УДК 004.056.53

SOFTWARE SYSTEM FOR PREVENTION OF INTERNAL DATA LEAKS

ПРОГРАММНАЯ СИСТЕМА ДЛЯ ПРЕДОТВРАЩЕНИЯ ВНУТРЕННИХ УТЕЧЕК ДАННЫХ

P.I. Banokin, G.P. Tsapko
П.И. Банокин, Г.П. Цапко

Tomsk Polytechnic University
Томский политехнический университет
E-mail: pavel805@gmail.com

The paper considers the issues of corporative data leaks. Features of modern DLP-systems are presented and compared briefly. The authors have considered the approaches to development of data breach prevention systems. Possible scenarios of internal data theft are discussed. The necessity of database protection against data leaks is proved by statistical information. The paper describes the architecture consisting of middleware database driver and asynchronous statistics handler. Dataflow, sequence and deployment diagrams are used for describing the architectures. The authors examined the alternative architecture for multi-layered application. The article introduces the data breach detection based on behavior analysis.

Статья рассматривает проблему утечек корпоративных данных. Функциональные возможности современных систем предотвращения утечек данных кратко анализируются и сравниваются. Необходимость защиты источника данных подтверждается анализом общедоступных статистических данных. Представлены архитектуры программных систем предотвращения внутренних утечек данных для клиент-серверных и многоуровневых корпоративных приложений. Описан способ предотвращения внутренних утечек данных путем анализа поведения пользователей информационных систем.

Key words:
Information security, internal data leaks, DBMS, data loss prevention, DLP, user profile, users' behavior.

Ключевые слова:
Информационная безопасность, внутренние утечки данных, СУБД, предотвращение утечек данных, профиль пользователя, поведение пользователей.

Павел Иванович Банокин, аспирант, ассистент кафедры автоматики и компьютерных систем Института кибернетики ТПУ.
Email: pavel805@gmail.com
Научные интересы: анализ поведения пользователей программных приложений, безопасность данных, предотвращение внутренних утечек данных.

Геннадий Павлович Цапко, профессор, заведующий кафедрой автоматики и компьютерных систем Института кибернетики ТПУ.
Email: tsapko@gmail.com
Научные интересы: безопасность данных, проектирование мультипроцессорных распределенных динамических систем управления.

Introduction

Data is one of the most valuable business assets. Disclosure of secure data entries could be more devastating than material damage. The necessity of data protection is not only concern the owner but personal data protection is required in legislation system of many countries including Russia (Federal law «About personal data» issued 27.06.2006).

Number of confirmed data breach cases increases year by year [1]. Usually data breaches are prevented by specific software or hardware equipment, which could be installed on server or on every client host. The following solutions against data leaks are available on the market:

1. Complex DLP-systems which are able to monitor all data traffic. Linguistic and static (pattern-bases) methods of data analysis are commonly used in these systems [2].

2. Specific DLP-systems for data protection for one type of data sources or from one type of data breaches. Oracle Security Vault is an example of software product intended for DBMS protection.

Commonly data breach cases are divided into two categories: internal and external. External data breaches could be prevented by proper configuration of network monitors, anti-virus software utilities, timely installation of security updates and user account management. Despite of the fact that a number of internal data breaches is much lower in comparison with the external the internal ones are the hardest cases for detection [3]. Often accidents of internal data leaks are kept hidden and not presented in official statistics. Former employees could take valuable data or use their existing credentials to get data after dismissal. Internal data leaks could be performed for quite a long time period without awareness of organization higher authorities [4].

Centralized data source protection could be a more effective solution than data traffic analysis. Corporative databases are the main data repositories. A report issued by Verizon company confirms that database was affected in 6 % of data leaks accidents and 96 % of all data entries were stolen from databases [3].

