DVGAR: Distraction of Web Vulnerabilities to Provide Grasp Accessing Capability of The Web Resources

[1]J.Nataraj, [2]P.Muralikrishnan
Department of Software Technology, VIT University, Vellore, Tamil Nadu, India
[1]natarajlogin@gmail.com, [2]muralikrishnan1190@gmail.com

Abstract — **An explosive growth of web applications like Online Banking, eStore, eCommerce, Military secrets and so on. Such fabulous resources are intruding by hacker in the form of cyberpunk like Injecting Query, Cross-Site Scripting and so on. In that, one of the most web vulnerable attacks is the SQL injection. It is the attack to compromise the database and broken the security wall to access the securable data. In the prior research technique, they have implemented the model of Parameterised Query transformation to prevent the SQL vulnerabilities. But it might not be sufficient technique to handle the Second Order SQL Injection vulnerability as well as it could not effectively discover the vulnerabilities spot based on the functional level. It might produce high level of false rate injection. In this proposed system we presented the concept of pattern based query evaluation technique. This technique will reduce the false rate of injection and ease to handle the vulnerable spot. In this simulation result shows that the security of false rate variation increase the detection of vulnerability spot and input time complexity are superior.**

Keywords: Web application form, Pattern Evaluation, Second order SQL Injection, intruders, Server Database

I. INTRODUCTION

Web Service is the integral part of the network. In the communication part, sharing of information is an essential to every human dependable life. On the Web resource have been crafted some unknown people by using many other techniques to form the malicious network like SQL Injection, Cross-site scripting, Denial-of-service, Remote File inclusion attack and so on.

SQL Injection is one of the web vulnerabilities in order to access the unauthorized user information unlimitedly. An intruder can compromise the data from the corresponding database of the web server. In web services consist of three tier approach of the basic communication services. One is for web clients, web server and back-end repository. Web server contain Database repository to retrieve the data whenever invoked it. User can access the login details or other transaction details from the server. That request can be converted into query structured form and transformed to the network. In the world wide network have the possibilities of problems occur like injection, traffic load, congestion, packet dropping and so on. From the cloud network, intruders try to inject some crafted code in the query and transformed to the web server. In server side, there is no proper input validation query processing system. Due to that, weak query processing in the server-side can response unauthorised information to the intruders.

Second order SQL injection is the attack in which user can submit the data and it can be stored in the database. When the data can retrieve from the database, it affects other queries functionality and also that query act as a part of malicious query statement. This kind of code injection is hard to find the vulnerability spot. It can be categories into 4 classes as follow,

1. *Frequency-based Primary Application*
   This class have the application can reprocessed the customer request using statistical frequency range of model. As below the example how the application functionality can be degraded by attackers.
   Example: “Most interesting book”, “Most interactive conversation at yesterday”.
2. *Frequency-based Secondary Application*
   This class include the application did not receive the injected code initial stage, but later submission of process from an application represent how the statistical view. As below example how the review of web-request can be crafted by intruders.
   Example: “Most frequent failed”
3. *Secondary Support Application*
   This class added application used to attack internal user of the primary application. Secondary application support and maintain the data of primary application.
   Example: In customer-care dept is present to provide some business related help and also maintain the company log book. Some of the unknown people try to get the company details while contact with them.
4. *Cascaded Submission Application*
   This class include application that makes multiple client submission within the single processing statement. Attackers within the class can access SQL code to manipulate the user data and targeted the back-end queries.
   Example: “to find nearest store to the temple”

Then second order SQL injections are stored in the manner as follow, Injected code can be stored as varies method will order the class of attack. There are three categories for code storage:

- Temporary Storage – the previous application search criteria
- Short-term Storage – the information stored in daily/weekly logs that can be reviewed irregularly.
- Long-term Storage – Data can be permanently stored in back-end systems that must be manually removed.

