Privacy violation detection in XML databases

Joint work with Rita Steinmetz
Motivation/Scenario 1

Alice → blood test → Health_Co
Alice → blood test → Med_Care
Motivation/Scenario 2

- Secret data
- Employees reveal data to third party

Motivation

Problem

Architecture

Approach

Conclusions
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 secret is uncovered if the user could prove that A evaluates to true
- **"secret":** balance of Jane’s bank account is negative
- **audit query:** exists /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?
Problem Description

let
Q be a user query,
Dt be a database state at time t
R be the result of applying Q to Dt, and
A be the audit query,

then the secret of A is uncovered, if and only if

\[ Q(Dt) = R \implies A \]

general case \(\implies\) theorem prover (too complex)

\(\rightarrow\) only solutions and ideas for small subsets

\(\rightarrow\) XPath queries

- provably suspicious
- ?
- provably non-suspicious
System Architecture (1) Normal Operation

Motivation

Problem

Architecture

Approach

Conclusions

Original Database

Query log

Backlog

Query / Write operation + UserID + Timestamp

Privacy Layer

Query / Write operation

Query + UserID + Timestamp

insert / delete / update (+UserID) + Timestamp
System Architecture (2) Audit Process

Query / Write operation + UserID + Timestamp

Privacy Layer

<table>
<thead>
<tr>
<th>Query / Write operation</th>
<th>Query + UserID + Timestamp</th>
<th>insert / delete / update (+UserID) + Timestamp</th>
</tr>
</thead>
</table>

Original Database

Query log

Backlog

Suspicious queries

Audit query
**Overview**

**Statistical analysis**: (Queries in time interval)
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?
XPath Subset for Audit Expression and Query

\[
\text{AuditExp} ::= \text{‘during’ datetime ‘to’ datetime ‘audit’ exists Path}
\]

\[
\text{Path} ::= \Sigma | \text{Path}/\text{Path} | \text{Path}//\text{Path} | \text{Path} [\text{FExp}]'
\]

\[
\text{FExp} ::= \text{Path} | \text{Path BoolOp constant}
\]

\[
\text{BoolOp} ::= \text{‘=’} | \text{‘<’} | \text{‘>’} | \text{‘≥’} | \text{‘≤’} | \text{‘≠’}
\]
“Secret”: The balance of Jane’s bank account is negative

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

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

Audit query: exists /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 $h:\text{Nodes}(TA)\to\text{Nodes}(TQ)$ 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 $TQ$. 

![Diagram](image)
Idea behind *suspicious* queries \( Q \):
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 \( Q(D_t) \) depends on nodes of \( D_t \) 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 $e: \text{Nodes}(TQ) \rightarrow \text{Nodes}(TD)$ is an embedding iff

1. $e(\text{root}(TQ)) = \text{root}(TD)$
2. $\forall x \in \text{Nodes}(TQ):$ $\text{Label}(x) = \text{Label}(e(x))$
3a. $\forall x,y \in \text{Nodes}(TQ):$ $(x,y)$ is a child-edge in $TQ$ $\Rightarrow$ $(h(x),h(y))$ is a child-edge in $TD$
3b. $\forall x,y \in \text{Nodes}(TQ):$ $(x,y)$ is a descendant-edge in $TQ$ $\Rightarrow$ $h(y)$ is a descendant of $h(x)$ in $TD$
4. $\forall x \in \text{Nodes}(TQ)$ $\forall y \in \text{CompNodes}(TQ):$ $(x,y)$ is a child-edge in $TQ$ $\Rightarrow$ $e(x)$ fulfills condition stated in $\text{label}(y)$
The diagram illustrates the concept of finding suspicious candidates in XML databases. The readSet(Q(Dt)) is the union of all result nodes of embeddings of Q in TD plus their paths to the root of Dt.
Definition 2 (Query Q is suspicious w.r.t. A and D):
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 $A(\text{readSet}(Q(D_t))) = true$.
**Transformed Tree Patterns for User Queries**

**Idea (transformed user query tree pattern TQ):**

Goal: Find embedding  
Use DTD information to transform query

---

**Example**

**Tree Pattern**

**Candidates**

**Suspicious**

**Conclusions**

---

Databases and Information Systems 1 - Prof. Dr. Stefan Böttcher - WS 2005/06 - Privacy violation detection in XML databases 19/21
Algorithm

\begin{algorithm}
\begin{algorithmic}
\State \textbf{audit}(AuditQuery }A\text{, querylog }QL\text{, DTD }D\{\\)
\State \hspace{1em} Q,C,S := \emptyset; \\
\State \hspace{1em} \textbf{for each } (q \in QL) \{ \\
\State \hspace{2em} \textbf{if}(A.during \leq q.timestamp \leq A.to) \\
\State \hspace{2em} \hspace{1em} Q := Q \cup \{q\}; \\
\State \hspace{1em} \textbf{for each } (q \in Q) \{ \\
\State \hspace{2em} \hspace{1em} q' := \text{transform}(q,D); \\
\State \hspace{2em} \hspace{2em} \textbf{if}(\text{existsHomomorphism}(A,q')) \\
\State \hspace{2em} \hspace{2em} \hspace{1em} C := C \cup q; \\
\State \hspace{2em} \hspace{2em} \textbf{for}(q=C.newest;C.moreQueries;C.next) \\
\State \hspace{2em} \hspace{2em} \hspace{2em} Dt:=\text{restoreDB(lastTime,q.timestamp)}; \\
\State \hspace{2em} \hspace{2em} \hspace{2em} \text{lastTime}:=q.timestamp \\
\State \hspace{2em} \hspace{2em} \hspace{2em} \text{if } A(\text{readSet}(Q(Dt))) \\
\State \hspace{2em} \hspace{2em} \hspace{2em} \hspace{2em} \hspace{1em} S := S \cup \{q\}; \\
\State \hspace{2em} \hspace{2em} \textbf{return } S; \} \hspace{1em} // S = \text{set of suspicious queries} \\
\end{algorithmic}
\end{algorithm}
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 (database, backlog) might be replaced with a temporal database
- We assume that this approach is easily adaptable to other query languages like XQuery, XSLT