Preventing SQL Injection attack by using Pattern Matching Algorithm

Monali Rajurkar: Department of Information Technology Engineering, SCOE, Pune
Savitribai Phule Pune University, Pune, India.
Nilesh Uke: Department of Information Technology Engineering, SCOE, Pune
Savitribai Phule Pune University, Pune, India.

***Abstract*—*Security of network frameworks is getting to be progressively essential as more delicate data are being put away and controlled online and more attacks are being propelled consistently. The security of a machine framework is traded off when an intrusion happens as it may cause information burglary or programmer making the machine frameworks more helpless. There are various algorithms which can be used for the seeking the results on web. Pattern matching system is one of them. Few models consider the detection of obscure attacks with decreased false positives and restricted overhead. This paper depicts a method to keep this sort of control and subsequently kill vulnerabilities of SQL injection. This paper proposed a detection and counteractive action method for counteracting SQL Injection Attack (SQLIA) utilizing Ahocorasick pattern matching calculation. The concentration of this paper is on positive tainting so detection makes it simple. The principle object is intrusion detection. Analyses demonstrate that proposed framework has higher detection rate than existing framework.*

Key words—SQL injection, database security, pattern matching, dynamic pattern, static pattern.

1.INTRODUCTION

Pattern matching is the methodology of checking an apparent grouping of string in presence of the involvement of several patterns [1]. As different from pattern recognition, the match normally must be correct. The examples by and large have the structure groupings of pattern matching in corporate yielding the areas of an example inside a sequence of string, to yield some segment of the pattern matched, and to put the pattern matching with some other sequence of string. Pattern matching idea is utilized as a part of numerous applications like web crawler, NLP and spam filters and so on [1].

The World Wide Web has been created with exceptionally quickly now a days. Web applications utilize the database at the backend for putting away information and SQL (Structural Query Language) for insertion and fetching of information. There are a few malicious codes that can be connecting to the SQL called SQL Injection. It is a sort of attack in a web application, in which the attacker gives SQL code to a client in formation box of a web structure to increase unapproved and unlimited access. The attacker's data is transmitted into a SQL query in such a path, to the point that it will structure a SQL code [2]. SQL injection is one of the numerous web attack components utilized by hackers to take information from associations. It may be a standout amongst the most widely recognized application layer attack methods utilized today. It is the sort of attack that exploits uncalled for coding of our web applications that permits hacker to inject SQL orders into say a login structure to permit them to get access to the information held inside the database. Basically, SQL injection emerges in fact that the fields accessible for client data permit SQL explanations to pass through and query the database straightforwardly. SQL injection is a hacking system which endeavors to pass commands through a web application to execute it with the help of backend database [1]. If not sterilized appropriately, web applications may bring about SQL injection attacks that permit hackers able to view data from the database and wipe it out.

Once an attacker understands that a framework is helpless against SQL injection, he finds himself able to inject SQL query/orders through an information structure field. This is equal to giving the attacker our database and permitting to execute several SQL commands to the database. An attacker may be executing discretionary statements on the helpless framework. This may trade off the integrity of database and/or uncover sensitive data. On the basis of the back-end database being used, SQL injection vulnerabilities lead to differing levels of information/framework access for the attacker.

Numerous programming frameworks have developed to incorporate a web-based framework that makes them accessible to people in general by means of the internet and can open them to a mixture of web-based attacks. One of these attacks is SQL injection, which can give attackers, not restricted access to the databases that underlie web applications and has get to be progressively frequent and genuine. This paper shows another exceedingly computerized methodology for securing web applications against SQL injection that has both applied and pragmatic points of interest over most existing procedures.

SQL Injection Attacks (SQLIA) represents a genuine issue to the security of web applications on the grounds that they can give unrestricted access attackers to databases that contain sensitive data. In this paper, we propose another, exceptionally mechanized methodology for ensuring existing web applications against SQL injection. Our methodology has both theoretical what's more viable favorable circumstances over most existing systems. From the theoretical point, the methodology is focused around the clever thought of positive tainting and the idea of syntax-aware assessment.

