Engineering Universe for scientific Research and management
ISSN (Online): 2319-3069
Vol. 11 - Issue VIII
August 2019

# AN EMPIRICAL MODEL FOR WEB APPLICATIONS TO DETECT AND PREVENT SQL INJECTION ATTACKS (SQLIA)

**Arvind Kumar[1], Prof. Ritesh Kumar Yadav[2], Dr. Varsha Namdeo[3]**
[1,2,3] **Department of CSE, RKDFIST, Bhopal, India**
arvind86ojha@gmail.com

**ABSTRACT:**

The Web applications have developed into the most significant communication manner among various sorts of service suppliers and clients. It has evolved from a static medium, with user interaction limited to navigation among web application pages, to a highly interactive dynamic medium performing concurrent transactions and serving up personalized content. Several recent works attempt to develop a general and schematic methodology for automatically inferring the specifications for internet applications that in turn facilitates automatic and sound verification of application logic. During this research work we proposed "Improved Method to Detect and Prevent SQL Injection Attacks (SQLIA) in Web Applications". Nevertheless the suggested system will be likewise successful for both simple and complex data types. The proposed method is an advanced tool to discover and avoid attacks in web applications, this method is trouble free to implement and deploy. Throughout this scheme all the information validations rules are kept at a protected location. These information validation rules as well prepared into various XML format and that why we referred them as XML validation rules. At any time when server obtains any input from user, the server could validate the entire script maintained by the XML validation rules already stored inside the server. We conclude by highlighting the robust features of the efficient proposed method, which can detect the error during the development statically and can protect web applications from the future SQL injection.

**Key Words:** SQL injection, XML, WWW, Web Security, Web Application, Threats.

## 1. INTRODUCTION

The Web applications have developed into the most significant communication manner among various sorts of service suppliers and clients. It has evolved from a static medium, with user interaction limited to navigation among web application pages, to a highly interactive dynamic medium performing concurrent transactions and serving up personalized content. This dynamic medium application offers a wide range of services, such as on-line stores, e-commerce, social network services, etc. As progressively services are offered by the World Wide Web (WWW), attempts from both academia and business are determined to produce technologies and principles that convene the complicated constraints of today's venture Web applications and clients.

### 1.1 WEB APPLICATION CONSTRUCTION

A web application could be described through the number of layers that data will go by through on its journey from the database layer (where it is saved in a database server normally) to the presentation layer (where it is exhibited to the user). All layers normally execute on a dissimilar system or in a dissimilar procedure space on the similar system [1].

A client-server model was sufficient when information were mainly a text in the web applications. Inside the client-server model, the client and server are splitted by logics. This 2-layer client-server circumstances efforts very well for a little business that only utilizes, or requires, a single data source [2]. Nevertheless, the objective of the majority companies is to rise, and this wants extra logics to convene the business demands. Regrettably, the 2-layer model does not balance very well. If the business rules modify then the application requires be rebuilding and redeploying.

Because of the constraints of the 2-layer client-server model, distributed applications are frequently splitted into 3 or more layers [1]. Elements in every of these execute a precise sort of processing. There is a client Services (Presentation) layer, a Business Services layer, and a Data Services layer in a layer application, as shown in Figure 1.