Example code of Second Order SQL injection
Username: john'—
Password: helloo5
But the applications predicate the single quote and stored in the database as

insert into users value(45,'john''--',’helloo5',00fffx);
After user table want to update the username
pquery = "update users set password = '" + newpwd + "' where username = '" + as ("username") + "'";

as("username") is the username retrieved from the login query. Given the username=john'-- in the query.

update users set password = 'helloo5' where username = 'john'--'

Above query may occur as serious defect to access the valuable information.

Many researchers could try to prevent the loss of authorised information from the code injection technique by using Online-Scanners, hashing techniques, SQLrand and so on. SQLIA is the top most serious web vulnerability in the network. In the existing approach, they deployed the parameterized query transformation techniques. While processing the query at the server side, it found the unknown query validation and it fixes the vulnerability spot. But it may not control the positive and negative false rate.

After analysed the problem, implement the pattern based query evaluation technique to solve proper input validation process and defence to stop malicious code injection in the server-side. Our experimental approach shows that resist malicious code from user request and provide authorised service.

The residual part of the paper is following manner. Section II discuss about the SQL injection preliminary activities, section II related work about the prevention techniques of SQLIA. In section IV describe the proposed system and descriptive algorithms. Section V explains the result analysis, section VI is the conclusion of the paper.

II.BACKGROUND

SQL injection is the malicious code injection in the data transformation in the network. Like intruder, hacker, code injector and so on are trying to compromise the database in order to fetch or drop the secured information. There are 3 kinds of SQL injection is

1. First order SQL injection,
2. Second order SQL injection,
3. Lateral injection.

Many process involved in these attack strategy.

A. Defence the stored SQL data

Normally, user can access the eBook information from the eBook store database. So the request is passed to the server in the form of SQL queries like

SELECT bk_name, bk_author, bk_qty FROM Book WHERE bk_name LIKE 'G%'
Now attacker crafted the above query as,
String query = "SELECT bk_name, bk_author, bk_qty "+ "FROM Book"+" WHERE bk_name LIKE '"+ his.search.Text+"%';
SqlDataAdapter access= new SqlDataAdapter(query, dbcommand);
Above the SQL injection occurred to compromise the database and tried to retrieve the data.

B. Prepared Statement

Prepared statements are resilient against SQL injection, because parameter values, which are transmitted later using a different protocol, need not be correctly escaped. If the original statement template is not derived from external input, SQL injection cannot occur. It can allow database engine before pre-compile the SQL statement. Such an efficient statement could be injected by attackers. Due to improper validation of query processing in the database, intruders are retrieving bulk amount of data. This can be happened whenever the query execution occurs.

preparedStatement = "SELECT * FROM customers WHERE name = '" + userName + "';";
In the SQL query processing, the termination symbol ";" used to end the running operation of the task.
SELECT * FROM users WHERE name = '' OR '1'='1' ";
Above code is the tautology query model. It satisfied the validation process and get user information from the database.

III. RELATED WORKS

From the past decade itself, researchers had tried to implement many defence techniques to stop the SQLIA (SQL Injection Attack) in the network. But day by day, varies kind of crafted technique come likewise researchers find out "how to prevent the query injection in the application". Here some methods,

A. *Structured Query Analysis*

A light-weight approach of Query transformation and hashing technique [1] is used to prevent the SQLIA from the server-end. It enhances defence code to fix the vulnerability spot. In this scheme to convert SQL queries into their respective structural form and then applying MD5 hashing technique to generate unique hash keys for each legal query during usage period. At run-time, hash key for every dynamic query is generated in the same manner and matched against the previously stored hash keys to prevent SQL injection attempts. This approach minimizes the size of the legitimate query repository and facilitates fast and efficient searching at run-time using a primary index. But this scheme didn't handle the second order SQL injection when it comes.

