SURGICAL STRIKING SQL INJECTION ATTACKS USING LSTM

Joshi Padma N

JNTUH Research Scholar, Associate Professor
Dept of CSE, Sreyas Institute of Engineering and Technology
Hyderabad, India
padmajoshi2015@gmail.com

Dr. N. Ravishankar

Professor, Dept of CSE
Geethanjali College of Engineering and Technology
Hyderabad, India
ravish00@yahoo.com

Dr. M. B. Raju

Professor, Dept of CSE
Pallavi Engineering College
Hyderabad, India
drrajucse@gmail.com

N.Ch. Ravi

Associate Professor,
Dept of CSE,
Pallavi Engineering College,
Hyderabad, India
ravi@saimail.com

Abstract

When a software program or website employs SQL databases such as Oracle, SQL Server, or MySQL, a SQL injection attack is possible. Phishers utilize SQL injection attacks to infiltrate confidential corporate and personal information, increasing the risk of compromise. If a SQL injection exploit is successful, it can be used to read and alter database data (e.g., insert, update, or delete), perform database administration operations, retrieve the data of a database management system file, also issue commands to the operating system in some cases. In comparison to basic recurrent architectures, LSTM networks have been demonstrated to learn long-term dependencies more quickly and reduce gradient disappearance and explosion. Preventing SQL injection with the usage of LSTMs is the subject of the proposed research. The LSTM model has been suggested as a training method for tracing or filtering requests from an unauthentic user. The dataset trained is looking at records that include the sender and recipient information, as well as the success or failure of the transmission. If this transaction has repeatedly failed due to an on-going assault on the system, then this is a red flag. Then, this system will flag the transaction and block the user from accessing any of the system's resources. This model has been trained to increase security by creating a system that allows the user to authenticate a transaction.

***Keywords*: SQL injection, Neural networks, LSTM.**

1. Introduction

SQL databases such as Oracle, SQL Server, or MySQL are prone to a SQL injection attack. Phishers utilize SQL injection attacks to infiltrate sensitive corporate or individual data (PII). When a SQL injection attack succeeds, the attacker may be able to read or manipulate confidential database data, carry out database administration operations (such as inserting, updating or removing data), extract database file content, and even send operating system commands. Long-term dependencies can be learned more slowly by recurrent and LSTM networks, respectively, and the problem of gradient disappearance or explosion can be lessened. As part of the suggested solution, LSTMs are employed to guard against SQL injection.

1.1. *SQL INJECTION*

It is possible for an Phisher to tamper with an online application's database queries by using SQL injection, which is a web security vulnerability. In most cases, it gives an attacker access to information they otherwise wouldn't have. The data that the app has access to may include your own or that of other users. An attacker may be able to alter or remove this data, resulting in long-lasting effects on the app's functionality or content. Denial of service attacks or SQL injections can be used to compromise a server's underlying infrastructure, as well as other backend systems. Any sensitive data that is compromised as a result of an effective SQL injection attack is at risk of being accessed by an attacker. Recently, SQL injection attacks were implicated in many well-known data breaches, leading to reputational harm and regulatory punishments. Having a long-term backdoor into an organization's systems may allow an attacker to remain undiscovered for an extended period of time. All kinds of SQL injection flaws, attacks, and techniques can be employed in different ways. Some common examples of SQL injection are discussed ahead:

- Hidden data can be retrieved by altering a SQL query to produce additional results.
- You can update a query to alter the application's reasoning by subverting its logic.
- In a UNION attack, you can access data from many database tables at the same time.
- In this step, you'll learn more about the database's version and structure.
- Query results that you control are not returned in the application's answers, which is known as "blind SQL injection."

1.1.1 Detecting SQL injection vulnerabilities

The web vulnerability scanner in Burp Suite can rapidly and consistently detect the vast majority of SQL injection issues. By running a systematic set of tests against each entry point in the application, SQL injection can be spotted manually. Typically, this entails:

- Searching for errors or other anomalies by submitting the ‘character and examining it.
- Looking for systematic changes in the ensuing application answers by submitting SQL-specific syntax which evaluates to the entry point's base (original) value and to a different value.
- Checking for discrepancies in the application's replies to Boolean criteria like OR 1=1 and OR 1=2.
- For testing SQL queries, submitting payloads designed to cause delays in response time when run.
- Payloads designed to trigger out-of-band network interactions when run within a SQL query are sent and monitored.

WHERE clause of SELECT queries are most vulnerable to SQL injection. Experienced testers are familiar with SQL injection of this type. There are several places in a query where SQL injection vulnerabilities can occur, and they can occur in a variety of different ways. SELECT statements, within the table or column name; SELECT statements, within the ORDER BY clause; and UPDATE statements, within the WHERE clause are some of the other common sites where SQL injection takes place.

1.1.2 SQL Injection Attack Performance

A structured query is used to elicit the desired response when executing SQL injection. The response is crucial for the attacker to understand the database design and get access to the application's protected data. SQL injection can be achieved in a variety of ways, including the following:

The only true statement in SQL. A Phisher uses a SQL statement which is always true to do a SQL injection. Rather than just typing in the "wrong" number, the Phisher utilizes a statement that is always true. A table's details can be retrieved by typing "100 OR 1=1" into the query input box.

""" This SQL injection technique is identical to the one outlined in the previous section. Entering "OR ""=" into the query box is required by a nefarious user. The malicious code used to get access to the program is hidden in these two indicators. Take a look at this example. Simply typing "OR=" in the user ID or password is all it takes for an attacker to steal data from an application. Using this SQL statement, the database's user table can be accessed by returning its data.

It's possible to use semicolons to separate multiple SQL statements while doing batched SQL injection. If this strategy is to be successful, then the SQL statements must be correct and legitimate, which means the statement following the semicolon must be correct as well. Phishers can use SQL statements like "105; DROP TABLE Supplier;" to destroy a table from an application's database.

1.1.3 Types of SQL Injection

The three types of SQL injection are: in-band, inferential, and out-of-band. In-band SQL injection is the most common.

(1) In-band SQL injection is the most common sort of SQL injection attack. The UNION operator in SQL statements, as well as error messages on the web, can be used to transfer data during in-band attacks It is possible to perform in-band SQL injection using either a union or an error. A union-based SQL injection. When an application is vulnerable to SQL injection, attackers can obtain data from other tables in the application database using the UNION keyword. An error led to a SQL injection. SQL injection attacks are predicated on the application database servers' error messages. Using this information, the attackers can find out which database entities are in the system.

