KATY Katy Independent School District

Office of Public Information

March 2, 2018

Office of the Attorney General
Open Records Division
P.O. Box 12548
Austin, TX 78711

Re: Public Information Request 17405_40_Dolan
15 Day letter
3rd Party Proprietary Information Implicated (PowerSchool)

Dear Open Records Division:

The Katy Independent School District received a written request for information from Mr. Sean Dolan, a copy of which is attached and labeled "A." The request seeks a list or printout of all data fields currently used by the district in the eSchoolPlus program for housing student data. The District believes that the eSchoolPlus data dictionary is responsive to this request. A copy of the data dictionary is attached and labeled "B."

The fifteen business day deadline for this letter, pursuant to Tex. Gov't Code Sec. 552.301(e) is Monday, March 5, 2018.

In lay terms, what is responsive to Mr. Dolan's request is a set of files that contains database metadata. A data dictionary contains records about other objects in the database, such as data ownership, relationships to other objects, and other data. Because of the contents, release of the eSchoolPlus data dictionary implicates the proprietary interest of the software vendor. The District notified the software vendor of Mr. Dolan's request for information and its' right pursuant to Sec. 552.305(d) to submit arguments as to why the information at issue should not be released. A copy of PowerSchool's objection is attached and labeled "C."

The eSchoolPlus data dictionary describes aspects of the database that are used as guides by database programmers who maintain the information system. The dictionary also acts as a guide for future programmers to understand the logic used in developing relationships between the data and data tables. Consequently, this information is not "documentation" of the kind made public by section 552.002 of the Public Information Act. Rather, it is tools, keys or similar explanatory information.

This type of information is excluded under the definition of what must be disclosed as public information. The Public Information Act is applicable only to "public information." See Gov't Code Secs. 552.002-552.021. Section 552.002(a) defines "public information" as "information that is written, produced, collected, assembled, or maintained under a law or ordinance or in

connection with the transaction of official business" by (1) a governmental body; or (2) for a governmental body and the governmental body owns the information, has a right of access to the information, or spends public money for the purpose of writing, producing, collecting assembling, or maintaining the information; or (3) by an officer or employee in their official capacity and the information pertains to official business. *See* Sec. 552.002(a).

Prior Open Records Decisions have specifically held that data dictionaries are not documentation that must be disclosed under the Public Information Act. Computer information not covered by the Act (and excluded from the definition of public information) was notably discussed in Open Records Decision No. 581 (1990) where it was determined that certain computer information, such as source codes, documentation and computer programing has no significance other than its use of a tool for the maintenance, manipulation, or protection of private property and, therefore, is not the kind of information made public under section 552.021. *See* ORD 581 at 6 (construing predecessor statute).

Considering the very issue at hand, Open Records Decision No. 14338 (2011) held precisely that computer data dictionaries are not public information as defined by the Act and, therefore, are not required to be released. *See* ORD 14338 at 3. And more recently in 2016, Open Records Decision No. 26487, it was again determined that data dictionaries are not public information subject to the Act. *See* ORD 26487 at 2.

The potential security risks associated with release of data dictionary information support the decision not to require disclosure. Here, release of the data dictionary would compromise the security of the eSchoolPlus database which contains confidential student data belonging to more than 77,000 current students and additionally thousands of former students. The release of the specific structure of the database would give an attacker the knowledge needed to attempt a targeted SQL injection attack and would arm an attacker with the means to gather the most secure and personally identifiable information housed by the District. SQL injections involve attacks by which an attacker alters the structure of the original SQL query by injecting SQL code in the input fields in the web form in order to gain unauthorized access to the database. A SQL injection attack can read customer data from the server database, modify the database data, execute administration operations, recover the content of a table stored on the database file and, in some cases, issue commands to the operating system. *See* attached article K.P., Biji (2015) Data Dictionary Based Mechanism against SQL Injection Attacks, *International Journal of Engineering and Computer Science, Volume 4*, Issue 6, 12453, 12453-12457. A copy of this article is attached and labeled "D." It is this sort of potential security breach that underlies the decision to exclude data dictionaries as information subject to disclosure under the Act. For example, the decision in ORD 581 (1990) specifically highlighted, "While acknowledging the

comprehensive scope of the term 'information' the attorney general nevertheless determined the legislature could not have intended that the Public Information Act compromise the physical security of information management systems or other government property." *See* ORD 581 at 6.

