Research Article

iMedPub Journals
www.imedpub.com

American Journal of Computer Science and Engineering Survey

2021
Vol.9 No.5:8168

An optimistic Security Model for Improving the Cyber Security using Adaptive Algorithm to prevent SQL Injection Attacks

P Salman Raju[*]

Department of Forensic Sciences, NIRDPR-Hyderabad, Hyderabad

[*]**Corresponding author:** P Salman Raju, Department of Forensic Sciences, NIRDPR-Hyderabad, Hyderabad, Tel: 8688811609; Email: rajupolavarapu.nird@gmail.com

Received date: April 14, 2020; **Accepted date:** September 03, 2021**; Published date:** September 13, 2021.

Citation: Salman PR (2021) An optimistic Security Model for Improving the Cyber Security using Adaptive Algorithm to prevent SQL Injection Attacks. Am J Comput Sci Eng Surv Vol: 9 No: 5.

Abstract

Internet usage has grown many folds in the last two decades. For many businesses, web applications have become one of the most significant and critical interfaces. Throughout today's economic and social life, the use of web-based services (such as e-commerce, online banking, and web-based communications, to name a few) has become a common habit. Countless applications operate worldwide on millions of servers, and their numbers are steadily increasing. It has become focus of attackers and hackers for the attacks because of the huge growth of internet usage. It is necessary for all companies to develop and protect their applications in order to maintain their credibility and keep their products relevant for users. Web applications have brought in new classes of computer security vulnerabilities, such as SQL injection (SQLIA) and it has exceeded previously prominent vulnerability classes in recent years, such as buffer overflows, in both new vulnerability reports and exploit reports. SQL injection is the instance of the broader class of vulnerabilities that are based on input validation. The primary purpose of this research is to study the vulnerabilities of SQL Injection and to propose an optimistic Security model for secure data transmission. In this work, I Proposed an adaptive algorithm to prevent SQL injections.

Keywords: SQL Injection Attacks; Cyber Security; Adaptive algorithm; Vulnerability; SQL-IF; Web Applications

Introduction

Background of the study

What Is Cybersecurity?: Cybersecurity is the practice of protecting systems, networks, and programs from digital attacks. These cyberattacks are usually aimed at accessing, changing, or destroying sensitive information; extorting money from users; or interrupting normal business processes. Implementing effective cybersecurity measures is particularly challenging today because there are more devices than people, and attackers are becoming more innovative.

Types of cybersecurity threats

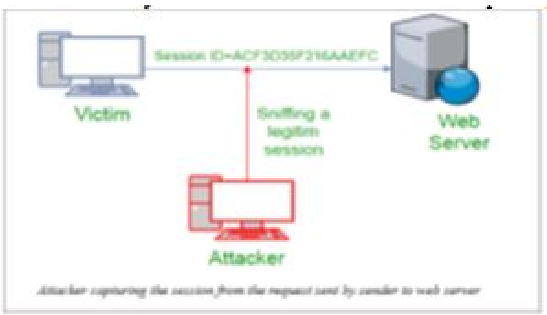
Phishing: It is the practice of sending fraudulent emails that resemble emails from reputable sources. The aim is to steal sensitive data like credit card numbers and login information. It's the most common type of cyber-attack. You can help protect yourself through education or a technology solution that filters malicious emails.

Ransomware: It is a type of malicious software. It is designed to extort money by blocking access to files or the computer system until the ransom is paid. Paying the ransom does not guarantee that the files will be recovered or the system restored.

Malware: It is a type of software designed to gain unauthorized access or to cause damage to a computer. Social engineering: It is a tactic that adversaries use to trick you into revealing sensitive information. They can solicit a monetary payment or gain access to your confidential data. Social engineering can be combined with any of the threats listed above to make you more likely to click on links, download malware, or trust a malicious source [1].

Session Hijacking: Web-based applications often use sessions to upgrade the client amicable experience for their clients. Usage of different sorts of the session the board does this. Session the executive chips away at the accompanying idea. At some early point in the client cooperation, the server creates a session identifier ID which is sent to the client's program and guarantees that a similar ID is sent back by the program alongside each consequent solicitation. Session IDs are recognizable proof tokens for the clients, and are utilized by the servers to keep up the session information (e.g., factors) [2].