SQLIA has topmost priority in web based security problems. It is very difficult to detect the SQLIAs. Tautology attack, union queries, timing attack, piggybacked queries, blind SQL injection attacks are the most important types of SQLIA. The existing techniques cannot detect all types of attacks they have limitations like the development of the attack rule library, less detection capability and also difficulty of identifying all sources of user input. So there is a need to protect system from all these attack.

This paper is composed further as: Section II talks about related work studied till now. Section III presents implementation details, algorithms used, mathematical model and experimental setup tended to by this paper. Section IV depicts results and discussion part. Section V draws conclusions and presents future work.

1.1 Related work:

The utilization of web application is incrementing step by step. The web applications are being used for searching information on the web. String algorithm assumes an essential part for this. Diverse individuals are taking a shot at programming and equipment levels to make example quick searching. By applying different algorithms in different applications the surmised best algorithm for diverse applications is resolved [3]. The suggested algorithm give the decreased complexity furthermore diminished processing time.

The algorithm relegated to different applications may not be best optimal algorithm however better than the general algorithm. As opposed to applying every algorithm to each application one application is clarified with specific optimal algorithm [4]. To help an alternate kind of information, diverse algorithms are utilized. Every algorithm assumes an alternate part to searching information. Pattern matching is utilized as a part of distinctive application. Web search engine includes one of it. In searching it manages distinctive organization of information like Text, Image, Audio, Video documents.

Numerous existing systems, for example, separating, information flow, defensive coding can locate and keep a subset of the exaggerate that prompt SQLIAs. In this area, there is list the most important procedures like Ali et al's. Scheme - [5] embraces the hash value methodology to further enhance the client authentication technique. They utilize the client name and password values of hash SQLIPA (SQL Injection Protector for Authentication) model were created with a specific end goal to test the structure. The client name and password hash qualities are made and ascertained at runtime for the first time the specific client record is made. William G. j. halfond et. al's. Scheme [6] meets expectations by consolidating static investigation and runtime observing. In its static part, strategy employments program investigation to naturally fabricate a model of the real questions that could be created by the application. In its dynamic part, procedure screens the alertly produced queries at runtime and checks them for consistence with the statically produced model. Some queries which are violet the model presented to potential SQLIAs and are therefore kept from executing on the database and reported.

Thomas et. al. [7] proposed a automated arranged explanation era calculation to evacuate SQL injection vulnerabilities. They execute their research work utilizing four open source ventures. In light of the trial results, their arranged articulation code had the capacity effectively supplant 94 percent of the SQLIVs in some open source projects.

In high computerized methodology detection and avoidance of SQLIAs takes place. Naturally, our methodology meets expectations by recognizing "trusted" strings in an application and permitting just these trusted strings to be utilized to make several part of a SQL query, for example, keywords or administrators. The general component that we use to execute this methodology is focused around dynamic tainting, which stamps and tracks certain information in a project at runtime. The identification of web-based attacks has as of late got impressive attention as a result of the inexorably basic part that web-based administrations are playing. For instance, in [8] the researcher exhibit a framework that investigates web logs looking for some pattern of well-known attacks. An alternate kind of examination is performed in [9] where the location methodology is coordinated with the web server application itself.

2. Implementation Details:

2.1 System Architecture

In the Fig. 1, we discussed the proposed system.

