IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 1, Issue 1, March, 2013
ISSN: 2320 - 8791
www.ijreat.org

Detection Of Data Divulged Agent

Mr.J.Maruthupandi(AP/IT)[1], J.Dhanushya[2], M.Raji Priyanga[3] and S.V.Suruthi[4]
[1,2,3,4] Department of Information Technology,MSEC,
Sivakasi, Tamilnadu, India.

Abstract

Sometimes sensitive data must be handed over to the supposed trusted third parties. In some situations those data is leaked and found in the unauthorized places. At that instant the distributor has to find out the likelihood that the leaked agent came from one or more agents. This paper focus on detecting when the distributor's sensitive data has been leaked by agents, and also in some cases, to identify the agent that leaked the data. We present a model for calculating "guilt" probabilities for data Leakage. We also present algorithms for distributing objects to agents, in a way that improves our chances of identifying a leaker. At the end, we also consider the option of adding "fake" objects to the distributed set.

Keywords: *sensitive data, third parties, guilt probabilities, data leakage, fake objects.*

1.Introduction

For the purpose of doing business, sometimes sensitive data must be handled by trusted third parties. Example Hospitals maintaining the patients' record, Business process outsourcing. Demanding market conditions encourage many companies to outsource certain business processes (e.g. marketing, human resources) and associated activities to a third party. This model is referred as Business Process Outsourcing (BPO).The recent surge in the growth of the Internet results in offering of a wide range of web-based services, such as database as a service, digital repositories and libraries, e-commerce, online decision support system etc. These applications make the digital assets, such as digital images, video, audio, database content etc, easily accessible by ordinary people around the world for sharing, purchasing, distributing, or many other purposes. The owner of the data is called as **distributor.** The supposed trusted third parties are called **agents.**

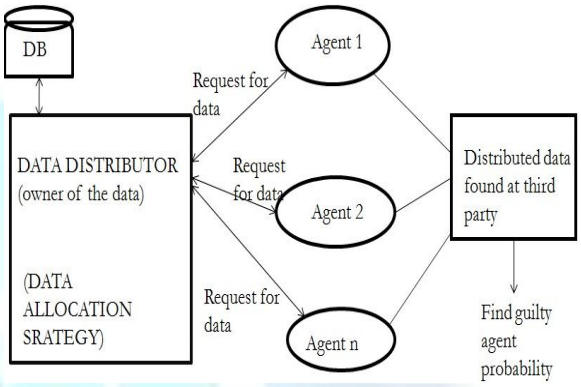

Fig 1: System Architecture

2.Related Work

The guilt detection is related to data provenance problem. Here the work mostly depends on the watermarking .These watermarks were initially used for image, video and audio data whose data representation includes redundancy. The distributed environment is provided by means of Hadoop software. Setting the Hadoop environment includes creating the node.Thus it is necessary to form a hadoop cluster

3.Existing Technique

The existing leakage detection technique is known as **watermarking**. Here the unique code is embedded with each distributed copy. But this unique code can sometimes be destroyed. Watermarking aims to identify a data owner and, hence, is subject to attacks where a pirate claims ownership of the data or weakens a merchant's claims. To overcome this, we study some unobtrusive Techniques for detecting leakage of a set of records.

4.Proposed System

4.1 Data Allocation Problem

The two types of requests we handle: sample and explicit. Fake objects are objects generated by the distributor that are not in set T. The objects are designed to look like real objects, and are distributed

to agents together with the T objects, in order to increase the chances of detecting agents that leak data. Fake objects are represented using four problem instances with the names EF, EF, SF and SF. 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.

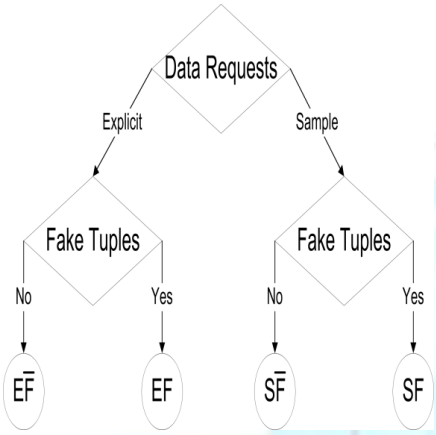

Fig 2: Leakage problem instances

Sample request **Ri = SAMPLE (T; mi)**: Any subset of mi records from T can be given to Ui.