Figure 1: Capturing user session id.


Session ID=ACF3D35F216AAEFC
Victim
Sniffing a legitim session
Web Server
Attacker
Attacker capturing the session from the request sent by sender to web server

SQL Injection

An SQL injection is a security weakness that occurs within database application layers. It is an act of passing SQL code to web-based interactive applications used in database services. SQL Injection is a web application database utilisation tool. This is achieved by inserting the SQL statements as an input string to access the database unauthorized [3].

An SQL injection is a severe weakness that results in a high level of compromise-usually the ability to execute a database query. This is a web-based code attack linking backends to the database and allowing to bypass the firewall. The downside of vulnerable code and inadequate validation of the data is that the attacker executes unauthorized SQL commands.

The interactive database-driven web page focuses on generating HTML content based on user-received feedback. For example, a web page on a news site may display articles related to a specific category, such as Sports, Politics, etc., depending on the value passed through the URL query string [4].

Figure 2: Attacker hacking through SQL Injections.

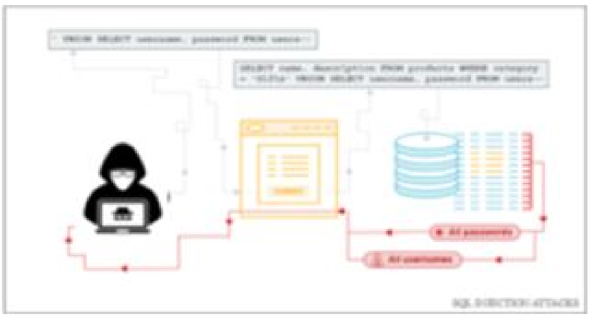
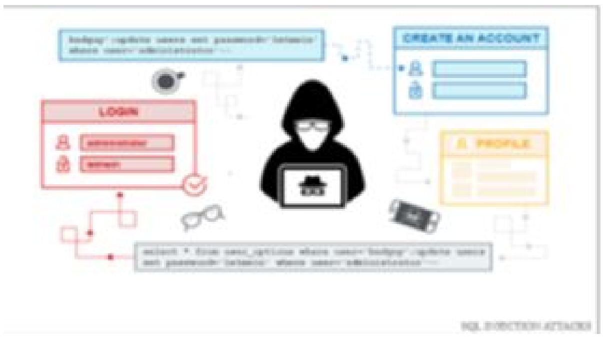


Figure 3: Attacker password hacking through SQL Injections.



Similarly, the search page would display the results based on the keywords entered by the user in the input box. Generally, the web page receives these inputs through the parameters of the string query URL and/or the form fields. Cookie values and other HTTP headers included in the request are also other input forms and may be used in the logic of the program as required. For example, the news site web page may display articles related to a specific category, such as Sports, Politics, etc., depending on the value passed through the URL query string.

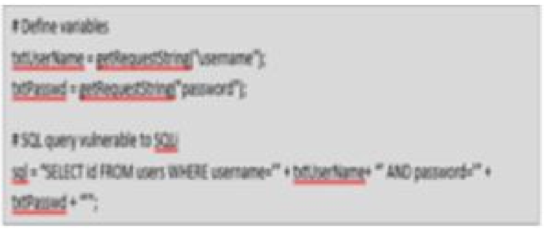
Using this injection technique, the attacker can gain unauthorized access to the restricted areas of the web application and can also retrieve, alter or damage the information in the backend database. In most cases, the attacker's intention is to steal sensitive information, such as credit card details, email addresses, passwords, and other private information, stored in the backend database.

How SQL Injection works?

The insecure website needs to include user input directly within a SQL statement to attack the web application. In order to run malicious SQL queries against a database server, an attacker must first locate an input inside the web application which is included inside a SQL query. An intruder can then attach a payload which will be included in the SQL query and run against the database server.

The following pseudo-code on the server side is used for authenticating users into the web application.

