SQL Injection Prevention System using Machine Learning Algorithms [1,2,3]

Soubhik Sinha[1] , Tushar Kumar B P[2] , Sukant Jha[3] ,
School of Information Technology and Engineering ,
VIT University , Vellore , TN , India

Abstract

Existing Web system flaws jeopardize information systems' ability to operate normally. SQL injection is by far the most prevalent Web system flaw. In this article, you'll find information on how to secure Web applications from SQL injection attacks. To increase the security of Web applications, a defensive mechanism has been designed to defend Web resources against SQL injection. PHP, JavaScript, and regular expressions are utilized to implement this programme. As a result, a application for protecting Web applications against SQL injection vulnerabilities has been developed. The programme allows a user to utilize SQL to secure his own Web application from an attack.

Keywords: SQL Injection, Web App Security, Detection Strategies, Scanning Method, Prevention of SQL Injection Attacks, Input Validation, Parameterized Queries, Escaping, Stored Procedures, Mitigation using Log-In Gateway System

I. Introduction

Attackers frequently use SQL injection attacks to gain unauthorized access to systems. This programme was created to prevent SQL injection attacks from gaining unauthorized access to the system. This is accomplished by using a signature-based authentication mechanism to check validity and adding a unique value. These days, SQL injection is a critical security concern that allows an attacker to get access to an online system or application by exploiting certain vulnerabilities. This approach takes use of a variety of web application aspects, such as conveying traveling form data parameters and efficiently integrating amino acid codes that are aligned in it. To put it another way, this software project proposes a mechanism for analyzing and detecting malicious code in order to identify and prevent an attack. It employs a distinct signature-based scanning technique that employs a different divide-and-conquer strategy to detect assaults based on multiple time/space factors. Based on its effective attack detection methodologies, this unique technology has proven successful in blocking numerous SQL injection attacks.

II. Background Knowledge

Gradient Boosting works by adding predictors to an ensemble in a progressive manner, each one correcting the one before it. Unlike AdaBoost, however, this strategy aims to adapt the new predictor to the residual errors created by the prior predictor rather than modifying the instance weights at each iteration. Boosting algorithms work on the idea of first building a model on the training dataset, then building a second model to correct the faults in the first model. Gradient Boosting Regressor is used when the goal column is continuous, while Gradient Boosting Classifier is used when it is a classification problem.

The "Loss function" is the sole difference between the two. The goal here is to use gradient descent to reduce this loss function by adding weak learners.The first stage in gradient boosting is to create a base model that predicts the observations from the training dataset. For the sake of simplicity, we take the target column's average and assume it to be the anticipated value. This step can be stated mathematically as :

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$

L denotes our loss function, gamma (**γ**) denotes our predicted value, and **arg min** denotes that we must find a forecasted value/gamma for which the loss function is least. Our loss function will be (because the target column is continuous) :

$$L = \frac{1}{n} \sum_{i=0}^{n} \left(y_i - \gamma_i\right)^2$$

In this case, $\mathbf{y_i}$ is the observed value. And the projected value is **gamma(γ)**. Now we need to find a gamma value that is as low as possible while still minimizing this loss function – thus we shall use the below function:

$$\frac{\mathrm{dL}}{\mathrm{d}\gamma} = \frac{2}{2}\left(\sum_{i=0}^{n}\left(y_i - \gamma_i\right)\right) = -\sum_{i=0}^{n}\left(y_i - \gamma_i\right)$$

Remember that $\mathbf{y_i}$ represents our observed value and $\boldsymbol{\gamma}_i$ represents our expected value. The pseudo residuals, which are (observed value – anticipated value), are then calculated – via the following formula :

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

$F(x_i)$ is the prior model, and m is the number of DT produced. Now we need to determine the output values for each leaf of our decision tree. That means there's a chance one leaf will acquire more than one residual, therefore we'll need to figure out the total production of all the leaves. To obtain the output, simply take the average of all the numbers in a leaf, regardless of whether there is only one value or more than one – mathematically :

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

Here, $\mathbf{h_m(x_i)}$ is the residual-based DT, and m is the number of DT. When m=1, we are referring to the first DT, and when it is "**M**" we are referring to the last DT. The leaf's output value is the gamma value that minimizes the Loss function. The left-hand side "**Gamma**" represents the output value of a certain leaf. Finally, we must revise the prior model's predictions :

$$F_m(x) = F_{m-1}(x) + \nu_m h_m(x)$$

where **m** is the number of decision trees created.

Now this was all about Gradient Boosting , let us go towards Gradient Boosting Classifier. When the target column is binary, a gradient boosting classifier is applied. All of the procedures described in the Gradient boosting regressor are applied here; the only difference is that the loss function is changed.

When the target column was continuous, we utilized Mean squared error as our loss function; this time, we would use log-likelihood. The classification problem's loss function is shown below.

