2012 Fourth International Conference on Intelligent Networking and Collaborative Systems

Inference Detection and Database Security for a Business Environment

Traian Popeea, Anca Constantinescu, Laura Gheorghe, Nicolae Țăpuș
"Politehnica" University of Bucharest
Bucharest, Romania
{traian.popeea, anca.constantinescu, laura.gheorghe, ntapus}@cs.pub.ro

Abstract— **The number of data collections of person-specific information is increasing exponentially. The risks of compromising privacy are also increasing, but can be limited through data anonymity and data security. Security models, developed for databases, differ in many aspects because they focus on different features of the problem, leading to incomplete implementations of the organizational security strategy. Our paper presents a multi-layer approach to data anonymity and database security, covering the entire process from inference detection to secure communication and database integrity. This approach is useful for protecting sensitive data and assuring in-depth mitigation of different possible attacks.**

Keywords- data anonymity, database security, operating system security, secure communication

I. INTRODUCTION

Data collections stored in data warehouses may contain large quantities of sensitive information. Therefore, data privacy is a very important aspect of data publishing. Sensitive data, such as name, address, social security number, or other personal attributes represents a valuable source of information for providers of personalized services, ranging from medical services to targeted commercials and marketing messages [1].

The main method for protecting data privacy is data anonymity. The use of anonymous data can improve the degree of privacy provided by data warehouses. One method of generating anonymous data is data re-identification, through static manipulation of stored data, generating common values for some records, and replacing the original values with the common ones. However, data re-identification cannot be over-used, as it may lead to decay in the statistical relevance of the stored information, and, in the end, to a replicated entry representing the statistical average of the data located in the warehouse.

A second method of data anonymization consists in the use of inference detection mechanisms. Inference is a method to subvert access control in database systems. An inference occurs when a user is able to infer some data without directly accessing them. A simple way to monitor user accesses is to examine each user query, and reject any query that accesses sensitive data. For example, an attempt to obtain the average salary of all the employees of a company can be considered a legitimate query, but an attempt to obtain the salary of the CEO of the same company should not be allowed. However, it is possible for a user to use a series of unsuspicious queries to infer data in the database [2].

Database security is a growing concern, as evidenced by an increase in the number of reported incidents of loss of or unauthorized exposure to sensitive data. With the expansion of data shared electronically, a better understanding of database security is required.

Similar to all other fields of security, database security uses authentication, authorization, and accounting to ensure that only authenticated users perform authorized activities at authorized points in time. Database security includes many layers of security, that can be classified in topics such as physical security, network security, encryption, and authentication.

Database security is based on three main premises: confidentiality, integrity, and availability. Confidentiality represents the protection of data from unauthorized disclosure, integrity refers to the prevention of unauthorized data access, while availability is achieved through identification of and recovery from hardware and software errors or malicious activity resulting in the denial of data availability [3].

Although the goal may seem simple, in reality, the path to achieving data security is very complex. Traditionally, database security focused on user authentication and managing user privileges to database objects [4]. This has proven to be inadequate, given the growing number of successful database attacks and the increase in the number of organizations reporting loss of sensitive data.

In order to enhance database security, a multi-layer approach is required. Not only that the database must be secured, but also the underlying host machine, its operating system, and the communication with the clients. The best way to secure an operating system is to use a Mandatory Access Control (MAC) layer, that, properly configured, disallows any unauthorized access to the database files. In a Discretionary Access Control-based system, a malicious system user may give access rights to another user with a lower security clearance, while in a MAC-based system, this scenario is impossible, as all access rights are controlled by the administrator. Securing communication can be performed through the use of a SSL layer, that provides data encryption and authentication via certificates.

978-0-7695-4808-1/12 $26.00 © 2012 IEEE
DOI 10.1109/iNCoS.2012.95

The remainder of this paper is organized as follows. The paper describes the data anonymity perspective in Section II and related work in Section III, followed by architecture and implementation in Section IV. Section V presents the experimental results obtain after the practical evaluation of the implemented security design. Finally, Section VI presents concluding remarks and outlines future work.

II. DATA ANONYMITY

With the current research and technology advances in supercomputing, commodity clusters, network connectivity and storage space, the exponential growth of data collections of sensitive, person-specific information is inevitable. Data holders are releasing information for research or profit, but they are facing the delicate problem of releasing only information that does not compromise privacy, confidentiality or individual and organizational interests. If the information released is not properly protected, it may harm the public or others, while not releasing such information at all may diminish the need for the data. Thus, the data holders' ability to produce anonymous data is the key factor in the survivability of the database itself.

