Privacy violation detection in XML databases

Joint work with Rita Hartel
Motivation/Scenario 1

employees reveal data to third party

Motivation:
employees reveal data to third party

secret data

Approach:

Conclusions:
Motivation/Scenario 2

Alice → blood test → Health_Co

Alice → Med_Care
Motivation/Scenario 3

Jane

MoneyCo

Who did reveal information on Jane’s bank account to a third party?

Bank account
Name: Jane
Balance: -2431 $

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

Query log

Backlog
System Architecture (2) Audit Process

Motivation

Problem

Architecture

Approach

Conclusions

Privacy Layer

Query / Write operation + UserID + Timestamp

Suspicious queries

System Architecture (2) Audit Process

Query / Write operation + UserID + Timestamp

Query + UserID + Timestamp

insert / delete / update (+UserID) + Timestamp

Original Database

Query log

Backlog

Audit query

Query / Write operation

Original Database

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

BoolOp ::= '=' | '<' | '>' | '>=' | '<=' | '!='
"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
Audit query: /Bank/Department/Customer
[Name="Jane"][/Balance<0]
Tree Pattern of an XPath query

Query: / Bank / Department / Customer
[ Name="Jane" ] [ //Balance<0 ]

- One Element 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 Comparison Node for each comparison with a value
Embedding tree patterns in an XML document

Idea: Map each element node of the tree pattern to an element node in the XML document with the same name in such a way that hierarchy relationships are preserved.

/ Bank / Department / Customer [ Name="Jane" ] [ // Balance<0 ]
Embedding tree patterns in an XML document

Let \( TQ \) be the tree pattern of the user query \( Q \).
Let \( TD \) be the tree representation of the XML document.
A mapping \( e:\text{Nodes}(TQ) \rightarrow \text{Nodes}(TD) \) is an embedding iff

1. \( e(\text{root}(TQ)) = \text{root}(TD) \)
2. \( \forall x \in \text{Nodes}(TQ) : \)
   \( \text{Label}(x) = \text{Label}(e(x)) \) or \( \text{Label}(x) = "*" \)
3a. \( \forall x,y \in \text{Nodes}(TQ) : \)
    \( (x,y) \) is a child-edge in \( TQ \) \( \Rightarrow \)
    \( (e(x),e(y)) \) is a child-edge in \( TD \)
3b. \( \forall x,y \in \text{Nodes}(TQ) : \)
    \( (x,y) \) is a descendant-edge in \( TQ \) \( \Rightarrow \)
    \( e(y) \) is a descendant of \( e(x) \) in \( TD \)
4. \( \forall x \in \text{Nodes}(TQ) \)
   \( \forall y \in \text{ComparisonNodes}(TQ) : \)
   \( (x,y) \) is a child-edge in \( TQ \) \( \Rightarrow \)
   \( e(x) \) fulfills condition stated in \( \text{label}(y) \)

/ Bank / Department / Customer [ Name="Jane" ] [ / // Balance<0 ]
Homomorphism between Tree Patterns

Query Q represented by TQ is more specific than query A represented by TA, i.e. TQ ⊑ TA, means:

for every XML document D: A(D) ⊇ Q(D)

```
/ * [ // C ] [ D ]    / B [ D / C ] [ E ]
```

Homomorphism between Tree Patterns

If there is a homomorphism between two Tree Patterns $TA$ and $TQ$

$h: Nodes(TA) \to Nodes(TQ)$ such that

1. $h(\text{root}(TA)) = \text{root}(TQ)$

2. $\forall x \in \text{Nodes}(TA) :$
   - $\text{Label}(x) = \text{Label}(h(x))$ or $\text{Label}(x) = "\ast"$

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 $TA$ $\Rightarrow$
   - $h(x)$ is a descendant of $h(x)$ in $TQ$.

4. $\forall x \in \text{Nodes}(TA) \forall y \in \text{ComparisonNodes}(TA):$
   - $(x,y)$ is a child-edge in $TA$ $\Rightarrow$
   - $(h(x),h(y))$ is a child-edge in $TQ$
   - $h(y)$ is at least as specific as $y$, i.e. "Path $h(y)$" $\Rightarrow$ "Path $y$"

and $h(\text{outputNode}(TA)) = \text{outputNode}(TQ)$,
then the query represented by $TQ$ is more specific than the query represented by $TA$, i.e. $TQ \sqsubseteq TA$
Exercise Tree Patterns / Homomorphism

Given \( Q_1 = // \ B \ / \ C \ / \ D \)
\[ Q_2 = // \ * \ [ \ C \ ] \ // \ D \]
\[ Q_3 = / \ B \ [ \ C \ ] \ // \ D \]
\[ Q_4 = // \ B \ [ \ C \ ] \ / \ C \ / \ D \]

1. Draw tree patterns for \( Q_1, Q_2, Q_3, Q_4 \)
2. Check for homomorphisms among these tree patterns
3. Consequences for query containment?

