IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

DATA LEAKAGE DETECTION SYSTEM IN DATA MINING

1M.Akhil Kumar Reddy, 2Narendra.B, 3Ravi Varama Moru, 4V.Sai Surya Teja, 5Pabbathi Dileep

1Student, 2Student, 3Student, 4Student, 5Student

1SRM Institute Of Science and Technology,

2SRM Institute Of Science and Technology,

3SRM Institute Of Science and Technology,

4SRM Institute Of Science and Technology,

5SRM Institute Of Science and Technology

ABSTRACT:

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.

Key words: data leakage, data mining, problem setup and notations, guilt agents

I. INTRODUCTION:

Data leakage is an uncontrolled or unauthorized transmission of classified information to the outside. It poses a serious problem to companies as the cost of incidents continues to increase. Many software solutions were developed to provide data protection. However, data leakage detection systems cannot provide absolute protection. Thus, it is essential to discover data leakage as soon as possible.

Data leakage can be defined as an event in which classified information,

e.g., sensitive, protected or confidential data has been viewed, stolen or used by somebody who is not authorized to do so. Data leakage causes serious and expensive problems to companies and organizations, because the number of events continues to rise. Data leak prevention helps ensure that confidential data like customer information, personal employee information, trade secrets, financial data and research and development data remains safe and secure. Data leak prevention solutions prevent confidential data by securing the data itself. Once most critical data and its location are identified on the network, it can be monitored to determine who is accessing and using it; where it is being sent, copied, or transmitted. Several methods and systems have been developed to prevent data leakage.

2: SOFTWARE REQUIREMENT SPECIFICATIONS

2.1 .GENERAL DESCRIPTION:

In this, general functions of the product which includes the objective of the user, a user characteristic, features, benefits, about why its importance is mentioned. It also describes features of the user community.

2.2.FUNCTIONAL REQUIREMENTS :

In this, the possible outcome of a software system which includes effects due to operation of the program is fully explained. All functional requirements which may include calculations, data processing, etc. are placed in a ranked order.

2.3. INTERFACE REQUIREMENTS:

In this, software interfaces which mean how software programs communicate with each other or users either in the form of any language, code, or message are fully described and explained. Examples can be shared memory, data streams, etc.

2.4. NON FUNCTIONAL ATTRIBUTES :

In this, non-functional attributes are explained that are required by the software system for better performance. An example may include Security, Portability, Reliability, Reusability, Application compatibility, Data integrity, Scalability capacity, etc.

2.5.SOFTWARE REQUIREMENTS:

- The module is written in ASP .net and C# .net.
- It is developed in Visual Basics Platform.
- Windows is the operating system chosen for the module.
- The database used in the project is MS-SQL server 2005 or higher.

2.6. HARDWARE REQUIREMENTS :

- PROCESSOR: Pentium 4 or above.
- RAM: 256 MB or more.
- Hard disc Space: 500 MB to I GB

3: MODULE ANALYSIS

3.1 PROBLEM SETUP AND NOTATION:

A distributor owns a set T = {t1 . . . tm} of valuable data objects. The distributor wants to share some of the objects with a set of agents U1; U2; . . . ; Un, 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. An agent Ui receives a subset of objects Ri_ T, determined either by a sample request or an explicit request: Sample request Ri = SAMPLE (T, mi): Any subset of records from T can be given to Ui. Explicit request Ri=EXPLICIT (T, $cond_i$): Agent Ui receives all T objects that satisfy condi.

3.2 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 has 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.

3.3 IMPLEMENTATION METHODS :

3.3.1 DATA ALLOCATION:

The main focus of this paper is the data allocation problem: How can the distributor "intelligently" give data to agents in order to improve the chances of detecting a guilty agent? As illustrated in Fig. 1, 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.

3.3.2 FAKE OBJECT:

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, 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 than perturbing real objects. Our use of fake objects is inspired by the use of "trace" records in mailing lists.