Most explicit identifiers, such as name, address, social security number, or telephone number, are removed when sensitive, person-specific data is released to the public, considering that anonymity is maintained because the data seems anonymous. It contains, however, unique characteristics that may be used to re-identify individuals.

One of the most extensive data re-identification experiments was conducted using 1990 U.S. Census summary data. The goal was to determine how many persons from a specific population had infrequent combinations of demographic values [5] as combinations of few non-sensitive characteristics often combine in populations to uniquely identify a great majority of individuals. For example, a stunning discovery of the experiment was that 216 million of 248 million (around 87%) of the population of the United States, at that time, could be uniquely identified based on the combination of gender, date of birth and 5-digit ZIP code.

Although health records and other private data are publicly available, containing such information about individuals, these types of data sets should not be considered anonymous. The problem of data anonymization when releasing a version of privately held data has been a main concern for many years.

Another area of interest is aggregation and inference in databases that concerns restricting the release of lower classified information such that higher classified information cannot be derived. Functional and multi-valued dependencies determine many inferences. The problem of precisely eliminating all possible inference is NP-complete.

As with any NP-complete problem, it is impossible to guarantee the precise and complete elimination of all inferences regarding the identities of the individuals whose person-specific information is included in the data set. [6] As with every aspect of security, the data holder cannot consider every possible attack, and attackers are very creative. Also, other available knowledge can be linked to the released data, leading to more possible inferences.

Computer security principles enforce that only the recipient of information be allowed to receive that information, through access control and authentication. Although they protect against direct disclosure of data, they do not address disclosures based on drawn inferences. [7]

The initial problem, the accessibility of certain information to a recipient, possibly an attacker, becomes a new problem: what information should that recipient receive. The new general approach is shifting towards public releases, accessible even to malicious entities. However, the anonymity of the people who are subjects of the data should be protected. [8, 9]

III. RELATED WORK

K-anonymity [10] is one of the most important concepts in data anonymity through re-identification. Let PT be a private table, containing sensitive data, that has to be released from the data warehouse. K-anonymity demands that for every tuple in the private table released, its sensitive information be identical with at least k other tuples.

Although there are many data sets available for linking to external attackers, k-anonymity does not make any assumptions regarding them. The sheer quantity of information would make that task impossible. All the requirements of k-anonymity apply to the released table only. In this table, all individuals must be indistinguishable (within a given set) with respect to the released set of attributes. It can, therefore, be used as a static method for increasing data anonymity.

There are many techniques proposed for providing anonymity in the release of data, but sampling, swapping values, or adding noise to the data decrease the quality of individual tuples. The k-anonymity concept is focused on two techniques: generalization and suppression.

Generalization consists in substituting the values of a given attribute with more general values, while suppression is used to "moderate" the generalization process when a limited number of outliers (i.e., tuples with less than k occurrences) would force a great amount of generalization.

Inference detection represents a dynamic mean of protecting sensitive data, using query logs preventing data leaks at runtime. Chen and Chu [11] have developed an inference violation detection system to protect sensitive data content. Based on data dependency, database schema and semantic knowledge, a semantic inference model is constructed.This model represents the possible inference channels from any attribute to the pre-assigned sensitive attributes. The model is then instantiated to a semantic inference graph for query-time inference violation detection.

For a single user case, when a user poses a query, the detection system will examine the user's query log and determine the probability of inferring sensitive information. The query request will be denied if the inference probability exceeds a certain specified threshold.

For multi-user cases, where the users may share their query answers to increase the inference probability, the authors of [11] have developed a model to evaluate collaborative

inference based on the query sequences of collaborators and their task-sensitive collaboration levels.

IV. Architecture and Implementation

In this paper we propose a multi-layer approach to database anonymity and database security in a data warehouse containing information about current and past employees for a large company.

From an anonymity perspective, we have developed an engine that provides both static and dynamic means of securing sensitive data. The static anonymity is used for personal data, and filters out the direct identifiers, such as name, address or social security numbers, while applying re-identification for the other information, such as gender, age, or city.

The re-identification algorithm is an implementation of k-anonymity, and is triggered after every change of the data set (insertion, deletion or update). The k factor is dynamically decided based on the number of records existing in the data warehouse.

The dynamic anonymity engine applies to all the other records of the database schema, such as the salary log of employees or managers, or the history of managers for a specific department.