$$L = -\sum_{i=1}^{n} y_i \log(p) + (1-p)\log(1-p)$$

Let us first turn this loss function into a log function (odds)

$$\begin{aligned}
L &= -\left[\sum_{i=1}^{n} y_i \log(p) + \left(1-y_i\right) log(1-p)\right] \\
&= -y*\log(p) - (1-y)*\log(1-p) \\
&= -y*log(p) - log(1-p) + y*log(1-p) \\
&= y*\left[log(p) - log(1-p)\right] - log(1-p) \\
&= -y*\left[\frac{\log(p)}{\log(1-p)}\right] - log(1-p) \\
&= -y*log\left(\frac{p}{1-p}\right) - log(1-p) \\
&= -y*\log(\text{odds}) - \log(1-p)
\end{aligned}$$

$$\text{Now, } log(1-p) = log\left(1 - \frac{e^{\log(\text{odds})}}{1+e^{\log(\text{odds})}}\right)$$

$$\begin{aligned}
&= log\left(\frac{1+e^{\log(\text{odds})}}{1+e^{\log(\text{odds})}} - \frac{e^{\log(\text{odds})}}{1+e^{\log(\text{odds})}}\right) = log\left(\frac{1}{1+e^{\log(odds)}}\right) \\
&= log(1) - log\left(1+e^{\log(\text{odds})}\right) \\
&= 0 - log\left(1+e^{\log(\text{odds})}\right)
\end{aligned}$$

$$\text{Hence , } -y*log(\text{odds}) - \left(-log\left(1+e^{\log(\text{odds})}\right)\right)$$

$$L = -y*log(\text{odds}) + log\left(1+e^{\log(\text{odds})}\right)$$

Further , taking the derivative –

$$\frac{dL}{d[\log(odds)]} = -y + \frac{e^{\log(odds)}}{1+e^{\log(odds)}}$$

We know $\frac{e^{\log(odds)}}{1+e^{\log(odds)}} = p$, *hence we can substitue* p

$$\frac{dL}{d[log(odds)]} = -y + p$$

, here **y** is the observed values

Now ,

$$\frac{dL}{d[log(odds)]} = -y + p$$

$$\frac{dL}{d[log(odds)]} = -(-y + p) = (y - p) = (\text{observed} - \text{predicted})$$

Now that we have our first decision tree, we need to get the final output of the leaves because a leaf may have more than one residual, thus we need to compute the final output value –

$$\gamma = \frac{\sum_{i=1}^{n} \text{Residual}_i}{\sum_{i=1}^{n} [\text{Previous probability}_i \times (1 - \text{Previous probability}_i)]}$$

Finally, we are ready to generate fresh predictions by combining our base model with the residuals tree we created.

II. Literature Survey

Sr. No	Title (Year)	Authors	Algorithm / Method used	About
1.	SQL Injection Attacks Prevention System Technology	Fairoz Q Kareem Siddeeq Yousif Ameen Awder Ahmed Sulaimani Azar Abid Salih	Nikto, SQLMAP SQL injection with SVM SQL injection with neural network SQLA, DIAVA, SQL injection	1. All content management systems were not susceptible toassaults via SQLi but provided alerts on otherpotential flaws. 2. Launch new SQL assaults with comparable assault patterns and payloads with the current one 3. Its advantages are the accuracy of neural networks for detecting SQL injection is superior to the relevant machine learning algorithms. 4. DIAVA not only conducts advanced WAFs for SQLA detection from the point of view of precision and recall but also allows the assessment of leaked data causing SQL injection in real-time
2.	A Classification of SQL Injection Attacks and Countermeasures	William G.J. Halfond, Jeremy Viegas, and Alessandro Orso	Black Box Testing. Static Code Checkers AMNESIA (Combined Static and Dynamic Analysis). Taint Based Approaches New Query Development Paradigms Intrusion Detection Systems. Proxy Filters. Instruction Set Randomization	Well suited and efficient for large code segments Code access is not required.
3	SQL Injection Attacks: Techniques and Protection Mechanisms	Nikita Patel, Fahim Mohammed, Santosh Soni	Taking User Input from Predefined Choices. Bind Variables. Mechanism Parameterized Statements.	The attacker cannot insert custom queries or any type of harmful script which can disturbthe integrity of the database. helps in improving web-

			Input Validation.	application performance. Every passed string parameter ought to be validated. Many web applications use hidden fields and other techniques, which also must be validated. If a bind variable is not being used, special database characters must be removed or escaped.
4	PROTECTION OF WEB APPLICATION AGAINSTSQL INJECTION ATTACK	Manisha A. Bhagat, Prof. Vanita Mane	JDBC-Checker. ADMIRE. SQL-PROB. WAVES. SQLRand. SQL DOM. VIPER. CANDID.	SQL Injection attack is one of the most popular attacks used in system hacking or cracking. Web applications are consisting of web forms, web server and backend. These applications are vulnerable due to attacks and scripts as the number of web application users are increasing. When attacker gains control over web application maximum damage is caused.
5	SQL Injection Attack Detection and Prevention TechniquesUsing Machine Learning	Ines Jemal, Omar Cheikhrouhou, Habib Hamam, Adel Mahfoudhi.	Query-model based. Obfuscation based. Monitoring and Auditing Based. Entropy Based. Ontology Based.	The process begins by analyzing the source codeused in the web application and vulnerable in execution, then optimizing these vulnerable queries. The optimization engine generates a set of valid execution in accordance with the heuristic rules. Finally, the hotspot (vulnerable part of code) is replaced by the web application code with its optimized query. The authors in proposed an automated testing approach, namelyμ4SQLi.μ4SQLi can produce effective inputs that lead to executable and malicious SQL queries. The idea is to produce inputs that bypass web application firewalls. The goal of thistool is to detect potential SQL vulnerabilities in a given web application.
6	A novel technique to prevent SQL injection and cross-site scripting	Oluwakemi Christiana Abikoye, Abdullahi Abubakar, Ahmed	Formation of SQL injection string patterns.	Every form of attacks has certain characters and keywords that hackers do

