Privacy violation detection in XML databases

Joint work with Rita Steinmetz

Motivation

Motivation/Scenario 1

Alice

blood test

Health_Co

Med_Care
Motivation/Scenario 2

Motivation/Scenario 3

Who did reveal information on Jane’s bank account to a third party?
Common Assumptions for Scenarios 1-3

- Sensitive data is stored in an XML database with known DTD or XML schema
- only accessible to users that query this database, i.e. not communicated using other tools (e.g. fax, printout,…)
- accessed by multiple users via XPath queries
- Sensitive data has been given illegally to a third party by one of the users querying the XML database

Common goal:
reduce the number of suspicious users by analyzing the query protocol

Problem Description and Used Example

- idea: transform "secret" into an "audit query" A, such that the audit query returns an answer if the secret is uncovered
- "secret": balance of Jane’s bank account is negative
- audit query: /Bank/Department/Customer
  [Name="Jane"][/Balance<0]

- given: audit query A
- user query Q
- question: did Q access information specified by A, i.e., can A≠∅ be inferred from answer to Q?
System Architecture (1) Normal Operation

Query / Write operation + UserID + Timestamp

Privacy Layer

<table>
<thead>
<tr>
<th>Original Database</th>
<th>Query log</th>
<th>Backlog</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query / Write operation</td>
<td>Query + UserID + Timestamp</td>
<td>insert / delete / update (+UserID) + Timestamp</td>
</tr>
</tbody>
</table>

System Architecture (2) Audit Process

Query / Write operation + UserID + Timestamp

Privacy Layer

<table>
<thead>
<tr>
<th>Original Database</th>
<th>Query log</th>
<th>Backlog</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query / Write operation</td>
<td>Query + UserID + Timestamp</td>
<td>insert / delete / update (+UserID) + Timestamp</td>
</tr>
</tbody>
</table>

Audit query

Suspicious queries
Overview

statical analysis: (Queries in time intervall)
select queries with a timestamp between
during datetime and to datetime

structural analysis (Candidate Queries):
select queries with a structure
similar to the structure of the audit query

data analysis (Suspicious Queries):
Which queries have „touched“ the secret, i.e. the data specified by the audit query?

Audit Expression

AuditExp ::= ‘during’ datetime ‘to’ datetime
‘audit’ Path

Path ::= Σ | Path‘/’Path | Path‘/’/’Path | Path ‘[‘ FExp ‘]’
FExp ::= Path | Path BoolOp constant
**Example**

- **"Secret":** The balance of Jane’s bank account is negative
- **Audit query:** /Bank/Department/Customer
  
  [Name = “Jane”][ //Balance < 0]

- **User query which is currently examined:**
  /Bank/Department/Customer
  
  [Account/Balance < 0]/Name

---

**Tree Pattern**

- **Audit query:** /Bank/Department/Customer
  
  [Name = “Jane”][ //Balance < 0]

- **One node for each node name test**
- **One single edge for each child axis location step**
- **One double edge for each descendant axis location step**
- **One CompNode for each comparison with a value**
**Candidate Queries**

**Idea (Candidate queries):**
Let TQ be the transformed tree pattern of the user query Q, and let TA be the tree pattern of the audit query A.

Query Q is a candidate query with respect to audit query A if and only if the structure of Q combines all the information described or selected by the structure of A.

Note that: Name alone or Balance alone is non-critical.

---

**Definition 1:**
Let TQ be the transformed tree pattern of the user query Q, and let TA be the tree pattern of the audit query A.

**Q is a candidate query w.r.t. A** if and only if there is a homomorphism $h: \text{Nodes}(TA) \rightarrow \text{Nodes}(TQ)$ so that

1. $h(\text{root}(TA)) = \text{root}(TQ)$
2. $\forall x \in \text{Nodes} (TA) : \text{Label}(x) = \text{Label}(h(x))$
3a. $\forall x,y \in \text{Nodes} (TA) :$ $(x,y)$ is a child-edge in TA $\Rightarrow$ $(h(x),h(y))$ is a child-edge in TQ
3b. $\forall x,y \in \text{Nodes} (TA) :$ $(x,y)$ is a descendant-edge in TA $\Rightarrow$ $h(y)$ is a descendant of $h(x)$ in TQ.
Transformed Tree Patterns for User Queries

