Data Leakage Detection in an organisation

Janhavi Sawant, Harshad Kadam, Shweta Menghani

Abstract-We study the following problem: A data distributor has given sensitive data to a set of supposedly trusted agents (third parties). Some of the data are 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.

Index Terms-Allocation strategies, data leakage, data privacy, fake records, leakage model.

1 INTRODUCTION

In today' technically empowered data rich environment, it is a major challenge for data holders to prevent data leakage. Loss of large volumes of protected information has become regular headline event, forcing companies to re-issue cards, notify customers and mitigate loss of goodwill from negative publicity.

While great deal of attention has been given to protecting companies electronic assets from outsiders threats – from intrusion prevention systems to firewalls to vulnerability management – organizations now turn their attention to an equally dangerous situation: the problem of data loss from the inside. Whether its email, instant messaging, webmail, a form of website, or a file transfer, electronic communications exiting the company still go largely uncontrolled and unmonitored on their way to their destinations – with the ever present potential for confidential information to fall into wrong hands. Should sensitive information be exposed, it can wreak havoc on the organization's bottom line through fines, bad publicity, loss of strategic customers, loss of competitive intelligence and legal actions.

Consider the example where a former employee of one company accidentally post IDs and bank accounts data for 150 employees of an advertising firm on a website. The list goes on and on.

There is major solution given is "watermarking" technique, where the unique code is embedded within the data. But it is not useful with sensitive information as it changes some of bits in data. Also if the recipient is malicious it, may destroy the watermark.

Also access control mechanism can be used, that allow only authorized users to access the sensitive data through an access control policies. But it also put restrictions on users and our aim is to provide service to all customers (you cannot deny the coming request).

In this paper, we proposed one model that can handle all the requests from customers and there is no limit on number of customers. The model gives the data allocation strategies featured with the forged objects injection proposed by Ref [6] to improve the probability of identifying leakages, but they can accept request from only some number of customers.

Also we study the application where there is a distributor, distributing and managing the files that contain sensitive information to users when they send request. The log is maintained for every request, which is later used to find overlapping with the leaked file set and the subjective risk assessment of guilt probability.

2 RELATED WORK

The data leakage prevention based on the trustworthiness [1] is used to assess the trustiness of the customer. Maintaining the log of all customer's request is related to the data provenance problem [2] i.e. tracing the lineage of objects. The data allocation strategy used is more relevant to the watermarking [3],[4] that is used as a means of establishing original ownership of distributed objects.

There are also different mechanisms to allow only authorized users to access the sensitive information [5] through access control policies, but these are restrictive and may make it impossible to satisfy agent's requests.

3 PROPOSED WORK

3.1 Problem Definition:

The distributor owns the sensitive data set T= { t_1, t_2, ……., t_n}. The agent A_i request the data objects from distributor. 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. The distributor

gives the subset of data to each agent. After giving objects to agents, the distributor discovers that a set L of T has leaked. This means some third party has been caught in possession of L.

The agent Ai receives a subset R_i of objects T determined either by implicit request or an explicit request.

- Implicit Request R_i = Implicit(T, m_i) : Any subset of m_i records from T can be given to agent A_i
- Explicit Request R_i = Explicit(T, $Cond_i$) : Agent A_i receives all T objects that satisfy $Cond_i$

3.2 Guilt Assessment:

Let L denote the leaked data set that may be leaked intentionally or guessed by the target user.

Since agent having some of the leaked data of L, may be susceptible for leaking the data. But he may argue that he is innocent and that the L data were obtained by target through some other means.

Our goal is to assess the likelihood that the leaked data came from the agents as opposed to other resources.

e.g. if one of the object of L was given to only agent A_1, we may suspect A_1 more. So probability that agent A_1 is guilty for leaking data set L is denoted as Pr {G_i | L}.

3.3 Guilt Probability Computation:

For the sake of simplicity our model relies on two assumptions:

Assumption 1: For all $t_1, t_2, \ldots\ldots, t_n \in L$ and $t_1 \neq t_2$, the provenance of $t_1$is independent of t_2.