(2) The term "blind SQL injection attack" is sometimes utilized to describe an inferential SQL injection. In a blind SQL injection attack, the attacker sends a data payload and monitors the database's activity and answers to identify the database's data structure. Both Boolean and time-based SQL injection methods are considered blind or inferential attacks. In order to push the program to deliver a Boolean result—which is, either TRUE or FALSE—the Boolean-based approach sends SQL queries to the database. In order to discover a vulnerability, attackers use a variety of blind queries. When an application creates generic error messages, the time-based SQL injection technique is widely deployed. With this strategy, the database is forced to wait for a specified period of time before it may be accessed. The attacker can detect if a query's results are TRUE or FALSE depending on the server's response time.

(3) Out-of-band Any protocol can be used, including HTTP, DNS, or SMB in an out-of-band SQL injection attack. The following SQL and MySQL functions can be used to launch an attack of this nature: SQL: master..xp _dirtree MS Access LOAD FILE is a MySQL function ()

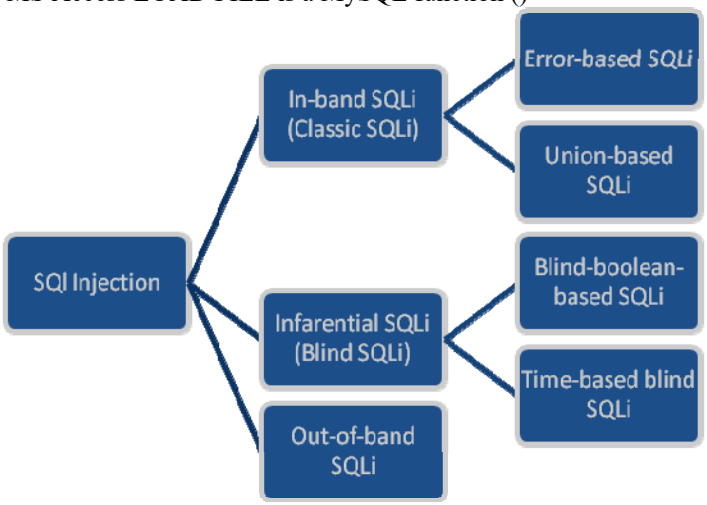

Fig 1 Types of SQL infusion Types

1.1.3 Preventing SQL Injection Attacks

SQLi attacks are dangerous, but they can be avoided if you follow a few best practices for secure coding, which include understanding the basics like these:

Finding flaws in your code is only possible through thorough testing. When it comes to securing your app, go with powerful tools like dynamic analysis (DAST), which examines the app from the outside in, and static analysis (SAST), which looks for flaws in the code. Look for places in your software where it communicates with a database and try to pass it odd values. If a single quote appears in the value, does the software treat it as user data or as code? As if we had supplied a legitimate password, we can acquire access even if we include a tautological test (such as 'OR '1=1').

It's time to fix any vulnerabilities that have been found and remediate the situation. By using placeholder values in our statements and feeding user-inputted values into the statements at the time of execution, we can achieve this goal more easily. Using a sanitization or escape function before submitting input to a database can fix your code if your programming language doesn't provide parameters. Rather than executing code, our app now understands that the input it receives is data.

Reduce impact: Mitigation is a critical step in reducing risk, but it does not address the underlying problem. If a defect can be found in our app's database accounts instead than its code, we may make a fix by ensuring those accounts have the least amount of access necessary to read and insert data into our database. In my previous research paper works[1][2][3][4][5] ,I proposed a frame work for prevention of SQL injection and XSS.

1.2. *Neural Networks and RNN*

In Neural Networks, deep learning is accomplished utilizing biological neurons, which is nothing more than a brain cell. In this case, Thus, neural networks with a large number of hidden layers are employed to implement deep learning. It is a set of algorithms that tries to recognize the patterns, correlations, and information from the dataset through the process inspired by and working like the human brain. Neural Networks Recurrent neural networks rely heavily on data from prior runs. A person must be familiar with the words that came before in order to properly guess the following word in a phrase. Since no future input is taken into account for the current state, the computational speed of this recurrent neural network is extremely poor. Despite the fact that it has a hard time recalling the past. Recurrent neural networks can be used to address problems using time series data, text data, and audio data. The parameters of RNNs are shared between time steps. Parameter Sharing is a common term for this. There are fewer variables to train, which reduces the amount of computing time required.

1.3. *Networks with long short-term memory (LSTMs)*

Many activities can benefit greatly from the use of "LSTMs," a special type of recurrent neural network. Recurrent neural networks are used to achieve nearly all of the most fascinating findings. The long-term reliance problem is specifically avoided with LSTMs. They have a natural ability to retain information for lengthy periods of time, and it's not something they have to work at. This is the structure that every recurrent neural network takes: a sequence of neural network modules that keep repeating. The repeating module of LSTMs has a different structure than that of an LSTM chain. There are four layers of neural networks instead of just one, and these layers interact in a unique way.

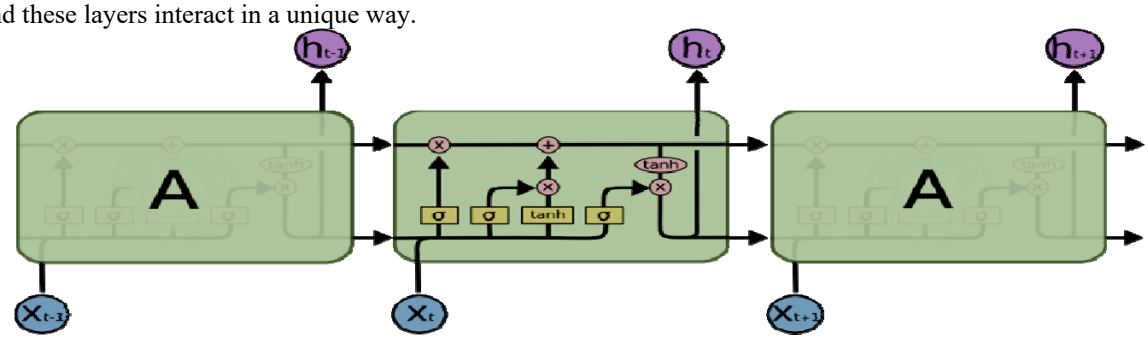

Fig 2 Networks with long short-term memory (LSTMs)

A LSTM neural network is shown schematically in the above figure. An LSTM's repeating module has four levels that interact with one another.

