

PREDICTIVE ANALYSIS OF SQL INJECTION ATTACK DETECTION

Submitted in partial fulfillment of the
requirements for the award of
Bachelor of Engineering degree

in

Computer Science and Engineering by

K.Pravalika (37110318)

I.Jyotsna (37110278)



**DEPARTMENT OF COMPUTER SCIENCE
AND ENGINEERING SCHOOL OF
COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by
NAAC JEPPIAAR NAGAR, RAJIV
GANDHI SALAI, CHENNAI - 600**

MARCH – 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **K.Pravalika** (37110318), **I.Jyotsna** (37110278) and who carried out the project entitled "**PREDICTIVE ANALYSIS OF SQL INJECTION ATTACK DETECTION**" under my supervision from November 2020 to April 2021.

Internal Guide

Dr.A.SivaSangari.,ME.,Ph.D

Head of the Department

Dr. S. Vigneshwari, M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **Kannekanti Pravalika** (Reg.No-37110318) here by declare that the Project Report on “**PREDICTIVE ANALYSIS OF SQL INJECTION ATTACK DETECTION**” done by me under the guidance of **Dr.A.SivaSangari.,M.E., Ph.D** at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE :06/04/2021

k.pravalika

PLACE:Chennai

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala, M.E., Ph.D., Dean, School of Computing** and **Dr. S.Vigneshwari, M.E., Ph.D. and Dr. L. Lakshmanan M.E., Ph.D., Heads** of the Department, Dept. of Computer Science and Engineering for providing necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.A.SivaSangari.,M.E.,Ph.D** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of **Computer Science and Engineering** , who were helpful in many ways for the completion of the Project.

ABSTRACT

Injection attack is the first of the top 10 security threats announced by the OWASP. Meanwhile, SQL injection is one of the most important types among the injection attacks. Because of its various types and fast variations, SQL injection can cause great harm to the network, resulting in data leakage and website paralysis. Due to the heterogeneity of attack load, the diversity of attack methods and the variety of attack modes, SQL injection detection is still a challenging problem. How to defense SQL injection attack effectively becomes the focus and frontier of web security nowadays.

Therefore, this paper proposes an adaptive deep forest-based method to detect the complex SQL injection attacks. Firstly, the structure of deep forest is optimized in our paper, the input of each layer is concatenated by the raw feature vector and average of previous outputs. Experiments show that our proposed method effectively solves the problem that the original features of deep forests are degraded with the increasing number of layers.

Then, we introduce an AdaBoost algorithm based deep forest model which utilize error rate to update the weights of features on each layer. That is, in the process of training, different features are assigned with different weights based on their influence on the result. Our model can automatically adjust the structure of the tree model and deal with multi-dimensional fine-grained features to avoid over-fitting problem effectively. The experimental results show that the proposed method has a better performance than classical machine learning methods and deep learning methods.

In Internet age, the most critical security risk of vulnerable web applications is SQL injection attack. SQL is a communication medium between Web application and back end database. So the mostly attackers use SQL for accessing a database. A SQL Injection attack (SQLIA) is that in which a malicious mind person injects its own crafted query as input. The back end server executes the query statement and sends the result to the attackers. An attacker obtains the result and back end database. So the mostly attackers use SQL for accessing the database.

CONTENTS

TITLE	Page No.
ABSTRACT	I
LIST OF FIGURES	IV
1: INTRODUCTION	05
1:1 OVERVIEW OF THE PROJECT	05
1:2 OBJECTIVE OF THE PROJECT	06
1:3 SCOPE OF THE PROJECT	06
1:4 EXISTING SYSTEM	07
1:5 PROPOSED SYSTEM	08
2: LITERATURE SURVEY	09
2:1 LITERATURE-1	09
2:2 LITERATURE-2	10
2:3 LITERATURE-3	11
2:4 LITERATURE-4	12
2:5 LITERATURE-5	13
3: METHODOLOGY	14
SCHEMATIC PROCESS	14
RELATED WORK	16
3:2 FRAMEWORK OVERVIEW	18
3:3 ADABOOST DEEP FOREST MODEL	20
3:3 INCORPORATING DEEP FOREST MODEL	22
3:4 EXPERIMENTAL EVALUATION	25
3:5 SYSTEM MODULES	32
4: RESULTS AND DISCUSSION	34
5: CONCLUSION AND FUTURE WORK	35
5:1 CONCLUSION	35

5:2 FUTUREWORK	35
REFERENCES	37
APPENDIX	17
a. SOURCE CODE	40
b. SCREEN SHOTS	41
c. PUBLICATIONS	45
d. PLAGARISM REPORT	

LIST OF FIGURES

FIGURE NUMBER	NAME OF THE FIGURE	PAGE NUMBER
1.4.1	EXISTING SYSTEM ARCHITECTURE	8
1.4.2	DISADVANTAGES	8
1.5.2	PROPOSED SYSTEM ARCHITECTURE	9
3.1.1	SCHEMATIC DIAGRAM OF SQL INJECTION	14
3.1.2	DATA COLLECTION	16
3.2.1	THE STRUCTURE OF CASCADE FOREST	18
3.3.1	INCORPORATING ADABOOST ALGORITHM	20
3.4.1	OVERALL PROCEDURE OF ADP	22
	ACCURACY COMPARED TO ML ALGORITHM	27
	RECALL COMPARED TO ML ALGORITHM	28
	RECALL COMPARED TO DNN STRUCTURES	29
	F1 SCORE COMPARED TO DNN	31
	OPEN ANACONDA PROMPTH	33
	GATHERING OF DATA SETS	34
	USING GET AND POST METHODS	41

CHAPTER –

1

INTRODUCTION

OVERVIEW OF PROJECT

With the rapid development of Web 2.0 technology, network applications have gradually become an indispensable part of our lives. At the same time, Web applications are confronted with more challenges. As announced by the OWASP organization, injection attack has been the first of the top 10 security vulnerabilities in 2013 and 2017, and SQL injection attack is one of the most important types

SQL (Structured Query Language) is a standard database programming language for accessing and manipulating data in database WEB programming languages (such as PHP, JAVA) provide various methods for constructing and executing SQL statements. Developers usually construct SQL statements by concatenating string which is submitted by users from web page. Due to the wide variety of SQL languages, there are too many encoding methods for constructing SQL statements, so there is a risk of being attacked anywhere via constructing SQL statements. In the process of SQL injection attack, the attacker inserts malicious code fragments into the request parameters, which causes the server to execute illegal queries, resulting in data leakage and database damage. For example, the attacker can obtain the username, password and other privacy data of website users by SQL injection, which seriously threatens the data security. As shown in , the attacker sends malicious SQL statements into the application through methods like user input injection, cookie injection, server variable injection and so on. Since SQL attacks only need to construct structured query statements without developing additional malwares, SQL injection has become one of the most frequently used attacks by attackers, and the attacking methods change frequently with the development of website technology. The structure of SQL language is changeable and supports different coding methods. Therefore, some traditional detection methods, such as blacklist mechanism and rule-based detection mechanism,

cannot produce better defensive effect.

In recent years, many researchers have done a lot of work on SQL injection detection, but the detection scope is usually limited to some subsets of SQL

injection.

Objective of the project:

Attackers can use sql injection vulnerabilities to bypass application security measures. They can go around authentication and authorization of a web page or web application and retrieve the content of the entire sql database. They also use sql injection to add, delete, modify records in databases.

Scope of the project:

This method uses the SQL lexical and grammatical analysis technology to parse the SQL statement into the SQL grammar tree via stain analysis, and judges whether there is an SQL injection attack according to the stain state. While, dynamic analysis detection can only detect pre-defined vulnerabilities by application developers, and it has no detection capability.

when encountering attacks against unknown vulnerabilities. Parameter filtering is based on regular expressions and blacklists to filter invalid characters. This method needs predefined regular expressions, but due to the diversity of SQL grammar and user input, it cannot meet the detection requirements. Attackers can adjust the input to bypass detection. Appiah et al. proposed an improved pattern matching method for signature-based SQL injection attack detection framework, which integrates fingerprint identification and pattern matching to distinguish normal and malicious SQL queries.

Existing System

In the existing application, the malicious attacks are controlled by certain decisions and programming coding. The decision is based on certain set of rules. These set of rules are designed to detect attack patterns. Other Approaches

- White-box Testing
- Black-box Testing
- Static Analysis
- Model Based Testing

Existing System Architecture:

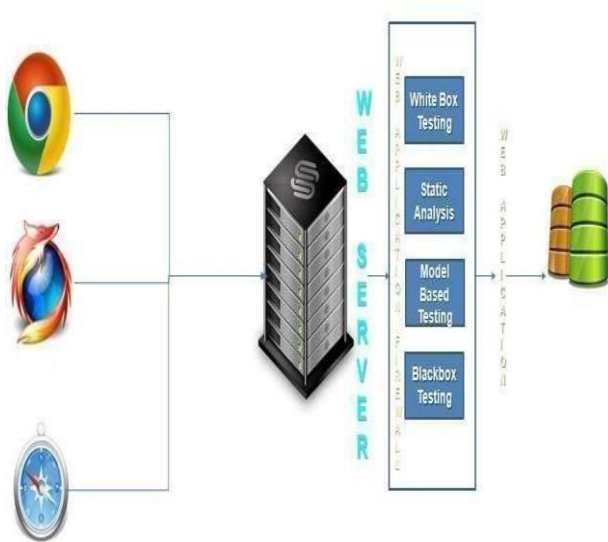


Fig.1.4.1 Existing System Architectur

Disadvantages:

For huge applications, the set of rules become more complex.It is more difficult to maintain the set of rules manually and test

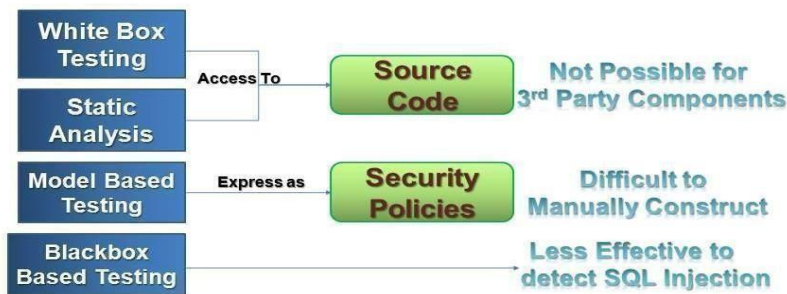


Fig.1.4.2 Disadvantages

C

Literature 1

H
A
P
T
E
R
2
L
I
T
E
R
A
T
U
R
E
S
U
R
V
E
Y

TITLE: Deep Learning Approach for Intelligent
Intrusion Detection System

AUTHOR: VINAYAKUMAR R, MAMOUN ALAZAB,

YEAR: 2018

DESCRIPTION: Machine learning techniques are being widely used to develop an intrusion detection system (IDS) for detecting and classifying cyber-attacks at the network-level and host-level in a timely and automatic manner. However, many challenges arise since malicious attacks are continually changing and are occurring in very large volumes requiring a scalable solution. There are different malware datasets available publicly for further research by cyber security community. However, no existing study has shown the detailed analysis of the performance of various machine learning algorithms on various publicly available datasets. Due to the dynamic nature of malware with continuously changing attacking methods, the malware datasets available publicly are to be updated systematically and benchmarked. In this paper, deep neural network (DNN), a type of deep learning model is explored to develop a flexible and effective IDS to detect and classify unforeseen and unpredictable cyber- attacks. The continuous change in network behaviour and rapid evolution of attacks makes it necessary to evaluate various datasets which are generated over the years through static and dynamic approaches. This type of study facilitates to identify the best algorithm which can effectively work in detecting future cyber- attacks.

