AN EXTENSIBLE FRAMEWORK FOR DATABASE SECURITY ASSESSMENT AND VISUALIZATION

Tran Khanh Dang 1), Thieu Hoa Le 2), Duy Tin Truong 3)

Abstract
By using database security metrics to evaluate how risky the current database environment is and visually displaying the metric results on graphs, database security visualization and assessment method assists the administrators in holding a panoramic view of security over the database system as well as the detailed activity of each DBMS in the system. The present trend shows that it is obviously a new potential branch and this paper will introduce an extensible framework to build flexible database security measuring systems. The proposed framework allows metrics processing cores to be written in different programming languages and reside in various places of the system, the final result to be displayed on a variety of user-predefined reports and graphs, and it also provides a holistic view of accesses to the whole database systems.

1. Introduction

The need for database security arising from the explosion of Web applications such as e-commerce, online-banking, etc. has been growing ceaselessly. The fact that core enterprise databases, the most valuable information assets of most enterprises, have been kept relatively safe for many years has now completely changed when the attackers realize they have a broad, accessible and available pipe through such applications into the databases. As a result, companies are urged to spend more budgets on improving database security mechanism by purchasing the newly-devised database protection products or by training the DBAs more advancedly. Unfortunately, those do not provide the final answer for the companies on their way seeking the safest shelter for their databases. The security managers instantly feel nervous when the board of directors ask them “Are we more secure today than we were before?” or “How do we compare to our peers in this domain?”.

It is generally agreed that "You cannot control what you cannot measure” and "To measure is to know” [2]. The answer to those questions might only be obtained if there is some system able to evaluate the security level of the running system and present the assessment clearly, concisely and, most importantly, visually to the administrators or the directors. Currently, there are some security products available on the market such as Guardium [5], SecureSphere [6], etc. and they do support in measuring the security level of a database management system (DBMS) within their specific standards. However they require the running system to be equipped with expensive specialized hardwares in order to operate properly. In this paper, we introduce a general extensible framework useful for implementing such a system but in the way as cheap as possible and using no specialized hardwares but the performance still remains reasonable and acceptable. That framework combines:

1,2,3 Faculty of Computer Science & Engineering, HCMC University of Technology, National University of Ho Chi Minh City, Vietnam, dtkhanh@hcmut.edu.vn
− Database security visualization and assessment.
− Controlling and measuring simultaneously the security of many different DBMSs.
− Data auditing and analyzing.
− Flexibly configuring database security metrics.
− Complying with laws, namely data privacy acts and regulations such as SOX (Sarbanes-Oxley) [7], PCI (Payment Card Industry) [8], etc.

The rest of this paper is organized as follows: Section 2 presents our proposed framework at both logical and physical layers and how the materialized system which is built on this framework will work. Section 3 elaborates on the most noticeable traits of the framework. In section 4, we discuss key points as building a system prototype with Oracle DBMS. Finally, section 5 presents concluding remarks and the future work.

2. Basic Functional Principle

The framework will be discussed in more details later, but first of all, let’s look at the operational principle of this-framework-based system. The system collects all necessary working information of various DBMSs from their log files, reports, etc., it then calculates the final score based on the set of database security metrics, and displays it to the user. Both the DBMSs and metrics set are identified and configured by the user beforehand. The final score is comprised of the metrics’ subscores. Each subscore will be multiplied by a predefined metric coefficient before being added up to the final score. The subscore, in turn, is a sum of values obtained when running its metric on the DBMSs alternately. Thus, if there are five DBMSs in figure 1, then the subscore of each metric will be the sum of five different values evaluated by multiplying the mark when performing that metric on each DBMS and the predefined DBMS’s coefficient alternately.

![Fig. 1. Evaluate the risky level of many DBMSs](image)

When the user receives the fairly low final score which means his running system is not secured enough, he might want to find out which metric component contributes greatly to lessen the final score and hence would envisage what problem is happening. And if he is not satisfied, he will delve into that problematic metric to see which DBMS being in dangerous situation. The database security measuring system, in short, functions as a balance, putting a running system on one dish and scores on the other dish. The higher the score is, the more secure the running system is.

