND e0.pathID = p0.pathID AND d.docID = e0.docID ))
AND NOT EXISTS (
  (SELECT e0.elementID FROM document d, element e0, path p0
   WHERE p0.pathExp LIKE '%people%' AND e0.pathID = p0.pathID AND d.docID = e0.docID)
OR NOT EXISTS (
  (SELECT e0.elementID FROM document d, element e0, path p0
   WHERE p0.pathExp LIKE '%people/person%' AND e0.pathID = p0.pathID AND d.docID = e0.docID))

```

25

09/24/2007 Pragmatic XML Access Control using Off-the-shelf RDBMS

Equivalent Conversion of Deep Except

- Deep except operation is for negative rules.
 - Negative rules are used to "revoke" access rights.
 - Node elimination and descendant elimination rules

node elimination (NE) negative rule

descendant elimination (DE) negative rule

26

09/24/2007 Pragmatic XML Access Control using Off-the-shelf RDBMS

Equivalent Conversion of Deep Except (2)

positive rule

A negative rule in ACR restricts user from access a set of nodes $\{r_1^-, \dots, r_n^-\}$. If none of the nodes is a descendant of the context node of a positive rule, then it is called a **node elimination (NE)** negative rule. Else, if one of the nodes is a descendant of the context node of a positive rule, it is called a **descendant elimination (DE)** negative rule.

27

09/24/2007 Pragmatic XML Access Control using Off-the-shelf RDBMS

Equivalent Conversion of Deep Except (3)

- Node elimination (NE) negative rules do not generate new tree
 - Any X-2-R algorithm that supports deep union and deep intersect is able to support NE negative rules.
- Descendant elimination (DE) negative rules generate new trees
 - Some X-2-R algorithms **cannot support** DE negative rules

Lemma 3. When deep-except operator takes node specified by descendant elimination negative rules as the second operand, it is implemented through deepRemove() operation. To implement deep-except operator that supports descendant elimination negative rules in XRDB(C), the X2R conversion algorithm X should: (1) fully satisfy Lemma 2; and (2) for any node n_1 and its descendant n_2 , $C(n_2)$ should be part of $C(n_1)$; and in the reverse conversion of $n_1 = C^{-1}(C(n_1))$, node n_2 in the subtree is entirely converted from $C(n_2)$.

28

09/24/2007 Pragmatic XML Access Control using Off-the-shelf RDBMS

Equivalent Conversion of Deep Except (4)

- Example: Shared-inlining

Person	Person_name	Person_...	Person_Creditcard
Rens Rifaat	xxxxxxxxxx	8814 4441 4702 6117	
Rinli Trebbe	xxxxxxxxxx	4464 2718 4111 2553	
Soheil Cosar	xxxxxxxxxx	8116 1498 1997 5316	

– //person deep-except //creditcard

– Output (remove creditcard column)

Person	Person_name	Person_...	Person_Creditcard
Rens Rifaat	xxxxxxxxxx	8814 4441 4702 6117	
Rinli Trebbe	xxxxxxxxxx	4464 2718 4111 2553	
Soheil Cosar	xxxxxxxxxx	8116 1498 1997 5316	

– Could be easily implemented in RDBMS

29

09/24/2007 Pragmatic XML Access Control using Off-the-shelf RDBMS

Equivalent Conversion of Deep Except (5)

- Example: XRel

DOCID	ELEMENTID	PATHID	ST	ED	
0	252	164	33906	36229	#<people>
0	293	165	35002	35826	#<person>
0	299	165	35832	36217	#<person>
0	303	188	35989	36032	#<creditcard>

//person deep-except //creditcard

Desired output (new person nodes)

DOCID	ELEMENTID	PATHID	ST	ED	
0	x	165	35832	35989	
0	x	165	36033	36217	

Not feasible in relational algebra!

30

09/24/2007 Pragmatic XML Access Control using Off-the-shelf RDBMS

PENNSYLVANIA STATE UNIVERSITY
College of Information Sciences and Technology

IST

Implementation

- Way 1: view-based implementation
 - Done in our experiments
- Way 2: pre-preprocessing
 - Relational query rewriting, through say Oracle VPD
 - Done in our experiments
- Way 3: post-preprocessing
 - Not done
- Preliminary conclusion: **insignificant** degradation

31

09/24/2007 Pragmatic XML Access Control using Off-the-shelf RDBMS

PENNSYLVANIA STATE UNIVERSITY
College of Information Sciences and Technology

IST

Questions?

Thank you!

32

09/24/2007 Pragmatic XML Access Control using Off-the-shelf RDBMS

PENNSYLVANIA STATE UNIVERSITY
College of Information Sciences and Technology

IST

Background: XML-2-Relational Conversion

- An X-2-R conversion algorithm is **Lossless**:
 - Lossless node conversion: $C_D^{-1}(C_D(x)) = x$
 - Lossless node set decomposition:
 $C_D^{-1}(C_D(\{x_1, \dots, x_n\})) = C_D^{-1}(\{C_D(x_1), \dots, C_D(x_n)\}) = \{C_D^{-1}(C_D(x_1)), \dots, C_D^{-1}(C_D(x_n))\}$
 - Exclusive conversion:
 $C_D(x_i) = C_D(x_j)$ only when $x_i = x_j$; and $C_D^{-1}(r_i) = C_D^{-1}(r_j)$ only when $r_i = r_j$;
- An X-2-R conversion algorithm is **Correct**:
 - Correct query processing:
 $Q \langle D_X \rangle = C^{-1}(Q_R \langle D_R \rangle) = C^{-1}(C_D(Q_X) \langle C_D(D_X) \rangle)$
- An X2R conversion algorithm A is **sound** iff it is lossless and correct.

33

09/24/2007 Pragmatic XML Access Control using Off-the-shelf RDBMS