Literature 2

TITLE: Defending Against Web Application Attacks: Approaches, Challenges and Implications

AUTHOR: Dimitris Mitropoulos, Panos Louridas, Michalis Polychronakis

YEAR: 2017.

DESCRIPTION: Some of the most dangerous web attacks, such as Cross-Site Scripting and SQL injection, exploit vulnerabilities in web applications that may accept and process data of uncertain origin without proper validation or filtering, allowing the injection and execution of dynamic or domain-specific language code. These attacks have been constantly topping the lists of various security bulletin providers despite the numerous countermeasures that have been proposed over the past 15 years. In this paper, we provide an analysis on various defense mechanisms against web code injection attacks. We propose a model that highlights the key weaknesses enabling these attacks, and that provides a common perspective for studying the available defenses. We then categorize and analyze a set of 41 previously proposed defenses based on their accuracy, performance, deployment, security, and availability characteristics. Detection accuracy is of particular importance we discuss the results of our analysis, with emphasis on factors that may hinder the widespread adoption. Detection accuracy is of particular importance, as our findings show that many defense mechanisms have been tested in a poor manner. In addition, we observe that some mechanisms can be bypassed by attackers.

Literature 3

TITLE: Detecting SQL Vulnerability Attack based on the Dynamic and Static Analysis Technology

AUTHOR: Yaohui Wang, Dan Wang, Wenbing Zhao, Yuan Liu

YEAR: 2015.

Description: Targeting at PHP program, this paper proposes an SQL vulnerability detection method based on the injection analysis technology. This method makes a detailed analysis on the one-time injection in the aspects of data flow and program behavior, on the basis of the combination of dynamic and static analysis technique. Then it implements the SQL vulnerability determination algorithm which is based on lexical feature comparison. At last, this paper combines alias analysis technology, behavior model and SQL which is based on lexical feature comparison to design and establish a prototype system for SQL vulnerability detection. The experiment shows that our system has a good strong ability of SQL vulnerability detection and very low time cost.

Literature 4

TITLE: Parental Arrest and Adolescent Delinquency in Singapore: The Moderating Roles of Narcissism, Callous-Unemotional Traits, and Impulsivity
AUTHOR: Xiang Li Rebeccap
YEAR: 2019.

Description: Adolescent delinquency is an important issue, and of universal concern. Parental arrest history has been recognized to have negative impacts on children's delinquent behavior. This study focused on the predictors of adolescent delinquency, including parental influences and adolescents' own personal traits. Methods We explored the relationship between parental arrest and children's delinquency among 1027 Singaporean secondary school students aged between 12 and 19 years ($M = 14.10$). Specifically, we used hierarchical multiple regression to examine the moderating effects that different dimensions of psychopathic traits, including narcissism, callous-unemotional (CU) traits, and impulsivity, have on the relation between parental arrest and adolescent delinquency.

Literature 5

TITLE: Using Deep Convolutional Neural Network Architectures for Object Classification and Detection within X-ray Baggage Security Imagery
AUTHOR: Samet Akcay, Mikolaj E. Kundegorski
YEAR: 2018.

Description: We consider the use of deep Convolutional Neural Networks (CNN) with transfer learning for the image classification and detection problems posed within the context of X-ray baggage security imagery. The use of the CNN approach requires large amounts of data to facilitate a complex end-to-end feature extraction and classification process. Within the context of X-ray security screening, limited availability of object of interest data examples can thus pose a problem. To overcome this issue, we employ a transfer learning paradigm such that pre-trained CNN, primarily trained

CHAPTER 3

METHODOLOG

Y

3.1.1 Schematic process

To detect the SQL injection attacks. Firstly, we improve the structure of deep forest. The input of each layer is the concatenation of the raw feature vector and the average of previous outputs, which can effectively solve the problem of the continuous degradation of raw features caused by the increasing number of deep forest layers. Then, we introduce the AdaBoost algorithm with adaptive ability into the deep forest model, which can use error rate of each level in the deep forest to update the weights

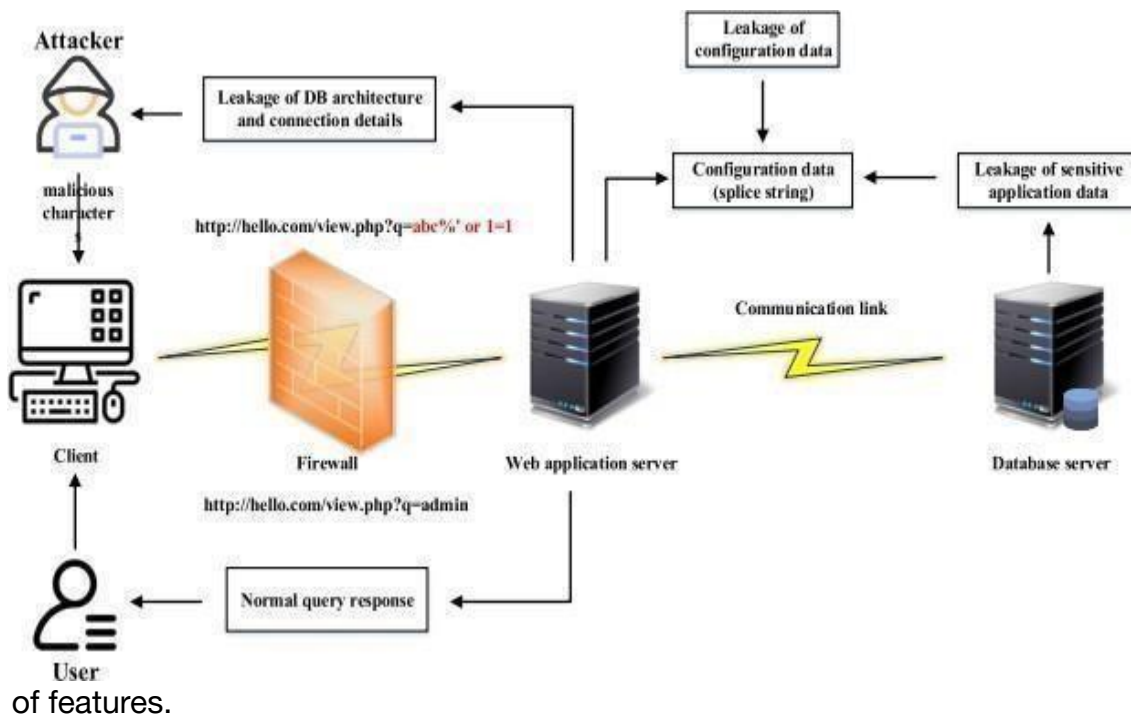


Fig .3.1.1 Schematic Diagram Of SQL injection attack

Related Work:

Traditional SQL injection detection methods can be divided into static analysis, dynamic analysis and parameter filtering . Static analysis method detects the type error and grammatical error by analyzing the input statements . Santhosh Kumar et al. detected

SQL injection through the program interface instead of considering the internal analysis determines whether a sentence is malicious by analyzing its final implementation. Naderi-Afooshteh combined the stain tracking and black box testing techniques into the software development life cycle to defense SQL injection. characteristics . Wassermann and Su extend white-box testing method to detect tautological attacks . This method uses the SQL lexical and grammatical analysis technology to parse the SQL statement into the SQL grammar tree via stain analysis, and judges whether there is an SQL injection attack according to the stain state. While, dynamic analysis detection can only detect pre-defined vulnerabilities by application developers, and it has no detection capability when encountering attacks against unknown vulnerabilities. Parameter filtering is based on regular expressions and blacklists to filter invalid characters .

This method needs predefined regular expressions, but due to the diversity of SQL grammar and user input, it cannot meet the detection requirements. Attackers can adjust the input to bypass detection. Appiah et al. proposed an improved pattern matching method for signature- based SQL injection attack detection framework , which integrates fingerprint identification and pattern matching to distinguish normal and malicious SQL queries. Some researchers introduced the artificial intelligence to detect the SQL injection. In recent years, SQL injection detection based on deep learning has been paid more and more attention. Compared with traditional machine learning algorithms, deep learning does not require complex feature engineering, and its performance is usually enhanced with the increase of training data. That is to say, its scalability is significantly better than traditional machine learning algorithms . Fang et.al design transforms the sample data into word vector as the input of the model and uses LSTM model to detect the SQL injection, the authors proved the LSTM-based method has better detection effect than RNN and CNN based methods . But there are still many challenges and problems. First, due to the lack of open SQL injection data sets, it is difficult to obtain sufficient training samples and lead to overfitting in the training process. In our previous works.