Explicit request **Ri = EXPLICIT (T; Condi)**: Agent Ui receives all the T objects that satisfy condition.

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.
We represent our four problem instances with the names EF, EF, SF and SF, where E stands for explicit requests, S for sample requests, F̄ for the use of fake objects, and F for the case where fake Objects are not allowed.

4.2 Adding 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. Sometimes adding fake records may change the correctness of what the agents do. 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 this case, perturbing the set of distributor objects by adding fake elements is done. 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.

4.3 Agent Guilt Model

To compute P *r{Gi |S}*, we need an estimate for the probability that values in *S* can be "guessed" by the target. For instance, say some of the objects in *S* are emails of individuals. For example we conduct an experiment and ask a person to find an email of 100 individuals. If this
Person can find say 90 emails, and then we can reasonably guess that the probability of finding one email is 0.9. On the other hand, if the objects in question are bank account
Numbers, the person may only discover say 20, leading to an estimate of 0.2. We call this estimate *pt*, the probability that object *t* can be guessed by the target.

5.Allocation Strategies

We describe allocation strategies that solve exactly or approximately the scalar versions of for the different instances presented in Fig.2. We resort to approximate solutions in cases where it is inefficient to solve accurately the optimization problem.

5.1 Explicit data request:

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.

There are two algorithms to obtain the explicit data request.
They are

- E-Optimal
- E-Random

In algorithm for data allocation for explicit request, the input to this is a set of request *R1, R2,…, Rn* 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 *bi* of fake Objects to every set *Ri* yielding optimal solution.

Allocation for Explicit Data Requests
Input: R1. Rn, cond1. . . condn, b1, . . . ,bn, B
Output: R1. Rn, F1. Fn

1: R←Ø //Agents that can receive fake objects
2: for i=1…….n do
3: if bi > 0 then
4: R ←R U {i}
5: Fi ←Ø
6: while B > 0 do
7: i ←SELECT AGENT(R; R1 . . . Rn)
8: f ←CREATE FAKE OBJECT (Ri; Fi; condi)
9: Ri ←Ri U {f}
10: Fi ←Fi U {f}
11: bi ←bi – 1
12: if bi = 0 then
13: R←R / {Ri}
14: B ←B – 1

Algorithm explanation:

- R set contains the resources
- "cond " are used as the condition for watermark
- B set contains the already created fake objects
- SELECTAGENT function used to select the agent with their needed resources R1

a) Agent Selection for E-Random:
1:**function** SELECTAGENT (**R**,R_1, . . .,R_n)
2:i ←select at random an agent from **R**
3:**return** i

b) Agent selection for E-Optimal
1: **Function** SELECTAGENT (**R**,R_1 ,…,R_n)
2: i←argmax($\frac{1}{|R_{i'}|}$ - $\frac{1}{|R_{i'}|+1}$)$\sum_j |R_i \cap R_j|$
3: **return** *i*

5.2 Sample data request

An object allocation that satisfies requests and ignores the distributor's objective is to give each agent Ui a randomly selected subset of T of size mi. The algorithms to perform sample data request includes

- S-Random
- S-Overlap
- S-Max

Allocation for Sample Data Requests

Input: m_1….., m_n, |T|
Output: R1……., Rn

1: a← $0_{|T|}$ //a[k]:number of agents who have received object t_k
2: R ← Ø……., Rn ← Ø
3: remaining ← $\sum_{i=1}^{n}$ m_i
4: while remaining > 0 do
5: for all i=1,…., n : | Ri | < $m_i$do
6: k ← SELECT OBJECT (i, Ri)
7: Ri ← Ri U { t_k }
8: a[k] ← a[k] + 1
9: remaining ← remaining – 1

a)Object Selection for S-Random

1:**function** SELECTOBJECT(i,R_i)
2:k←select at random an element from set{k'|t_k Є R_i}
3:**return** k

b)Object Selection for S-Overlap

1:**function** SELECTOBJECT(i,R_i,a)
2:K←{k|k=argmin a[k']}
k'
3:k←select at random an element from set
{k'|k'Є K ^ t_k ЄR_i}
4:**return** k