Figure 4: Pseudo-code on the server side is used for authenticating users.



```
#Define variables
txtUserName = getRequestString("username");
txtPasswd = getRequestString("password");

#SQL query vulnerable to SQLi
sql = "SELECT id FROM users WHERE username='" + txtUserName+ "' AND password='" +
txtPasswd + "'";
```

Problem statement

This section will discuss the main issues of the research and the solutions to the issues.

Web applications, for example, e-banking and electronic correspondence highlights, have turned out to be basic to the day-to-day lives of numerous individuals with the rise of the Internet and online utilization. Web applications have brought new kinds of vulnerabilities to PC protection, for example, SQL infusion (SQLIA), that as of late have surpassed already conspicuous helplessness classes, for example, support floods, in the two reports of new vulnerabilities and reports of adventures. SQL infusion is the two occasions of the more extensive class of info approval-based vulnerabilities. At their center, both include one framework accepting, changing, and developing string esteems, some of which originate from untrusted and obscure sources, and showing those qualities to another framework that deciphers them as projects or program sections.

These information approval-based vulnerabilities in this way require on a very basic level new systems to describe and alleviate them [5].

Aim and Objectives

This section will discuss the aims and objectives of the research.

The main objective of the proposed research work is to analyse the security challenges and vulnerability attacks on web applications. This research also prescribes and recommend defence methods for security attacks on web applications.

The proposed development has the following subsidiary objectives:

1. Designing characterization and runtime research of SQL injections on web applications and XPath injections. Also, this research studies about Static analysis for SQL injections.

This thesis additionally describes implementations of secure algorithms, showing them to be effective for their respective settings [6].

Significance of the Study

This section will discuss the importance of current research.

The study tends to enter approval-based vulnerabilities that emerge with regards to web applications, or all the more for the most part, with regards to meta-programming. This exposition gives the primary principled portrayal, in view of ideas from programming dialects and compilers, for such vulnerabilities, with formal definitions for SQL infusion specifically.

Expanding on this portrayal, the examination likewise contributes handy calculations for runtime assurance, static investigation, and testing-based examination of web applications to distinguish vulnerabilities in application code and keep assailants from abusing. This research moreover gives an account of the usage of these calculations, appearing to be successful for their separate settings. They have low runtime overhead, approve the definitions, scale to huge codebases, have low false-positive rates, handle true application code, and find already unreported vulnerabilities [7].

Literature Review

The growth of technology equally increases vulnerability. Subsequently, the more researches center around innovation, the more importance laid on the detection and prevention of related vulnerabilities. There is a rich amount of literature in this field.

History of Cyberspace: The word 'cyberspace' was first utilized by William Gibson in 1982 in a sci-fi 'Neuromancer', which he later portrayed as "a reminiscent and basically good for nothing popular expression that could fill in as a figure for the majority of his computerized insights. Presently it is utilized to depict anything related to PCs, data innovation, the web and the diverse culture of the internet. In this way, "cyberspace" is the electronic medium of PC systems, wherein online 41 correspondence happens and where people can cooperate, trade ideas, share data, give social help, lead business, direct activities, make creative media, participate in political exchanges and so forth [8].

The aspect of cyber-crime and security, its awareness: The Twentieth Century saw an ascent in cybercrimes reports, the rise of CERT techniques and merchant obligation idea for the security of products of information technology as a prosperous stage.

The current circumstance of IT security still keeps on marching on comparable tracks that concern the client networks (Kenneally. 93-95). With the developing size, multiplicity and pattern of netizens the security concerns expect an alternate measurement. Accordingly, cybercrimes like cracking passwords, accessing the account/system in an unauthorized manner, theft of email accounts, deceitful conduct within chat rooms and so on are practically equivalent to the offenses like pickpocketing in rural trains, selling movie tickets in black, eve-teasing, etc. This segment portrays some rising difficulties of cybercrimes emerging out of developing cyber civilizations.

A partial perspective on the data framework which can be actualized with the help of the UN was talked about in this paper. Various kinds of Cyber-crimes are DOS attacks, copyright violation, password trafficking, hacking and online pornography, and some other crimes carried out utilizing a computer framework or network, for example, crimes influencing on the online transactions identified with web-based business, burglaries of credit cards, online stalking, defaming online, digital fear-mongering and so on must be contemplated and act against them [9].

