Global Journal of Computer Science and Technology: E
Network, Web & Security
Volume 17 Issue 4 Version 1.0 Year 2017
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Data Leakage Detection

By Rajesh Kumar
Manav Rachna International University

Abstract- Perturbation is a very useful technique where the data is 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. Traditionally, leakage detection is handled by watermarking, 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. Watermarks can be very useful in some cases, but again, involve some modification of the original data. Furthermore, watermarks can sometimes be destroyed if the data recipient is malicious. In this paper 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.

GJCST-E Classification: K.8.1, B.4.2

DATALEAKAGEDETECTION

Strictly as per the compliance and regulations of:

Data Leakage Detection

Rajesh Kumar

Abstract- Perturbation is a very useful technique where the data is 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. Traditionally, leakage detection is handled by watermarking, 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. Watermarks can be very useful in some cases, but again, involve some modification of the original data. Furthermore, watermarks can sometimes be destroyed if the data recipient is malicious. In this paper 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.

I. Introduction

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, 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. 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.

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. Using an analogy with cookies stolen from a cookie jar, if we catch Freddie with a single cookie, he can argue that a friend gave him the cookie. But if we catch Freddie with 5 cookies, it will be much harder for him to argue that his hands were not in the cookie jar. If the distributor sees 'enough evidence´ that an agent leaked data, he may stop doing business with him, or may initiate legal proceedings. In this paper 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[1].

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[1]. The idea of perturbing data to detect leakage is not new, e.g.,. 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. 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. 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 Buses the purchased mailing list, A receives copies of the mailing. These records area 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. 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. In many cases, the distributor may be limited in how many fake objects he can create. For example, objects may contain email addresses, and each fake email address may require the creation of an actual inbox (otherwise the agent may discover the object is fake). The inboxes can actually be monitored by the distributor: if email is received from someone other than the agent who was given the address, it is

Author: *Manav Rachna International University.*
e-mail: rajesh.sharmag96@gmail.com

evidence that the address was leaked. Since creating and monitoring email 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 agent's activities. Thus, we say that the distributor can send up to bi fake objects to agent Ui Creation.

The creation of fake but real-looking objects is a non-trivial problem whose thorough investigation is beyond the scope of this paper. Here, we model the creation of a fake object for agent Ui as a black-box function CREATE FAKE OBJECT(Ri; Fi; Condi) that takes as input the set of all objects Ri, the subset of fake objects.Fi that Ui has received so far and Condi, and returns anew fake object. This function needs Condi to produce a valid object that satisfies Ui's condition. Set Ri is needed as input so that the created fake object is not only valid but also indistinguishable from other real objects. For example, the creation function of a fake payroll record that includes an employee rank and a salary attribute may take into account the distribution of employee ranks, the distribution of salaries as well as the correlation between the two attributes. Ensuring that key statistics do not change by the introduction of fake objects is important if the agents will be using such statistics in their work.

II. Literature Survey

a) *Agent Guilt Model*

Suppose an agent Ui is guilty if it contributes one or more objects to the target. The event that agent Ui is guilty for a given leaked set S diesnoted by G i| S. The next step is to estimate Pr {Gi| S }, i.e., the probability that agentGi is guilty given evidence S.

To compute the Pr {Gi| S}, estimate the probability that values in Sbcean "guessed" by the target. For instance, say some of the objects in t are emails of individuals. Conduct an experiment and ask a person to find the email of say 100 individuals, the person may only discover say 20, leading to an estimate of 0.2. Call this estimate as pt, the probability that object t can be guessed by the target.

The two assumptions regarding the relationship among the various leakage events.

Assumption 1: For all t, t ∈ S such that ≉ T the provenance of t is independent of the provenance of T.

The term provenance in this assumption statement refers to the source of a value t that appears in the leaked set. The source can be any of the agents who have t in their sets or the target itself.

Assumption 2: An object t ∈ S can only be obtained by the target in one of two ways.

- A single agent Ui leaked t from its own Ri set, or
- The target guessed (or obtained through other means) t without the help of any of the n agents.

