Data Leakage Detection
Major Project Report
Submitted by
Ankit Kumar Tater : 08J41A1206
Y. V. Pradeep Kumar Reddy : 08J41A1235
Pradeep Saklani : 08J41A1236

Under the Guidance
of
Mr. V. Satish Kumar
Assistant Professor
B.Tech(CSIT), M.Tech (CSE)

to
Jawaharlal Nehru Technological University Hyderabad
in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY
in
INFORMATION TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY
MALLA REDDY ENGINEERING COLLEGE
Maisammaguda Dhulapally, (post Via) Hakimpet,
Hyderabad - 500014
April 2012
Agenda

- INTRODUCTION
- ISSUES
- SCOPE
- ANALYSIS
- DESIGN
- IMPLEMENTATIONS
Introduction

In the course of doing business, sometimes sensitive data must be handed over to supposedly trusted third parties.

Our goal is to detect when the distributor's sensitive data has been leaked by agents, and if possible to identify the agent that leaked the data.
A data distributor has given sensitive data to a set of supposedly trusted agents (third parties). Some of the data is leaked and found in an unauthorized place (e.g., on the web or somebody’s laptop).

The distributor must assess the likelihood that the leaked data came from one or more agents, as opposed to having been independently gathered by other means. We propose data allocation strategies (across the agents) that improve the probability of identifying leakages.

These methods do not rely on alterations of the released data (e.g., watermarks). In some cases we can also inject “realistic but fake” data records to further improve our chances of detecting leakage and identifying the guilty party.
Types of employees that put your company at risk.

- The security illiterate
  - Majority of employees with little or no knowledge of security
  - Corporate risk because of accidental breaches
- The gadget nerds
  - Introduce a variety of devices to their work PCs
  - Download software
- The unlawful residents
  - Use the company IT resources in ways they shouldn't
  - i.e., by storing music, movies, or playing games
- The malicious/disgruntled employees
  - Typically minority of employees
  - Gain access to areas of the IT system to which they shouldn't
  - Send corporate data (e.g., customer lists, R&D, etc.) to third parties
Issues

• We develop a model for assessing the “guilt” of agents. We also present algorithms for distributing objects to agents, in a way that improves our chances of identifying a leaker.

• Finally, we also consider the option of adding “fake” objects to the distributed set.

• Such objects do not correspond to real entities but appear realistic to the agents.

• In a sense, the fake objects acts as a type of watermark for the entire set, without modifying any individual members. If it turns out an agent was given one or more fake objects that were leaked, then the distributor can be more confident that agent was guilty.
Scope

- There are conventional techniques being used and these include technical and fundamental analysis.

- The main issue with these techniques is that they are manual, and need laborious work along with experience.
Implementation

The system has the following:

- Data Allocation
- Fake Object
- Optimization
- Data Distributor
Data Allocation:

- The main focus of our project is the data allocation problem as how can the distributor “intelligently” give data to agents in order to improve the chances of detecting a guilty agent.
Fake Object :

Fake objects are objects generated by the distributor in order to increase the chances of detecting agents that leak data. The distributor may be able to add fake objects to the distributed data in order to improve his effectiveness in detecting guilty agents. Our use of fake objects is inspired by the use of “trace” records in mailing lists.
Optimization:

- The Optimization Module is the distributor’s data allocation to agents has one constraint and one objective. The distributor’s constraint is to satisfy agents’ requests, by providing them with the number of objects they request or with all available objects that satisfy their conditions. His objective is to be able to detect an agent who leaks any portion of his data.
Data Distributor:

- A data distributor has given sensitive data to a set of supposedly trusted agents (third parties). Some of the data is leaked and found in an unauthorized place (e.g., on the web or somebody’s laptop). The distributor must assess the likelihood that the leaked data came from one or more agents, as opposed to having been independently gathered by other means.
Proposed System

- In the Proposed System the hackers can be traced with good amount of evidence. In this proposed system the leakage of data is detected by the following methods viz., generating fake objects, Watermarking and by Encrypting the data.
Existing System

- The Existing System can detect the hackers but the total number of cookies (evidence) will be less and the organization may not be able to proceed legally for further proceedings due to lack of good amount of cookies and the chances to escape of hackers are high.
Modules

- Admin module
- User module
Admin Module

- Administrator has to logon to the system.
- Admin can add information about a new user.
- Admin can add/view/delete/edit the user details.
- Admin can create user groups and place users in it.
User Module

- A user must login to use the services.
- A user can send data sharing requests to other users.
- A user can accept/reject data sharing requests from other users.
- A user can trace the flow of its data i.e. can see what all users possess its data.
## Software Requirements

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Any Graphical OS</td>
</tr>
<tr>
<td>Technologies</td>
<td>Java (JSP, JDBC)</td>
</tr>
<tr>
<td>Back end</td>
<td>MS Access/SQL Server</td>
</tr>
<tr>
<td>Hardware Requirements</td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td></td>
</tr>
<tr>
<td>Processor</td>
<td></td>
</tr>
<tr>
<td>Hard Disk</td>
<td></td>
</tr>
<tr>
<td>Ram</td>
<td></td>
</tr>
<tr>
<td>Pentium – IV</td>
<td></td>
</tr>
<tr>
<td>40 GB</td>
<td></td>
</tr>
<tr>
<td>512 MB</td>
<td></td>
</tr>
</tbody>
</table>
Use case diagram for data leakage detection