Review of existing studies: Web use and its application have come to a wide degree in the previous decade due to the simplicity in accessing and the availability of data. With this widespread utilization of web information, different lapses in security have additionally risen which remains to be a significant disadvantage.

For any web client, the data that are being utilized ought to be verified to stay away from any unapproved use. Significantly these web security issues make hackers extract the complete data of the users. This can cause a different impact on the credentials of the user. SQL injection has emerged as the recent vulnerability in the web world. In such a situation, it is important to give extra protection systems to secure the critical data that is recovered by SQL queries developed cautiously by programmers.

This section discusses the most recent advancements in the territory of web security components for the prevention of SQL injections [10].

SQL Injection Attacks: SQL injection attacks (SQLIAs) Structured query language (SQL) is a deciphered language that is used in web applications powered by a database that generates SQL articulations that fuse the client's knowledge or content. If this is performed in an unsafe manner, at that stage the web application may be defenceless against SQL Injection Attack, for example, in the event that the information provided by the client is not properly checked, then customers may modify or construct a harmful SQL statement and may execute self-assertive code on the server or change the content of the database. SQL language paves the way for different attacks.

Comparison of Existing Studies: Many research studies have been directed in addressing the issues identified with SQLI. In spite of the considerable number of efforts throughout the years to dispense with them, SQLI vulnerabilities are as yet predominant in web application source codes and attacks are still occurring exploiting owners of the webpage and honest users.

Table 1: Comparison of some of the studies discussed.

Author	Objective	Techniques developed	SQLI	Results obtained	Research gap
Ciampa et al. (pp. 43-49)	Identifying vulnerabilities of SQLI in web applications.	Heuristic-based approach and a tool-named V1p3R for penetration testing of the web applications.	SQLI	The developed tool reduced the cost by restricting the consumption of resources and interaction of the testers.	Improvement in tool heuristics, specifically concerning the correct interpretation of web pages that are a result of attacks.
Ghafarian (pp. 833-838)	Reducing the SQLIA vulnerability in web applications.	A combination of static and dynamic approach is developed.	SQLIA	In comparison with the existing developed technique is more effective since it had the ability to handle queries of any kind. Also, the algorithm is independent of the platform.	The proposed method is in its theoretical stage and practical implementation is yet to be carried out.

Research Methodology

Collection of database and SQL Injection

SQL Injection Attacks: Web applications are vulnerable to threats that makes the attackers to easily access the application interface and its data. An SQL injection is a type of attack in which the attackers insert Structured Query Language (SQL) code into the inputs of the web in order to access (or modify) data. This kind of vulnerability permits the attacker to commands directly to the web applications database to diminish the functionality (or) confidentiality. Several tools are available to detect and prevent these vulnerabilities.

Classification of SQL injection attacks: SQL injection is a model which exploits a web application's database. The information provided is description of attacks with SQL injection [12].

Order wise:

It composes of first-order injection attack and second-order injection attack. The common data types such as DATE and NUMBER which is considered as exploitable. In some cases, the database is manipulated by the attacker using creative coding.

II. Classical SQL Injection attack:

This attack combines with two SQL queries and then the attacking process is initiated. It does not allow to access the additional data in a certain table. Direct access of data from a database management system makes classic SQL injection easier.

III. Blind SQL Injection attack:

Although the application is vulnerable to SQL injection, the outcomes are hidden from the attacker. It is different from that of the original. It operates from logical statements being injected into the legitimate SQL statement of that page. This helps the threat agents to create the database by analysing the expressions. It is further classified as:

A) Standard Blind: In classical techniques, the SQL injection attack is a difficult task. It embeds operator and separator <".">
with application of <union>.

B) Double-Blind SQL injection: It returns error messages from queries that removed a web application from the

returned list. The response received doesn't affect the returned page.