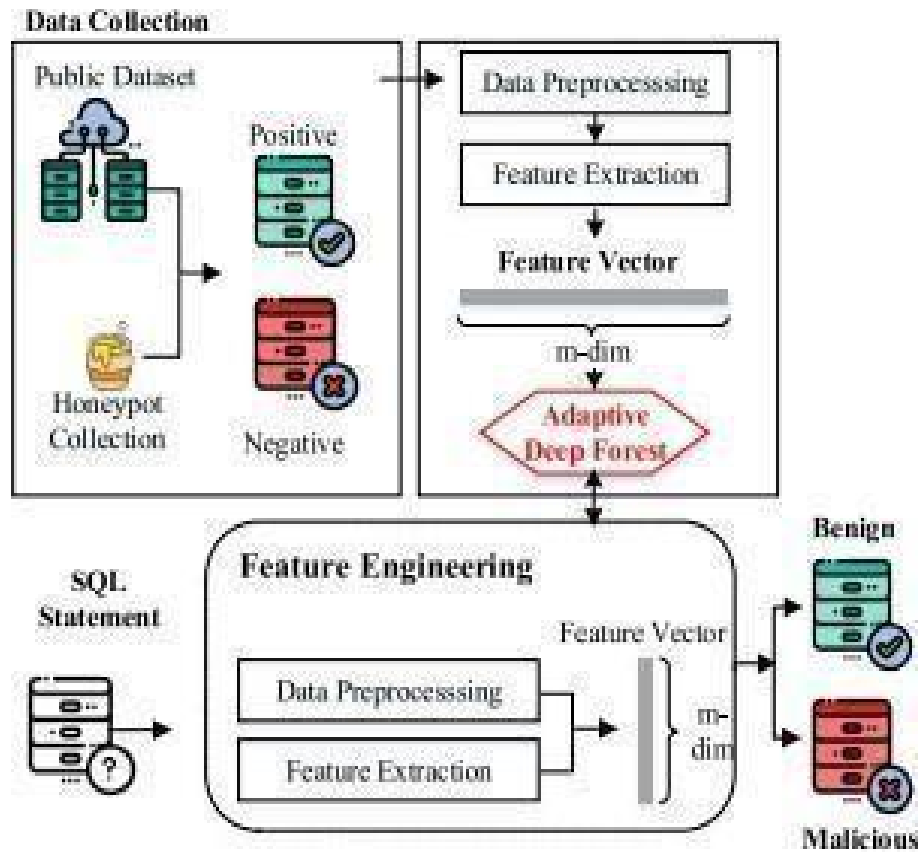


Fig . 3.1.2 Data collection

Framework Overview

As shown in Figure 2, the proposed deep forest-based SQL injection detection method consists of two important stages, off-line training stage and online testing stage. In the off-line training stage, we first collect the training data from honeypot Technology and SQL injection samples on the vulnerability submission platform. The encoded samples are preprocessed by data preprocessing module to decode and then input into feature extraction module. The feature extraction method mainly focuses on the SQL injection statement length, typical SQL injection keywords, uppercase characters, numeric characters, annotations, null characters and special characters. Finally, the feature vectors are used as the input of the deep forest model. In the online testing stage, the unknown type SQL statement should be decoded by data preprocessing module firstly and then input into feature extraction module to generate a same dimensional vector as off-line training stage. Finally, the SQL statement will be classified as malicious or normal by using the trained model. The Adaptive Deep Forest Model (ADF) is the key part of the proposed algorithm. Firstly, compared with the existing machine learning methods, ADF has the advantage that the model depth is sufficiently complex and the features can be processed layer by layer, it can complete the intra-layer transformation, so that the information contained in each dimension of the feature can be fully utilized. The redundant part of the feature vector can be ignored, so as to avoid influences of the redundant features on the model effectively. Secondly, compared with other deep learning methods, the training process of ADF does not depend on the back propagation of the loss. Therefore, it has fewer parameters and higher computational efficiency and can also perform very well on small-scale data. Thirdly, we introduce the AdaBoost algorithm to enhance the self-adjusting ability of the deep forest model, and the weights of features are updated according to the classification error rate of each level of deep forest, which effectively solves the problem of unbalanced distribution of multi-type samples.

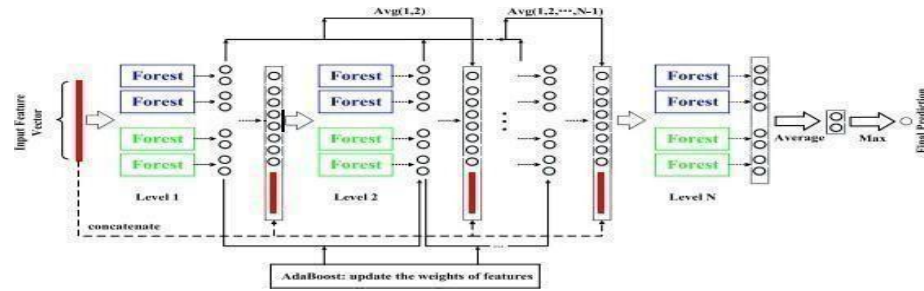


Fig. 3.2.1 The structure of cascade forest

Adaptive deep forest model

In this section, we proposed an Adaptive deep forest model (ADF) which is improved on deep forest. In ADF, feature performance is enhanced by feature transformation according to multi-grained scanning, and then characterization learning is performed layer by layer through using a cascade structure. Compared with deep neural networks, adaptive deep forest has obvious advantages in hyper-parameter settings and performance. Firstly, adaptive deep forest has much less hyperparameters than deep neural networks, so it is more robust to hyper-parameter settings. Secondly, in the default hyperparameter setting, adaptive deep forest can achieve excellent performance when dealing with different data in different fields and the parallel computing has a higher efficiency. Inspired by the principle of characterization learning in deep neural network, each level in cascading structure receives the average value of all previous levels as input, and outputs its results to the next level. Among them, the number of layers in series can be automatically set according to the data, which simplifies the complexity of the model. The cascade forest structure is shown as follows: In order to meet the diversity of structural design, in the cascade structure, the adaptive deep forest consists of two parts: completely random forest and random forest. The main difference between them is the candidate feature space. Completely random forest selects feature randomly in the complete feature space, while a normal random forest randomly selects the feature of $\sqrt{d}$ as a candidate

feature (d is the dimension of the original input feature), and then selects the best Gini coefficient. The feature acts as a split node until each leaf node only contains instances of the same class or no more than 10 instances. The decision tree is responsible for continuously dividing the subspaces in the feature space and labeling each subspace with a class label. That means each tree will generate a category of probability distribution according to the category proportion of the training samples in the subspace for a specific SQL sample, and then average the proportions of all trees in the forest to output the entire forest. As shown in the Fig.4, it is a simplified forest of the two-category problem. Each sample will find a path in each tree to find its corresponding leaf node, and the type of training data in this leaf node is likely to be different. For categories, we can statistically obtain proportions of different categories, and then generate a probability distribution. The process of multi-grained scanning is shown as follows: First, a P -dimensional raw input feature is scanned by a K -dimensional sliding window and S -dimensional transformed feature vectors are generated, where $S = (P - K) / 1 + 1$. Then each of transformed feature vector is used in the training of cascade forest and transformed to a C -dimensional vector of probability in each forest. That is, each forest will generate $S * C$ representation features. Eventually, the output result of forests within each level will be concatenated together. Fig.5 shows the size of one sliding window. In the experiments, our proposed method uses sliding windows with multiple sizes, so as to contain more features in the final transform feature vector.

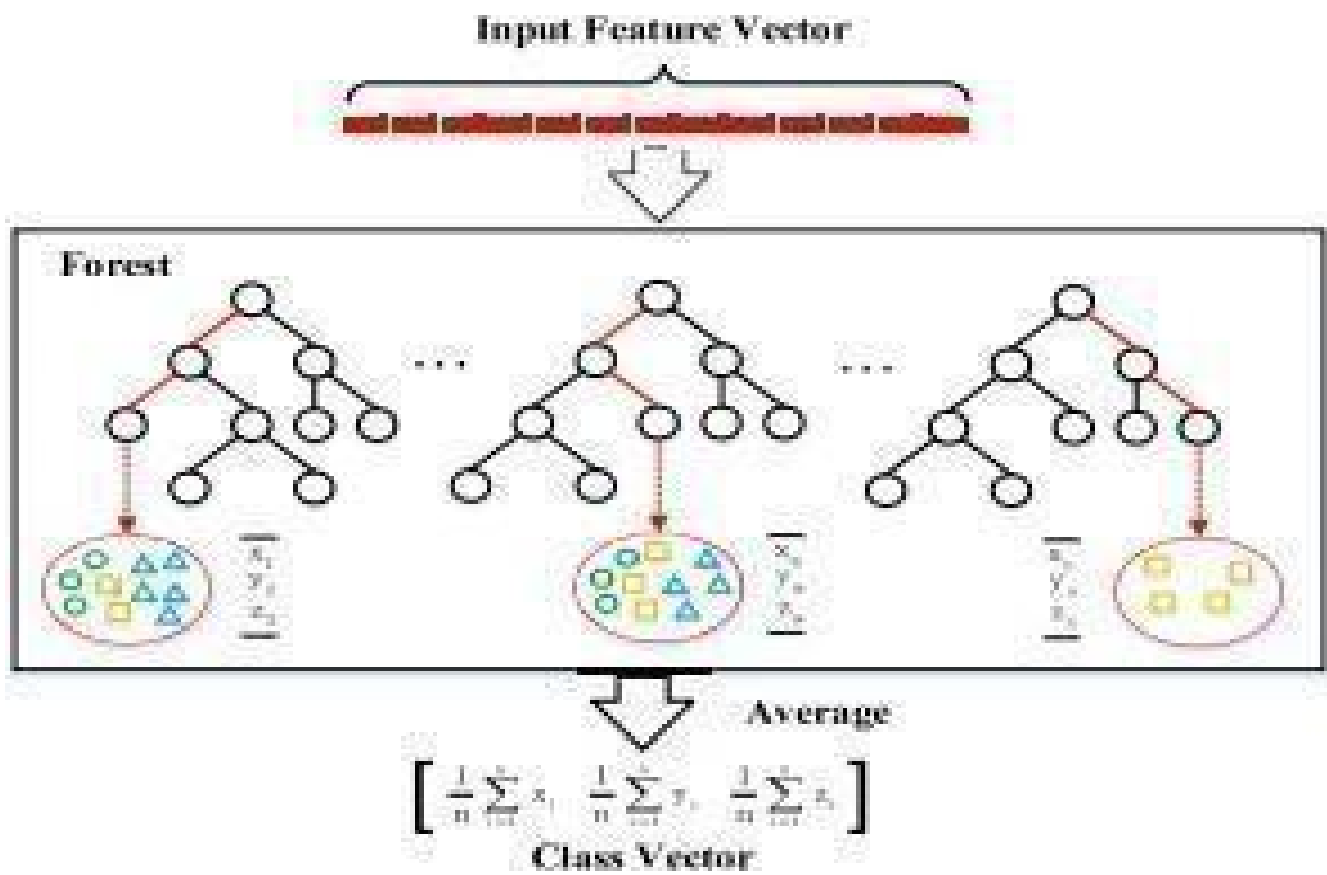


Fig.3.3.1 incorporating with adaboost algorithm

3.3 Incorporating with Adaboost algorithm

Since the contribution of different features to the classification results is not considered in the deep forest model, this paper proposes a method to integrate the AdaBoost algorithm into the deep forest model. In the process of training random forests and completely random forests, the model assigns initial weights to the features and reduces the weights of the features that have a great influence on the classification results. The model pays more attention on the features which have less influence in the following training process. The procedure of the algorithm is as follows:

Step 1: Set the layer of cascade trees: m

Step 2: Initialize the weights of raw features: $w_{i,j} = 1/N$ where N is the size of raw featurespace, i is the layer of cascade tree, j is the index of the feature.

