IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 8, Issue 4 (Jan. - Feb. 2013), PP 79-84
www.iosrjournals.org

Detection of Data Leakage Using Unobtrusive Techniques

Mr. Ajinkya S. Yadav[1], Mr. Ravindra P. Bachate[2], Prof. Shadab A. Pattekari[3].
[1,2]*Department of Computer Engineering, JSCOE-Pune, Pune University, Maharashtra-411028, India.*
[3] *Department of Computer Science and Engg. TKIET, Warananagar.-416113 Maharashtra, India.*

Abstract: *A data distributor has given sensitive data to a set of supposedly trusted agents (third parties). Some of the data is 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.*

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

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[2].

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.

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) and it allows companies to focus on their core competency by subcontracting other activities to specialists, resulting in reduced operational costs and increased productivity. Security and business assurance are essential for BPO. In most cases, the service providers need access to a company's intellectual property and other confidential information to carry out their services. For example a human resources BPO vendor may need access to employee databases with sensitive information (e.g. social security numbers), a patenting law firm to some research results, a marketing service vendor to the contact information for customers or a payment service provider may need access to the credit card numbers or bank account numbers of customers. The main security problem in BPO is that the service provider may not be fully trusted or may not be securely administered. Business agreements for BPO try to regulate how the data will be handled by service providers, but it is almost impossible to truly enforce or verify such policies across different administrative domains [1]. Due to their digital nature, relational databases are easy to duplicate and in many cases a service provider may have financial incentives to redistribute commercially valuable data or may simply fail to handle it properly. Hence, we need powerful techniques that can detect and deter such dishonest. We study unobtrusive techniques for detecting leakage of a set of objects or records. Specifically, we study 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 web site, 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. 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[2].

1.2 Techniques used in proposed system

1.2.1 Perturbation Technique

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[4].

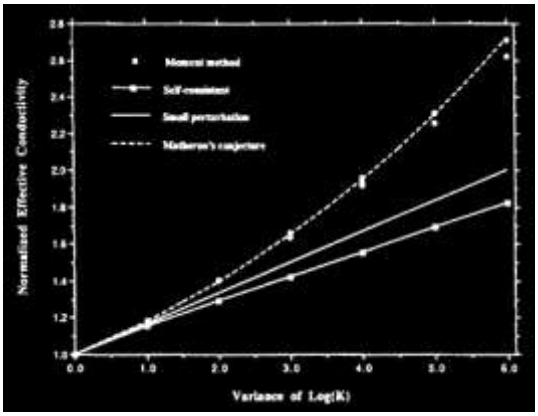

Fig. 1.2.1Graph showing perturbation

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[7].

1.2.2 Unobtrusive Techniques

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 [6].

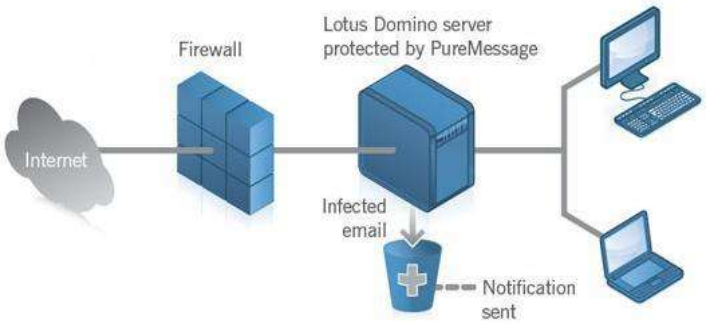

Fig. 1.2.2 Typical Block Diagram Showing the Process of Data Loss in Blocking spam.

II. Entities And Agents

A distributor owns a set $T=\{t_{1,\ldots},t_m\}$of valuable data objects. The distributor wants to share some of the objects with a set of agents $U_1,U_2,\ldots U_n$, but does not wish the objects 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 [2]. An agent Ui receives a subset of objects, determined either by a sample request or an explicit request:

1. Sample request
   Ri=SAMPLE(T,mi): Any subset of mi records from T can be given to Ui.
2. Explicit request
   Ri=EXPLICIT(T,condi): Agent Ui receives all T objects that satisfy condi.

2.1 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[5].

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. Or perhaps X can be reconstructed from various publicly available sources on the web. Our goal is to estimate the likelihood that the leaked data came from the agents as opposed to other sources. Intuitively, the more data in S, the harder it is for the agents to argue they did not leak anything. Similarly, the "rarer" the objects, the harder it is to argue that the target obtained them through other means. Not only do we want to estimate the likelihood the agents leaked data, but we would also like to find out if one of them, in particular, was more likely to be the leaker. For instance, if one of the S objects was only given to agent U1, while the other objects were given to all agents, we may suspect U1 more. The model we present next captures this intuition. We say an agent Ui is guilty and if it contributes one or more objects to the target. We denote the event that agent Ui is guilty by Gi and the event that agent Ui is guilty for a given leaked set S by Gi|S. Our next step is to estimate Pr{Gi|S}, i.e., the probability that agent Ui is guilty given evidence S [12].