The District believes that the data dictionary information is not information subject to disclosure under the Public Information Act. The District requests a determination that it can withhold the requested information.

By my signature below, I certify that the request for information was received on February 12, 2017. If you have any questions, or need additional information or documentation, please contact me at (281) 396-7883.

Sincerely,

Sabrina E. Fernandez

Sabrina E. Fernandez
KATY INDEPENDENT SCHOOL DISTRICT
PUBLIC INFORMATION SPECIALIST

cc: Sean Dolan via email to sean@pushfire.com (excluding Exhibit "B")

Fernandez, Sabrina E (SPEC SRVCS)

From: Sean Dolan <sean@pushfire.com>
Sent: Monday, February 12, 2018 12:39 PM
To: PublicRecords
Subject: Re: PIR 17374_30S_Dolan Clarification on Possible New Public Information Request

This is correct, thank you!

On Mon, Feb 12, 2018 at 12:25 PM, PublicRecords <PublicRecords@katyisd.org> wrote:

Mr. Dolan,

Thank you for speaking with me over the phone. As we discussed, you would like to withdraw the request for a fee estimate below and instead submit a new request for a list or printout of all data fields currently used by the district in the e-school plus program for housing student data.

Please confirm that this is correct, and I will proceed by submitting a request with our technology department.

Thank you,

Sabrina Fernandez

Public Information Unit

Office of the General Counsel

Katy Independent School District

PowerSchool

PowerSchool Group LLC
150 Parkshore Drive
Folsom, CA 95630

February 28, 2018

Via Certified Mail
The Honorable Ken Paxton
Attorney General for the State of Texas
Open Records Division
P.O. Box 12548
Austin, Texas 78711-2548

RE: Katy ISD Public Information Request 17404_40_Dolan:
Arguments to Prevent the Disclosure of Confidential Information

Dear Mr. Attorney General:

On February 27, 2018 PowerSchool Group LLC ("PowerSchool") was notified of a Texas Public Information Request made to Katy Independent School District ("District"), a PowerSchool customer, for information related to one of PowerSchool's products, eSchoolPLUS. The Request was made by Sean Dolan of Pushfire. ("Requestor"). Mr. Dolan is requesting a "list of all data fields currently used by the district in eSchoolPLUS program". This list is confidential information excepted from disclosure under Chapter 552 of the Texas Government Code.

While information submitted to a public agency is subject to disclosure, Section 552.110 exempts from disclosure, "a trade secret obtained from a person and privileged or confidential by statute or judicial decision." TEX. GOV'T Code §552.110(a). A trade secret under Texas law, includes, "a compilation of information which is used in one's business, and which gives him an opportunity to obtain an advantage over competitors who do not know or use it. *Hyde Corp. v. Huffines,* 314 S.W.2d 763, 776 (Tex. 1958). PowerSchool's data fields for eSchoolPLUS are just that: a compilation of information which creates the database structure central to this product. This structure is what sets the product apart in the marketplace and is the backbone of the product's functional capabilities.

Additionally, Section 552.104(a) exempts information from disclosure when doing so, "would give advantage to a competitor or bidder." TEX. GOV'T CODE § 552.104(a). The Texas Supreme Court held in The Boeing Co. v. Paxton that a private company may assert this exception to protect sensitive information from a competitor. *Boeing Co. v. Paxton,* 466 S.W. 3d 831 (Tex. 2015). Further, Section 552.104 protects information when the governmental body solicits bids for services on a recurring basis and the disclosure of information would provide an opportunity for competing companies to undercut the other's bid. TEX. ATT'Y GEN. ORD-541.

Requestor may be working for a direct competitor of PowerSchool in the state of Texas.

While the release of information will not impact the contract for District, PowerSchool and its competitors compete for the same school district business time and time again. The database structure and capability is often the distinguishing factor between companies in this market.

Allowing Requestor access to the list of all data fields would allow insight to PowerSchool's overall database structure and functionality, exposing trade secrets and providing an "advantage to a competitor or bidder". TEX. GOV'T CODE § 522.104(a). Therefore, PowerSchool requests that its list of data fields remain confidential and be withheld from disclosure to the requestor.