Idea (transformed tree pattern user query TQ):

Goal: Find embedding

Use DTD information to transform query

TA (audit A) tree pattern for Q

TA (audit A) TQ (candidate Q)

Suspicious Queries

Idea behind suspicious queries Q:
Let Dt be the state of the database D at time t when Q is executed. We call Q suspicious with respect to an audit query A and database D if Q(Dt) depends on nodes of Dt relevant to the query A.
Let TQ be the transformed tree pattern of the user query Q. Let TD be the tree representation of document Dt. A mapping e:Nodes(TQ)→Nodes(TD) is an embedding iff

1. e(root(TQ)) = root(TD)
2. ∀ x ∈ Nodes (TQ) : Label (x) = Label(e(x))
3a. ∀ x,y ∈ Nodes (TQ): (x,y) is a child-edge in TQ ⇒ (h(x),h(y)) is a child-edge in TD
3b. ∀ x,y ∈ Nodes (TQ): (x,y) is a descendant-edge in TQ ⇒ h(y) is a descendant of h(x) in TD
4. ∀ x ∈ Nodes (TQ) ∀ y ∈ CompNodes (TQ): (x,y) is a child-edge in TQ ⇒ e(x) fulfills condition stated in label(y)

**Embedding**

**readSet(Q(Dt))**

readSet( Q( Dt ) ) is the union of all result nodes of embeddings of Q in TD plus their paths to the root of Dt
Suspicious Queries

Definition 2 (Query Q is suspicious w.r.t. A and D):
Let $D_t$ be the state of the database $D$ at time $t$ when $Q$ is executed. We call $Q$ suspicious with respect to an audit query $A$ and database $D$ if $A(\text{readSet}(Q(D_t))) \neq \emptyset$.

Algorithm

\begin{align*}
\text{audit}(\text{AuditQuery} \ A, \ \text{querylog} \ QL, \ \text{DTD} \ D) \{ & \\
\hspace{1em} 1. & Q,C,S := \emptyset; \\
\hspace{1em} 2. & \text{for each } (q \in QL) \{} \\
\hspace{2em} & 3. \hspace{1em} \text{if}(A.\text{during} \leq q.\text{timestamp} \leq A.\text{to}) \\
\hspace{2em} & 4. \hspace{1em} Q := Q \cup \{q\}; \} \\
\hspace{1em} 5. & \text{for each } (q \in Q) \{} \\
\hspace{2em} & 6. \hspace{1em} q' := \text{transform}(q,D); \\
\hspace{2em} & 7. \hspace{1em} \text{if}(\exists \text{Homomorphism}(A,q')) \\
\hspace{2em} & 8. \hspace{1em} C := C \cup q; \} \\
\hspace{1em} 9. & \text{for}(q = C.\text{newest}; C.\text{moreQueries}; C.\text{next}) \\
\hspace{2em} & 10. \{ \text{Dt} := \text{restoreDB}(\text{lastTime},q.\text{timestamp}); \\
\hspace{3em} & 11. \hspace{1em} \text{lastTime} := q.\text{timestamp} \\
\hspace{3em} & 12. \hspace{1em} \text{if} (A(\text{treeCopyOf}(Q(D_{t}))) \neq \emptyset) \\
\hspace{3em} & 13. \hspace{1em} S := S \cup \{q\}; \} \\
\hspace{1em} 14. & \text{return } S; \} \quad // \ S = \text{set of suspicious queries}
\end{align*}
Summary

**Statistical analysis:** (Queries in time interval)
select queries with a timestamp between
during datetime and to datetime

**Structural analysis** (Candidate Queries):
homomorphism of tree patterns:
$TA(\text{audit query}) \rightarrow TQ(\text{user query}^{DTD})$

data analysis (Suspicious Queries):
$A(\text{readSet}(\text{Q(Dt)})) \neq \emptyset$
where $\text{readSet}(\text{Q(Dt)}) = \text{union of all embeddings of TQ into Dt}$

Conclusions

- Overall runtime is in PTIME
- Suspicious queries return are a superset of queries uncovering the secret
- Only single queries are examined, sometimes information can be revealed by a series of queries
- Combination $(\text{database, backlog})$ might be replaced with a temporal database
- We assume that this approach is easily adaptable to other query languages like XQuery, XSLT