To find the probability that an agent Ui is guilty given a set S, consider the target guessed t1 with probability p and that agent leaks t1 to Sthweith probability 1-p. First compute the probability that he leaks a single object t to S. To compute this, define the set of agents $V_t = \{U_i \mid t <- R_t\}$ that have t in their data sets. Then using Assumption 2 and known probability p,

We have,

Pr {some agent leaked t to S} = 1- p (1.1)

Assuming that all agents that belong to Vt can leak t to S with equal probability and using Assumption 2 obtain,

Pr {Ui leaked t to S} = (1.2)

Given that agent Ui is guilty if he leaks at least one value to S, with Assumption 1 and Equation 1.2 compute the probability Pr { Gr| S}, agentUi is guilty,

Pr {Gi| S} (1.3)

b) *Data Allocation Problem*

The distributor "intelligently" gives data to agents in order to improve the chances of detecting a guilty agent. There are four instances of this problem, depending on the type of data requests made by agents and whether "fake objects" [4] are allowed. Agent makes two types of requests, called sample and explicit. Based on the requests the fakes objects are added to data list.

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.

c) *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.

We consider the constraint as strict. The distributor may not deny serving an agent request and may not provide agents with different perturbed versions of the same objects. The fake object distribution as the only possible constraint relaxation. The objective is to maximize the chances of detecting a guilty agent that leaks all his data objects.

The Pr $\{G_i \mid S = R_i\}$ or simply Pr $\{G_i \mid R_i\}$ is the probability that agent U_i is guilty if the distributor discovers a leaked table S that contains all R_i objects. The difference functions Δ (i, j) is defined as:

$$\Delta (i, j) = \Pr \{G_i \mid R_i\} - \Pr \{G \mid R_i\} \qquad (1.4)$$

i. *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.

Assuming that the sets satisfy the agents' requests, we can express the problem as a multi-criterion

ii. *Optimization Problem*

Maximize (. . . , Δ (i, j), . . .) i! = j (1.5)

(Over R1,….., Rn,)

The approximation [3] of objective of the above equation does not depend on agent's probabilities and therefore minimize the relative overlap among the agents as

Minimize (. . . ,(|Ri∩Rj|) / Ri , . . .) i != j (1.6)

(over R1 , . . . ,Rn)

This approximation is valid if minimizing the relative overlap, (|Ri∩Rj|) / Ri maximizes Δ (i, j).

III. Allocation Strategies Algorithm

There are two types of strategies algorithms

a) 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 requestR1, 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.

Algorithm 1 : Allocation for Explicit Data Requests (EF)

Input: R1, . . . , Rn, cond1, . . . , condn, b1, . . . ,bn, B

Output: R1, . . . , Rn, F1,. . . ,Fn

Step 1: R ← Ø , Agents that can receive fake objects

Step 2: for i = 1,……., n do

Step 3: if bi > 0 then

Step 4: R ← R U {i}

Step 5: Fi ← Ø; Step 6: while B > 0 do

Step 7: i ← ELECTAGENT(R,R1,……..,Rn)

Step 8: f ← CREATEFAKEOBJECT (Ri, Fi, condi)

Step 9: Ri ← Ri U {i}

Step 10: Fi ← Fi U {i}

Step 11: bi ← bi - 1

Step 12: if bi = 0 then

Step 13: R ← R \{Ri}

Step 14: B ← B – 1.

Algorithm 2 : Agent Selection for e-random

Step 1: function SELECTAGENT(R,R1,……,Rn)

Step 2: i ← select at random an agent from R

Step 3: return I

Algorithm 3: Agent selection for e-optimal

Step 1: function SELECTAGENT(R;R1; : : : ;Rn)

Step 2: i ← argmax $\left(\frac{1}{R_{i'}} - \frac{1}{R_{i'}+1}\right) \sum_j |R_{i'} \cap Rj|$

Step 3: return i $i' : R \in R$

b) Sample Data Request

With sample data requests, each agent Ui may receive any T from a subset out of $\binom{|T|}{m}$ different ones. Hence,there are $\prod_{i=1}^{n}\binom{|T|}{m}$ 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 are $\prod_{i=1}^{n}\binom{|T|}{m} / |T|$ 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.

