International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 03 | Mar 2021 www.irjet.net p-ISSN: 2395-0072

SQL Injection Attack on Web Base Application: Vulnerability Assessments and Detection Technique

Alphonsus O. Agbakwuru[1], Donatus O. Njoku[2]

[1] *Department of Computer Science, Imo State University, Owerri, Imo State-Nigeria*
[2]*Department of Computer Science, Federal University of Technology, Owerri, Imo State-Nigeria*

Abstract - *An Web application developers are still busy writing insecure code that only get fixed when they are noticed or become a problem. This has given room to structured Query Language Injection attack (SQLIA) that result to stealing sensitive information or bypassing authentication procedures on Web based data. Notwithstanding billons of dollar expenses on various mitigations process already in place, yet for the last 15 years SQLIA has always been rated the number one on the OWASP top ten web attack mechanisms employed by hackers to steal data from organizations' database. It is on record that the majority of actual hacks reported by the press of hackers stealing credit card numbers/ private information etc., are in fact successful SQL injection attacks. Again the fact remains that as the countermeasures become more sophisticated, different forms of SQLIA also continues to evolve, thus thwarting the attempt of eliminating it completely. SQLIA have no specific pattern hence the major difficulty is in detecting and militating against it. The objective of this work is to develop Web based detection software capable of tackling the current Web based data security challenges. Using a hybrid of structured systems analysis and an Object Oriented Methodology, it was able to study and understand most of the current SQLIA. Again using fuzzy rule-based classification system (FRBCS) that adopted genetic fuzzy system for SQLI detection where not only the accuracy is a priority, but also the learning and the flexibility of the obtained rules, a top-down software design approach, a web based application software has been written in C# programming language that runs on MY SQL as backend database. This has been evaluated using a number of well-known malicious data sets. The result is a tool which has been able to restore security in our web based transactions that assures confidence, transparency, integrity, and privacy in our transactions. Hence, recommended for implementation to government and organizations that depend on web based data storage for their operation. For IT based service providers, is a guide in uncovering design flaws, providing sanity checks for programmers and system designers conformity with existing Software standards.*

***Key Words*: Database, Detection, MY SQL, SQLIA, Web**

1. INTRODUCTION

Structured Query Language (SQL) is the computer language that permits the storage, manipulation, and retrieving of data stored in an organizational database (or a collection of tables which organise and structure data). SQL is actually the only means that a web application (and users) can interrelate with the database. However, SQL injection has turn out to be a common problem with database-driven web sites. SQL injection attack comprises insertion or "injection" of a SQL query via the input data from the client to the application. A successful SQL insertion take advantage of being able to read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system. Today SQL Injection is one of the many web attack mechanisms used by hackers to steal data from organizations. It is perhaps one of the most common application layer attack techniques used today. It is the type of attack that takes advantage of improper coding of the web applications that allows hacker to inject SQL commands into say a login form to allow them to gain access to the data held within a database.

Web applications and databases allow individual to regularly do business. Such features as login pages, support and product request forms, feedback forms, search pages, shopping carts and the general delivery of dynamic content, shape modern websites all provide businesses with the means necessary to communicate with prospects and customers. The problem here is that these website features are all susceptible to SQL Injection attacks which arise because the fields available for user input allow SQL statements to pass through and query the database directly. SQL Injection is the hacking technique which attempts to pass SQL commands (statements) through a web application for execution by the backend database.

This study evaluates how the databases work and how an intruder use SQL injection to attack these databases and the type of attacks. The information from these evaluations of strategies now forms the bases for providing solution to reduce these attacks on the databases use in commercial wireless computing.

SQL language can be classified into two types which are data manipulation language and data definition language.

2. THEORETICAL FRAMEWORK

A. *Structured Query Language*

Structured Query Language is the American National Standards Institute (ANSI) 1986, and of the International Organization for Standardization (ISO) in 1987, special-purpose programming language designed for managing data held in a relational database management system (RDBMS). It is a standardized query language for requesting information from a database.

Each type of web application is hard coded with specific SQL queries that it will execute when performing its legitimate functions and communicating with the database. If any input field of the web application is not properly sanitised, a hacker may inject additional SQL commands that broaden the range of SQL commands the web application will execute, thus going beyond the original intended design and function.

SQL language can be classified into two types which are data manipulation language and data definition language.

- Data Manipulation language (DML): Data manipulation language contains the commands that are used to perform operations like Selection, Insertion, and deletion of records in database tables. There are four types of data manipulation commands namely, select, insert, delete, and update. Select is a command used to select a particular information from database. Insert is mainly used to insert records into table like new records. Delete is used to delete records from inside database. Update is used to update existing records in the database with new value. The most common operation in SQL is the query, which is performed with the declarative SELECT statement.

- Data definition language (DDL): Data definition language statement is used to create new structures for the records. These commands are used to perform operations against the structure of the database. The SQL statements are formed with one or more of these command that are defined by data definition language. DDL manages table and index structure. The most basic items of DDL are the CREATE, ALTER, RENAME, DROP and TRUNCATE statements. CREATE: creates an object (a table, for example) in the database. ALTER: modifies the structure of an existing object in various ways, for example, adding a column to an existing table or a constraint. TRUNCATE: deletes all data from a table in a very fast way, deleting the data inside the table and not the table itself. It usually implies a subsequent COMMIT operation, i.e., it cannot be rolled back (data is not written to the logs for rollback later, unlike DELETE). DROP: deletes an object in the database, usually irretrievably, that is, it cannot be rolled back.