As stated in [12], inferences can be classified into six main categories: split queries, subsume inferences, unique characteristic inferences, overlapping inferences, complementary inferences, and functional dependency inferences.

Split queries appear when a query can be split into two smaller inferred queries with respect to another query. A query Q_i can be split into two smaller queries when the user can identify the return tuples of Q_i that relate to some other query. For example, if a query requesting the salaries of all the employees aged 40 returns two values, and a query requesting the age of all the managers returns a unique record with the age of 40, one can determine the salary of the manager, as it is usually the bigger one.

Subsume inferences can occur when all returned records of a query share a common attribute that can be used to infer on the returned records of another query. If one knows the age of a certain employee, for example Alice, and all employees of that age have the same salary, then the salary of Alice is revealed.

The unique characteristic inference appears when all but one record of a query can be obtained through another legitimate query. For example, let there be only one manager older than 50, a query that requests the pair (age, salary) for all employees that are older than 50, and a second query that requests the same pair for all non-manager employees that are older than 50. If the second returned set is identical to the first, except for a missing record, then the user can infer the age and salary of a manager.

The overlapping inferences occur when a non-sensitive information can be obtained through a restrictive query, while another sensitive information can be obtained through several, more general queries, that are overlapping. The premises for the overlapping queries are: a user can obtain the salaries of all the accountants in the company, the salaries of all the employees aged 30, and the intersection of the two returned sets contains exactly one value. By submitting a more restrictive query (the name of all the accountants aged 30), the user can determine, the name, salary and age of a specific accountant.

The complementary inferences are obtained negating the overlapping inferences, while the functional dependency inferences use functional dependencies existing in the database schema.

The implementation of our anonymity engine is based on Java, that offers a thin layer of connectivity with the database, while allowing manipulation of entity objects and logging. From a security stand point, we have used a multi-layer security system, also known as defense in depth, describing the practice of combining multiple mitigating security controls to protect resources and data.

A complete multi-layer database security should include physical layer security, data-link layer security, secure communication, a secure operating system, a secure database, information integrity and anonymity by protecting from inference-based attacks [13]. However, in this paper we focused only on securing the communication, the operating system and the database, because physical and data-link security cannot be assured only from the server side, involving the entire network infrastructure, in which the server is placed.

In an unsecure database system all clients are able to access all data from the database server in an unrestricted manner. All hosts are allowed to connect to the server from any IP address and create a connection in the database server's Connection Pool, thus accessing all the data available in the Storage Engine.

In order to achieve the proposed security system, we have chosen an appropriate operating system, and enhanced it with a MAC module. Also, we chose a database management system (DBMS) that supports authentication, authorization, accountability and integrity, considering encryption as a requirement for the DBMS. Last but not least, a method to secure communication between the server and the clients is mandatory. The final architectural design of our project is presented in Figure 1.

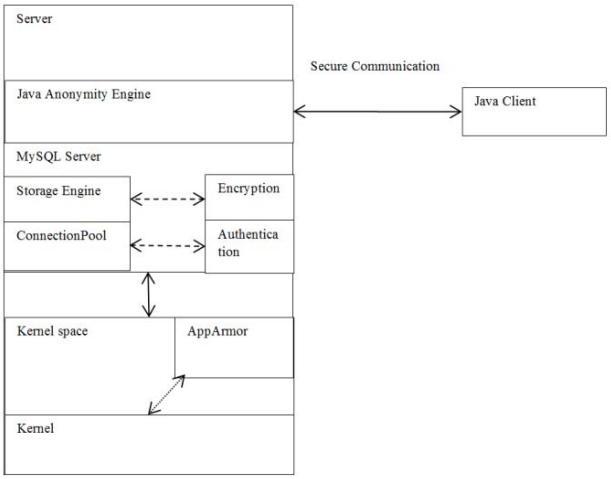

Figure 1. Architectural design

From a communication point of view, our enhancements include secure, encrypted communication that uses a certificate system for authentication and authorization. This type of communication prevents any type of man-in-the-middle attacks, or information sniffing. The communication uses the Java layer we have added for the anonymity engine.

The chosen operating system is Ubuntu, which can be secured with a mandatory access control layer (AppArmor) that prevents users and processes from accessing resources for which they are not authorized. This helps maintaining operating system integrity and reliability by limiting the risk of privilege escalation attacks.