Algorithm 4: Allocation for Sample Data Requests (SF)

Input: m1, . . . , mn, |T| . Assuming mi <= |T|

Output: R1,……..,Rn

Step 1: a ← 0|T| . a[k]:number of agents who have received object tk

Step 2: R1,……….,Rn ;

Step 3: remaining ← $\sum_{i=1}^{n} mi$

Step 4: while remaining > 0 do

Step 5: for all i = 1,….., n : |Ri| < mi do

Step 6: k ← SELECTOBJECT (i, Ri). May also use additional parameters

Step 7: Ri ← Ri U {tk}

Step 8: a[k] ← a[k] + 1

Step 9: remaining ← remaining–1.

Algorithm 5 : Object Selection for s-random

Step 1: function SELECTOBJECT(i , Ri)

Step 2: k ← select at random an element from set{ k' ∉ tk' Ri }

Step 3: return k.

Algorithm 6 : Object Selection for s-overlap

Step 1: function SELECTOBJECT(i;Ri; a)

Step 2: K ← {k | k = argmin a[k']}

Step 3: k ← select at random an element from set {k' | k' ∈ K ^ tk' ∉ Ri}

Step 4: return k.

Algorithm 7 : Object Selection for s-max

Step1: function SELECTOBJECT(i, R1,…….,Rn ,m1,……..,mn)

Step 2: min_ overlap ← 1 . The minimum out of the maximum relative overlaps that the allocations of different objects to Ui yield

Step 3: for k {k' | tk' Ri } do

Step 4: max_ rel_ ov ← 0. The maximum relative overlap between Ri and any set Rj that the allocation of tk to Ui yields

Step 5: for all j = 1,…………, n : j i and tk Rj do

Step 6: abs_ ov ← | Ri ∩ Rj | + 1

Step 7: rel_ ov ← abs_ ov /min (mi , mj)

Step 8: max_ rel_ ov ← MAX(max_rel_ov , rel_ov)

Step 9: if max_ rel_ ov <= min_ overlap then

Step 10: min_overlap ← max_ rel_ ov

Step 11: ret_ k ← k

Step 12: return ret_ k.

IV. Existing System

There are conventional techniques being used and include technical and fundamental analysis. The main issue with these techniques is that they are manual and need laborious work along with experience.

Traditionally, leakage detection is handled by watermarking, 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. Watermarks can be very useful in some cases, but again, involve some modification of the original data. Furthermore, watermarks can sometimes be destroyed if the data recipient is malicious. E.g. . A hospital may give patient records to researchers who will devise new treatments. 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[4].

We call the owner of the data the distributor and the supposedly trusted third parties the agents. The distributor gives the data to the agents. These data will be watermarked. Watermarking is the process of embedding the name or information regarding the company. The examples include the pictures we have seen in the internet. The authors of the pictures are watermarked within it. If anyone tries to copy the picture or data the watermark will be present. And thus the data may be unusable by the leakers.

a) *Disadvantage*

This data is vulnerable to attacks. There are several techniques by which the watermark can be removed. Thus the data will be vulnerable to attacks.

V. Proposed System

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. We also present algorithm for distributing object to agent.

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 records. 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. Today the advancement in technology made the watermarking system a simple technique of data authorization. There are various software which can remove the watermark from the data and makes the data as original[5].

a) Advantage

This system includes the data hiding along with the provisional software with which only the data can be accessed. This system gives privileged access to the administrator (data distributor) as well as the agents registered by the distributors. Only registered agents can access the system. The user accounts can be activated as well as cancelled. The exported file will be accessed only by the system. The agent has given only the permission to access the software and view the data. The data can be copied by our software. If the data is copied to the agent' system the path and agent information will be sent to the distributors email id thereby the identity of the leaked user can be traced[2].