- Admin
- User1
- User2
- Database

Activities:
- User Registration
- User Group Information
- Data Maintenance
- Set Request For Data Sharing
- Accept/Reject Data Sharing Requests
- Trace Data Sharing Flow
- Remove Data Sharing
Use case for Admin

Admin

User Registration

View User Group
Activity Diagram for Admin login

Admin login

Check

User Registration

View User Groups

Exit
Sequence Diagram for User Login

User Login

DLD

User Info

User login request

prompt for username and password

enter username and password

verify

return status (success/failure)

load modules on success

load error page on failure

verify

return status (success/failure)
Sequence Diagram for User Info

Admin

New user info request

DLD

Verify and fetch max(UID)
load max(UID)+1
enter user details verify and save
return status(success/failure)
view user info request verify and fetch record
load user list

User Info
Sequence Diagram for User Group

- User
  - set user group request
  - select users for the group
  - view usergroup request

- DLD
  - verify and fetch max(GID)
  - load max(GID)+1
  - verify and fetch user list
  - load users list

- User Info
  - verify and save
  - return status(success/failure)

- User Group
  - verify and fetch request
  - load user groups list
Sequence Diagram for Datatable Maintainance

- User
  - set database request
    - prompt for table and fields
      - enter tablename and fields
      - verify and save
        - verify and create table
          - return status (success/failure)
  - view usertables list
    - verify and fetch records
      - return status (success/failure)

- DLD

- Datatable

- Table
Sequence Diagram for Data Sharing Requests

User → DLD
- set request for data sharing
  - verify and fetch max(RID)
   - load max(RID)+1
     - verify and fetch user list
       - load user list
         - select user
           - verify and fetch Tnames
             - load Tnames
               - verify and fetch Tnames for request
                 - select Tnames for request
                   - verify and save
                     - return status (success/failure)
                       - view request status
                         - verify and fetch request list
                           - load status for requests made

User Info → Datatable → Sharing Request
Admin Login Form

![Login Form](image)
Add New User

User Id: 10007
Name:
Address:
Contacts:
Email:
Join Date: 31-Mar-2012
Designation: --Select--
UserName:
PassWord:
Group Id: --Select--

SUBMIT   CLOSE
Edit/Delete User Information

Select User Id: --Select--

User Id:
Name:
Address:
Contacts:
Email:
Join Date:
Designation: --Select--

UPDATE  DELETE  CLOSE
### View User Information

<table>
<thead>
<tr>
<th>User_Id</th>
<th>SName</th>
<th>Address</th>
<th>Contacts</th>
<th>EMail</th>
<th>DOR</th>
<th>Designation</th>
</tr>
</thead>
<tbody>
<tr>
<td>10001</td>
<td>Sachin Tyagi</td>
<td>Secunderabad</td>
<td>1234567890</td>
<td><a href="mailto:sachin@gmail.com">sachin@gmail.com</a></td>
<td>01-Mar-2012</td>
<td>Clerk</td>
</tr>
<tr>
<td>10002</td>
<td>Arun Kumar</td>
<td>Secunderabad</td>
<td>33333666666</td>
<td><a href="mailto:arun@gmail.com">arun@gmail.com</a></td>
<td>15-Mar-2012</td>
<td>Cashier</td>
</tr>
<tr>
<td>10003</td>
<td>Tina Zakaria</td>
<td>Hyderabad</td>
<td>66666555555</td>
<td><a href="mailto:tina@yahoo.com">tina@yahoo.com</a></td>
<td>15-Mar-2012</td>
<td>Operator</td>
</tr>
<tr>
<td>10004</td>
<td>Mahesh</td>
<td>Secunderabad</td>
<td>44444666666</td>
<td><a href="mailto:mahesh@gmail.com">mahesh@gmail.com</a></td>
<td>15-Mar-2012</td>
<td>Manager</td>
</tr>
<tr>
<td>10005</td>
<td>Rahul</td>
<td>Secunderabad</td>
<td>9876543210</td>
<td><a href="mailto:rahul@yahoo.com">rahul@yahoo.com</a></td>
<td>15-Mar-2012</td>
<td>Asst. Manager</td>
</tr>
<tr>
<td>10006</td>
<td>ankit</td>
<td>hyd</td>
<td>9441835872</td>
<td><a href="mailto:abcd@gmail.com">abcd@gmail.com</a></td>
<td>29-Mar-2012</td>
<td>Manager</td>
</tr>
</tbody>
</table>
Set Groups

Select Desgination: --Select--
Select Desgination: --Select--
Select Desgination: --Select--