Sincerely,

DocuSigned by:
Alexandra Hodge
CC3F0382E1A34D4...

Alexandra Hodge
Corporate Counsel
PowerSchool Group LLC

Cc:

Via Electronic Mail – Sabrina Fernandez, publicrecords@katyisd.org
Katy ISD

Via Electronic Mail – Sean Dolan, sean@pushfire.com
PushFire

ijecs
open access www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 4 Issue 6 June 2015, Page No. 12453-12457

Data Dictionary Based Mechanism against SQL Injection Attacks

***Biji.K.P.*[1]**

[1]ME CSE,
Dept. of CSE, PGI, Palladam, TN, India
bijikp@gmail.com

Abstract: ***"Data Dictionary Based Mechanism Against SQL Injection Attacks" which helps and manages the important private customer data in a secured manner by mirroring the database structures into unique secure mirroring tables which is managed in a differently managed secure data management system. This plays an effective medium in the prevention of SQL Injection, which is one of the main web attack terminology which is effective utilized by various hackers to steal data from organizations which managed their transactions through online transactions and databases. This is a unique type of intrusion that takes advantage of improperly managed coding in the web applications. SQLIA allows intruders to inject SQL commands into access data's from the web forms to allow them to gain access to the data held within your database. In this paper we will discuss several types of SQLIAs, existing techniques and their drawbacks. Finally I have proposed a solution for detection using data dictionary and prevention using the intrusion search along with ASCII values. I have implemented it using ASP.net with VB.net and SQL server 2008, although this algorithm can be implemented in any language and for any database platform with minimal modifications.***

Keywords: Data dictionary; SQL IA; ASCII; intrusion search.

1. Introduction

A SQL injection attack defined as insertion or "injection" of a SQL query through the input web data from the user to the software/webpage. A proper SQL injection attack can read customer data from the server database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a table stored on the database file and in some cases issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected through the user data input in order to control the execution of written SQL commands. SQL injection mainly works as an injection technique that exploits a security vulnerability occurred in the database layer of an application through improper coding styles. SQL injection is one of the oldest attacks against web applications. This type of vulnerability may present when user input is either incorrectly filtered for string literal escape characters coded and embedded in SQL statements or user input is not strongly typed and thereby unexpectedly executed. It is an instance of a most generic class of vulnerabilities that may occur whenever one program or script is embedded inside another.

2. SQL Injection Attacks

SQL Injections are attacks by which an attacker alters the structure of the original SQL query by injecting SQL code in the input fields of the web form in order to gain unauthorized access to the database. Although the vulnerabilities that lead to SQLIAs are well understood, they persist because of lack of effective techniques for detecting and preventing them. Several solutions have been proposed in literature to prevent SQLIAs in the application layer. Although these solutions prevent SQLIAs at the application layer, very little emphasis is laid on securing objects residing in the database layer. A data dictionary is a file or a set of files that contains a database's metadata. The data dictionary contains records about other objects in the database, such as data ownership, data relationships to other objects, and other data.

3. Data Dictionary

A data dictionary is a file or a set of files that contains a database's metadata. The data dictionary contains records about other objects in the database, such as data ownership, data relationships to other objects, and other data. The proposed Data Dictionary based algorithm is a Combination of DDL & DML Mapping along with the collection of SQL Queries coming inside the webpages/application forms. In DDL & DML Mapping we generate a Sparse Image of the Structure and Data contents of the Database to a Mirror Database which will be stored in parallel. Along with the Sparse Image the Vectorization of the SQL Queries is also store in tables of the Mirror Database, to include different syntax, we resolve the parse tree of different generated queries. As a result of the output of the different generated queries, we can monitor the detection of abnormalities among the queries within our database structure.

3.1 Types of Sql injection attacks

SQL injections can be implemented in the following ways:

- Tautology
- Illegal/logically incorrect queries
- End of line comment
- Timing attack
- Union queries
- Blind SQL injection attacks
- Piggy backed queries

- **Tautology:** This technique injects sql statements which will be always true so that the queries must return results upon evaluation of WHERE condition. Injected query: select name from user_details where

Exhibit "D"

username='softech' and password=''or '1'='1'.true so that the queries always return results upon evaluation of WHERE condition. Injected query: select name from user_details where username='softech' and password=''or '1'='1'.