	attacks using Knuth-Morris-Pratt string match algorithm	Haruna Dokoro, Oluwatobi Noah Akande& Aderonke Anthonia Kayode	Designing parse tree for the various formsof attacks.	manipulate to perpetuate their attacks these characters and keywords are used to form malicious codes that are used to carry out the various forms of attacks. Identifying these injection codes will help in coming up with how todetect and prevent theseattacks. Parse tree was used to represent the syntactic pattern of the various forms of SQL-Injection and Cross Site Scripting attacks
7	An Efficient Technique for Detection and Prevention of SQL Injection Attack using ASCII Based String Matching	Indrani Balasundarama, E. Ramarajb	Proposed SQLIA - Protector Mechanism An Authentication Scheme using Hybrid Encryption	Because of high level security the proposed scheme consists of three filtration phases with static and dynamic level. First level phase is Malicious Text Detector, second level phase is Constraint Identifier, third level phase is StaticQuery Analysis and the Fourth level Phase is Textbased Key Generator.
8	TransSQL: A Translation and Validation-based Solution for SQL-Injection Attacks	Kai Xiang Zhang, Chia-Jun lin et al	a solution in form of TransSQL to detect malicious SQL queries	It is a server side solution, so modificationor update is not requiredin the legacy web application. Automatic and platform independent deploymentand installation. No need to learn LDAP, thus user's training cost is null.
9	An Approach for SQL Injection Vulnerability Detection- AMNeSIA 2009	M. Junjin	Static analysis Runtime analysis	Automatically generate SQL query on the basisof possible legitimate queries by analyzing web application code. scans all dynamically generated SQL queriesand checks their compliance to the statically generated models.

10	A SQL Injection Detection Method Based on Adaptive Deep Forest (2019)	Qi Li, Weishi Li, Junfeng Wang, Mingyu Cheng	1 An adaptive deep forest-based method is incorporated to detect the complexSQL injection attacks. 2 An algorithm, AdaBoost, algorithm is used. Itis based on deep forest model whichutilizes error rate toupdate the weights of features on each layer.	1 Using this methodology,the existing problem of deterioration in efficiency of features ofdeep forests as the number of layers increases is tackled completely. 2 The structure of the tree model can be modified automatically. 3 Multi-dimensional fine-grained features can be dealt with properly to avoid the problem of over-fitting. 4 The methodology used experimentally demonstrates not only better detection accuracyand low computational cost but also high flexibility and high robustness.
11	SQL Injection Detection for Web Applications Based on Elastic-Pooling CNN (2019)	Xin Xie, Chunhui Ren, Yusheng Fu, Jie Xu, Jinhong Guo	1. Convolutional Neural Network (CNN) is a powerful deep feedforward neural network which can replicate the formation mechanism employed in organisms for vision cognition. 2. CNN is applied tothe detection of SQL injection in Web applications,and detection of SQL injection attacks from veryvast web logs is done.	1. Identification of new and harmful attacks can be successfully done with the help of irregular matching characteristicsand is thus, much more difficult to bypass. 2. Since the vocabulary is usually small, the training difficulty and cost are reduced considerably. 3. This method can also be beneficially used as an auxiliary method of existing traditional SQL detection methods.

			3. This method automatically extracts the hidden common features ofSQL injection and is thus able to identify new attacks, bypassing the regular SQL injection.	
12	JS-SAN: defense mechanism for HTML5- based web applications against javascript code injection vulnerabilities (2016)	Shashank Gupta, B.B. Gupta	1. injection and clustering-based sanitization framework, i.e. JS- SAN (JavaScript SANitizer) for the mitigation of JS codeinjection vulnerabilities. 2. It generates an attack vector template by performing the clustering on the extracted JS attack vector payloads corresponding to their level of similarity. Asa result, it then sanitizes the extractedJS attack vector template by an automated technique of placement of sanitizers in the source code of generated templates of web applications.	1. The proposed framework validates its novelty by producing a less rate of false negatives and tolerable runtime overhead as compared toexisting sanitization- based approaches.
13	Detection and Prevention of Code Injection Attacks on HTML5-Based Apps (2015)	Xi Xiao; Ruibo Yan; Runguo Ye; Qing Li; Sancheng Peng; Yong Jiang	1. JavaScript code is encoded in a human-unreadable form. Then we use classification algorithms of machine learning to determine whether anapp suffers from the code injection attack or not.	1. Precision of the proposed detection method reaches 95.3%. 2. Compared to the other methods, new approach improves a lot in detection speed with theprecision nearly unchanged. 3. An improved access control model is

				proposed to mitigate theattack damage. 4. Filters are adopted to remove JavaScript codefrom data to prevent theattacks.
14	Detecting and Removing Web Application Vulnerabilities with Static Analysis and Data Mining (2016)	Ibéria Medeiros; Nuno Neves; Miguel Correia	1. We combine taint analysis, which finds candidate vulnerabilities, withdata mining, to predict the existenceof false positives 2. Automatic code correction by inserting fixes in thesource code. 3. The proposed approach was implemented in the WAP tool, and an experimental evaluation was performed with a large set of PHP applications.	1. It is an enhanced form of detection. 2. This novel study brings together two approaches that have been considered orthogonal until now. 3. It combines humans coding the knowledge about vulnerabilities (fortaint analysis) and automatically obtaining that knowledge (with machine learning, for data mining).