Step 3: From 1 to m , train DF: Select features from feature space b) Calculate the error rate at layer i : $e_i = \sum_{j=1}^N w_{i,j} \times I(y_i \neq b_{yi})$ c) If $e_i > e_{average}$: break

Step 4: If $e_i \leq e_{average}$, calculate the weight coefficient of the classifier: $\alpha_i = \frac{1}{2} \log \frac{1}{1 - e_i}$

Step 5: Update weights of raw features: $w_{i+1,j} = w_{i,j} \exp(-\alpha_i y_i G_i(x_i))$
 $P_m = \sum_{i=1}^m w_{i,j} \exp(-\alpha_i y_i G_i(x_i))$

Step 6: End

Step 7: Output ADF

model

The algorithm introduces the AdaBoost algorithm in the deep forest model, initializes the weight before the model training, and then updates the feature weight through the classification error rate of each level of the deep forest, and finally outputs the model.

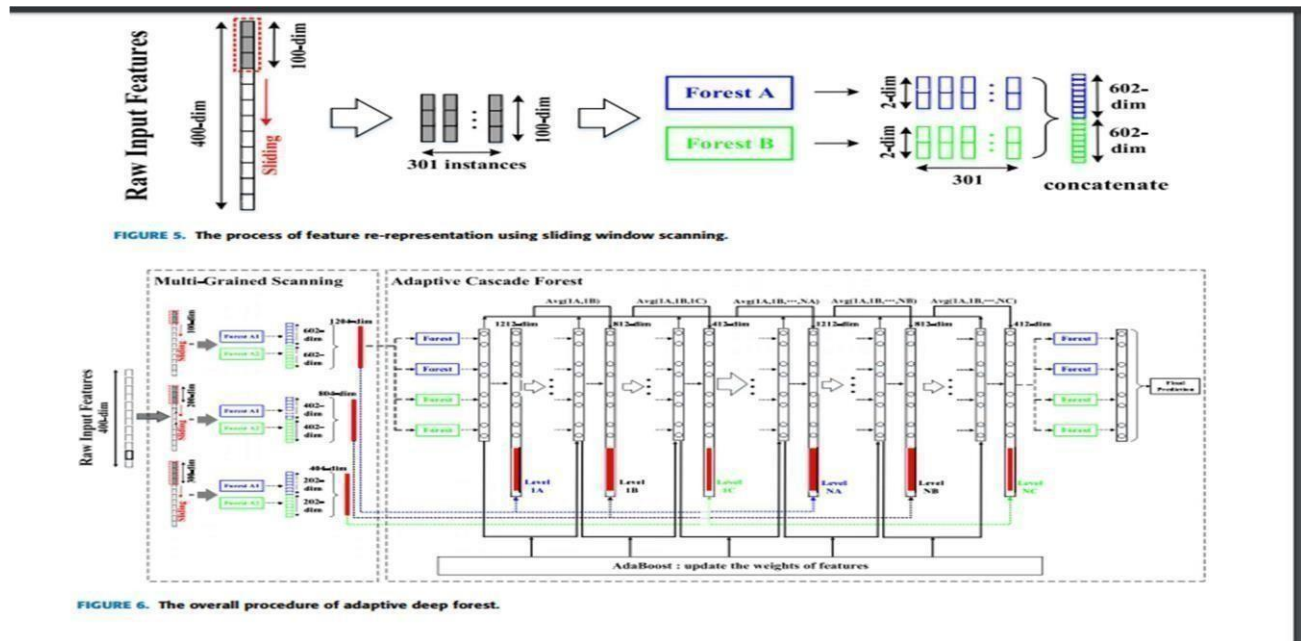


Fig.3.4.1 The overall procedure of adaptive deep forest

Experimental Evaluation:

The positive samples used in the experiments includes two parts: one part of data is obtained from the vulnerability submission platform exploit-db and WooYun vulnerability submission platform. The other part is acquired by runningSQLmap and its tamper scripts on certain websites within one month. The total number of SQL injection samples we collected is 10000. The dataset totally includes 10,000 negative samples and 10,000 positive samples. Deep forest has less parameters than DNN, including parameters of multi-grained scanning and cascade treestructure. The specific parameter types and settings are shown in Table 2. All experiments carried out in PC with an Intel Core i5-8400 CPU (2.8 GHz), GPU is TITAN(X) (Pascal). We use python 3.5.4 and Scikit-learn 0.20.0 library to build machine learning model, Keras 2.1.2 to build neural network, anduse Tensorflow 1.4.1 as a backend

computing framework. In the experiments, the evaluation criteria includes accuracy(ACC), precision(P), recall (R) and F1-score. According to the Ref. [6], SQL injection can be divided into the following main types, such as error-based injection, UNION query, blind injection and so on. Researchers have extracted relevant detection features for each type of attack. In order to make a comprehensive comparison between the proposed methods and the state of art, we firstconduct

ID	Attribute	Description	Dimension	Bool/ Num
1	N_LEN	The total length of SQL statement	1	N
2	N_NUM	Number of numeric characters in the statement	1	N
3	N_BL	Number of uppercase characters in the statement	1	N
4	N_SP	Number of space in the statement	1	N
5	N_NL	Number of NULL in the statement	1	N
6	N_PF	Number of prefix, such as "\\x", "\\u"	2	N
7	N_AO	Number of arithmetic operator, such as "+", "-", "*", "/"	4	N
8	B_AN	Whether there are annotation including single line "--" and multiple lines "/* */"	2	B
9	B_QR	Whether there are "query" keywords such as "select"	1	B
10	B_UN	Whether there are "union" keywords such as "Union" and "union all"	1	B
11	B_DB	Whether there are "database" keywords such as, "sysobjects", "msysobjects", "version", "information schema"	1	B
12	N_CN	Number of "connection" keywords, such as "inner join", "and", "or", "xor"	4	N
13	B_FL	Whether there are "file" keywords, such as "load_file", "load data infile", "into outfile", "into outfile"	1	B
14	B_OS	Whether there are "operating system" keywords, such as "exec"	3	B
15	N_ST	Number of "string", such as "substr", "substring", "mid", "asc"	1	N
16	N_CO	Number of comparison operator, such as "=", "<", ">", ">=", "<="	5	N
17	N_BK	Number of brackets such as "(", ")", "[", "]", "{", "}"	3	N
18	N_QM	Number of quotation marks including quotation marks	2	N
19	N_SC	Number of special characters, such as "!", "@", "#", "&", "%"	1	N
20	B_EX	Whether there are "exists" keyword	1	B
21	B_FL	Whether there are "floor()" function	1	B
22	B_RD	Whether there are "rand()" function	1	B
23	B_GAO	Whether there are "group" and "order" keywords	2	B
24	B_LEN	Whether there are "length()" function	1	B
25	B_ASC	Whether there are "Ascii()" function	1	B
26	B_CC	Whether there are "concat()" function	1	B
27	B_IF	Whether there are "if" keyword	1	B
28	B_CT	Whether there are "count()" function	1	B
29	B_SL	Whether there are "sleep()" function	1	B
30	B_ID	Whether there are "identity", such as "I=I", "a=a"	1	B

be seen from Fig.7 to Fig.10, a comparative experiment for each type of SQL injection attack. The datasets and detection features corresponding to each attack type are shown in Table 3. a) Comparison with classical machine learning As shown in Fig. 7 to 10, the vertical axis represents the mean value of detection indexes, and the horizontal axis represents the combination mode of attack types. That is, ACx represents the combination mode of anyx attack types. Take the F1-score value for example, $F1(AC2) = \text{Avg}(F1(DS1 + DS2), F1(DS1 + DS3), F1(DS1 + DS4), F1(DS2 + DS3), F1(DS2 + DS4), F1(DS3, DS4))$ It can only when the network environment is relatively simple, that is, there are only a few types of attacks, the traditional machine learning methods can achieve satisfying detection results. However, in practical applications, the attack scenario is very complex, and multiple attack types often occur simultaneously. The detector needs to extract multiple features to complete the detection of all kinds of attacks. While the redundancy and mutual exclusion between different features will directly affect the detection results. It is impossible to extract detection features independently for specific attack types, and serial detection methods will cause a lot of waste. The structure of ADF is composed of a random forest and a completely random forest. So it's possible to select from a wider range of feature

TABLE 2. Parameter setting of deep forest.

Parameter	Description	Value
n_mgsRFtree	number of trees in a random forest during multi-grained scanning	30
window	window sizes to use during multi-grained scanning	[10,20,30]
stride	Step used when slicing the data	1
ca_test_size	split fraction or absolute number for cascade training set splitting	0.2
n_caRF	number of random tree in a cascade layer	2
n_caRFtree	number of trees in a single random forest in a cascade layer	101
min_sam_mgs	minimum number of samples in a node to perform a split during the training of multi-grained scanning random forest	0.1
min_sam_ca	minimum number of cascade layers allowed	0.1
Ca_layer	maximum number of cascade layers allowed	np.inf
tolerance	accuracy tolerance for the cascade growth	0

TABLE 3. The features extracted in different datasets.

