IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

PREVENTION AND DETECTION OF SQL INJECTION ATTACK USING MACHINE LEARNING PREDICTIVE ANALYTICS

[1]Akshar Patel, [2]S Vishnu Vardhan, [3]B Sunil Kumar, [4]Ms. Shruti Kansal
[1,2,3]Student, [4]Assistant Professor
[1,2,3,4] Computer Science and Engineering, Vardhaman College of Engineering, Hyderabad, India

Abstract : The back-end database is fundamental for storing enormous information created by Web trades, for example, cloud-facilitated applications and IoT shrewd gadgets. Interlopers keep on utilizing the Structured Query Language (SQL) Injection Attack (SQLIA) to take private data set data, and the outcomes will be appalling. The current methods, which are largely signature techniques, are unable to deal with new signatures hidden in internet requests because they were all developed prior to the most recent problems of massive data mining. To dissect and forestalling SQLIA, elective machine learning (ML) prescient examination gives a helpful and versatile technique for mining enormous amounts of information.

Unfortunately, a common issue in SQLIA research is the lack of strong corpora, or data sets, that contain patterns and historical data items and can be used to train classifiers. In this work, we explore the construction of a data set that incorporates extraction from known attack patterns. Some examples of these patterns include SQL words and symbols that are present at injection locations. The data set is pre-processed, labeled, and feature hashed for supervised learning. The trained classifier will intercept SQLIA in internet requests, stopping malicious internet requests to get to back-end database. This paper gives broad proof of the implementation of ML predictive analysis that predicts and keeps away from SQLIA by using observational evaluations expressed in the Confusion Matrix (CM) and Receiver Operating Curve (ROC).

IndexTerms - SQLIA, SVM Classifier, Injection of SQL, data-driven SQLIA, Big data for SQLIA.

1. INTRODUCTION

SQLIA has normally been blamed on developers' loss of safety consciousness over the years. As a result, code-based solutions have been suggested as a way to prevent SQLIA. In addition, the SQLIA vulnerability arises from the SQL engine's well-intentioned free text processing, putting legacy and cloud installations without sanitation, at risk. The frequency of this kind of assault is established via the SQL corridor of shame [1], which covers present day pilfering of data through Injection Attack. As a result, the ability to defend back-end databases from SQLIA in an age of large records remains a present-day concern.

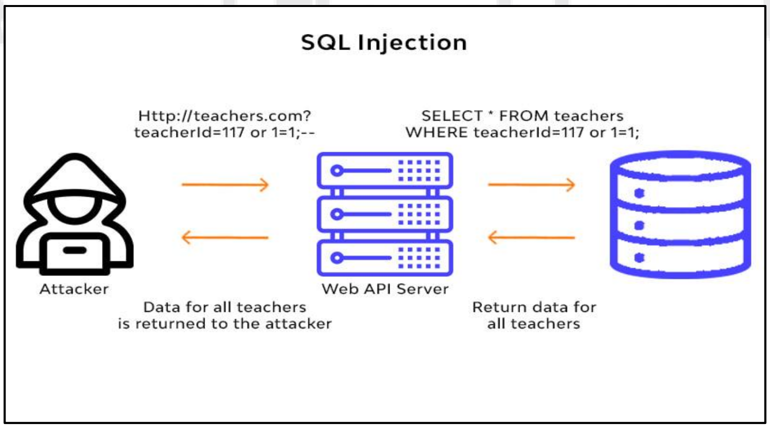

Fig. 1: Attack Scenario

The syntax of the SQL language might be basically the same as that of standard English [2], and the key expressions utilized in the SQLIA are additionally written in plain text. Thus, in the era of Bigdata, the SQLIA issue can be handled by building a classifier using machine learning predictive analytics, prepared using secure web patterns and old attack signatures.

Valid internet requests may take the form of predicted data for the web-application, whereas attack signatures at injection location may be SQLIA positive and include various styles of SQL tokens and symbols. In this paper, predictive analytics web interface is built, which trains the classifier with lots of mastering facts. The mastering facts are both SQL tokens (SQLIA high-quality) or capabilities of dictionary phrase listing styles (SQLIA negative).

One of the commitments of this study is a specialist informational index that undergoes feature hashing to make a learning model that precisely determines SQLIA utilizing the Support Vector Machine (SVM) strategy. This model prevents the goal to back-cease database from being accessed by fraudulent internet requests.