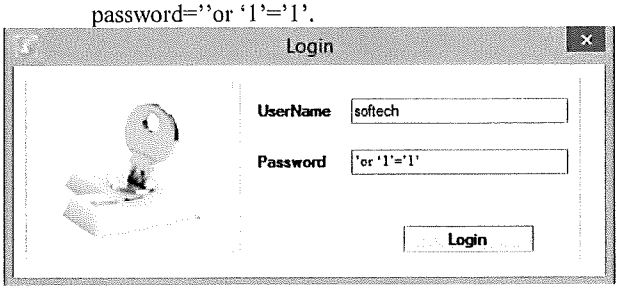


Figure 2.1: Tautology

- **Illegal/Logically incorrect queries:** This method is used by the hackers to retrieve related information about the database. Attacker intentionally inputs invalid SQL tokens or junk data input in the sql query to generate logical errors, type mismatches or syntax errors.

For example:
Original Query: sam/profile.aspx/?id=10
Injected Query: sam/profile.aspx/?id=10'

Error: select name from user_details where u_id=10\'

From this error the hacker can easily retrieve the stored table name user_details and the attribute u_id.

- **End of line comment:** In this technique the values are entered in the input field in such a way that rest of the query is treated as a comment. For example if the attacker knows the username but not the password then he can use this technique easily as shown in the Figure 2.2.

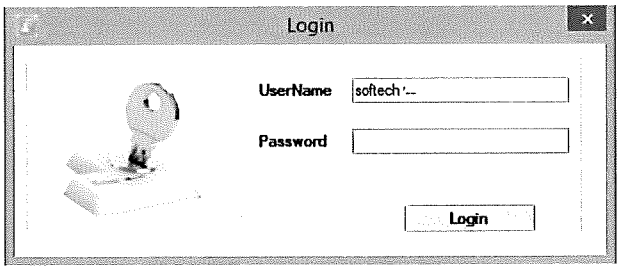


Figure 2.2: End of line comment

- **Timing attack:** In this type of attack, the attacker guesses the information character by character, depending on the output form of true/false. In time based attacks, attacker introduces a delay by injecting an additional SLEEP (n) call into the query and then observing if the webpage was actually by n seconds.
- **Union Query:** Injected query is concatenated with the original SQL query using the keyword UNION in order to get information related to other tables from the application. Original query: select acc-number from user_details where u_id='500' Injected query: select acc-number from user_details where u_id='500' union select pin from acc_details where u_id='500'.
- **Blind SQL injection attacks:** Attackers typically test for SQL injection vulnerabilities by sending the input that would cause the server to generate an invalid SQL query. If the server then returns an error message to the client, the attacker will attempt to reverse-engineer portions of the original SQL query using information gained from these error messages. The typical administrative safeguard is simply to prohibit the display of database server error messages. Unfortunately, that's not sufficient. If your application does not return error messages, it may still be susceptible to "blind" SQL injection attacks.
- **Piggy backed queries:** Additional query is added to the original query. This can be done by using a query delimiter such as ";", which deletes the table specified. Injected Query: select name from user_details where username ='softech'; drop table acc—

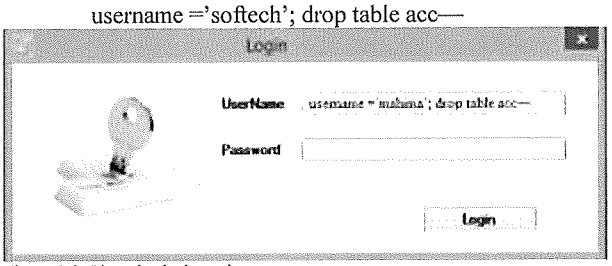


Figure 2.3: Piggy backed queries

4. RELATED WORK

J.Nataraj and P.Muralikrishnan proposed a pattern based query processing approach that can nullify the problem of web vulnerability. In the SQL injection are insert into the user requested queries and hack their authorized data without knowing the user. The second order SQL injection attack is the trickiest and emerging vulnerability to compromise the database. It start the malicious activity by inject it with other related queries that are already stored in the database whenever the second order SQLIA is invoked. In this paper, describe the three phases of input validation and also based the problem of injection attack explain with validation algorithm for each phases. From that, the injected queries are easily validated and reduce the false rate. It provides more security and strong input validation on the server. This induces to avoid the Second order SQL injection attack completely from the database. Even this framework is having more phases of validation[3].