- Data Control Language: Data Control Language (DCL) authorizes users to access and manipulate data. Its two main statements are: Grant, which authorizes one or more users to perform an operation or a set of operations on an object. REVOKE, which eliminates a grant that may be the default grant.

- Data types: Each column in an SQL table declares the type(s) that column may contain. ANSI SQL includes the following data types. (a) character strings like CHARACTER*(n)* or CHAR*(n)*,CHARACTER VARYING*(n)* OR VARCHAR*(n)*,NATIONAL CHARACTER*(n)* or NCHAR*(n)*, and NATIONAL CHARACTER VARYING*(n)* or NVARCHAR*(n)* (b) bit strings like BIT*(n)*, BIT VARYING*(n)* (c) numbers like INTEGER and SMALLINT, FLOAT, REAL and DOUBLE PRECISION, NUMERIC*(precision, scale)* or DECIMAL *(precision, scale)*.

B. *Security of Databases in Web base Application*

Database is a collection of data or records stored in the computer in a structured format. Database nowadays are in relational database format, where the data in a table is mutually related to data in another table. This model is used to reduce the complexity to gather records from database. Databases contain tables with unique names with respect to database, each table with unique names with respect to the database; each table contains records of data.

Databases are fundamental components of Web applications. Databases enable Web applications to store data, preferences and content elements. Using SQL, Web applications interact with databases to dynamically build customized data views for each user. A common example is a Web application that manages products. In one of the Web application's dynamic pages (such as ASP), users are able to enter a product identifier and view the product name and description. The request sent to the database to retrieve the product's name and description is implemented by the following SQL statement.

The security model used by many Web applications assumes that an SQL query is a trusted command. This enables attackers to exploit SQL queries to circumvent access controls, authentication and authorization checks. In some instances, SQL queries may allow access to host operating system level commands. This can be done using stored procedures. Stored procedures are SQL procedures usually bundled with the database server. For example, the extended stored procedure xp_cmdshell executes operating system commands in the context of a Microsoft SQL Server. Using the same example, the attacker can set the value of Product ID to be "123;EXEC master..xp_cmdshell dir--", which returns the list of files in the current directory of the SQL Server process.

- Prevention: The most common way of detecting SQL injection attacks is by looking for SQL signatures in the incoming HTTP stream. For example, looking for SQL commands such as UNION, SELECT or xp_. The problem with this approach is the very high rate of false positives. Most SQL commands are legitimate words that could normally appear in the incoming HTTP stream. This will eventually case the user to either disable or ignore any SQL alert reported. In order to overcome this problem to some extent, the product must learn where it should and shouldn't expect SQL signatures to appear. The ability to discern parameter values from the entire HTTP request and the ability to handle various encoding scenarios are a must in this case. There are certain measures to prevent SQL injection attack summarize in Table 1. Also avoidance techniques can be used as well.

Table -1: Preventing sql injection attacks (OWASAP 2015)

Principle	Implementation
Never trust user input	Validate all textbox entries using validation controls, regular expressions, code, and so on
Never use dynamic SQL	Use parameterized SQL or stored procedures
Never connect to a database using an admin-level account	Use a limited access account to connect to the database
Don't store secrets in plain text	Encrypt or hash passwords and other sensitive data; you should also encrypt connection strings
Exceptions should divulge minimal information	Don't reveal too much information in error messages; use custom Errors to display minimal information in the event of unhandled error; set debug to false

- Avoidance Techniques: While it remains obvious that an attacker must possess at least some knowledge of the database architecture in order to conduct a successful attack, obtaining this information is often very simple. For example, if the database is part of an open source or other publicly-available software package with a default installation, this information is completely open and available. This information may also be divulged by closed-source code - even if it's encoded, obfuscated, or compiled - and even by your very own code through the display of error messages. Other methods include the user of common table and column names. For example, a login form that uses a 'users' table with column names 'id', 'username', and 'password'. These attacks are mainly based on exploiting the code not being written with security in mind. Never trust any kind of input, especially that which comes from the client side, even though it comes from a select box, a hidden input field or a cookie. The first example shows that such a blameless query can cause disasters. Hence, the following should be adhered to. (a) Never connect to the database as a super user or as the database owner. Use always customized users with very limited privileges. (b) prepared statements with bound variables. They are provided by PDO, by MySQL and by other libraries. (c) Check if the given input has the expected data type. PHP has a wide range of input validating functions, from the simplest ones found in Variable Functions and in Character Type Functions (for example is_numeric(), ctype_digit() respectively) and onwards to the Perl compatible Regular Expressions support. (d) the application waits for numerical input, consider verifying data with ctype_digit(), or silently change its type using settype(), or use its numeric representation bysprintf().

C. *Database Administration*

Data base administrator takes very important role to avoid attacks against database servers. The administration deals with maintaining integrity, and security to the records. If proper planning is not done when creating the application it can do severe damage to the organisation. Hence the duty of the administrator will include:

- Analyse the present state of security present by performing a thorough audit of the website and web applications for SQL Injection and other hacking vulnerabilities.
- Making sure that you use coding best practice, sanitising your web applications and all other components of your IT infrastructure.
- Regularly performing a web security audit after each change and addition to your web components.