DS	Type	Features
DS1	Error-based injection	1,2,3,4,5,7,8,17,18,19
DS2	UNION query	1,2,3,4,5,7,8,9,10,12,18,19
DS3	Blind Injection	1,2,3,4,5,7,8,12,16,18,19,29,30
DS4	Execute SQL commands	1,2,3,4,5,7,8,12,13,14,17,18,19

System Modules

spaces, which can more effectively prevent the over-fitting problem. That's the reason why our proposed method has a better performance in complex attack environment.

b) Comparison with deep neural network In the above experiments, our proposed method has been proved that has a better performance than classical machine learning methods. As we all know, DNN and DF are better than the other deep learning models such as CNN and RNN when dealing with multi-feature classification. In the following experiments, we should compare our proposed method with deep neural network. For comparison experiments, we adopt three different structures of DNN, respectively with 5 layers, 10 layers and 15 layers. The dataset and feature vector we used contained all kinds of SQL injection in Table.3 and all feature of samples. The detail information of datasets is shown in Table.4. It can be seen from Fig.11 to Fig.14, when the number of samples is less than 16000, the detection accuracy of ADF is higher than that of DNN. When the number of samples is more than 16000, the accuracy of DNN is higher than

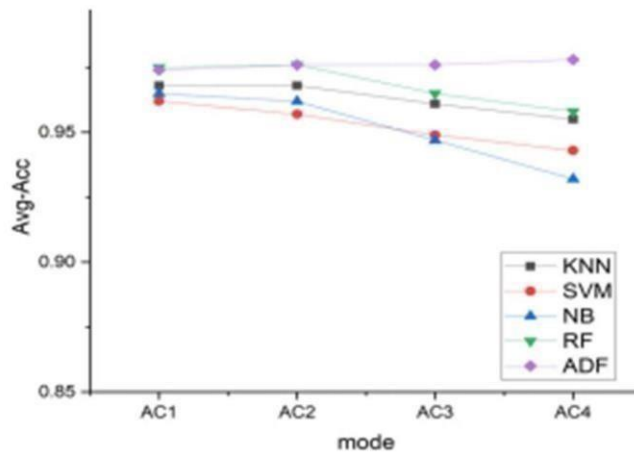


FIGURE 7. Accuracy compared with machine learning algorithms.

that of ADF. This shows that ADF still has a good performance on limited samples, but DNN is not effective on the insufficient

Fig3.6.1 Accuracy compared with machine learning algorithms

FIGURE 7. Accuracy compared with machine learning algorithms.

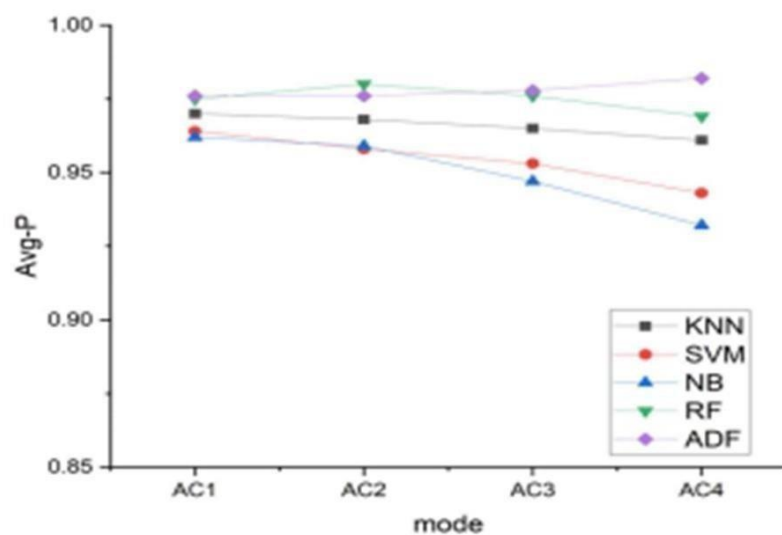


FIGURE 8. Precision compared with machine learning algorithms.

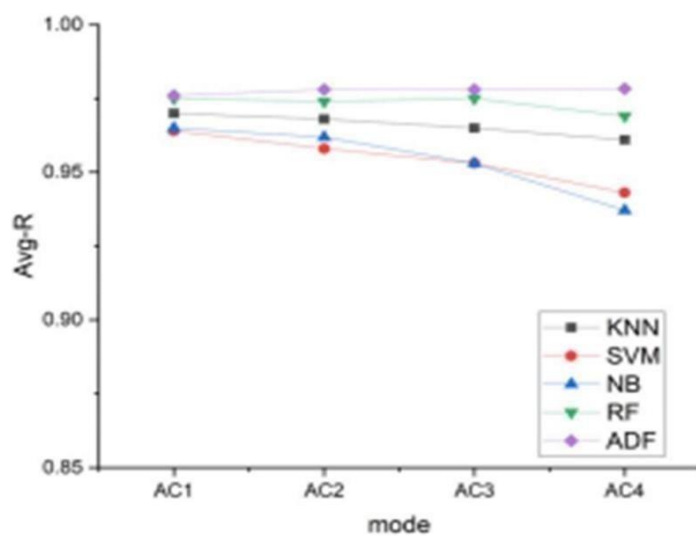


FIGURE 9. Recall compared with machine learning algorithms.

Fig3.6.2 Recall compared with machine learning algorithms

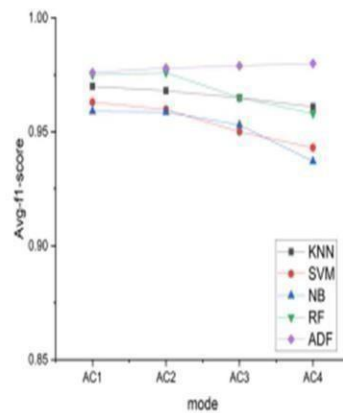


FIGURE 10. F1-score compared with machine learning algorithms.

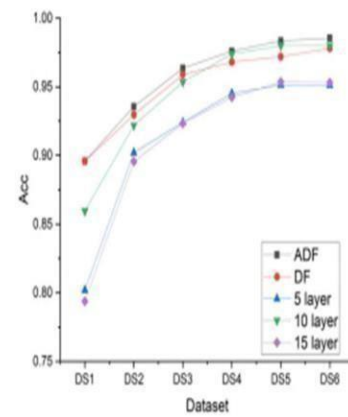


FIGURE 11. Accuracy compared with different DNN structures.

TABLE 4. The number of samples in datasets.

Dataset	Negative	positive
DS1	800	800
DS2	3,000	3,000
DS3	5,000	5,000
DS4	8,000	8,000
DS5	10,000	10,000
DS6	15,000	15,000

TABLE 5. Comparison of time consuming.

Dataset	ADF	DNN-5 layer	DNN-10 layer	DNN-15 layer
DS1	2.4s	13.0s	24.7s	-
DS2	11.5s	39.9s	78.47s	-

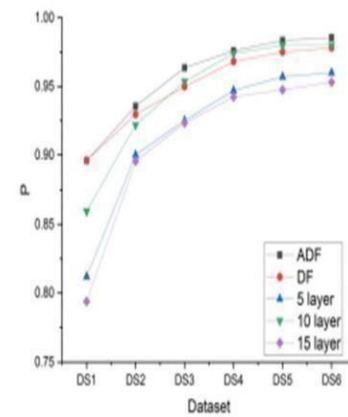


FIGURE 12. Precision compared with different DNN structures.

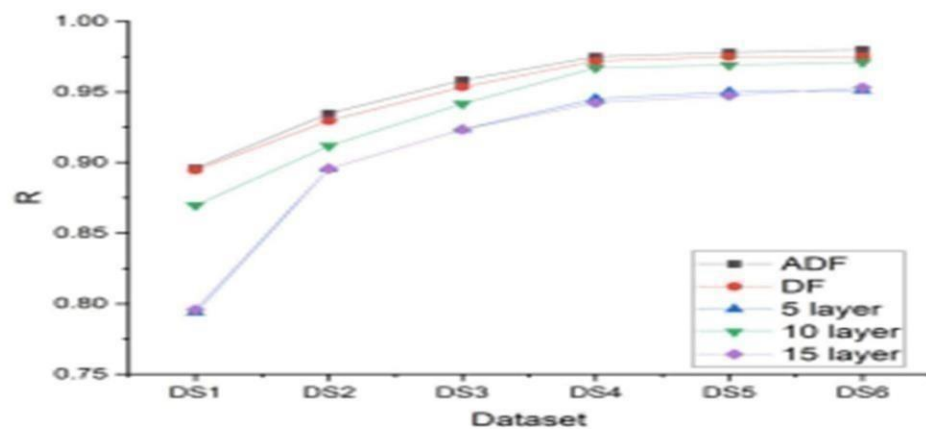


FIGURE 13. Recall compared with different DNN structures.

Fig .3.6.3 Recall compared with different DNN structures

As can be seen, our model has a higher accuracy than DF, because our model considers the output of each cascaded tree. Compared with DNN, we can see that DNN can't achieve satisfying results when the sample size is insufficient, and it is easy to cause overfitting. With the increasing of samples, the accuracy of DNN increases, but the recall rate is lower than other detection methods. Because ADF considers the characteristics that have a great influence on the classification results. After reaching a satisfying accuracy, our model pays more attention to the features that have less influence on the classification results, so as to make full use of the characteristics of each dimension, thus achieving a higher accuracy. But DNN ignores such features, so the recall rate will be lower than ADF. In addition, ADF can be seen that the DNN model needs to determine multiple hyperparameters. The setting of hyper-parameters and the selection of neural network structure have great influence on the detection results. In practical implementation, it is difficult to complete and optimize the hyper-parameter setting according to prior knowledge. However, the deep forest has less hyperparameters and it can adjust the hyper-parameters automatically during the training process. In summary, compared with the deep neural network model, the proposed method has the following advantages: 1) better detection accuracy on less samples; 2) low computational cost; 3) High flexibility and high robustness

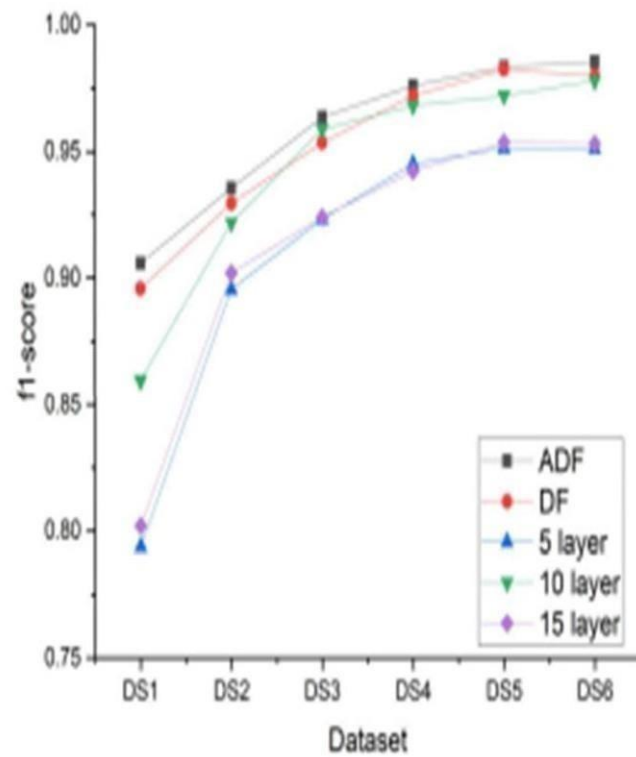


FIGURE 14. f1-score compared with different DNN structures.