The database server we use is MySQL, configured such as not to allow any connections except from our Java layer, using the MySQL authentication module based on a grant system, that requires explicit access to information. Also, we have attempted to encrypt the data from the storage engine, but is has added a severe overhead, as is shown in Section 5.

This entire back-end system sustains a database schema for a large company, that stores information related to both current and previous activity. The complete schema can be found in Figure 2.

The employee table stores information related to current and previous employees, such as date of birth, name, surname, gender, and hiring date. The employees are identified through an unique number.

The departments table stores department names and unique identifiers.

The dept_manager table stores information regarding managerial history. It has a list of current and previous managers, using the employee ID, the department ID, and the period of time while the managerial relationship was active.

The dept_emp table is similar to the dept_manager one, only that it stores information regarding the membership of an employee to a department.

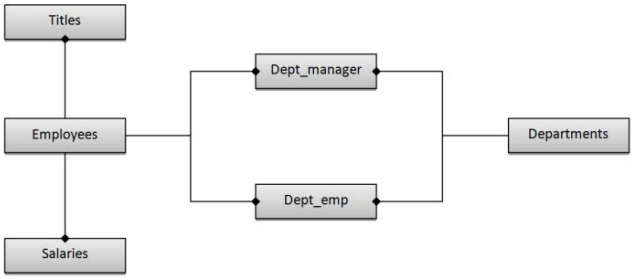

Figure 2. Database schema

The salaries and titles tables refer to the history of salaries obtained and titles held, respectively, by an employee in a certain period. The identifier is the combination of employee ID and the start date of the specific period.

The entire database contains more than 4 million records, which were synthetically generated, and can be retrieved from [14].

V. Experimental Results

The tests we have performed have consisted in choosing a specific security level, running multiple tests, and extracting significant results.

In order to observe the performance penalty induced by any security level, we have run tests while using six levels of security.

The baseline test was run on an unsecure database, placed on an unsecure operating system, using plain-text communication (Unsecure). Over the Unsecure test, we have, subsequently placed different layers of security: mandatory access control (MAC), secure communication (COM) and database storage encryption (ENC) in order to observe the overhead introduced, independently, by each security layer. The final two tests involved two (MAC + COM) and three (MAC + COM + ENC) simultaneous security layers, to determine, the overhead added by their interactions.

The tests we have run consisted in dropping any existing database and then creating a new database with the required tables and constraints. The next step consisted in inserting the millions of records in their specific tables, and, finally, we have checked the integrity of data, using pre-computed CRC values.

The testing infrastructure consisted of two virtual machines, through VMware technology. The server used an Ubuntu 11.10 32bit operating system, with Linux kernel 3.0. The resources granted to the virtual machine were 2 CPU cores, and 4 GB of RAM. The client was a Debian 5 Linux with 2.6 kernel and 1 GB of RAM.

The virtual infrastructure had the following physical system configuration underneath: a GenuineIntel Intel(R) Core(TM) i5-2500K CPU @ 3.30GHz with 4 processors, 8 GB of DDR3 RAM, Microsoft Windows 7 Ultimate x64 Edition, Service Pack 1, (06.01.7601.00) operating system.

On the server we have installed MySQL Server 5.5 and patched both the database server and the operating system. For

the unsecure database, we have uninstalled AppArmor. For the COM test, we have used SSL communication via the Java anonymity engine, using X.509 certificates. Finally, for the encryption tests, we had to modify our testing scripts, because MySQL does not support natively data encryption, and the encryption function had to be called explicitly from the client. The encryption algorithm used was AES with a key length of 128 bits.

For every level of security we have run the tests ten times, without any supplemental work on the physical machine. We have measured the time of each SQL query and the total run time for every test. The final results can be seen in Table 1, Table 2, and Figure 3.

Due to the input-output nature of the database server, we can notice that the mandatory access control layer and the secure communication add a negligible overhead, of less than 10%, because the added amount of security checks are performed simultaneous with other database tasks. On the other hand, the encryption level has an overhead of over 100%, due to intense computation on the server side. As a result, we have chosen not to use storage encryption for our infrastructure.

