Advanced LSTM Attention with Hybrid Framework for Detection and Prevention of SQLI and XSS Attacks

Mrs. 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
An attacker can use SQL injection, which is a web security vulnerability, to tamper with an online application's database queries. In most circumstances, it grants an attacker access to data that they would not otherwise have. Cross-Site Scripting (XSS) assaults are an injection technique in which malicious scripts are injected into websites that are otherwise innocent and trustworthy. The use of an online application by an attacker to deliver malicious code, often in the form of a browser side script, to a separate end user is known as cross-site scripting (XSS). Present research has LSTM attention to detect Sql injection, XSS and prevention using Multiplicative Inverse, Filtering, Session Sanitization and WAF using Block Chain technology. The results like accuracy, f-score, recall value and precision are also evaluated.

Keywords: *Block chain, Multiplicative inverse, LSTM Attention, Accuracy Parameter, SQL Injection, Xss attacks, WAF*

1. INTRODUCTION

A block chain supports immutable ledgers, or records of transactions that cannot be changed, deleted, or destroyed. The largest challenge is the quantity of data a block chain can contain. This is either because to the protocol's size restriction or the expensive transaction fees. It is keeping only the data hash in the block chain is one way to get the benefits of a block chain without paying transaction fees. A hash is a string formed using the data we supply. A Phisher can exploit SQL injection, a web security flaw, to manipulate database queries in an online application. In most cases, it gives an attacker access to data they otherwise wouldn't have. Cross-Site Scripting (XSS) attacks insert malicious scripts into otherwise innocent and trustworthy websites.

1.1 BLOCKCHAIN

The shared database known as a block chain differs from conventional databases in that data is stored in blocks that are then connected using encryption. As new information is received, it is added to a new block. A data chain is created once the block has been filled with data and is connected onto the one before it. Although a block chain can hold a variety of data, a transaction ledger is now its most popular application. Block chain is employed in a decentralized manner in the case of Bit coin, which means that all users have collective control rather than a single person or organization.

The goal of block chain is to make it possible to record and distribute digital data without the capacity to alter it. In this way, a block chain is the foundation for immutable ledgers, or transaction records that cannot be modified, erased, or destroyed. Distributed ledger technology (DLT) is another term for block chains (DLT).

1.1.1 Storage on Block chain

The amount of data that can be stored on a block chain is the biggest issue. This is either because of the protocol's limit on the amount you can send or because of the exorbitant transaction costs you'd have to pay. We have a certain amount of data that we can store. To give you an idea, most chains accept a file size of a few kilobytes or less.

We might theoretically get around this limitation by dividing our data into several little bits. This would allow us to store greater files/data, but it would also boost our prices dramatically. This is because we would have to pay the transaction's (high) base price numerous times.

Only keeping the hash of the data in the block chain is one approach to receive the benefits of a block chain without paying a fortune for transactions. A hash is a created string that is computed using the information we provide. The output hash will always be the same if the input is the same. Other input yields a different hash. We can tell if our data has been updated simply by glancing at the hash.

The hash of our data is the only item we save on the block chain using standard storage mechanisms. The hash is quite small in compared to our data; hence the cost of a transaction is negligible. We can save the raw data in whatever way we wish. We could, for example, employ a relational database or simply a file system.

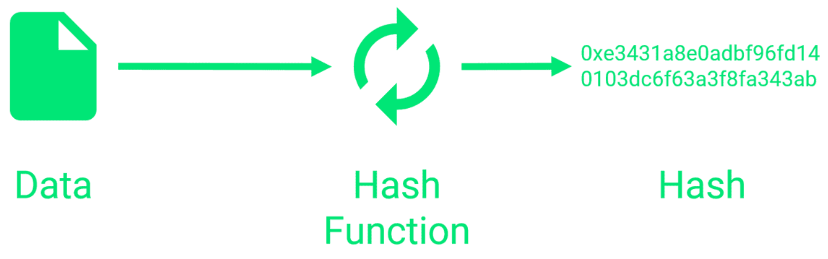


Figure 1.Block chain

All we have to do now is make sure that we give the block chain transaction's id (hash) to our raw data. We'd add another column to a relational database to store the transaction id.

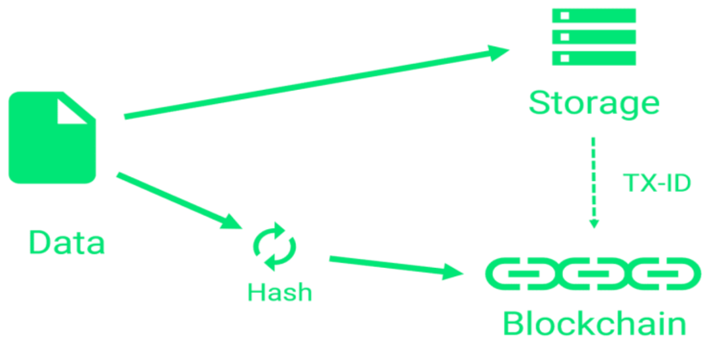


Figure 2.Storage on Block chain

That way, we may take advantage of traditional storage techniques (such as searches) while yet benefiting from the block chain’s tamper-proofing. We can hash the raw data and compare it to the hash in the allocated transaction in the block chain at any point when there is a doubt regarding the data. Benefits like as decentralization and transparency may be lost depending on your storage strategy.

1.2 SQL INJECTION

A web security flaw called SQL injection allows phishers to interfere with database queries used by online operations. In utmost circumstances, it grants an attacker access to data that they would not else have. Your particular information, as well as that of other clients, may be accessible to the app. An attacker might be suitable to change or cancel this information, affecting the app's operation or content for a long time. The underpinning structure of a server, as well as other backend systems, can be compromised through denial- of- service attacks or SQL injections. An attacker could pierce any sensitive data that has been compromised as a result of a successful SQL injection attack. SQL injection attacks have lately been linked to a number of high profile data breaches, performing in public embarrassment and nonsupervisory penalties. An attacker with a long- term backdoor into an association's systems may be suitable to go undetected for a long time. SQL injection faults, attacks, and ways of all kinds can be used in a variety of ways. The following are a few common examples of SQL injection:

• By changing a SQL query to obtain more results, hidden data can be retrieved.
• By subverting the logic of an application, you can amend a query and change the program's reasoning.
• A UNION attack allows you to simultaneously access data from multiple database tables.
• You'll find out more about the database's version and structure in this phase.
• "Blind SQL injection" occurs when you control the query results that are returned in the application's responses.

1.2.1 SQL injection detection using LSTMs

The following are the steps for LSTM-based SQL injection detection: Labeling, normalizing, word segmentation and embedding word vectors are a few of the preprocessing steps. We assign the number 1 to SQL injection samples and the number 0 to negative samples based on the categorization of the data. In order to recognize and decode various encodings like base64 encoding and url encoding, all English letters in example files have been converted to lowercase. Based on your existing information, construct a word segmentation model using the symbols "(,"")", "", and "%." Word Embedding is applied to the segmentation findings, and the resulting numerical word vectors serve as each word's representation. Sentences of varying durations are transmitted into the LSTM layer, which runs them via a transformation algorithm to create hidden LSTM neural unit vectors. A vector h can be derived once these vectors have passed through the mean pooling layer. Lastly, there is SoftMax. To choose the final prediction category, we may obtain a class distribution vector and then look for the output category with the highest probability.
Hyper-parameters are adjusted as necessary once the model has been trained using training data. The most recent epoch began 30,000 years ago. 3000 training runs later, the model is developed. The primary measures used to assess the system are accuracy and F1-score. The best model from the output models is used as a classifier to identify SQL injection attacks after training. After normalization, word segmentation, and embedding word vectoring, the test data is fed into the classifier and evaluated. If not, the user input is probably a statement from a SQL injection attack. If the classifier's result is 0.