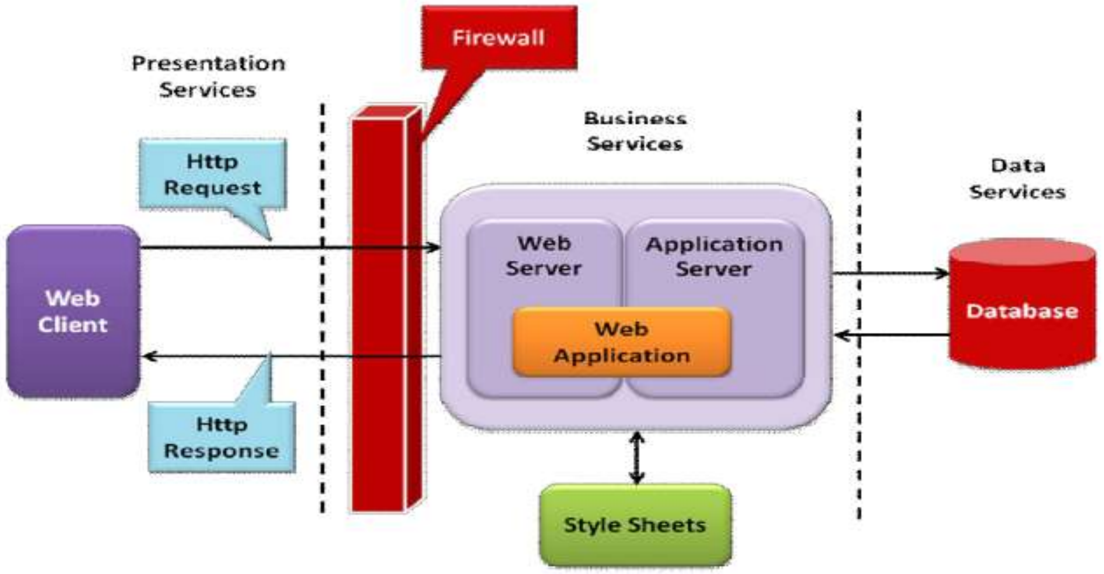

Figure 1: Three-layer web applications construction

The layer construction, business logic, is splitted from the client interface and the data source. Breaking up applications into these divide layers or segments could decrease the difficulty of the overall application, and this result in applications that could gather the growing requires of today's businesses.

## 2. WEB APPLICATIONS

The structure of the Web application is the integration of a Hyper Text Markup Language (HTML), a Client Script and a Server script that will be formatted and distributed from web servers to client systems and viewed through a web browser. Web applications are placed in an appropriate place in the web server which will receive document requests and data submission from web browsers through the Hyper Text Transfer Protocol (HTTP) top of the TCP/IP layer [3]. The main function of the web server is to feed HTML files to the web browsers. If the client requests an existing static file, it will be retrieved from the server’s hard disk and sent reverse to the web browser. The communication among the web browser and the web server is usually condition less. The intricacy of web based applications has as well developed considerably from information distribution to online operations, enterprise wide planning, scheduling systems and Web based joint work situations. Numerous characteristics of quality web based applications such as simplicity of navigation, accessibility, scalability, maintainability, usability, compatibility, inter operability, safety and consistency are not specified the deliberation they deserve throughout development. Various web applications also fail to address the intellectual, solitude, ethical, and lawful features. In order to deal with these major matters, all the web applications must have web safety features.

### 2.1 Web Application Modules

A module is a summarized part of rules that executes a definite function for a web application. This function might be the processing of a business rule, or it might be the recovery of some data from a database layer for an application. Usually, client Interface modules are at the presentation layer [4], business modules are at the business layer and database related modules are at the data access layer. The data centric module or the data access layer is liable for integrating with the data resources like Relational Database Management System (RDBMS) tools, which the application requires, to be capable to function. These data resources might be the SQL Server or Oracle databases, Exchange message stores, or UNIX legacy applications. It gives steady data access to dissimilar data sources, and creates the position of the data transparent. The business module or the business logic layer is proposed to cover the complex interactions that a set of business rules requires, in order to process and guard the client interface designer from having to recognize something about the underlying data. The requirement to utilize the business module is to summarize business rules within a module. The client interface module or the presentation layer presents the application content to the user which would be created by the business logic. Hence, an abstraction in the business logic module, data module and client interface module can be offered in the web application safety.

### 2.2 Web Functionality

Web applications employ numerous different technologies to deliver their functionality. Reasonable functional application may employ dozens of distinct technologies within its server and client modules.

#### 2.2.1 Server-Side Functionality

Whenever a user requests a resource, the server's response is created on the fly, and each user may receive content that is uniquely customized for him. When a user's browser makes a request for a dynamic resource, it does not normally ask for a copy of that resource. In general, it will also submit various parameters along with its request. These parameters enable the server-side application to generate content that is tailored to the individual user. There are three main ways in which HTTP requests can be used to send parameters to the application, namely, the URL query string, HTTP cookies and the body of requests using the GET / POST method. In addition to these primary sources of input, the server-side application may, in principle, use any part of the HTTP request as an input to its processing. Web applications employ a wide range of technologies on the server side to deliver their functionality. Some of them are given below

- Scripting languages, such as PHP, VBScript, JavaScript and Perl.
- Web application platforms, such as ASP.NET and Java.
- Web servers, such as Apache, Tomcat, IIS, and Netscape Enterprise.
- Databases, such as MS-SQL, Oracle, and MySQL.
- Other back-end modules, such as file systems, SOAP-based web services, and directory services.

