Secure XML Database Access with Views

SecReT’09

Benoit Groz
(joint work with Anne-Cécile Caron, Yves Roos, Sławek Staworko, Sophie Tison)

Mostrare

10 juillet 2009
Securing databases with views

Many ways to enforce access control for XML. Among others:

- Checking the queries:
  - statically ⇒ may reject proper queries and access
    [Oasis project: XACML]
  - dynamically ⇒ incurs costly runtime security check
    [Murata et al. CCS’03]
Securing databases with views

Many ways to enforce access control for XML. Among others:

- Checking the queries:
  - statically ⇒ may reject proper queries and access [Oasis project: XACML]
  - dynamically ⇒ incurs costly runtime security check [Murata et al. CCS’03]

- Annotating the data:
  - annotating the data, or materializing the view ⇒ expensive maintenance [Damiani et al. EDBT’00, Cho et al. VLDB’02]
  - annotating the DTD with Non-materialized view

Rewriting queries from the view to the document [Fan et al. SIGMOD’04, Vercammen et al, Rassadko et al . . . ]
Outline

1. Non-materialized views and query rewriting

2. Comparing Access Control Policies
“Whoever wishes to keep a secret must hide the fact that he possesses one”.

attributed to Johann Wolfgang von Goethe
Answer to query $Q = \text{evaluation of } Q'$ on the original document $t$
Framework: XML

- XML document = tree.
- No data-values.

<projects>
  <project>
    <name>
    </name>
    ... license
    src doc free
  </project>
  <project>
    ... license
    src doc free
  </project>
</projects>
We use *Regular XPath* queries

Query $q_1 = \downarrow^* / \downarrow :: \text{doc}$

$\text{Ans}(q_1, t) = \{n_{11}, n_{14}\}$

"get all documentations"
We use Regular XPath queries

Query $q_2 = \downarrow :: project[\downarrow :: stable]/\downarrow :: name$

$\text{Ans}(q_2, t) = \{n_3\}$

“get names of stable projects”
Access control for XML

We wish to hide:

- whether a project is *stable* or *in-development*
- the *binaries*
- the *sources* for non-free projects
DTD and Annotation

Example

\[
\begin{align*}
\text{projects} & \rightarrow \text{project}^* \\
\text{project} & \rightarrow \text{name},(\text{stable} \mid \text{dev}),\text{license} \\
A_0(\text{project},\text{stable}) &= \text{false} \\
A_0(\text{project},\text{dev}) &= \text{false} \\
\text{license} & \rightarrow \text{free} \mid \text{propr} \\
\text{stable} & \rightarrow \text{src},\text{bin},\text{doc} \\
A_0(\text{stable},\text{src}) &= [\uparrow^*:\text{project} / \downarrow^*:\text{free}] \\
A_0(\text{stable},\text{doc}) &= \text{true} \\
\text{dev} & \rightarrow \text{src},\text{doc} \\
A_0(\text{dev},\text{src}) &= [\uparrow^*:\text{project} / \downarrow^*:\text{free}] \\
A_0(\text{dev},\text{doc}) &= \text{true}
\end{align*}
\]
The security view

document $t$

View $A(t)$
Annotating the DTDs

▷ annotation as a function $A : \Sigma \times \Sigma \rightarrow \{\text{true, false, } [f]\}$.

### Example

<table>
<thead>
<tr>
<th>Category</th>
<th>Element</th>
<th>Annotation Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>projects</td>
<td>project*</td>
<td></td>
</tr>
<tr>
<td>project</td>
<td>name, (stable</td>
<td>$A_0(\text{project, stable}) = \text{false}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(dev), license</td>
<td>$A_0(\text{project, dev}) = \text{false}$</td>
</tr>
<tr>
<td>license</td>
<td>free</td>
<td></td>
</tr>
<tr>
<td>stable</td>
<td>src, bin, doc</td>
<td>$A_0(\text{stable, src}) = [\uparrow^<em>:\text{project}/\downarrow^</em>:\text{free}]$</td>
</tr>
<tr>
<td>dev</td>
<td>src, doc</td>
<td>$A_0(\text{dev, src}) = [\uparrow^<em>:\text{project}/\downarrow^</em>:\text{free}]$</td>
</tr>
</tbody>
</table>

Proposition: This model of annotation is equivalent to defining accessible elements with a $X_{Reg}$ filter $f_A$ such that:

$$\forall n \in \mathbb{N}, n \text{ accessible wrt. } A \iff (t, n) \models f_{A_{acc}}$$
Annotating the DTDs

▷ annotation as a function $A : \Sigma \times \Sigma \rightarrow \{\text{true, false, }[f]\}$.