1.3 XSS ATTACKS

Cross-Site Scripting (XSS) attacks are fundamentally an injection technique in which malicious scripts are introduced into otherwise trustworthy and innocent websites. Cross-site scripting is the exploitation of an online application by an attacker to send malicious code, frequently in the form of a browser side script, to a different end user (XSS). Cross-site scripting flaws often allow an attacker to assume the identity of a victim user, do any tasks that the victim is able to complete, and access any of the victim's data. The attacker may be able to gain total control over all of the functionality and data of the application, depending on whether or not the target user has privileged access inside the programme.
Cross-site scripting is done by altering a vulnerable website so that it displays malicious JavaScript to users. The attacker has total control over how the victim interacts with the software in question when malicious code is run inside the victim's browser
XSS attacks may be classified into three categories. These are the ones:

In Reflected XSS, when XSS occurs through the active HTTP request, the malicious script is returned to the attacker

In XSS Stored, the malicious script is retrieved from the website's database and stored as an XSS in the database.

Client-side cross-site scripting, often known as DOM-based XSS, is a vulnerability that affects client-side code rather than server-side code.

An attacker that successfully exploits a cross-site scripting vulnerability is often able to do the following actions:

- Impersonate or pose as the target user
- Any action that the user is capable of performing will be carried out.
- Read any information that the user has permission to see.
- Take note of the user's login details and save them somewhere safe.
- Deface the website with a virtual defacement technique.
- Inject Trojan functionality into a website's source code.

1.3.1 Preventing XSS vulnerabilities

The following steps are often used in conjunction with one another to successfully guard against XSS flaws:

- Incoming data should be filtered. It is important to filter out all non-valid input at the moment of receiving user input.
- The output should be encoded when it is sent out. Encode the output of user-controllable data in HTTP replies to prevent it from being misinterpreted as active content. It's possible that a combination of HTML, URL, JavaScript, and CSS encoding is required, depending on the output environment.
- Use response headers that are relevant. If you don't want browsers to read your HTTP replies in the way you intend, you should use the Content-Type and X-Content-Type-Options headers to protect against XSS in non-HTML/JavaScript responses.
- Security Policy for Content. Finally, you may utilise Content Security Policy (CSP) to mitigate any XSS vulnerabilities that remain.

2. LITERATURE REVIEW

Padma Joshi.N etal discussed block chain method to prevent sql injection,xss by proposing a frame work[1].Ravi,N.Ch. Joshi Padma etal discussed access controls using block chain technology [2]. Padma Joshi.N etal discussed multiplicative inverse method for sql injection and xss prevention[3]. Padma Joshi.N etal proposed LSTM method to detect sql injection [4]. To stop sql injection and XSS attacks, Padma Joshi.N et al recommended pattern locking and session id sanitization[5]. Numerous SQL Injection attacks were covered by Padma Joshi.N et al. in their discussion [6]. A framework for SQL injection and XSS protection was proposed by Padma Joshi.N et al. [7]. NH.Ravi and Joshi An enhanced access control system for a cloud-based e-wallet was proposed by Padma etal. o stop sql injection and XSS attacks, Padma Joshi.N et al recommended pattern locking and session id sanitization[5]. Numerous SQL Injection attacks were covered by Padma Joshi.N et al. in their discussion [6]. A framework for SQL injection and XSS protection was proposed by Padma Joshi.N et al. [7].

NH.Ravi and Joshi An enhanced access control system for a cloud-based e-wallet was proposed by Padma etal. [8]. N.CH.Ravi etal discussed various access controls for prevention of web attacks [9]. Padma Joshi.N etal proposed integrated xss sql injection detection, prevention using Web application Firewall to be inserted in their frame work[10].Various web attacks types of sql injection,XSS attacks discussed[11].SethFogie etal discussed different XSS attacks and their prevention[12]. Justin Clarke-Saletal proposed various Sql injection attacks and their defenses [13]. Prithvi B etal[14] proposed dynamic candidate evaluations approach. A model using a hybrid method, i.e. static analysis and run time analysis, was proposed by Kunal et al. [15]. The Text based Key Generator with four different filtration techniques was proposed by Indrani B.E. Ramaraj[16] as a way to identify and stop SQL Injection Attacks from accessing databases. An adaptive deep forest-based technique was proposed by QI LI et al.[17] to identify complex SQL injection attempts. By connecting the raw feature vector and averaging prior findings at the input of each layer, they first improved the deep forest's structure. A lengthy short-term memory SQL injection attack detection method that can automatically learn the best data format and has a considerable advantage when handling complex high-dimensional large data has been created by Fang Wang and colleagues [18]. The SQLIA detection technique, WOVSQLI, was proposed by Yong Fang et al. [19] and uses an LSTM neural network and a SQL word vector. With a substantial number of SQL query strings gathered from various sources, they create an LSTM neural network model and a model using SQL token vectors. The experiment's findings demonstrate that the suggested strategy performs well in terms of precision and recall. A framework called XSS-SAFE was proposed by Gupta BB et al. [20] and is capable of detecting and mitigating XSS attacks based on automatic feature injection and the insertion of JavaScript sanitization routines into the source code.

Before converting the payloads of XSS attacks to vectors, Raed Waheed Kadhim et al.[21] employed word2vec to extract their semantic features. They then used CNN with LSTM to create classification models. CODDLE[22], a deep learning model with varied evaluation with various parameters, was proposed by Abaimov S et al. The audit for XSS vulnerability was proposed by Li C et al. [23] and is based on a PHP code parsing tool and Bi-LSTM model. To begin with, they tokenized the source-filter-sink triple sequences using the Word2vec model. [24] Fawaz Mahiuob and others For the purpose of detecting web-based XSS assaults, the XGBXSS detection framework is proposed. The detection framework has been shown to be effective in achieving exceptional accuracy and detection rate with low FP and FN rates. A security engine was presented by Bharti Nagpal et al. [25] to thwart SQL Injection, XSS, and CSRF attacks in PHP online applications. For increasingly complicated online applications, this security engine modifies the current integrated method to stop reflected XSS attacks and SQLIA. A hybrid approach query model generator with logic to stop SQL injection attacks and reflected cross-site scripting attacks was developed by Sharma et al. [26].Jun Yang etal.[27] proposed an MLAB-BiLSTM method that employs a multi-layer attention based bidirectional LSTM deep neural network to accurately detect Web threats in real time. To create the encoded representation of the segments in the original request, they employed a bidirectional LSTM model carefully. To improve performance, drawbacks include increasing the number of harmful classes, modifying parameters, and reducing the use of parameters across networks. B. Kranthikumar etal. [28] suggested model uses its pattern checking mechanism to determine whether a SQL query is an injected one or a regular query whenever one is generated from the client side, as demonstrated. Python was used to implement the suggested model and the machine learning models, and the synthesized dataset from GITHUB was used for testing.

3. PROBLEM STATEMENT

In previous researches it has been observed that wild characters are restricted during login operations. But if attacks are repeated again and again then such cases may lead to overburden on server. Moreover, there is threat to information that is stored in dataset. However previous research made use of encryption mechanism to secure the data but there is need to improve the reliability of such systems. There is need to introduce machine learning model that would restrict repeated invalid login. On other hand there is need to introduce block chain mechanism where data is stored in encrypted format using multiplicative inverse.