All these technologies are to be properly integrated and configured to provide secure service to the web clients.

#### 2.2.2 Client-side functionality

In order to make the server-side application receive user input and actions, and present the results of these back to the user, it needs to provide a client-side user interface. Since all web applications are accessed via the web browser, all these interfaces share a common core of technologies. HTML forms are the usual mechanism for allowing users to enter arbitrary input via the browser. Hyperlinks and form elements can be used to create a rich user interface capable of easily gathering most kinds of input which web applications require. However, most applications employ a more distributed model, in which the client side is used not simply to submit the user data and actions but also to perform the actual processing of data.

A significant development in the use of JavaScript has been the appearance of Asynchronous JavaScript and XML (AJAX) techniques for creating a smoother user experience, which is closer to that provided by the traditional desktop applications [5]. AJAX involves issuing dynamic HTTP requests from within the HTML page, to exchange data with the server and update the current web page accordingly, without loading a new page altogether. These techniques can provide very rich and satisfying user interfaces. The integration of various complex technologies and an inadequate design may introduce vulnerabilities, which attract attackers to create threats in web applications. Most of the vulnerabilities in web applications are due to the poor development of web applications that continue to expand a high probability of low performance and/or failure.

## 3. LITERATURE REVIEW

A risk is any incident or occurrence with the possible cause to damage to a web application throughout the discovery, variation or ruin of data, or by the rejection of serious services. Risks could vary from gaining something of cost at no cost, to razing the important data or reputation of furthers. To provide the necessary security measures in web applications, security solutions need to be implemented by conducting a security risk assessment to identify the key issues and vulnerabilities. Based on the assessment, a security plan has to be developed that includes installing hardware devices in conjunction with security software to protect valuable assets in a web environment. The standard elements of the plan for perimeter security include firewalls, encrypting files and messages, Intrusion Detection Systems (IDS), Using digital signatures to protect transactions, Intrusion Prevention System, Protecting logs with immutable files, Demilitarized Zone (DMZ), Adopting advanced routing protocols and patch and vulnerability management systems. Unfortunately, the installed security elements do not provide the required solutions for the application layer intrusions.

### 3.1 Vulnerability Analysis and Scanner

The application layer protocol vulnerabilities are the access point for the intruders, and those intrusions are very hard to defend. Several tools and techniques have been developed to analyze the vulnerabilities in web-based applications. Implementing a daily vulnerability scan is one of the most effective ways in which a website owner could ensure the overall health of his website. It proactively identifies the vulnerabilities, lets the owner remove the questionable code, and helps to mitigate issues before cybercriminals exploit them. Reactive measures include quick identification of Zero-Day vulnerabilities [6], which affect a large number of websites in a short span of time. There is no patch available for this vulnerability. Even though a Zero-Day intrusion may occur, it is possible to identify where the compromised websites are extracting the intrusion information from, or where the malicious website visitors are being redirected to. Authors [7] have designed and developed a tool called SecuBat which is a vulnerability scanning tool, using black the box test approach and it automatically analyzes web applications in generic and specific manners to expose the SQL insertion and XSS vulnerabilities. This tool scans security flaws in web pages for exploitable vulnerabilities, using multi-threaded crawling, and intrusion and analysis components, equipped by a graphical user interface. Though the tool SecuBat emphasizes on creating various attacking vectors for detecting XSS vulnerabilities, it does not pay enough attention to detect SQL insertion vulnerabilities like blind SQL insertion and irrational Queries.