15	Study On SQL Injection Attacks: Mode, Detection And Prevention (2016)	Subhranil Som, Sapna Sinha ,Ritu Kataria	1. The paper has dealt with the security onthe ends by proposing the two systems for avoiding SQL Injection Attacks. Its two stages are: a) Frontend Phase b) Backend 2. Initially at front endthe Database is secured from any SQLIA. An additional section in client table is uesd to store the Final Hash Code, which is obtained during enrollment time of a client for the first time and isput into client tablealong with client name and secret key. 3. In the backend phase, The framework noticeson how SQLIA onWeb applications by tokenization and encryption for detection and prevention.	1. This paper has demonstrated a strategy to change over SQL query into number of helpful tokens by applying tokenization and after that encoding all literals, fields, table and information on the query by AES-algorithmto avoid SQLIA. 2. This methodology encourages quick and proficient getting to system with database and keeps away from memory necessities to store the actual query instorehouse. 3. This methodology because of its low preparing overhead hasimmaterial impact on execution even at higherburden conditions and does not require real changes to application code.
16	Web based testing application security system using	Akbar Iskandar, Muhammad Resa Fahlepi Tuasamu ,	1. This research uses semantic comparison method to detect SQL injection. Semantic comparisons are	1. The design and testing ofthe system on the registration form shows that if there is field that has not been filled or emptyon the registration

	semantic comparison method (2018)	Suryadi Syamsu , M Mansyur , Tri Listyorini , Sulfikar Sallu , S Supriyono, Kundharu Saddhono , Darmawan Napitupulu and Robbi Rahim	performed by parsingeach statement and comparing syntactic data structures. If the syntax structure of the two queries is equivalent, then the query induces equivalent semantic action on the databaseserver. 2. Research based on ause case of a registration and loginform.	form then it shows a message notification that the field must be filled and after all the fields are filled, semanticcomparison will compare thequery in the system database.The password created by the user has been encrypted and the encrypted password will be decrypted using MD5 decrypter where the data is encrypted using an encryption key to be something that is difficult to read by someone. Therefore, every website or web application needs to apply the semantic comparison method and apply Message Digest 5 (MD5) algorithm which is widely used to prevent the occurrence of attacks from creckers who are trying to find a web vulnerability because most applications accessed via the internet hasa login page which can be used to authenticate app users. The attacker intends to log in without using the correct username and correct password. But as if entering the correct username, where if the attacker uses injection like "hacker OR 1 '=' 1'-" as username and suppose "something" is used as password, then the query willbe like this: Select * from login where user = 'hacker' OR '1' = '1' - 'and pass =' something 'When this query is run in the

				database, it will always be considered correct and authentication will work. Based on the results and designof research, it can be concluded that using semantic comparison method can prevent dangerous Structured Query Language Injection Attack and can secure the account of web testing users by using MD5.
17	SQL Injection (2013)	Stephanie Reetz, SOC Analyst	1. This paper talks about various SQL injection vulnerability, attack scenarios, blind SQL Injection and mitigation.	1. The three main defense strategies against SQL injection are parameterized queries, stored procedures, and input validation. The first option is the use of parameterized queries. They require that all SQL code isfirst defined and then parameters are passed to thequery. They are easier to write than dynamic queries and help prevent SQL injection by differentiating between the SQL code and the user-supplied data. 2. The second defense strategy, comparable to the first, is theuse of stored procedures which prevent SQL injectionas long as they do not include any unsafe dynamic SQL generation. 3. The third approach is to escape all user-supplied input before adding it to a query. When user supplied input is escaped, special characters are replaced withcharacters the database should not confuse with SQL code written by the developer. The specific functions used for escapinguser supplied input vary byserver-side scripting language.

18	A Study on SQL Injection Techniques (2016)	Rubidha Devi.D, R.Venkatesan , Raghuraman.K	1. Comprehensive coverage about topicslike basics of SQL Injection, types, recent attacks as a case study. 2. Tautology SQL injection – one of thecode injection techniques is widely used as a data – driven attack as per the security related literature and causes severe damage to theorganizational data banks.	1. SQL Injection through Tautologies used for bypassing authentication like -> Select*from user detwhere uid='abcd' and pwd ='a' or'3'='3' 2. SQL Injection through Union used for extracting Datalike -> Select * from userdet where uid=''union select * from details -- and pwd='a'; 3. SQL Injection through Piggybacked Queries to extract different datasets like -> SELECT Rno FROM St WHERE login = 'abc' AND pass = ''; DROP table St --' 4. SQL Injection through InferenceDetermining used for database Schema like -> SELECT name, email FROM members WHERE id=1; IF SYSTEM_USER='sa' SELECT1/0 ELSE SELECT 5 5. SQL Injection through Stored Procedure used for executing remote Commands like -> SELECT Eid, Ename FROM Employee WHERE Ename LIKE '8' or '8' = '8'; EXEC master.dbo. xp_cmdshell'dir'--'

19	Using Parse Tree Validation to Prevent SQL Injection Attacks	Gregory T. Buehrer, Bruce W. Weide, and Paolo A. G. Sivilotti	1.A parse tree is a data structure for the parsed representation of a statement. Parsing a statement requires the grammar of the statement'slanguage. By parsing two statements and comparing	1. Most web applications employa middleware technology designed to request information from a relational database in SQL. SQL injection is a common technique hackers employ t attack these web-based applications. These attacks