S.Anjugam and A.Murugan proposes how to detect and prevent SQL injection attacks on web applications using encryption and tokenization technique. The tokenization process is applied on the input query by detecting spaces, single quotes and double dashes etc. This process converts the input query into fruitful tokens and that are stored in a dynamic table at the client side. The table name, field name and data are encrypted using AES algorithm .The encrypted the original input query and the tokenized table are sending to the server side. At the server side, input query is decrypted and in turn converts into various token which are stored in to another dynamic table. Both dynamic tables are compared and if both are equal, it seems that there is no injection attacked in the given query ,hence the query is proceed further to main database for retrieving result .If they are different, query is rejected and not forwarded to the database server[5]

4.1 PROPOSED SOLUTION

The proposed system is a simple and effective method to accurately detect and prevent SQLIAs by using a Combination of DDL & DML Mapping along with insertion of SQLIA detection snippets inside the web/application forms to detect and search data insertions and sniff SQLIA. In DDL & DML Mapping we generate a Sparse Image of the Schema Structure and Data contents of the SQL Queries implemented in the web/application forms from Database to a Mirror Database which will be stored in parallel. Along with the Sparse Image the Vectorization of the SQL Queries is also store in tables of the Mirror Database, to include different syntax, we resolve the parse tree of different generated queries. As a result of the output of the different generated queries, we can monitor the detection of abnormalities among the queries within our database structure; we are in forward for a resemblance monitoring for the space of controlled objects, i.e. the space of valid SQL parse tree structure. Thus, we strive with the problem of having to generate a comparison utility for matching trees.

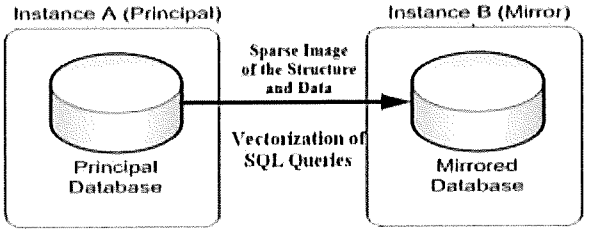
The Formula we generate is a Combination of DDL & DML Mapping along with Vectorization of SQL Queries.

In DDL & DML Mapping we generate a Sparse Image of the Structure and Data contents of the Database to a Mirror Database which will be stored in parallel.

Along with the Sparse Image the Vectorization of the SQL Queries is also store in tables of the Mirror Database, to include different syntax, we resolve the parse tree of different generated queries. As a result of the output of the different generated queries, we can monitor the detection of abnormalities among the queries within our database structure; we are in forward for a resemblance monitoring for the space of controlled objects, i.e. the space of valid SQL parse tree structure. Thus, we strive with the problem of having to generate a comparison utility for matching trees.

4.1.1 Data mirroring

Database mirroring is the creation and maintenance of redundant copies of a database. The purpose is to ensure continuous data availability and minimize or avoid downtime that might otherwise result from data corruption or loss or from a situation when the operation of a network is partially compromised. Redundancy also ensures that at least one viable copy of a database will always remain accessible during system upgrades. In DDL & DML Mapping we generate a Sparse Image of the Structure and Data contents of the Database to a Mirror Database which will be maintained in parallel. Along with the Sparse Image the Vectorization of the SQL Queries is also store in tables of the Mirror Database, to include different syntax, we resolve the parse tree of different generated queries. As a result of the output of the different generated queries, we can monitor the detection of abnormalities among the queries within our database structure; we are in forward for a resemblance monitoring for the space of controlled objects, i.e. the space of valid SQL parse tree structure. Thus, we strive with the problem of having to generate a comparison utility for matching trees.



4.1.2 Sql injection detection

There are two types of SQLIA Detection:

Static Approach: This approach is also known as pre-generating approach. Programmers follow some guidelines for SQLIA detection during web application development. An effective validation checking mechanism for the input variable data is also requires for the pre-generated method of detecting SQLIA.