Scanning tools are useful in identifying the vulnerabilities in a web application. The vulnerability scanners must be properly designed and evaluated, so that these tools do not come up with false positives. Authors [8] proposed a method to evaluate and benchmark automatic web vulnerability scanners, using software fault insertion techniques. The most common types of software faults are inserted in the web application code, which is then checked by the scanners. The results are compared by analyzing the coverage of vulnerability detection and false positives. The author evaluated the leading three commercial scanning tools, and the results show, that in general, the coverage is low and the percentage of false positives is very high. In addition, authors [9] have conducted a case study with three different scanners for detecting web application vulnerabilities, and investigated the effectiveness of using the Application Vulnerability Description Language (AVDL) to describe the vulnerabilities and their contribution in developing a unified data model used a technology independent, rule-based solution for the vulnerability analysis of web-based applications. But automatic web vulnerability scanners are available, that could help to locate the vulnerabilities, and are popular tools among developers of web applications.

The purpose of the vulnerabilities scanning tool is to stress the application from the intruder's point of view, by issuing a large number of interactions within it. Trusting the results of web vulnerability scanning tools is of utmost importance. Without a clear idea of the coverage and false positive rate of these tools, it is difficult to judge the relevance of the results they provide. Furthermore, it is difficult, if not impossible, to compare the key figures of the merit of web vulnerability scanners. Hence, web application scanners are helpful to some extent to identify the vulnerabilities in web applications.

### 3.2 SQL Injection

Command Execution is a common type of web application threat in the World Wide Web. Dangerous intrusions under command insertions are SQL insertion, XPath insertion, Buffer Overflow, Light weight Directory Access Protocol (LDAP) Insertion and Format string intrusion. Statistically, the SQL Injection (SQLI) vulnerability is a serious and widespread security vulnerability of web applications. In this literature review, a detailed study has been conducted, to analyze the different kinds of SQL injection detection and prevention strategies, proposed by various authors. Many researchers and practitioners are familiar with only a subset of the wide range of techniques available to intruders who try to take advantage of SQL injection vulnerabilities. As a consequence, many of the solutions proposed, address only limited issues related to SQL injection. To solve this problem, [33] made an extensive review of the different types of SQL injection intrusions, known to date. For each type of intrusion, they provided descriptions and examples of how intrusions, of that type could be made. They also presented and analyzed the existing detection and prevention techniques against SQL injection intrusions. They discussed each technique, and its strengths and weaknesses in addressing the entire range of SQL injection intrusions. The various strategies and mechanisms proposed by different authors to detect and prevent SQL injection intrusions are shown in Figure 2.

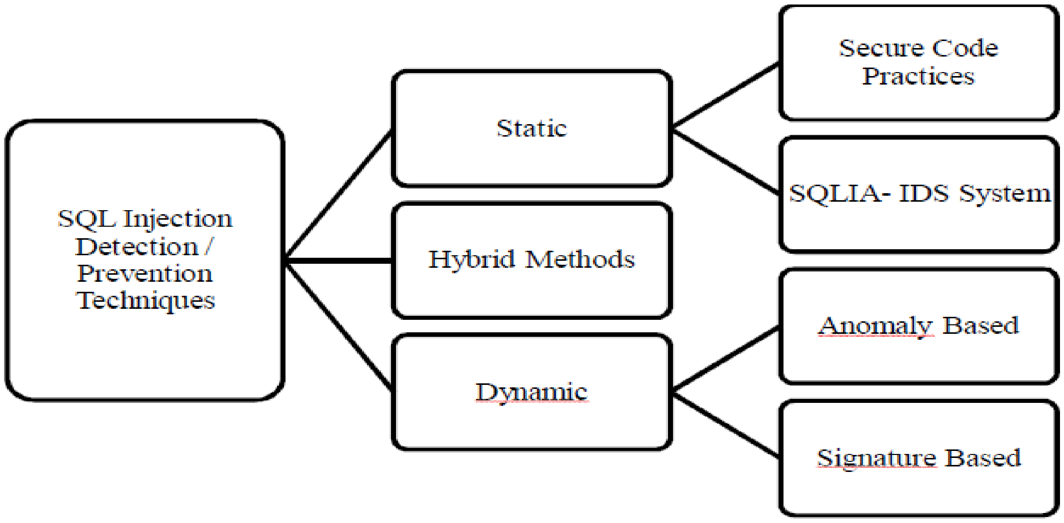

Figure 2: Prevention strategies of SQL injection intrusions

## 4. PROPOSED WORK

We select SQL injection in web applications for several reasons. Building a web application is not so typical these days. Mainly a user who has some knowledge of programming may well create a web application practically with any trouble. Nevertheless, it is so simple to create web applications without any protection vulnerabilities. In a survey of "Hewlett-Packard (HP) in 2011 topmost cyber safety threats report," all the considered web applications have at least one or more susceptibility. One of the main causes is that a programmer doesn't give adequate awareness to the safety element because of alternate causes such as lack of knowledge, deadline stress, and resources restriction etc.