Example

| projects $\rightarrow$ project* | stable $\rightarrow$ src, bin, doc |
| project $\rightarrow$ name, (stable | dev), license | $A_0$(stable, src) $= [\uparrow^*::project/\downarrow^*::free]$ |
| $A_0$(project, stable) $= \text{false}$ | $A_0$(stable, doc) $= \text{true}$ |
| $A_0$(project, dev) $= \text{false}$ | dev $\rightarrow$ src, doc |
| license $\rightarrow$ free | $A_0$(dev, src) $= [\uparrow^*::project/\downarrow^*::free]$ |
| $\mid$ propr | $A_0$(dev, doc) $= \text{true}$ |

Proposition

This model of annotation is equivalent to defining accessible elements with a $\text{\textit{XReg}}$ filter $f^A_{\text{acc}}$ such that:

$\forall n \in N_t. \ n \text{ accessible wrt. } A \iff (t, n) \models f^A_{\text{acc}}$
Rewriting Queries

Theorem: *Regular XPath* is closed under query rewriting

There exists a function Rewrite such that:

\[ \forall t. \text{Ans}(Q, A(t)) = \text{Ans}(\text{Rewrite}(Q, A), t) \]

Moreover, Rewrite(Q, A) is computable in time \( O(|A| \ast |Q|) \).
Rewriting Queries

Theorem: Regular XPath is closed under query rewriting

There exists a function Rewrite such that:

\[
\forall t. \text{Ans}(Q, A(t)) = \text{Ans}(\text{Rewrite}(Q, A), t)
\]

Moreover, Rewrite\((Q, A)\) is computable in time \(O(|A| \times |Q|)\).

Proof.

Translate the base axes using \(f^A_{\text{acc}}\):

\[
\text{Rewrite}(\uparrow, A) = \text{self}[f^A_{\text{acc}}]/(\uparrow[\neg f^A_{\text{acc}}])^*/\text{self}[f^A_{\text{acc}}]
\]

Rewrite the query inductively.
Rewriting Queries

Hidden part

User part

Query $Q' = \text{Rewrite}(Q, A)$

Answer

projects

project

name

stable

license

src
bin
doc
free

project

name

stable

license

src
bin
doc
propr

project

name

dev

license

src
doc
free

B. Groz, S. Staworko et al (Mostrare) Secure XML Database Access with Views 10 juillet 2009 12 / 19
Rewriting Queries

Hidden part

User part

\[ Q' = \downarrow :: \text{project}[\text{license}/\text{free}] / \downarrow :: * / \downarrow :: \text{src} \]

\[ Q = \downarrow :: \text{project} / \downarrow :: \text{src} \]

Query \( Q \)

Answer

\( Q' = \text{Rewrite}(Q, A) \)

document \( t \)

\begin{itemize}
  \item projects
  \begin{itemize}
    \item project
      \begin{itemize}
        \item name: stable
        \item license
        \begin{itemize}
          \item src
          \item bin
          \item doc
          \item free
        \end{itemize}
      \end{itemize}
    \end{itemize}
  \end{itemize}
\end{itemize}
Outline

1. Non-materialized views and query rewriting

2. Comparing Access Control Policies
Comparing access control policies

Definition

Two annotations $A_1$ and $A_2$ over DTD $D$ are \textit{equivalent} iff they hide the same nodes:

$$A_1 \equiv^D A_2 \text{ iff } \forall t \in L(D). \ A_1(t) = A_2(t)$$
Comparing access control policies

Definition

Two annotations $A_1$ and $A_2$ over DTD $D$ are equivalent iff they hide the same nodes:

$$A_1 \equiv^D A_2 \text{ iff } \forall t \in L(D). \ A_1(t) = A_2(t)$$

Proposition

Testing equivalence of annotations is $EXPTIME$-complete.

Proof.

This problem is polynomially equivalent to the problem of equivalence of $\mathcal{X} Reg$ filters over a DTD.
Comparing Access control policies

Definition

$A_1$ and $A_2$ annotations over DTD $D$. $A_1$ is \textit{1-restriction} of $A_2$ in the presence of $D$, denoted

$$A_1 \preceq^D_1 A_2 \iff \forall t \in L(D). \ N_{A_1}(t) \subseteq N_{A_2}(t)$$

Intuition:

The simplest way for comparing two annotations: $A_1$ is more “restrictive” than $A_2$ if it shows no element hidden by $A_2$.

Proposition

Testing 1-restriction is \textit{EXPTIME}-complete.
Does \( \preceq_1 \) ensure the properties we expect?

Example

\[
\begin{align*}
\text{projects} & \rightarrow \text{project}\star \\
\text{project} & \rightarrow \text{name},(\text{stable} \mid \text{dev}),\text{license} \\
A_0(\text{project},\text{stable}) &= \text{false} \\
A_0(\text{project},\text{dev}) &= \text{false} \\
\text{license} &\rightarrow \text{free} \mid \text{propr} \\
\text{stable} &\rightarrow \text{src},\text{bin},\text{doc} \\
A_0(\text{stable},\text{src}) &= [\uparrow^*:\text{project}/\downarrow^*:\text{free}] \\
A_0(\text{stable},\text{doc}) &= \text{true} \\
\text{dev} &\rightarrow \text{src},\text{doc} \\
A_0(\text{dev},\text{src}) &= [\uparrow^*:\text{project}/\downarrow^*:\text{free}] \\
A_0(\text{dev},\text{doc}) &= \text{true}
\end{align*}
\]
Does $\preceq_1$ ensure the properties we expect?