III. Agent Guilt Model

We can conduct an experiment and ask a person with approximately the expertise and resources of the target to find the email of say 100 individuals. If this person can find say 90 emails, 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.

In this model we taken T as the total content of objects and the R's are used as the set of objects given to the agents and s is the target set which contains the leaked objects. There are T={t1,t2,t3};R1={t1,t2};R2{t2,t3};S={t1,t2,t3} In this case, all three of the distributor's objects have been leaked and appear in S. Let us first consider how the target may have obtained object t1, which was given to both agents. The target either guessed t1 or one of U1 or U2 leaked it [3]. We know that the probability of the former event is p, so assuming that probability that each of the two agents leaked t1 is the same we have the following cases:

- The target guessed t1 with probability p;
- Agent U1 leaked t1 to S with probability (1 - p)/2;
- Agent U2 leaked t1 to S with probability (1 -p)/2;

Similarly, we find that agent U1 leaked t2 to S with Probability 1 - p since he is the only agent that has t2. Given these values, the probability that agent U1 is not Guilty, namely that U1 did not leak either object is:(1 -(1 -p)/2) -(1 -(1 - p)); (1) And the probability that U1 is guilty is: 1 - Pr{G1} Note that if did not hold, our analysis would be more complex because we would need to consider joint events, e.g., the target guesses t1 and at the same time one or two agents leak the value. In our simplified analysis we say that an agent is not guilty when the object can be guessed, regardless of whether the agent leaked the value. Since we are "not counting" instances when an agent leaks information, the Simplified analysis yields conservative values (smaller Probabilities) [10].

IV. Modules Of Proposed System

4.1. Data Allocation Module:

Our main focus is the data allocation problem: how can the distributed "intelligently" give data to agents in order to improve the chances of detecting a guilty agent? As illustrated in Fig. 2, there are four instances of this problem we address, depending on the type of data requests made by agents and whether "fake objects" are allowed. The two types of requests we handle were defined in Section 2: sample and explicit [13]. 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 T objects, in order to increase the chances of detecting agents that leak data. We discuss fake objects in more detail in later section [14].

As shown in Fig. 2, we represent our 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 [10]. Note that, for simplicity, we are assuming that in the E problem instances, all agents make explicit requests, while in the S instances, all agents make sample requests. Our results can be extended to handle mixed cases, with some explicit and some sample requests. We provide here a small example to illustrate how mixed requests can be handled, but then do not elaborate further.

4.2. Fake Object Module:

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. The idea of perturbing data to detect leakage is not new, e.g., [1]. 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 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 [8].

For example, say that 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[11].

The distributor creates and adds fake objects to the data that he distributes to agents. We let Fi _ Ri be the subset of fake objects that agent Ui receives. As discussed below, fake objects must be created carefully so that agents cannot distinguish them from real objects.

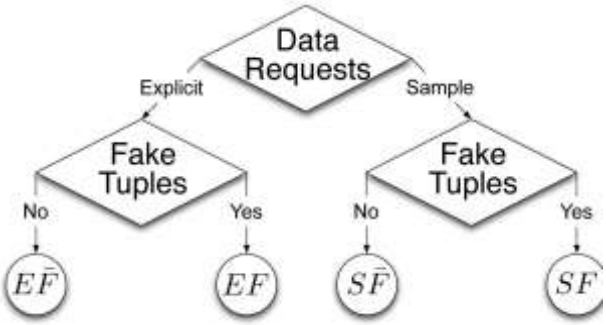

Fig.4.2.1 Fake Object

In many cases, the distributor may be limited in how many fake objects he can create. For example, objects may contain e-mail addresses, and each fake e-mail address may require the creation of an actual inbox (otherwise, the agent may discover that the object is fake). The inboxes can actually be monitored by the distributor: if e-mail is received from someone other than the agent who was given the address, it is evident that the address was leaked. Since creating and monitoring e-mail accounts consumes resources, the distributor may have a limit of fake objects. If there is a limit, we denote it by B fake objects. Similarly, the distributor may want to limit the number of fake objects received by each agent so as to not arouse suspicions and to not adversely impact the agents' activities. Thus, we say that the distributor can send up to bi fake objects to agent Ui.

The distributor can also use function CREATEFAKEOBJECT() when it wants to send the same fake object to a set of agents. In this case, the function arguments are the union of the Ri and Fi tables, respectively, and the intersection of the conditions condis. Although we do not deal with the implementation of CREATEFAKEOBJECT(), we note that there are two main design options. The function can either produce a fake object on demand every time it is called or it can return an appropriate object from a pool of objects created in advance [2].

