Volume: 08 Issue: 06 Pages: 3290-3293 (2017) ISSN: 0975-0290

SECURE: An Ameliorated SQL Semiotic for Security

Dr. Ajeet A. Chikkamannur
Department of Computer Science and Engineering, R. L. Jalappa Institute of Technology, Bangalore 561203, India
ac.ajeet@gmail.com
Dr. Shivanand M. Handigund
Department of Information Science and Engineering, Vemana Institute of Technology, Bangalore 560034, India
smhandigund@gmail.com

ABSTRACT

The Information Systems, Databases are vital and critical in the arena of Business, Enterprise, Governance etc., for the decision making. Further, the Information Systems are designed to reach maximum number of people for interaction with system. This intent of system usage appears to be resourceful but the identification of reliable users of system has challenged the technocrats

Most of the applications uses relational data base system (RDBMS) for the database support but with intelligent exploitation of vulnerability in Structured Query Language (SQL) of RDBMS, the dishonest people are interested in manipulation or processing of the data for business value reduction or damage

This paper attempts to endow with security to a relational database system by the inclusion of newly designed semiotic to keyword SECURE into a lexicon of SQL. Then the semiotic is articulated in relational algebraic expression for current query engine execution.

Keywords **- Query, Security, Relation, Set Difference, Semiotic**

Date of Submission: June 02, 2017 **Date of Acceptance**: June 16, 2017

I. INTRODUCTION

Any business organization keeps a purpose of attracting the various populaces for their business services. To support this desirability, the information systems like Web Technology etc. are used for design a system to reach maximum number of populace for interaction. This intent of system usage is admirable but the *identification of reliable users of system* has hardened the aptitude of developers in a development of information systems. Among the user, the dishonest community exercise, spoil and ruin the value of information. Hence, the technological convene we observed is that how to make available the right information to a right person?

Currently, many contemporary information systems are designed with a database support. The data of databases are vital and decisive in the arena of Business, Enterprise, Governance etc., for various analysis. But, with a smart *exploitation of vulnerability in a database system*, the dishonest people are manipulating or processing the data for value reduction or damage. This is another technical apprehensive activity in a database? Hence, it is time to re-inspect or reengineer the core database system design in purview of security, which causes the clout to the system.

This paper tries to provide security to a relational database by the blend of innovatively intended semiotic i.e. syntax, semantics and pragmatics into a lexicon of Structured Query Language (SQL). Then the semiotic is translated to the relational algebraic expression for practical query engine execution.

II. BACKGROUND

The relational database model is developed by Codd [7, 8]. Most of the applications are using the relational database for their data storage and retrieval. The Structured Query Language (SQL) is designed and developed to access the Relational Database. In relational database, the security to multiple users is given by the Database Administrator depending on the access privilege of users i.e. GRANT privileges. At early stages, the relational model is subject for a centralized computer system. But over a period of time the evolution in the technology viz., client-server, distributed computing, cloud computing etc. have used the SQL carefully in many applications by several deviation in the database system.

Currently, the web applications with database are developed for multi consumers to operate the database. These multi loggers may belong to legitimate or illegitimate type. The recognition of reliable and trustworthy users of web technology is a herculean task for proviso of accessing the relational database. On ignorance of recognition, the dishonest populace tries to damage or use the database to ruin the business value of an organization.

Though the security is provided by the pragmatic relational database systems, the hackers of the system are enjoying with the SQL injection vulnerability [5, 6]. Though many security solutions are provided to a database but in literature, we have not observed the effort in the arena of broadening the lexicon of the SQL. Hence, the semiotic of SQL, has to be reengineered or broaden for

Volume: 08 Issue: 06 Pages: 3290-3293 (2017) ISSN: 0975-0290

security point of view with addition of new word(s). Further, we have observed that most of the work of security on relational database is carried in middleware without ameliorating the basic semiotic of language. Various ways of broadening the base of SQL can be referred in [1, 2, 3, 4] where the limitation of language phrase is broadened or expanded.

III. FRAME WORK

This paper proposes a formula of securing the relational database. The formula is underpinned with the second principle of information technology; "*the right information is given to the right person"*. The study on a statement discloses in a two practical measurements i.e. "right information', "right person". These practical measurements are to be ensured to facilitate the multi user systems in an efficient and secured way. *Hence, we consider the quantum of data stipulation to an end user for security i.e. the quantization is enumerated on the amount of data retrieval through a SQL query.*

The unlawful users' curiosity is that they use SQL semiotics for accessing the entire data of a database. This is our observed trace of *vulnerability of unauthorized user* where the user tries to access the irrelevant data. With superfluous information and/or logical query (ies) combinations, they exploit the database for gaining the control of system. The SELECT, FROM and WHERE, query semiotic of SQL is logically used for accessing the database. Hence, this semiotic is ameliorated to resolve the drawback by defining further distinctiveness to a relation.