c)Object Selection for S-Max

1:**function** SELECTOBJECT(i,R_1,…,R_n,m_1,…,m_n)
2: Min_overlap←1 the minimum out of the maximum relative overlaps that the allocations of different objects to Ui yield.
3:for k ∈ {k'|t_k' ∉ Ri} do
4:max_rel_ov←0 the maximum relative overlap between Ri and any set Rj that the allocation of t_k to Ui yields.
5:for all j=1…n:j≠I and t_k∉ Rj do
6: abs_ov←|Ri∩ Rj| + 1
7:rel_ov←abs_ov/min(mi,mj)
8:max_rel_ov←Max(max_rel_ov,rel_ov)
9:if max_rel_ov ≤ min_overlap then
10:min_overlap ← max_rel_ov
11:ret_k←k

12:**return** ret_k

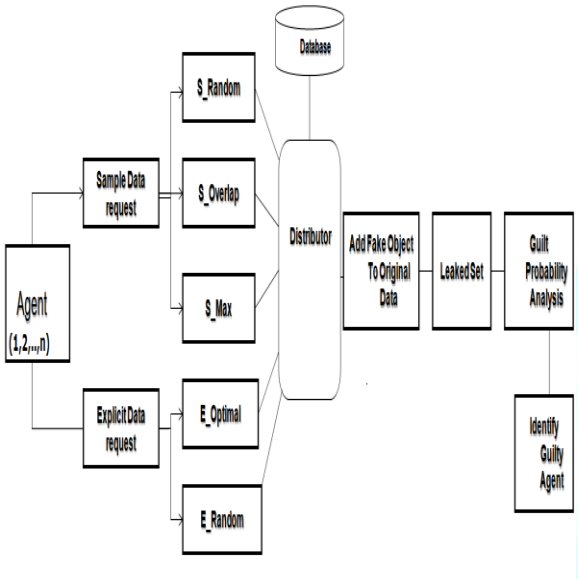

Fig 3: Project Flow

6.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 *Uj* agent is guilty if the distributor
Discovers a leaked table *S* that contains all *Ri* objects.
The difference functions
Δ (i,j) is defined as:

$$\Delta(i,j) = \Pr\{Gi|Ri\} - \Pr\{Gj|Ri\}$$

1) *Problem definition:*
Let the distributor have data requests from *n* agents. The distributor wants to give tables *R1,.....Rn* to agents *U1* , . . . , *Un* respectively, so that
- Distribution satisfies agents' requests; and
- Maximizes the guilt probability differences Δ (*i, j*) for all *i, j* = 1. . . *n* and *i* = *j*.

2) *Optimization problem:*
Maximize (. . . , Δ (*i, j*), . . .) *i* ! = *j*
(*over R1* , . . . ,*Rn*)

The approximation of objective of the above equation does not depend on agent's probabilities
Therefore minimize the relative overlap among the agents as

Minimize (…, $\left|Ri \cap Rj\right|/_{Ri}$, … … . .) $i! = j$
(*over R1* , . . . , *Rn*)

7.Implementation technique

Hadoop tool is used to form the distributed environment. In this environment one system acts as a master and other systems act as a slave. Entire database is in the master system and the slaves act as a agents. The third parties can be any other system outside our network. The agents who act as a slave can be a master to the third parties which will perform the part of the slave.Leaked set can be given as input or identified from the website.

8.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. 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 its 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.

Acknowledgement

Students work is incomplete until they thank the almighty & his teachers. We sincerely believe in this and would like to thank Dr. T.Revathi, Head of the Department, and Dr.K.Vijayalakshmi, project

coordinator, Information and Technology,Mepco, Sivakasi, for her encouragement and motivation to write this paper. Also we are grateful to Mr. J.MaruthuPandi, Assistant Professor., (IT), Mepco, Sivakasi for guiding me in writing this paper.

References

[1] P. Papadimitriou and H. Garcia-Molina, "Data leakage detection," IEEE transactions on Knowledge and data engineering, Vol.23, No.1, January 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] P. Papadimitriou and H. Garcia-Molina, "Data leakage detection," Stanford University, Tech. Rep., 2008

[4] 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 4-6,2001Computer Science*, pages 316–330. Springer, 2001.