4.3. Optimization Module:

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[9].

4.4. Data Distributor:

A data distributor has given sensitive data to a set of supposedly trusted agents (third parties). Some of the data is 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 [9].

V. Algorithms

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. In algorithm for data allocation for explicit request, the input to this is a set of request *R1, R2… R_n* 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 R_i yielding optimal solution.

Step 1: Calculate total fake records as sum of fake records allowed.
Step 2: While total fake objects > 0
Step 3: Select agent that will yield the greatest improvement in the sum objective
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.

5.2 Sample Data Request

With sample data requests, each agent U_i may receive any T from a subset out of different ones. Hence, there are different 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. Therefore, from the distributor's perspective there / |T| are different allocations. An object allocation that satisfies requests and ignores the distributor's objective is to give each agent a 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.

Step 1: Initialize Min_overlap ← 1, the minimum out of the maximum relative overlaps that the allocations of different objects to Ui

Step 2: for k ∈ {k |tk ∈ Ri} do
Initialize max_rel_ov ← 0, the maximum relative overlap between Ri and any set Rj that the allocation of tk toUi

Step 3: for all j = 1... n: j = i and tk ∈ Rj do
Calculate absolute overlap as
abs_ov← |Ri ∩Rj | + 1
Calculate relative overlap as
rel_ov ← abs_ov / min (mi ,mj)

Step 4: Find maximum relative as
max_rel_ov← MAX (max_rel_ov, rel_ov)
If max_rel_ov ≤ min_overlap then
min_overlap ← max_rel_ov
ret_k ←k
Return ret_k

VI. 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 percent 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 that 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 that 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.

References

[1] Technical Report TR-BGU-2409-2010 24 Sept. 2010 1 A Survey of Data Leakage Detection and Prevention Solutions P.P (1-5, 24-25) A. Shabtai, a. Gershman, M. Kopeetsky, y. Elovici Deutsche Telekom Laboratories at Ben-Gurion University, Israel.

[2] IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 22, NO. 3, MARCH 2011 Data Leakage Detection Panagiotis Papadimitriou, Member, IEEE, Hector Garcia-Molina, Member, IEEE P.P (2,4-5)

[3] Data Leakage: What You Need to Know by Faith M. Heikkila, Pivot Group Information Security Consultant. P.P (1-3)

[4] International Journal of Computer Applications in Engineering Sciences [VOL I, ISSUE II, JUNE 2011] [ISSN: 2231-4946] P.P (1, 4) Development of Data leakage Detection Using Data Allocation Strategies Rudragouda G Patil Dept of CSE, The Oxford College of Engg, Bangalore.

[5] Mr.V.Malsoru, Naresh Bollam/ International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com Vol. 1, Issue 3, pp.1088-1091 1088 | P a g e REVIEW ON DATA LEAKAGE DETECTION.

[6] Mr.V.Malsoru, Naresh Bollam/ International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com Vol. 1, Issue 3, pp.1088-1091 1088 | P a g e REVIEW ON DATA LEAKAGE DETECTION.

[7] International Journal of Computer Applications in Engineering Sciences[VOL I, ISSUE II, JUNE 2011] [ISSN: 2231-4946] P.P (1, 4)Development of Data leakage Detection Using Data Allocation StrategiesRudragouda G Patil Dept of CSE, The Oxford College of Engg, Bangalore.patilrudrag@gmail.com

[8] A Model for Data Leakage Detection Panagiotis Papadimitriou 1, Hector Garcia-Molina 2 Stanford University 353 Serra Street, Stanford, CA 94305, USA P.P (1, 4-5) 1papadimitriou@stanford.edu

[9] Web-based Data Leakage Prevention Sachiko Yoshihama1, Takuya Mishina1, and Tsutomu Matsumoto2 1 IBM Research - Tokyo, Yamato, Kanagawa, Japan fsachikoy, tmishinag@jp.ibm.com, P.P (2,14) 2 Graduate School of Environment and Information Sciences, Yokohama National University, Yokohama, Kanagawa, Japan tsutomu@ynu.ac.jp

[10] Data Leakage: Affordable Data Leakage Risk Management by Joseph A. Rivela Senior Security Consultant P.P (4-6)

[11] Data Leakage Prevention: A news letter for IT Professionals Issue 5 P.P (1-3)

[12] The Who, What, When & Why of Data Leakage Prevention/Protection Presented by: Archie Alimagno California Department of Insurance P.P (2-7)

[13] An ISACA White Paper Data Leak Prevention P.P (3-7)

[14] Mr.V.Malsoru, Naresh Bollam/ International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com Vol. 1, Issue 3, pp.1088-1091 1088 | P a g e REVIEW ON DATA LEAKAGE DETECTION.