### 4.1 Proposed Solution

During this research work we proposed "Improved Method to Detect and Prevent SQL Injection Attacks (SQLIA) in Web Applications". Nevertheless the suggested system will be likewise successful for both simple and complex data types. The proposed method is an advanced tool to discover and avoid attacks in web applications, this method is trouble free to implement and deploy. Throughout this scheme all the information validations rules are kept at a protected location. These information validation rules as well prepared into various XML format and that why we referred them as XML validation rules. At any time when server obtains any input from user, the server could validate the entire script maintained by the XML validation rules already stored inside the server.

Typical dissimilarity among proposed method and other SQL injection attack detection and prevention techniques is that our proposed method authenticates the entire input wed application information simultaneously in one execution. Furthermore the XML validation rules are designed and saved on a particular basis that builds the programmer task easier, they have to do is just to judgment the authentication task for validating any web application information. Right now few validation functions are written for some web applications. This validation information could authenticate the input scripts maintained the basis on which they designed within the XML validation rules. For all application the input XML validation scripts could have registered design for various client appeals. XML validation rules are designed on a particular base for all sorts of input XML codes and therefore the input XML code must do well the authentication process. This process could mainly separate the information authentication of a web application from the web application development cycle. The programmer currently should not to concern about the SQLIA and data legitimacy. The authentication phases of data are sustained through a separate cluster which might manage the XML VALIDATION rules. This is frequently conjointly valuable as a result of the traditional web programmers are going to be entirely uninformed about the protection rules of the web application.

The benefit of proposed technique is clarified as follows:

- No requirement to send many appeals to the server, we could send all the web application information inside one XML format file.
- Server side has validation rules to authenticate each XML format file. The validation rules are called data validation rules. These rules are designed and saved at protected location in the server.
- At any time a server obtains client's appeal in XML file format, it will parse the input XML file and authenticate it beside the validation rule designed in data validation rules. This will create the data validation system standard and there is no requirement to design code for data validation for each and every web page. Only we require to call the XML validation function.

Just like the SQL Injection, the XPath Injection attacks arise when a web application utilizes client supplied information to create an XPath query for XML data file. By sending deliberately abnormal information inside the web application, an attacker might discover how the XML data file is designed, or obtain data that he might not usually have access to. He might even be capable to raise his rights on the web application if the XML data file is being utilized for validation. Querying in XML is complete with XPath, a sort of easy expressive declaration that permits the XML query to place a part of information. Like SQL, you might give definite attributes to discover, and patterns to compare. When utilizing XML for a web application it is ordinary to recognize several form of input on the query string to recognize the content to place and exhibit on the page. This input must be clean to authenticate that it doesn't mess up the XPath query and return the incorrect data. Our proposed method will be hold such kind of matters strongly and verifies that such attacks could not modify the XML VALIDATION rules designed in the server.

**Technical Details of XML VALIDATION SQL**

This section contains discussion about the significant components of XML VALIDATION SQL. The understanding of the practical information of XML VALIDATION SQL lies on the understanding of such components. Here we clarify them briefly.

- Working of Method: As clarified previously, in XML VALIDATION SQL data authentication occurs within the database server. In database, we have designed a stored procedure which recognizes every input data mutually as a XML file format and authenticate them against the validation rules stored surrounded by the database. Therefore onward, we would identify this stored procedure as VALI-SP. At any time application server requires to authenticate any data, it would create an XML file format for every such data and give it to the stored procedure VALI-SP. The VALI-SP in database would authenticate all data values against the equivalent validation rules.
- XML file format: we utilize two XML file formats: first is utilized for client data input into the VALI-SP, another is utilized by VALI-SP to give error note to the application server.