A whole vector can be shown in the diagram above, with each line connecting the output of one node to the inputs of the others. Layers of a taught neural network are shown by the yellow boxes while the pink circles represent pointwise operations such as vector addition. Forking lines depict that the content of a line has been copied and sent to separate destinations, while merging lines indicate that the material has been concatenated. The horizontal line running through the top of the figure is the key to LSTMs. The status of a cell can be compared to that of a conveyor belt. With only minor linear interactions, it travels straight down the entire chain. Flowing information is a very common occurrence. Cell state information can be removed or added by structures called gates, which are carefully controlled by the LSTM. Gates are a way to allow or deny access to information, depending on the situation. Sigmoid neural nets and point wise multiplication are the building blocks of these algorithms. Numbers ranging from 0 to 1 indicate how much of each component should be allowed past the sigmoid layer. "Let nothing in," while "Let everything in!" is the meaning of a value of zero. Three of these gates protect and regulate the status of the LSTM cell. Sequence and time series problems will benefit greatly from LSTMs in the future. Despite this, they have one major drawback: training them is difficult. Even a simple model requires a significant amount of time and system resources to train. However, this is only a limitation of the hardware.

1.4. *SQL injection prevention using LSTMs*

LSTM-based SQL injection detection takes the following steps: Labeling, normalizing, word segmentation, and embedding word vectors are some of the preprocessing steps. We assign the number 1 to SQL injection samples and the number 0 to the negative sample based on the data classification. In order to detect and decode alternative encodings, such as url encoding and base64 encoding, all English characters in example files are converted to lowercase. Create a segmentation model for words based on your existing knowledge, utilizing symbols such as "(",")", "", and "percent". Word Embedding is applied to the segmentation results, and the resulting numeric word vectors represent each word. Sentences of varying durations are sent into the LSTM layer, which then runs them through a time series to produce the hidden LSTM neural unit vectors. A vector h can be derived once these vectors have passed through the mean pooling layer. SoftMax is the last layer. In

order to choose the final prediction category, we can obtain a class distribution vector and then look for the output category with the highest probability. Training data is used to train the model and hyper-parameters are adjusted as necessary. The last epoch is 30,000. The model is generated after 3000 training runs. The accuracy and F1-score are used as the primary metrics for assessing the system. To detect SQL injection attacks, the best model from the output models is used as a classifier after training. After normalization, word segmentation, and embedded word vectoring, the test data is fed into the classifier, where it is evaluated. Otherwise, it indicates that the user input is a SQL injection attack statement. If the classifier's result is 0.

2. Literature Review

For our research on the topic “SQL injection prevention using LSTM networks” we undertook ideas and knowledge from reading the various research papers prepared by different researchers on SQL injection, neural networks and that on LSTMs. Forthcoming is a brief intro to the papers that act as a basis for our work and help to keep the research in right direction towards our research objective.

There have been a variety of approaches developed to identify and fight SQL injection attacks by S.S. Anandha Krishnan et al. [6]. SQL Injection attacks remained a matter of concern for cyber safety experts. Since attackers were regularly deploying new types of SQL Injections, signature-based detection solutions for SQL Injections were no longer reliable. Safety solutions must be able to distinguish between novel, never-before-seen SQL Injection attempts. Many researchers were considering applying machine learning to cyber-safety. The use of machine learning in cyber-safety is still a developing topic, and as a result, there are various open-source libraries and tools that can be used to combat threats and assaults.

CNN-LSTM neural networks with particle swarm optimization (PSO) have been proposed [7] by Tae-Young Kim et al. for the classification of roles in the RBAC system. An analytical process in a convolutional neural network (CNN) can extract tiny details and features from parsed SQL queries. LSTM was also well-suited for modelling the temporal information in SQL queries in order to detect the context of user authorities. The CNN-hyper parameter LSTM's space was repeatedly searched and tuned by PSO. In the TPC-E benchmark SQL query statement, our CNN-LSTM neural networks based on PSO beat other deep learning and machine learning models. A last set of trials and analyses showed how effective PSO was at classifying users based on their SQL query features.

For the detection of SQL injection vulnerabilities in a web application using deep learning, Maruf Hassan et al. [8] recommended extracting numerous web application discovering points. The portion on SQL injection vulnerability prediction relied heavily on the deep learning component, which was extensively discussed in the article. A neural network model was used to forecast the susceptibility of more than 1,850 samples of injection vulnerability. The accuracy of the system's performance, at 98.04 percent, was higher than that of previous systems.

In this paper, Hazem M. Harba et al. [9] aims to present an overview of the SQL injection (SQLI) assault and to classify these attacks as well as preventive and detection instruments. They discussed the most recent methods and tools for preventing and detecting SQLI, as well as its advantages and disadvantages. Languages all have strengths and weaknesses. For every language, there was no universal method that could be implemented with the same level of efficacy. They opined that in order to evaluate and examine the various features of the procedures, further analysis and comprehensive research of the current approaches were required (e.g., detection rate, overhead, and false-positive rate, etc.). General criteria for these approaches' evaluation must also be developed.

An updated LSTM model with bidirectional deep fusion, alertness mechanisms, and parameter self-learning has been presented by Tianwei Zheng et al. [10] to accomplish online soft computing for industries and elements' coal quality evaluations. To begin, a model of latent structure was constructed to help with the preprocessing of the noisy and redundant sensor network data. Second, an awareness mechanism was provided, and the data feature extraction was done using the self-learning approach of the activation function parameters. A bidirectional layer of deep fusion was then added to the LSTM neural network model to address the issues of poor accuracy and generalization. The DFAS-LSTM model was developed using sensor network historical data. Finally, the DFAS-LSTM model was used to implement the online coal quality analysis using the sensor network's online data. In comparison to the typical bidirectional LSTM, the accuracy of the coal quality analysis was improved by 1 percent –2.42 percent in the experiment.

Xuyan Song et al. [11] have presented new deep learning-based approaches for detecting malicious JavaScript. A programmer dependency graph (PDG) was developed and semantic slices were generated, which were easy to translate into vectors, so that semantic information could be extracted from JavaScript programs. After then, it was suggested to use the BLSTM neural network to identify malicious JavaScript. With an accuracy of 97 percent and an F1-score that was nearly perfect, their model outperformed the rest of the approaches tested. Neuronal networks with tree or graph architectures were among the many novel concepts they proposed.

The Support Vector Machine (SVM) approach was introduced by T.P.Latchoumi et al. [12] to identify and prevent SQL injection. Their method involved training an SVM algorithm with every conceivable harmful expression before using that data to construct a model. To determine whether or not a given query contained any harmful expressions, SVM was applied to the model whenever a new query was submitted by a user. It is possible that SVM could detect a harmful expression by matching it against a small number of syntaxes, even if the user developed the new approach. They showed that predictive analysis can detect and prevent SQLIA in a big data setting with good outcomes that can be evaluated empirically in the mixing matrix and ROC chart.

