International Journal of Advance Research in Science and Engineering
Vol. No.5, Special Issue No. (01), February 2016
www.ijarse.com
IJARSE
ISSN 2319 - 8354

SQL INJECTIONDETECTION USING BAYE'S CLASSIFICATION

Amit Banchhor[1], Tushar Vaidya[2]

[1,2]PG.Scholar,Dept.of Computer Science &Engg. CSIT, Durg, (India)

ABSTRACT

SQL injection is a threat to those web applications which accepts input from the user and run queries in the backend as per the user's input and returns the desired output. Since the user's input plays an important role in the structure of the query that runs in the backend hence there lies a threat that if these inputs are modified and tailored in some specific way such that it changes the desired output then such modified query will be termed as injected query and the phenomenon is called SQL injection. There are many ways in which such manipulations can be done. Since there are many ways in which the injection can be done hence it becomes very tedious to detect all types of injection by using any particular tool, hence the tools are tailored for specific type of injection. This study is focused on detecting the SQL injection using improved Baye's theorem. The research involves detecting the injection on the basis of keywords rather than one statement of the SQL query. When the Baye's theorem is applied on the keyword then the performance and accuracy is greatly improved. In future both the methods that is, Baye's theorem based on statement of the SQL query and the SQL keyword can be combined to obtain better results.

Keywords: backend, Baye's theorem, query, SQL, Web application.

I. INTRODUCTION

In the development of web applications that takes input from the user it becomes a developer's prime responsibility to make the application's security strong enough to detect and avoid illegal activity in the application. SQL injection is one of such security threat which must be avoided [1]. SQL injection would open the gates to the backend database which may contain vital and personal information, and no one wants their personal information to be in the hands of strangers. SQL injection works on the principle of forcing a modified input into the application entry field which may alter the stored procedures and hence giving different results than the desired one. There are various ways in which these injections can occur and hence there are several tools that detect specific types of injection. AIIDA (An Adaptive Intelligent Intrusion Detector Agent)is one of such tools used for detecting SQL Injection attacks.[2]. AIIDA is an intelligent system requiring large resource and man power. There are tools that detect the injection based on matching the regular expression [3]. Some run time monitorsalso exist which are specifically tailored for detecting the tautological injections [4]. One of the methods uses machine learningfor detecting the injection and uses Baye's theorem to identify the detection [5]. Baye's theorem uses various probabilities to predict the future occurrence of the similar injection. Baye's theorem has been used previously for distinguishing and classifying web pages based on their visual contents [6]. The query on which we need to apply Baye's theorem must first be broken into tokens[7]. After which the

Baye's theorem is applied on the tokens to provide better results, rather than applying the theorem on the whole query.

II. BAYE'S CLASSIFICATION

Baye's classification is a family of probabilistic classification based on applying Baye's theorem.
Despite their naive design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. In 2004, an analysis of the Bayesian classification problem showed that there are sound theoretical reasons for the apparently implausible efficacy of naive Bayes classifiers [8] Still, a comprehensive comparison with other classification algorithms in 2006 showed that Bayes classification is outperformed by other approaches, such as boosted trees or random forests.[9]
An advantage of naive Bayes is that it only requires a small amount of training data to estimate the parameters (means and variances of the variables) necessary for classification. Because independent variables are assumed, only the variances of the variables for each class need to be determined and not the entire covariance matrix.
Abstractly, the probability model for a classifier is a conditional model [10]

$$p(C \mid F\_1,\ldots..,Fn) \quad (1)$$

over a dependent class variable C with a small number of outcomes or classes, conditional on several feature variables F1 through Fn. The problem is that if the number of features n is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable.
Using Bayes' theorem, this can be written as:

$$p(C \mid F\_1,\ldots..,Fn) = p(C)\ p(F1,\ldots..,Fn|C) / p(F1,\ldots..,Fn) \quad (2)$$

The above equation can be written as:

$$\text{Posterior} = \text{Prior} * \text{Likekihood} / \text{evidence} \quad (3)$$

In practice, there is interest only in the numerator of that fraction, because the denominator does not depend on C and the values of the features Fi are given, so that the denominator is effectively constant.