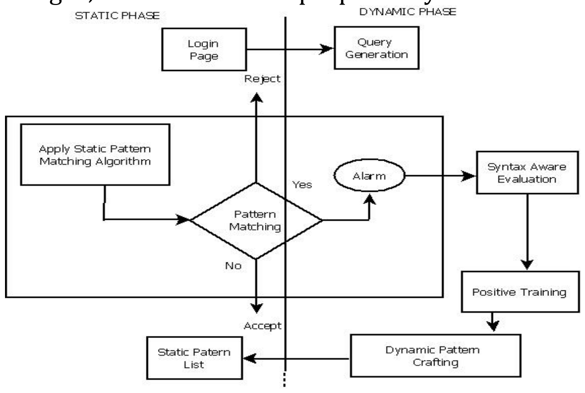

Fig1:System Architecture

We proposed a detection and preventive action system for avoiding SQL Injection Attack (SQLIA), in an attempt to overcome the drawbacks of existing system, but keeping the advantages, at the same time. In this paper, we proposed a construction modeling. This scheme also makes use of the pattern matching and the Aho-Corasick algorithm. And similar to existing system, it has two accompanying modules, Static Phase and Dynamic Phase. While the static phase still works the same way, as in the existing system. The changes are made in the Dynamic phase. This system introduces the Syntax-Aware Evaluation step and Positive Tainting step in the Dynamic Phase. The Proposed plan has the accompanying two modules as demonstrated.

1) Static Stage: In this, a static pattern list is maintained. And we keep up a list of known anomaly patterns. The client generated SQL queries are checked by applying the Static Pattern Matching Algorithm.
2) Dynamic Stage: In Dynamic Stage, for unknown attackquery if static stage failed then query is checked in the dynamic stage. First, query is converted into tokens then by using anomaly score untrusted tokens are removed by using trusted tokens the query is generated and executed.

In dynamic stage, following syntax-aware evaluation and positive tainting are two main important phases.

- Syntax-Aware Evaluation: In this step, the query is converted into tokens after that, it is being converted into regular expression for the evaluation of anomaly score. After syntax aware evaluation all the tokens are forwarded to the positive tainting for removing untrusted tokens.
- Positive Tainting: Positive tainting varies from traditional tainting, in spite of the fact that it is focused around the recognition, checking, and removal of untrusted tokens by the evaluation of anomaly score and the comparison with anomaly score threshold.

3. Algorithm:

3.1 Static Pattern Matching Algorithm

```
Input:
Step1: SPMA(Query, SPL[ ])
Query →User Generated Query
SPL[ ] →Static Pattern List with m Anomaly Pattern
Step2: For j = 1 to m do
Step3: If(AC(Query, String .Length(Query),SPL[j][0])==
)
```

$$AC_{SPM} = \frac{\text{No: of queries attack detected}}{\text{Total number of queries submitted}} * 100$$

```
Step4: Anomaly score = Matching value (Query, SPL[j][0]) / String Length(SPL[j])* 100
Step5: If (Anomaly_score ≥ Threshold_value) Then
Step6: Return Alarm → Administrator
else
Step 7: Return Query → Accepted
end if
Step 8: Return Query → Rejected
End if
End For
End Procedure
```

3.2 AhoCorasick Algorithm

```
Step 1: Procedure AC (y, n,q0 )
Step 2: Set of all Queries.
Step 3: For All Queries i = 1 to n do
Step 4: Check with Static pattern matching
Step 5: If (Detected (True)) show result;
Step 6: Else Send For Dynamic Pattern Matching
Step 7: Tokenize the query.
Step 8: Convert token into pattern matching syntax by using syntax aware
Step 9: For each token match with patterns
Step 10: Detect anomaly score for the query
Step 11: If (Anomaly Score < Threshold)
Step 12: Reject Query
Step 14: Else Start Positive Tainting
Step 15: Remove the attack pattern tokens
Step 16: After token removal combine all tokens
Step 17: Execute Query
Step 18: End for
Step 19: End Procedure
```

First the input for the algorithm is the SQL queries there may be possibility of presence of attacker pattern in the query. To remove the attacker pattern and to detect illegal query we are using two algorithms, static pattern matching algorithm and dynamic pattern matching algorithm. First the query is checked in static pattern Matching (SPM) algorithm. SPM is the known type of attacks that is previously detected patterns and the current query is matched with previous query pattern. If the query is not detected in SPM then the query is forwarded to Dynamic Pattern Matching (DPM).