B. *Vulnerability Scanner*

Due to web vulnerabilities, to deploy some defence tool like Acunetix, Netsparker, and webCruiser and so on. From the way of constructed the Network Based Vulnerability scanner [2] (NVS), to work against the SQL injection in the web resource. In which the web scanner is able to detect all the pages in a web application which are vulnerable to SQLI, on behalf of the simulation attack, this tool makes a report which helps programmers to work and fix only the vulnerable pages, so this approach helps programmer to focus only the bad pages rather than the whole web application, at the same time NVS provides no false positive, provides up to maximal of coverage, and also the completeness.

C. *Mining Input Sanitization*

Data mining method is used to resist the code injection activities in the back-end server. In this method, could implement the proper validation scheme. This method claim the proof-of-concept tool called PHPMiner1 to extract the static code attribute. It has low precision of false rate as they don't have to analyze the correctness of input sanitization operations. The inputs and sinks that may cause security vulnerabilities. Detailed classification based on the types of sanitization methods that are commonly applied to the inputs to avoid security issues. For every sensitive sink in a web programming structure, to collect the static code attributes that characterize these classification schemes. Vulnerability prediction models are then built using the collected data and the vulnerability information of each sink. Even though this technique may not able to perform their defense continuation to conduct more comprehensive experiments on a larger set of systems to further validates these results.

D. *Piggybacking Query Evaluation*

It is the process in which provide reliable data service in the communicating network. Piggyback query transfer the packet from sender to receiver without loss of data. Similarly, it can prevent the problem like code injection, malicious code updating, congestion and traffic load and so on in the communication medium. It also provides the quality of service by getting acknowledgement from sender's side.

IV. PROPOSED APPROACH

In the web communication medium, variety of code injection problem occurred "How to solve the problem of every form of query injection in the web?" In day to day life, many researchers are trying to prevent the SQL injection attack in different way. In this paper, present pattern evaluation technique to solve the problem of SQL injection attack. Revising the above problem (II, III) to know how the Second Order SQL Injection attack compromise the database and how difficult to detect it.

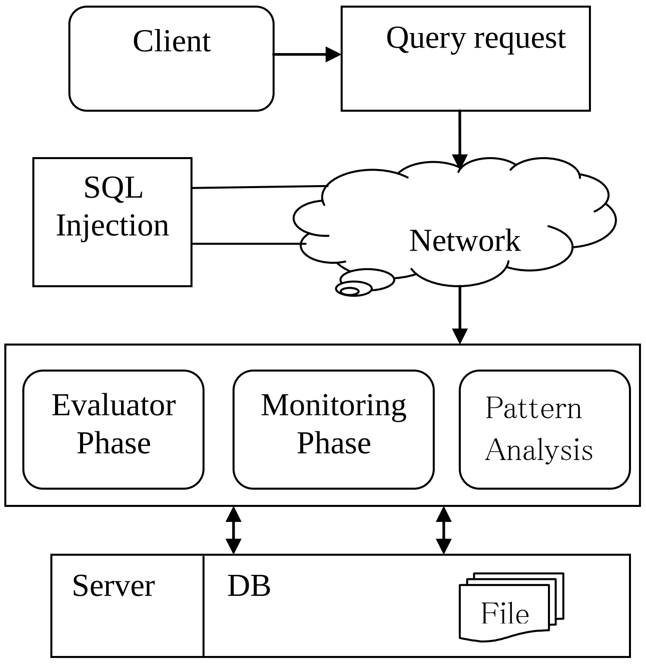


Fig 4.1. Architecture of Validation framework

From that issue, to improve the strong input validation in the server side. In this process, it consists of 3 phases namely, Query Evaluator phase, Pattern validations phase, Monitoring and transformation phase. These phases can extract and access the authorized data to the client-side.

1. *Query Evaluator phase* - Users sent the legitimate query is injection-free if there is no vulnerable spot found. In this phase, queries are analyzed and detect the vulnerable spot using vulnerable injection discarding algorithm as follows.