4. RESEARCH METHODOLOGY

Proposed research has considered the existing work related to sql injection and security mechanism such as encryption and block chain model along with machine learning model. Research considered the security and performance issues in previous researches. Then the hybrid approach has been proposed in order to provide security against sql injection attack by resolving the performance issues faced in previous research works. Finally the LSTM model accuracy has been tested and performance of proposed work is evaluated.

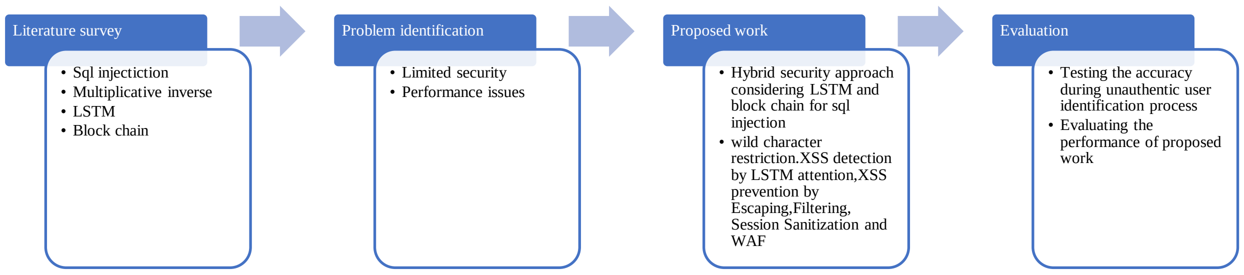

Literature survey
• Sql injectiction
• Multiplicative inverse
• LSTM
• Block chain
Problem identification
• Limited security
• Performance issues
Proposed work
• Hybrid security approach considering LSTM and block chain for sql injection
• wild character restriction.XSS detection by LSTM attention,XSS prevention by Escaping,Filtering, Session Sanitization and WAF
Evaluation
• Testing the accuracy during unauthentic user identification process
• Evaluating the performance of proposed work

Figure 3. Research Methodology

5. PROPOSED WORK

Considering issues faced in previous researches where limited security is provided. Propose work is using LSTM based machine learning model that would restrict repeated invalid login. On

other hand proposed model is making use of on-screen keyboard to restrict wild character entry. Proposed system is also considering block chain mechanism where data is stored in encrypted format using multiplicative inverse. Proposed work has considered block chain for storing data to restrict sql injection. On other hand LSTM model has been used to train the web model to reject the unauthentic access that were restricted previous time due to unauthentic efforts. After completing Sql injection security layer valid log in , XSS detection done with LSTM advanced Attention method with transformers and prevention by Session id sanitization, Escape sequencing, filtering and WAF methods.

5.1. Proposed Novel approach Hybrid model

My Research work has been presented in following model of figure 4. In this, research is considering login panel and getting the input data from user side. Research is getting the IP address and other login details of user. Now, present research is containing the LSTM model to check the authenticity of user. if no, then they reject the request of login. If yes, then it check whether input has been made form on screen, keyboard or digital keyword. If input has been made from keyboard, then get the corresponding details from blockchain. If no, then check wild character used and get he corresponding details from blockchain. Now check the valid login credential and get Authenticate login[1],[3],[4],[6],[7] of my previous research work. After log in the next phase is to detect and prevent XSS attacks by using LSTM advanced Attention method with transformers and prevention by Session id sanitization which discussed in my previous paper[5], Escape sequencing, filtering and WAF methods[10] of my previous work.

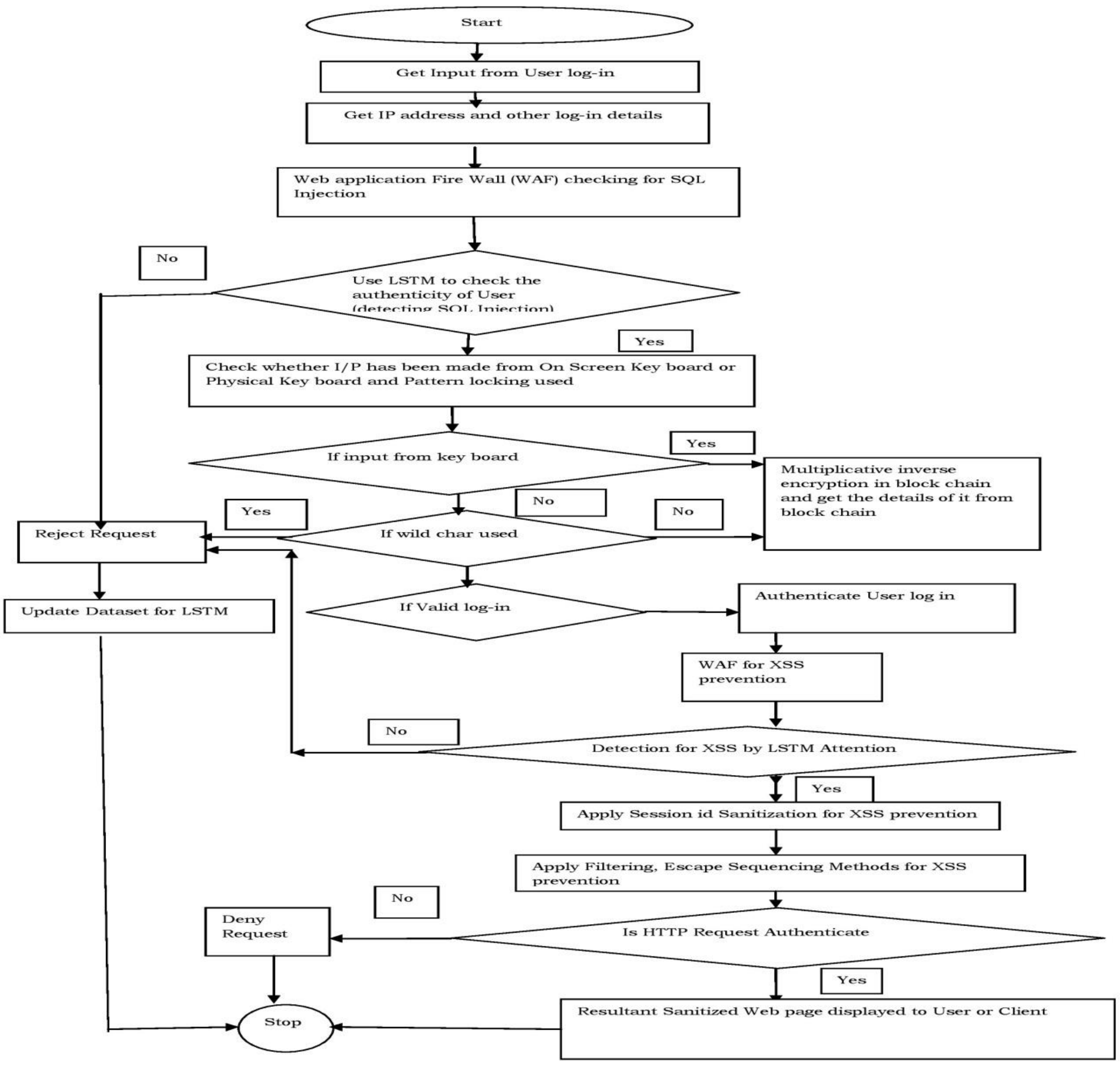

Start
Get Input from User log-in
Get IP address and other log-in details
Web application Fire Wall (WAF) checking for SQL Injection
No
Use LSTM to check the authenticity of User (detecting SQL Injection)
Yes
Check whether I/P has been made from On Screen Key board or Physical Key board and Pattern locking used
If input from key board
Yes
Multiplicative inverse encryption in block chain and get the details of it from block chain
No
Yes
No
Reject Request
If wild char used
Update Dataset for LSTM
If Valid log-in
Authenticate User log in
WAF for XSS prevention
No
Detection for XSS by LSTM Attention
Yes
Apply Session id Sanitization for XSS prevention
Apply Filtering, Escape Sequencing Methods for XSS prevention
No
Deny Request
Is HTTP Request Authenticate
Yes
Stop
Resultant Sanitized Web page displayed to User or Client