Hence, to protect a relation of relational database, the keyword SECURE is proposed to the lexicon of SQL. Like use of keyword PRIMARYKEY for elimination of duplication of values in a relation, the keyword SECURE is added to the Data Definition Language (DDL) of SQL for the protection of relation in threefold viz., secure at table level, secure at column level and secure at row level.

The SECURE table name differentiates that the content of entire table is non-visible i.e. out of sight to user. The SECURE column name or tuple are designed to protect the column name contents or row values by the non-display of data belonging to the specified column or tuple. Since contents are not put on show, for the reliable users the data corroboration is ensured by returning the true condition. Further, the query with convened character '*" in SQL semiotic is blocked to carry out on SECURED relation.

The proposed semiotic for SECURE keyword is given below

SECURE [table_name | attribute | primary key value];

The semiotic is used to SECURE the table, attribute or tuple depending on the application requirement. The secured table_name, attribute or tuple is stored in the metadata. Before executing the SQL query given by the user, the table_name, attributes or tuple specified in the query are verified with the meta data contents. For regular condition i.e. tables, attributes or tuples without security are executed with existing process of execution. On secure condition; the following functionalities are to be carried out depending on the secure semiotic expression.

CASE 1: User needs to secure the contents of entire table. The designer has to utilize the below semiotic for a relation definition in a relational database. This causes the non-visibility of entire table contents on access. Further the SQL query with meta character '*" is disabled for execution on the table.

SECURE table_name;

On expression of a query, the table name specified is appended with word "security". This is stored in a meta data for further utilization. Any SQL query expressed in a relational algebraic form is implemented and executed on SQL compiler. Hence, on encountering the secured relation name in the SQL query, the routine semiotic of SELECT, FROM, WHERE execution is ameliorated to a following relational algebraic expression.

$$\textbf{Result} = \sigma_{\text{<condition>}} \text{<table name>} - \sigma 1\text{<table name>}$$

CASE 2: User needs to secure the particular column's content. The proposed designed semiotic of a query is given below

SECURE attribute;

The below semiotic is designed to disable the visibility of the entire column content on access. Again the meta character '*', expression in SQL is prevented for execution in query engine. The relational algebraic form of query execution is;

$$\textbf{Result} = \sigma_{\text{<condition>}} \text{<table name>} - \sigma 2_{\text{<attribute>}} \text{<table name>}$$

CASE 3: User needs to secure the particular row content. The designed semiotic of query is given below;

SECURE primary key value;

This semiotic is designed to secure the row of relation for non access i.e. the particular row is not visible to the user. In this case the primary key value of particular row is to be specified. The relational algebraic form of query execution is given below;

$$\textbf{Result} = \sigma_{\text{<condition>}} \text{<table name>} - \sigma 3_{\text{<primary key attribute = value>}} \text{<table name>}$$

IV. CASE STUDY

To demonstrate the execution of designed semiotic of query, the relation MYTABLE is constituted with some data values. Most of the applications are using this table for authentication of user in web application.

Volume: 08 Issue: 06 Pages: 3290-3293 (2017) ISSN: 0975-0290

MYTABLE

UID	PASSWORD
abc@org.com	Abc
lmn@org.com	Lmn
ijk@org.com	Ijk
xyz@org.com	Xyz

The pragmatic SQL semiotic allows routine queries like SELECT * FROM MYTABLE, SELECT * FROM MYTABLE WHERE UID= " ". The unrelated users access the data values from a table with vulnerability in such query expression [5, 6]. The exploitation of query leads to confidentiality destruction of a data. Our approach works in the following way;

CASE 1: Entire table is secured. The entire table is secured with execution of following steps.
Step1: The table name MYTABLE is to be secured for non access. Hence, the following semiotic is used

SECURE MYTABLE;

The table name MYTABLE is appended with secure keyword and stored in the metadata for executing ensuing steps. Any retrieval access on table initially execute the below relational algebraic expression query.

σ 1<MYTABLE>;

Step2: The result of the step1 execution is

UID	PASSWORD
abc@org.com	Abc
lmn@org.com	Lmn
ijk@org.com	Ijk
xyz@org.com	Xyz

Step 3: The unauthorized user of this table attempts the following query

SELECT * FROM MYTABLE:

The relational algebraic form this query is;

σ <MYTABLE>;

Step 4: The result of step 3 execution is;

UID	PASSWORD
abc@org.com	Abc
lmn@org.com	Lmn
ijk@org.com	Ijk
xyz@org.com	Xyz

Step 5: Here the ameliorated relational algebraic expression is executed on the table

Result = σ $_{<condition>}$ <table name> − σ 1<table name>

This is executed with **set difference** operation among the results of step 2 and 4.

Result = {step 4 values} – {Step 2 values}
= NULL

Hence, the content of table is displayed as NULL.