A neural network-based SQL injection detection system was developed by Peng Tang et al. [13]. In order to ensure that their approach was real, successful, and practicable, they first obtained authentic user URL access log data from the Internet Service Provider (ISP). Then, a statistical analysis was performed on both normal and SQL injection data sets. In light of the statistical findings, they created eight types of features and trained an MLP model. The model's accuracy remained above 98%. Comparing their method to other machine learning algorithms (such as LSTM) and evaluating their training effect found that their way was more accurate. Although LSTM and MLP had a performance disparity, LSTM had certain advantages. Because of the order in which the characters appear in the string, LSTM was able to distinguish between SQL statements and regular statements using the association between characters before and after that order.

New detection and prevention methods for SQL injection have been proposed by Ines Jemal and her colleagues [14]. They categorized the various assault sources, goals, and types. As a follow-up to this discussion, they categorized the most recent and important proposed solutions to prevent this threat, particularly those based on ontology and machine learning. Completeness of the training and testing of the classifier is achieved by selecting a well-ordered dataset that is near to reality. Because of this, it was able to identify SQLIA successfully.

An adaptive deep forest-based technique to identify complicated SQL injection attempts was proposed by QI LI et al. [15]. They began by improving deep forest's structure by concatenating the raw feature vector and averaging prior outputs at each layer's input. They found that their proposed strategy worked, however the original features of deep woods were damaged as the number of layers increased. AdaBoost was then used to update the feature weights in each layer using error rate data from the deep forest model. According to their impact on the final outcome, distinct aspects were given varied weights during training. Over-fitting can be avoided by adjusting the tree model's structure and handling multi-dimensional, fine-grained features.

[16] Fang Wang and colleagues have developed a long short-term memory SQL injection attack detection approach that can automatically learn the most appropriate representation of data and has a significant advantage in dealing with complex high-dimensional enormous data. In addition, from the standpoint of penetration, their paper offered a way for creating injectable samples based on data transmission channels. Formally modelling a SQL injection attack, they were able to create and validate affirmative samples using their method. Because of the lack of positive samples, the problem of over-fitting might be efficiently solved by this method. For SQL injection detection and false positive rate reduction, experimental results reveal that the suggested method outperformed numerous comparable classical machine learning methods and popular deep learning algorithms.

Database risks were identified by Mohd Amin Mohd Yunus et al. [17] as an immoral action by some hackers to take data and information illegally. The second purpose was to discuss database concerns such excessive privilege abuse, legitimate privilege abuse, privilege elevation, and platform vulnerabilities and how to avoid them. Thus, the Blockchain concept was established to solve the SQL injection problem by IP verification of nodes. Blockchain and Augmented Reality will be used to prevent SQL injection in the future (AR). SQL injection attacks may be enhanced by using AR to view them.

According to Jinpei Yan et al. [18], HDN is a new static malware detection method for Android that makes use of LSTM to learn from raw opcode sequences extracted from decompiled Android files. But because of this difficulty, many opcode sequences were too long for LSTM to learn. HDN's hierarchical structure made use of a first-level LSTM. Occurrence detection and classification might be achieved at the second level by employing LSTM, which could learn and detect malware via method block sequences while simultaneously learning the dense representation. HDN used adaptive gradient scaling for data denoising in partial sequence segments using the technique block denoise module because the malicious activity appears to be a one-time occurrence (MBDM). On the basis of the loss of cache. They investigated and contrasted HDN with the most recent mainstream research on three datasets. HDN surpassed the results of several Android malware detection algorithms, and it was able to capture longer sequence features and identify malware. When they tried to use the same method they used, the detection rate for malware was lower.

3. Problem Statement

It is possible to carry out SQL injection attacks by adding malicious SQL instructions into a website's form or domain name or page request and then tricking the website's server into executing them. When it comes to traditional methods of SQL injection defense, blacklist filtering and regular expressions are used. Although this method does not exclude keywords and strings that are not on the blacklist, web applications are still exposed to

new SQL injection attacks. Previous studies haven't been able to establish a definitive level of security. In other words, it is possible to carry out SQL injection attacks by adding malicious SQL instructions into a website's form or domain name or page request and then fooling the website's server into executing them. Blacklist filtering, which stores a blacklist and regular expressions to filter keywords or unlawful phrases, is a traditional SQL injection protection mechanism. Although this approach does not exclude keywords and strings that are not on the blacklist, web applications are still exposed to new SQL injection attacks. The findings of earlier studies do not give complete safety.

4. Proposed Work

In order to trace or filter the request from un authentic person, proposed work has considered the LSTM model for training. The dataset trained is considering the records that are presenting sender, receiver details along with success and failure. If someone is continuously, attacking on system and this transaction have failed frequently. Then this system detects such transaction and restrict same user from any type of access. This trained model is increasing security by building a mechanism that enable authentication of transaction from user side.

The proposed frame work handles detection and prevention of both SQL injection and Cross site scripting using LSTM, Block Chain etc methods