Figure 4.Process flow of proposed work

5.2. XSS detection by LSTM Advanced Attention

The LSTM output's hidden layer adds attention, and the fully connected layer transforms the hidden state (hi) into the target attention weight (u I The equation reads as follows: tan (h_i)= u_i

A. Weights for attention probabilities: The softmax function produces the attention probability distribution values a1, a2, a3, a4, and so forth. The equation reads as follows:

$$a_i = \exp(u_i) / \sum_{i=1}^{m} exp(u_i)$$

B weights assigned to attention. Based on a_t and h_t, the context vector v is calculated. The equation reads as follows:

$$v = \sum_{i=1}^{m} a_i . h_i$$

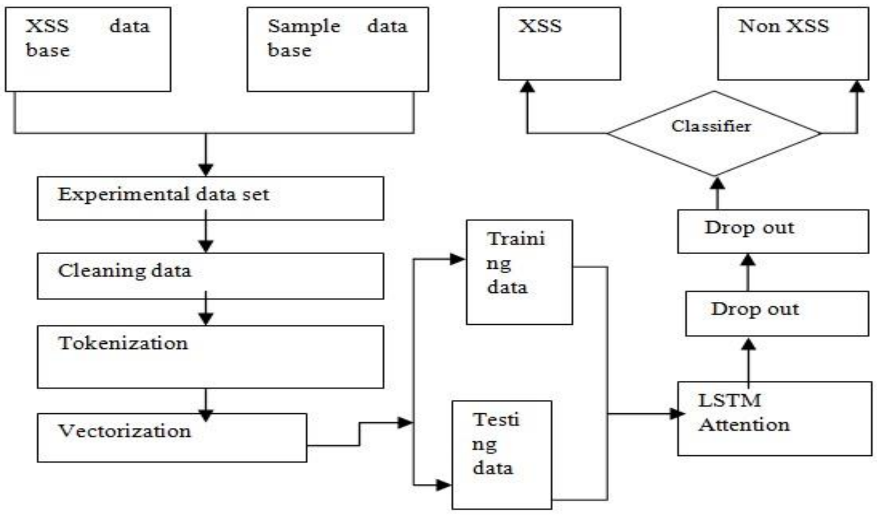


Figure 4.XSS detection by LSTM Attention flow chart

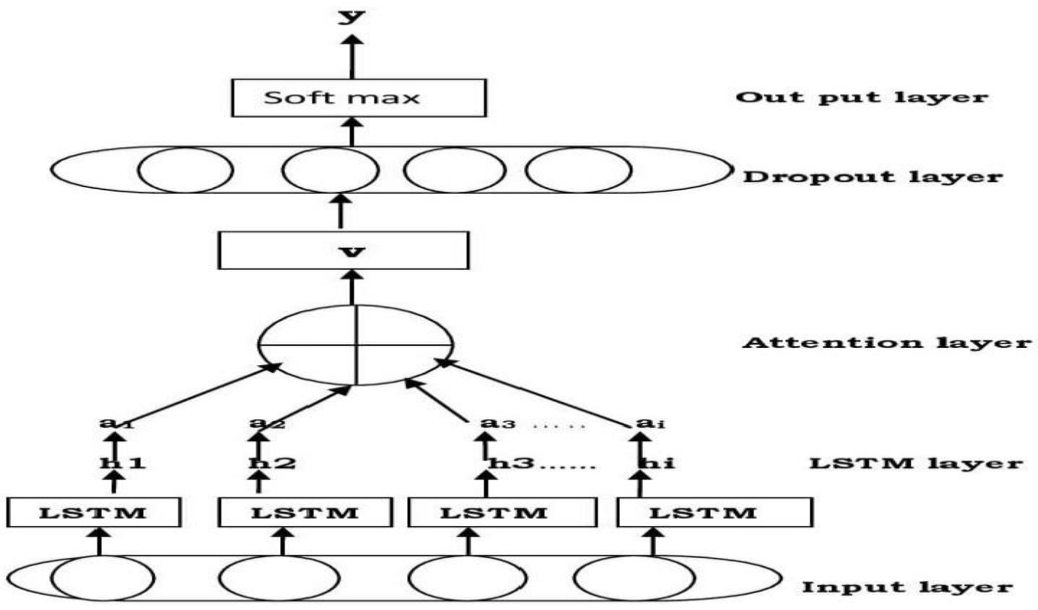


Figure 5.XSS detection model by LSTM awareness

In Entry layer1, there is a sample of experimental data, for instance. After tokenization, the data's content is first represented as the vectors x1 through xi. The embedding layer is no longer used in this part because the data vector matrix embedding training has already been finished in the data pretreatment procedure**.**

In LSTM layer 2, each input can be passed through this layer to the LSTM unit in order to get the output of the associated hidden layer, such as h1, h2, h3,..., hi. It leverages LSTM to address the issue of long-term dependencies and instantly picks up on XSS assaults' abstract features.

The attention probability distribution values a1, a2, a3,..., ai are obtained by applying an attention mechanism to the third layer, which is the hidden layer.

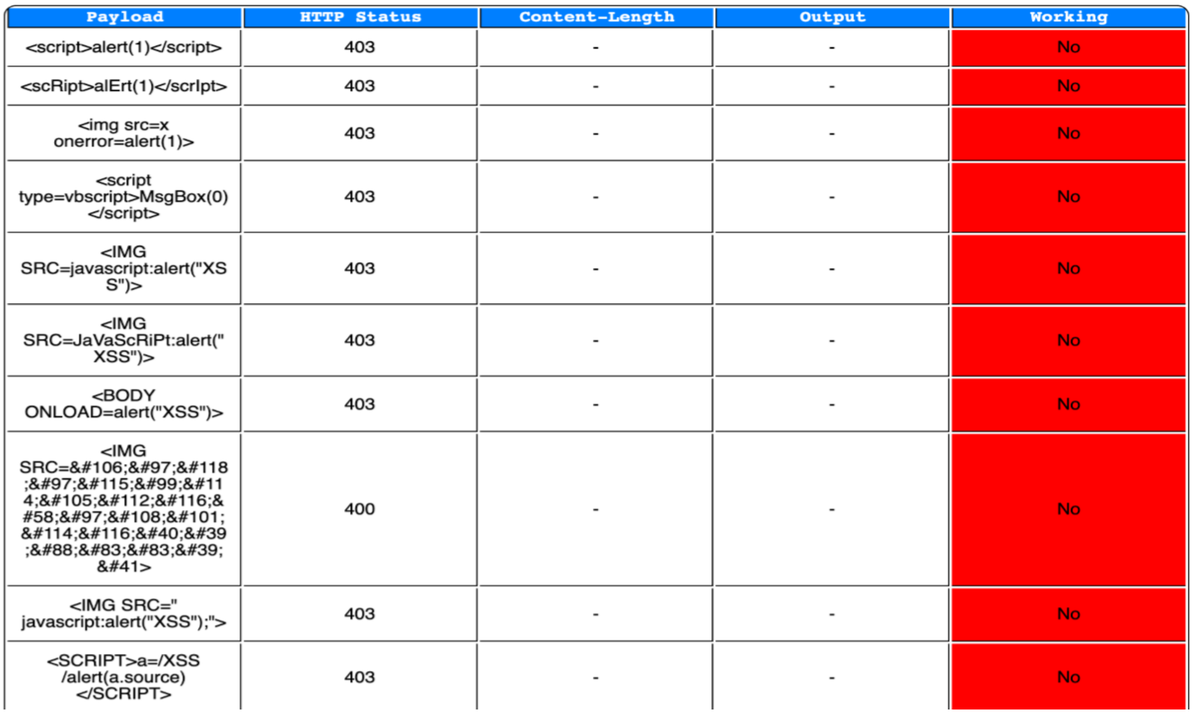
In fourth layer, We employ dropout technique in this layer to stop over fitting.