In this XML file format, for each input data there are two elements:

- <data-value>
- <data-type>

The <data-value> includes the value of data entered through the client whereas <data-type> includes the name of data validation rule which must equivalent the value of data in the authentication procedure. The <data-type> includes just the name of the data validation rules, the real validation rule is designed via regular language and saved within the database.

**Type of Data:** type of Data doesn't signify the common types of data of programming languages. Here types of data denote some pre described validation rules based on which client data should be authenticate. The types of data are saved in database. The table name given to these types of data is "certificate", in database which holds every the data types. The description of the table is as follow:

- Name: it is an exclusive name the type of data.
- Rule: designed in regular expression for the type of data.
- Error message: it is the message that will launch to application server if the validation rules fails.

The types of data which are by default obtainable in database server (table "certificate") are described below:

- INT: any Integer digit.
- NUM: Any digit with decimal point.
- STRING: Any string includes alphabet of English language, spaces and number.
- PINCODE: Postal pin code format.
- ALPHA: A solitary English alphabet word.

If we wish to add some fresh type of data than it might be inserted into the database but the validation rules must have to design appropriately. To insert a fresh type of data the programmer must know the basic perceptive of the MySQL syntax based on regular expressions.

**XML creator:** This element obtains all input information and data type id's as input and creates an XML from above inputs. The organization of the XML format is clarified in previous Para. After creating the XML, it executes the stored procedure (VALI-SP) via passing the created XML as a constraint. The complete element is designed in the form of function.

The stored procedure (VALI-SP) gives three constraints:

- Result: that provide info about the complete data authentication succeeded or not.
- UID: If the authentication succeeded then it would also create one unique id (UID) which might be utilized for more protected communication with the database server. The idea of UID will be conferred in afterwards.
- Error Message: If the data authentication be unsuccessful for several data, then the error messages of every such type of data would be generated jointly as an XML file. The particulars would be clarified afterwards.

**Stored Procedure (VALI-SP):** this procedure is designed for MySQL (expendable for other databases according to need). It receives an XML file formatted input from the client and executes data authentication for the data includes in the XML. The entire organization of the XML file is previously clarified. The job of VALI-SP is described step-by-step below:

- First Parse the XML file.
- Authenticate all data into the XML: As we described previously, all data in the input XML file has two components: <data-value> and <data-type>. <data-type> is utilized to obtain the validation rules designed for the database server for that particular data type.
- In authentication, if any data be unsuccessful to assure the equivalent validation rules then the resultant error message for that validation rule would be adjoined to the error message. The error message is as well designed in XML format and would be conversed afterwards.
- If authentication succeeded then a 38 bit unique id (UID) would produced and stored into the database server. The function of this UID is described afterwards.
- The stored procedure doesn't give any output, in fact it executes a choose query immediately before terminating. This choose query produces an outcome set for 'result xml', 'result' and 'unique id'. Here 'result' gives either 0 or 1 depending on wither the data authentication processes be successful entirely or not. 'result xml' is the error memo and would be blank if 'result' encloses 1. 'unique id' includes the created unique id and would be blank if 'result' encloses 0.

**Principle of UID:** The VALI-SP produces a UID if the data authentication procedure succeeded. This UID is saved in the database server and also give back to the application server. The principle of UID is simply to compose the additional protected database server communication. It might be probable that an attacker can crack the security protocol of the application server. In that case they have to eliminate the XML VALIDATION SQL based authentication from the designed application code. If they try to crack security then the data authentication based on XML VALIDATION SQL would be avoided and the SQL injection would be penetrated into the database server. To avoid these types of detour, UID might be utilized. Throughout the database server function the UID must require to pass to the database server. In database server the UID passed by the client would be evaluated with the UID collected within the database server and only agree to to progress the operation if both equivalents. in its place of executing SQL statements straight from application server it is suitable to employ them stored procedure for operations of database. In this manner the UID examination will be easier for the database server.

**Error Message format:**

The layout of the XML created by VALI-SP for error messages is described below:

```
<result>
<error_message>
<data_id>1</data_id>
<message>Only integer digit recognized.</message>
</error_message>
<error_message>
<data_id>2</data_id>
<message>Only integer digit recognized.</message>
</error_message>
</result>
```

It is depends on the application programmer that how to parse and employ these kind of error messages.