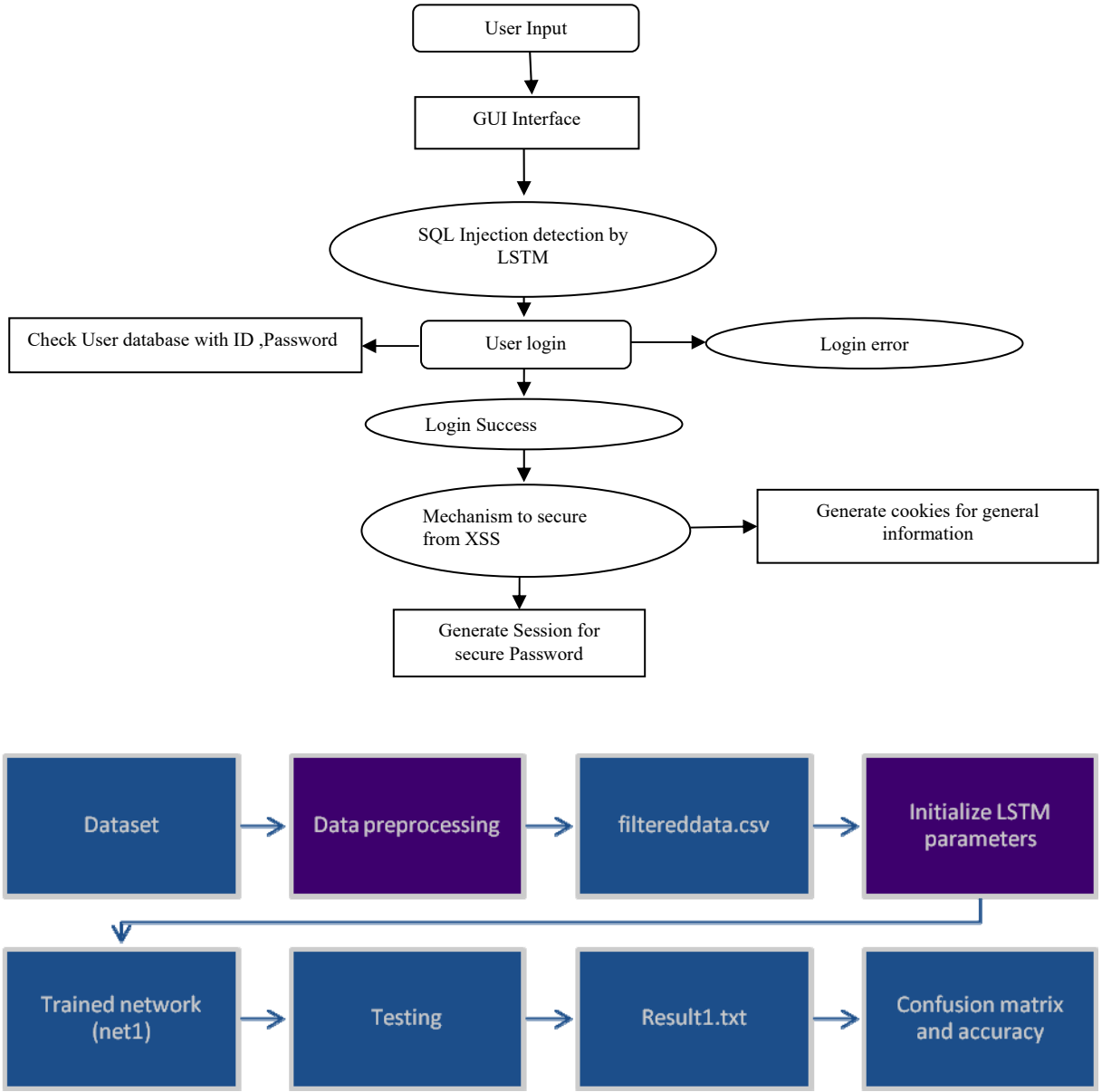

Fig 3 LSTM Trained model

Summarized process flow of proposed work

- Initially, the dataset of transaction dataset is considered for training. Train and test of dataset are performed using traditional methods to get the error values.
- After getting the error values and naive classifier is applied in order to find the accuracy by considering the cases of exact difference (ed) and negligible difference (nd).
- Then a comparison of the accuracy in the case of (ed) and (nd) is made. If Accuracy (ed) is greater than Accuracy of (nd) then the data set is filtered considering ed otherwise dataset is filtered considering nd.
- Training of filtered dataset is made using LSTM model where epoch, batch size, hidden layer is considered to improve the accuracy. After getting the trained model, the testing module is passed with a dataset of transaction records to get predictions for 3 classes. KNN classifier is used to classify the result of 3 classes and present the confusion matrix. Here class 1 is presenting highly authentic IP, while class 2 is presenting system that requires authentication, class 3 is presenting restricted IP.
- Finally, the f1score, accuracy, recall value, and precision are calculated using a confusion matrix. These parameters are compared.

