International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.3, Issue.1, Jan-Feb. 2013 pp-538-540 ISSN: 2249-6645

A Novel Data Leakage Detection

Priyanka Barge, [1] Pratibha Dhawale, [2] Namrata Kolashetti[3]
Ass. Prof., Department of Computer Engineering, NIRMALA CHOUHAN

Abstract: This paper contains concept of data leakage, its causes of leakage and different techniques to detect the data leakage. The value of the data is incredible, so it should not be leaked or altered. In the field of IT, huge database is being used. This database is shared with multiple people at a time. But during this sharing of the data, there are huge chances of data vulnerability, leakage or alteration. So, to prevent these problems, a data leakage detection system has been proposed. This paper includes brief idea about data leakage detection and a methodology to detect the data leakage persons.

Keywords: *Guilty agent, data distributor, watermarking, fake object, data leakage.*

I. Introduction

Data leakage is defined as the accidental or unintentional distribution of private or sensitive data to unauthorized entity. Sensitive data of companies and organizations includes intellectual property (IP), financial information, patient information, personal credit-card data, and other information depending on the business and the industry.

Furthermore, in many cases, sensitive data is shared among various stakeholders such as employees working from outside the organizational premises (e.g., on laptops), business partners and customers. This increases the risk of confidential information falling into unauthorized hands. Whether caused by malicious intent, or an inadvertent mistake, by an insider or outsider, exposed sensitive information can seriously hurt an organization.

The potential damage and adverse consequences of a data leak incident can be classified into the following two categories: direct and indirect loss[3]. Direct loss refers to tangible damage that is easy to measure and estimate quantitatively. Indirect loss, on the other hand, is much harder to quantify and has a much broader impact in terms of cost, place and time. Direct loss includes violating regulations (such as those protecting customer privacy) resulting in fine/settlement/customer compensation fees; litigation of lawsuits; loss of future sales; costs of investigation and remedial/restoration fees. Indirect loss includes reduced share-price as a result of the negative publicity; damage to company's goodwill and reputation; customer abandonment; and exposure of Intellectual Property (business plans, code, financial reports, and meeting agendas) to competitors.

II. Existing System

Traditionally, leakage detection is handled by watermarking, 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. E.g. 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.

III. Proposed System

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. Perturbation is a very useful technique where the data is modified and made "less sensitive" before being handed to agents. We propose to develop unobtrusive techniques for detecting leakage of a set of objects or records.

In this section, we propose to 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 [2]. 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.

IV. Methodology

4.1 *Entities And Agents*

Distributor owns set of data objects $T= \{t_1,t_2,\ldots,t_n\}$. Distributor has to share some of the objects with set of agents U_1, $U_2 \ldots U_n$, but does not wish the object be leaked to other third parties. The objects in T could be of any type and size, e.g., they could be tuples in a relation or relations in a database.

An agent U_i receives a subset R_i of objects T, determined either by a sample request or an explicit request.

- **Evaluation of Explicit Data Request Algorithms**

In this request the agent will send the request with appropriate condition. Agent gives the input as request with input as well as the condition for the request. After processing on the data he will get the new data by adding fake object using watermarking technique.

Explicit request R_i= EXPLICIT(T, $cond_i$): Agent U_i receives all T objects that satisfy $cond_i$.

- **Evaluation of Sample Data Request Algorithms**

In this request agent request does not have condition. The agent sends the request without condition as per his query he will get the data by adding fake object using watermarking technique.

Sample request= SAMPLE(T, m_i) : Any subset of m_i records from T can be given to U_i.

4.2 Guilty Agents

Suppose that after giving objects to agents, the distributor discovers that a set S belongs to T has leaked. This means that the third party, called the target, has been caught in possession of S.

We say an agent U_i is guilty and if it contributes one or more object to target. We denote the event that agent U_i is guilty as G_i and the event that agent U_i is guilty for a given leaked set S as $G_i|S$. Our next step is to estimate $Pr\{G_i|S\}$, i.e. the probability that agent U_i is guilty given evidence S.