**Installation of XML VALIDATION SQL:**

Previously we discussed about XML VALIDATION SQL, it composes the web application programmer work easier. Programmers not require being anxious about the defense rules since the rules would be design by XML VALIDATION SQL. They simply require to utilizing those rules by using the id of the equivalent rule. Also programmer might not observe the rules design in the database server. In this manner the application would be more protected since even programmer won't have comprehensible awareness about the authentication rules.

For installing XML VALIDATION SQL into your system, just follow these simple steps:

- Authentication Database: If your web application has a predefine database then immediately execute the "verify.sql" into that database. After executing the SQL script it would produce two charts (certificate chart and key checker chart) and one stored procedure (VALI-SP) in that database. And if you don't contain any database previously then build one database initially and then execute the verify.sql script on it.
- Verify.web: Store this web file someplace in your application system so that you could access it on requirement. This file includes a function validate with XML which efforts as XML creator as clarified in previous section. Also you require modifying the 4 invariables designed at the top of the file based on MySQL database.

That's it, now your web application is wholly prepared to utilize XML VALIDATION SQL.

**Manual of XML VALIDATION SQL:**

When installation is completed and you want to utilize XML VALIDATION SQL in your web application. Do these simple steps:

- Obtain all the client inputs (either by POST or GET method). In PHP, we usually do it similar to $data1=$_POST ['data-field1']; this element is universal and have to perform it to get back the client input.
- After getting all the information, authenticate the suitable data type (name of the data type) for all data. If at present accessible data types are not accurately fitting on constraints then communicate with management to generate fresh data types in XML VALIDATION SQL database server.
- Now call the verify function Where typei is the type of data name of the ith data and data1 is the genuine content of i$^{th}$ data.
- As clarified previously, the function would return the outcome as "ERROR" or "VERIFIED". Also the function stored two values into session: error_message and uid.

**5. RESULT ANALYSIS**

We developed a TESTBED, which is coded in JAVASCRIPT and able to run SQL queries of web Applications. To create Client server environment in PHP web application we employed XAMPP 5.6.23 version. XAMPP 5.6.23 is open source cross platform for web server solution stack package, consisting mostly of the Apache HTTP Server, My SQL database server with interpreter for codes programmed in the PHP and Perl. For the result analysis of proposed technique we developed a TESTBED, which is programmed in JAVA language and capable to execute SQL queries of any web based Applications. To generate Client Server background in web based application which are developed in PHP, we utilized XAMPP 5.6.23 version for this purpose. XAMPP 5.6.23 is a tool, which is open source platform for web server explanation stack package, having mostly of the HTTP Apache Server, database like My SQL and analysts for scripts coded in the Perl and PHP programming languages.

Table 1: Explanations of Web Application used for experiment

| Web Application Name | Description of web Application |
|---|---|
| Management Information System | A web based application for members of staff in organization is coded in PHP |
| Digital Library | A online library computerization based application coded in PHP |
| Project manager | A academic web application for supervision of project written in PHP |
| Shop_ Online | A web application written in PHP for shopping online |
| Net medicine | An Web application for online medical shop written in PHP |

To estimate the efficiency of proposed method the TESTBED is utilized. The developed TESTBED has been utilized for evaluating a variety of PHP web applications given in table 1. It inspected so many test cases on an open source web based applications. It includes two kinds of test cases: the XML laws from which SQL queries gain validation and another is to return error note via the server. Table 6.1 reviews the various Web applications which are developed in PHP.

The Proposed Tool experimented on some PHP based web applications (descript in table 1) and results demonstrates that proposed Tool efficiently identified and stops vulnerabilities. Table 2 illustrates these susceptible web applications. The first column explains the Web applications name, the second column explains total number of input for every web application, third column explains total malevolent inputs, fourth column signifies total assaults identified and the fifth column explains the recognition rate.