The following approaches could be used for implementation of database protection system:

1. Real-time analysis of database queries. This analysis could not be implemented with SQL-triggers because popular database management systems do not support triggers for SQL SELECT statement. Tracing of DBMS process memory is implemented in McAfee software products for data protection [5]. The alternative solution for real-time monitoring could be an intermediate database driver presented hereinafter.
2. Analysis of log-files. Log-files could be created by database audit components. Then these files are stored as usual text files which need to be parsed by special software. In case of log file analysis, the detection and notification of data theft are possible only while termination of suspicious user's activity is not implementable.

Problem solution

A typical user of corporative application has a possibility to copy valuable information he has access to. The most obvious approach is to copy valuable data to portable USB storage devices. In case of forbidden access to USB ports a dishonest employee can make a screenshot or take a photo using digital camera or simply write data on a piece of paper. The only evidence that could be left after this type of crime is a fact of access to data source.

The following types of personnel are taken in consideration:

1. Universal employees. This group consists of top management, chief executive officers and other individuals who access to a wide range of corporative data during the day. This category of employees implies high level of trust.
2. Employees with predefined set of duties: technicians, experts, accountants, call-center specialists, customer service managers, etc. The number of these employees is much greater and a rate of personnel rotation is much higher than in the first group. Thus, this group requires more attention and supervision.

The solution is primarily intended for data protection from second mentioned above category of personnel. Everyday duties demand a user to access different data sources. He generates reports, views tables, creates new data entries. In case of data theft, user's behavior changes. A user can perform intensive access to tables or data categories he previously didn't use. Two basic architectures are developed for internal data leaks prevention. The first architecture is developed for client-server (client-DBMS) applications protection and consists of the following components:

1. Statistics processor is a platform-independent component responsible for storage of DBMS-requests history and detection of malicious queries. Malicious query detection process uses the history of requests performed previously.
2. Intermediate database driver is a component that acts as a statistics provider for statistics processor. This component is platform-specific. Intermediate ODBC-driver for Microsoft SQL Server is developed.

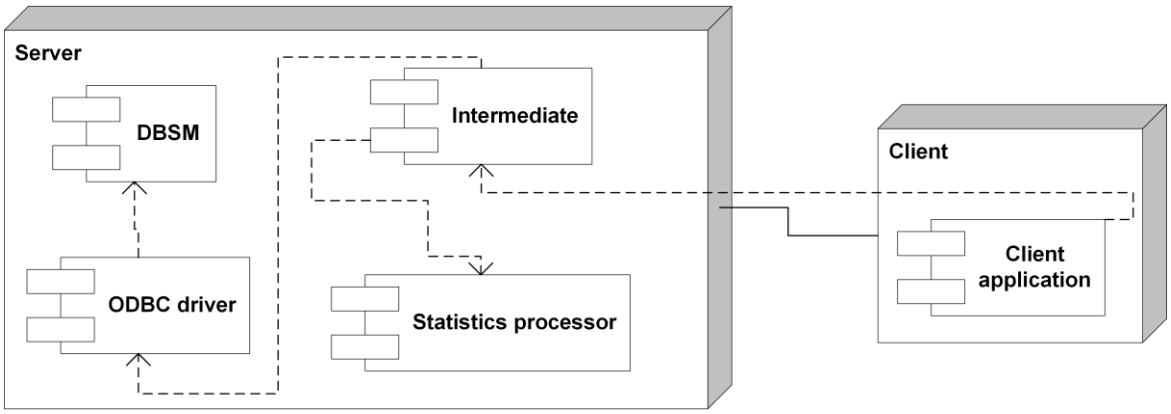

Fig. 1. First base architecture

Intermediate database driver could be used with applications like Business Studio and 1C (deployed in sql-server mode). ODBC technology was used as a primary solution for intermediate driver implementation. ODBC interface is widely used, standardized and supported by many DBMS and client software applications. These advantages of ODBC interface could help developers to create their own versions of intermediate database driver for this architecture. The main drawbacks of this architecture are slower performance in comparison with native DBMS drivers and necessity of application configuration for connection to intermediate driver. Client application connects to intermediate database driver to perform a query (Fig. 2). The intermediate driver in its turn sends the query to original DBMS driver. At each iteration this driver also requests permission for every query and sends statistical data to statistics data processor.