Also, provides a framework for prevention of SQLIA in data-heavy online environments and verification of ideas to the model that utilizes ML procedures like the Support Vector Machine (SVM), to foresee SQLIA. The experimental assessment inside the Receiver Operating Curve (ROC) depends on this procedure.

2. LITERATURE REVIEW

In the past, many attempts were made to detect and mitigate SQL Injection attack using various methods. SVM algorithm was also used by some, but it lacked text-preprocessing.

The following are works related to this paper using some popular approaches -

2.1 SQL intrusion detection using context-sensitive modelling:

Most modern multi-level application frameworks are based on high-speed data set frameworks to process and store business data. Aggressors are especially keen on these systems, which contain huge undertaking information, and extra safety efforts are taken to forestall inadvertent admittance to this data set layer. Using machine learning (ML) techniques like exception identification or grouping, a novel tool was presented by Bockermann et al. [3] to distinguish malicious behavior at the information set exchange level by utilizing the SQL parse tree structure [4] as a component in addition to researching feature vectorization of input data to a Classification algorithm but discovered there were issues with the computational burden of tree-kernels.

2.2 Efficient malicious code detection using N-gram analysis and SVM:

As the utilization of the web has developed, so has the scattering of electronic destructive projects. As an outcome, broad examination was being directed to forestall and distinguish hurtful code. Research has been carried out by Choi et al. [5] on intelligible articulations that need great language. Past word investigation or record characterization calculations can't arrange this type of destructive code. This examination gives a technique to extricating n-gram highlights from hurtful code and classifying executables as malignant or harmless, utilizing Support Vector Machine (SVM) as the ML classifier but only problem with this approach was, there were no sufficient patterns to improve the accuracy.

2.3 SQLi-GoT: Detecting SQL Injection Attacks using Graph of Tokens and SVM:

Identifying SQL infusion attacks has been a troublesome errand because of the gigantic inconstancy of the assault ways. In this exploration, Kar et al. [6] proposed an exceptional approach to distinguish infusion dangers by demonstrating SQL questions as a graph of tokens and preparing a Support Vector Machine (SVM) classifier utilizing the centrality proportion of hubs. The development was expected to chip away at the informational index firewall layer and protect various web applications working with climate. However, their focus was on online applications assembled utilizing PHP and MySQL, the technique is promptly versatile to different stages. The exploratory discoveries show that this technique may effectively recognize false SQL inquiries while causing little execution difficulty.

2.4 Numerical Encoding to Tame SQL Injection Attacks:

A numerical encoding to tame SQL attacks by adaptable numerical encoding [7] of features to dataset with stamped radiance assembled from broad web traffic inspection is introduced in this examination. In mathematical developments, encoding the version functions as a middleman to trap and get to the bottom of online information. This paper shows a strategy for integrating numerical qualities from any dataset into two-class logistic regression (TCLR) and two-class averaged perceptron (TCAP)-based artificial neural network and machine learning computations, individually. An experimental test of the version's ease in precisely arranging all legitimate internet requirements and SQLIA payloads is then conducted using this approach.

3. PROPOSED ARCHITECTURE

The existing project does not utilize a detection system to warn of a possible SQL attack, run to compromise the server. There is no user validation that informs the user to input correct credentials. It does not indicate the sort of attack used or the attacker who attempted to exploit the system.

When an attack happens, an ML-based Classification Algorithm will be utilized to mitigate it by identifying it. The utilization of defined system is essential to protect against SQL Injection Assaults.

