Graduate Research in Engineering and Technology (GRET)

Volume 1 | Issue 1 Article 18

July 2013

LEAKING AGENT DETECTION AND EMAIL FILTERING

GIRISH SHANKAR
Computer Engineering, Sinhgad Institute OF Technology & Science, Pune,Maharashtra,PIN :411041,
girish.shankar25@gmail.com

SOMESHWAR DHAYALAN
Computer Engineering, Sinhgad Institute OF Technology & Science, Pune,Maharashtra,PIN :411041,
dhayalan@gmail.com

ASHISH ANAND
Computer Engineering, Sinhgad Institute OF Technology & Science, Pune,Maharashtra,PIN :411041,
ashishanand6566@gmail.com

LEAKING AGENT DETECTION AND EMAIL FILTERING

GIRISH SHANKAR[1], SOMESHWAR DHAYALAN[2] & ASHISH ANAND[3]

[1,2&3]Computer Engineering, Sinhgad Institute OF Technology & Science, Pune,Maharashtra,PIN :411041
EMAIL ID:girish.shankar25@gmail.com, dhayalan@gmail.com, ashishanand6566@gmail.com

Abstract: Leaking of confidential data to an unauthorized agent is a major concern for an organization. In this article we seek to detect the trusted node that leaks the confidential data to an unauthorized agent. Traditionally, leakage of data is handled by water marking technique which requires data modification. If the watermarked copy is found at some unauthorized site then distributor can claim his ownership. But one of the issues with watermarking method is data modification. To overcome the disadvantages of using watermark, data allocation strategies are used to improve the probability of identifying guilty third parties. The idea is to distribute the data intelligently to agents based on sample data request and explicit data request in order to improve the chance of detecting the guilty agents. Modern business activities also rely on extensive email exchange. Email leakages have become widespread, and the severe damage caused by such leakages constitutes a disturbing problem for organizations. Hence, filtering of E-mails is also necessary. This can be done by blocking E-mails which contains images, videos or sensitive data and filtering the text file of an organization.

Keywords*: Sensitive Data, Fake Objects, Data Allocation Strategies.*

1. LEAKING AGENT DETECTION

I. INTRODUCTION

In the course of doing business, sometimes sensitive data must be handed over to supposedly trusted third parties. For example, personal information provided to cellular companies may be used by other companies for advertising. 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. There may be the case that a trusted agent may leak confidential data to an unauthorized agent. Therefore, it becomes necessary to identify the leaking agent in order to have a successful working within an organization.

In the last years watermarking techniques have emerged as an important building block which allow the owner of the data to embed an imperceptible watermark into the data 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. Furthermore, watermarks can sometimes be destroyed if the data recipient is malicious. We study unobtrusive techniques for detecting leakage of a set of objects or records. Specifically, 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. If the distributor sees "enough evidence" that an agent leaked data, he may stop doing business with him, or may initiate legal proceedings.

II. 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. 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.

III. 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.

IV. DATA ALLOCATION PROBLEM

A. 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. 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 :

- Explicit request with fake tuples (EF)
- Explicit request without fake tuples (EF')
- Implicit request with fake tuples (SF)
- Implicit request without fake tuples (SF').

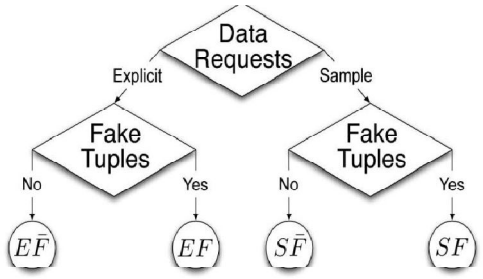

Fig 1: Leakage Problem Instances

B. 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.

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 $\Delta (i, j)$ for all i, j= 1, 2, ……n and i≠j.

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

ii. minimize(overR1,….,Rn)(.., | Ri∩Rj | ÷|Ri| …) i≠j …….(B)

C. GUILTY AGENTS

Suppose that after giving objects to agents, the distributor discovers that a set S _ T has leaked. This means that some third party, called the target, has been caught in possession of S. For example, this target may be displaying S on its website, or perhaps as part of a legal discovery process, the target turned over S to the distributor. Since the agents U1; . . . ; Un have some of the data, it is reasonable to suspect them leaking the data. However, the agents can argue that they are innocent, and that the S data were obtained by the target through other means. For example, say that one of the objects in S represents a customer X. Perhaps X is also a customer of some other company, and that company provided the data to the target.

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)÷U1) 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)

D. 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.

- EXPLICIT DATA REQUESTS

In problems of class EF, the distributor is not allowed to add fake objects to the distributed data. So, the data allocation is fully defined by the agents' data requests. Therefore, there is nothing to optimize. Objective values are initialized by agents' data requests. Say, for example, that T={t1,t2} and there are two agents with explicit data requests such that R1={t1,t2} and R2={t1}. The value of the sum objective is in this case :

$$\sum_{i=1}^{2} 1/|Ri| \sum_{\substack{j=1 \\ j\neq i}}^{2} |Ri \cap Rj| = \frac{1}{2} + \frac{1}{1} = 1.5$$

The distributor cannot remove or alter the R1 or R2 data to decrease the overlap R1∩ R2. However, say that the distributor can create one fake object (B =1) and both agents can receive one fake object (b1 =b2 =1). In this case, the distributor can add one fake object to either R1 or R2 to increase the corresponding denominator of the summation term. Assume that the distributor creates a fake object f and he gives it to agent R1. Agent U1 has now R1= {t1,t2,f} and F1 ={f} and the value of the sum-objective decreases to :

$$\frac{1}{3} + \frac{1}{1} = 1.33 < 1.5$$

If the distributor is able to create more fake objects, he could further improve the objective. We present in Algorithms 1 and 2 a strategy for randomly allocating fake objects. Algorithm 1 is a general "driver" that will be used by other strategies, while Algorithm 2 actually performs the random selection. We denote the combination of Algorithm 1 with 2 as e-random. We use e-random as our baseline in our comparisons with other algorithms for explicit data requests.