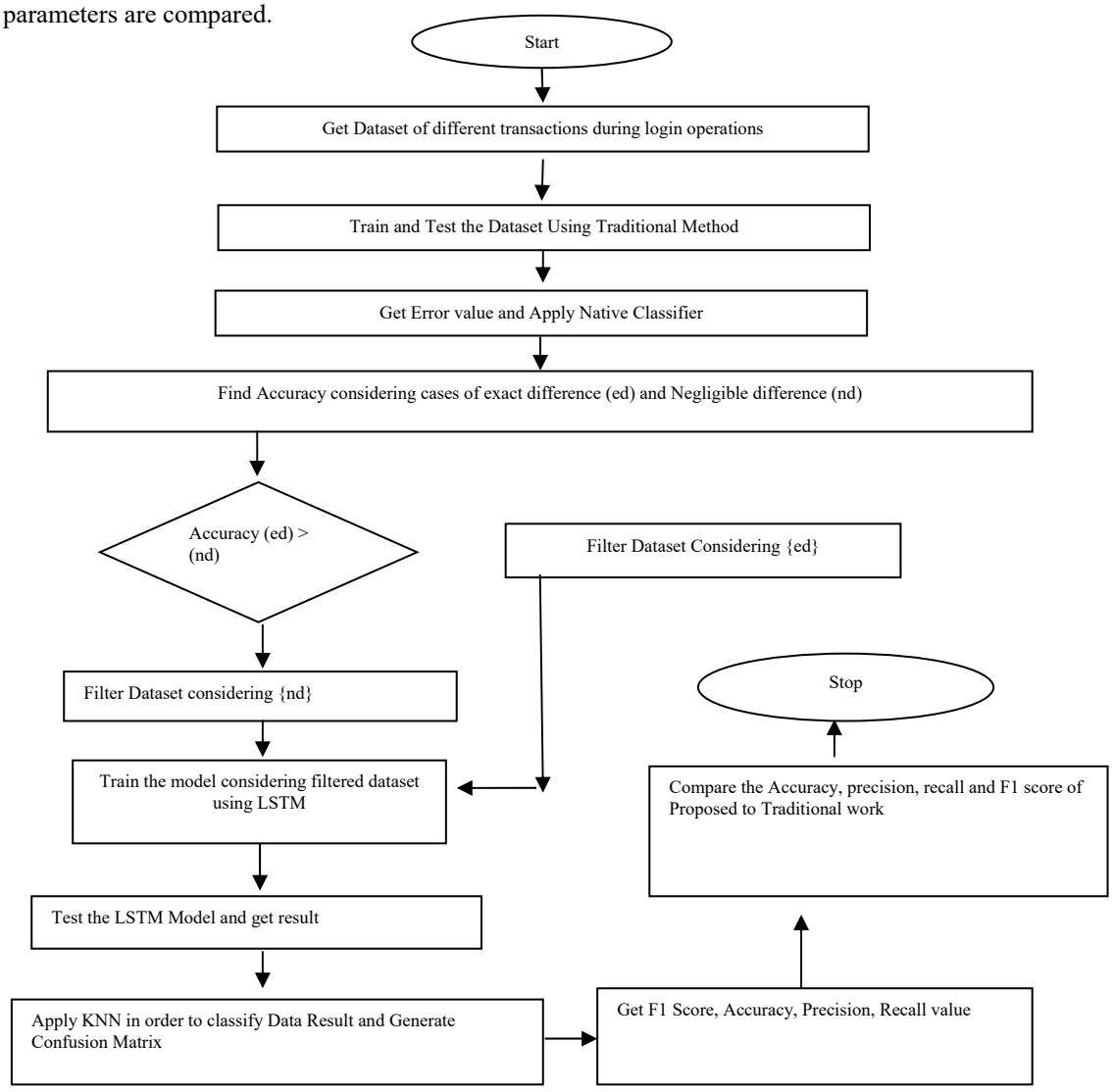

Fig 4 Flow chart of proposed work

5. Result & Discussion

5.1. *Simulation result for Traditional model*

Here during simulation dataset of 1530 records has been considered for training. And testing has been performed on 510 records. Where 411 records belongs to class 1, 426 records belongs to class 2, and 432 records belong to class 3. After simulation out of 1530 records 1269 predication came true while 261 came false.

Table 1 Error and Accuracy In Previous Work

True Value	1269
False Value	261
Total	1530
Accuracy	82.94%
Error	17.04%

Table 2 Confusion Matrix in Case Of Previous Work after Simulation

	1	**2**	**3**	**TOTAL**
1	411	23	76	510
2	33	426	51	510
3	27	51	432	510
	471	500	559	1530

The accuracy, precision, recall and fscore chart on the above confusion matrix is shown below. The overall accuracy is 82.94%.

Table 3 Calculation of Accuracy, Precision, Recall, F1 Score in Case of Previous Work

Class	**n(truth)**	**n(classified)**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
1	471	510	89.61%	0.81	0.87	0.84
2	500	510	89.67%	0.84	0.85	0.84
3	559	510	86.6%	0.85	077	0.81

5.2. *Simulation result for proposed model*

Here during simulation dataset of 1530 records has been considered for training. And testing has been performed on 1530 records. Where 300 records belongs to grade 1, 253 records belongs to grade 2, 357 records belong to grade 3, 712 records where from grade 4 while 1378 records where belonging to grade 5. After simulation out of 3000 records 2608 predication came true while 392 came false.

Table 4 Error and Accuracy in Proposed Work

True Value	1386
False Value	144
Total	1530
Accuracy	90.59%
Error	9.41%

Table 5 Confusion Matrix In Case Of Proposed Work after Simulation

	1	**2**	**3**	**TOTAL**
1	453	13	44	510
2	16	472	22	510
3	18	31	461	510
TOTAL	487	516	527	1530

The accuracy, precision, recall and fscore chart on the above confusion matrix is shown below. The overall accuracy is 90.59%.

Table 6 Calculation of Accuracy, Precision, Recall, F1 Score in Case of Proposed Work

Class	**n(truth)**	**n(classified)**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
1	487	510	94.05%	0.89	0.93	0.91
2	516	510	94.64%	0.93	0.91	0.92
3	527	510	92.48%	0.90	0.87	0.89

5.3. *Comparative Analysis*

Accuracy is a measure of the degree to which the findings are accurate, and is calculated using a confusion matrix. Recall is a measure of how accurate a classifier is in predicting the correct answer. A measurement's precision is determined by calculating its accuracy. Accuracy is measured by the test's F1 score, or F measure. The accuracy of the existing model is compared to the traditional work.

Table 7 Comparison of accuracy in both cases

TRADITIONAL ACCURACY	**PROPOSED ACCURACY**
89.61%	94.05%
89.67%	94.64%
86.6%	92.48%

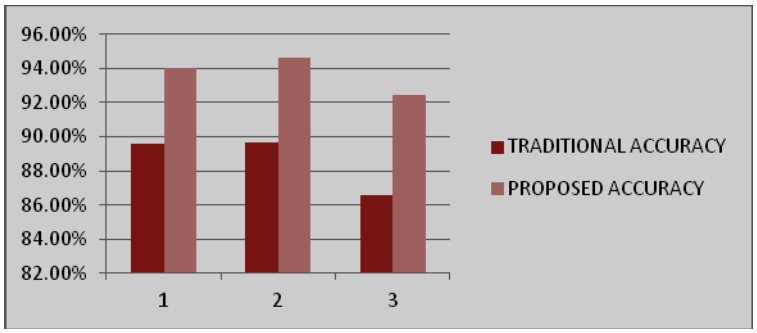

Fig 5 Comparison of accuracy in both cases

The precision of the existing model is compared to the traditional work.

Table 8 Comparison of precision

TRADITIONAL PRECISION	PROPOSED PRECISION
0.81	0.89
0.84	0.93
0.85	0.90

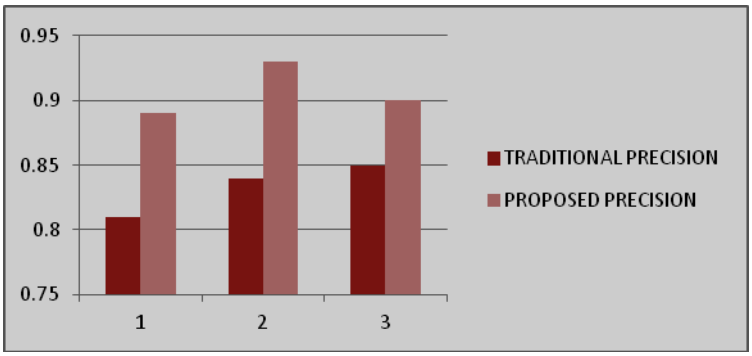

Fig 6 Comparison of precision

The recall of the existing model is compared to the traditional work.

Table 9 Recall Value

TRADITIONAL RECALL	PROPOSED RECALL
0.87	0.93
0.85	0.91
0.77	0.87

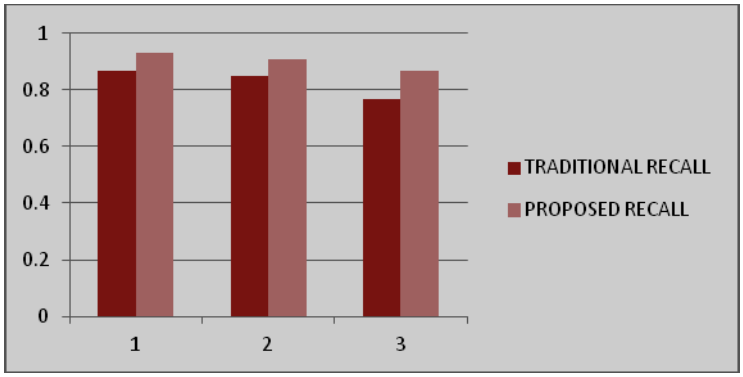

Fig 7 Recall Value

The f1 score values of the existing model are compared to the traditional work.