Fig.3.6.4 f1-score compared with different DNN structures

Hardware Requirements:

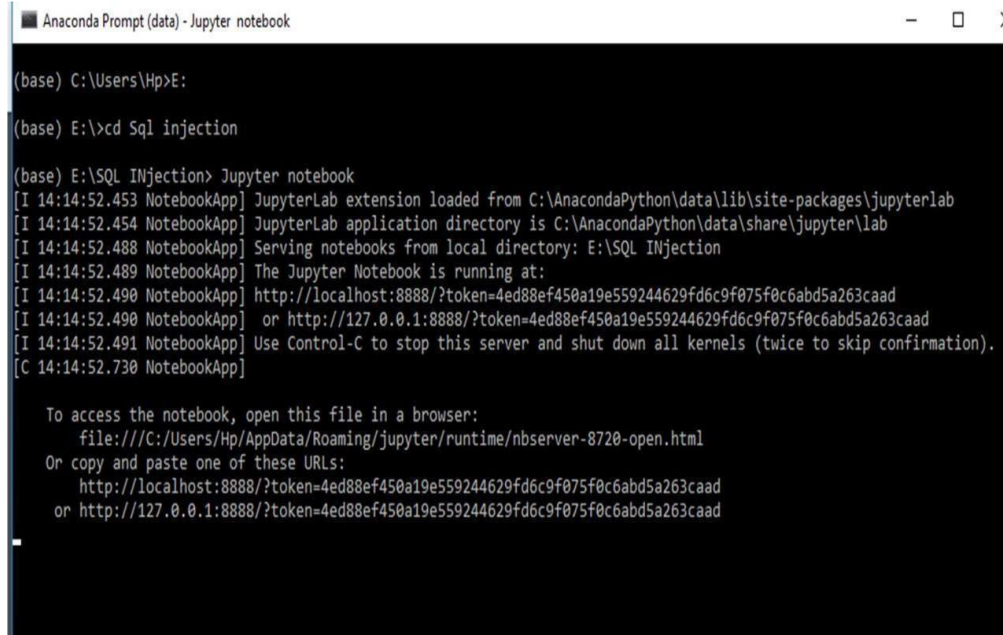
- Two Core Intel Xeon Processors
- 1 TB Hard Disk SAS/ SATA
- 32 GB DDR4 RAM

Software Requirements:

- Microsoft Windows Server 2016
- Python
- SQL Server 2016

CHAPTER 4

RESULT AND DISCUSSION



```
Anaconda Prompt (data) - Jupyter notebook

(base) C:\Users\Hp>E:

(base) E:\>cd Sql injection

(base) E:\SQL Injection> Jupyter notebook
[I 14:14:52.453 NotebookApp] JupyterLab extension loaded from C:\AnacondaPython\data\lib\site-packages\jupyterlab
[I 14:14:52.454 NotebookApp] JupyterLab application directory is C:\AnacondaPython\data\share\jupyter\lab
[I 14:14:52.488 NotebookApp] Serving notebooks from local directory: E:\SQL Injection
[I 14:14:52.489 NotebookApp] The Jupyter Notebook is running at:
[I 14:14:52.490 NotebookApp] http://localhost:8888/?token=4ed88ef450a19e559244629fd6c9f075f0c6abd5a263caad
[I 14:14:52.490 NotebookApp] or http://127.0.0.1:8888/?token=4ed88ef450a19e559244629fd6c9f075f0c6abd5a263caad
[I 14:14:52.491 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 14:14:52.730 NotebookApp]

To access the notebook, open this file in a browser:
file:///C:/Users/Hp/AppData/Roaming/jupyter/runtime/nbserver-8720-open.html
Or copy and paste one of these URLs:
http://localhost:8888/?token=4ed88ef450a19e559244629fd6c9f075f0c6abd5a263caad
or http://127.0.0.1:8888/?token=4ed88ef450a19e559244629fd6c9f075f0c6abd5a263caad
```

Fig .4.1.1: open anaconda prompt

In fig 15 open anaconda prompt and run commands first of all press which drive it is present followed with semi colon after that type sql injection in the console after that enter Jupyter notebook and press enter

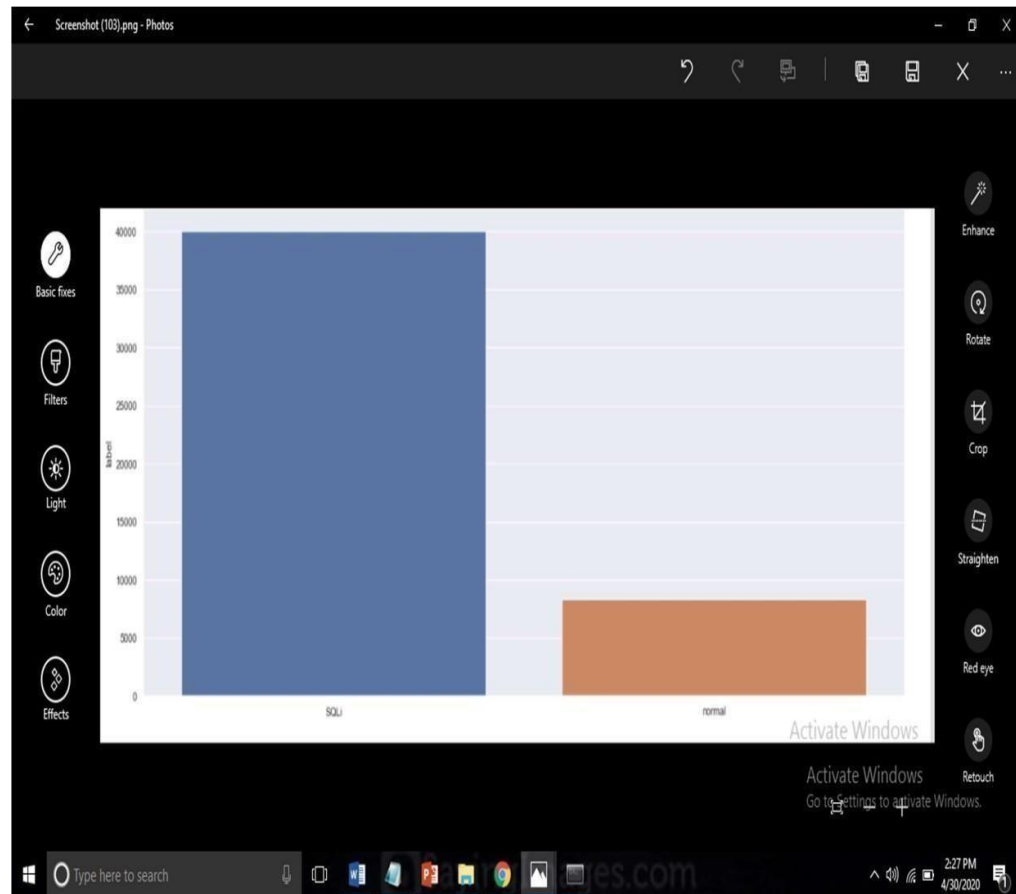


Fig .4.1.2 .gathering of data sets

In fig 16.we give four sql injection attack datasets and one non sql injection dataset to train the adaboost algorithm and based on it can provide information whether the website is attacked or non attacked .

Detecting SQL injection attack in complex network environment is a hotspot in the field of network security. The methods based on classical machine learning are difficult to process multiple features and redundant features, while the methods based on deep learning have multiple hyper-parameters, and it's easy to cause over-fitting. Therefore, we proposed a SQL injection detection method based on adaptive deep forest model. By using ADF, the model parameters can be automatically adjusted during the training process, which improves the detection accuracy. We compared the accuracy, precision, recall, f1-score between our proposed method and classical machine learning methods. The results show that our method achieved higher efficiency

Future work:

There is scope for further improvement in the WAPS-CIVS prevention system. The SQL injection preventer of WAPS-CIVS does not address injection through the stored procedure. The Stored procedure is a common method to operate the database at the database server. The WAPSCIVS considers the SQL statement crafted at the application / business logic of the web application. The XPath injection is similar to SQL injection, and nowadays, it has become popular where XML data is used as a data store for the web application. The WAPS-CIVS addressed only the tautology based XPath injection in a web application. It does not address the insert XQuery and union XQuery. The XPath injection preventer can be further improved by addressing issues other than tautology based injections. A Cross-site request forgery

attack, also known as CSRF or XSRF (pronounced sea-surf) is the well known, but equally dangerous like the Cross Site Scripting (XSS) attack. The cross-site scripting preventer system of WAPS-CIVS can be further improved by integrating the CSRF prevention mechanism. The session

hijacking prevention can be further enhanced, based on complex encryption techniques and taint information flow in web applications.

REFERENCES

- [1] B. Appiah. (2017). OWASP Top 10, 2017. [Online]. Available: https://www.owasp.org/index.php/Top_10-2017_Top_10 Mitropoulos, P. Louridas, M. Polychronakis, and A. D. Keromytis, "Defending against Web application attacks: Approaches, challenges and implications," *IEEE Trans. Depend. Sec. Comput.*, vol. 16, no. 2, pp. 188–203, Mar./Apr. 2017.
- [2] E. Pollack, "Protecting against SQL injection: Applications, performance, and security in microsoft SQL server," in *Proc. Dyn. SQL*, 2019, pp. 31–60.
- [3] A. Maraj, E. Rogova, G. Jakupi, and X. Grajcevci, "Testing techniques and analysis of SQL injection attack. A. Sarhan, S. A. Farhan, and F. M. Al-Harby, "Understanding and discovering SQL injection vulnerabilities," in *Proc. Int. Conf. Appl. Hum. Factors Ergonom.*, 2017, pp. 1063–1075.
- [4] D. Das, U. Sharma, and D. K. Bhattacharyya, "Defeating SQL injection attack in authentication security: An experimental study," *Int. J. Inf. Secur.*, vol. 18, no. 1, pp. 1–22, 2017.
- [5] H.-C. Huang, Z.-K. Zhang, H.-W. Cheng, and S. W. Shieh, "Web application security: Threats, countermeasures, and pitfalls," *Computer*, vol. 50, no. 6, pp. 81–85, 2017.
- [6] H. Shahriar, S. North, and W.-C. Chen, "Client-side detection of SQL injection attack," in *Advanced Information Systems Engineering Workshops (Lecture Notes in Business Information Processing)*, vol. 148. Berlin, Germany: Springer, 2013, pp. 512–517.
- [7] P. R. McWhirter, K. Kifayat, Q. Shi, and B. Askwith, "SQL injection attack classification through the feature extraction of SQL query strings using a gap-weighted string subsequence kernel," *J. Inf. Secur. Appl.*, vol. 40, pp. 199–216, Jun. 2018.
- [8] Y. Wang, D. Wang, W. Zhao, and Y. Liu, "Detecting SQL vulnerability attack based on the dynamic and static analysis technology," in *Proc. IEEE Comput. Softw. Appl. Conf.*, Jul. 2015, pp. 604–607.