Algorithm 1. Allocation for Explicit Data Requests (EF)
Input: R1; . .;Rn, cond1; . . ; condn, b1; .. ; bn,
Output: R1; . . .;Rn, F1; . . . ; Fn
1: R←Φ (Agents that can receive fake objects)
2: for i= 1 to n do
3: if bi > 0 then
4: R←R □ {i}
5: Fi← Φ
6: while B> 0
7: i← SELECTAGENT(R,R1,….,Rn)
8: f← CREATEFAKEOBJECT(Ri,Fi,condi)
9: Ri← Ri □ {f}
10: Fi← Fi □ {f}
11: bi←bi-1
12: if bi=0 then
13: R←R/{Ri}
14: B←B-1

In lines 1-5, Algorithm 1 finds agents that are eligible to receiving fake objects in O(n) time. Then, in the main loop in lines 6-14, the algorithm creates one fake object in every iteration and allocates it to random agent. The main loop takes O(B) time. Hence, the running time of the algorithm is O(n + B). If , the algorithm minimizes every term of the objective summation by adding the maximum number bi of fake objects to every set Ri, yielding the optimal solution. Otherwise, if B , the algorithm just selects at random the agents that are provided with fake objects. We return back to our example and see how the objective would change if the distributor adds fake object f to R2 instead of R1. In this case, the sum-objective would be

$$\frac{1}{2} + \frac{1}{2} = 1 < 1.33$$

The reason why we got a greater improvement is that the addition of a fake object to R2 has greater impact on the corresponding summation terms, since

$$\frac{1}{|R1|} - \frac{1}{|R1|+1} = \frac{1}{6} < \frac{1}{|R2|} - \frac{1}{|R2|+2} = \frac{1}{2}$$

The left-hand side of the inequality corresponds to the objective improvement after the addition of a fake object to R1 and the right-hand side to R2.

- 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.

Algorithm 2: Implicit sample Data Request

1: a← t // a[k]: number of agent who have received object tk
2: R1←Φ,…,Rn ←Φ
3: remaining ←
4: while remaining > 0 do
5: for all i=1…,n : |Ri| < mi do
6: k // May also use additional parameters
7: Ri ←Ri □ {tk}
8: a[k]←a[k]+1
9: remaining← remaining -1

V. METHODOLOGY

We present the algorithm and the corresponding results for the explicit data allocation with the addition of fake tuples. 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. 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.

2. Email-Filtering

Suspicious email detection is a kind of mailing system where suspicious users are identified by determining the keywords or URL used by him. Mails containing keywords or important files are blocked by the administrator so that they cannot be forwarded. All these blocked mails are checked by the administrator to identify the users who sent such mails. The users of this system are composing mails to the other users who are authenticated already.

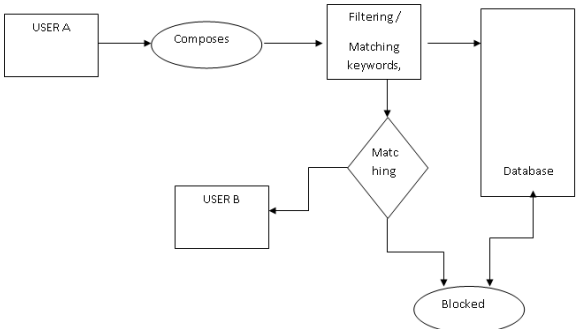

Fig 2: Suspicious Email Blocking

VI. CONCLUSION

Data leakage is a silent type of threat. An employee as an insider can intentionally or accidentally leak sensitive information which can be electronically distributed via e-mail, Web sites, FTP, instant messaging, spreadsheets, databases, and any other electronic means available – all without 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. Our model is relatively simple, but we believe it captures the essential trade-offs. The algorithms we have presented implement data distribution strategies that can improve the distributor's chances of identifying a leaker.

ACKNOWLEDGMENT

Students work is incomplete until they thank the his teachers. I sincerely believe in this and would like to thank Prof. Nihar Ranjan, Computer Engineering, S.I.T.S, PUNE for his encouragement and motivation to write this paper.

REFERENCES

[1] Leakage detection, IEEE Transactions on Knowledge and Data Engineering, pages 51-63, volume 23, 2011.

[2] International Journal of Computer Trends and Technology- volume3Issue1- 2012 ISSN:2231-2803

[3] R. Agrawal and J. Kiernan. "*Watermarking Proceedings of the 28th international relational databases"*. In VLDB '02:conference on Very Large Data Bases, pages 155–166. VLDB Endowment, 200

[4] P. Papadimitriou and H. Garcia-Molina, "*Data Leakage Detection*," technical report, Stanford Univ., 2008

[5] Buneman P. and Tan W.C. (2009) *ACM SIGMOD*, 38(2), 42-49.

[6] Panagiotis Papadimitriou, Hector Garcia-Molina (2009) *IEEE International Conference on Data Engineering*. 1307-1310.