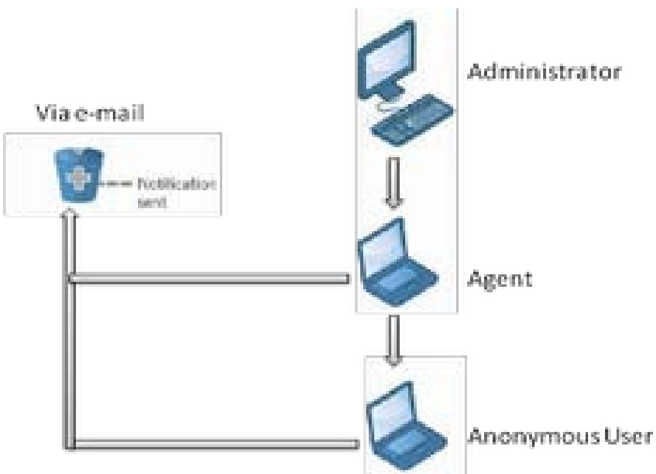

Figure 1: Illustration Diagram

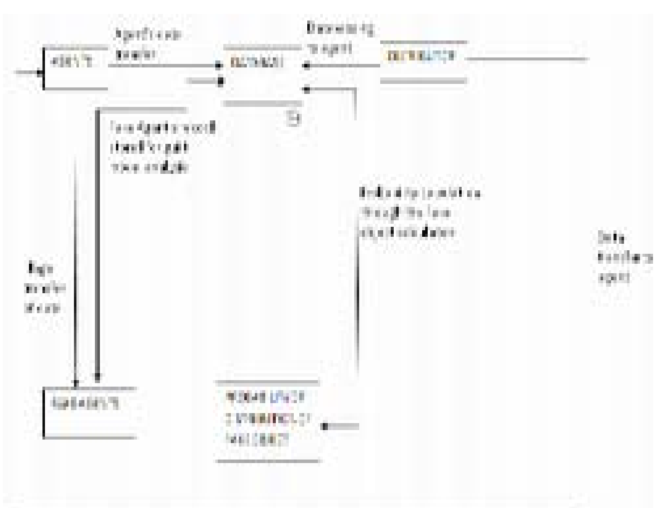

Figure 2: System Architecture Design

b) System Implementation

The implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

i. *Modules*

(1) Data Allocation Module,
(2) Data Distribution Module,
(3) Data Leakage & Detection Module.

ii. *Module Description*

1. *Data Allocation Module*

In this module, administrator has to login with his id and password. Administrator has all the agent information, user data inside his database. Administrator is now able to view the database consisting of the original data as well as the fake data.

Administrator can also list the agents here. He will be able to add additional information to the database. Agent's information can be added here.

2. *Data Distribution Module*

Once the agent has been added by the administrator, he can create one username and password for that particular agent, in fact registering. After the agent has been successfully registered we now want to send the data to agent according to their request. Administrator will now select a requested amount of data and then export these data into an excel file in byte format. After the file is created, the administrator will send the data to agent. Sending the data includes transferring the data through the network (LAN).At the same time the administrator will keep the record of the agent with his id.

3. *Data Leakage and Detection Module*

Agent can login with their given username and password. Now they can view the data that is being sent by the administrator, but they cannot edit nor do any changes with it. He can now copy the data anywhere he wants to. The path and the agent which is copying the file will be recorded and the notification is sent through e-mail. Whenever a guilty agent tries to send the data to any other anonymous user i.e. leaking the data, a notification will be sent through email. The administrator has an email id with all the notifications, including the path to which the data is saved along with agent id[6].