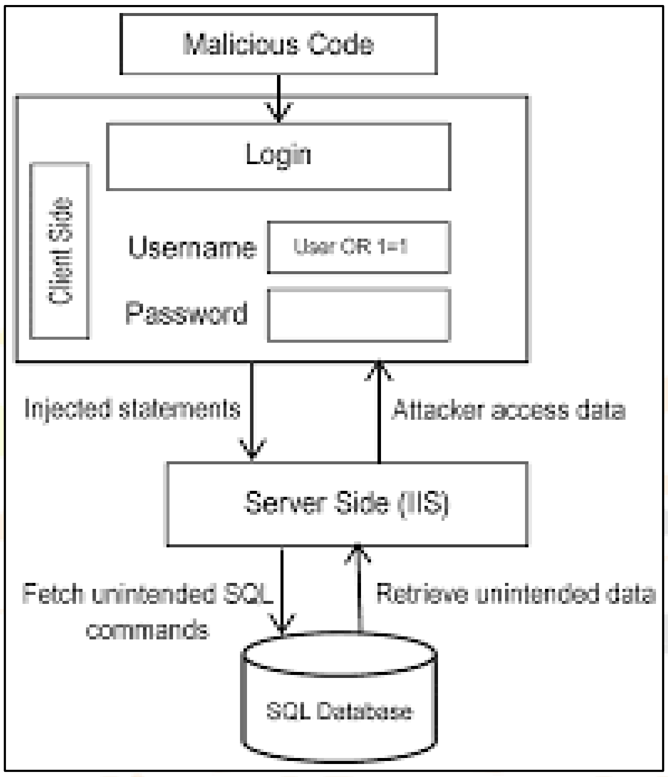

Fig. 2: Proposed Architecture

A) Malicious Code (SQL Structure)

The attacker may use any of the following attacks to access the database- Tautology, Union, Piggy-backed, Time-based attack and many more in the paper [8]. They are carried out using SQL statements which consists of two parts - predicates (Ex- 1=1), clauses (Ex- update, set, where), expressions (Ex- union, concat).

B) Login Form (Injection Point)

Attack may originate from web-page forms (Ex- Login Page or through cookies) where Predicate of SQL statement and expression after clause is the Injection point.

C) SQL Database (Back-end server)

The Internet requests of the user are sent to the server-side database where all the related data is stored, and in return it sends a response to the client system based on the need of the user.

4. METHODOLOGY

Below are the steps followed to implement the algorithm and build the model-

1) Data-set collection: The dataset is a collection of Microsoft dictionary words lists [9] which are labelled SQLIA negative and SQL symbols and tokens which are labelled SQLIA positive. The dataset is labelled based on pattern of input data which may either be tokens, semicolons, SQL symbols, comments or quotations etc.

2) Query pre-processing: It is done to improve accuracy of the model by removing missing words, normalizing data to lower case and recognizing patterns using regular expression to separate the input into valid and invalid requests.

3) Feature selection: The important features are selected by filtering the dataset in descending order to reduce the computation complexity and improve accuracy of the model.

4) Split of data into training and testing set: The dataset is divided into two sets where 80% is used for training while 20% is used as test data for prediction.

5) Training the model: The model is trained using SVM classifier to predict whether the request is valid or not. The algorithm uses data inputs which are labelled vectors and have undergone pre-processing to train the model to intercept web requests.

6) Prediction using Trained model: The trained model is deployed as a web-service, to predict the type of attack such as tautology attack, expressions, clauses or predicate attack.