2.1. Logical Model

There are four layers in figure 2, but the framework logically consists of only three upper layers. Raw data about DBMS activities are collected and confined gradually to yield the final score at the visualization module. Note that there is a Laws component in three upper layers, when laws compliance is enforced this component will ensure that the data transmitted through it to the upper layer will conform to the required laws. Figure 3 show three upper layers in more detail.
−  **VISUALIZATION layer**: interacts with users, receives user requests and selects suitable presentation to show the final score. User utilizes Metrics Adapter to add, delete or edit metrics. Through User Request, user chooses which metrics to achieve his target of analyzing his current database environment. DB Security Evaluator then sends this request to Metric Parser to start processing and return the calculating result when completing and the former will show the final result in graphs or reports using Report-Graph Builder, to the user depends on his privilege and role identified by the laws.

−  **METRICS layer**: performs user request passed from visualization layer and returns metrics calculation result to visualization layer. Metrics Parser receives user request from DB Security Evaluator and has Formatter got all the necessary data conforming to laws if required and starts performing the metrics calculation with metrics from Metrics Set and finally sends the result back to DB Security Evaluator.

−  **AUDITING layer**: communicates with DBMS layer gets their raw data, refines and sends them to Metrics layer to carry out metrics calculation. The Logger will communicate with DBMSs to acquire the raw data which might have to go through Laws to ensure law compliance. When the raw data reaches Analyzer, it is refined and handed over to Formatter to reformat in some standards for Metrics layer to use later.

−  **DBMS layer is where the DBMSs reside**.

### 2.2. Physical Model

Seven main steps shown in figure 6 are explained thoroughly as follows. Each explanation is indexed corresponding to each step. (1) The user logs into the DB security measuring system, his identity as well as his role and privilege is verified by the system. When logging in successfully, he might use his privilege to add, delete or edit the metrics definition through a Metrics Adapter form (2) because the system’s default metrics do not always satisfy user requests. Alternatively, the user enters his requests using a User Request form (3) and has the system sent his request to DB Security Evaluator module (4). This module, in turn, hands this request over to Metrics Parser(5) which will put all the coming requests into the Request Pool and process them in order of their priorities, in case it is busy performing some user request. To process one user request, Metrics Parser will first parse the request, identify the chosen metrics, the coefficient of each metric and each DBMS, the type of graph (pie chart, bar chart), the format of report, etc. On finishes parsing, it
will call Metrics Controller to get all the processing cores (*.dll files, web services) of those metrics.

![Fig. 4 Physical Model](image)

After that, it will fetch the metrics input data from Auditing Database and assure the data’s laws compliance property if enforced. If such data are not available, it will have the Logger fetched them from the DBMSs (6). When all the data are ready, it will order the processing cores to perform calculating metrics and transmit the result back to the DB Security Evaluator and simultaneously save that result together with the user request in Request and Result database if he requires. This module then call Visualization module to show the final score to the user, based on his privilege and role, in the required type of graph and report (7). Note that depend on the system efficiency requirement in practice, Metrics Controller and Metrics Parser can be implemented in form of web services or modules. There might be one last question “where should all these physical components reside?” Our suggestion is to implement the following components: Metrics Adapter, Login, User Request, DB Security Evaluator, Visualization on the client side whilst putting Metrics Controller, Metrics Parser as well as all the database mentioned so far in one centralized server, also setting Logger, Analyzer, Formatter on the DBMSs side.

3. Prominent Features

3.1. Metrics

According to Oxford’s American Dictionary, a metric simply is “a system or standard of measurement”. Thus, it is accepted that security metrics are those measuring security or particularly how well security services are in the information system and database security metrics would be the security metrics applied in database area. Metric is the heart of the framework. From metrics comes the name “database security measuring system”. Without metrics, the system is defunct or no longer able to measure the security level of database environment.