**Example**

- $\text{projects} \rightarrow \text{project}^*$
- $\text{project} \rightarrow \text{name}, (\text{stable} | \text{dev}), \text{license}$
  - $A_0(\text{project}, \text{stable}) = \text{false}$
  - $A_0(\text{project}, \text{dev}) = \text{false}$
- $\text{license} \rightarrow \text{free} | \text{propr}$

- $\text{stable} \rightarrow \text{src}, \text{bin}, \text{doc}$
- $A_0(\text{stable}, \text{src}) = [\uparrow^*::\text{project}/\downarrow^*::\text{free}]$
- $A_0(\text{stable}, \text{doc}) = \text{true}$

- $\text{dev} \rightarrow \text{src}, \text{doc}$
- $A_0(\text{dev}, \text{src}) = [\uparrow^*::\text{project}/\downarrow^*::\text{free}] \text{false}$
- $A_0(\text{dev}, \text{doc}) = \text{true}$
Does $\lesssim_1$ ensure the properties we expect?

**Example**

$$\text{projects} \rightarrow \text{project}^*$$
$$\text{project} \rightarrow \text{name},(\text{stable} \mid \text{dev}),\text{license}$$
$$A_0(\text{project},\text{stable}) = \text{false}$$
$$A_0(\text{project},\text{dev}) = \text{false}$$
$$\text{license} \rightarrow \text{free} \mid \text{propr}$$

$$\text{stable} \rightarrow \text{src},\text{bin},\text{doc}$$
$$A_0(\text{stable},\text{src}) = \lceil\text{*:project} / \text{*:free}\rceil$$
$$A_0(\text{stable},\text{doc}) = \text{true}$$
$$\text{dev} \rightarrow \text{src},\text{doc}$$
$$A_0(\text{dev},\text{src}) = \lceil\text{*:project} / \text{*:free}\rceil\text{false}$$
$$A_0(\text{dev},\text{doc}) = \text{true}$$

**Document t**

- **User can select all projects under free license that are not stable!**
An information-oriented comparison

Argument

$A_1$ should be more “restrictive” than $A_2$ if every information inferred from $A_1$ can be inferred from $A_2$.

Definition

$A_1$ and $A_2$ annotations over DTD $D$. $A_1$ is 2-restriction of $A_2$ in the presence of $D$, denoted

$$A_1 \preceq^D_2 A_2 \text{ iff } \forall Q_1 \exists Q_2. \forall t \in L(D). \text{Ans}(Q_1, A_1(t)) = \text{Ans}(Q_2, A_2(t))$$
An information-oriented comparison

Definition

$A_1$ and $A_2$ annotations over DTD $D$. $A_1$ is a 2-restriction of $A_2$ in the presence of $D$, denoted

$$A_1 \preceq^D_2 A_2 \text{ iff } \forall Q_1 \exists Q_2. \forall t \in L(D). \text{Ans}(Q_1, A_1(t)) = \text{Ans}(Q_2, A_2(t))$$

Theorem

This property is undecidable.
An information-oriented comparison

Definition

$A_1$ and $A_2$ annotations over DTD $D$. $A_1$ is 2-restriction of $A_2$ in the presence of $D$, denoted

$$A_1 \preceq_D^2 A_2 \iff \forall Q_1 \exists Q_2. \forall t \in L(D). \text{Ans}(Q_1, A_1(t)) = \text{Ans}(Q_2, A_2(t))$$

Theorem

This property is undecidable.

Alternative characterization

$A_1 \preceq_D^2 A_2$ if and only if

$$\exists f. \forall t \models D \forall n \in N_{A_2}(t). (n, A_2(t)) \models f \iff n \in N_{A_1}(t)$$

$\models$ : if filter $f$ is provided, then one can verify the property in $\text{EXPTIME}$
An information-oriented comparison

**Definition**

$A_1$ and $A_2$ annotations over DTD $D$. $A_1$ is 2-restriction of $A_2$ in the presence of $D$, denoted

$$A_1 \preceq^2_{D} A_2 \text{ iff } \forall Q_1 \exists Q_2. \forall t \in L(D). \text{Ans}(Q_1, A_1(t)) = \text{Ans}(Q_2, A_2(t))$$

**Theorem**

This property is undecidable.

**Theorem**

However, for non-recursive DTDs, 2-restriction can be tested in EXPTIME
Further work

- implementation
- update propagation
- richer schema and query language
- other view formalisms