Available at: www.researchpublications.org

DATA LEAKAGE DETECTION

Mr. Amol O. Gharpande
Comp. Sci & Engg Dept.
PRMIT&R Badnera
amol_gharpande@hotmail.com

Prof. Ms. V. M. Deshmukh
Computer. Sci & Engg Dept.
PRMIT&R Badnera

Abstract- In both the commercial and defense sectors a compelling need is emerging for rapid, yet secure, dissemination of Information. This paper gives review idea about data leakage detection techniques. 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.

Keywords: Distributor, agent, guilty agent.

1. INTRODUCTION

In the course of doing business, sometimes sensitive data must be handed over to supposedly trusted third parties. For example, a hospital may give patient records to researchers who will devise new treatments. Similarly, a company may have partnerships with other companies that require sharing customer data. Another enterprise may outsource its data processing, so data must be given to various other companies. We call the owner of the data the distributor and the supposedly trusted third parties the agents. Our goal is to detect when the distributor's sensitive data have been leaked by agents, and if possible to identify the agent that leaked the data.

We consider applications where the original sensitive data cannot be perturbed. Perturbation is a very useful technique where the data are modified and made "less sensitive" before being handed to agents. For example, one can add random noise to certain attributes, or one can replace exact values by ranges .However, in some cases, it is important not to alter the original distributor's data. For example, if an outsourcer is doing our payroll, he must have the exact salary and customer bank account numbers. If medical researchers will be treating patients , they may need accurate data for the patients. Traditionally, leakage detection is handled by water marking,e.g., a unique code is embedded in each distributed copy. If that copy is later discovered in the hands of an unauthorized party, the leaker can be identified. Watermarks can be very useful in some cases, but again, involve some modification of the original data. Furthermore, watermarks can sometimes be destroyed if the data recipient is malicious. In this paper, we study unobtrusive techniques for detecting leakage of a set of objects or records. Specifically, we study the following scenario: After giving a set of objects to agents, the distributor discovers some of those same objects in an unauthorized place. (For example, the data may be found on a website, or may be obtained through a legal discovery process.) At this point, the distributor can assess the likelihood that the leaked data came from one or more agents, as opposed to having been independently gathered by other means. Using an analogy with cookies stolen from a cookie jar, if we catch Freddie with a single cookie, he can argue that a friend gave him the cookie. But if we catch Freddie with five cookies, it will be much harder for him to argue that his hands were not in the cookie jar. If the distributor sees "enough evidence" that an agent leaked data, he may stop doing business with him, or may initiate legal proceedings.

2. LITERATURE REVIEW

The guilt detection approach we present is related to the data provenance problem tracing the lineage of S objects implies essentially the detection of the guilty agents. It provides a good overview on the research conducted in this field. Suggested solutions are domain specific, such as lineage tracing for data warehouses, and assume some prior knowledge on the way a data view is created out of data sources. Our problem formulation with objects and sets is more general and simplifies lineage tracing, since we do not consider any data transformation from Ri sets to S. As far as the data allocation strategies are concerned, our work is mostly relevant to watermarking that is used as a means of establishing original ownership of distributed objects. Watermarks were initially used in images, video, and audio data whose digital representation includes considerable redundancy, and other works have also studied marks insertion to relational data. Our approach and watermarking are similar in the sense of providing agents with some kind of receiver identifying information. However, by its very nature, a watermark modifies the item being watermarked. If the object to be watermarked cannot be modified, then a watermark cannot be inserted. In such cases, methods that attach watermarks to the distributed data are not applicable. Finally, there are also lots of other works on mechanisms that allow only authorized users to access sensitive data through access control policies. Such approaches prevent in some sense data leakage by sharing information only with trusted parties. However, these policies are restrictive and may make it impossible to satisfy agent's requests.

2.1 USE OF FAKE OBJECTS

The distributor may be able to add fake objects to the distributed data in order to improve his effectiveness in detecting guilty agents. However, fake objects may impact the correctness of what agents do, so they may not always be allowable. The idea of perturbing data to detect leakage is not new. However, in most cases, individual objects are perturbed, e.g., by adding random noise to sensitive salaries, or adding a watermark to an image. In our case, we are perturbing the set of distributor objects by adding fake elements. In some applications, fake objects may cause fewer problems that perturbing real objects. For example, say that the distributed data objects are medical records and the agents are hospitals. In this case, even small modifications to the records of actual patients may be undesirable. However, the addition of some fake medical records may be acceptable, since no patient matches these records, and hence, no one will ever be treated based on fake records. Our use of fake objects is inspired by the use of "trace" records in mailing lists. In this case, company A sells to company B a mailing list to be used once (e.g., to send advertisements). Company A adds trace records that contain addresses owned by company A. Thus, each time company B uses the purchased mailing list, A receives copies of the mailing. These records are a type of fake objects that help identify improper use of data.