\[
\begin{align*}
\text{TA} & \rightarrow \text{TQ} \\
\text{A(D)} & \supseteq \text{Q(D)} \\
* & \rightarrow B \\
\emptyset & \rightarrow E \\
// & \rightarrow /*/\ldots/
\end{align*}
\]
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 $T_Q$ be the transformed tree pattern of the user query $Q$, and let $T_A$ 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 $h: \text{Nodes}(T_A) \rightarrow \text{Nodes}(T_Q)$ so that

1. $h(\text{root}(T_A)) = \text{root}(T_Q)$
2. $\forall x \in \text{Nodes}(T_A): \text{Label}(x) = \text{Label}(h(x))$
3a. $\forall x, y \in \text{Nodes}(T_A): (x, y) \text{ is a child-edge in } T_A \Rightarrow (h(x), h(y)) \text{ is a child-edge in } T_Q$
3b. $\forall x, y \in \text{Nodes}(T_A): (x, y) \text{ is a descendant-edge in } T_A \Rightarrow h(y) \text{ is a descendant of } h(x) \text{ in } T_Q$. 

Example

Tree Pattern

Candidates

Suspicious

Conclusions

Databases and Information Systems 1 - Prof. Dr. Stefan Böttcher -WS 2009/2010 - Privacy violation detection in XML databases 19/27
**Transformed Tree Patterns for User Queries**

**Idea (transformed tree pattern user query TQ):**
- **Goal:** Find embedding
- **Use DTD information to transform query**

**Example**

```
root
  Bank
  Department
    Customer
      Balance
      Name
  Account
    Balance
    Name
```

**TA (audit A) tree pattern for Q**

```
root
  Bank
    Department
      Customer
        Balance
        Name
  Account
    Balance
    Name
```

**TA (audit A) TQ (candidate Q)**

```
root
  Bank
    Department
      Customer
        Balance
        Name
  Account
    Balance
    Name
```

**Candidates**

```
<0
```

**Suspicious**

```
<0
```
Motivation

Problem

Architecture

Approach

Conclusions

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 $e: \text{Nodes}(TQ) \rightarrow \text{Nodes}(TD)$ is an embedding iff

1. $e(\text{root}(TQ)) = \text{root}(TD)$

2. $\forall x \in \text{Nodes} (TQ): \quad \text{Label}(x) = \text{Label}(e(x))$

3a. $\forall x, y \in \text{Nodes} (TQ): \quad (x, y)$ is a child-edge in $TQ \Rightarrow (h(x), h(y))$ is a child-edge in $TD$

3b. $\forall x, y \in \text{Nodes} (TQ): \quad (x, y)$ is a descendant-edge in $TQ \Rightarrow h(y)$ is a descendant of $h(x)$ in $TD$

4. $\forall x \in \text{Nodes} (TQ) \quad \forall y \in \text{CompNodes} (TQ): \quad (x, y)$ is a child-edge in $TQ \Rightarrow e(x)$ fulfills condition stated in label($y$)
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 \textit{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 $Q$ \textit{suspicious} with respect to an audit query $A$ and database $D$ if \[ A(\text{readSet}(Q(D_t))) \neq \emptyset. \]
Algorithm

\[
\text{audit}(\text{AuditQuery } A, \text{ querylog } QL, \text{ DTD } D)\{ \\
(1) \text{ Q,C,S } := \emptyset; \\
(2) \text{ for each (q } \in \text{ QL) } \{ \\
(3) \text{ if}(A.\text{during } \leq q.\text{timestamp } \leq A.\text{to}) \\
(4) \text{ Q:=Q } \cup \{q\}; \} \\
(5) \text{ for each (q } \in \text{ Q) } \{ \\
(6) \text{ q'} := \text{ transform}(q,D); \\
(7) \text{ if}(\text{existsHomomorphism}(A,q')) \\
(8) \text{ C:=C } \cup \text{ q}; \} \\
(9) \text{ for(q=C.\text{newest};C.\text{moreQueries};C.\text{next})} \\
(10) \{ \text{ Dt:=restoreDB(lastTime,q\text{.timestamp});} \\
(11) \text{ lastTime:=q\text{.timestamp} } \\
(12) \text{ if } A(\text{treeCopyOf}(Q(Dt))) \neq \emptyset \\
(13) \text{ S:=S } \cup \{q\}; \} \\
(14) \text{ return } S; \} \quad \text{// } S = \text{ set of suspicious queries}
\]
motivation

problem

architecture

approach

conclusions

**Approach**

**Suspicious**

**Summary**

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

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

**data analysis** (Suspicious Queries):
\[ A(\text{readSet}(\text{Q( Dt )})) \neq \emptyset \]
where \( \text{readSet}(\text{Q( Dt )}) = \) 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 (database, backlog) might be replaced with a temporal database
- We assume that this approach is easily adaptable to other query languages like XQuery, XSLT