Dynamic Approach: This approach is also known as post-generated approach. Post-generated technique are useful for analysis of dynamic or runtime SQL query, generated with user input data by a web application. Detection techniques under this post-generated category executes before posting a query to the database server [5]. Here I propose static approach to detect SQLIA.

Algorithm for SQL Injection Detection

Step 1: Gathering the information about the Database Schema.

Step 2: Store the Schema values into the Sparse Tables.

Step 3: Gather the Output of Randomly generated SQL Queries from various tables.

Step 4: Store the Queries and Output into Tables of Mirror Database.

Step 5: Search for the different sql queries which is coded inside the web forms.

Step 6: Store the SQL Query detected along with the position information.

Step 7: Confirm the Gathered information.

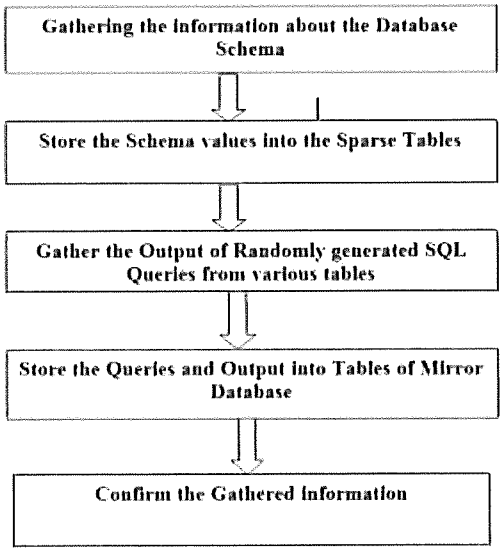


Fig: SQL Injection detection

4.1.3 Runtime SQLIA prevention

Researchers have proved that query injection can't be applied without using space, single quotes or double dashes (--). Researchers have developed tools and techniques that could prevent all SQLIAs by checking actual runtime against legitimate queries.

Learning-based prevention. This type of approach uses a runtime monitoring system deployed between the application server and database server. It intercepts all queries and checks SQL keywords to determine whether the queries' syntactic structures are legitimate (programmer-intended) before the application sends them to the database. It also cross check the queries with data dictionary and confirm that there is no possibility for an SQLIA.

Static Analysis. AMNESIA (Analysis for Monitoring and NEutralizing SQL Injection Attacks) uses static analysis to deduce valid queries that might appear at each database access point in Web programs via isolation of tainted and untainted data. Another runtime SQLIA prevention technique uses a query learning approach similar to AMNESIA, but, instead of targeting query statements in a server program, it targets stored procedures in a database and also crosscheck with the data dictionary generated.

Dynamic Analysis. Statically inferred legitimate query structures might not be accurate, and attackers could exploit this weakness to conduct SQLIAs. Researchers have thus proposed dynamic-analysis-based approaches to provide more accuracy. SQL Check tracks tainted data at runtime by marking it with meta characters and store the sql queries inside the Data dictionary. When a Web application invokes a query, SQL Checker learns the query's legitimate structure by verifying it with the data dictionary.

4.1.4 Applications

This paper can be referenced at several crucial areas in which the customer data security is important. Some of the Major areas are given below.

- Online Banking
- Online Shopping
- Social Networking
- Security System
- Education

5. RESULT AND EVALUATION

The proposed algorithm for the detection of SQL Injection can be implemented on real web applications. It can store total number of parameter in web applications and make a list to compare with the real time generated value of parameter. Also, it can profile legitimate dynamic query generated by normal users between the web application and the database server and compare it with the dynamically generated query to detect attacks.

This paper compares the detection rate of the proposed method with other researches under the same conditions. It also compares detection and prevention methods of particular attack forms.

It detects attacks by comparing the structure and the grammar of the query. If the dynamically generated query has a different structure or if it uses different grammar, it will be detected. The algorithm proposed in this paper does not use complex analysis methods such as Parse trees. It uses a very simple method which compares the queries after the removal of the attribute values. Therefore, it can be implemented into any type of DBMS.