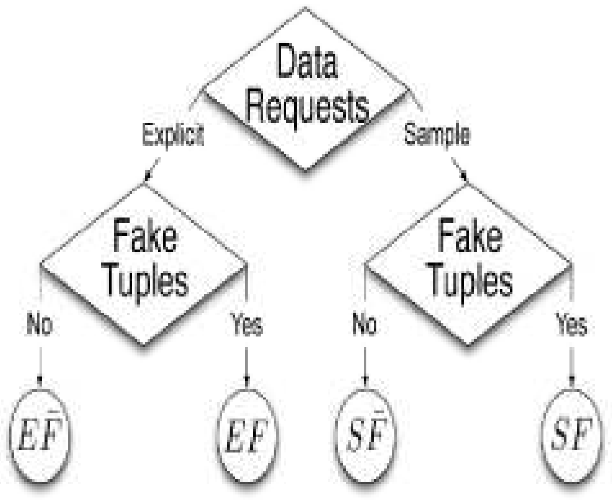


Fig.1: Leakage problem instances.

The Fig.1 represents four problem instances with the names *EF, E*F& *, SF* an *S*F& , where *E* stands for explicit requests, *S* for sample requests, *F* for the use of fake objects, and F & for the case where fake objects are not allowed. The distributor may be able to add fake objects to the distributed data in order to improve his effectiveness in detecting guilty agents. Since, fake objects may impact the correctness of what agents do, so they may not always be allowable. Use of fake objects is inspired by the use of "trace" records in mailing lists. The distributor creates and adds fake objects to the data that he distributes to agents. In many cases, the distributor may be limited in how many fake objects he can create.

3. ANALYSIS OF PROBLEM

1) We consider applications where the original sensitive data cannot be perturbed. Perturbation is a very useful technique where the data is modified and made "less sensitive" before being handed to agents.
2) However, in some cases it is important not to alter the original distributor's data.
3) Traditionally, leakage detection is handled by watermarking, e.g., a unique code is embedded in each distributed copy.
4) If that copy is later discovered in the hands of an unauthorized party, the leaker can be identified.
5) Watermarks can be very useful in some cases, but again, involve some modification of the original data.
6) Furthermore, watermarks can sometimes be destroyed if the data recipient is malicious.

4. PROPOSED WORK

1) After giving a set of objects to agents, the distributor discovers some of those same objects in an unauthorized place.

2) At this point the distributor can assess the likelihood that the leaked data came from one or more agents, as opposed to having been independently gathered by other means.
3) If the distributor sees "enough evidence" that an agent leaked data, he may stop doing business with him, or may initiate legal proceedings.
4) In this project we develop a model for assessing the "guilt" of agents.
5) We also present algorithms for distributing objects to agents, in a way that improves our chances of identifying a leaker.
6) Finally, we also consider the option of adding "fake" objects to the distributed set. Such objects do not correspond to real entities but appear.
7) If it turns out an agent was given one or more fake objects that were leaked, hen the distributor can be more confident that agent was guilty.

5. MODULE DESCRIPTION

5.1 Distributer

A) Login / Registration

This is a module mainly designed to provide the authority to a user in order to access the other modules of the project. Here a user can have the accessibility authority after the registration.

B) Data Transfer

This module is mainly designed to transfer data from distributor to agents. The same module can also be used for illegal data transfer from authorized to agents to other agents.

5.2AGENT

A) Guilt Model Analysis:

This module is designed using the agent – guilt model. Here a count value (also called as fake objects) is incremented for any transfer of data occurrence when agent transfers data. Fake objects are stored in database.

B) Agent-Guilt Model:

This module is mainly designed for determining fake agents. This module uses fake objects (which is stored in database from guilt model module) and determines the guilt agent along with the probability.

6. FILE TRANSFER PROTOCOL

File Transfer Protocol (FTP) is one of the oldest applications in use on the Internet. First proposed in April of 1971, it predates TCP/IP, the pair of protocols FTP needs in order to operate. File Transfer Protocol is designed to do exactly that, transfer files between a server and a client. There are many applications which use FTP to transfer files between computers. To use FTP, you can start up the FTP program on your machine and connect to a server. The FTP application you use to connect to the server is a client, and your client software can connect to an FTP server running on port 21 on a remote machine. FTP is included on most of today's operating systems.