IV. Attacks against Database:

a) SQL manipulation: This shall be classified as:

Manipulating the SQL queries is the common cause of SQL injection attacks. It is being categorized as a tautology, Inference, basic union queries and piggybacked queries.

b) Code Injection: It adds the additional SQL statements (or) commands to an existing SQL statement. This is easily applied to the Microsoft SQL applications.

c) Functional call injection: It is the addition of database functions into vulnerable SQL statements. These function calls can be used to make operating system calls or manipulate data in the database.

Proposing an adaptive algorithm to prevent SQL injections

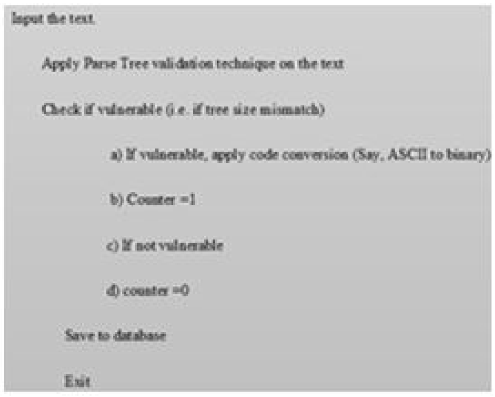
According to the Open Web Application Security Project (OWASP) top 10 security applications for the year 2017, the SQL injection attack is considered in the first place [13]. Several algorithms were designed to prevent SQL injections. Here, various algorithms were examined to prevent SQL injections. Ashish John et al suggested an adaptive algorithm to prevent SQL injection that combined the Parse Tree Validation technique and Code Conversion Method [14]. The task of the Parse Tree validation technique assists to parse the user input characters and filters the special characters from the input. In the second layer of security on top of the parsing technique, a novel implementation code is executed. It is emancipated that prior work on prevention of SQL injection, such as,

Parse Tree validation technique: Comparison at run time, the SQL statement's parse tree before user input is included with that corresponding after input is included.

Code conversion method: It converts the user's input into code such as ASCII, binary, hexa, etc., and search, the availability of converted data table input, and returns valid User-id and Password.

The prevention method comprises of best features of both parse tree validation technique and code conversion method. The user’s input is parsed and the level of a vulnerability is being checked. If the chance of vulnerability is higher, then the code conversion model is applied. By doing so, the detection and prevention of SQL injection are done. The followings are the codes of detecting and preventing codes of SQL injections.

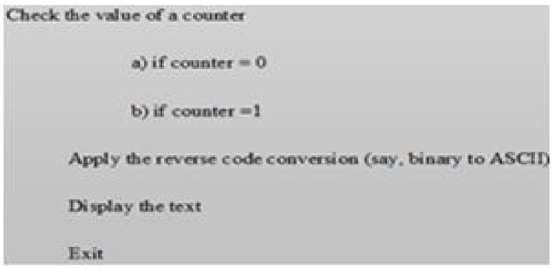
A. Code for web pages that saves data to the database:



```
Input the text.
    Apply Parse Tree validation technique on the text
    Check if vulnerable (i.e. if tree size mismatch)
            a) If vulnerable, apply code conversion (Say, ASCII to binary)
            b) Counter =1
            c) If not vulnerable
            d) counter =0
      Save to database
      Exit
```

The above code represents the input data saved to the database. At first, the input is taken from the user. Then, the Parse Tree validation technique is applied to the text. Based on the matching constraint of the tree size, the vulnerability of the threat is being validated. If the tree size mismatches, then the code conversion is applied. The counter value is 1 for the presence of vulnerable and counter value is 0 for the absence of vulnerable.

B. Codes for web pages that retrieve data from the database:



```
Check the value of a counter
          a) if counter = 0
          b) if counter =1
    Apply the reverse code conversion (say, binary to ASCII)
    Display the text
    Exit
```

The above code depicts the retrieval of data from the database. Depends on the counter value, the data will be retrieved. From the value of a counter, we can come to know whether the user input is converted or not.

The author, Ashish John et al, has given better results for detecting and preventing SQL injection attacks. Static analysis, dynamic analysis, a combination of static and dynamic analysis, web framework, defensive programming, and machine learning techniques are the most preferred techniques. Most of the static analysis makes use of the code vulnerability through defensive programming and machine learning techniques. Model checking, data flow analysis, abstract interpretation and use of assertions are the tasks of static code analysis.