TABLE I. DATABASE SYSTEM TEST RESULTS (S)

Test type	Time (s)		
	Total time	*Total SQL time*	*Communication and OS time*
Unsecure	76.11	56.74	19.37
MAC	79.10	58.90	20.20
COM	79.74	58.10	21.64
ENC	155.87	130.12	25.75
M+C	80.14	59.02	21.12
M+C+E	160.39	136.04	24.35

TABLE II. DATABASE SYSTEM TEST RESULTS (S)

Test type	Time (s)			
	Database creation	*Insert queries*	*Integrity checks*	*Total SQL time*
Unsecure	0.05	44.01	12.68	56.74
MAC	0.08	45.90	12.92	58.90
COM	0.08	45.57	12.45	58.10
ENC	0.11	103.26	26.75	130.12
M+C	0.09	45.86	13.07	59.02
M+C+E	0.12	108.79	27.13	136.04

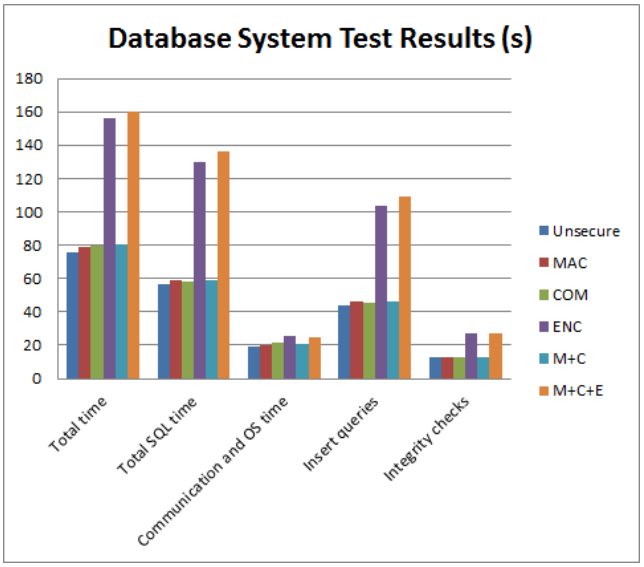

Figure 3. Test results

Also, we have performed penetration tests under the same conditions, using tools such as BackTrack, Nessus and Metasploit. Their goal was to reveal the vulnerabilities that were fixed at every level. A total number of six major vulnerabilities have been found. Four of those vulnerabilities were related to MySQL, while two of them were related to Ubuntu 11.10.

After patching and applying mandatory access control, all of the major vulnerabilities have been fixed. The four MySQL vulnerabilities fixed after applying the database patches were:

CVE-2012-1703 [15]: "Unspecified vulnerability in the MySQL Server component in Oracle MySQL 5.1.61 and earlier, and 5.5.21 and earlier, allows remote authenticated users to affect availability via unknown vectors related to Server Optimizer", having no impact to the confidentiality or integrity of the system, but can cause a total shutdown of the affected resource, leading to unavailability.

CVE-2012-0113 [16]: "Unspecified vulnerability in the MySQL Server component in Oracle MySQL 5.1.x and 5.5.x allows remote authenticated users to affect confidentiality and availability via unknown vectors", leading to partial informational disclosure and reduced performance or interruptions in resource availability.

CVE-2011-2262 [17]: "Unspecified vulnerability in the MySQL Server component in Oracle MySQL 5.1.x and 5.5.x allows remote attackers to affect availability via unknown vectors" leading to reduced performance or interruptions in resource availability.

CVE-2012-0486 [18]: "Unspecified vulnerability in the MySQL Server component in Oracle MySQL 5.5.x allows remote authenticated users to affect availability via unknown vectors" leading to reduced performance or interruptions in resource availability.

Two Ubuntu vulnerabilities that were fixed after applying operating system patches and activating AppArmor.

CVE-2011-4405 [19]: “The cupshelpers scripts in system-config-printer in Ubuntu 11.04 and 11.10, as used by the automatic printer driver download service, uses an "insecure connection" for queries to the OpenPrinting database, which allows remote attackers to execute arbitrary code via a man-in-the-middle (MITM) attack that modifies packages or repositories” leading to considerable informational disclosure, modification of some system files or information, without the attacker having control over what can be modified, and reduced performance or interruptions in resource availability.

CVE-2011-3150 [20]: “Software Center in Ubuntu 11.10, 11.04 10.10 does not properly validate server certificates, which allows remote attackers to execute arbitrary code or obtain sensitive information via a man-in-the-middle (MITM) attack” leading to considerable informational disclosure, modification of some system files or information, without the attacker having control over what can be modified, and reduced performance or interruptions in resource availability.

The added security layers have solved all of the critical vulnerabilities detected, vulnerabilities which could lead to unauthorized access or denial of service.