III. METHODOLOGY

From the review of the literature it has been found that none of the methods can absolutely detect all kind of SQL injections, each method is particularly favourable to specific type of injection. After literature review conceptual framework was performed to develop a method which can offer better results and detecting more kind of injections simultaneously.
The proposed methodology either returns "YES" for non-malicious query or "NO" for malicious query.
The algorithm can be described as follows:
1) Calculate the probabilities for each attribute for being malicious or not.
2) Find number of malicious query in training test=Nm
3) Find number of non-malicious query in training test=Nnm
4) Find total number of queries to be tested=Tq
For each query do the following.

a) Lqm=probability product of all attributes for being malicious

b) Lqnm=probability Product of all attributes for being non malicious

c) Pprior(M)=Nm/Tq (4)

d) Pprior(NM)=Nnm/Tq (5)

5) Ppost(M)=Pprior(M)*Lqm (6)

6) Ppost(NM)=Pprior(NM)*Lqnm (7)

7) Depending on the Ppost the query is declared as malicious or non malicious

Following is the flowchart for the proposed method:

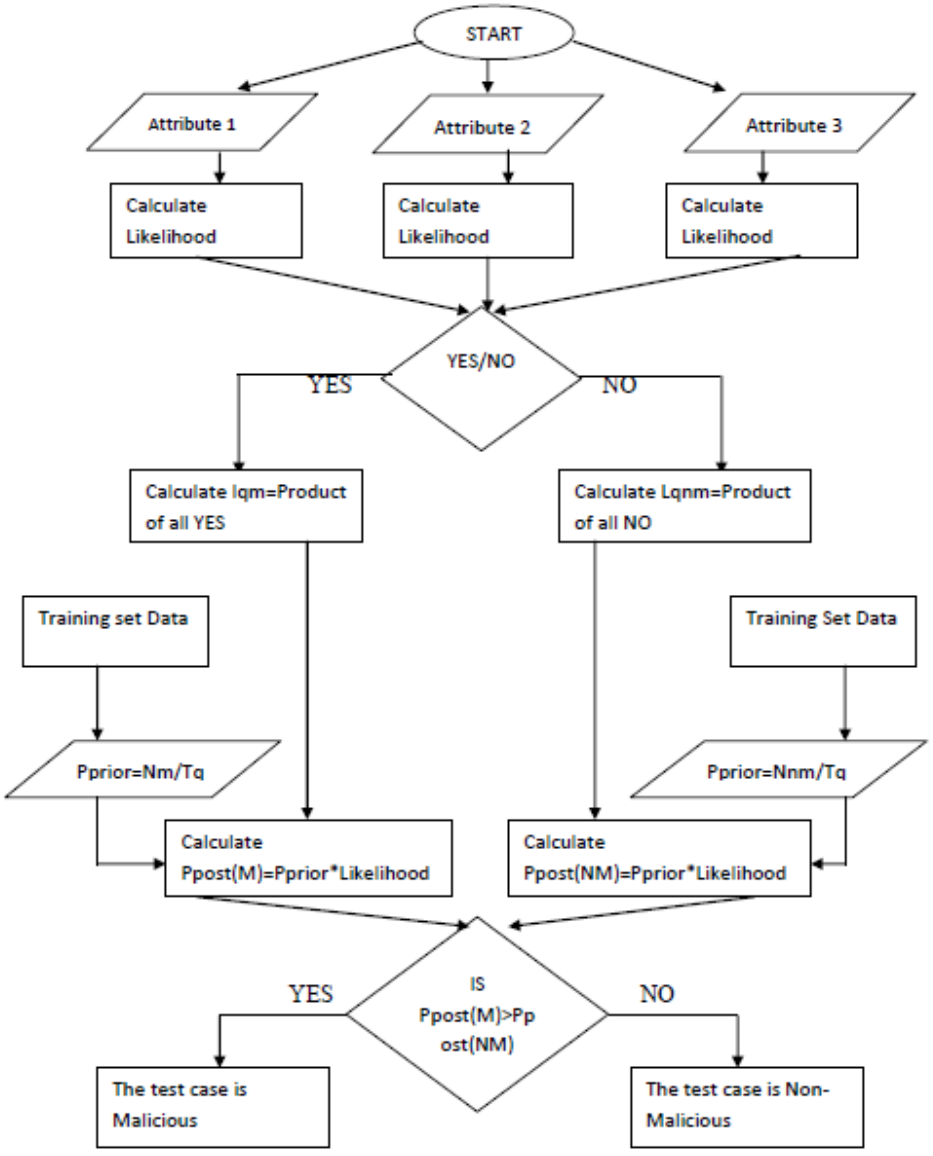

Fig 3.1: flowchart for deciding a query to be malicious or non-malicious

IV. CONCLUSION

Baye's classification is applied to detect the SQL injection, the probability distribution is calculated on the queries, but when the probability distribution is applied over the keywords of SQL instead on the whole query

then we achieve tighter boundaries. Here we can see improvement over the decision of the new queries to be malicious or not.

The graph below plots the various probabilities for classifying the injection to be malicious or not, when the same Bayesian classification is applied on the whole query and when it is applied on the individual keyword we obtain different results, which proves that the classification when done on the keywords as tokens is better than when we apply it on whole query.

In the figure below the "probability" is the probability calculated on the whole query and "classified probability" is the probability calculated on the keywords from the query.