Emphasis when checking for SQL Injection and all other hacking techniques will be: "Which parts of a website thought to be secured are open to hacker attacks?" and "what data can we throw at an application to cause it to perform something it shouldn't do?". Hence the software will indicate which URLs/scripts are vulnerable to SQL injection so that we can immediately fix the code.

D. *Proposed Approaches to SQL Injection*

Table 2 below presents a brief description of some SQL injection approaches as proposed by different authors.

3. SYSTEM DESIGN

Table 2: shows different proposed approaches of SQL injection (Bharti Nagpal et al., 2013), The purpose of this session is to design an IDS corporation's databases from malicious attacks and maintain the integrity of sensitive information. The software should be capable of detecting the type of intrusion behavior and deploy the appropriate detection technique for the right system.

SQL injection is an application layer problem that can be prevented by taking good care in the application design and protecting every function and procedures that are used by web applications. If the application designer takes good care about protecting every single SQL statement that is dynamically generated by web applications an SQL issues can be a forgone problem.

Table -2: SQL injection approaches (Bharti Nagpal et al., 2013)

Author (s)	Proposed Approach
Sharma et al. (2012)	A query model generator based on hybrid approach for PHP application provides cent percent detection for known attacks.
Jiao et al. (2012)	Enforcing authentication by introducing a middleware ie. SQLMW in the middle. It implements hash mechanism which is faster than any other encryption.
Dharam and Shiva (2012)	Injection detection on the basis of identified valid path and critical variables.
Balasundram and Ramaraj (2011)	RSA and AES encryption used to enforce login query formation.
Zhang et al. (2011)	Duplicate database and queries into a LDAP database which is platform independent and doesn't propose any change in legacy web applications
Pomeroy and Tan (2011)	Network forensics are used to analyse recorded network packets. It uses network based IDS

A. Design Objectives

SQL injection can be prevented by taking good care in the application design and protecting every function and procedures that are used by web applications. The application designer must take good care about protecting every single SQL statement that is dynamically generated by web applications. If the statements are not well protected by the web application designer, a single statement can cause great damage to the database. Strings for passing the usernames and passwords to the databases must be created carefully with bind variables. If the SQL is not crafted carefully using the bind variables, the value from the user is passed as a form of strings to the database this makes the attackers work easy to execute malicious SQL over the database. And the SQL statements should be created such as there is no concatenation in between SQL strings and parameters.

The Input against the database must be validated properly to stop attacks against the database. If the input is validated properly it is possible for the administrator to find malicious code that the attacker is trying to use against the databases. By using the special characters like single quotes the attacker can successfully enter into the database. It is recommended that all the special types of characters must be validated properly before the application passes them into the database.

Hence the objectives are to study various SQL-Injection Attack behaviors and be able build an SQLIA IDS using Genetic Fuzzy Rule Base system capable of producing optimum solution in the mitigation of SQLIA on web base applications without having to reprogram the existing systems or increase overhead cost for the use of the software. It will also seek to answer the following research questions:

How to secure the SQL-Injection Attack IDS itself?

How to improve the effectiveness of the SQL-Injection Attack IDS?

And how to improve the efficiency of the new built SQLIA intrusion detection software?

Thus our expected achievements will focus on:

I. Detect SQLIA intrusion in a network
II. generate the SQL Injection Attack on one computer and detect the intrusion on the other computer
III. Protect the personal computer system using the host-based intrusion

B. Design Considerations

Having looked at all available technologies employed so far for mitigation of SQLIA and their results, we are of the opinion that a fuzzy rule-based classification system (FRBCS) can tackle the requirements of the current stage of SQLIA security treats. In a genetic fuzzy system for detection of SQLI not only the accuracy is a priority, but also the learning and the flexibility of the obtained rules. To create the rules having high generalization capabilities, our algorithm builds on initial rules, data-dependent parameters, and an enhancing function that modifies the rule evaluation measures. The enhancing function helps to assess the candidate.

This paper will make use of function that is very important program that is used to do critical actions over the database. By default the functions in the database are created as public to allow all the users to use them. As they are public the functions can also be used by the attacker. The functions can do critical operations such as changing administration rights to the users and changing user names and passwords for the databases. So it is recommended that the functions which are not used by the application are restricted with access limitations. An end-user can send some commands to the database by inserting malicious code in the URL strings. So by validating the code that is coming to the server from the end user the SQL injection attacks can be determined easily.

Most applications prepare SQL statements with the strings which are already constructed by the programmer. The SQL statement takes the user inputs and it forms a string with valid data variables from the user. Then the string is executed against the database by using the application programming techniques. As the application gets the normal data to it in the form of URL

strings the user data passed through it would be in a form of a string.

The ultimate aim for the attacker is to craft a SQL string which can be executed against the database successfully. The attacker tries to use the keywords like „OR", „AND" which normally evaluates the executed conditions are true all the times. The attacker also uses the character Single Quote („) which are used to specify a character string in SQL data type. By using the Single Quote character successfully the attacker.