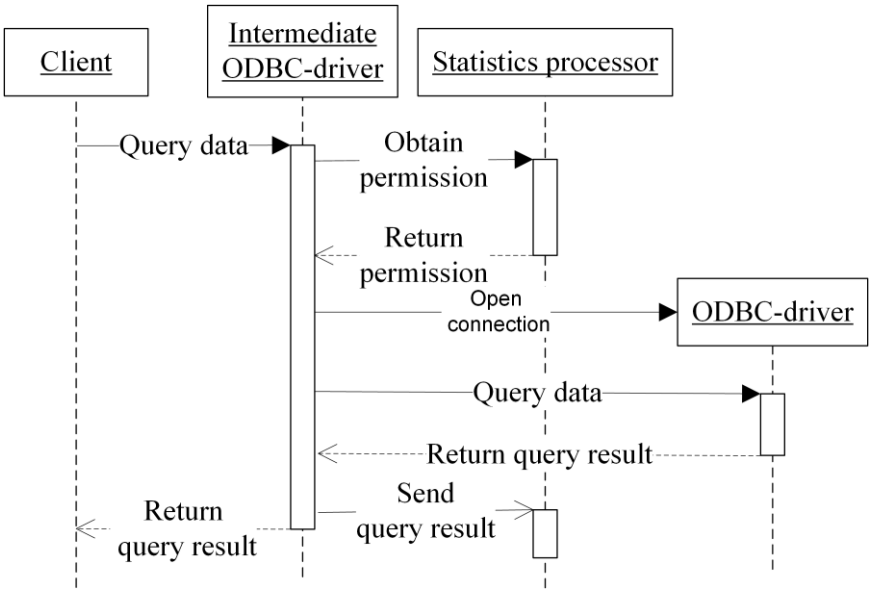

Fig. 2. Interactions between objects

In multi-layered applications users are authenticated and authorized on application server. Process of queries monitoring via intermediate driver does not allow detection of user name or role because application server interacts with DBMS thought its own account. Alternative architecture for multi-layered application is shown on Fig. 3. Extensions to application server could be developed for protecting multi-layered applications including Microsoft Dynamics NAV, Microsoft Dynamics CRM, Axapta and 1C Enterprise Platform. These applications have instruments allowing developer to modify the existing configurations and create extensions, triggers and global modules.

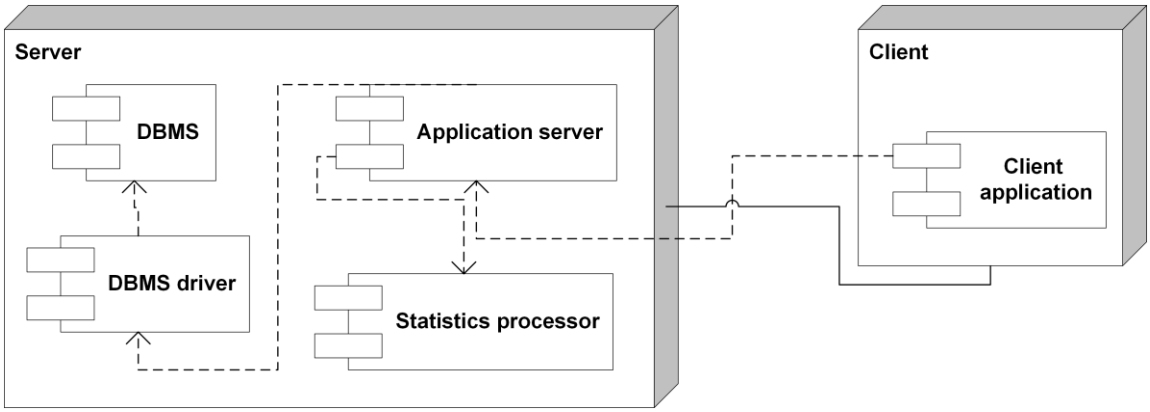

Fig. 3. Second base architecture

The architectures use two main storages: statistics storage and user profile storage. Data flow diagram for the first architecture is shown on Fig. 4. With slight changes this diagram is also applicable for the second architecture.