Table 2: Results analysis of proposed method on TESTBED

| Application | Total inputs | Total Malevolent inputs | Total Assaults identified | Rate of recog nition |
|---|---|---|---|---|
| **Management Information System** | 200 | 75 | 75 | 100% |
| **Digital Library** | 250 | 92 | 91 | 98.9% |
| **Project manager** | 300 | 157 | 157 | 100% |
| **Shop_ Online** | 220 | 87 | 87 | 100% |
| **Net medicine** | 225 | 87 | 87 | 100% |

## 6. CONCLUSION

In this work, we have converse about the concept of SQL injection which has become a common problem of all the web-based applications. We have summarized a variety of techniques existing to protected data from SQL injections, which can modify the program behavior permitting the attacker to retrieve and modify the database information. In this research work we present a method called “Improved Method to Detect and Prevent SQL Injection Attacks (SQLIA) in Web Applications” which can detect the vulnerability spots in the source code. Also, it provides the developer with an additional option of checking the syntax. Detection is done by validating the SQL queries using general validation procedure based on XML rules and the nature of the injection type. Any possibility of vulnerability or violation of the analyzed nature is reported as a warning message or error message to the developer.

The warning message or error message includes the injection type description and the line number of vulnerable spot in the source code. The concepts explained in this work assist the Developer to modify the SQL statements and make the code attack free. We conclude by highlighting the robust features of the efficient proposed method, which can detect the error during the development statically and can protect web applications from the future SQL injection.

We believe that the ideas presented in this research work can be further extended to include new injection types. This work also paves way for the development of vulnerability detection services, which can be used by developers to detect vulnerability spots in the source code. We feel the area of SQL injection vulnerabilities is wide open for research and we conclude by suggesting that this step to verify the code against SQL injection vulnerabilities can be added in the checklist for performance review in the static code analysis of the source code of data driven applications.

## REFERENCES

[1] Noack, J., Mehmaneche, H., Mehmaneche, H., and Zendler, A. “Architectural Patterns for Web Applications”, In Proceedings of the 18th IASTED International Conference on Applied Informatics, Innsbruck, Austria, ACTA Press, 2000

[2] Lieven Desmet, Bart Jacobs, Frank Piessens and Wouter Joosen “A Generic Architecture for Web Applications to Support Threat Analysis of Infrastructural Components”, Communications and Multimedia Security, IFIP- The International Federation for Information Processing, Springer, Vol.175, pp.125-130, 2005.

[3] Andrews, J. “Understanding TCP Reset Attacks”, KernelTrap, KernelTrap.org, Florida, USA, 2006.

[4] Kadri, Reda, Chouki Tibermacine, and Vincent Le Gloahec. "Building the presentation-tier of rich web applications with hierarchical components." InWeb Information Systems Engineering–WISE 2007, pp. 123-134. Springer Berlin Heidelberg, 2007.

[5] Zepeda, J.S. and Chapa, S.V. “From Desktop Applications Towards AJAX Web Applications”, 4th International conference on Electrical and Electronics Engineering, IEEE, pp.193-196, 2007

[6] Inghama, K.L., Somayajib, A., Burgea, J., and Forresta S. “Learning DFA representations of HTTP for protecting web applications”,Computer Networks, Elsevier, Vol.51, No.5, pp. 1239-1255, 2007

[7] Kals, S., Kirda, E., Kruegel, C., and Jovanovic, N. "SecuBat: A Web Vulnerability Scanner", 15th International Conference on World Wide Web ACM, Edinburgh, Scotland, pp.247-256, 2006.

[8] Fonseca, J., Vieira, M. and Madeira, H. "Testing and comparing web vulnerability scanning tools for SQL injection and XSS attacks", 13th IEEE Pacific Rim International Symposium on Dependable Computing Conference, Melbourne, Victoria, Australia, pp.365-372, 2007.

[9] Pushpraj Tanwar, Ajay Somkuwar, Hard component detection of transient noise and its removal using empirical mode decomposition and wavelet-based predictive filter, IET Signal Processing, 12(7), 2018, pp. 907-916.

[10] Sandeep Rajurker, Anurag Jain, Management Information Systems Over Social Network Analysis: Towards Web Personalization, EUSRM, 9(3), 2017.

[11] Le, H.T. and Loh, P.K.K. "Evaluating AVDL Descriptions for Web Application Vulnerability Analysis", in IEEE International Conference on Intelligence and Security Informatics, Taipei, Taiwan, pp.279-281,2008.

[12] Halfond, W.G, Viegas, J. and Orso, A. "A classification of SQLinjection attacks and countermeasures", In Proceedings of the international symposium on secure software engineering, 2006