   Algorithm 1: Discarding the bad character

```
begin
    input:    pattern p, character i
    output: removal of bad character from the query.
    initialize
       kn_bad=array {“— “,”; “,” #”,” ’ ”,” ’\ “}
        type_checker=array{“select”, ”insert”,          ”update",      “delete”,
                      “drop”,      ”update”}
     validate
        initialize p=true;
           For i=ibound(kn_bad, type_checker) to fbound(kn_bad, type_checker)
       if(query(1,input,kn_bad(i),type_checker(i),    vbtextcompare)<>0)  then
                            p=false;
else      return p;
end
```

   From the above algorithm, violate the property of SQL injection attack because most of the injection occurred in the case of known bad characters.

2. *Monitoring and transformation phase* - In this phase, monitoring SQL queries based on the SQL structure even if any other bad character is found in the query statement. If the injected queries found, then resultant SQL queries are transformed into structured form like convert the vulnerable character into structured query pattern and then it allows accessing the database. Even in the persistent query stored in the database can detect and replace the query pattern whenever it starts to execute. The known bad character is pre-fetched from the injected queries and it may also reconstruct as predefined function if special character injection is found.

   Algorithm 2: Replace structured pattern

```
Begin
     Input: pattern p, character i
     Output: Conversion of structured query
Function escape(p)
     p=replace(i,” ‘ “,” ‘ ‘ “);
     escape=p;
 end function
end
```

   This induce the query structure can execute properly without affect their functionality.

3. *Pattern Validation phase* - The resultant query from the monitoring phase can be matched with Query Pattern and then allow to access the database whether the pattern can be validated. In this phase, work based on the below algorithm.

   Algorithm 3: Accept syntactic structure

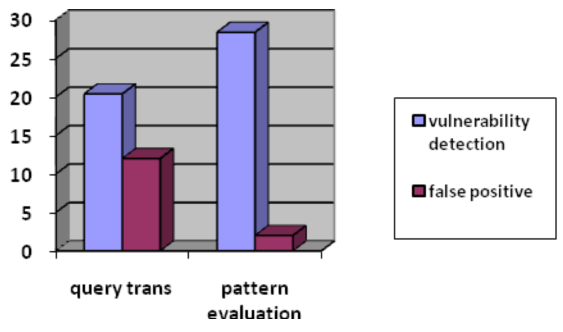
```
 Procedure pattern (Query, kn_bad[ ])
     Input: Query = user’s request
              kn_bad [] = ‘m’ updated pattern
    Output: Pattern evaluation
For i = 1 to m do
If (AC(Query, String.Length(Query),kn_bad[i]) = =0)
then
   VARIANCE rank=Compare (sql, kn_bad[i]) X100
                 ─────────────────────
                               String length (kn_bad[i])
If (VARIANCE rank >= Threshold rank) then
return Query is violate
```

```
Else
return Query is transformed to database
 End If
End For
End Procedure
```

Analyze the query structure and then compared with the known bad characters. When the query sink with its structural length string and special characters, to compare and find the rank of the matched queries. Then check if the resultant value is greater than the threshold rank, then it will violate the structural property of the queries and return to the developer record. Otherwise it transformed into the server and retrieving data from the database.

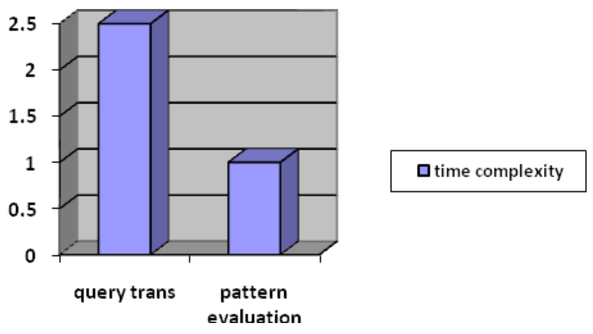
V. RESULT AND DISCUSSION

SQL injection attack can be carried out in several techniques by intruders in the network. The pattern evaluation technique is the effective process of strong input validation before access to the database. The existing method have also implemented for the prevention of SQL attack in the web application using parameterized form of hashing techniques. But it didn't able to handle the Second Order SQL injection attack that are executed and start its activities later in the database. By comparing these two approach, pattern based query evaluation technique is the most efficient to handle with effective in manner. The graph explains the quality of data transmission in the network. The query pattern can detect the vulnerability spot almost peak in the graph. The stored procedure of dynamic SQL technique has achieved to the 90% of SQL injection but the pattern evaluation method is achieved 95% of the web injection vulnerability.