In the output layer, the softmax classifier outputs the classification result y.

33,300 standard data points are collected for the experiment, and the negative samples are normal samples from the SKKEDU database.79, 263 normal samples in total are collected. In order to train and test neural networks, all the data is finally randomly split into 70% training data and 30% testing data.

5.3. SQL Injection, XSS prevention by WAF

We host the WAF on premise. This will bring on some extra costs as we will need to provide a server for the WAF to run on. This server can either be virtual or physical. Due to the complexity we need several people to host this solution. We need a WAF expert to make sure our WAF is configure properly and we also need a networks engineer to set up our server and make sure that the traffic is routed through the WAF first and also through the firewall still as a WAF does not replace a firewall.



Payload	HTTP Status	Content-Length	Output	Working
<script>alert(1)</script>	403	-	-	No
<scRipt>alErt(1)</scrIpt>	403	-	-	No
<img src=x onerror=alert(1)>	403	-	-	No
<script type=vbscript>MsgBox(0) </script>	403	-	-	No
<IMG SRC=javascript:alert("XSS")>	403	-	-	No
<IMG SRC=JaVaScRiPt:alert("XSS")>	403	-	-	No
<BODY ONLOAD=alert("XSS")>	403	-	-	No
<IMG SRC=&#106;&#97;&#118;&#97;&#115;&#99;&#114;&#105;&#112;&#116;&#58;&#97;&#108;&#101;&#114;&#116;&#40;&#39;&#88;&#83;&#83;&#39;&#41>	400	-	-	No
<IMG SRC=" javascript:alert("XSS");">	403	-	-	No
<SCRIPT>a=/XSS /alert(a.source) </SCRIPT>	403	-	-	No

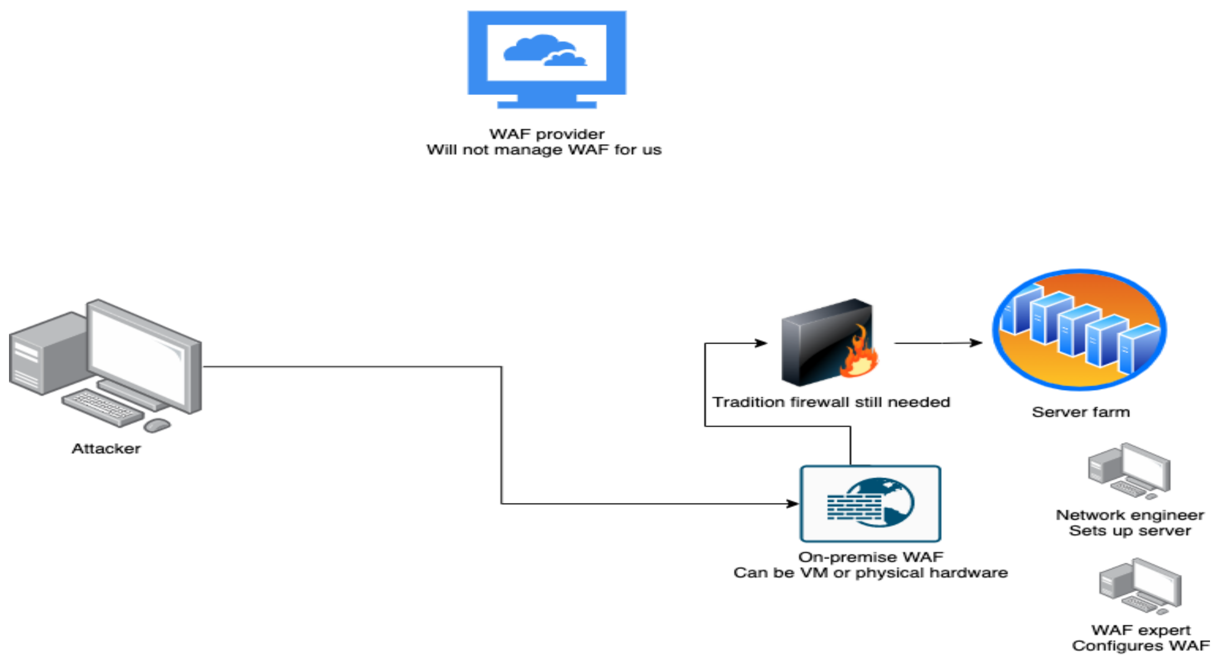


Figure 6.SQL injection, XSS prevention by WAF

6. RESULT & DISCUSSION

A graph has drawn between Number of attacks detected verses detection methods. It is observed that my Hybrid method detects more attacks.

Detection methods	Number of attacks detected
Johari2014	110
BNPal2016	111
lstm2021	115
DEEPXSS2021	114
H_SQLI_XSS	116
Xgbss	113
wovsqli	112

Table 1.No.of attacks detected

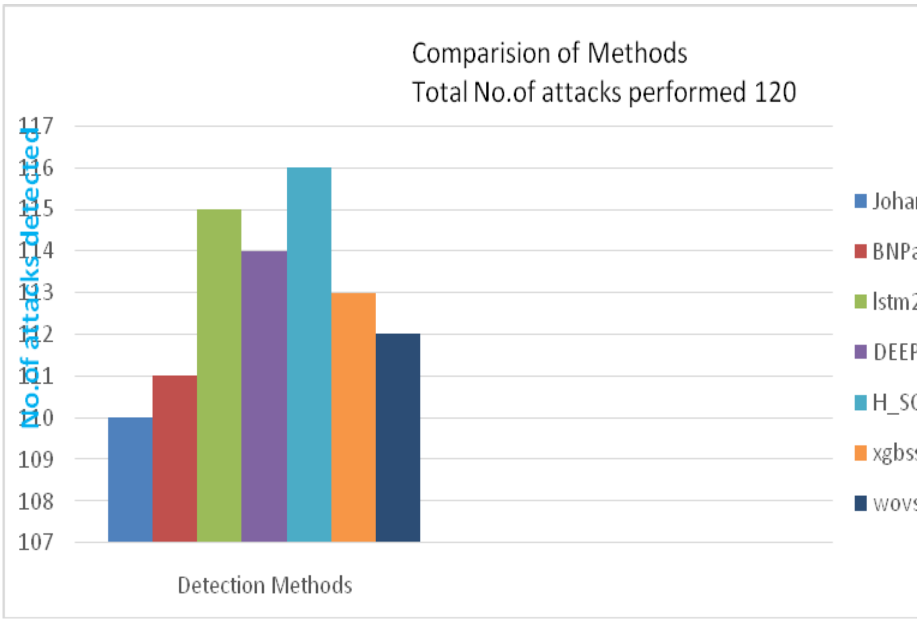


Figure 6 . No.of attacks detected vs Detection methods

Detection methods	Number of attacks Prevented
Johari2014	88
BNPal2016	95
lstm2021	113
DEEPXSS2021	110
H_SQLI_XSS	118
xgbss	109
wovsqli	108