C. IDS Design

Intrusion detection system (IDS) is a technique of detecting unauthorized access to a computer system or a computer network. An intrusion into a system is an attempt by an outsider to the system to illegally gain access to the system. Intrusion prevention, on the other hand, is the art of preventing an unauthorized access of a system's resources. The two processes are related in a sense that while intrusion detection passively detects system intrusions, intrusion prevention actively filters network traffic to prevent intrusion attempts.

An Intrusion is a deliberate unauthorized attempt, successful or not, to break into, access, manipulate, or misuse some valuable property and where the misuse may result into or render the property unreliable or unusable. The person who intrudes is an *intruder.* Six types of intrusion have been identified:

- Attempted break-ins, which are detected by atypical behavior profiles or violations of security constraints. An intrusion detection system for this type is called anomaly-based IDS.
- Masquerade attacks, which are detected by atypical behavior profiles or violations of security constraints. These intrusions are also detected using anomaly-based IDS.
- Penetrations of the security control system, which are detected by monitoring for specific patterns of activity.
- Leakage, which is detected by atypical use of system resources.
- Denial of service, which is detected by atypical use of system resources.
- Malicious use, which is detected by atypical behavior profiles, violations of security constraints, or use of special privileges.

For the purpose of our work we make the IDS classified into Four major groups: - Intruder Type, Detective Behavior, Detective Approach And Systems Type.

Client-Side (Browser): The starting point of SQLI attacks is the slide (browser). If attack inputs can be detected early at the browser side, then it could be thwarted early by not forwarding the malicious inputs to the server-side for further processing. Here the focus is that a client -side approach to detecting SQLI attacks be adopted. The client -side accepts shadow SQL queries from the server-side and checks any deviation between shadow queries with dynamic queries generated with user supplied inputs. We measure the deviation of shadow query and dynamic query based on conditional entropy metrics and propose four metrics in this direction. We evaluate the approach with three PHP applications containing SQLI vulnerabilities.

The paper will include an application protocol intrusion detection system which is designed to scan the input characters from a user for SQL keywords use maliciously against the Database. Such keywords will include SELECT, DELETE, INSERT, UPDATE, CREATE, DROP, ALTER, WHERE, LIKE, AND, OR, =, FROM., ; etc.

D. Control Centre

The Control Centre is responsible for total operations of SQLIA IDS system. A diagrammatic representation of Common Intrusion Detection Framework (CIDF) of the proposed control Centre for the SQLIA IDS in this paper is shown in Fig. 1.

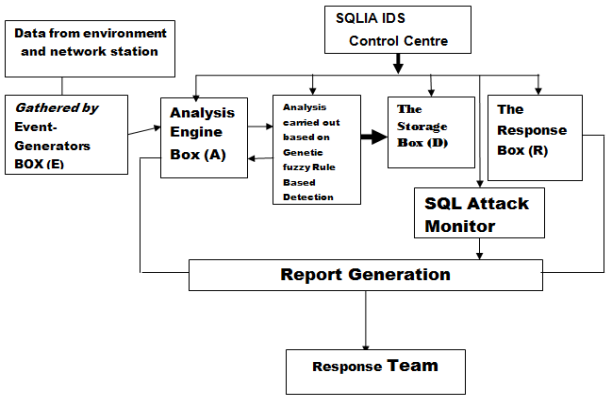

Fig.-1: IDPS Control Centre

- The Algorithm for control center: Network traffic Information from the environment gathered by Event generators are fed into analysis engine. Analysis engine analyses events based on the delectation method**.** The storage units stores the needed information needed by the IDS for the required Response by the IDS RESPONSE Team.
- How the controls centre Operates: To maintain interoperability between different intrusion detection systems, a frame-work called the Common Intrusion Detection Framework (CIDF) is proposed as shown in Fig. 1. The proposed CIDF, intrusion detection system is composed of four basic processes called event generators (E-box), analysis engines (A-box), storage mechanisms (D-box), and response units (R-box).

The event generator (E-Box) collects events and sends them to the analysis engine. The analysis engine analyzes events according to its detection method, (in this design, the Genetic Fuzzy Rule-Base system). The storage

mechanisms store the information needed by the intrusion detection system. The response unit is used to initiate some kind of manual or automatic attack responses.

The use of different intrusion detection methods provides the possibility to mitigate the harm resulting from attacks and intrusions. Three research questions deserve special attention; how to secure the IDS itself, how to improve the effectiveness of the IDS, and how to improve the efficiency of intrusion detection. The intrinsic security of the IDS is very important. This includes protecting all phases in the intrusion detection and response process, as well as the protection of distributed security policies. The effectiveness of intrusion detection refers to the ability to correctly classify audit events as being intrusive or not. This most often includes a binary decision, but the decision could also include a percentage of uncertainty. Efficiency, on the other hand, is specified by the ability to effectively process incoming events. A low efficiency may lead to an IDS being flooded with audit data, resulting in lost or late events. To measure the effectiveness and efficiency of the current IDS, it is very important to rely on relevant metrics.

Fig. 2 is architecture of the basic operations and components of the IDPS.

