International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 03 | Mar 2019 www.irjet.net p-ISSN: 2395-0072

DATA LEAKAGE DETECTION USING CLOUD COMPUTING

M.Sai Charan Reddy[1] T. Venkata Satya Yaswanth[2] T. Gopal[3] L. Raji[4] Dr. K.Vijaya[5]

[123] *Student, Department of Computer Science, R.M.K. Engineering College*

[4] *Assistant professor, R.M.K. Engineering College*

[5] *Head of Department, Information Technology, R.M.K. Engineering College*

Abstract- *This paper presents the conception of data leakage, its causes of leakage and different techniques to protect and detect the data leakage. The value of the data is incredible, so it should not be leaked or mishandled. In the world of IT, a huge database is being consumed. This database is shared with several people at a time. But during this distribution of the data, there are huge chances of data exposure, insecurity or alteration. So, to nullify these problems, a data leakage detection system has been proposed. This paper includes a curt idea about data leakage detection and a methodology to detect data leakage persons.*

Keywords-Watermarking guilty agent; Explicit data; DLP (Data Leakage Prevention)

1.Introduction

Data leakage is explained as the accidental or unmeant distribution of private or sensitive data to an unlicensed entity. Sensitive data of companies and firms include intellectual property (IP), financial information, patient information, personal credit-card data, and other information hanging on the business and the industry. Besides, in many cases, sensitive data is split among various patrons such as employees working from outside the organizational places (e.g., on laptops), business partners and customers.

In the route of doing business, sometimes sensitive data must be handed over to supposedly trusted third parties. For example, a hospital may give patient records to testers who will formulate new treatments. Identically, a company may have collaboration with other companies that need sharing customer data. Other endeavors may obtain its data processing, so data must be given to diverse companies. We call the owner of the data the distributor and the purposely trusted third parties as the agents. Our motive is to note when the distributor's sensitive data has been leaked by agents, and if possible to identify the agent that leaked the data.

1.1 How Was Ingress To The Data Gained?

The "How was ingress to the data gained?" attribute extends the "Who created the leak?" attribute.

These attributes are not replaceable, but fairly complementary and the various ways to obtain ingress to sensitive data can be clustered into the following groups.

The categorization by leakage channel is important in order to see how the incidents may be stopped in the future and can be classified as physical or logical.

Physical leakage channel is that physical media (e.g., HDD, laptops, workstations, CD/DVD, USB devices) holding vulnerable information or the document itself was moved away from the organization. This more frequently means that the control over data was lost even before it left the organization.

2. Literature Survey

a) Agent Guilt Model

Assume an agent Ui is guilty if it grants one or more objects to the target. The event that agent Ui is guilty for a given leaked set S denoted by Gi| S. Now estimate Pr {Gi| S }, i.e., the probability that agent Gi 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 50 individuals, the person may only discover say 10, leading to an estimate of 0.2. Assume this estimate as pt, the probability that object ' t' can be guessed by the target.

Consider two assumptions regarding the relationship between the various leakage events.

Assumption 1: For all t, t ∈ S such that t ≠ 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 without the help of any of the n agents. To observe 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 calculate the probability that he leaks a single object t to S. To compute this, define the set of agents Vt = {Ui | t<-Rt } that have t in their data sets. Then use Assumption 2 and known probability p, We have the, Pr { agent leaked t to S} = 1- p

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}

Given that agent Ui is guilty if he leaks at least one value to S, compute the probability Pr { Gr| S}, agent Ui is guilty, Pr {Gi| S}

b) Data Allocation Problem

The distributor "intelligently" delivers data to agents to improve the chances of finding a guilty agent. There are four illustrations of this problem, hanging on the type of data requests made by agents and whether "fake objects" are allowed. An agent makes two types of requests, called sample and explicit. Based on the requests the fakes objects are attached to the data list.

Fake objects are objects created by the distributor that are not in set T. The objects are planned to look like real objects and are distributed to agents together with the T objects, in order to increase the rates of detecting agents that leak data.

c) Optimization Problem

The distributor's data assignment to agents has one constraint and one objective. The distributor's curb is to satisfy agents' requests, by giving them the number of objects they request or with all handy objects that satisfy their conditions. His goal is to be able to find an agent who leaks any portion of his data.