Dynamic analysis operates intelligently by exploiting the vulnerabilities during the execution of web applications. For example, the CANDID tool operates automatically to examine the level of vulnerabilities. Both dynamic and static tool analysis has its own efficacy. However, the prior study stated that the dynamic analysis has effective performance in web applications.

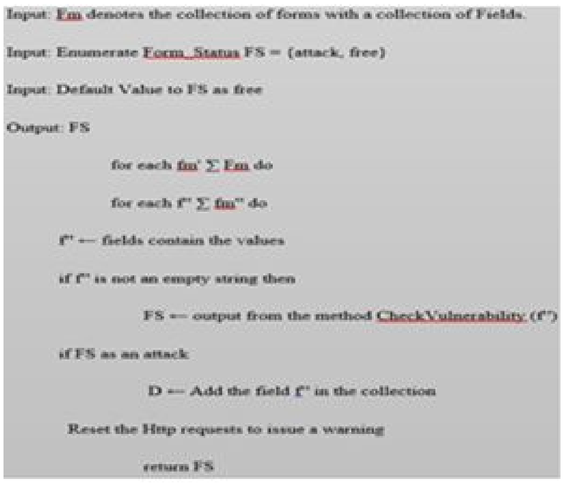
Most of the websites have utilized penetration testing tools that ensure the security information systems to the users by fixing its pitfalls. Owing to the merits of penetration testing in dynamic analysis, the authors, Kanc

- It does not require modification in the life cycle development.
- Avoids challenges in static analysis.
- No need for the source code
- Deployment is secured.

The best example is AMNESIA (Analysis and Monitoring to Neutralize SQL Injection Attacks), in which the technique tracks all dynamically generated queries at runtime and tests for compliance with the generated static model. When a query that violates the model is detected by the technique, it classifies the query as an attack, prevents it from accessing the database, and logs the information about the attack. Therefore, the authors proposed a protected SQL Injection Free (SQL- IF) algorithm that ignores the SQL injection attack feature.

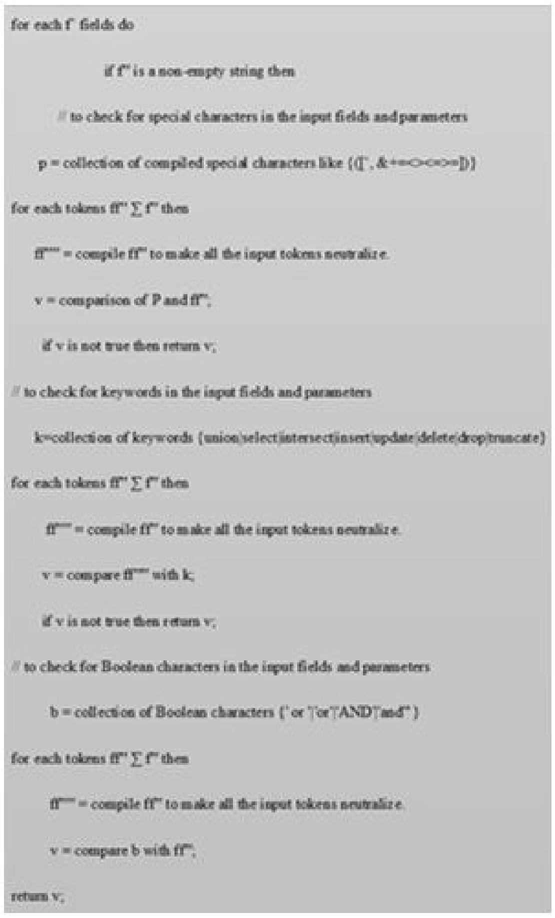
This algorithm is optimized to identify attacks by IF (Injection Free); while a different form of test suite is developed to detect attacks by SQL injection. The Algorithm below performs its task by assigning the Form Status (FS) as an attack and free with the set of fields obtained from form collection. The Form (fm) is obtained from the Form Set (Fm) whereas each field values are obtained from the Form (Fm). Inside the method called CheckVulnerability (f) three well-defined functions are generated to check for any special characters, keywords and Boolean characters. The following are the code for the proposed stable algorithm SQL-IF.

