Pragmatic XML Access Control using Off-the-shelf RDBMS

Bo Luo, Dongwon Lee, and Peng Liu
The Pennsylvania State University
ESORICS’07
Dresden, Germany

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

Access control in XRDB: naïve approach
Step 1. conversion of access control rules
  Rule RX = \{subject, object, action, sign\}
  Rule R_R = \{subject, CD(RX.object), action, sign\}

Step 2: Query processing in XRDB:
\[ A_X = C^{-1}(A_R) = C^{-1}(C_D(Q_X) < C_D(D_X) >) \]

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?
XML-2-Relational Conversion

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

```
[Diagram showing XML and relational schema conversion]
```

- **Example of Schema-based X2R conversion**
  - Shared-inlining
  - XML schema <-> relational schema

```
[Diagram showing XML and relational schema conversion]
```

- **Properties**
  - Lossless node conversion
  - Lossless node set decomposition
  - Exclusive conversion
  - Correct query processing

- **Existing approaches**
  - Schema-based: XML schema <-> relational schema
    - Shared-inlining
  - Schema-oblivious: XML data <-> relational data
    - XRel, Edge

```
[Diagram showing XML and relational schema conversion]
```

- **Access Control?**
  - Fine-grained Access Control is desired.
  - It is not supported/researched in existing XRDB or commercial RDBMS with XML DB.

```
[Diagram showing access control model]
```

- **The research questions:**
  - Why is the naïve approach not secure?
  - How to fix the two problems?

```
[Diagram showing access control model]
```

- **XML Access Control Model**
  - Role based access control
  - 4-tuple access control rules
    - (subject, object, action, sign)

```
[Diagram showing access control model]
```

- **Deep Set Operators**
  - Regular set operators: only consider operands’ context nodes
  - Deep set operators: consider context node and descendants

```
[Diagram showing deep set operators]
```
11/6/2007

Deep Set Operators

- INTERSECT
- DEEP-INTERSECT
- EXCEPT
- DEEP-EXCEPT

Our Definition of “Security” in Native XDB

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

Given XML document \( D \),

- \( \text{Safe Answer} \)
- \( \text{ACR}^+(D) \)
- \( \text{ACR}^-(D) \)

Secure XDB

- Safe answer denoted by deep set operators

\[
\text{SA}_Q = \langle Q \rangle \cap X \left( \{R_{X1} \cup \ldots \cup R_{Xn} + \} - X \{R_{X1} - \ldots - R_{Xm} - \} \right)
\]

Problem Statement

- The basic finding: in many cases,

\[
\text{SA}_Q \neq \text{FA}_Q
\]

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

Naive Enforcement of \( R_G \)

- We convert an XML document using XRel [Yoshikawa et al, ACM TOIT 2001]

- We have the following two rules:

\[
\{\text{user, /site/people/person, read, +}\}
\]
\[
\{\text{user, /site/people/person/creditcard, -}\}
\]
enforce XML access control in XRDB(C) such that

Lemma 1: In XRDB(C), if we can find relational operators, \( \hat{O}_P \), \( \hat{\cap}_R \), and \( \hat{\cup}_R \) which are equivalent to XML deep set operators, \( \hat{\cap} \), \( \cap \), and \( \cup \) w.r.t. the X2R conversion algorithm C, we are able to enforce XML access control in XRDB(C) such that \( C^{-1}(C_P(S_0), C_P(S_1), C_P(S_2)) = S_4 \) always holds.

Naive Enforcement of \( R_P \) - Problem 1

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

Naive Enforcement of \( R_P \) - Problem 2

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

How to Fix the Two Problems?

- Naive enforcement of \( R_P \) generates incorrect answer.
- How can we fix it?

Object and Operation Equivalency

- Object equivalence
  
  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. \( C \) if: (1) \( C(X) \) and \( C^{-1}(R) \) hold for XML node set \( X \) and relation \( R \), we consider both \( X \) and \( R \) equivalent w.r.t. \( C \) if:

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 of \( C(n) \) is in \( C(P) \), e.g., it should recognize if \( C(n) \) is a descendant of any node \( C(P) \).
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:

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

Does NOT generate new tree

A negative rule in ACR restricts user from access a set of nodes \( r_{1}^{-}, ..., r_{n}^{-} \). 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.

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 \( C(n_{2}) \).

Equivalent Conversion of Deep Except (5)

- Example: XRel

\[
\text{DOCID} \quad \text{ELEMENTID} \quad \text{PATHID} \quad \text{ST} \quad \text{ED} \\
0 \quad 262 \quad 161 \quad 13990 \quad 23220 \quad \text{Person} \\
0 \quad 229 \quad 181 \quad 6102 \quad 1117 \quad \text{Creditcard} \\
\]

//person deep-except //creditcard Desired output (new person nodes)

Not feasible in relational algebra!

Equivalent Conversion of Deep Except (4)

- Example: Shared-inlining

<table>
<thead>
<tr>
<th>Person</th>
<th>Person_name</th>
<th>Person_Creditcard</th>
<th>Creditcard</th>
<th>Creditcard</th>
<th>Creditcard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rens Rifaut</td>
<td>xxxxxxxxx</td>
<td>8814 4447 1702 6117</td>
<td>Schellos</td>
<td>Soheil Cosar</td>
<td>8116 1498 1997 5316</td>
</tr>
</tbody>
</table>

- //person deep-except //creditcard
- Output (remove creditcard column)

<table>
<thead>
<tr>
<th>Person</th>
<th>Person_name</th>
<th>Person_Creditcard</th>
<th>Creditcard</th>
<th>Creditcard</th>
<th>Creditcard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rens Rifaut</td>
<td>xxxxxxxxx</td>
<td>8814 4447 1702 6117</td>
<td>Schellos</td>
<td>Soheil Cosar</td>
<td>8116 1498 1997 5316</td>
</tr>
</tbody>
</table>

- Could be easily implemented in RDBMS
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

Questions?

Thank you!

Background: XML-2-Relational Conversion

- An X-2-R conversion algorithm is Lossless:
  - Lossless node conversion: \( C_D(x_j) = x_i \)
  - Lossless node set decomposition:
    \( C_D^{-1}(C_D(x_{j1}) \ldots x_{jn}) = C_D^{-1}(C_D(x_{j1}) \ldots C_D(x_{jn})) = C_D^{-1}(C_D(x_{j1})) \ldots C_D^{-1}(C_D(x_{jn})) \)
  - Exclusive conversion:
    \( C_D(x_{j1}) = C_D(x_{j2}) \) only when \( x_{j1} = x_{j2} \), and \( C_D^{-1}(r_{j1}) = C_D^{-1}(r_{j2}) \) only when \( r_{j1} = r_{j2} \)

- An X-2-R conversion algorithm is Correct:
  - Correct query processing:
    \( C^{-1}(Q_A \cap D_A) = C^{-1}(C_A(Q_A) \cap C_A(D_A)) \)

- An X2R conversion algorithm A is sound iff it is lossless and correct.