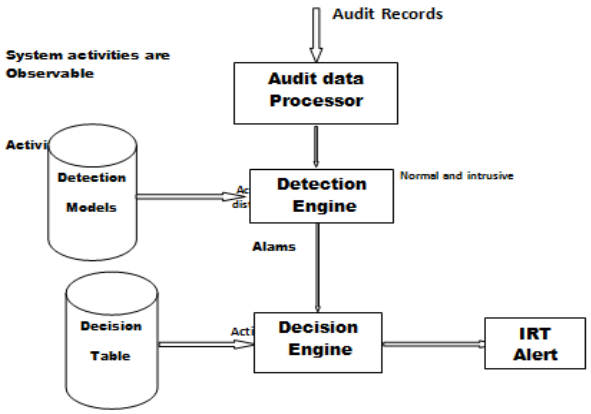

Fig 2-: Basic Operations Architecture

E. Algorithms

1) *A simple Genetic Algorithm*: Given a clearly defined problem to be solved and a bit-string representation for candidate solutions, the simple GA works as follows:

1. Start with a randomly generated population of N L-bit chromosomes (candidate solutions to a problem).
2. Calculate the fitness F(x) of each chromosome x in the population.
3. Repeat the following steps (a)-(c) until N offspring have been created:
   (a) Select a pair of parent chromosomes from the current population, with the probability of selection being an increasing function of fitness. Selection is done \with replacement," meaning that the same chromosome can be selected more than once to become a parent.
   (b) With probability pc (the crossover probability), cross over the pair at a randomly chosen point (chosen with uniform probability) to form two offspring. If no crossover takes place, form two offspring that are exact copies of their respective parents.
   (c) Mutate the two offspring at each locus with probability pm (the mutation probability), and place the resulting chromosomes in the new population.
4. Replace the current population with the new population.
5. Go to step 2.

Procedure (GA)
BEGIN
g ←0
Initialize P(g)
Evaluate P(g)
While (not matching the ending conditions)
Recombine P(g) to yield S(g)
Evaluate S(g)
Select P(g+1) from P(g) and S(g)
g ← g +1
END

In general, the methods that combine the genetic and fuzzy approaches for generation of knowledge bases (KBs) can be divided into two main groups: genetic tuning and genetic learning. If there exists a KB, we apply a genetic tuning process for improving the FRBS performance while preserving the existing RB. That is, to adjust FRBS parameters for improving its performance, maintaining the same RB. The second possibility is to learn KB components (an adaptive inference engine can be included). That is, to involve the learning of KB components among other FRBS components. Our system employs the genetic learning to learn the flexible inference engine.

F. Database Structure

Login file field structure: Table 3 is the authentication data. This table is used to register new users into the database so that they will be able to access the Web base application on the Net.

Table -3: Admin Data

Field	Data type	Data length
Id	Int	2
Username	Varchar	20
Password	Varchar	20

On the other hand, Table 4 is the new record register. It is used to register new record into the database so that it

will be accessible to any Web base application user of the system.

Table -4: New Record Details (Source Field Work 2017)

Field	Data type	Data length
Call id	Varchar	5
Caller phone Number	Varchar	15
Reciever phone Number	Varchar	15
Call cost	Varchar	5
Airtime Balance	Varchar	5
Call Time	Varchar	6
Call Duration	Varchar	3
Date	Varchar	8
Action	Varchar	20

1. Program/System Design: Below is a block diagram (Fig. 3) showing the arrangement of the main concept that makes up the system application. The modules involved in this system's design include: administrator' module and User's module

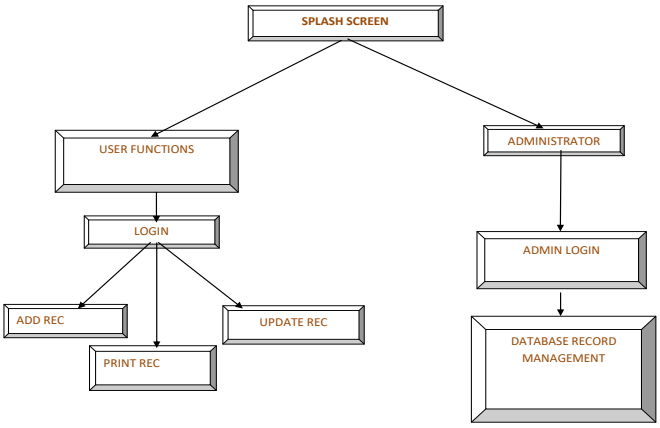

Fig 3-: Program Modules/Menu of Proposed System

2. *Program Module Specifications*: The program modules specify which program modules control an activity throughout the running of the program. Modules have been identified. Diagram of the approach is shown below in Fig. 4, along with the main functionalities of the module of SQL Injection Attack and how to prevent it.

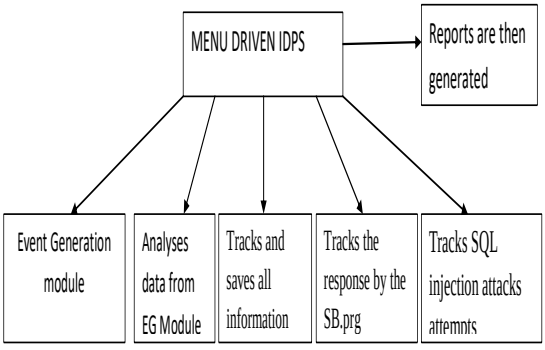

Fig 4-: Control Centre for an IDPS

In Fig. 4 above, the diagrammatic representation of the functions or subprograms required to determine a given SQLIA in a given web page is presented.

a. CONTROL CENTRE MENU DRIVEN IDPS. This Controls All the Subroutines Throughout The Program
b. EG.Prg Event Generation module that tracts intrusion activities and sends reports to related activity modules
c. AE.prg Analyses data from EG.prg
d. SB Tracks and saves all information produced by AE.prg
e. RB Tracks the response by the SB.prg and send a message to management teach for action.
f. SQL monitors Tracks SQL injection attacks attempts and reports each case to management team for action.
g. RG Reports are then generated for broader analysis and implementation.