Table 2 .No.of attacks prevented

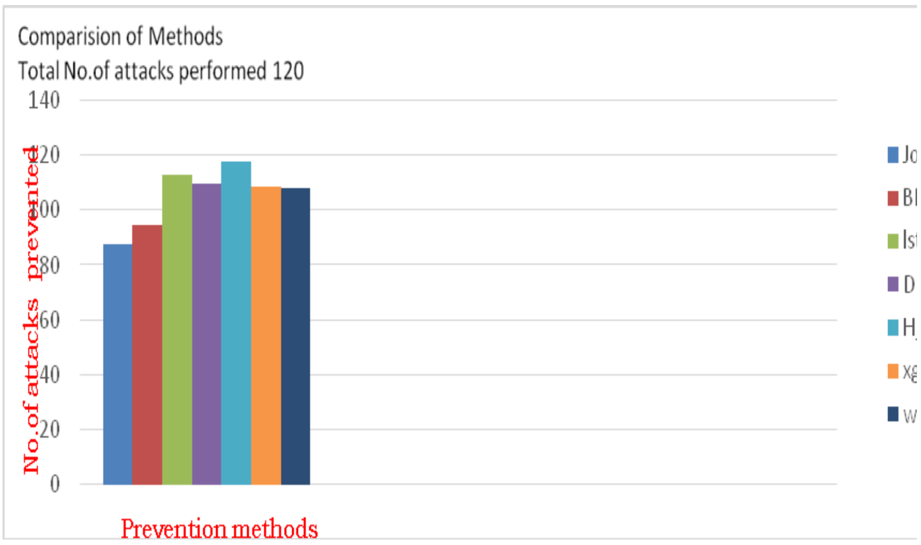


Figure 7 .No.of attacks prevented vs Prevention methods

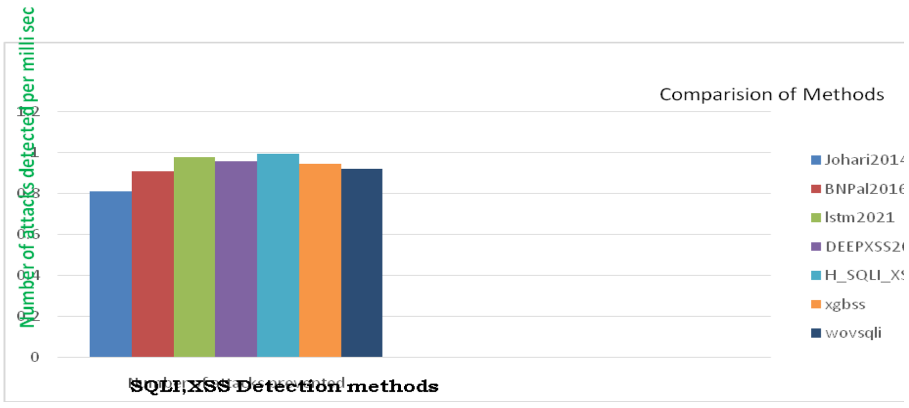


Figure 8. Detection time per millisec vs Detection methods

A graph has drawn between Number of attacks detected per sec verses detection methods. It is observed that my Hybrid method rate of detection is more.

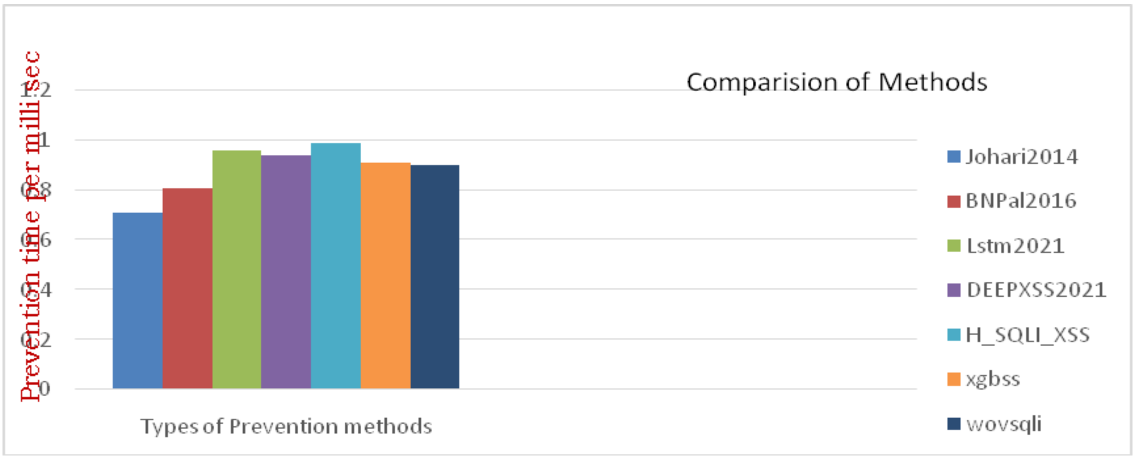


Figure 9. Prevention time per millisec vs Prevention methods

Security overhead defined as Percentage ratio of security operation time like detecting, preventing attack packets to the Transmission time of data packets.

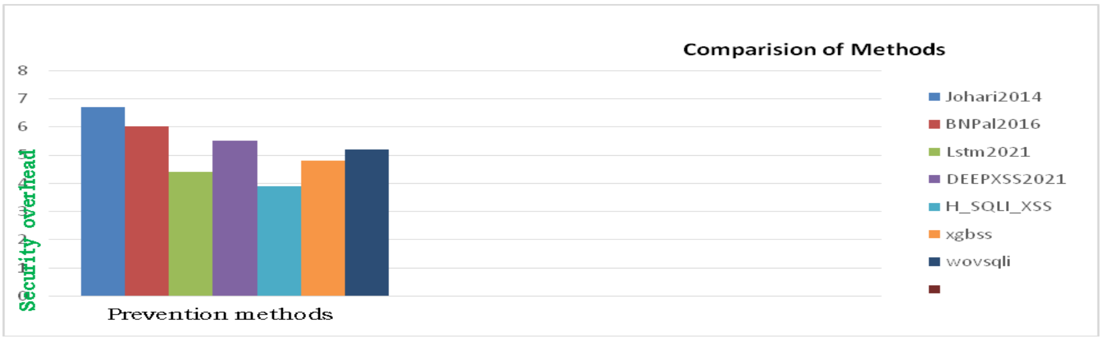


Figure 10. Security overhead vs Prevention methods

It is observed that my Hybrid method security over head value is low. So the resources used and time taken for processing are less.

