International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2015): 6.391

Detecting Forgery in Trigger Based Database Transaction

Bhagyalakshmi S[1], Maniveena C[2]

[1,2]Department of Computer Science and Engineering, College of Engineering, Kallooppara

Abstract: ***Nowadays, traditional digital investigations often excluded databases even though evidence can usually be found in them. Although the field is still in its early years, it is quickly becoming an important part of many investigations due to the increased volume of information, that may be helpful in solving different crimes and the greater risks associated with the data stored on many databases. Of major importance in database, forensics is the ability to retrace the operations performed on a database and reconstruct deleted or compromised information on the database. This requirement affects the data collection and analysis process during the forensics analysis of a database. An aspect of database forensics that has not received much awareness in the academic research community yet is the presence of database triggers. Database triggers and their implementations have not yet been thoroughly analysed to establish what possible impact they could have on digital forensic analysis methods and processes. Conventional database triggers are defined to execute automatic actions based on changes in the database. This paper attempts to develop an application, which can interface with MySQL database, helps to detect the SQL injection hijacking and reverse the actions of the intruder. This helps to protect the credibility of an organization.***

Keywords: SQL Trigger, Database SQL Injection Attacks, Database watcher, File log, Trigger Schema Table

1. Introduction

Data are used by everyone in different departments for different reasons. Therefore, data management must specify the concept of shared data. The DBMS facilitates are: 1) Interpretation and presentation of data in useful formats by transforming raw data into information. 2) Distribution of data and information to the right people at the right time. 3) Data preservation and monitoring the data usage for adequate periods of time [1]. 4) Control over data duplication and use, both internally and externally [2].

You may not be aware of it, but your life is obviously affected by database technology. Computerized databases are must to the functioning of modern organizations. You come into deal with databases on a daily basis through activities such as shopping, withdrawing cash using an ATM, ordering a book online, and registering for classes. The convenience of your daily life is partly due to growth of computerized databases and supporting database technology.

Database technology is not only improving the day to day functions of organizations but also the quality of decisions that affect our lives. Databases contain a pool of data about many aspects of our lives: consumer desires, telecommunications consumption, credit history, television observation habits, and so on. Database technology helps to encapsulate this mass of data into useful information for decision making. Management uses information obtained from databases to make long-range decisions such as investing in plants and equipment, locating stores, adding new items to inventory, and entering new businesses.

An organization's regulatory structure might be divided into three levels: higher, middle, and operational. Higher-level management makes deliberate decisions; middle management makes procedural decisions, and operational management makes daily operative decisions. The DBMS must provide solutions that give each level of management a helpful view of the data and that support the required level of decision making.

2. Proposed System

Monitoring, logging, analysis, intrusion detection, and other processes are very helpful in system architectures to increase the security of the database [3]. If the system is not applying a strong input, validation technique to check every database input to the system it will create a significant problem, because the input parameters are the first channel to the intruder that can used to push malicious code with this input [4].

Here we are proposing a new approach to detect the SQL injection attacks in a trigger based database system, and provides a suitable solution to reverse the intruder's action effects. The first step of this approach is to provide a way to capture the trigger occurrences. Whenever the trigger occurs, it shouldnote down. In addition, the time of the occurrence and which all are the documents affected by the triggers. In MySQL database already, there is a solution available for this purpose. There is an Information Schema Database is available and it consists of a Trigger table.

Here in our method we try to get this information explicitly. I.e. first, we are collecting all the details regarding the database. We are logging all these details in a separate word document named 'logs'. This document has already linked with our database. So that every updates can be logged in the 'log' document. Whenever the trigger occurs, those details will collected in a separate database. From there we get all the details regarding the trigger actions and timing. By monitoring those records, the administrator is able to recognize whether it was an official function or an intruder action. The timing will provide the clues about the intruder action.

By collecting the details of triggers, we are not getting out of the complete problem. If the manipulation of data have occurred by the action of any intruder or by an unintentional mistake, we have to reverse the action result [5]. So that only we get the correct data back. Either if it was an intruder attack or not, theadministrator is able to roll backs the total action effects.

Here what we are doing is, whenever we find some SQL injection attacks in database we can determine which manipulation process has taken place. Here we are providing a response watcher to monitor all the queries. The file log window will provide all the queries and the response watcher is responsible to monitor this window, and these details will be stored in separate database.

If the trigger database has updated and the manipulation of data detected then we can conclude that the triggers have executed. Then we can collect the complete action details from the Trigger table itself.

Whether the trigger occurred is from an intruder or not, is determined by analysing the pattern, sequence and time of the quarries. If the action has performed by the attacker then we tries to reverse the occurred actions using corresponding quarries.

It is a major problem that the manipulations made by the intruder or the attacker will affect the integrity of data [6]. Moreover, it cannot accepted at any cost. In case of databases, the chances are low, but if the database uses the trigger, there is a chance of manipulation due to the trigger. Especially in case of non-data triggers. In such cases, it is not necessary to perform any actions on the database; here the triggers occur during the normal running and usage of the database.