Assumption 2: Tuple $t \in L$ can only be obtained by third user in one of the two ways:

1. Single user A_1 leaked t or
2. Third user guessed t with the help of other resources.

Now to compute the guilt probability that he leaks a single object t to L, we define a set of users.

To find the probability that an agent A_i is guilty for the given set L, consider the target guessed t_1 with probability p and that agent leaks t_1 to L with probability 1-p. First compute the probability that he leaks a single object to L. To compute this, define the set of agents U_t = { A_i | $t \in R_i$ } that have t in their data sets. Then using Assumption 2 and known probability p,we have,

Pr{Some agent leaked t to L=1-p----------**(1)**

Assuming that all agents that belongs to U_t can leak t to L with equal probability and using Assumption 2 we get,

$Pr\{A_i \text{ leaked t to L}\} = \frac{1}{|U_t|}$ if $A_i \in U_t$ ----**(2)**

Given that user A_i is guilty if he leaks at least one value to L, with assumption 1 and equation 2, we can compute the probability Pr {G_i | L} that user A_i is guilty :

$$Pr\{G_i \mid L\} = 1 - \prod_{t \in L \cap R_i} \left(\frac{|U_t| - |U_t \cap R_i|}{|U_t|}\right) \quad \text{-----(3)}$$

3.4 Data Allocation Strategies:

The distributor gives the data to agents such that he can easily detect the guilty agent in case of leakage of data. To improve the chances of detecting guilty agent, he injects fake objects into the distributed dataset. These fake objects are created in such a manner that, agent cannot distinguish it from original objects. One can maintain the separate dataset of fake objects or can create it on demand. In this paper we have used the dataset of fake tuples.

Depending upon the addition of fake tuples into the agent's request, data allocation problem is divided into four cases as:

i. Explicit request with fake tuples
ii. Explicit request without fake tuples
iii. Implicit request with fake tuples
iv. Implicit request without fake tuples.

For example, distributor sends the tuples to agents A_1 and A_2 as

R_1= {t_1, t_2} and R_2= { t_1}. If the leaked dataset is L={ t_1}, then agent A_2 appears more guilty than A_1. So to minimize the overlap, we insert the fake objects in to one of the agent's dataset.

3.5 Overlap Minimization:

The distributor's data allocation to agents one constraint and one objective. The distributor's constraint is to satisfy agent's request, 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.

We consider his constraint as strict. The distributor may not deny serving an agent request and may not provide agents different perturbed versions of the same object.

The objective is to maximize the chances of detecting guilty agent that leaks all his data objects.

The Pr {G_i | L=Ri } is the probability that agent A_i is guilty if distributor discovers a leaked table L that contains all Ri objects.

The difference function Δ (i, j) is defined as

$$\Delta(i,j) = Pr\{G_i \mid R_i\} - Pr\{G_i \mid R_j\} \quad i,j = 1, \ldots\ldots n$$

3.5.1 Problem definition: Let the distributor have data request from n agents. The distributor wants to give tables R_1 ,R_2…… R_n to agents A_1 ,A_2…… A_n respectively, so that

- Distribution satisfies agent's request; and
- Maximizes the guilt probability differences Δ (i, j) for all i, j= 1, 2, ……n and i≠j.

Optimization Problem:

Maximizing the difference among distributed dataset increases the minimization of overlap.

i.e $\max_{over\ R_1,\ldots\ldots,R_n} (\ldots\ldots, \Delta(i,j), \ldots\ldots)$

Then

$\min \ldots\ldots \left|\frac{\ldots\ldots}{\ldots\ldots}\right| \quad i \neq j$

4. EXPERIMENTAL SETUP

In this paper, we presented the algorithm and the corresponding results for the explicit data allocation with the addition of fake tuples. We are still working on minimizing the overlap in case of implicit request.