[9] B J. Santhosh Kumar and P. P. Anaswara, "Vulnerability detection and prevention

of SQL injection,” *Int. J. Eng. Technol.*, vol. 7, no. 2.31, pp. 16–18, 2018.

[10] C. Gould, Z. Su, and P. Devanbu, “Static checking of dynamically generated queries in database applications,” in *Proc. 26th Int. Conf. Softw. Eng.*, May 2004, pp. 645–654.

[11] A. Naderi-Afooshteh, A. Nguyen-Tuong, M. Bagheri-Marzijarani, J. D. Hiser, and J. W. Davidson, “Joza: Hybrid taint inference for defeating Web application SQL injection attacks,” in *38 2015*, pp. 172–183.

[12] W. Yi, L. Zhoujun, and A. Guo, “Literal tainting method for preventing code injection attack in Web application,” *J. Comput. Res. Develop.*, vol. 49, no. 11, pp. 2414–2423, 2012.

[13] L. Li, J. Qi, N. Liu, L. Han, and B. Cui, “Static-based test case dynamic generation for SQLiVs detection,” in *Proc. Int. Conf. Broadband Wireless Comput.*, Nov. 2015, pp. 173–177. [14] B. Appiah, E. Opoku-Mensah, and Z. Qin, “SQL injection attack detection using fingerprints and pattern matching technique,” in *Proc. Int. Conf. Softw. Eng. Service Sci.*, Nov. 2017, pp. 583–587.

[15] L. K. Shar and H. B. K. Tan, “Defeating SQL injection

[16] J. Choi, H. Kim, C. Choi, and P. Kim, “Efficient malicious code detection using N-gram analysis and SVM,” in *Proc. Int. Conf. Netw.-Based Inf. Syst.*, Sep. 2011, pp. 618–621.

[17] X. Li, J. Qu, G. Yao, J. Chen, and X. Shen, “Design and implementation of an automatic scanning tool of SQL injection vulnerability based on Web crawler,” in *Proc. Int. Conf. Secur. Intell. Comput. Big Data Services*, 2018, pp. 481–488.

[18] L. Zheng, L. Yuan, X. Peng, G. Zhu, Y. Guo, and G. Deng, “Research and implementation of Web application system vulnerability location technology,” in *Proc. Int. Conf. Adv. Intell. Syst. Comput.*, 2010, pp. 937–944.

[19] S. Akcay, M. E. Kundegorski, C. G. Willcocks, and T. P. Breckon, “Using deep convolutional neural network architectures for object classification and detection within X-ray baggage security imagery,” *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 9, pp. 2203–2215, Sep. 2018.

[20] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and

S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," IEEE Access, vol. 7, pp. 41525– 41550, 2019.

[21] T. Zhou, X. Sun, X. Xia, B. Li, and X. Chen, "Improving defect prediction with deep forest," Inf. Softw. Technol., vol. 114, pp. 204–216, Oct.2019.

[22] Q. Li, F. Wang, J. Wang, and W. Li, "LSTM-based SQL injection detection method for intelligent transportation system," IEEE Trans. Veh. Technol., vol. 68, no. 5, pp. 4182–4191, May 2019.

[23] Y. Fang, J. Peng, L. Liu, and C. Huang, "WOVSQLI: Detection of SQL injection behaviors using word vector and LSTM," in Proc. ICCSP, 2018, pp. 170–174.

APPENDICES

Sample Code:

```
In [ ]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from keras import layers
from keras.models import load_model
from keras.preprocessing.text import Tokenizer
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.callbacks import TensorBoard
from sklearn import metrics
from urllib.parse import urlparse, parse_qs
```

```
In [ ]: df_attacks1 = pd.read_csv('SQLInjectionDataSet1.csv')
df_attacks2 = pd.read_csv('SQLInjectionDataSet2.csv')
df_attacks3 = pd.read_csv('SQLInjectionDataSet3.csv')
df_attacks4 = pd.read_csv('SQLInjectionDataSet4.csv')
```

```
In [ ]: df_attacks = pd.concat([df_attacks1,df_attacks2,df_attacks3,df_attacks4])
```

```
In [ ]: df_attacks.head()
```

```
In [ ]: df_normal = pd.read_csv('NonSQLInjection.csv')
```

```
In [ ]: df_normal.head()
```

```
In [ ]: dataset = pd.concat([df_attacks, df_normal])
```

```
In [ ]: dataset.head()
```

```
In [ ]: labels_type = dataset['label'].value_counts().keys().tolist()  
labels_type
```

```
In [ ]: dataset['label'].value_counts()
```

```
In [ ]: dataset.groupby("label").size()
```

```
In [ ]: sns.set(rc={'figure.figsize':(19.7,8.27)})  
sns.barplot(x=dataset['label'].value_counts().index, y=dataset['label'].value_counts())
```

```
In [ ]: dataset['method'].value_counts()
```

```
In [ ]: sns.set(rc={'figure.figsize':(19.7,8.27)})  
sns.barplot(x=dataset['method'].value_counts().index, y=dataset['method'].value_counts())
```

```
In [ ]: result = dataset.loc[dataset['method'] == 'GET*']  
print(result)
```

```
In [ ]: dataset.loc[result.index, "method"] = "GET"
```

```
In [ ]: sns.set(rc={'figure.figsize':(19.7,8.27)})  
sns.barplot(x=dataset['method'].value_counts().index, y=dataset['method'].value_counts())
```

```
In [ ]: result = dataset.loc[dataset['method'] == 'GET*']  
print(result)
```

```
In [ ]: dataset.loc[result.index, "method"] = "GET"
```

```
In [ ]: dataset.loc[dataset['method'] == 'GET*']
```

```
In [ ]: dataset['method'].value_counts()
```

```
In [ ]: sns.barplot(x=dataset['method'].value_counts().index, y=dataset['method'].value_counts())
```

```
In [ ]: dataset.loc[dataset['path'].str.contains('jsp')]
```

```
In [ ]:
```

Screenshots:

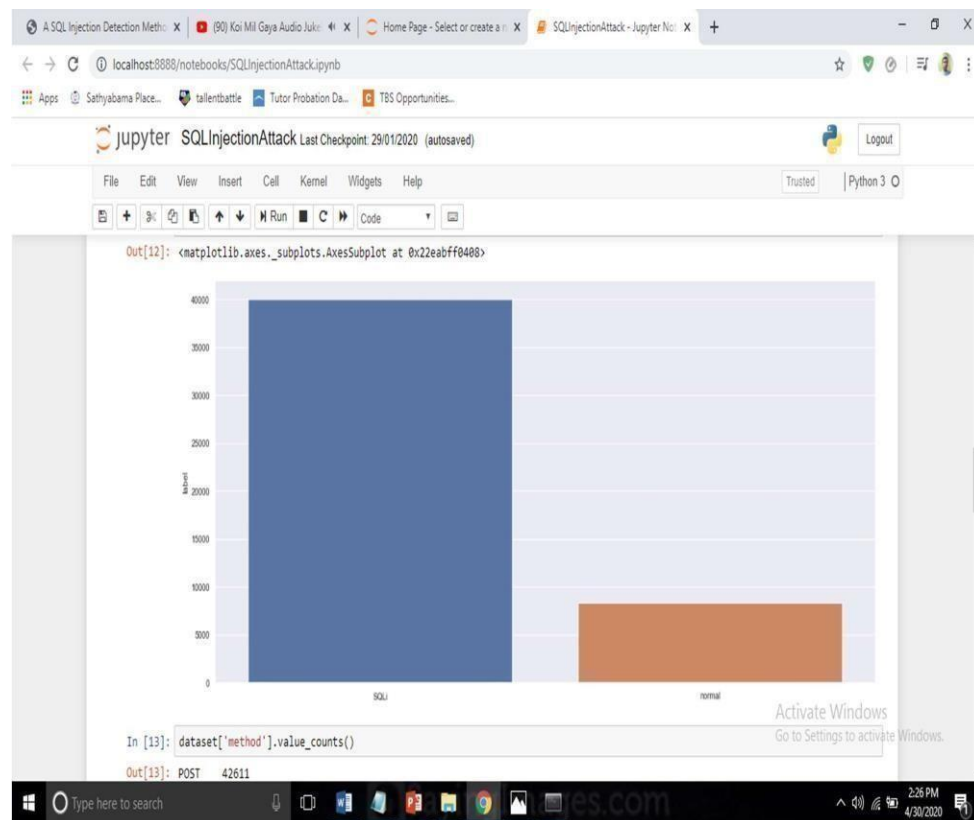


Fig .5.1.1 using get and post methods

In fig 17 by using get and post methods we can determine whether it is sql injection attack or not .

0.0
%

PlagScan by Original Results of plagiarism analysis from 2021-03-09 04:35 UTC

paper detection of sql-converted-converted (1) (1).pdf

Date: 2021-03-09 04:34 UTC

All sources 0

4 pages, 1320 words

PlagLevel: 0.0% selected / 0.0% overall

0 matches from 0 sources, of which 0 are online sources.