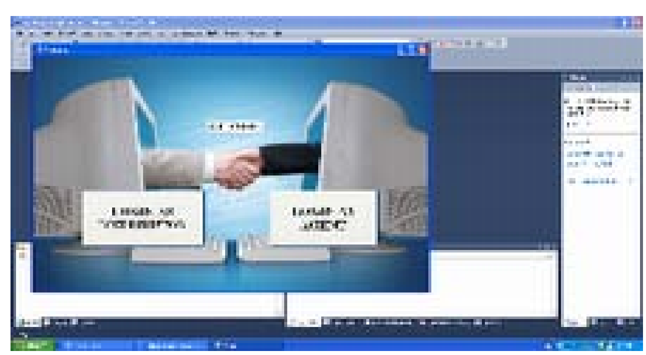

Figure3: Login for Distributor & Agent

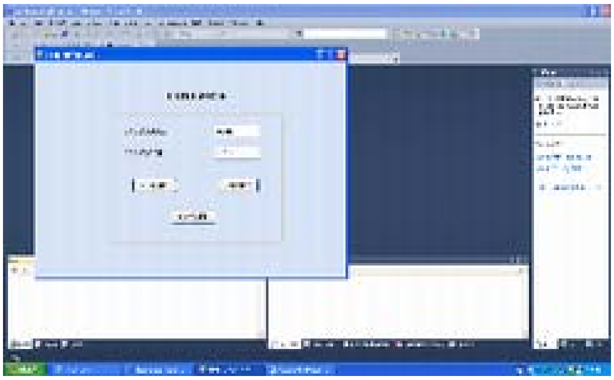

Figure 4: Distributor Login

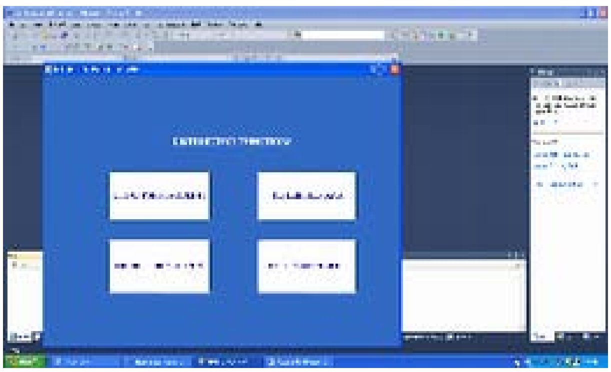

Figure 5: Distributor Function

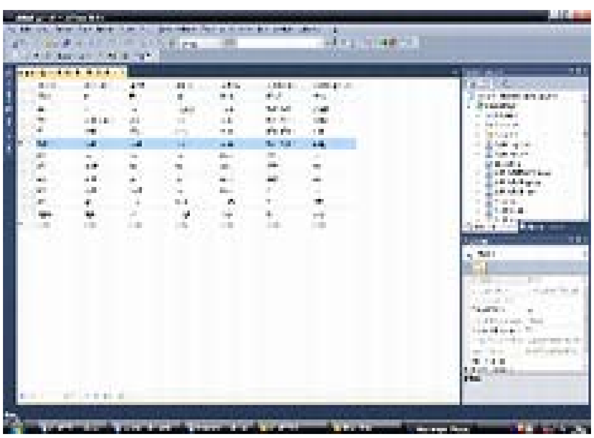

Figure 6: The Agent Detai in Database Table

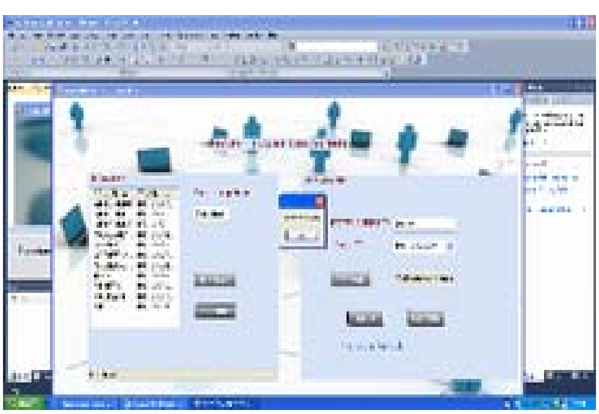

Figure 7: Distributor Sending Data to Agent