7. HOW FTP WORKS

1) FTP creates both a control and a data connection in order to transfer files. The control connection is based on telnet and is used to negotiate the parameters for the data transfer. This is called inactive FTP connection.
2) The client FTP application opens a control connection to the server on destination port 21, and specifies a source port as the source to which the FTP server should respond (using TCP).
3) The FTP server responds on port 21.
4) The FTP server and client negotiate the data transfer parameters.
5) The FTP server opens a second connection for data on port 20 to the original client.
6) The client responds on the data port, completing a TCP connection. Data transfer begins.
7) The server indicates the end of the data transfer
8) Client closes the connection once the data is received.
9) The data connection is closed.
10) The FTP connection is closed.

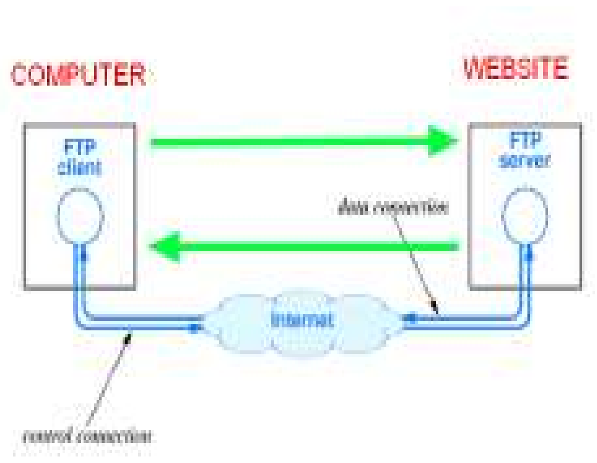
8. INITIATING FILE TRANSFERS FROM THE COMMAND LINE

1) On the command line, enter FTP <server name>.
2) Enter your login information if prompted.
3) Set your transfer mode to either 'ascii' or 'binary' depending upon the type of file you are transferring.
4) You can discover what directory you have connected to by entering the command 'pwd'.
5) To change directories on the remote machine, enter 'cd' and the name of the directory.
6) To change directories locally, enter 'lcd' To put a file on the remote machine, enter PUT and the name of the file.
7) Once the transfer completes, you cen enter 'close' and then 'quit' ('!' and 'bye' also serve the same function as quit).

9. PASSIVE MODE

The passive mode behavior of an FTP server:

1) The client opens a connection to the server on TCP port 21 (command channel)
2) The server accepts the connection.
3) The server initiates a connection to the client using port 20 as the source port (for the data channel)
4) The client accepts the connection and acknowledges all data transfers on port.



10. RESULTS AND DISCUSSION

1) A data distributor has given sensitive data to a set of supposedly trusted agents (third parties).
2) Some of the data is leaked and found in an unauthorized place (e.g., on the web or somebody's laptop).
3) 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.
4) We propose data allocation strategies (across the agents) that improve the probability of identifying leakages.
5) 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.
6) 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.

11. CONCLUSION

In a perfect world, there would be no need to hand over sensitive data to agents that may unknowingly or maliciously leak it. And even if we had to hand over sensitive data, in a perfect world, we could watermark each object so that we could trace its origins with absolute certainty. However, in many cases, we must indeed work with agents that may not be 100 percent trusted, and we may not be certain if a leaked object came from an agent or from some other source since certain data cannot admit watermarks. On review it is observed that there are various data leakage detection techniques are available.

REFERENCES.

[1] R. Agrawal and J. Kiernan. Watermarking relational databases. In VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases, pages 155–166. VLDB Endowment, 2002.

[2] P. Bonatti, S. D. C. di Vimercati, and P. Samarati. An algebra for composing access control policies. ACM Trans. Inf. Syst. Secur., 5(1):1–35, 2002.

[3] P. Buneman, S. Khanna, and W. C. Tan. Why and where: A characterization of data provenance. In J. V. den Bussche and V. Vianu, editors, Database Theory - ICDT 2001, 8th International Conference, London, UK, January 4-6, 2001, Proceedings, volume 1973 of Lecture Notes in Computer Science, pages 316–330. Springer, 2001.

[4] P. Buneman and W.-C. Tan.Provenance in databases. In SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data, pages 1171–1173, New York, NY, USA, 2007. ACM.