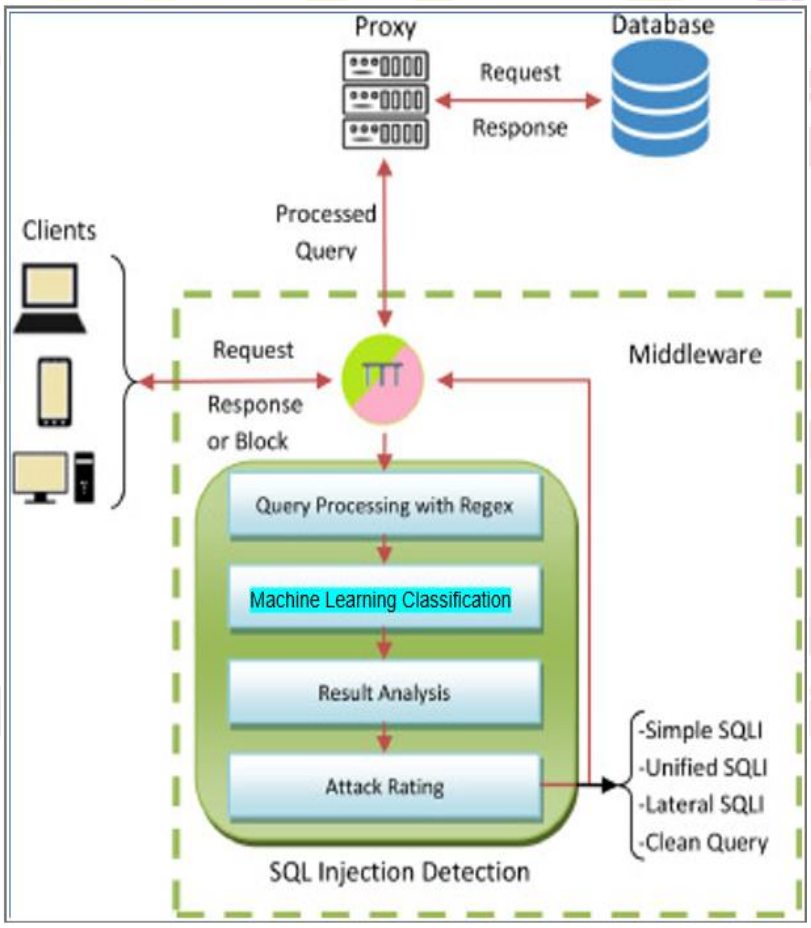

Fig. 3: Process model

5. EXPERIMENTAL RESULTS

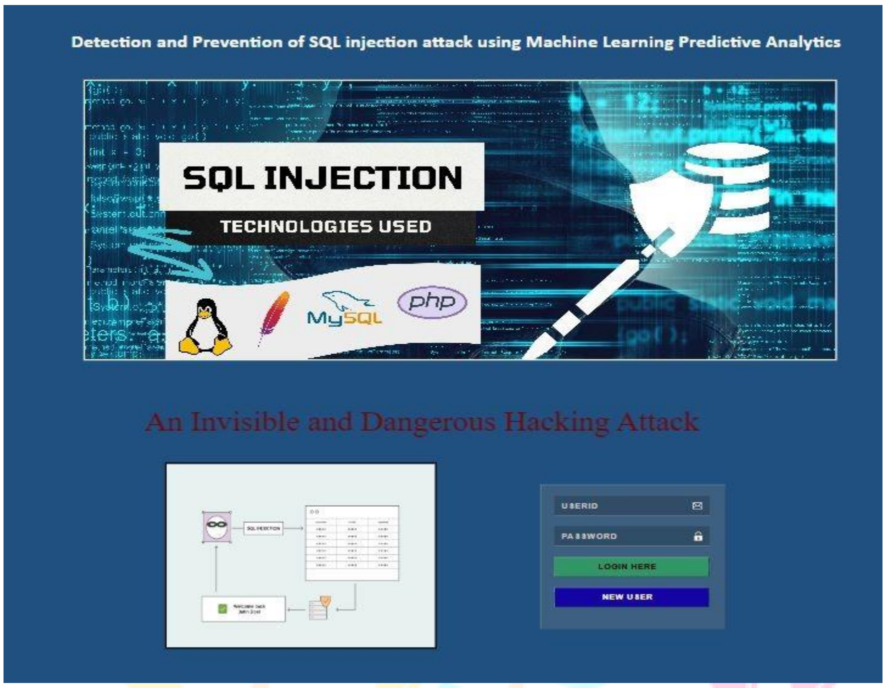
Detection and Prevention of SQL injection attack using Machine Learning Predictive Analytics
SQL INJECTION
TECHNOLOGIES USED
MySQL
php
An Invisible and Dangerous Hacking Attack
USERID
PASSWORD
LOGIN HERE
NEW USER