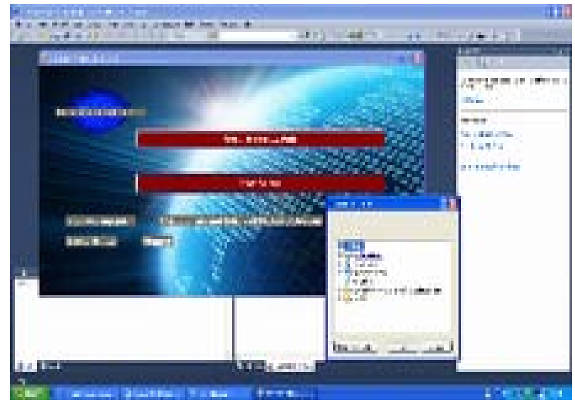

Figure 8: Selection of Agent Side Path

Figure 9: Conformation of Data Reception

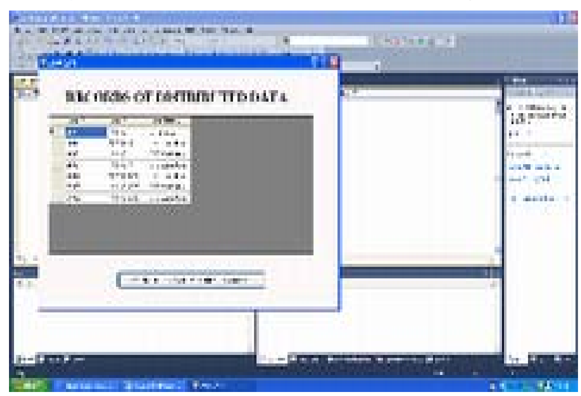

Figure 10: Transfer Data to the Agent is Saved in Record of Distributor Data

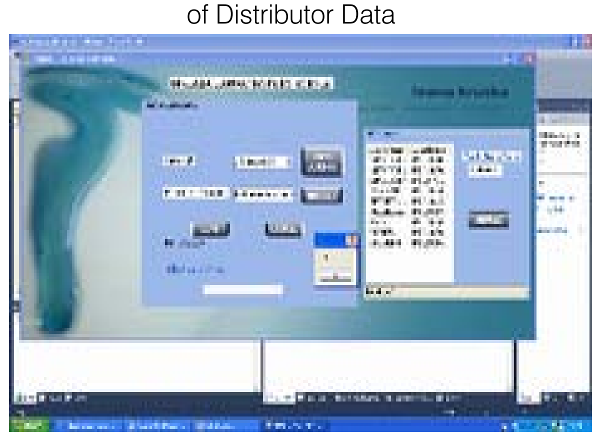

Figure 11: Agent to Agent Data Transfer

Global Journal of Computer Science and Technology (E) Volume XVII Issue IV Version I Year 2017

