Data Allocation Strategies for Data Leakage Detection Tool

Archana U. Bhosale [1], Vharkate M.N [2] & Aparna U. Bhosale [3]

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.**

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, 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 (as opposed to simply computing statistics), they may need accurate data for the patients.

Keywords: **Data Mining, Allocation strategies, data leakage, data privacy, fake records, leakage model.**

---

1. College of Engineering, Osmanabad .
archana17_bhosale@yahoo.com

2. College of Engineering, Osmanabad
archana17_bhosale@yahoo.com

3.D. Y. Patil College Of Engineering, Pimpri, Pune
aparnabhosale39@yahoo.co.in

1. INTRODUCTION

In this paper, we develop a model for finding the guilty 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 act as a type of watermark for the entire set, without modifying any individual members. If it turns out that an agent was given one or more fake objects that were leaked, then the distributor can be more confident that agent was guilty. We also consider optimization in which leaked data is compared with original data and accordingly the third party who leaked the data is guessed. We will also be using approximation technique to encounter guilty agents. 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 to improve the probability of identifying leakages. Also there is 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 and for Assessment of guilt probability.

2. LITERATURE SURVEY

The guilt detection approach we present is related to the data provenance problem [3]: tracing the lineage of S objects implies essentially the detection of the guilty agents. 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 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. [3] Finally, there are also lots of other works on mechanisms that allow only authorized users to access sensitive data through access control policies [9], [2]. 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.

3. PROPOSED WORK

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 develop unobtrusive techniques for detecting leakage of a set of objects or record In this section 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

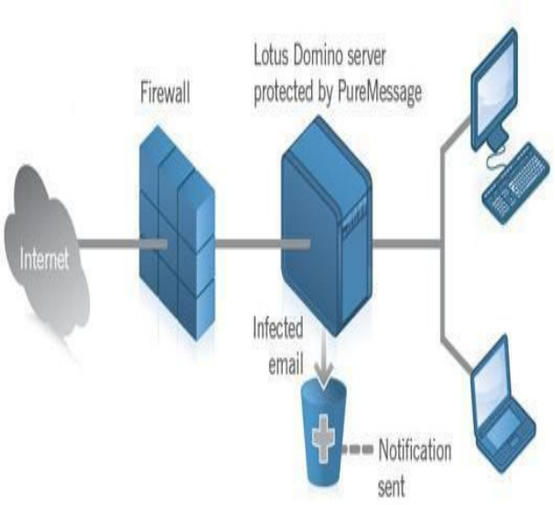

Fig 1: Filtering of data objects.

A. Problem Definition

The distributor owns the sensitive data set T= {t1, t2… tn}. The agent Ai 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 Ri of objects T determined either by implicit request or an explicit request.

- Implicit Request Ri = Implicit (T, mi) : Any subset of mi records from T can be given to agent Ai
- Explicit Request Ri = Explicit (T, Condi) : Agent Ai receives all T objects that satisfy Condition.

B. Data Allocation Problem

1. *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. 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. 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 (EF)
ii. Explicit request without fake tuples (E~F)
iii. Implicit request with fake tuples (IF)
iv. Implicit request without fake tuples (I~F).

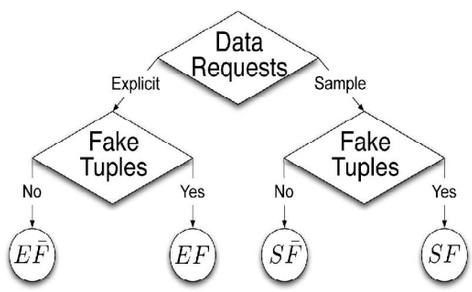

Fig. 2. Leakage problem instances.

2. Optimization Problem:

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. The objective is to maximize the chances of detecting a guilty agent that leaks all his data objects. The Pr { Gj|*S* =*Ri* } or

simply Pr {Gj |*Ri* } is the probability that agent is guilty if the distributor discovers a leaked table *S* that contains all objects .The difference functions $\Delta (i, j)$ is defined as:

$\Delta (i, j) =$ Pr {Gj |*Ri* } – Pr {Gj |*Ri* } …….

Let the distributor have data request from n agents. The distributor wants to give tables R1 ,R2……..Rn to agents A1 ,A2……. An 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.

maximize(overR1….,Rn) (…,.Δ(i,j),…) i≠j……..(A)

minimize(over R1,….,Rn) (.., $\left|Ri \cap Rj\right| \div \left|Ri\right|$,…) i≠j

C. Guilt Model 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 A1, we may suspect A1 more. So probability that agent A1 is guilty for leaking data set L is denoted as Pr{Gi| L}.