In DPM query is divided into number of tokens and then tokens are converted into patterns i.e. syntax aware evaluation and then we are calculating the anomaly score for the whole query. If the score is lower than threshold then we reject that query and if the score is greater than threshold then we are doing positive tainting, which is a process of removal of attacker pattern i.e. untrusted pattern from the query and to execute the query with trusted pattern. The AC algorithm utilizes a refinement of a tries to store a set of anomaly keywords in a pattern matching technique.

3.3 Mathematical Model

Let S, be a system such that, S = { s, e, X, Y, T, fme, DD, NDD, ffriend, MEMshared,CPUCoreCnt, ∅}
Where,
S-Proposed System
s-Initial state at T (init) i.e. constructor of a class i.e. User
Query pattern () function.
e- End state of destructor of a class i.e. Stored Query pattern() function.
X- Input of System i.e. marked patterns.
Y- Output of System i.e. Trusted Patterns.
T- Set of serialized steps to be performed in pipelined machine cycle. - User Query pattern, Stored Query Pattern, MarkedPattern, Anomaly Detection, Trusted Query pattern.
fme- Main algorithm resulting into outcome Y, mainly focus on success defined for the solution - Pattern Matching Algorithm.
DD- Deterministic Data helps identifying the load store function or assignment function. - marking patterns i.e. identifying trusted or untrusted queries.
NDD - Non Deterministic Data of the system is time required to match users query with static pattern list and mark whether it is trusted or untrusted.
Ffriend- Set of random variables.
MEMshared- Memory required processing all these operations,memory will allocate to every running process.
CPUCoreCnt- More the number of count doubles the speed and performance.
∅Null value if any.

3.4 Expected Experimental setup

Now a day's detection as well as prevention of SQL injection attack is more important in web based security. It causes many privacy and legal issues.

For building system we made use of Java framework (version jdk 6)on Windows platform. Netbeans (version 6.9) is utilized for development. System can be worked on any normal hardware no need of any specific hardware.

For the modules setup and algorithm implementation Mysql Database is useful and to implement SPM and DPM algorithms java language and Netbeans IDE is useful and more flexible. Query Submit Module submitting the query to the Mysql database using Msql syntax for the query creation. Before the query is submitted and executed by Mysql it is evaluated by SPM and DPM. In static pattern matching algorithm the query is first converted into the regular expression by using pattern conversion classes in java. After the conversion it is matched with previous attack patterns stored in Mysql database.

4.RESULT AND DISCUSSION

4.1 DATASET

In this work several queries are applied. For generation result different query inputs to the system are applied and different set of outputs are generated for different inputs.

4.2 Results

Following table I show the expected results of detection accuracy from static and dynamic pa+tern matching algorithm. Dynamic pattern matching with positive tainting improves the detection accuracy of SQL injection attack. And the accuracy for the static and dynamic pattern matching is formulated as;

$$AC_{SPM} = \frac{\text{NO. of queries attack detected}}{\text{Total number of queries submitted}} * 100$$

AC_{SPM} is the accuracy for the static pattern matching algorithm which formulated as for how many queries the attack is accurately detected is divided by total queries submitted by user. And for dynamic pattern matching if the query is not detected by SPM then it is forwarded to the DPM for attack detection and hence it is formulated as follows,

$$AC_{DPM} = AC_{SPM} + \frac{\text{No. of queries attack detected}}{\text{Total no. of queries submitted}} * 100$$

As per the results DPM with positive tainting gives more accurate results in addition to SPM.

TABLE I. DETECTION ACCURACY

No. of Query	Static pattern Matching	Dynamic Pattern Matching(with Positive Tainting)
1	100	100
2	50	50
3	67	100
4	50	75
5	60	80
6	67	100
7	71	86
8	63	88
9	78	89
10	70	100

In the above table I analysis of existing system and proposed system is shown. For every different input the detection accuracy is evaluated. In proposed system results gets improved than existing system in terms of accuracy rate. Following figure 2 shows graph for detection in accuracy according to table I.