Proposed algorithm SQL-IF



```
Input: Fm denotes the collection of forms with a collection of Fields.
Input: Enumerate Form_Status FS = (attack, free)
Input: Default Value to FS as free
Output: FS
              for each fm' ∑ Fm do
              for each f'' ∑ fm'' do
       f'' ← fields contain the values
       if f'' is not an empty string then
                  FS ← output from the method CheckVulnerability (f'')
       if FS as an attack
                  D ← Add the field f'' in the collection
        Reset the Http requests to issue a warning
                  return FS
```

The above code represents the inputs taken from the users. Here, three inputs are taken, viz, collection of forms with a collection of fields, the status of the form and default value FS as free. Finally, the status of the form (FS) such as it is attack (or) non-attack. The field that contains values are checked, whether it is under the norms of forms constraint. Then, the vulnerability (f’) is checked as follows:



```
for each f' fields do
                  if f'' is a non-empty string then
          // to check for special characters in the input fields and parameters
     p = collection of compiled special characters like ({[`, &+=<>< >>=]})
for each tokens ff'' ∑ f'' then
     ff''' = compile ff'' to make all the input tokens neutralize.
     v = comparison of P and ff'';
      if v is not true then return v;
// to check for keywords in the input fields and parameters
     k=collection of keywords (union|select|intersect|insert|update|delete|drop|truncate)
for each tokens ff'' ∑ f'' then
       ff''' = compile ff'' to make all the input tokens neutralize.
       v = compare ff''' with k;
       if v is not true then return v;
// to check for Boolean characters in the input fields and parameters
        b = collection of Boolean characters ('or '|'or'|'AND'|'and'' )
for each tokens ff'' ∑ f'' then
        ff''' = compile ff'' to make all the input tokens neutralize.
        v = compare b with ff'';
return v;
```

The code above is the detection code used to check the web pages vulnerability. At first it checks the input fields for special characters, keywords and Boolean characters. It also tests keywords such as union, pick, overlap, insert, change, delete, remove, truncate and Boolean characters such as' or',' AND'' and the actual input parameters to neutralize all input values to the database. If there is any discrepancy in the parametric values, it is sent directly to the vulnerability data collector and resets the request to warnings.

The algorithm senses the SQLIAs that can be added to any specific web-based applications wherever the user interacts with the database. The detection algorithm is implemented via the Java platform which is feasible against SQLIAs to secure applications.

The above-proposed algorithm works efficiently against SQL injection attacks. In order to ensure the quality and performance of the web applications, SQL injection attacks have to detect (or) preventive measures should be provided at an earlier stage. Post testing process done by Ashish John, Ajay Agarwal, Manish Bhardwaj was more secure and reliable than the proposed SQL Injection Free (SQL-IF) algorithm.

The security concepts are given by Ashish John et al, during the credentials and restrict from the irrelevant query execution. Once the characters are filtered, the counter method help to flag the responses.

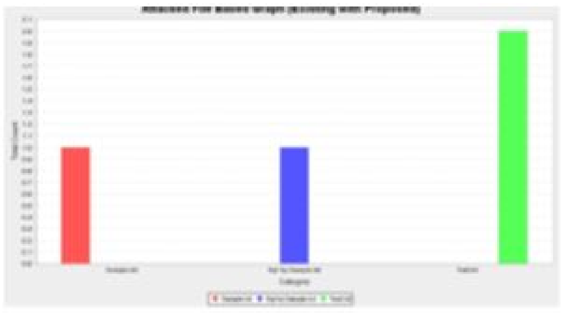
The code above is the detection code used to check the web pages vulnerability. At first it checks the input fields for special characters, keywords and Boolean characters. It also tests keywords such as union, pick, overlap, insert, change, delete, remove, truncate and Boolean characters such as' or',' AND'' and the actual input parameters to neutralize all input values to the database. If there is any discrepancy in the parametric values, it is sent directly to the vulnerability data collector and resets the request to warnings.