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

The Pr {Gi |S =Ri } or simply Pr {Gi |Ri } is the probability that agent Ua is guilty if the distributor discovers a leaked table S that contains all Ri objects. The difference functions Δ (i, j) is defined as:

$$\Delta (i, j) = Pr \{Gi |Ri \} – Pr \{G |Ri \}$$

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 (Over R1,….., Rn,)

The estimation of objective of the above equation does not rely on the agent's probabilities and therefore minimize the relative overlap among the agents as

Minimize (. . . ,(|Ri∩Rj|) / Ri , . . .) i != j (over R1 , . . . ,Rn)

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

3. Allocation Strategies Algorithm

There are two types of strategies algorithms

a) Explicit data Request

In case of the explicit data request with dupes 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 the explicit data request with fake allowed, the distributor cannot remove or alter the requests R from the agent. However, the distributor can add a fake object. In algorithm for data allocation for the 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 receive fake objects. Then create one fake object in iteration and allocate it to the agent selected. The optimal algorithm minimizes every term of the objective summation by adding maximum number bi of fake objects to every set Ri yielding an 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 ←SELECTAGENT(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$^{(1/Ri' - 1/Ri'+1)}$ Σ_j $|R_i \cap R_j|$

Step 3: return i i':: R ∈ R

Sample Data Request

With sample data requests, each agent Ui may receive any T from a subset out of ($^{|T|}_{ni}$) different ones. Hence, there are $\pi_{i=1}^{n}$ ($^{|T|}_{ni}$) 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 $\pi_{i=1}^{n}$ ($^{|T|}_{ni}$)/ |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 smax 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

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'}

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.

4. 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, escape detection is handled by watermarking, e.g., a novel code is embedded in each distributed copy. If that replicate is later discovered within the hands of AN unauthorized party, the leaker can be identified. Watermarks may be terribly helpful 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 could offer patient records to researchers World Health Organization can devise new treatments. Similarly, an organization could have partnerships with alternative corporations that need sharing client knowledge. Another enterprise may

outsource its process, this information should lean to varied different companies. We decide the owner of the info the distributor and therefore the purportedly trusty third parties the agents. The distributor gives the data to the agents. These data will be watermarked. Watermarking is the method of embedding the name or info concerning the corporate. The examples embody the images we've seen within the net. The authors of the pictures are watermarked within it. If anyone tries to repeat the image or knowledge the watermark are going to be present. And thus the data may be unusable by the leakers.

5. Proposed System

We propose data allocation strategies (across the agents) that improve the chance 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 any improve our chances of detecting leakage and identifying the guilty party. We also present an algorithm for distributing the object to an agent.

Our goal is to detect when the distributor's sensitive information has been leaked by agents, and if possible to spot the agent that leaked the information. Perturbation may be a terribly helpful technique wherever the information 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 have a tendency to develop a model for assessing the 'guilt´ of agents. We also present algorithms for distributing objects to agents, in an exceedingly means 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 however seem realistic to the agents. In a sense, the faux objects act as a kind of watermark for the entire set, without modifying any individual members. If it turns out an agent was given one or a lot of faux objects that were leaked, then the distributor is a lot of assured 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 information and makes the data as original.

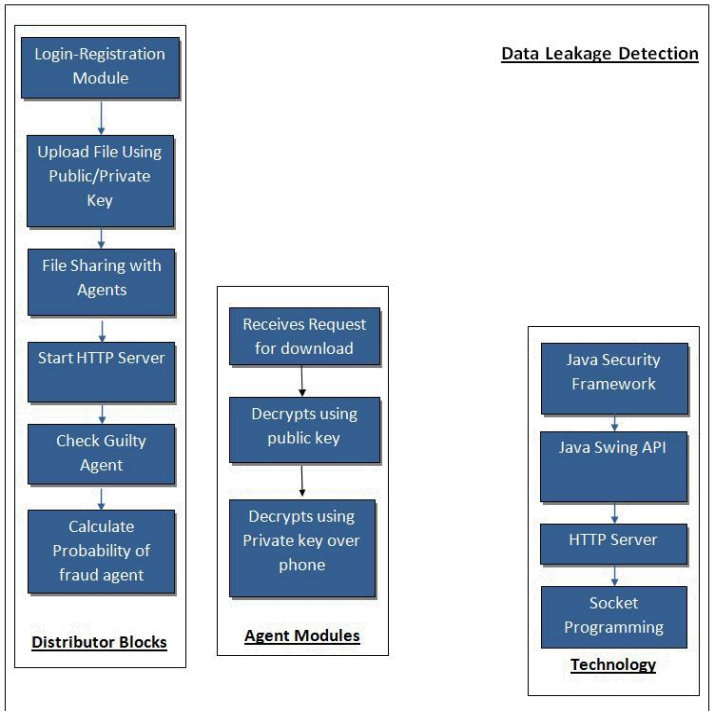

Fig-1: Data Leakage Detection

6. Conclusion

In an excellent world, there would be no ought to hand over sensitive data to agents that may unknowingly or maliciously leak it. And even if we had to hand over sensitive knowledge, in an exceedingly excellent world we tend to may watermark every object so we tend to may trace its origins with absolute certainty. However, in many cases, we must so work with agents which will not be 100% trusted. In spite of these difficulties, we have shown it is possible to assess the chance that AN agent is responsible for a leak, based on the overlap of his data with the leaked knowledge and therefore the knowledge of alternative agents, and supported the chance that objects are often 'guessed´ by other means. Our model is relatively

simple, however, we tend to believe it captures the essential tradeoffs. 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 wherever there's giant overlap within 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

[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.

[8] Data Leakage Detection by Rajesh Kumar