Settings

Data policy: *Compare with web sources, Check against my documents*

Sensitivity: *Medium*

Bibliography: *Consider text*

Citation detection: *Reduce PlagLevel*

Whitelist: --

SQL Injection Attack Detection Using Deep Forest

¹A.Sivasangari

Dept of IT, Sathyabama University
Chennai, India
sivasangarikavya@gmail.com

²I.Jyotsna

Dept of CSE, Sathyabama
University Chennai, India.
jyotsnainnampudi@gmail.com

³K.Praval

ika
Dept of CSE, Sathyabama University
Chennai, India.
pravalikayadav42@gmail.com

Abstract— An injection attack may be a malicious code that's injected within the network that fetches all the data from the info to the assailant. this kind of attack is taken into account as major downside in internet security and is listed because the darling internet application security risk within the OWASP prime ten. To sight a SQL injection attack continues to be a tough downside because the admin of the info won't apprehend the attack is going on till there's amendment within the content of the info. The way to sight the injection attack with success is the foremost vital part of web applications. Hence, the paper introduces adaboost formula that sight numerous styles of injection attacks. The worth of the each weak tree is given highest weight to get the strong model by updating the weights every step in our paper by training of the dataset, by adding the input of every layer by calculating average of previous outputs. Results indicate that the proposed algorithm and program accurately detects the injection attack effectively than that of the initial options of neural techniques area unit degenerated due to increasing variety of intermediate layers within the program.

Keywords—comparison with neural network, adaboost algorithm

I. INTRODUCTION

With the speedy development of internet network applications have step by step become an essential in our part of lives. At identical time, internet applications area unit confronted with a lot of challenges. As there are many ways to write the SQL query to attack the database or website it is difficult to know which type of attack has been used. There are several methods in which SQL injection can be induced in database such as error based, union based, out of band and blind SQL injection attacks. In this era where the data of individuals is mostly stored online attackers find it very easy to attack through SQL attacks and gain benefits from the data. This type of attack became one among the foremost type of malicious attacks through which offensive ways of modification often with the event of web site technology can be done by the attacker for his own benefit.

II. RELATED WORK

Earlier to detect this type of SQL attacks there are traditional methods such as techniques to validate the input, white box approach for input validation, defensive coding etc., the defensive coding most of the times cannot detect the SQL injection attack. Sometimes the vulnerabilities is due to the wrong input that is given by the user to avoid this white box approach was used for validating the input.

William G.J. Halfond et al worked to counter the SQL injection attack by runtime monitoring of the web application by identifying the root cause of the SQL injection attack. They proposed the model based approach by which they target the SQL injection attack. When the attacker trying to execute the query that is generally not in the structure of a general non malicious query it detects and prevent from executing and through notification giving warning to the admin.

The recent researchers used the AI technology to identify the SQL injection attack in the websites mechanically observe attacks. Rubidha Devi et al identified the types of SQL injection attacks with the queries and prevention techniques that are used before 2011.

Ryan Wheeler et al used a tool BCSQL to detect the blind SQL injection. The tool mainly is useful for the Boolean based type of attacks. IN database the tool automatically allows access to the log files, number of times the query has been requested, whether it is then incomplete query or complete which allows us to know the prominent query the attacker used to attack.

III. FRAMEWORK OVERVIEW

In this architecture it consists of following steps first to collect the data, collect the SQL statements samples, train the data, for input validation check the inputs are given to the model checking whether the output is valid or not and then using model. In the process of the validation check the input data are processed and decoded.

The algorithm that has been used is Adaboost algorithm which is considered to the best out of all the algorithms in classifiers because it is less prone to overfitting. In this data is converted into stumps and the stumps are classified into weak stumps and strong stumps. The more weight is generally given to wrongly classified data to classify it correctly in the next step, so it is a layer by layer approach. The strong stumps given the highest weight in the overall output and weak stumps given the less weight in the output. The learner can be of weak but the final model can be of strong learners. This algorithm improves the accuracy, precision in the weak learners which ultimately make the model as strong model though sensitive to noisy data. If an error is encountered that is corrected with the updating of weights until the accurate model is designed.

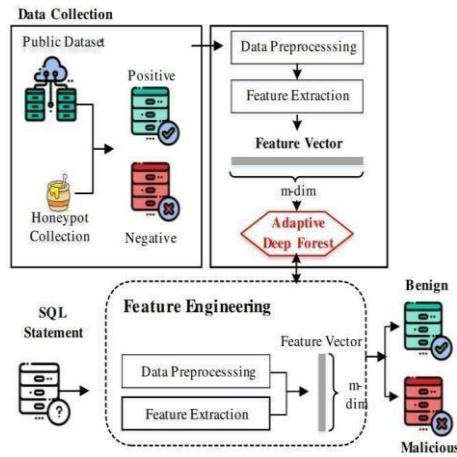


Fig. 2 Model implementation and application

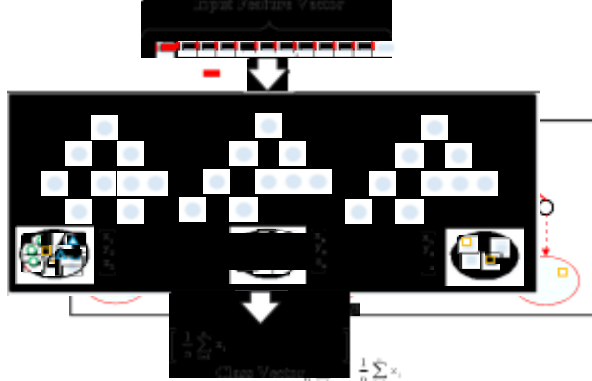
A. Adaboost model

To calculate error for the train model

$$\text{Error} = (\text{correct} - N) / N$$

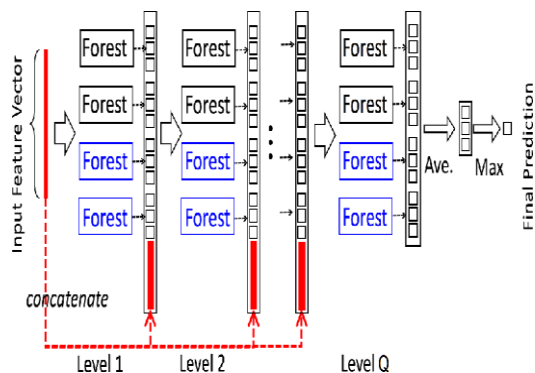
To modify the weight

$$\text{Error} = \sum(\text{weight}(i) * t \text{ error}(i)) / \sum(\text{weight})$$

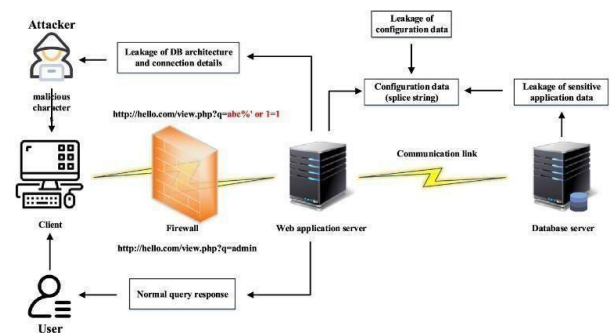


To detect injection attack the algorithm classifies the input data into two that is binary classification that is every stump produces a class vector length is 2.

B. Incorporating Adaboost algorithm



C. Data and experiment setting:



Hardware Requirements:

Two Core Intel Processors,

- 1TB hard disk
- 32Gb ram

Software

Requirements:

- Microsoft Windows Server 2016,
- Sql server,python

Import required libraries in python:

Import AdaBoost Classifier from sklearn to classify the data
Import datasets
Import metrics library for accuracy calculation

```

Load Data
Dataset=load_injection()
X = injection.data Y =
injection.target

```

```

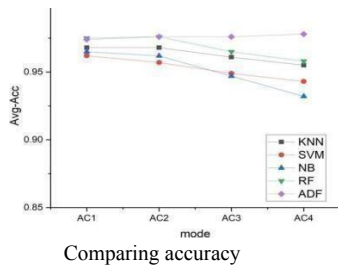
Adaboost Model:
modelinjection= fit(X train, Y train) Y.pred
= modelinjection.predict(X_test)

```

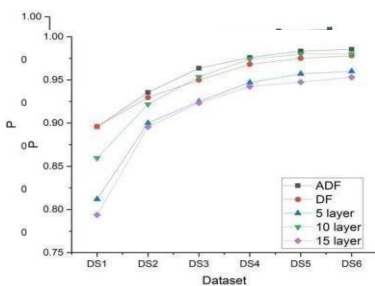
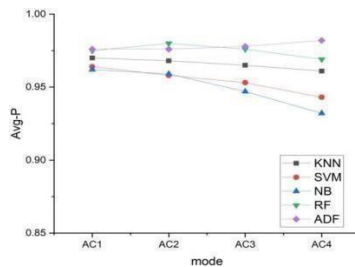
IV. PERFORMANCE ANALYSIS

An experimental analysis for detecting SQL injection is conducted.

Comparing with other algorithms that are exiting to classify data



Comparing Precision



When compared to the other algorithms that can be used for binary classification the advantages for using the adaboost algorithm are

Not prone to overfitting High accuracy

Accommodative for categorical and continuous values Higher dimensionality and flexibility

CONCLUSION

In this paper, using adaboost algorithm we detected the SQL injection attack. As there are major 4 types in which attacker can attack the database or the website we tried to incorporate the data samples and SQL statements in data set and tried to train the model to get accurate result when subjected to real time environment. This is more useful of the companies that manage the data online. Due to the increasing use of internet to manage, store the data it is high time to protect data by such real time projects.

REFERE NCES

- [1] B. Appiah, "OWASP Top 10, 2017," 2017. [Online]
- [2] Das D, et al. Defeating SQL injection attack in authentication security: an experimental study, 2017.
- [3] <https://ieeexplore.ieee.org/xpl/conhome/8906127/proceeding,2019>
- [4] Maraj A , Rogova E , Jakupi G , et al. Testing Techniques and Analysis of SQL Injection Attacks[C], IEEE, 2017.
- [5] Mitropoulos D, Louridas P, Polychronakis M, et al. Defending Against Web Application Attacks: Approaches, Challenges and Implications. 2019.
- [6] Pollack E. Protecting Against SQL Injection: Applications, Performance, and Security in Microsoft SQL Server 2019.
- [7] Rubidha Devi.D* et al. IJPT| Dec-2016 | Vol. 8 | Issue No.4 | 22405-22415
- al. BlindCanSeeQL:|Graduate Theses and

[9] Shahriar H , North S , Chen W C . Client-side detection of SQL injection attack[J],2013

[10] Vinayakumar R, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," IEEE Access vol7

[11] William G. Halfond, Alessandro Orso, "Using Positive Tainting and Syntax Aware Evaluation to Counter SQL Injection Attacks", V. Nithya, IJECS Volume 2 Issue 4 April,