VI. Conclusions and Future Work

The architecture we propose in this paper offers a multi-layer approach to database anonymity and security. Although a complete multi-layer database security should include numerous layers of security, beginning with physical layer, due to the virtual environment we have used we have focused only on three aspects: securing the communication, the operating system and the database.

In order to achieve the proposed security system, we have studied the existing options for operating systems, mandatory access control layer, and secure communication, choosing an appropriate operating system, Ubuntu, and enhancing it with a MAC module, AppArmor. Also, we have chosen an open-source database management system, MySQL, which supports authentication, authorization, accountability and integrity, although the encryption layer had to be added manually. Last but not least, we have chosen to use our own method to secure communication, using SSL communication through our server, rather than the native SSL of the MySQL server.

We have selected a testing framework, based on a powerful hardware infrastructure and a stable virtual network, that did not limit the software capabilities, and we have determined the penalty introduced by each layer of security. We can conclude that applying secure communication and a MAC layer to a database server does not involve a significant penalty, while data integrity through encryption on the server side generates an overhead that cannot be ignored, and is thus considered unnecessary in the framework of this paper.

We intend to further develop our data anonymity system, because it has not been completed yet. We hope that we will soon succeed to prevent as many inference-based attacks on any sensitive data stored on the server as possible, with a satisfactory level of false positives.

Acknowledgements

This work has been partially funded by the European Commission under grant agreement FP7-ICT-258280 TWISNet project and partially by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labour, Family and Social Protection through the Financial Agreement POSDRU/88/1.5/S/60203.

References

[1] S. Kiyomoto, K Fukushima, Y Miyake, "Data Anonymity in Multi-Party Service Model", Security Technology: International Conference, SecTech 2011, Korea, pp 21-30, December 2011

[2] R. Yip , K. Levitt, "Data Level Inference Detection in Database Systems", Proceedings of the 11th IEEE Computer Security Foundations Workshop, pp 179-189, June 1998

[3] M. Murray, "Database Security: What Students Need to Know", Journal of Information Technology Education: Innovations in Practice, Volume 9, 2010

[4] M. Guimaraes, "New challenges in teaching database security", Proceedings of the 3rd Annual Conference on Information Security Curriculum Development, Kennesaw, USA, pp 64-67, 2006

[5] L. Sweeney, “Uniqueness of Simple Demographics in the U.S. Population”, Carnegie Mellon University, Laboratory for International Data Privacy, Pittsburgh, USA, 2000

[6] Z. Yang, S. Zhong, R. Wright, “Anonymity-Preserving Data Collection”, In KDD ’05: Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, pp 334-343, 2005

[7] Y. Pan, “Research on network database encryption technology”, Proceedings of 2011 IEEE 3rd International Conference on Communication Software and Networks (ICCSN), pp 690-693, 2011

[8] S. Sesay, Z. Yang, J. Chen, D. Xu, “A secure database encryption scheme”, Proceedings of 2005 Second IEEE Consumer Communications and Networking Conference, 2005, pp 49-53, January 2005

[9] L. Willenborg, T. De Waal, “Statistical Disclosure Control in Practice”, Springer-Verlag, 1996

[10] L. Sweeney, "Achieving k-anonymity privacy protection using generalization and suppression" , International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, pp 571-588, 2002

[11] Y. Chen, W. Chu, “Database Security Protection via Inference Detection”, Intelligence and Security Informatics, SCI 135, pp. 275–303, Springer-Verlag Berlin Heidelberg, 2008,

[12] R. Yip, "A Data Level Database Inference Detection System", Doctoral Dissertation, University of California, USA, 1998

[13] G. Canfora, C. A. Visaggio, “Tuning anonymity level for assuring high data quality: an empirical study”, ESEM 2007

[14] https://launchpad.net/test-db, retrieved on December 2011

[15] http://www.cvedetails.com/cve/CVE-2012-1703/, retrieved on 20 June 2012

[16] http://www.cvedetails.com/cve/CVE-2012-0113/, retrieved on 20 June 2012

[17] http://www.cvedetails.com/cve/CVE-2011-2262/, retrieved on 20 June 2012

[18] http://www.cvedetails.com/cve/CVE-2012-0486/, retrieved on 20 June 2012

[19] http://www.cvedetails.com/cve/CVE-2011-4405/, retrieved on 20 June 2012

[20] http://www.cvedetails.com/cve/CVE-2011-3150/, retrieved on 20 June 2012