G. Mathematical specifications

The model adopted for this dissertation is the Genetic Algorithm stated thus: The fitness function (delta) is given by:

$$F(\delta_i) = \frac{\alpha}{A} - \frac{\beta}{B} \quad (1)$$

The Certainty formula (Cn):

$$C_i(x) = \sum_{j=1}^{n} \Re_{i,j} \times \chi_j \quad (2)$$

GA is the type of algorithm that rather than starting from a single point (or guess) within the search space, GAS are initialized with a population of guesses. These are usually random and will be spread throughout the search space.

A typical algorithm then uses three operators, selection, crossover and mutation (chosen in part by analogy with the natural world) to direct the population (over a series of time steps or generations) towards convergence at the global optimum.

The three central operators behind the method are selection, crossover and mutation.
Selection attempts to apply pressure upon the population in a manner similar to that of natural selection found in biological systems.

Crossover allows solutions to exchange information in a way similar to that used by a natural organism undergoing sexual reproduction.

Mutation is *used* to randomly change (flip) the value of single bits within individual strings. *Mutation* is typically used very sparingly.
After selection, crossover and mutation have been applied to the initial population, a new population will have been formed and the generational counter is increased by one.

This process of selection, crossover and mutation is continued until a fixed number of generations have elapsed or some form of Convergence criterion has been met.

Genetic algorithms are general search algorithms based on metaphors with natural selection and natural genetics. The central differences between the approach and more traditional algorithms are:

The manipulation of a population of solutions in parallel, rather than the sequential adjustment of a single solution; the use of encoded representations of the solutions, rather than the solutions themselves; and the use of a series of stochastic (i.e. random based) operators.

The approach has been shown to be successful over a growing range of difficult problems.

H. Hardware and Software Requirement

1. Hardware Requirement

a. A computer system (SERVER) with the following specifications; 80 GB hard disk, 512 MB RAM, Pentium M (Intel) processor, 1.6 GHz
b. Printer
c. Uninterruptible Power Supply (UPS)
d. Stabilizer
e. Surge protector
f. Power generating set LCD monitors

Software requirement:

a. PHP
b. Internet Information Services (IIS)
c. MYSQL
d. ANTI VIRUS e.g MACfee, norton
e. Windows XP

2. Input/output formats

a. Input Design

Fig. 5 is the screen that displays and accepts users name and password that authenticates or verifies a user into the system. The figure represents the interface that first display and allow user to log into the system.

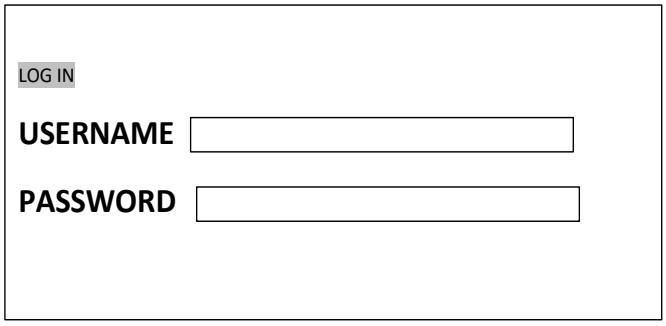

Fig 5-: Password Input Screen- Form 1

The screen shown in Fig. 6 accepts new details on the on the screen which is on sample database for testing the software

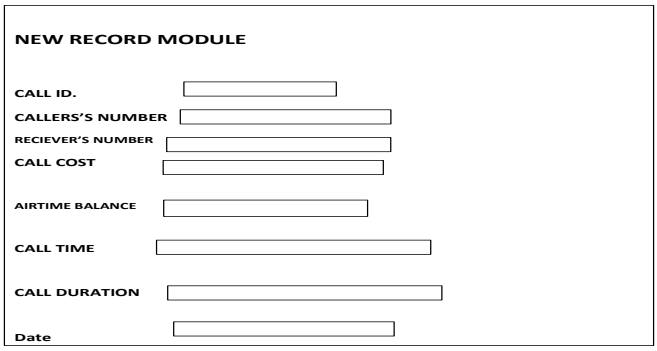

Fig 6-: Sample Input for Database Record-Form 2

b. Output design

Fig. 7 shows the hacker's report sheet. It is a listing of hackers/attempted hacking that took place over a given period. The result could be arranged according to date and time it occurred.