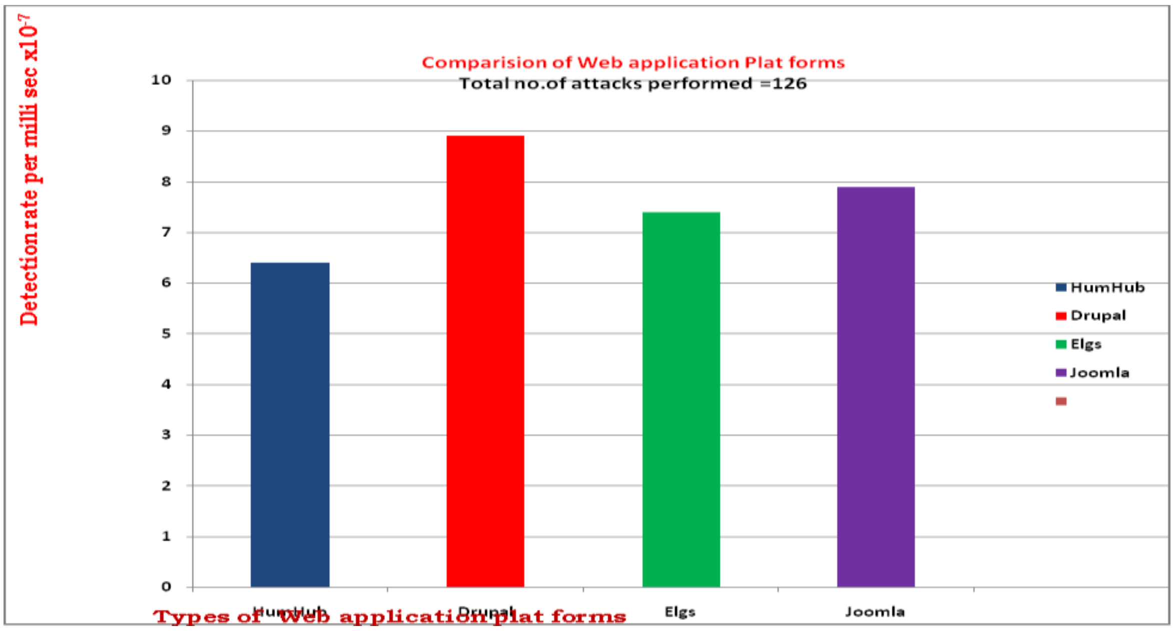


Figure 11. Plat forms vs Detection time per milli sec

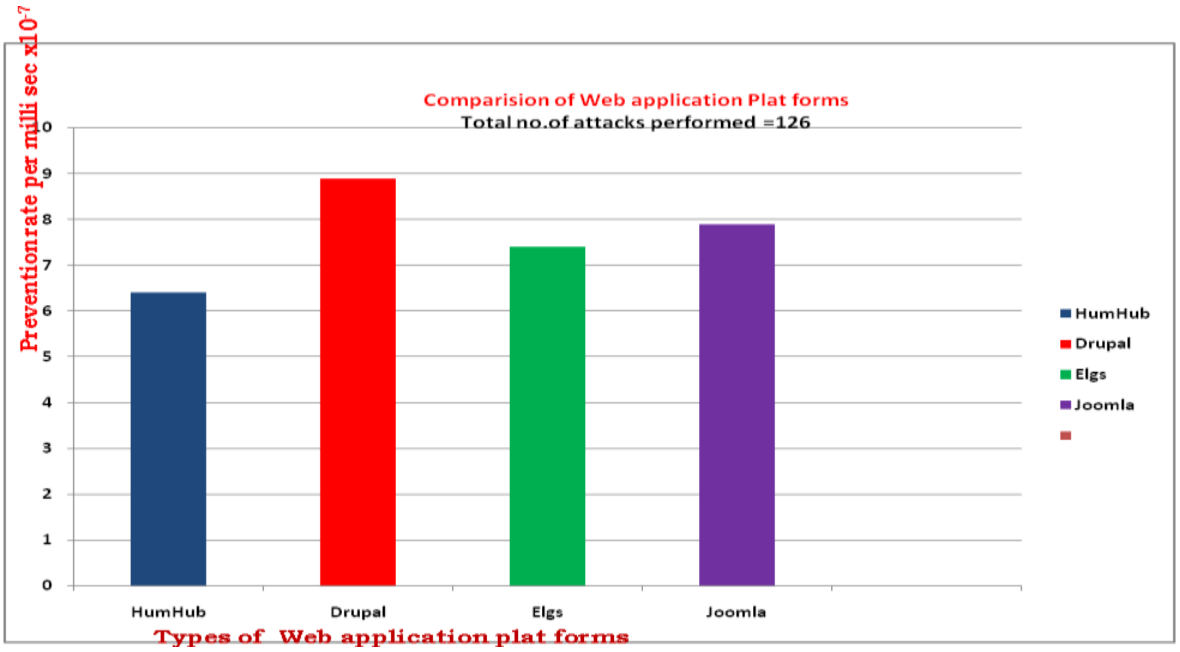


Figure 12. Plat forms vs Prevention time per milli sec

A graph has drawn between performance parameters vs their value for different detection, prevention methods. It shows my Hybrid method shows best performance values.

Detection methods	Accuracy	Precision	Recall	Epoch	f1 Score
Johari2014	0.92	0.933	0.932	0.933	0.938
BNPal2016	0.94	0.96	0.96	0.97	0.99
lstm2021	0.995	0.996	0.9975	0.998	0.9983
DEEPXSS2021	0.985	0.987	0.988	0.989	0.989
H_SQLI_XSS	0.997	0.998	0.998	0.998	0.999
xgbss	0.9941	0.9961	0.9962	0.9971	0.9974
wovsqli	0.99	0.9939	0.9941	0.9947	0.9951

Table 3 .Proposed method Performance Values

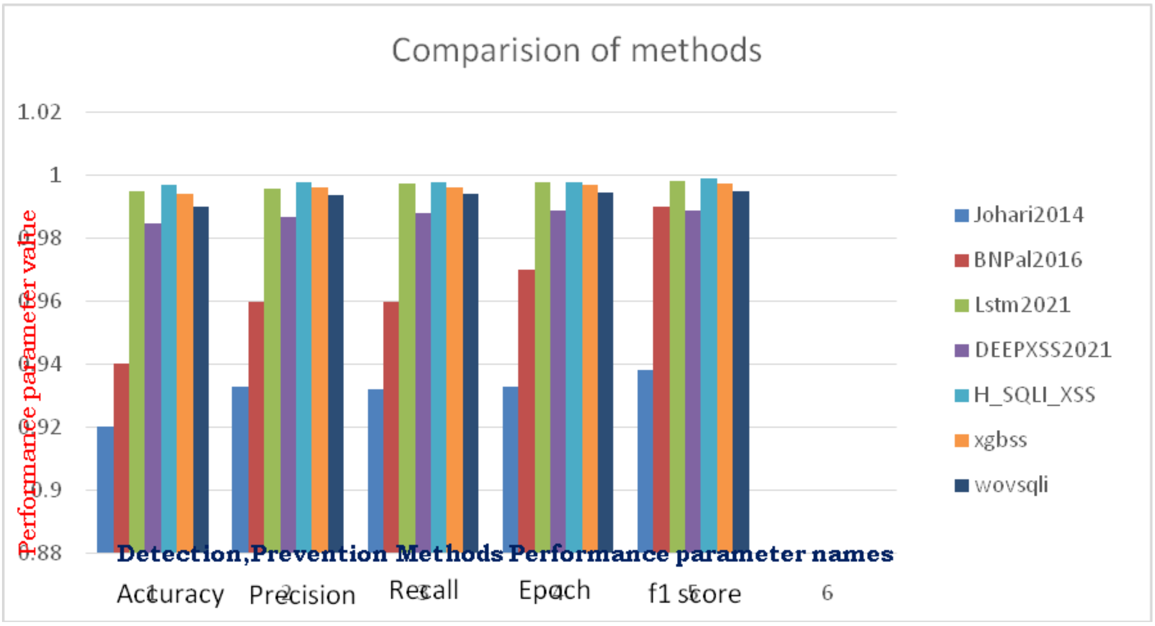


Figure 13.Performance parameters values vs Prevention methods

ROC GRAPH

It is a graph between true positive and false positive.My hybrid method shows best values than other methods.