Fig. 4: Home screen

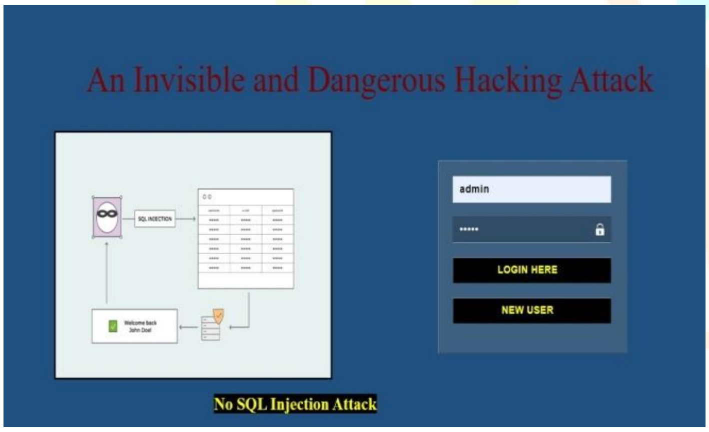

Fig. 5: No Attack

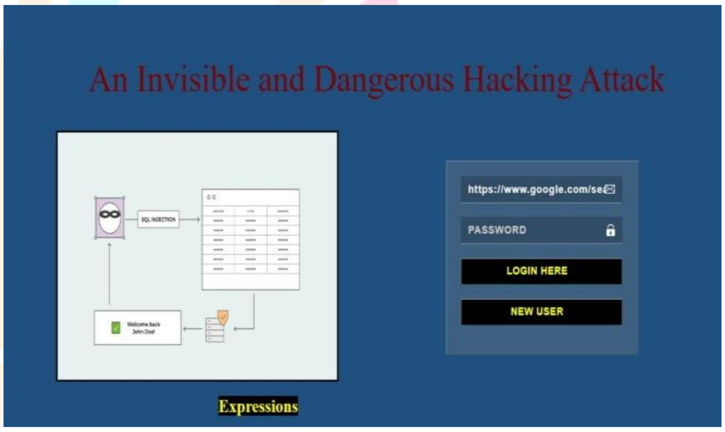
An Invisible and Dangerous Hacking Attack
https://www.google.com/se
PASSWORD
LOGIN HERE
NEW USER
Expressions

Fig. 6: Expressions Attack

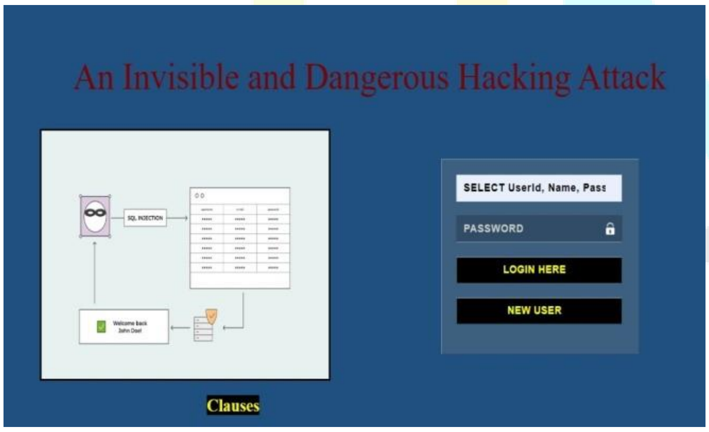
An Invisible and Dangerous Hacking Attack
SELECT UserId, Name, Pass
PASSWORD
LOGIN HERE
NEW USER
Clauses

Fig. 7: Clauses Attack

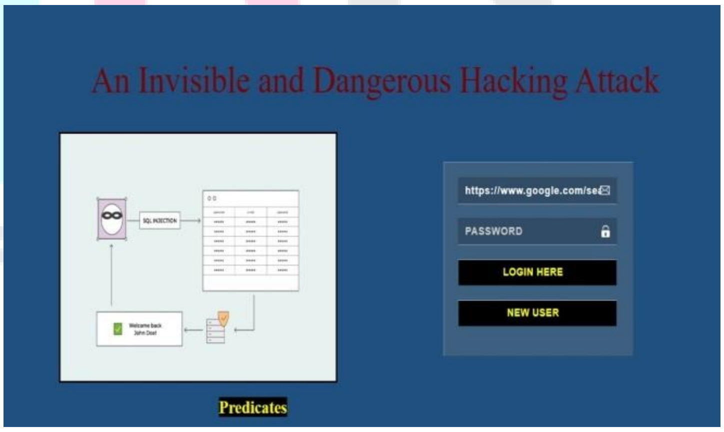
An Invisible and Dangerous Hacking Attack
https://www.google.com/se
PASSWORD
LOGIN HERE
NEW USER
Predicates

Fig. 8: Predicates Attack

6. PERFORMANCE METRICS

Table. 1: Comparison Table with stats

TECHNIQUE	CLASSIFIERS	PERFORMANCE	DATASET
Ingre e al.	Decision Tree	Accuracy=83.7%	NSL-KDD
moosa et al.	Neural Network	66.70%	300 SQL Injection signatures form diff. websites
Josha et al.	Navie Bayes	89.90%	178 codes including 100 valid and 78 malicious codes
SQLI-GOT	SVM	93%	Microsoft dictionary word list and SQL positive tokens and symbols

The table below provides information on various classifiers used to detect SQLIA and its accuracy to do so using specified datasets. We trained an SVM classifier which provides more accurate results utilizing the limited input data available.