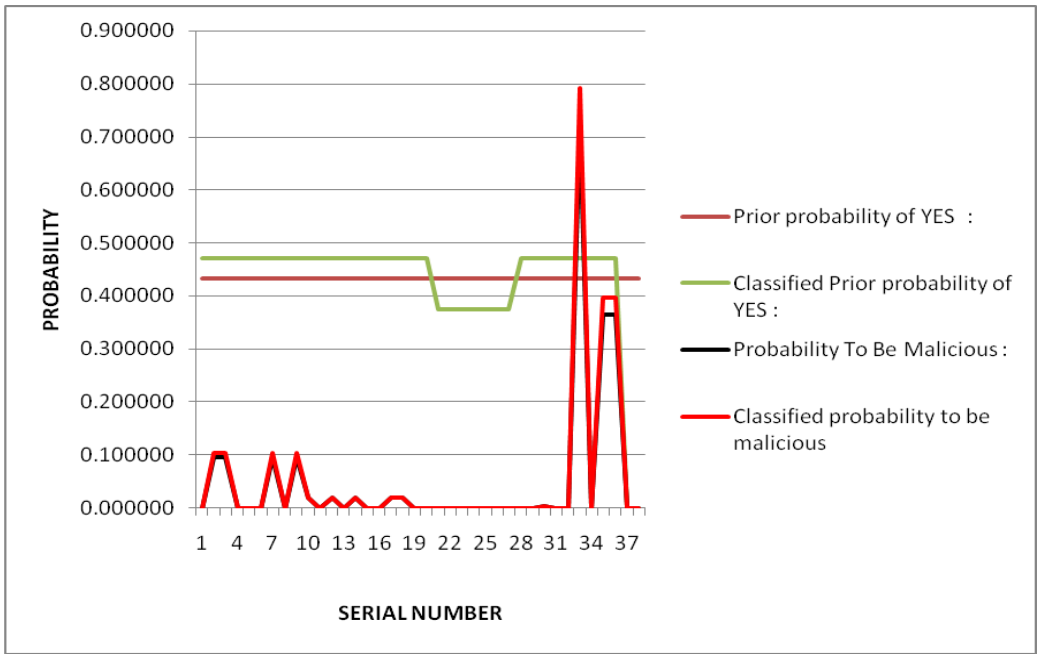

Fig 4.1: probability distribution for query being Malicious

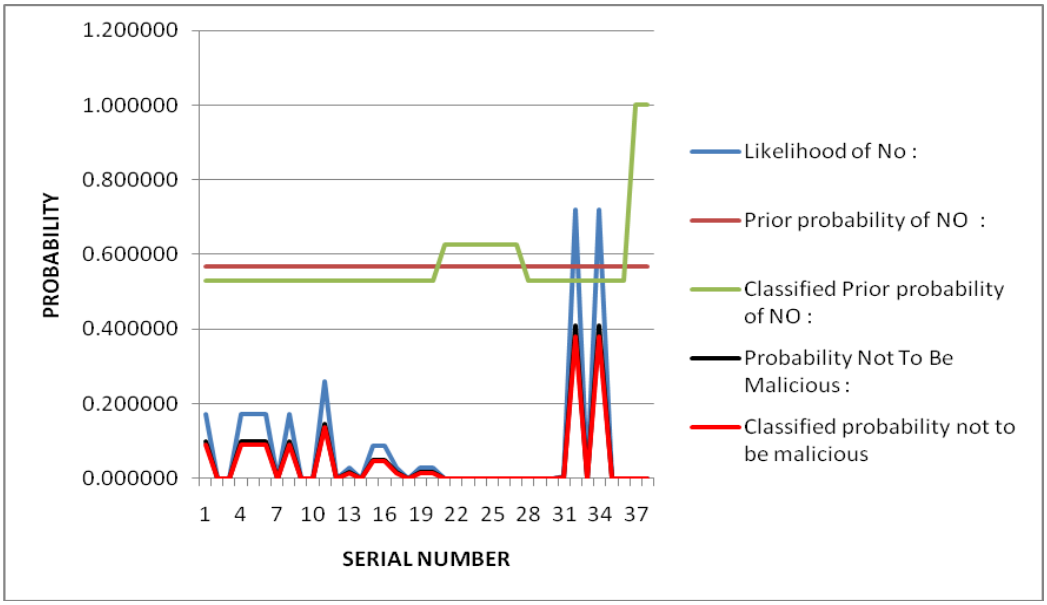

Fig 4.2: probability distribution for query being Non-Malicious

REFERENCES

[1] W. G. J. Halfond, et al., *A Classification of SQL-Injection Attacks and Countermeasures*, in Proceedings of the IEEE International Symposium on Secure Software Engineering, Arlington, VA, USA, 2006.

[2] Pinzon, C (2010). AIIDA-SQL: *An Adaptive Intelligent Intrusion Detector Agent for detecting SQL Injection attacks*. Hybrid Intelligent Systems (HIS), 2010 10th International Conference, IEEE, *ISBN: 978-1-4244-7363-2*, August 2010.

[3] Wan, Min (2012). *An Improved Eliminating SQL Injection Attacks Based Regular Expressions Matching*. Control Engineering and Communication Technology, 2012 International Conference, IEEE, *ISBN: 978-1-4673-4499-9*, Dec. 2012.

[4] Dharam, R (2012). *Runtime monitors for tautology based SQL injection attacks*. Cyber Security, Cyber Warfare and Digital Forensic, 2012 International Conference, IEEE, *ISBN:978-1-4673-1425-1*, June 2012.

[5] Joshi A. and Geetha V., (2014) *SQL injection detection using machine learning*. IEEE, *ISBN 978-1-4799-4191-9*, Nov 2014

[6] Haijun Zhang (2011). *Textual and Visual Content-Based Anti-Phishing: A Bayesian Approach*. Neural Networks, IEEE Transactions, *Vol. 22, Issue 10*, 04 August 2011, *ISSN: 1045-9227*

[7] Lambert, N (2010). *Use of Query tokenization to detect and prevent SQL injection attacks*. Computer Science and Information Technology, 2010 3rd IEEE International Conference, *ISBN:978-1-4244-5537-9*, July 2010.

[8] Hang, Harry. *The Optimality of Naive Bayes*. FLAIRS2004 conference.

[9] Caruana, R.; Niculescu-Mizil, A. (2006). *An empirical comparison of supervised learning algorithms*, Proceedings of the 23rd international conference on Machine learning. *CiteSeerX: 10.1.1.122.5901.*

[10] NarasimhaMurty, M.; Susheela Devi, V. (2011). *Pattern Recognition: An Algorithmic Approach*. ISBN 0857294946.