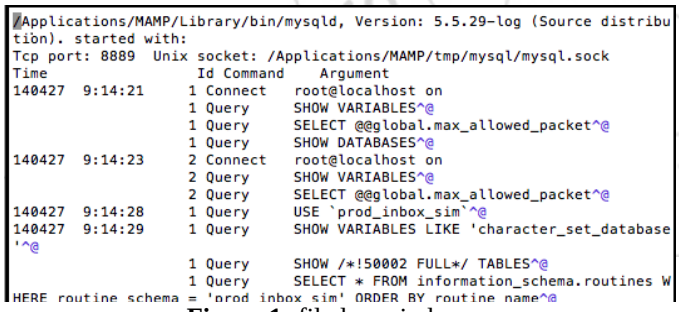

Figure 1: file log window

We have provided the solution to look in the trigger actions. I.e. an administrator or an investigator can easily find out, which all triggers have occurred in the database, at what time they occurred,and which action have been takes place [7]. The solution provided by this paper will also help to roll back the trigger effects. Moreover, we can access the original data back [8]. I.e. whenever a manipulation occurred in the data by trigger, we can get the details from the Trigger schema. From there the complete details about the trigger event or the action will collected. If any actions occurred by the trigger, by our proposed solution we tries to reverse the occurred action by corresponding quarries. As a result, we will get the original data back.

A. Database watcher

Here the file log window is acting as database watcher. It watches the updates (query) of the database and collects all the details regarding the updates. Each bit of data can be obtained from this [9]. By analysing these details, the SQL injections can easily detected.

B. Response watcher

The response watcher is responsible for making decisions about SQL injection attacks. Manipulations can be of many types, which have done by an intruder or unintentionally by an employee [10]. The administrator or the investigator can take decision about it. He can decide whether to conclude it as an attack and take necessary actions or not.

C. Trigger action

Here in this section we collect the data, which have connected with the triggers, from the log file. The queries, which are responsible for invoking triggers, have noted down here. From this stage, an administrator or an investigator performs the reverse action to get back the original data.

3. Conclusion

Although very little amount of research has been done on database forensics, current research has typically focused on digital examination and reconstruction of databases. Database triggers are designed to perform automatic actions based on events that occur in a database. There arewide varieties of actions that can performed by the triggers. These actions can potentially have an effect on data inside and outside of the DBMS. Thus, triggers and the actions they perform are forensically important.

The effect that triggers can have on data raises the concern that they could compromise the integrity of the data being investigated. We have proposed an application, which can interface with MySQL database, to protect the credibility of an organization. This paper tries to detect the SQL injection attack in the database system and by identifying the time and condition of occurrences of the triggers and the affected file lists, the main purpose is to rollback, the intruders trigger action effects, which lead to get back the original data.

4. Acknowledgement

We would like thank all the faculty members of college of engineering, Kallooppara.

References

[1] S. W. Boyd and A. D. Keromytis, (2004). “SQLrand: Preventing SQL Injection Attacks,” In Proceedings of the 2nd Applied Cryptography and Network Security (ACNS) Conference, pages 292–302.
[2] Russell A. McClure and Ingolf H. Kruger- (2005): “SQL DOM: Compile Time Checking of Dynamic SQL Statements”, pages 91-92
[3] KeWei; Muthuprasanna, M.; Kothari, S., (2006), "Preventing SQL injection attacks in stored

procedures," Software Engineering Conference. Australian, vol., no., pp.8 pp., 18-21 April 2006.

[4] J.C. Lin and J.M. Chen, (2007), ―The Automatic Defence Mechanism for Malicious Injection Attack,‖ Seventh International Conference on Computer and Information Technology, IEEE Computer Society.

[5] Iain, M.; Clark, A.; Mohay, G. (March 2008), "Evaluation of Anomaly Based Character Distribution Models in the Detection of SQL Injection Attacks," Availability, Reliability and Security, 2008. ARES08. Third International Conference on, vol., no., pp.47,55, 4-7.

[6] R. Ezumalai, G. Aghila(March 2009), "Combinatorial Approach for Preventing SQL Injection Attacks", 2009 IEEE International Advance Computing Conference (IACC 2009) Patiala, India, 6-7.

[7] Ankit Anchlia, Sheela Jain (2010)," A novel Injection Aware Approach for the Testing of Database Applications", International Conference on Recent Trends in Information, Telecommunication and Computing, IEEE.

[8] Tajpour, A.; JorJorZadeShooshtari, M. (July 2010), "Evaluation of SQL Injection Detection and Prevention Techniques," Computational Intelligence, Communication Systems and Networks (CICSyN), 2010 Second International Conference on, vol., no., pp.216,221, 28-30.

[9] Elia, I.A.; Fonseca, J.; Vieira, M. (Nov. 2010), "Comparing SQL Injection Detection Tools Using Attack Injection: An Experimental Study," SoftwareReliability Engineering (ISSRE), 2010 IEEE 21st International Symposium on, vol., no., pp.289, 298, 1-4.

[10] Tian Wei; Yang Ju-Feng; Xu Jing; Si Guan-Nan (July 2012), "Attack Model Based Penetration Test for SQL Injection Vulnerability," ComputerSoftware and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual, vol., no., pp.589,594, 16-20.