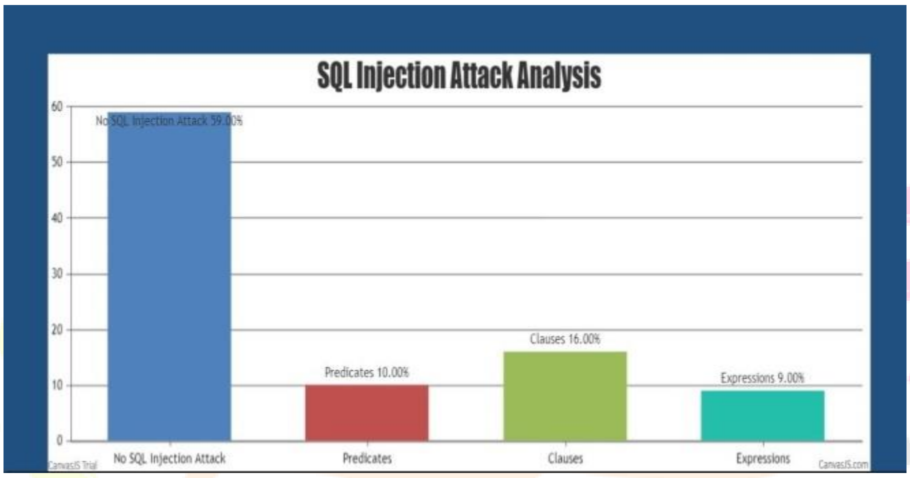

Fig. 9: Graphical Analysis

The stats in Fig.10 demonstrate if the request to the back-end server is valid or malicious, by specifying the type of attack performed by the intruder.

7. CONCLUSION AND FUTURE SCOPE

In this paper, we presented an approach to detect SQL injection attack using Machine Learning predictive analytics using SVM classifier to determine the attack type in big data scenario and the outcomes are tentatively surveyed in graphical manner. Parameterized Stored Procedures are used, which are the key defense to prevent SQL Injection Attack. When contrasted with past examinations to prevent and detect SQLIA, the methodology depicted here is useful in a huge information setting, which is missing in past endeavors on SQLIA as far as anyone is concerned. Future additions to this model include a multi-class classifier, which will be utilized to recognize and coordinate different SQLIA sorts.

REFERENCES

[1] CodeCurmudgeon, "SQLiHall-of-Shame," The Code Curmudgeon, 2016. [Online]. [Accessed: 12-Aug-2016]. Available: http://codecurmudgeon.com/wp/sqlinjection-hall-of-shame/.

[2] Microsoft, "Access SQL: basic concepts, vocabulary, and syntax - Access," MS Office, 2007. [Online]. Available:https://support.office.com/en-gb/article/Access-SQL-basic-concepts-vocabularyand-syntax-444d0303-cde1-424e-9a74-e8dc3e460671.

[3] C. Bockermann, M. Apel, and M. Meier, "Learning SQL for database intrusion detection using context-sensitive modelling (extended abstract)," in Lecture Notes in Computer Science, 2009, vol. 5587 LNCS, pp. 196–205.

[4] G. T. Buehrer, B. W. Weide, and P. A. G. Sivilotti, "Using Parse Tree Validation to Prevent SQL Injection Attacks," in Proceedings of the 5th international workshop on Software engineering and middleware SEM 05, 2005, no. September, p. 106.

[5] J. Choi, C. Choi, H. Kim, and P. Kim, “Efficient malicious code detection using N-gram analysis and SVM,” in Proceedings - 2011 International Conference on Network-Based Information Systems, NBiS 2011, 2011, pp. 618–621.

[6] D. Kar, S. Panigrahi, and S. Sundararajan, “SQLiGoT: Detecting SQL Injection Attacks using Graph of Tokens and SVM,” Comput. Secur., vol. 60, pp. 206–225, 2016.

[7] S. Uwagbole, W. Buchanan, and L. Fan, “Numerical Encoding to Tame SQL Injection Attacks,” in IEEE/IFIP DISSECT, 2016.

[8] W. G. J. Halfond, A. Orso, D. A. Kindy, and A. S. K. Pathan, “AMNESIA: Analysis and Monitoring for Neutralizing SQLinjection Attacks,” Int. J. Commun. Networks Inf. Secur., vol. 5, pp. 80–92, 2013.

[9] Available: https://msdn.microsoft.com/enus/library/ms18922.aspx. Microsoft, “Reserved Keywords (Transact-SQL),” MSDN. [online].

“acknowledgment” inAmericaiswithoutan “e” afterthe “g”.Avoidthestiltedexpression, “Oneofus(R.B.G.)thanks...” Instead,try“R.B.G.thanks”.Putapplicablesponsoracknowledgmentshere;DONOTplacethemonthefirstpageofyourpaperorasafootnote.Bhatti, U. and Hanif. M. 2010. Validity of Capital Assets Pricing Model.Evidence from KSE-Pakistan.European Journal of Economics, Finance and Administrative Science, 3 (20).