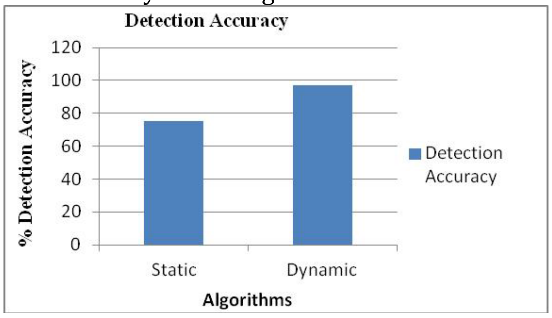

Fig.2 Detection Accuracy Graph

5 CONCLUSION

This framework prevents attacks like SQL control and also visual SQL injection. We likewise propose Positive tainting changes from conventional tainting, in spite of the way that it is concentrated around the detection, checking, and emulating of trusted, rather than non-trusted, information. Furthermore Syntax-Aware Evaluation is use the taint markings to perceive genuine from malicious queries. An approach that basically precludes the use from claiming untrusted data in SQL commandsis unfeasible plan. The peculiarity query utilized as a part of a few attacks, this sort of attacks can be prevent by our framework. In this paper, we anticipated a plan for preventive and detection action of SQL injection attack utilizing ahocorasick of pattern matching algorithmic.

ACKNOWLEDGMENT

The authors would like to thank the researchers as well as publishers for making their resources available and teachers for their guidance. We are thankful to the authorities of Savitribai Phule University of Pune and concern members of IRJET conference, organized by, for their constant guidelines and support. We are also thankful to the reviewer for their valuable suggestions. Finally, we would like to extend a heartfelt gratitude to friends and family members.

REFERENCES

[1] M. A. Prabakar, M. KarthiKeyan, K. Marimuthu, An Efficient Technique for Preventing SQL Injection Attack Using Pattern Matching Algorithm, IEEE Int. Conf. on Emerging Trends in Computing, Communication and Nanotechnology, 2013.

[2] William G.J.Halfond ,JeremyViegas, Alessandro Orso A Classification of SQL injection Attacks And Countermeasures.

[3] Georgy Gimel'farb, String matching Algorithms, COMPSCI 369 Computational Science.

[4] Nimisha Singla, Deepak Garg , String Matching Algorithms and their Applicability in various Applications, International Journal of Soft Computing and Engineering (IJSCE), Volume-I, Issue-6, January 2012.

[5] Shaukat Ali, Azhar Rauf, Huma Javed SQLIPA:An authentication mechanism Against SQL Injection.

[6] William G.J.Halfond and Alessandro Orso AMNESIA:Analysis and Monitoring for Neutralizing SQLInjection Attacks.

[7] S. Thomas, L. Williams, and T. Xie, On automated prepared statement generation to remove SQL injection vulnerabilities, Information and Software Technology 51, 589598 (2009).

[8] M. Almgren, H. Debar, and M. Dacier, A lightweight tool for detecting web server attacks, In Proceedings of the ISOC Symposium on Network and Distributed Systems Security, San Diego, CA, February 2000.

[9] M. Almgren and U. Lindqvist, Application-Integrated Data Collection for Security Monitoring, In Proceedings of Recent Advances in Intrusion Detection (RAID), LNCS, pages 2236, Davis,CA, October 2001. Springer.

[10] G. Vigna,W. Robertson, V. Kher, and R.A. Kemmerer. A Stateful Intrusion Detection System for World-Wide Web Servers, In Proceedings of the Annual Computer Security Applications Conference (ACSAC 2003), pages 3443, Las Vegas, NV, December 2003..

http://www.watchfire.com/