3.3.3 OPTIMIZATION:

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 as and may not provide agents with different perturbed versions of the same objects. We consider fake object distribution as the only possible constraint relaxation.

3.3.4 ALLOCATION STRATAGIES

EXPLICIT DATA REQUEST: In the first place, the goal of these experiments was to see whether fake objects in the distributed data sets yield significant improvement in our chances of detecting a guilty agent. In the second place, we wanted to evaluate our e-optimal algorithm relative to a random allocation.

ALGORITHM:

1: R ←∅ Agents that can receive fake objects

2: for i = 1, . . . ,n do

3: if bi > 0 then

4: R ← R∪ {i}

5:Fi ←∅

6: while B> 0 do

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

SAMPLE DATA REQUEST :

With sample data requests agents are not interested in particular objects. Hence, object sharing is not explicitly defined by their requests. The distributor is "forced" to allocate certain objects to multiple agents only if the number of requested objects exceeds the number of objects in set T. The more data objects the agents request in total, the more recipients on average an object has; and the more objects are shared among different agents, the more difficult it is to detect a guilty agent.

ALGORITHM:

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

2:R1 ←∅, . . . ,Rn←∅

3:remaining ←

4:while remaining > 0 do

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

6:k ← SELECTOBJECT (I , Ri) May also use additional

Parameters7:Ri ← Ri∪ {t k}

8:a[k] ← a[k] + 1

9:remaining ← remaining − 1

4: SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner.

There are various types of tests. Each test type addresses a specific testing requirement.

4.1 :UNIT TESTING:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach:

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed . All links should take the user to the correct page

4.2 INTEGRATION TESTING:

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or — one step up — software applications at the company level — interact without error.

Test Results - All the test cases mentioned above passed successfully. No defects encountered.

4.3 ACCEPTANCE TEST:

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

5.CONCLUSION:

In a perfect world, there would be no need to hand over sensitive data to agents that mayun knowingly 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 guiltyagents, especially in cases where there is large overlap in the data that agents must receive.

6.REFERENCES:

[1]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.

[2]P. Bonatti, S.D.C. di Vimercati, and P. Samarati, "An Algebra for Composing Access

Control Policies," ACM Trans. Information and System Security, vol. 5, no. 1, pp. 1-35,2002.

[3]P. Bune man, S. Khanna, and W.C. Tan, "Why and Where: A Characterization of Data Provenance," Proc. Eighth Int'l Conf. Database Theory (ICDT '01), J.V. den Bussche and V. Vi anu, eds .,pp. 316-330, Jan. 2001.

[4]P. Bune man and W.-C. Tan, "Provenance in Databases," Proc. ACM SIGMOD, pp.1171-1173, 2007.

[4]Y. Cui and J. Widom, "Lineage Tracing for General Data Warehouse Transformations,"

The VLDB J., vol. 12, pp. 41-58, 2003.

[5]S. Czerwinski, R. Fromm, and T. Hodes, "Digital Music Distributionand Audio Watermarking," http://www.scientificcommons.org/43025658, 2007.

[6]F. Guo, J. Wang, Z. Zhang, X. Ye, and D. Li, "An Improved Algorithm to Watermark Numeric Relational Data," Information Security Applications, pp. 138-149, Springer,2006.

[7]F. Hartung and B. Girod, "Watermarking of Uncompressed and Compressed Video," Signal Processing, vol. 66, no. 3, pp. 283-301, 1998.

[8]S. Jajodia, P. Samarati, M.L. Sapino, and V.S. Subrahmanin,"Flexible Support for Multiple Access Control Policies," ACMTrans. Database Systems, vol. 26, no. 2, pp.

214-260, 2001.

[9]Y. Li, V. Swarup, and S. Jajodia, "Fingerprinting Relational Databases: Schemes and Specialties," IEEE Trans. Dependable and Secure Computing, vol. 2, no. 1, pp. 34-45,Jan.-Mar. 2005.