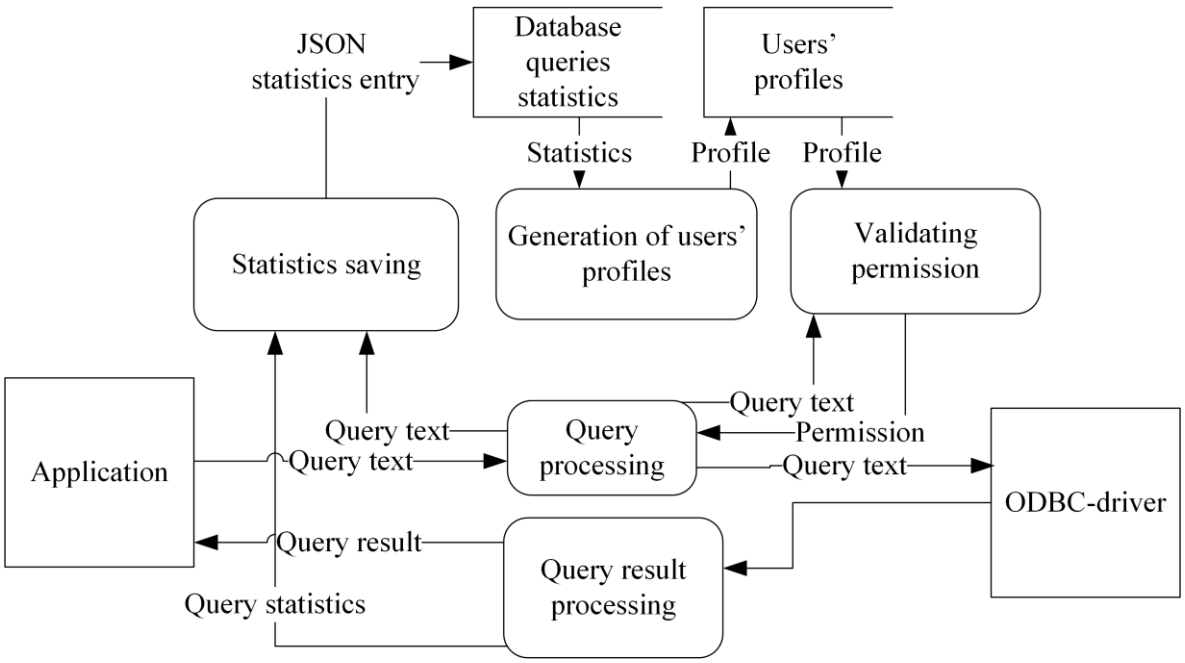

Fig. 4. Dataflow diagram

Each entry stored in the statistics database is created from query text and result. Entities include the following fields:

- Query type (SELECT, UPDATE, DELETE, CREATE);
- Time and date (Timestamp);
- User name or role name;
- A list of entities from query text;
- A quantity of rows in query result;
- Query params;
- Width (number of fields) of result row.

User or role profile is generated after a certain amount of time. User profile is a calculated mathematical model of user's behavior based of his previous requests to data storage. User profile describes usual activity. If some user's activity does not match the profile this activity could be considered as malicious. Unnatural requests to data category which previously was never accessed could be

an example of such activity. Users' profiles are generated not constantly but on some predefined time-table set by system administrator in configuration file or web-based administration interface.

The following data could be received after statistics data is collected:

- ration between different types of database queries: *SELECT*, *UPDATE*, *DELETE* или *INSERT*;
- list of entities to which user requests;
- average row width which is calculated as an average value of fields number in query results;
- density of requests – a number of queries per a certain amount of time (this value could be also calculated for every type of request separately)*;*
- a number of queries to special entities or data categories marked by system administrator as sensitive;
- usual log in time and log out time.