20	Detection of SQL Injection attacks: A Machine Learning Approach	Musaab Hasan, Zayed Balbahaith, Mohammed Tarique	This paper talks about various SQL injection vulnerability, attack scenarios, Support Vector Machine(SVM), Naïve Bayes, RBC control mechanism and SQL detection.	It is easy and fast to predict theclass of the test data set. It also performs well in multi-class prediction. When assumption ofindependence holds, a Naive Bayes classifier performs bettercompare to other models like logistic regression and you needless training data.
21	Applied Machine Learning Predictive Analytics to SQL Injection Attack Detection and Prevention	T.P.Latchoumi, Manoj Sahit Reddy, K.Balamurugan	The proposed detection model may report vulnerabilities in web applications.	Continuous web security against OWASP Give developers a confidence boostin thei own code.
22	Applied Machine Learning Predictive Analytics to SQL Injection Attack Detection and Prevention	Solomon Uwagbole, William J Buchanan, L.Fan	Support vector machine learning, proxy filters	SVM works relatively well when there is a clear margin of separation between classes. SVM is more effective in high dimensional spaces. SVM is effective in cases where the number of dimensions is greater than the number of samples. SVM is relatively memory efficient.
23	An Improved SQL Injection Attack Detection Model Using Machine Learning Techniques	Yazeed Abdulmalik	Sql static and dynamic Analysis.	It identifies vulnerabilities in a runtime environment. Automated tools provide flexibility on what to scan for. It allows for analysis of applications in which you do not have access to the actual code.
24	SQL Injection Detection Using Machine Learning 2019	Sonali Mishra	SQL Injection (Union, Blind. Error), Naïve Bayes, Ensemble Learning, Bagging and Boosting.	Performance: An ensemble can make better predictions and achieve better performance than any single contributing model. Robustness: An ensemble reduces the spread or dispersion of the predictions and model performance.

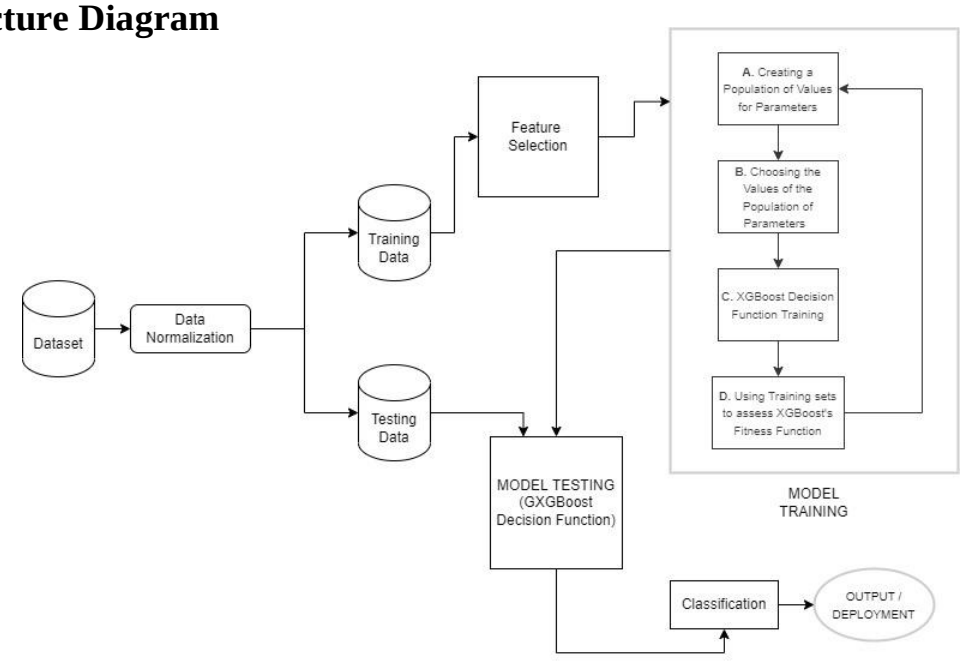
25	Detection of SQL Injection using Machine Learning: A Survey-Nov2019	Tareek Pattewar, Hitesh Patil, Harshada Patil , Neha Patil , Muskan Taneja , Tushar Wadile	Naïve bayes, bagging and boosting, gradient boosting Encryption alog's: MD5 algorithm, AEs algorithm	MD5 Algorithms are useful because it is easier to compare and store these smaller hashes than store a large variable length text. ... Moreover, it is very easy to generate a message digest of the original message using this algorithm.
26	SQL Injection Detection using Machine Learning	Anamika Joshi and Geetha V.	Using Naïve Bayes ML algorithm.	It is easy and fast to predict the class of the test data set. It also performs well in multi-class prediction. When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
27	SCAMM: Detection and Prevention of SQL Injection Attacks	Auninda Alam, Marjan Tahreen, Md Moin Alam, Shahnewaz Ali Mohammad, Shohag Rana	They use machine learning algorithms in order to train their model so that it may detect any known kind of SQL injection attempts and even identify new approaches of attacks from its previous training	This model can be used to supplement the protective measures of any kind of website. It can be used to protect the database of various web applications. It is expected to serve as an added layer of protection to database servers of any kind of system
28	Prevention Of Sql Injection Attack Using Unsupervised Machine Learning Approach	M.N.Kavith a, V. Vennila, G.Padmapri ya, A. Rajiv Kannan	Sql Injections Using Machine Learning Serves As prevention Mechanism To The Utilization Of Any Database In The Server Side. In This Methodology, A Segment Is Built Up That Lives In The Server Side[12]. The System Checks For The Sql Injection Attack Patterns That Are Affixed With The Values Passed To The Application Server Prior To The Processing Of The Query Made By The Client	. Two Levels Of Security Are Defined In This System: In The First Level The Patterns Produced By The Cfg Rules And Relating These Values With The Pattern Produced By The Rules That Are Set For Sql Attacks
29	Web Application Attacks Detection Using Machine Learning Techniques	Gustavo Betarte , Rodrigo Mart´ınez and Alvaro Pardo	Here they are using some ML techniques for Web Application Attacks Detection.	Most web applications employa middleware technology designed to request information from a relational database in SQL.