Table 10 F1 Score

TRADITIONAL F1 SCORE	PROPOSED F1 SCORE
0.84	0.91
0.84	0.92
0.81	0.89

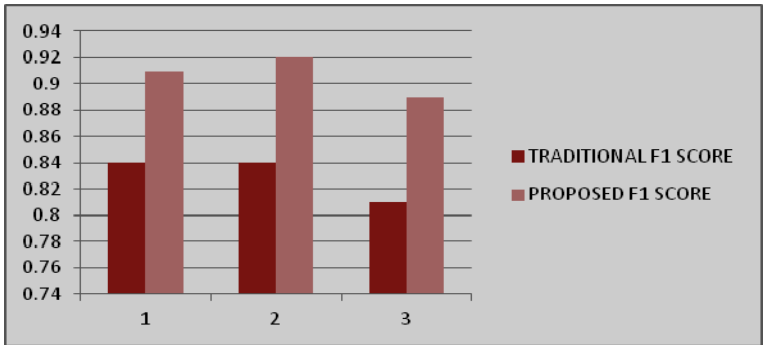

Fig 8 F1 Score

6. Conclusion

There are many advantages of LSTM. For SQL statements, LSTM takes use of how characters are arranged in a string and actively learns the associations of characters before and after based on the structure of the string. For natural language processing, this feature uses the LSTM to infer an expression's emotional tone from its context. To detect threats, there is still a tremendous deal of promise in using LSTM. The LSTM-based model for SQL injection detection transforms URLs into vectors, which it then feeds into the model for training. The LSTM model can be used for other sorts of Web attacks, such as XSS (cross-site scripting attacks) or phishing sites, as long as the URL text or HTTP request text has unique properties. It has a high degree of scalability because it is able to learn the properties of a wide range of threats.

7. Future Scope

The advantages of LSTM are unmatched. By taking advantage of how the string is structured and learning how characters are linked to each other, LSTM can identify SQL queries from conventional statements. When it comes to natural language processing, LSTM can even discern a statement's empathetic tone by studying its context. The threat intelligence detection capabilities of LSTM are still very promising. As part of the SQL injection detection model, LSTM employs vector input from the URL as a training data set. It is possible to use the LSTM model to detect XSS (cross-site scripting assaults) and phishing sites, provided that the URL or HTTP request text has some distinctive features. If we have enough data to train, it can learn the characteristics of many threats. The scalability of this system needs to be studied to understand how the whole integrated system works in a real-time environment. This work is deployable in a production environment where multiple packet sniffers can be set in place to capture packets as local data stores and engage in the whole process. This deployment can be done and real-time evaluation can also be done to enhance the credibility of this work. In other words, LSTM has unequalled advantages. LSTM can distinguish between SQL queries and ordinary statements by analyzing the structure of the string and the relationships between characters. LSTM may even determine a statement's empathic tone by evaluating its context when it comes to natural language processing. LSTM's threat intelligence detection is still highly promising. The URL is used as a training data set for LSTM's SQL injection detection model. Because of the LSTM model's unique properties, it is possible to utilize it to detect XSS (cross-site scripting assaults) and phishing sites. It is possible to train it to recognize many different types of dangers if we have adequate data. It is necessary to investigate the system's scalability in order to learn how the entire integrated system functions in real time. Multiple packet sniffers can be deployed to collect packets as local data stores and participate in the entire process in a real-world production environment, as this work has been demonstrated. A real-time review of this work can be done in order to increase the credibility of this deployment.