It is more suitable to handle advanced SQL injection attack in the network. In this framework have strong input validation phases used for enhance the authorized information gathering from the server. The Structured query processing in the network is better performance when compared to parameterized evaluation system. The intruders are inject the malicious code into the back-end query and try to modify the authorized information from the server. Even the web application developer also didn't know the violation of security process. In the developer might use different kind of cryptographic techniques and algorithms were implemented to prevent such ridiculous activities occurred in the web application. This simulation result shows that the efficiency of secured validation should avoid the almost complete code injection in the query processing system.

Another aspect of the SQL injection is the time complexity. Many techniques were performing the task of providing the security to the web application. Based on the result, analyze the information to describe in the graph.



From the graphical view, Time taken by each method is increased in the time of prevention of the SQL injection attack. In the pattern evaluation technique is very less time when compared to other methods.

VI. CONCLUSION

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

REFERENCE

[1] Debabrata Kar and Suvasini Panigrahi, "Prevention of SQL Injection Attack Using Query Transformation and Hashing", *3rd IEEE International Advance Computing Conference (IACC)*, pp. 1317-1323,2013

[2] Dr.M.Amutha Prabakar, M.KarthiKeyan and Prof.K. Marimuthu, "An efficient technique for preventing sql injection attack using pattern matching algorithm", 2013 IEEE International Conference on Emerging Trends in Computing, Communication and Nanotechnology, pp.503-506,(ICECCN 2013)

[3] Avinash Kumar Singh and Sangita Roy, "A Network Based Vulnerability Scanner for Detecting SQLI Attacks in Web Applications 1st Int'l Conf. on Recent Advances in Information Technology | RAIT-2012 |

[4] Lwin Khin Shar and Hee Beng Kuan Tan, " Mining Input Sanitization Patterns for Predicting SQL Injection and Cross Site Scripting Vulnerabilities", ICSE 2012, Zurich, Switzerland New Ideas and Emerging Results, pp. 1293-1296,2012.

[5] Franc¸ois Gauthier, Ettore Merlo, "Fast Detection of Access Control Vulnerabilities in PHP Applications", In the proc. of 19th Working Conference on Reverse Engineering, pp.247-256, 2012.

[6] M. Alalfi, J. Cordy, and T. Dean, "Recovering role-based access control security models from dynamic web applications," in *ICWE '12*. Springer, 2012, pp. 121–136

[7] S. F. Yusufovna., "Integrating Intrusion Detection System and Data Mining", International Symposium on Ubiquitous Multimedia Computing, 2008

[8] William G.J. Halfond, Alessandro Orso, and Panagiotis Manolios, "WASP: Protecting Web Applications Using Positive Tainting and Syntax- Aware Evaluation", IEEE Transactions on Software, Engineering, Vol. 34, No. 1, pp 65-81, 2008.

[9] Kamra A, Terzi E., and Bertino, E.,"Detecting anomalous access patterns in relational databases", the VLDB Journal VoU7, No. 5, pp.1063-1077, 2009

[10] Bertino, E., Kamra, A, and Early, J., "Profiling Database Application to Detect SQL Injection Attacks", In the Proceedings of 2007 IEEE International Performance, Computing, and Communications Conference, 2007

[11] G. Lu and K. Lü, "Logical Trees: An Essential Method of Parsing SQL Statement with Semantic Analysis," 2011. Brunel University, UK.

[12] C. Anley. Advanced SQL Injection In SQL Server Applications, White paper, Next Generation Security Software Ltd., 2002.