30	SQL-IDS: Evaluation of SQLi Attack Detection and Classification Based on Machine Learning Techniques	Naghmeh Moradpoor Sheykhkanlo o	Here they have proposed a neural network-based model for detection and classification of SQLi attacks.	The NN model deals with the URL addresses that have already been classified into either benign or malicious. It also has knowledge about the type of SQLi attack for malicious URLs. The NN model receives this information from the URL classifier and takes it into account for three phases of: training, validating and testing with distribution rates of 70%, 15%.

III. Proposed Methodology

SQL Injections are one of the most used web attack vectors, which is used with the goal of retrieving sensitive data from organizations. Stolen Credit Cards or password files happens through SQL injection vulnerabilities. But now, we have ways to protect various websites from SQL Injection attacks. SQL Injection is a technique that is used by the attackers in which they insert SQL query into input fields which will then be processed by the underlying SQL database. These weaknesses will then be abused when entry forms allow the user generated SQL statements to query the database directly. There are various threats that an individual or an organization may suffer due to such attacks. Threats like extraction of private data such as credit card information, passport information, hospital records, etc. and sometimes enumeration of the authentication user details, allowing these logins to be used on other website along with a corrupted database as well as execution of OS commands, deleted/inserted data and destroyed operations for the entire website that might lead to system compromise. So, there is a need for developing a technique that will be able to protect the sensitive data of an individual or an organization and prevent such attacks in the future.

Architecture Diagram



In this programme, we have extensively used Regular Expressions. A regular expression (also known as a rational expression) is a set of characters that create a search pattern, mostly for pattern matching with strings or string matching, i.e., "find and replace" operations. It provides a robust and flexible pattern match that can assist us in implementing sophisticated database search tools. When matching regular expression patterns, the REGEXP operator is utilized. The word RLIKE is a synonym. It also includes a set of metacharacters that provide you additional freedom and control during pattern matching. As an escape character, the backslash is employed. If double backslashes are used, it is only considered in the pattern match. Above all, it is unaffected by case. We have another term – whose concept is inculcated with the usage of REGEXP – namely , Tokenization. Tokenization in Python is the process of breaking down a huge body of text into smaller lines, words, or even inventing new terms for a language other than English. We are using REGEXP for tokenizing each SQLI (SQL Injection) attack. Later the token count values will be considered for generation of a dataset. Apart from generating our own dataset , we shall also opt for importing / obtaining dataset from other website(s) – like Kaggle , Data.Gov , Datahub.io , UCI Machine Learning Repository , etc. Now , if the dataset is an imported one , we will have to tokenize the raw SQL (from the dataset). But SQL tokens are nothing but garbage for the Machine Learning Model – thus , we have to set real features for the aforementioned. After listing of feature vectors (feature detection done here) and Encoding categorical features , the dataset shall be splitted into Training and Testing dataset (the proportion is kept as 80% and 20% respectively – as training dataset has to be atleast $3/4^{th}$ (75%) the whole dataset OR the testing dataset has to be at most $1/4^{th}$ (25%) of the whole dataset. This can be done using train_test_split function of Scikit-learn Library (under model selection). Here , the algorithm taken is Gradient Boosting Classifier. The model training Gradient Boosting Classifier model consists of 4 parts. After extensive training , we shall proceed for testing the model on the testing dataset – for which , the dataset has to be tokenized. Afterwards , it shall denote whether the dataset-code is containing SQL Injection codes OR is it a normal / acceptable code – Hence , showing the classification result(s).

IV. Results

```
sqli     5921
plain    3694
Name: type, dtype: int64
Gradient Boosting Tree Acurracy: 1.000000
----- Result: -----
Machine Check Count: 13780
Sql-injection: 9731
Plain-text: 4049
--- 28636.5072966 seconds
```

The dataset taken has 13780 codes , of which the Gradient Boost Classifier could detect 9731 codes which are SQLI codes , and could detect 4049 normal / valid codes – without anomalies.

V. Conclusion

Injection vulnerabilities are still the most common and deadly web application attacks. In this paper we have represented SQL injection prevention using REGEXP (Regular expression) and did the analysis using Gradient Boosting Classifier algorithm. To perform the aforementioned, we either created our own dataset using Regular expression and tokenization OR imported dataset from any other trusted website. Data normalization techniques were used to sweep of any anomalies. Further , model training was done after dataset splitting in training and testing components. Tokenization method was again taken into consideration for model testing – resulting to successful model operation in classifying SQLI codes and normal / valid codes.

VI. References

[1] Kareem, F. Q., Ameen, S. Y., Salih, A. A., Ahmed, D. M., Kak, S. F., Yasin, H. M., Ibrahim, I. M., Ahmed, A. M., Rashid, Z. N., & Omar, N. (2021). SQL Injection Attacks Prevention System Technology: Review. *Asian Journal of Research in Computer Science*, *10*(3), 13-32. https://doi.org/10.9734/ajrcos/2021/v10i330242

[2] Halfond, William & Viegas, Jeremy & Orso, Alessandro. (2006). A Classification of SQL Injection Attacks and Countermeasures.

[3] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.301.5263&rep=rep1&type=pdf

[4] Manisha A. Bhagat , Vanita Mane (2013) , Protection of Web Application Against SQL Injection Attack

[5] Jemal, Ines & Cheikhrouhou, Omar & Hamam, Habib & Mahfoudhi, Adel. (2020). SQL Injection Attack Detection and Prevention Techniques Using Machine Learning. International Journal of Applied Engineering Research. 569-580.