The algorithm senses the SQLIAs that can be added to any specific web-based applications wherever the user interacts with the database. The detection algorithm is implemented via the Java platform which is feasible against SQLIAs to secure applications.

The above-proposed algorithm works efficiently against SQL injection attacks. In order to ensure the quality and performance of the web applications, SQL injection attacks have to detect (or) preventive measures should be provided at an earlier stage. Post testing process done by Ashish John, Ajay Agarwal, Manish Bhardwaj was more secure and reliable than the proposed SQL Injection Free (SQL-IF) algorithm.

The security concepts are given by Ashish John et al, during the credentials and restrict from the irrelevant query execution. Once the characters are filtered, the counter method help to flag the responses.

Figure 5: Comparison graph.



The comparison graph showed that the test data showed higher number of counts in comparison to the sample data as well as SQL database.

Conclusion

This research appears to include approval-based vulnerabilities that exist with regard to meta-programming in the specific circumstances of web applications, or even more so for the most part. In light of ideas from programming dialects and compilers, this thesis gives the primary principled portrayal for such vulnerabilities, with formal definitions specifically for SQL injection. In addition, increasing on this portrayal, the exploration contributes a reasonable algorithm for runtime security, static investigation, and test-based investigation of web applications to identify vulnerabilities in application code and prevent misuse of attackers.

The vulnerabilities are normally present in web applications. It appears to be misused by SQL injection attacks to collect the gain from the application's involvement. Careful coding has been recommended as an answer to mitigating the SQLIAs.

Since SQL injections (SQLI) is the extremely risk-filled that happens regularly in the present web attacks, we tried to focus more on them and attempted to clarify with point by point models. Additionally, it is streamlined depending 130 on the neural relapse systems. What's more, we executed the SQLIA location program. Our pages of structure inspections that can gain powerlessness follow during the position procedure. In line with these, our approach will naturally produce attack vectors, which is very important for engineers and analysers to identify as well, to overcome the vulnerabilities in the code.

References

1. https://www.cisco.com/c/en_in/products/security/what-is-cybersecurity.html
2. Desmet, Lieven, et al. (2008) "Provable protection against web application vulnerabilities related to session data dependencies." IEEE transactions on software engineering 34: 50-64.
3. Abdul Bashah Mat Ali, Ala' Yaseen Ibrahim Shakhatrehb, Mohd Syazwan Abdullahc, Jasem Alostadd (2010) "SQL-injection vulnerability scanning tool for automatic creation of SQL-injection attacks", Journal of Procedia Computer Science, Elsevier Ltd 453-458.
4. Ghafarian, Ahmad (2017) "A hybrid method for detection and prevention of SQL injection attacks." Computing Conference. IEEE, 2017.
5. Fonseca, J (2007) CISUC - Polytechnic Inst. of Guardia, Guardia Vieira, M.; Madeira, H., "Testing and Comparing Web Vulnerability Scanning Tools for SQL Injection and XSS Attacks Dependable Computing, 2007.PRDC 2007", 13th Pacific Rim International Symposium on Date of Conference: 17-19.
6. P. Umasankari, E. Uma, & A. Kannan (2013), Dynamic Removal of Cross Site Scripting Vulnerabilities in Web Application. International Journal of Advanced Computational Engineering and Networking. 2320-2106.
7. Som, Subhranil, Sapna Sinha, and Ritu Kataria (2016) "Study on sql injection attacks: Mode detection and prevention." International Journal of Engineering Applied Sciences and Technology, Indexed in Google Scholar, ISI etc., Impact Factor 1: 23-29.
8. Farmer, F. Randall, Chip Morningstar, and Douglas Crockford (1994)"From Habitat to global cyberspace." Proceedings of COMPCON'94. IEEE.
9. Govil, Jivesh, and Jivika Govil. "Ramifications of cyber-crime and suggestive preventive measures." 2007 IEEE International Conference on Electro/Information Technology. IEEE,.
10. I. Hydara, A. B. M. Sultan, H. Zulzalil, N. Admodisastro (2014) Current state of research on cross-site scripting (xss) c a systematic literature review, Information & Software Technology 58 170–186.