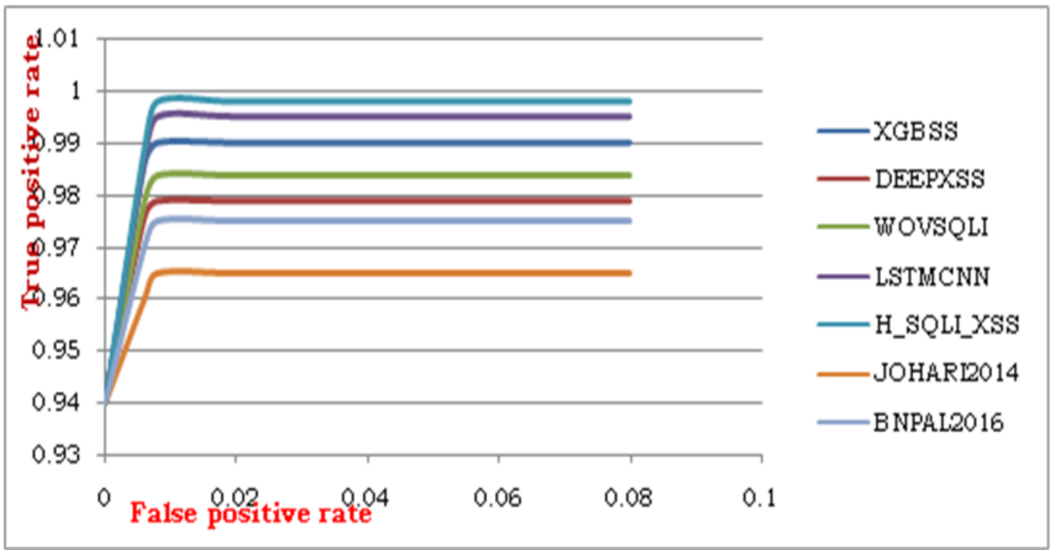


Figure 14.True positive vs False positive graph

Enhancement in this research work has increased the performance of Login System along with Security. The previously discussed model has enabled the security. But this enhancement provides Triple Layer Security along with better performance. The compression of content has improved the performance of Login System. In this work, Password is encrypted applying Multiplicative Inverse. As a result the password become small and secure before transmission in web application. When the data travels, there are less chances of error as small and encrypted data travels. To show the efficiency of proposed work, it is compared to existing work. For the comparative analysis, Security, Time Consumption, Data Size, Error Rate etc parameters are considered and Matlab simulation tool is used. The comparative analysis is clearly showing the applicability and efficiency of proposed work.

7. CONCLUSION

It has been concluded that proposed research has resolved the issue of sql injection by restrict wild character entry. Moreover LSTM model has restricted the reentry of unauthentic user that tried to login by some trick instead of using valid credential. Block chain has provided secure storage solution of user data. Multiplicative inverse is found unique mechanism to secure data in encrypted format. The integration of LSTM, block chain, multiplicative inverse with wild character restriction made the sql injection detection and prevention more efficient.

8.FUTURE SCOPE

This frame work with integrated SQLI,XSS detected and prevented SQLI,XSS attacks which are popular web attacks up to 99.989% of performance parameters. This can be extended by using transformers of deep learning. Also this frame work can be extended to prevent other attacks. We can include upgraded dataset and cognitive technologies with AI for increasing performance.

9.References

[1] *Joshi Padma, Dr.N.Ravishankar, Dr. M.B. Raju, N.CH.Ravi ,N.Ch.SaiVyuha "Building security barriers by modified algorithms in Blockchain to prevent sql injection and xss" Indian journal of Computer Science and Engineering,April,2022, DOI : 10.21817/indjcse/2022/v13i2/221302144.*

[2] *N.CH.Ravi, Joshi Padma.N etal "Smart security controls for strikers for dynamic computing models Indian journal of Computer Science and Engineering,April,2022" DOI : 10.21817/indjcse/2022/v13i2/221302125.*

[3] *Joshi Padma, Dr.N.Ravishankar, Dr. M.B. Raju, N.CH.Ravi "Smart Algorithms to Secure Web Based Applications from Sql Injection Attacks" Journal of Xidian University,Feb,2022,https://doi.org/10.37896/jxu16.2/026*

[4] *Joshi Padma, Dr.N.Ravishankar, Dr. M.B. Raju, N.CH.Ravi "Surgical Striking Sql injection attacks using LSTM"Indian journal of Computer Science and Engineering,Feb,2022, doi :10.21817/indjcse/2022/v13i1/221301182*

[5] 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

[6] 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.

[7] 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.

[8] 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)

[9] 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

[10] Joshi Padma, Dr.N.Ravishankar, Dr. M.B. Raju"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

[11] https://owasp.org

[12] Seth Fogie, Jeremiah Grossman" XSS Attacks: Cross Site Scripting Exploits and Defense " 26 June 2007.

[13] Justin Clarke-Sal ,:SQL Injection Attacks and Defense" 27 July 2012.

[4]Halfond WG, Viegas J, Orso A. "A classification of SQL-injection attacks and countermeasures," in Proceedings of the IEEE international symposium on secure software engineering. 2006;13-15

[14] Prithvi B, Madhusudan P, Venkatakrishnan VN (2010) CANDID: dynamic candidate evaluations for automatic prevention of SQL injection attacks. ACM Trans Inf Syst Secur 13(2):1–39.

[15] Kunal S, Mohan Das R, Pais AR (2011) Model based hybrid approach to prevent SQL injection attacks in PHP. InfoSecHiComNet'11 proceeding of the first international conference on security aspects of information technology. Springer, Berlin, pp 3–15.

[16] Indrani B., E. Ramaraj "An Efficient Technique for Detection & Prevention of SQL Injection Attack using ASCII Based String Matching" International Conference on Communication Technology & System Design 2012

[17]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

[18]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

[19] Yong Fang Jiaji peng,Liang LiuCheng Huang "WOVSQLI: Detection of SQL Injection

Behaviors Using Word Vector and LSTM" ICCSP 2018, March 16-18, 2018, Guiyang, China ACM ISBN 978-1-4503-6361-7/18/03

[20] Gupta BB etal.XSS-SAFE: a server-side approach to detect and mitigate cross-site scripting (XSS) attacks in javascript code (Gupta and Gupta 2015b

[21] Raed Waheed Kadhim,Methaq Talib Gaata "A hybrid of CNN and LSTM methods for securing web application against cross-site scripting attack" Indonesian Journal of Electrical Engineering and Computer Science Vol. 21, No. 2, February 2021, pp. 1022~1029.

[22] Abaimov S, Bianchi G. CODDLE: Code-Injection Detection With Deep Learning.IEEEAccess2019;7:pp.128617–27.

[23] Li C, Wang Y, Miao C, Huang C. Cross-site scripting guardian: A static XSS detector based on data stream input-output association mining. Appl Sci 2020;10. https:// doi.org/10.3390/app10144740.

[24] Fawaz Mahiuob Mohammed Mokbala,b,, Wang Dana, Wang Xiaoxic, Zhao Wenbina, Fu Lihuaa "XGBXSS: An Extreme Gradient Boosting Detection Framework for Cross-Site Scripting Attacks Based on Hybrid Feature Selection Approach and Parameters Optimization" Journal of Information Security and Applications ElsevierLtd.2021.*

[25] Bharti Nagpal Naresh Chauhan Nanhay Singh(2016) "security engine for CSRF, SQL injection & XSS attacks", Division of Operation & Maintenance, Lulea University of Technology, Sweden 2016

[26] Sharma P, Johari R, Sarma SS (2012) Integrated approach to prevent SQL injection attack and reflected cross site scripting attack. Int J Syst Assur Eng Manag 3(4):343–351

[27] Jun Yang1, Mengyu Zhou1 (B), and Baojiang Cui2 "MLAB-BiLSTM: Online Web Attack Detection Via Attention-Based Deep Neural Networks" SPDE 2020, CCIS 1268, pp. 482–492, 2020.

[28] B. Kranthikumar etal. B. Kranthikumar and R. Leela Velusamy "SQL injection detection using REGEX classifier" Journal of Xi'an University of Architecture & Technology Volume XII, Issue VI,pp.800-809 2020.