S/N	USERID	PASSWORD	SQLIA CODE	IP ADDRESS	DATE OF ATTACK	TIMEOF ATTACK
1	XXXXNN	XXXXXXX XX	XXXX	XXXX XXXXXX	MM/DD/YYYY	HHMM:PM/AM
2	XXXXNN	XXXXXXX XX	XXXX	XXXX XXXXXX	MM/DD/YYYY	HHMM:PM/AM
3	XXXXNN	XXXXXXX XX	XXXX	XXXX XXXXXX	MM/DD/YYYY	HHMM:PM/AM
4	XXXXNN	XXXXXXX XX	XXXX	XXXX XXXXXX	MM/DD/YYYY	HHMM:PM/AM

PRINT CLEAR

Fig 7-: Hacker's Report Sheet

The network database report use to list values or records in the network database is shown in Fig. 8. Only the system manager is allowed is to perform this function.

CALLER NO	RECIEVER NO	CALL COST	CALL BALANCE	CALL TIME	CALL DURAT.	CALL DATE
99999999999	99999999999	99999.99	99999.99	HH AM/AM	HH:MM	XX/XX/XX
99999999999	99999999999	99999.99	99999.99	HH PM/AM	HH:MM	XX/XX/XX
99999999999	99999999999	99999.99	99999.99	HH PM/AM	HH:MM	XX/XX/XX
99999999999	99999999999	99999.99	99999.99	HH PM/AM	HH:MM	XX/XX/XX

PRINT CLEAR

Fig 8-: Network Database Report

J. Overall data flow diagram of the new proposed system

The proposed system SQL injection detection system presented in this paper is shown in Fig. 9.

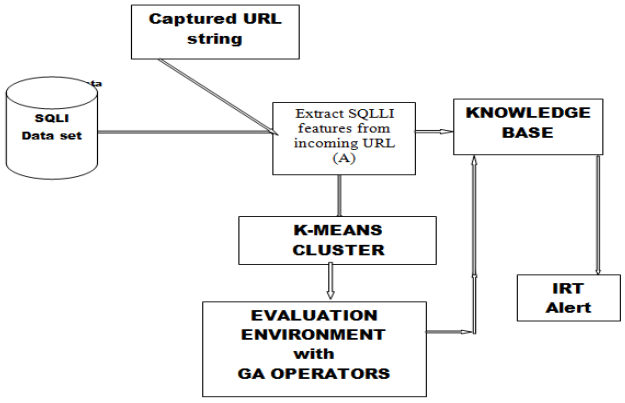

Fig 9-: Proposed SQL Injection Detection System

I. Extracting SQLI features from Dataset

The SQLI attack keywords are the popular keywords in SQL language which are generally used in order to perform operations on the tables inside a given SQL database. This approach extract tokens (keywords) which consist of specific terms of malicious web code as features. Tokenization is a process of breaking a sentence into a list of tokens. Based on these tokens, the system extracts the features, and represents one of the features mentioned below

- *F_1: Frequency of special character (dangerous characters) like = (--, #,/*,',"//,/**/,@@)*
- *F_2 : Frequency of special tokens (dangerous tokens) like (rename, drop, deflect, insert, create, exec, update, union, set, Alter, database, and, or information _schema, load_file, select, shutdown, cmdshell, hex, ascii).*
- *F_3: frequency of punctuations like (<,>,*,;,_,-,(,),=,{,},@,&,[,],+,?,%,!,:,/,\).*
- *F_4: frequency of SQL tokens like (where, table, like, select, update, and, or, set, like, in, having, values, into, alter, as, create, revoke, deny, convert, exec, concat, char, tuncat, ASCLL, any, asc, desc, check, group, by, order, by, delect, from, insert, into, drop, table, union, join).*
- *F_5: Length of SQL statement.*
- *F_6: frequency of space within the parameter of the query, which lead to the possibility of attacks.*
- *F_7: Existence of statements that always result in true value, for example "1=1" or "@=@" or "124=124".*

The appropriate selection of these features plays a crucial role in several aspects of the design of robust and feasible SQLI detection systems. the rational for choosing these types of features is its ability to identify most of SQIA types like tautologies, union, piggybacked, illegal/logical incorrect, alternate encoding and stored procedures which are treated the same as SQL queries. Other features can be included to increase the scalability of the system to detect new malicious code. in addition, by reducing the number of features (by eliminating redundant or irrelevant features), the performance of rule induction system can be improved as well as the classification of the rules produced.

II. K-Means Clustering:

To transfer the extracted numerical features (all mentioned above features except the 7th features into linguistic terms low (L), medium (M) and high (H): the system utilizes K-means clustering algorithm. K-means is one of the well-known clustering problems. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume K clusters) fixed Apriorl. Let X = $\{X_I, X_2, \ldots X_n\}$ be the set of data points that represents the Fi (i=1,....6) values across the queries and V = $\{v_1, v_2, \ldots v_n\}$ be the set of initial centers. Algorithmic steps [32,33] for K-means clustering are:

1. Randomly select k cluster centers, k=3 in our case.
2. Calculate the distance between each data point and cluster centers.
3. Assign the data point to the cluster center whose distance from the cluster id the minimum of all the cluster centers.
4. Recalculate the new cluster centers where ci represents the number of data points in i^{th} clusters centers. if no
5. Recalculate the distance between each data point and the new obtained cluster centers. if no point is reassigned then stop, otherwise repeat from step 3. this algorithm aims at minimizing an objective function, in this case, a squared error function ci is the number of data point in i^{th} cluster, and k is the number of cluster canters.