References

[1] Joshi Padma, Dr.N.Ravishankar,Dr. M.B. Raju,N.Ch.SaiVyuha"Secure Software Immune receptors from Sql injection and Cross site scripting attacks in Content delivery Network Web applications**"** 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO 2021)DOI: 10.1109/ICRITO51393.2021,Sept. 2021
[2] Joshi Padma, Dr.N.Ravishankar, Dr. M.B. Raju, N.Ch.SaiVyuha "Defensive Walls for Detecting and Preventing SQL Injection and XSS attacks in Dynamic Content Delivery Network Web Applications" Design Engineering (Toronto) ,vol.2021,issue 7,2021,pp10019-10039
[3] Joshi Padma, Dr.N.Ravishankar,Dr. M.B. Raju, N.CH.Ravi(2018) "Encountering SQL Injection in Web Applications" Proceedings of the Second International IEEE Conference on Computing Methodologies and Communication.
[4] N.CH.Ravi,Joshi Padma,etal., "Inspecting Access Controls in Cloud Based Web Application" Proceedings of the Second International IEEE Conference on Computing Methodologies and Communication.2018
[5] Joshi Padma, Dr.N.Ravishankar,Dr. M.B. Raju, N.CH.Ravi(2017) "Contemplating Security of Http From Sql Injection and Cross Script" 2017 IEEE International Conference on Computational Intelligence and Computing Research
[6] A. K. S.S., "SQL Injection Detection Using Machine Learning," Rev. Gestão Inovação e Tecnol., vol. 11, no. 3, pp. 300–310, 2021, doi: 10.47059/revistageintec.v11i3.1939.
[7] T. Y. Kim and S. B. Cho, "Optimizing CNN-LSTM neural networks with PSO for anomalous query access control," Neurocomputing, vol. 456, pp. 666–677, 2021, doi: 10.1016/j.neucom.2020.07.154.
[8] M. M. Hassan, R. B. Ahmad, and T. Ghosh, "Sql injection vulnerability detection using deep learning: A feature-based approach," Indones. J. Electr. Eng. Informatics, vol. 9, no. 3, pp. 702–718, 2021, doi: 10.52549/.v9i3.3131.
[9] H. M. Harb, D. Eleyan, and A. Eleyan, "SQL Injection Detection Tools Advantages and Drawbacks," Int. J. Wirel. Microw. Technol., vol. 11, no. 3, pp. 16–21, 2021, doi: 10.5815/ijwmt.2021.03.03.
[10] T. Zheng, M. Wang, Y. Guo, and Z. Wang, "The Bidirectional Information Fusion Using an Improved LSTM Model," Mob. Inf. Syst., vol. 2021, 2021, doi: 10.1155/2021/5595898.
[11] X. Song, C. Chen, B. Cui, and J. Fu, "Malicious javascript detection based on bidirectional LSTM model," Appl. Sci., vol. 10, no. 10, 2020, doi: 10.3390/app10103440.
[12] S. O. Uwagbole, W. J. Buchanan, and L. Fan, "Applied Machine Learning predictive analytics to SQL Injection Attack detection and prevention," Proc. IM 2017 - 2017 IFIP/IEEE Int. Symp. Integr. Netw. Serv. Manag., vol. 07, no. 02, pp. 1087–1090, 2017, doi: 10.23919/INM.2017.7987433.
[13] P. Tang, W. Qiu, Z. Huang, H. Lian, and G. Liu, "Detection of SQL injection based on artificial neural network," Knowledge-Based Syst., vol. 190, p. 105528, 2020, doi: 10.1016/j.knosys.2020.105528.
[14] I. Jemal, O. Cheikhrouhou, H. Hamam, and A. Mahfoudhi, "SQL Injection Attack Detection and Prevention Techniques Using Machine Learning," Int. J. Appl. Eng. Res., vol. 15, no. 6, pp. 569–580, 2020, [Online]. Available: http://www.ripublication.com.
[15] Q. Li, W. Li, J. Wang, and M. Cheng, "A SQL Injection Detection Method Based on Adaptive Deep Forest," IEEE Access, vol. 7, pp. 145385–145394, 2019, doi: 10.1109/ACCESS.2019.2944951.
[16] 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, 2019, doi: 10.1109/TVT.2019.2893675.
[17] M. Amin et al., "Review of SQL Injection : Problems and Prevention," Int. J. Informatics Vis., vol. 2, pp. 215–219, 2018.
[18] C. C. Tarelli, "Omissions, additions, and conflations in the chester beatty papyrus," J. Theol. Stud., vol. os-XL, no. 4, pp. 382–387, 1939, doi: 10.1093/jts/os-XL.4.382.
[19] S. Natarajan, "Available Online through CODEN : IJPTFI Research Article," vol. 8, no. January, pp. 25990–25994, 2017.
[20] N. M. Sheykhkanloo, "Employing Neural Networks for the detection of SQL injection attack," ACM Int. Conf. Proceeding Ser., vol. 2014-September, pp. 318–323, 2014, doi: 10.1145/2659651.2659675.
[21] S. M. S. Sajjadi and B. Tajalli Pour, "Study of SQL Injection Attacks and Countermeasures," Int. J. Comput. Commun. Eng., vol. 2, no. 5, pp. 539–542, 2013, doi: 10.7763/ijcce.2013.v2.244.
[22] R. Garde, D. R Anekar, N. Kulkarni, and M. Ghadge, "A System to Detect and Block SQL Injection with the help of Multi Agent System using Artificial Neural Network," Int. J. Comput. Appl., vol. 71, no. 12, pp. 21–26, 2013, doi: 10.5120/12411-9073.
[23] A. Joshi, V. Geetha, SQL Injection detection using machine learning, in: International Conference on Control, 2014.
[24] K. Kamtuo, C. Soomlek, Machine Learning for SQL injection prevention on server-side scripting, in: Computer Science & Engineering Conference, 2017.
[25] X.R. Wu, P.P.K. Chan, SQL injection attacks detection in adversarial environments by k-centers, in: International Conference on Machine Learning & Cybernetics, 2012.
[26] B.D. Priyaa, M.I. Devi, Fragmented query parse tree based SQL injection detection system for web applications, International Conference on Computing Technologies & Intelligent Data Engineering, 2016.
[27] Y. Lecun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436.
[28] Y. Wang, W.D. Cai, P.C. Wei, A deep learning approach for detecting malicious JavaScript code, Secure. Communication. Network. 9 (11) (2016) 1520–1534.
[29] P. Sirinam, M. Imani, M. Juarez, M. Wright, Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning, 2018.
[30] G. Yuan, L. Bo, Y. Yao, S. Zhang, A deep learning enabled subspace spectral ensemble clustering approach for web anomaly detection, in: International Joint Conference on Neural Networks, 2017.
[31] N.CH.Ravi, Joshi Padma etal. "ADVANCED ACCESS CONTROL MECHANISM FOR CLOUD BASED E-WALLET" in Volume 31 of the Lecture Notes on Data Engineering and Communications Technologies series of Springer of Proceeding of the International Conference on Computer Networks, Big Data and IoT (ICCBI - 2018)

Authors Profile

	Joshi Padma N is Associate Professor in Computer Science and Engineering Department in Sreyas Institute of Engineering and Technology, Nagole, Hyderabad in Telangana, India. She worked as Head of department for CSE in Sreyas for 2.5 years. She did B.E in CSE, M.Tech CSE and pursuing Ph.D from JNTUH has 19 years teaching experience and Consultant for projects in I.T industry. She has publications in Springer book chapter and IEEE. Her research interest areas are Web and Network Security, Deep learning, Block Chain Technology and Cognitive Security. She has strong logical and programming skills in JAVA, Adv.Java, Spring boot, REST API, Angular and Dev ops. She did awareness Programs in Mobile, Internet security in Rural areas, Villages in Schools and Colleges.
	Dr. N. Ravishankar is working as Professor in Dept of CSE *in* Geethanjali College Of Engineering and Technology, Keesara, Hyderabad.He is also Controller of Examinations in GCET. He has vast experience in academics as well as administration. He previously worked as HoD in CSE in Sreenidhi Institute of Science and Technology, KMIT and Lakki Reddy Bal Reddy Engineering College. He has publications in IEEE, Springer. His research Interest is Network and Information Security.
	Dr.M.B.Raju completed his B. E from Osmania University, M. Tech & Ph.D from JNTUH, Hyderabad. He has vast experience of 28yrs in academics as well as administration. He has authored 20 National and International research papers, 70 Journals, 02 books and also owns a patent.His research interest areas are Network security, Image processing, Machine and Deep learning and Data mining.
	N.Ch.Ravi is Associate Professor in Computer Science and Engineering Department in Pallavi Engineering College, Nagole, Hyderabad in Telangana, India. He is also Dean R&D with 20 years experience in which 5 years in I.T Industry and 15 years in teaching. He has publications in Springer book chapter and IEEE. His research interest areas are Network Security, Deep learning, Block Chain Technology and Cognitive Security. He has Logical, Programming abilities in Advanced JAVA, Spring boot, REST API, Micro services, Angular, Devops, Secure software Programming.