CASE 2: The attribute of table is secured. The attribute of a table is secured by the execution of following steps.
Step 1: The PASSWORD attribute of a table is secured by the following semiotic

SECURE PASSWORD;

Step 2: On encountering the secure tag with PASSWORD attribute the following relational algebraic operation is executed

σ 1$_{<PASSWORD>}$ <MYTABLE>

The result of this step is given below

PASSWORD
Abc
Lmn
Ijk
Xyz

Step 3: Suppose the following is attempted to execute

SELECT UID, PASSWORD
FROM MYTABLE;

The result of this step will yield the following on regular query

UID	PASSWORD
abc@org.com	Abc
lmn@org.com	Lmn
ijk@org.com	Ijk
xyz@org.com	Xyz

Step 4: Since the attribute PASSWORD is designated as secured attribute the following algebraic expression is executed.

Result = σ <MYTABLE> − σ2 $_{<PASSWORD>}$ <MYTABLE>

This is executed with a **set difference** operation on resulting values of step 3 and step2. The order is strictly tracked to shun the confusion. On completion of this step gives the following result

UID
abc@org.com
lmn@org.com
ijk@org.com
xyz@org.com

Volume: 08 Issue: 06 Pages: 3290-3293 (2017) ISSN: 0975-0290

The column PASSWORD contents are unseen by the users

CASE 3: The particular row value is secured. The following steps are executed.

Step1: The relation is constituted with one of its attribute value as primary key. Here, we will consider the PASSWORD attribute as primary key attribute. Now the row pertaining to "Abc" is to be secured. So the following semiotic is used to define it as secured.

SECURE "Abc";

Step 2: On encountering the SECURE tag with PASSWORD attribute's value "Abc", the following relational algebraic expression is executed

$$\sigma 3_{\text{<PASSWORD = "Abc">}}\ \text{<MYTABLE>}$$

The resulting values are depicted below;

UID	PASSWORD
abc@org.com	Abc

Step 3: Suppose the following routine query is executed by the user

SELECT *
FROM MYTABLE
WHERE PASSWORD ="Abc";

This step will yield the following

UID	PASSWORD
abc@org.com	Abc

Step 4: since the primary key value "Abc" is appended with SECURE keyword the following ameliorated algebraic expression is executed for the final result

$$\text{Result} = \sigma\ \text{<MYTABLE>} - \sigma 3_{\text{<PASSWORD = "Abc" >}}\ \text{<MYTABLE>}$$

Here the **set difference operator** is applied among values of step 3 and step 2. This order is strictly followed for avoid the ambiguity. This step results in following values.

UID	PASSWORD
lmn@org.com	Lmn
ijk@org.com	Ijk
xyz@org.com	Xyz

The row with primary key value "Abc" is unseen by the user.

V. CONCLUSION

The work carried out in this paper is including the semiotic for the new keyword SECURE at the level of table, attribute and tuple of the relation in Relational Database Management System. The designed semiotic appends the keyword "secure" with table, attribute or tuple depending on semiotic use and stores in metadata.

On encountering the appended keyword, the semiotic is disabling the execution of meta character '*" from SQL query and only the specific information is provided depending on the level of security on relation i.e. irrelevant data is made invisible

Further, the work is to be extended for the provision of data for reliable users.

REFERENCE

[1] Ajeet Chikkamannur, *"Design of Fourth Generation Language with Blend of Structured Query Language and Japanese Basic English",* Ph.D. thesis, Visvesvarya Technological University Belgaum, 2013

[2] Ajeet Chikkamannur, Shivanand Handigund, "A Concoct Semiotic for Recursion in SQL", *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS),* ISSN: 2278-6856, Vol. 2, Issue 3, page 204-210, June 2013

[3] Ajeet Chikkamannur, Shivanand Handigund, "Automated Methodology to Reduce the Redundancy in Relational Database", *Elixir Comp. Science. & Engineering 61,* ISSN: 2229-712X, page 16946-16949, August, 2013

[4] Ajeet Chikkamannur, Shivanand Handigund, "An Ameliorated Methodology for Ranking the Tuple" *International Journal of Computers and Technology(IJCT),* Vol 14, No. 4, pp 5616-5620, January 2015

[5] Amit Chaturvedi et al, "Analysis of SQL Injections Attacks and Vulnerabilities", *International Journal of Advanced Research in Computer Science and Software Engineering* ,Volume 6, Issue 3, 2016

[6] Atefeh Tajpour et al, "SQL Injection Detection and Prevention Techniques" *International Journal of Advancements in Computing Technology,* Vol 3, Issue 7, pp 82-91, DOI: 10.4156/ijact.vol3.issue7.11, 2011

[7] C. J. Date, A. Kannan, S. Swaminathan, "An Introduction to Database Systems", *8th Edition, Pearson Education (Dorling Kindersley (India) Pvt. Ltd.),* 2008.

[8] E.F. Codd, "A Relational Model of Data for Large Shared Data Banks", *Comm. ACM 12 (6)*, page 377-387, 1970