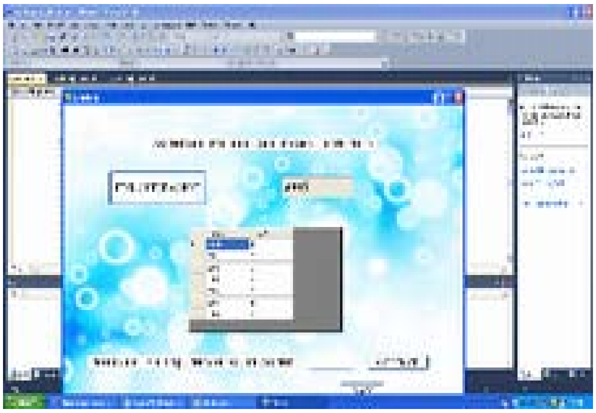

Figure12: Data Leakage can seen in Agent Guilt Model

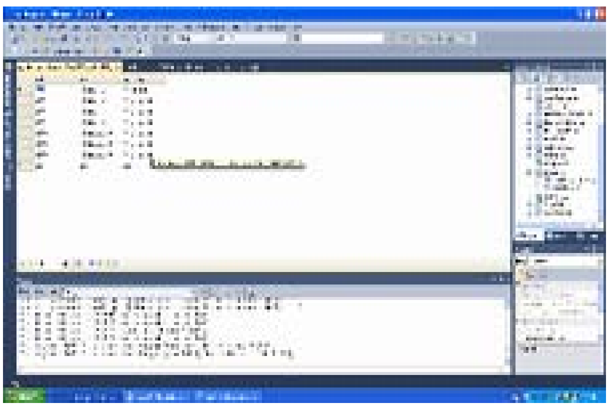

Figure 13: Agent Record

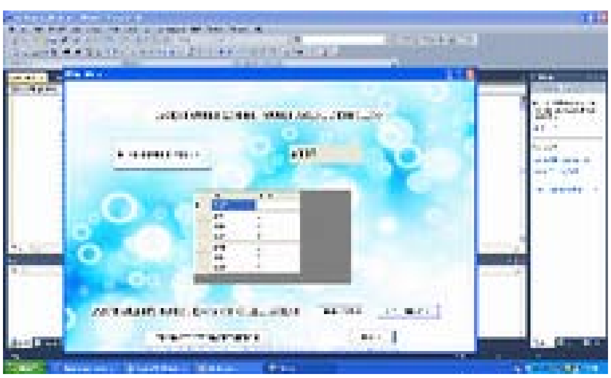

Figure 14: Find Probability of Agent Guilt Model

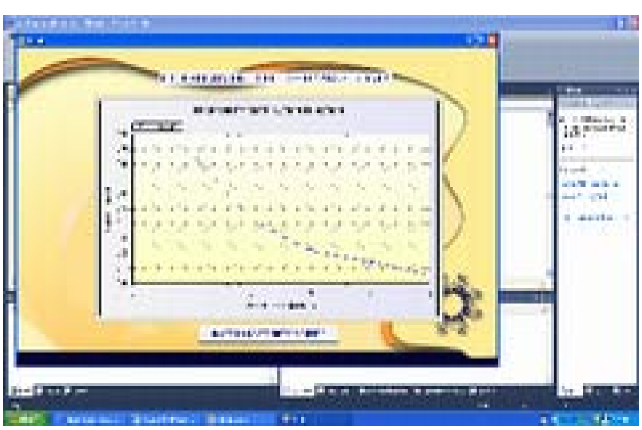

Figure 15: Draw Graph of Guilty Model

VI. Future Work

The notion of a trusted environment is somewhat fluid. The departure of a trusted staff member with access to sensitive information can become a data breach if the staff member retains access to the data subsequent to termination of the trust relationship. In distributed systems, this can also occur with a break down in a web of trust. Most such incidents publicized in the media involve private information on individuals, i.e. social security numbers, etc Loss of corporate information such as trade secrets, sensitive corporate information, details of contracts, etc or of government information is frequently unreported, as there is no compelling reason to do so in the absence of potential damage to private citizens, and the publicity around such an event may be more damaging than the loss of the data itself.

Although such incidents pose the risk of identity theft or other serious consequences, in most cases there is no lasting damage; either the breach in security is remedied before the information is accessed by unscrupulous people, or the thief is only interested in the hardware stolen, not the data it contains. Never the less, when such incidents become publicly known, it is customary for the offending party to attempt to mitigate damages by providing to the victims subscription to a credit reporting agency, for instance.

VII. 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 handover 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.

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 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 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. It includes the investigation of agent guilt models that capture leakage scenarios that are not studied in this paper.

References Références Referencias

1. Data Leakage Detection, an IEEE paper by Panagiotis Papadimitriou, Member, IEEE, Hector Garcia-Molina, Member, IEEE NOV-2010.
2. Watermarking relational databases. In VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases, By R. Agrawal and J. Kiernan, pages 155–166. VLDB Endowment, 2002.
3. An algebra for composing access control policies, By P. Bonatti, S. D. C. di Vimercati and P. Samarati, ACM Trans. Inf. Syst. Secur., 5(1):1–35, 2002.
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, 2001, Proceedings, volume 1973 of Lecture Notes in Computer Science, pages 316–330. Springer, 2001.
5. P. Buneman and W.-C. Tan. Provenance in databases. In SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data, pages 1171–1173, New York, NY, USA, 2007. ACM.
6. Lineage tracing for general data warehouse transformations, By Y. Cui and J. Widom, In The VLDB Journal, pages 471–480, 2001.
7. Digital music distribution and audio watermarking, by S. Czerwinski, R. Fromm, and T. Hodes.