(1) Table 1

SLNo	*File Name*	*Query*
1	Login.vb	Select username,password from tbl_Login
2	Home.vb	Select category from tbl_category
3	Home.vb	Select Date from tbl_datareport where category_id='2'
4	Vouchure.vb	Select * from tbl_vouchure where category_id='2' and date='date'

Structure of legitimate query

6. CONCLUSION

This paper proposes a new SQL injection attack detection method that utilizes both Static and Dynamic Analysis along with a data dictionary in parallel for data verification. SQL injection is a common technique hackers employ to attack underlying databases. These attacks reshape the SQL queries, thus altering the behavior of the program for the benefit of the hacker. In this paper, we present a fully automated technique for detecting, preventing and reporting SQLIA incidents in databases. In DDL & DML Mapping we generate a Sparse Image of the Structure and Data contents of the Database to a Mirror Database which will be stored in parallel. The proposed algorithm for the detection of SQL Injection can be implemented on real web applications. It can store total number of parameters and queries in web applications and make a list to compare with the real time generated value of parameter. Also, it can profile legitimate dynamic query generated by normal users between the web application and the database server and compare it with the dynamically generated query to detect attacks.

References

[1] Mahima Srivastava, "Algorithm to Prevent Back End Database against SQL Injection Attacks", Published in: Computing for Sustainable Global Development (INDIACom), 2014 International Conference. 978-93-80544-12-0/14/$31.00_c 2014 IEEE, Date of Conference: 5-7 March 2014, Page(s): 754 - 757

[2] Lwin Khin Shar and Hee Beng Kuan Tan, Nanyang, "Defeating SQL Injection", 0018-9162/13/$31.00 © 2013 IEEE Published by the IEEE Computer Society March 2013

[3] J.Nataraj and P.Muralikrishnan, "DVGAR: Distraction of Web Vulnerabilities to Provide Grasp Accessing Capability of The Web Resources", | ISSN: 2321-9939,

International Journal Of Engineering Development And Research | IJEDR
[4] Rajesh M. Lomtel , Prof. S. A. Bhura2 , "Survey of Different Web Application Attacks & Its Preventive Measures", IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 14, Issue 5 (Sep. - Oct. 2013), PP 46-51 www.iosrjournals.org
[5] S.Anjugam A.Murugan, "Efficient Method for Preventing SQL Injection Attacks on Web Applications Using Encryption and Tokenization", © 2014, IJARCSSE All Rights Reserved Page | 173 Volume 4, Issue 4, April 2014 ISSN: 2277 128X, International Journal of Advanced Research in Computer Science and Software Engineering.
[6] Debabrata Kar, Suvasini Panigrahi, "Prevention of SQL Injection Attack Using Query Transformation and Hashing", 3rd IEEE International Advance Computing Conference (IACC) pp. 1317-1323, 2013.
[7] Dr Amutha Prabakar, M.KarthiKeyan, Prof.K. Marimuthu. "An efficient technique for preventing sql injection attack using pattern Matching algorithm". 2013 IEEE International Conference on.M. Emerging Trends in Computing, Communication and Nanotechnology (ICECCN 2013).
[8] Sadeghian, Zamani.M., Ibrahim, S. "SQL Injection Is Still Alive: A Study on SQL Injection Signature Evasion Techniques", Date of Conference : 4-6 Sept. 2013, Publisher : IEEE.
[9] Joshi, Geetha.V , "SQL Injection detection using machine learning", Date of Conference : 10-11 July 2014, Publisher : IEEE.
[10] K. Beaver. Achieving sarbanes-oxley compliance for web applications through security testing. http://www.spidynamics.com/support/whitepapers/WI SOX white paper.
[11] Shruti Bangre, Alka Jaiswal.SQL Injection Detection and Prevention Using Input Filter Technique. International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-1, Issue-2, June 2012.
[12] Mihir Gandhi, Jwalant Baria, "SQL INJECTION Attacks in Web Application", International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-6, January2013

Author Profile

Biji.K.P received the B.Tech in Computer Science from Calicut University and undergoing Final Semester ME. Degrees in Computer Science and Engineering at Professional Group of Institutions. Worked as a Lecturer in Sreedevi Institute of Technology, Mangalore and Jawaharlal College of Engineering and Technology, Lakkidi, Kerala. Currently working as a consultant in Softech Infosystems, Shoranur, Kerala.