Detailed discussion about metrics is beyond this paper’s cope. For any further metrics information, we recommend some literatures at the end of this paper. Now we will exemplify database security metric (DbSecMetric) with the metric “SELECT statements for privacy sets”. Privacy sets are collections of data values together forming an important privacy perspective. For example, a person’s last name is not confidential, but it together with that person’s driving license number and social security number is confidential. To use this metric, we must classify what kinds of data are crucial and define the corresponding privacy sets, then we calculate S - the overall number of users’ SELECT accesses to these privacy sets, based on the audit trails of SELECT statement and count N - the total number of database user. Let A and F be the the average frequency and final result.
\[ A = \frac{S}{N} \quad ; \quad F = \frac{\text{Max}(A, \text{threshold}) - \text{threshold}}{A} \]

Threshold might be set by the administrator’s experience to imply all the acceptable frequencies falls below it. If the final result is close to 1, it indicates that the access frequency to sensitive data exceeds the frequency threshold unusually. The administrators, therefore, might consider putting effort to investigate further or take immediate preventive actions. It is not trivial to create a DbSecMetric that can truly reflect the present database state and risk level. DbSecMetric is still in its early stage and extensive research into this new area is really encouraged. Only after a long time testing the metrics on some practical DBMSs, can we assess their efficiency and effectiveness.

3.2. Metrics Parser

The content of metrics is saved into the database using a script language. Users can create new metrics by programming with this script language. To perform metrics calculation, there needs a metrics parser which can parse the metric content to get all the essential data such as the input, output data type, where the metric processing cores are stored, etc. Besides, Metrics Parser is not directly in charge of calculating but it will call web services or .dll libraries at run time, so the metric processing cores may be written in many different programming languages and stored in many places (on web servers if they are web services). Thus, users are free to write complicated metric algorithms as well as take advantages of other system’s power of calculating. Actually, this parser parses not only metric content but also user requests to get the required metrics. However the later’s percentage is very small compared to the former’s, hence the name Metrics Parser.

3.3. Visualization

The system presents the final score in the most visual way, using many graphs and reports. There would be lots of types of graph such as pie chart, bar chart, line chart, etc. and lots of types of reports such as daily report, weekly report, monthly report or quarterly report and so on. The final result could also be compared with other results in a period of time if the user required as long as there are such results available in the database.

4. A System Prototype with Oracle

We now present a system prototype based on such framework, and built with Oracle DBMS as an example. On Oracle side, three modules are set up. Logger communicates with Oracle to achieve log files saved by using Oracle's auditing function. Analyzer next refines those log files, taking only necessary data fields out for Metrics Core to calculate metrics. Formatter reforms data to meet XML standard, compresses and encrypts to improve the efficiency, and finally utilizes socket to transmit data to Auditing Database for later use. On the client side, five components are installed. Metrics Adapter, Login and User Request are user interactive forms, DB Security Evaluator a module and Visualization a graphical library containing many dll files. DB Security Evaluator receives user request and calls Metrics Parser to process it and Visualization library to show the final score. The library is updated automatically via network by regularly downloading dll files.

Lastly, Metrics Controller and Metrics Parser are web applications set up on the same centralized server. So are all the database in figure 6. Metrics Controller receives metrics information from Metrics Adapter by web services while Metrics Parser listens to DB Security Evaluator and gets its user request to perform. Additionally, Laws Compliance is dll files, called by Logger, Metrics Parser and DB Security Evaluator when needed. These files can also be updated via network.
Metrics Core is implemented as dll files on server side or web services on other computers. We are currently implementing and conducting performance evaluations of the system.

5. Conclusion and Future Work

In this paper, we have introduced an extensible framework for developing a database security visually-measuring system. We started with presenting the basic functional principles of such framework then we showed the framework’s tractability through the logical and physical models. Finally, we gave a system prototype with Oracle DBMS. The framework provides a practical way to build an extensible system which can evaluate simultaneously the risky levels of many different DBMSs visually through various graphs (line chart, bar char, pie chart) and flexible reports. In the future, we will take further research into solving some remaining problems: data explosion on centralized server, the impacts on DBMS’s performance after installing logging components, secure and effective data transmission through compressing and encrypting. We will also add some more commonly-used metric templates to meet users’ demand in adapting or coordinating these templates to make a new desired metrics.

References