[6] Abikoye, O.C., Abubakar, A., Dokoro, A.H. *et al.* A novel technique to prevent SQL injection and cross-site scripting attacks using Knuth-Morris-Pratt string match algorithm. *EURASIP J. on Info. Security* **2020,** 14 (2020). https://doi.org/10.1186/s13635-020-00113-y

[7] An Efficient Technique for Detection and Prevention of SQL Injection Attack using ASCII Based String Matching Indrani Balasundaram, Ramaraj E.

[8] International Journal of Engineering Applied Sciences and Technology, 2016 Vol. 1, Issue 8,ISSN No. 2455-2143, Pages 23-29 Published Online June - July 2016 in IJEAST (http://www.ijeast.com) “Study On SQL Injection Attacks: Mode, Detection And Prevention” by Subhranil Som AIIT, Amity University Uttar Pradesh, Noida, India ,Sapna Sinha AIIT, Amity University Uttar Pradesh, Noida, India and Ritu Kataria AIIT, Amity University Uttar Pradesh

[9] Q. Li, W. Li, J. Wang and M. Cheng, "A SQL Injection Detection Method Based on Adaptive Deep Forest," in IEEE Access, vol. 7, pp. 145385-145394, 2019, doi: 10.1109/ACCESS.2019.2944951.

[10] X. Xie, C. Ren, Y. Fu, J. Xu and J. Guo, "SQL Injection Detection for Web Applications Based on Elastic-Pooling CNN," in IEEE Access, vol. 7, pp.

[11] 2nd Nommensen International Conference on Technology and Engineering IOP Publishing IOP Conf. Series: Materials Science and Engineering 420 (2018) 012122 doi:10.1088/1757- 899X/420/1/012122 " Web based testing application security system using semantic comparison method" by Akbar Iskandar, Muhammad Resa Fahlepi Tuasamu , Suryadi Syamsu , M Mansyur ,Tri Listyorini , Sulfikar Sallu , S Supriyono, Kundharu Saddhono , Darmawan Napitupulu and Robbi Rahim

[12] Stephanie Reetz (2013) , SQL Injection

[13] Devi, Rubidha & Venkatesan, R. & Koteeswaran, Raghuraman. (2016). A study on SQL injection techniques. International Journal of Pharmacy and Technology. 8. 22405-22415.

[14] Gregory Buehrer, Bruce W. Weide, and Paolo A. G. Sivilotti. 2005. Using parse tree validation to prevent SQL injection attacks. In Proceedings of the 5th international workshop on Software engineering and middleware (SEM '05). Association for Computing Machinery, New York, NY, USA, 106–113. DOI:https://doi.org/10.1145/1108473.1108496

[15] Shashank Gupta and B. B. Gupta. 2016. JS-SAN: defense mechanism for HTML5-based web applications against javascript code injection vulnerabilities. Sec. and Commun. Netw. 9, 11 (July 2016), 1477–1495. DOI:https://doi.org/10.1002/sec.1433

[16] X. Xiao, R. Yan, R. Ye, Q. Li, S. Peng and Y. Jiang, "Detection and Prevention of Code Injection Attacks on HTML5-Based Apps," 2015 Third International Conference on Advanced Cloud and Big Data, Yangzhou, 2015, pp. 254-261.

[17] I. Medeiros, N. Neves and M. Correia, "Detecting and Removing Web Application Vulnerabilities with Static Analysis and Data Mining," in IEEE Transactions on Reliability, vol. 65, no. 1, pp. 54-69, March 2016. doi:10.1109/TR.2015.2457411

[18] Abdulmalik, Y. (2021). An Improved SQL Injection Attack Detection Model Using Machine Learning Techniques. International Journal of Innovative Computing, 11(1), 53–57. https://doi.org/10.11113/ijic.v11n1.300

[19] Uwagbole, Solomon & Buchanan, William & Fan, L.. (2017). Applied Machine Learning Predictive Analytics to SQL Injection Attack Detection and Prevention. 10.23919/INM.2017.7987433.

[20] Applied Machine Learning Predictive Analytics to SQL Injection Attack Detection and Prevention. *European Journal of Molecular & Clinical Medicine*, 2020; 7(2): 3543-3553.

[21] M. Hasan, Z. Balbahaith and M. Tarique, "Detection of SQL Injection Attacks: A Machine Learning Approach," *2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, 2019, pp. 1-6, doi: 10.1109/ICECTA48151.2019.8959617.

[22] Ines Jemal , Omar Chelkhrouhou , Habib Hamam , Adel Mahoudhi (2020) , SQL Injection Attack Detection and Prevention Techniques Using Machine Learning

[23] M.N. Kavitha , V. Vennila , G. Padmapriya , A. Rajiv Kannan (2021) , Prevention of SQL Injection Attack Using Unsupervised Machine Learning Approach

[24] Moradpoor, Naghmeh. (2015). SQL-IDS: Evaluation of SQLi Attack Detection and Classification Based on Machine Learning Techniques.

[25] Joshi, Anamika & Geetha, Vasantha. (2014). SQL Injection detection using machine learning. 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies, ICCICCT 2014. 1111-1115. 10.1109/ICCICCT.2014.6993127.

[26] Abdelhamid Makiou, Youcef Begriche, Ahmed Serhrouchni. Improving Web Application Firewalls to detect advanced SQL injection attacks. Information Assurance and Security (IAS), 2014 10th International Conference on, University of Okinawa, Japan, Nov 2014, OKINAWA, Japan. pp.35-40, 10.1109/ISIAS.2014.7064617. hal-01137542

[27] Friedman, N., Geiger, D. & Goldszmidt, M. Bayesian Network Classifiers. *Machine Learning* **29,** 131–163 (1997). https://doi.org/10.1023/A:1007465528199

[28] Naghmeh Moradpoor Sheykhkanloo. 2015. SQL-IDS: evaluation of SQLi attack detection and classification based on machine learning techniques. In Proceedings of the 8th International Conference on Security of Information and Networks (SIN '15). Association for Computing Machinery, New York, NY, USA, 258–266. DOI:https://doi.org/10.1145/2799979.2800011

[29] "A learning-based neural network model for the detection and classification of sql injection attacks," International Journal of Cyber Warfare and Terrorism (IJCWT), vol. 7, no. 2, pp. 16–41, 2017.