III. Fuzzy Logic System:

The FL system consists of (i) fuzzifier that takes input values and determines the degree to which they belong to each of the fuzzy set via membership functions (MFs): (ii) fuzzy inference that defines a non-linear mapping of the input data vector into a scalar output, using fuzzy rules and (iii) defuzzifier that maps output fuzzy sets into a Crips number.

A fuzzy set is defined as D = {[x,Ud(x))xcx,uD(x)c[0,1], where x represents the universal set, x is an element of x, D is a fuzzy subset in x and ta membership value (or degree of membership) between 0 and 1 next we will fuzzier the input (feature of SQLI) and the output (probability of injection), i.e. input and output are mapped into a set of fuzzy partitions. Here a seven point single output fuzzy system is used, which is given by $f_{:UCR}M_{---ZCR}M$.

IV. Program Implementation

A. Documentation

The word documentation according to the oxford advanced learners dictionary is a formulated detail record containing the procedures, processing the operations, problems and solution to problems, improvement and areas of necessary future attention of a new system.

Accurate documentation is essential so that implementation can be carried out and that the system can run on day to day basis. This is quite necessary as not only that it assist in the development process, its equally important in maintaining.

The software system maintenance, involves changes and correction by the programmer, sometimes other than the original developers. It therefore becomes imperative that the documentation that accompanies the completed tested software should be clear and complete
The documentation manual should be able to contain to contain the following among others

1. *The language and the size of the software*
2. *The number of core storage locations*
3. *System specification*
4. *Program specification*
5. *Source specification*
6. *Operating instructions*
7. *A list of errors and their meanings*
8. *A glossary of technical terms used*

This takes into consideration the orderly schedule of activities and material requirement of the new system. These are required for program implementation; Hardware and software requirement, Testing, Training. All these we maintained in this system.

B. Test and Evaluation

The testing of the new system is required to ensure workability and compatibility with the hardware components. Here, the program is subjected to trial with some life data and the behavior, manipulation and the output generated are compared with old system output.
Unit testing: a program module is tested here with life data to ensure compatibility and better result.
Integrated testing: this is the combination of all the modules of program, integration and full compilation to produce common goal. In the development of this package we have obeyed these procedures to come up with our conclusion presented in chapter five. We tested the software by generating several SQL Injection attack codes from SQL malicious characters and then adding those to the URL so obtained in the previous step. The URL with SQL code injected was send as a request to the web application. The response is written to a file and analyzed for SQLI pattern.
From the response we created a database that contains SQLI pattern and error response. The response from SQLI pattern logged into file will be further analyzed for further interpretation by the database administrator. It is on these interpretations that further security measures will depend on.

4. CONCLUSIONS

This study undertook a comprehensive survey of different SQL injection detection/prevention techniques and tools that have been proposed in the last decades. These techniques vary from enforcing authentication for signs-in web applications to tools that have been developed and proposed to analyze queries for any kind of injection. From the surveys of various article /research papers it has been found that the current techniques are not successful. Some have not been implemented yet and some techniques are impractical in reality because they could not address all types of attacks. In order to address the above issues this thesis employed a genetic-fuzzy rule-based classification system for the SQLI attack detection. Here the SQL statement is treated as a feature vector that characterizes the SQLI attack keywords.

The genetic algorithm is adapted for FRBCS to improve the quality of matching implemented by each rule by means of adjusting FRBS parameters to increase the generalization power of the classifier. We observed that the quality of the system output depended mainly on the selection of the attributes used to build the feature vector. It has the potential to give a higher accuracy when comparing to other solutions that use SQL keywords separately for the problem of SQL injection.

There are few cases of new patterns the system could not recognize, but if the system is retrained it can detect the new patterns without a significant increase in processing time. The solution can be used in combination with positive logic based filtering as in the prototype implementation. We have presented the evaluation methodology and reported the results that prove that:

(i) The proposed method performs other state-of-the art method.
(ii) Enhanced Cartesian of GA population type improves detection results.

REFERENCES

[1] OWASP. (1 November, 2015). O.W.A.S.P. Top 10 Vulnerabilities.Available:https://www.owasp.org/index.php/Top_10_2013-Top_10.

[2] B. Nagpal, N. Chauhan, and N. Singh. "A survey on the detection of SQL injection attacks and Their countermeasures." Journal of information processing System" ISSN 1976-193X{print}, 2013

[3] P. Sharma, R. Johari and S. Sarma "Integrated approach to prevent SQL injection attack and reflected across sit scripting attack," international journal of system Assurance Engineering and Management, vol. 3, no. 4, pp.343-351, 2012.

[4] R. Dharam and S.G. . Shiva "Runtime monitors for tautology based SQL injection attacks," in processing's of the international cyber security, cyber warfare and Digital forensic (cyberSec), Kuala Lumpur, 2012, pp. 253-258.

[5] I. Balasindaram and E. Ramaraj, "Authentication scheme for preventing SQL injection attack using hybrid encryption 9PSQLIA-HBE)," European journal of scientific Research vol. 53, no. 3, pp.359-368 , 2011.

[6] X. ZhangK, .J, Lin, S. J. Chen ., Y. Hwang, L. and F.H. Hsu, "TransSQL: a translation and validation-based solution for SQL-injection attacks," in proceedings of the 1st international conference on Robot, vision and Signal processing (RSVP), Kaohsiung, China, 2011,pp.248-251