SUBMIT  CLOSE
View Groups

<table>
<thead>
<tr>
<th>Group_Id</th>
<th>Members</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Manager, Asst. Manager, Clerk</td>
</tr>
<tr>
<td>2</td>
<td>Manager, Clerk, Cashier</td>
</tr>
<tr>
<td>3</td>
<td>Asst. Manager, Cashier, Operator</td>
</tr>
<tr>
<td>4</td>
<td>Manager, Asst. Manager, Cashier</td>
</tr>
</tbody>
</table>
User Login
Request for Data Sharing

```
Request Id : 6
Designation
Staff Id : --Select--
Table Name : --Select--
Request Date : 31-Mar-2012

SUBMIT   CLOSE
```
View Data Sharing Requests

<table>
<thead>
<tr>
<th>Request_Id</th>
<th>Staff_Id</th>
<th>Job</th>
<th>Table_Name</th>
<th>Req_Date</th>
<th>Group_Id</th>
</tr>
</thead>
</table>

Request Id: --Select--

Table Name: 

Set Status: --Select--

Submit    Close
## View Sharing Request Status

<table>
<thead>
<tr>
<th>Request_Id</th>
<th>Staff_Id1</th>
<th>Table_Name</th>
<th>Req_Date</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>10004</td>
<td>M_Table</td>
<td>29-Mar-2012</td>
<td>Accepted</td>
</tr>
</tbody>
</table>
## Trace Shared Data

### Shared Path

<table>
<thead>
<tr>
<th>Request_Id</th>
<th>Staff_Id</th>
<th>Job</th>
<th>Staff_Id1</th>
<th>Job1</th>
<th>Req_Date</th>
<th>Group_Id</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10002</td>
<td>Cashier</td>
<td>10001</td>
<td>Clerk</td>
<td>29-Mar-20...</td>
<td>2</td>
<td>Accepted</td>
</tr>
<tr>
<td>3</td>
<td>10001</td>
<td>Clerk</td>
<td>10004</td>
<td>Manager</td>
<td>29-Mar-20...</td>
<td>2</td>
<td>Accepted</td>
</tr>
<tr>
<td>5</td>
<td>10006</td>
<td>Clerk</td>
<td>10007</td>
<td>Asst. Man...</td>
<td>29-Mar-20...</td>
<td>1</td>
<td>Accepted</td>
</tr>
</tbody>
</table>

### Intruder List

<table>
<thead>
<tr>
<th>Request_Id</th>
<th>Staff_Id</th>
<th>Job</th>
<th>Staff_Id1</th>
<th>Job1</th>
<th>Req_Date</th>
<th>Group_Id</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>10006</td>
<td>Clerk</td>
<td>10007</td>
<td>Asst. Man...</td>
<td>29-Mar-20...</td>
<td>1</td>
<td>Accepted</td>
</tr>
</tbody>
</table>

CLOSE
# Test Cases

<table>
<thead>
<tr>
<th>Test Case ID:</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose:</td>
<td>To login as admin.</td>
</tr>
<tr>
<td>Input:</td>
<td>Input the admin’s username and password</td>
</tr>
<tr>
<td>Expected Result:</td>
<td>User should log in as admin.</td>
</tr>
<tr>
<td>Actual Result:</td>
<td>The user has got Administrative privileges by successfully logging in as admin.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test Case ID:</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose:</td>
<td>To create a user</td>
</tr>
<tr>
<td>Input:</td>
<td>Input a username and his details by logging in as admin.</td>
</tr>
<tr>
<td>Expected Result:</td>
<td>When submit button is pressed, new user should be created.</td>
</tr>
<tr>
<td>Actual Result:</td>
<td>A new user is created in the database.</td>
</tr>
<tr>
<td>Test Case ID:</td>
<td>3</td>
</tr>
<tr>
<td>-------------</td>
<td>---</td>
</tr>
<tr>
<td>Purpose:</td>
<td>To allow sharing of data.</td>
</tr>
<tr>
<td>Input:</td>
<td>Request a user of the same group for the data.</td>
</tr>
<tr>
<td>Expected Result:</td>
<td>If the user accepts the request then the data is shared and the owner of the data can see the flow of the data.</td>
</tr>
<tr>
<td>Actual Result:</td>
<td>After accepting the request the data is shared and the owner of the data can trace the flow of data.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test Case ID:</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose:</td>
<td>To trace data leakage</td>
</tr>
<tr>
<td>Input:</td>
<td>Request a user of different group for unauthorized data sharing.</td>
</tr>
<tr>
<td>Expected Result:</td>
<td>Upon acceptance of this unauthorized request, the owner of the data can find the agent guilty of data leakage.</td>
</tr>
<tr>
<td>Actual Result:</td>
<td>The owner of the data can see the guilty agent by tracing the flow of data.</td>
</tr>
</tbody>
</table>
Conclusions

- We have shown that it is possible to assess the likelihood that an agent is responsible for a leak. Our model is relatively simple, but we believe that it captures essential trade-offs.

- Our future work will be to investigate other complex data leakage scenarios such as appropriate model for cases where agents can collude and identify fake tuples.
Thank you!