4.3 Related Work

Here we proposed a watermarking algorithm that embeds the watermark bits in the least significant bits (LSB) of selected attributes of a selected subset of tuples. This technique does not provide mechanism for multibit watermarks, insead only secret key is used. For each tuple, a secure message authenticated code (MAC) is computed using the secret key and tuple's primary key. The computed MAC is used to select candidate tuples, attributes and LSB position in the selected attributes. However, the watermark can be easily compromised by very trivial attacks. For example a simple manipulation of the data by shifting the LSB is one position easily leads to watermark loss without much damage to the data. Therefore LSB based data hiding technique is not resilient. Moreover, it assumes that the LSB bits in any tuple can be altered without checking data constraints. Simple unconstrained LSB manipulations can easily generate undesirable results. Thus such technique is not resilient to deletion and insertion attacks.

V. User Model

To compute this Pr*{Gi |S}*, we need an estimate for the probability that values in *S* can be "guessed" by the target We call this estimate *pt*, the probability that object t can be guessed by the target.

Probability Pt is analogous to the probabilities used in designing fault-tolerant systems. That is, to estimate how likely it is that a system will be operational throughout a given period, we need the probabilities that individual components will or will not fail. A component failure in our case is the event that the target guesses an object of *S*. The component failure is used to compute the overall system reliability, while we use the probability of guessing to identify agents that have leaked information. The component failure probabilities are estimated based on experiments, just as we propose to estimate the Pt's. Similarly, the component probabilities are usually conservative estimates, rather than exact numbers. For example, say we use a component failure probability that is higher than the actual probability, and we design our system to provide a desired high level of reliability. Then we will know that the actual system will have at least that level of reliability, but possibly higher. In the same way, if we use Pt's that are higher than the true values, we will know that the agents will be guilty with at least the computed probabilities.

VI. Data Allocation Problem

The main focus of our paper is the data allocation problem: how can the distributor "intelligently" give data to agents in order to improve the chances of detecting a guilty agent?

6.1 *Fake Objects*

The idea of perturbing data to detect leakage is not new. 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 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.

The distributor creates and adds fake objects to the data that he distributes to agents. We let $Fi \subseteq Ri$ be the subset of fake objects that agent Ui receives. Fake objects must be created carefully so that agents cannot distinguish them from real objects.

6.2 Optimization Problem

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.

VII. 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% 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.

In spite of these difficulties, we have shown it is possible to assess the likelihood that an agent is responsible for a leak, based on the overlap of his data with the leaked data and the data of other agents, and based on the probability that objects can be "guessed" by other means. Our model is relatively simple, but we believe it captures the essential trade-offs. The algorithms we have presented implement a variety of data distribution strategies that can improve the distributor's chances of identifying a leaker. We have shown that distributing objects judiciously can make a significant difference in identifying guilty agents, especially in cases where there is large overlap in the data that agents must receive.

7.1 *Advantages*

- It is possible to assess the likelihood that an agent is responsible for a leak, based on the overlap of his data with the leaked data and the data of other agents and based on the probability that objects can be guessed by other means [1].

Disadvantages

- In given watermarking algorithm, watermark can be easily compromised by very trivial attacks. For example a simple manipulation of the data by shifting the LSB is one position easily leads to watermark loss without much damage to the data. Therefore LSB based data hiding technique is not resilient.

VIII. Acknowledge

For all the efforts behind this paper work, we first would like to express our sincere thanks to the staff of Dept. of computer Engg., for their extended help and suggestions at every stage. It is with a great sense of gratitude that I acknowledge the support, time to time suggestions to my guide prof. Nirmala Chouhan, Prof. G. M. Bhandari(HOD) and prof. Jayant Jadhav (Project Co-ordinator).

References

[1] P. Papadimitriou and H. Garcia-Molina, "Data leakage detection," IEEE Transactions on Knowledge and Data Engineering, pages 51-63, volume 23, 2011.
[2] R. Agrawal and J. Kiernan, "Watermarking Relational Databases," Proc. 28th Int'l Conf. Very Large Data Bases (VLDB '02), VLDB Endowment, pp. 155-166, 2002.
[3] Sandip A. Kale, Prof. S.V. Kulkarni/ IOSR Journal of Computer Engineering (IOSRJCE) ISSN:2278-0661 Volume 1, Issue 6 (July-Aug 2012), PP 32-35 **www.iosrjournals.org** / page Data Leakage Detection : A Survey