[30] R. Verbruggen and T. Heskes, "Creating firewall rules 579 International Journal of Applied Engineering Research ISSN 0973-4562 Volume 15, Number 6 (2020) pp. 569-580 ©Research India Publications. http://www.ripublication.com with machine learning techniques," Ph.D. dissertation, Kerckhoffs institute Nijmegen, 2015.

[31] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 darpa off-line intrusion detection evaluation," Computer networks, vol. 34, no. 4, pp. 579–595, 2000.

[32] Z. Wang, "The applications of deep learning on traffic identification," lackHat USA, 2015.

[33] B. Ingre, A. Yadav, and A. K. Soni, "Decision tree based intrusion detection system for nsl-kdd dataset," in International Conference on Information and Communication Technology for Intelligent Systems. Springer, 2017, pp. 207–218.

[34] A. Moosa, "Artificial neural network based web application firewall for sql injection," World Academy of Science, Engineering and Technology, vol. 4, 2010.

[35] D. Kar, S. Panigrahi, and S. Sundararajan, "Sqligot: Detecting sql injection attacks using graph of tokens and svm," Computers & Security, vol. 60, pp. 206– 225, 2016. [54] . P. G. Betarte and R. Martnez, "Web application attacks detection using machine learning techniques," in 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Dec 2018, pp. 1065–1072.

[36] I. T. Holdings, "Modsecurity: Open source web application firewall," http://www.modsecurity.org/,

[37] "Dataset csic-2010," http://www.isi.csic.es/dataset/.

[38] "Dataset drupal," https://www.drupal.org/project/dataset.

[39] "Dataset pkdd2007," http://www.lirmm.fr/pkdd2007-challenge/. [59] T. K. George, K. P. Jacob, and R. K. James, "Token based detection and neural network based reconstruction framework against code injection vulnerabilities," Journal of Information Security and Applications, vol. 41, pp. 75–91, 2018.

[40] CWE/SANS, "CWE/SANS most top 25 dangerous software http://www.sans.org/top25-software-errors/,

[41] Clark, A. D.: Modeling the Effects of Burst Packet Loss and Recency on Subjective Voice. IPtel 2001 Workshop.

[42] Jiang, W.: QoS Measurement and Management for Internet Real-time Multimedia Services. Columbia University, PHD Thesis, (2003).

[43] Anderson, D. Frivold, T. Valdes, A.: Next-generation Intrusion Detection Expert System (NIDES) : A Summary.
[44] Stevens, R. W.: TCP/IP Illustrated, Volume 1: The Protocols. Addison-Wesley (1994).

[45] Stevens, R. W.: TCP/IP Illustrated, Volume 2: The Implementation. Addison- Wesley (1995).

[46] The LIBNET project homepage: http://www.packetfactory.net/libnet/.
Accessed March 16, 2005.

[47] The libpcap project homepage: http://sourceforge.net/projects/libpcap/. Accessed
March 14, 2005.

[48] Iannaccone, G., Diot, C., Boutremans, C.: Impact of link failures on VoIP perfor- mance.
EPFL-DI-ICA, IC/2002/015 (2002).

[49] Altman, E., Avrachenkov, K., Barakat, C.: TCP in presence of bursty losses. Mea- surement and Modeling of Computer Systems (2000), 124–133.

[50] Ke Wei, M. Muthuprasanna, Suraj Kothari, "Preventing SQL Injection Attacks in Stored Procedures" Proceedings of the 2006 Australian Software Engineering Conference (ASWEC'06 IEEE).

[51] P. Bisht, P. Madhusudan, and V. N. Venkatakrishnan. CANDID: Dynamic Candidate Evaluations for Automatic Prevention of SQL Injection Attacks. ACM Trans. Inf. Syst. Secur., 13(2): 1-39, 2010.

[52] Mei Junjin, "An Approach for SQL Injection Vulnerability Detection," Proc. of TTNG '09, pp.1411-1414, 27-29 April 2009.

[53] YongJoon Park, JaeChul Park, "Web Application Intrusion Detection System for Input Validation Attack "Third 2008 International Conference on Convergence And Hybrid Information Technology.

[54] Needleman, S.B., Wunsch, C.D. "A general method applicable to the search for similarities in the amino acid sequence of two proteins", .T.MoI.BioI.48:443-453, 1970.

[55] Sangita, R., Avinash, K. S., Ashok S. S.: Detecting and Defeating SQL Injection Attacks International Journal of Information and Electronics Engineering, 2011.

[56] Nausheen, K.: Detection and Prevention of SQL Injection Attacks by Request Receiver, Analyzer and Test Model. 2011.

[57] Cristian, N., et al.: CBRid4SQL: A CBR Intrusion Detector for SQL Injection Attacks. 2010.

[58] Shikhar Jain & Alwyn R. Pais," Model Based Approach to Prevent SQL Injection Attacks on.NET Applications" International Journal of Computer Science & Informatics, Volume-1, Issue-11, 2011.

[59] RA. McClure, and J.H. Kruger, "SQL DOM: compile time checking of dynamic SQL statements," Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on, pp. 88-96, 15-21 May 2005.