D. Guilt Probability Computation (agent guilt model)

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

***Assumption** 1:* For all t1, t2… tn Є L and t1≠ t2, the provenance of t1is independent of t2

Assumption 2: Tuple tЄL can only be obtained by third user in one of the two ways: 1. Single user A1 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 Ai is guilty for the given set L, consider the target guessed t1 with probability p and that agent leaks t1 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 Ut = { Ai | tЄ Ri } 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 Ut can leak t to L with equal probability and using Assumption 2 we get,

Pr(Ai leaked t to L)={(1-p)÷U_1) if Ai € U1 ---- (2)

Given that user Ai is guilty if he leaks at least one value to L, with assumption 1 and equation 2, we can compute the probability that user Pr {Gi| L} Ai is guilty Pr {Gi| L=1- Π t € L∩Ri (1-((1-p)÷U1))} that user Ai is guilty :-----(3)

E. Data Allocation Strategies

In this section we describe allocation strategies that solve exactly or approximately the scalar versions of approximation equation. We resort to approximate solutions in cases where it is inefficient to solve accurately the optimization problem.

1. Explicit Data Requests

In case of explicit data request with fake not allowed, the distributor is not allowed to add fake objects to the distributed data. So Data allocation is fully defined by the agent's data request. In case of explicit data request with fake allowed, the distributor cannot remove or alter the requests R from the agent. However distributor can add the fake object. In algorithm for data allocation for explicit request, the input to this is a set of request ,……, from n agents and different conditions for requests. The e-optimal algorithm finds the agents that are eligible to receiving fake objects. Then create one fake object in iteration and allocate it to the agent selected. The e-optimal algorithm minimizes every term of the objective summation by adding maximum number of fake objects to every set yielding optimal solution.

Step 1: Calculate total fake records as sum of fake Records allowed.

Step 2: While total fake objects > 0

Step3:Select agent that will yield the greatest improvement in the sum objective

i.e. i =argmax((1/$\left|Ri\right|$)-(1/($\left|Ri+1\right|$)))ΣRi∩Rj

Step 4: Create fake record

Step 5: Add this fake record to the agent and also to fake record set.

Step 6: Decrement fake record from total fake record set. Algorithm makes a greedy choice by selecting the agent that will yield the greatest improvement in the sum-objective.

2. Sample Data Requests

With sample data requests, each agent Ui may receive any T subset out of different object allocations. In every allocation, the distributor can permute T objects and keep the same chances of guilty agent detection. The reason is that the guilt probability depends only on which agents have received the leaked objects and not on the identity of the leaked objects. 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. For example, distributor sends the tuples to agents A1 and A2 as R1= {t1, t2} and R2= {t1}.

If the leaked dataset is L= {t1}, then agent A2 appears more guilty than A1. So to minimize the overlap, we insert the fake objects in to one of the agent's dataset. Practically server (Distributor) has given sensitive data to agent. In that distributor can send data with fake information. And that fake information does not affect to Original Data. Fake formation cannot identify by client. it also finds the data leakage from which agent (client)

4. METHODOLOGY

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 Gi 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.

a. Algorithm1:

Allocation of Data Explicitly:
Input: -
i. T= {t1, t2, t3, .tn}-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=Φ, T'=Φ
2. For i=1 to n do
3. If(t .fields==cond) then
4. T'=T'U{ t i}
5. For i=0 to i<m do
6. D=DU{ ti}
7. T'=T'-{ ti}
8. If T'=Φ 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.

b. Algorithm2:

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=Φ 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 Ai

2. for k do Initialize max_rel_ov←0, the maximum relative overlap between Ri the allocation of tk to Ai

3. for all j=1,……,n:j=I and tk ЄRj do calculate absolute overlap as abs_ov← calculate relative overlap as rel_ov←abs_ov/min(mi, mj)

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. 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.
2. Leaked dataset given as an input to the system.
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, spread sheets, 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. REFERENCES

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

[2] International Journal of Computer Trends and Technology- volume3Issue1-2012 ISSN:2231-2803 http://www.internationaljournalssrg.org Data Allocation Strategies for Detecting

Data LeakageSrikanth Yadav, Dr. Y. Eswara rao, V. Shanmukha Rao, R. Vasantha

[3] International Journal of Computer Applications in Engineering Sciences **[**ISSN: 2231-4946**]**197 | P a g e Development of Data leakage Detection Using Data Allocation Strategies Rudragouda G Patil Dept of CSE,The Oxford College of Engg, Bangalore.

[4] 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

[5] S. Jajodia, P. Samarati, M.L. Sapino, and V.S. Subrahmanian, "Flexible Support for Multiple Access Control Policies," ACM Trans. Database Systems, vol. 26, no. 2, pp. 214-260, 2001.

[6] P. Bonatti, S.D.C. di Vimercati, and P. Samarati, "An Algebra for Composing Access Control Policies," ACM Trans. Information and System Security, vol. 5, no. 1, pp. 1-35, 2002.

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

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

.

.

.