Data queries statistics and user profiles are stored in document oriented database. The format of those data is JSON. Complicated queries to multiple collections are not performed by statistics processor. Document-oriented storage allows getting more performance (up to 50 times greater) and scalability on simple queries to large data array than tradition relational databases [6].

Statistics processor is a service-oriented web-application consisting of a set of REST web-services. These services are implemented as a NodeJS application. NodeJS technology is available on Windows, Mac OS and Linux platforms that make statistics processor a cross platform component. Asynchronous programming interface of statistics processor contains the following methods for:

- statistics sending and receiving permission for query execution;
- management of important entities list and data categories;
- management of exceptions;
- notifications management.

The important issue of developed application system is the problem of false alarms. At current stage of the system development the false alarms are inevitable. System administrator can decide which of alarms are false and which ones need attention. Activity of some user could not be always the same. Some events (necessity of annual reports generation, special order, etc.) could affect user's activity. System administrator can add exception for particular user to make system aware that the user is going to perform some unusual actions. To facilitate this process it is possible to make a special degree of trust for each alarm. This value could be calculated using the following parameters:

1. Usual login. It is probable enough that dishonest employee could perform data theft when his colleagues are absent.
2. Suspicious activity is proved by different time profiles. To calculate this parameter it is necessary to maintain several profiles for every user. Day, week and month profiles are planned to be generated in this system. If an alarm is proved by all time profiles, it receives the highest degree of trust (high probability of data theft).

Later administrator could set the lowest degree of alarms he would like to receive notification about.

Conclusion

It is very likely that data breaches could not be completely eliminated in nearest future and enterprises would have to purchase more sophisticated solutions.

The solution described could bring benefits to medical, financial and other organizations operating sensitive, personal or secure data. Healthcare institutions (hospitals, clinics, etc.), telecommunication businesses (internet providers, VOIP providers), government and educational organizations, banks and financial brokers may become possible customers of this software system. The architecture of the software system has the following unique qualities:

1. Focus on internal data leaks protection. The solution is developed as an addition to existing traditional solutions.
2. Simple integration and centralized management of application settings, no necessity to use special hardware components, possibility of deployment on internal servers and web-servers.
3. Affordability for small businesses. Low integration and maintenance cost.

Currently the solution analyses users' behavior. In addition, it could be improved by components of linguistic analysis and static data structure analysis which is reasonable for bank account numbers and others.

REFERENCES

1. Data Loss Statistics // Data Loss DB. Available at: http://datalossdb.org/statistics (accessed 07 December 2012).
2. Phua Cl. Protecting organisations from personal data breaches. Renewal date: 01.2009. Available at: http://www.sciencedirect.com/science/article/pii/S1361372309700119 (accessed 07 December 2012).
3. Data breach investigations report 2012 // Verizon Enterprise Solutions Worldwide Site. Available at: http://www.verizonbusiness.com/resources/reports/rp_data-breach-investigations-report-2012_en_xg.pdf (accessed 07 December 2012).
4. Data Leakage Worldwide: The Insider Threat and the Cost of Data Loss // Cisco. Available at: http://www.cisco.com/en/US/solutions/collateral/ns170/ns896/ns895/Cisco_STL_Data_Leakage_2008.pdf (accessed 07 December 2012).
5. McAfee Database Activity Monitoring // McAfee. Renewal date: 08.12.2011. Available at: http://www.mcafee.com/ru/resources/data-sheets/ds-database-activity-monitoring.pdf (accessed 07 December 2012).
6. MongoDB vs. SQL Server 2008 Performance Showdown | Michael Kennedy on Technology. Available at: http://blog.michaelckennedy.net/2010/04/29/mongodb-vs-sql-server-2008-performance-showdown/statistics (accessed 07 December 2012).

Approved on 20.11.2013