Whenever any user request for the tuple, it follows the following steps:

1. The request is sent by the user to the distributor.
2. The request may be implicit or explicit.
3. If it is implicit a subset of the data is given.
4. If request is explicit, it is checked with the log, if any previous request is same .
5. If request is same then system gives the data objects that are not given to previous agent.
6. The fake objects are added to agent's request set.
7. Leaked data set L, obtained by distributor is given as an input.
8. Calculate the guilt probability G_i of user using II.

In the case where we get similar guilt probabilities of the agents, we consider the trust value of agent. These trust values are calculated from the historical behavior of agents. The calculation of trust value is not given here, we just assumed it. The agent having low trust value is considered as guilty agent. The algorithm for allocation of dataset on agent's explicit request is given below.

4.1 Algorithm 1:

Allocation of Data Explicitly:

Input:- i. $T=\{t_1,t_2,t_3,\ldots\ldots t_n\}$
-Distributor's Dataset
ii. R- Request of the agent
iii. Cond- Condition given by the agent
iv. m= number of tuples given to an agent m<n, selected randomly

Output:- D- Data sent to agent

1. $D=\Phi$, $T'=\Phi$
2. For i=1 to n do
3. If(t_i.fields==cond) then
4. T'=T'U{ t_i}
5. For i=0 to i<m do
6. D=DU{ t_i}
7. T'=T'-{ t_i}
8. If $T'=\Phi$ then
9. Goto step 2
10. Allocate dataset D to particular agent
11. Repeat the steps for every agent

To improve the chances of finding guilty agent we can also add the fake tuples to their data sets. Here we maintained the table for duplicate tuples and add randomly these tuples to the agent's dataset.

ALGORITHM2:

4.2 ADDITION OF FAKE TUPLES:

Input: i. D- Dataset of agent
ii. F- Set of fake tuples
iii. Cond- Condition given by agent
iv. b- number of fake objects to be sent

Output:- D- Dataset with fake tuples

1. While b>0 do
2. f= select Fake Object at random from set F
3. D= DU {f}
4. F= F-{f}
5. b=b-1
6. if $F=\Phi$ then reinitialize the fake data set.

Similarly, we can distribute the dataset for implicit request of agent. For implicit request the subset of distributor's dataset is selected randomly. Thus with the implicit data request we get different subsets. Hence there are different data allocations. An object allocation that satisfies requests and ignores the distributor's objective to give each agent unique subset of T of size m. The s-max algorithm allocates to an agent the data record that yields the minimum increase of the maximum relative overlap among any pair of agents. The s-max algorithm is as follows:

1. Initialize Min_Overlap, the minimum out of the minimum relative overlaps that the allocations of different objects to A_i
2. for k do Initialize max_rel_ov←0, the maximum relative overlap between R_i the allocation of t_k to A_i
3. for all j=1,……,n:j=I and $t_k \in R_j$ do calculate absolute overlap as abs_ov←

calculate relative overlap as
rel_ov←abs_ov/min(m_i, m_j)

4. Find maximum relative overlap as
Max_rel_ov←MAX(max_rel_ov, rel_ov)
If max_rel_ov≤ min_ov then
Min_ov←max_rel_ov
ret_k←k
Return ret_k

The algorithm presented implements a variety of data distribution strategies that can improve the distributor's chances of identifying a leaker. It is 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.

5. EXPERIMENTAL RESULTS

In our scenarios we have taken a set of 500 objects and requests from every agent are accepted. There is no limit on number of agents, as we are considering here their trust values. The flow of our system is given as below:

1. Agent's Request: Either Explicit or Implicit

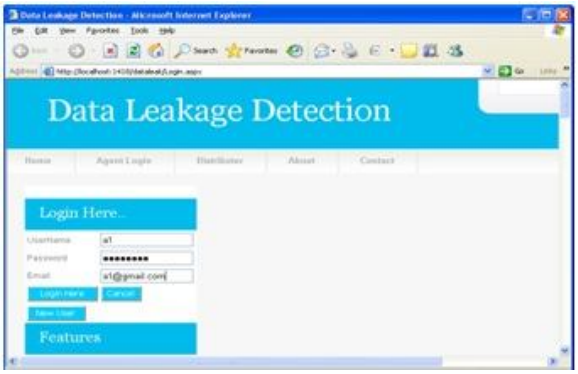

Fig1. Agent's Request

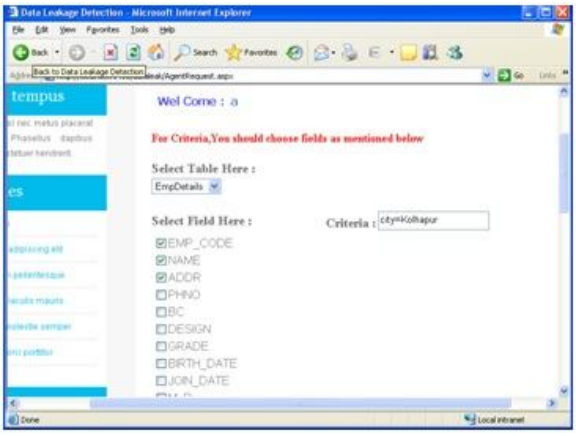

Fig2. Agent Selects the required fields

1. Distributor sends the tuples to the agent.

ViewDetails	Code	AgentId	AgentName	DateTime	Ta
ViewDetails	25	1	a	07/11/2011 7:14:03 PM	Em

EMP_CODE	NAME	BAC	Status
411	JADHAV PRALHAD TUKARAM	111509	
412	PARIT SHARAD RAMCHANRA	5061907	

Fig3. Request sent by distributor

2. Leaked dataset given as an input to the system

Fig4. Input leaked dataset

3. The list of all agents having common tuples as that of leaked tuples is found and the corresponding guilt probabilities are calculated.
4. It shows that as the overlap with the leaked dataset minimizes the chances of finding guilty agent increases.

6. CONCLUSION

Data leakage is a silent type of threat. Your employee as an insider can intentionally or accidentally leak sensitive information. This sensitive information can be electronically distributed via e-mail, Web sites, FTP, instant messaging, spreadsheets, databases, and any other electronic means available – all without your knowledge. To assess the risk of distributing data two things are important, where first one is data allocation strategy that helps to distribute the tuples among customers with minimum overlap and second one is calculating guilt probability which is based on overlapping of his data set with the leaked data set .

7. ACKNOWLEGDEMENT

The authors would like to thank Mrs.Leena Jacob and Mrs. Sampada Gadgil for their constant support and encouragement. As well as we are highly thankful to Faculties of SIES and PVPPCOE for throughout guidance.

7. REFERENCES

[1] YIN Fan, WANG Yu, WANG Lina, Yu Rongwei. A Trustworthiness-Based Distribution Model for Data Leakage Detection: Wuhan University Journal Of Natural Sciences.

[2] P. Buneman, S. Khanna and W.C. Tan. Why and where: A characterization of data provenance. ICDT 2001, 8th International Conference, London, UK, January4-6, 2001, Proceedings, volume 1973 of Lecture Notes in Computer Science, Springer, 2001.

[3] S. Czerwinski, R. Fromm, and T. Hodes. Digital music distribution and audio watermarking.

[4] Rakesh Agrawal, Jerry Kiernan. Watermarking Relational Databases// IBM Almaden Research Center

[5] S. Jajodia, P. Samarati, M. L. Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. ACM Trans. Dataset Systems, 26(2):214-260,2001.

[6] Papadimitriou P, Garcia-Molina H. A Model For Data Leakage Detection// IEEE Transaction On Knowledge And Data Engineering Jan.2011.

[7] L. Sweeney. Achieving k- anonymity privacy protection using generalization and suppression. International Journal on Uncertainty, Fuzzyness and Knowledge-based Systems-2002